aws

パターン

AWS 規範ガイダンス

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 規範ガイダンス: パターン

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

AWS 規範ガイダンスパターン 1
クラウド基盤 3
の Landing Zone Accelerator を使用してアカウント作成を自動化する AWS 4
概要 4
前提条件と制限 4
アーキテクチャ 5
ツール 6
ベストプラクティス 9
エピック 9
関連リソース 35
追加情報 36
AWS リソースを自動的にインベントリする 38
概要 38
前提条件と制限 38
アーキテクチャ 40
ツール 40
ベストプラクティス 42
エピック 42
トラブルシューティング 49
関連リソース 50
アカウント間の VPC フローログを設定 51
概要 51
前提条件と制限 51
アーキテクチャ 52
ツール 52
ベストプラクティス 53
エピック 55
関連リソース 56
追加情報 57
Transit Gateway アタッチメントに自動的にタグを付ける 59
概要 59
前提条件と制限 59
アーキテクチャ 59
ツール 61

エピック 62
関連リソース 69
その他のパターン 70
AI と機械学習 71
Athena でのML 予測のため、DynamoDB のデータを集約 73
概要 73
前提条件と制限 73
アーキテクチャ 73
ツール 74
エピック 75
関連リソース 86
リポジトリをアカウント間で Amazon SageMaker AI Studio Classic AWS CodeCommit に関連付ける 87
概要 87
前提条件と制限 87
アーキテクチャ 88
ツール 88
エピック 89
追加情報 95
PDF ファイルからコンテンツを自動的に抽出する 98
概要 98
前提条件と制限 98
アーキテクチャ 99
ツール 100
エピック 100
関連リソース 106
添付ファイル 106
SageMaker AI DeepAR を使用してコールドスタート予測モデルを構築する 107
概要 107
前提条件と制限 107
アーキテクチャ 108
ツール 109
ベストプラクティス 109
エピック 110
関連リソース 112
SageMaker AI と Azure DevOps を使用して MLOps ワークフローを構築する 113

概要 113
前提条件と制限 113
アーキテクチャ 114
ツール 116
ベストプラクティス 117
エピック 118
トラブルシューティング 127
関連リソース 128
CloudFormation を使用して Amazon Bedrock でログ記録を設定する 130
概要 130
前提条件と制限 130
アーキテクチャ 131
ツール 132
エピック 133
関連リソース 135
SageMaker でStep Functions のモデルトレーニング用の Docker コンテナを作成する 137
概要 137
前提条件と制限 137
アーキテクチャ 138
ツール 138
エピック 139
関連リソース 150
Amazon Bedrock エージェントを使用して Amazon EKS でアクセスコントロールを作成する 151
概要 151
前提条件と制限 151
アーキテクチャ 152
ツール 153
ベストプラクティス 154
エピック 154
トラブルシューティング 173
関連リソース 173
に RAG ユースケースをデプロイする AWS 175
概要 175
前提条件と制限 175
アーキテクチャ 177

ツール 178
ベストプラクティス 179
エピック 180
関連リソース 183
追加情報 184
1 つの SageMaker エンドポイントに複数のパイプラインモデルオブジェクトをデプロイする 186
概要 186
前提条件と制限 186
アーキテクチャ 187
ツール 187
エピック 188
関連リソース 198
RAG および ReAct プロンプトを使用して AI チャットベースのアシスタントを開発する 199
概要 199
前提条件と制限 200
アーキテクチャ 201
ツール 202
ベストプラクティス 204
エピック 204
トラブルシューティング 212
関連リソース 212
追加情報 213
Amazon Bedrock を使用してチャットベースのアシスタントを開発する 214
概要 214
前提条件と制限 214
アーキテクチャ 215
ツール 216
ベストプラクティス 218
エピック 219
関連リソース 223
追加情報 223
音声入力から組織の知識を文書化する 226
概要 226
前提条件と制限 226
アーキテクチャ 227

ツール 228
ベストプラクティス 229
エピック 230
関連リソース 236
Amazon Personalize を使用してパーソナライズされたレコメンデーションを生成する 238
概要 238
前提条件と制限 238
アーキテクチャ 239
ツール 240
エピック 241
関連リソース 243
追加情報 244
SageMaker AI と Hybrid を使用して ML ワークフローを合理化する 248
概要 248
前提条件と制限 248
アーキテクチャ 249
ツール 250
ベストプラクティス 251
エピック 251
トラブルシューティング 258
関連リソース 258
追加情報 258
GPU がサポートするカスタム ML モデルのトレーニングとデプロイ 260
概要 260
前提条件と制限 260
アーキテクチャ 261
ツール 261
エピック 261
関連リソース 277
追加情報 277
..... 280
概要 280
前提条件と制限 281
アーキテクチャ 284
ツール 286
ベストプラクティス 287

エピック 287
関連リソース 294
追加情報 294
Amazon Q Developer をコーディングアシスタントとして使用する 299
概要 299
前提条件と制限 299
ツール 300
ベストプラクティス 300
エピック 301
トラブルシューティング 309
関連リソース 309
SageMaker Processing を使用して、テラバイト規模の ML データセットの分散型特徴量エンジニアリングを実現 310
概要 310
前提条件と制限 310
アーキテクチャ 311
ツール 314
エピック 314
関連リソース 327
添付ファイル 328
フラスコとElastic Beanstalk を使用して AI/ML モデルの結果を視覚化 329
概要 329
前提条件と制限 329
アーキテクチャ 330
ツール 332
エピック 332
関連リソース 342
追加情報 342
その他のパターン 347
分析 348
Microsoft SQL Server Analysis Services の Amazon Redshift データを分析 350
概要 350
前提条件と制限 350
アーキテクチャ 350
ツール 351
エピック 351

関連リソース 353
..... 354
概要 354
前提条件と制限 354
アーキテクチャ 354
ツール 355
エピック 356
関連リソース 361
から Amazon S3 AWS Data Exchange へのデータ取り込みを自動化する 362
概要 362
前提条件と制限 362
アーキテクチャ 362
ツール 363
エピック 364
関連リソース 366
添付ファイル 366
AWS Glue での暗号化適用を自動化する 367
概要 367
前提条件と制限 367
アーキテクチャ 367
ツール 368
ベストプラクティス 369
エピック 369
関連リソース 372
AWS DataOps Development Kit を使用して Google Analytics データを処理するためのデータパイプラインを構築する 373
概要 373
前提条件と制限 373
アーキテクチャ 373
ツール 374
エピック 375
トラブルシューティング 377
関連リソース 378
追加情報 378
ビデオ処理パイプラインを構築する 381
概要 381

前提条件と制限 381
アーキテクチャ 382
ツール 382
エピック 383
関連リソース 390
追加情報 391
添付ファイル 391
AWS Glue を使用して、Amazon S3 から Amazon Redshift までの ETL パイプラインを構築する 392
概要 392
前提条件と制限 392
アーキテクチャ 393
ツール 393
エピック 394
関連リソース 401
追加情報 402
Amazon DataZone を使用してエンタープライズデータメッシュを構築する 403
概要 403
前提条件と制限 403
アーキテクチャ 404
ツール 405
エピック 406
関連リソース 420
追加情報 421
AWS のサービスを使用してバリューアットリスク (VaR) を計算 423
概要 423
前提条件と制限 423
アーキテクチャ 424
ツール 425
ベストプラクティス 426
エピック 426
関連リソース 430
Athena を使用して共有 AWS Glue データカタログへのクロスアカウントアクセスを構成する 431
概要 431
前提条件と制限 431

アーキテクチャ 431
ツール 433
エピック 433
関連リソース 446
追加情報 446
NORMALIZEを Amazon Redshift SQL に変換 447
概要 447
前提条件と制限 447
アーキテクチャ 447
ツール 448
エピック 453
関連リソース 453
RESET WHEN を Amazon Redshift SQL に変換 455
概要 455
前提条件と制限 455
アーキテクチャ 455
ツール 456
エピック 459
関連リソース 460
AWS にサーバーレスデータレイクをデプロイして管理する 461
概要 461
前提条件と制限 461
アーキテクチャ 462
ツール 463
エピック 464
関連リソース 466
..... 468
概要 468
前提条件と制限 468
アーキテクチャ 469
ツール 469
エピック 470
関連リソース 473
添付ファイル 473
Amazon S3 へのAmazon EMR ロギングを確認 474
概要 474

前提条件と制限 474
アーキテクチャ 475
ツール 475
エピック 476
関連リソース 479
添付ファイル 479
AWS Glue を使用してテストデータを生成する 480
概要 480
前提条件と制限 480
アーキテクチャ 481
ツール 481
ベストプラクティス 481
エピック 482
関連リソース 492
追加情報 493
IoT データを直接 Amazon S3 に取り込む 497
概要 497
前提条件と制限 497
アーキテクチャ 498
ツール 499
ベストプラクティス 499
エピック 500
トラブルシューティング 508
関連リソース 509
追加情報 509
Amazon EMR で、Lambda 関数を使用して Spark ジョブを起動する 513
概要 513
前提条件と制限 513
アーキテクチャ 514
ツール 514
エピック 515
関連リソース 518
追加情報 519
添付ファイル 521
Apache Cassandra ワークロードを Amazon Keyspaces に移行する 522
概要 522

前提条件と制限 522
アーキテクチャ 522
ツール 523
ベストプラクティス 524
エピック 524
トラブルシューティング 537
関連リソース 537
追加情報 538
WANdisco LiveData Migrator を使用して Hadoop データを Amazon S3 に移行する 539
概要 539
前提条件と制限 539
アーキテクチャ 540
エピック 541
関連リソース 546
追加情報 546
Oracle ビジネスインテリジェンス 12C を AWS クラウドに移行 548
概要 548
前提条件と制限 548
アーキテクチャ 549
ツール 550
エピック 551
関連リソース 564
追加情報 565
MirrorMaker を使用して、オンプレミスの Apache Kafka クラスターを Amazon MSK に移行する 570
概要 570
前提条件と制限 570
アーキテクチャ 571
ツール 571
ベストプラクティス 572
エピック 572
関連リソース 576
追加情報 577
ELK スタックを AWS 上のElasticクラウドに移行する 578
概要 578
前提条件と制限 578

アーキテクチャ 580
ツール 582
エピック 583
関連リソース 590
追加情報 592
Starburst AWS を使用してデータを に移行する 593
概要 593
前提条件と制限 593
アーキテクチャ 593
ツール 595
エピック 596
関連リソース 599
入力ファイルサイズの ETL 取り込みを最適化する 600
概要 600
前提条件と制限 600
アーキテクチャ 600
ツール 601
エピック 601
関連リソース 604
追加情報 605
AWS Step Functions で ETL パイプラインをオーケストレーション 606
概要 606
前提条件と制限 606
アーキテクチャ 607
ツール 608
エピック 609
トラブルシューティング 617
関連リソース 617
追加情報 617
Amazon Redshift ML機械学習を使用して高度な分析を実行する 618
概要 618
前提条件と制限 618
アーキテクチャ 619
ツール 620
エピック 621
関連リソース 624

Amazon Athena を使用して Amazon DynamoDB をクエリする Amazon Athena 626
概要 626
前提条件と制限 626
アーキテクチャ 627
ツール 627
エピック 628
トラブルシューティング 631
関連リソース 632
Athena を使用して DynamoDB テーブルにクエリを実行する 633
概要 633
前提条件と制限 633
アーキテクチャ 634
ツール 634
エピック 635
関連リソース 643
追加情報 643
最小実行可能なデータスペースを設定する 645
概要 645
前提条件と制限 646
アーキテクチャ 647
ツール 648
ベストプラクティス 649
エピック 650
トラブルシューティング 704
関連リソース 704
追加情報 704
Amazon Redshift クエリー結果の言語固有のソートの設定 709
概要 709
前提条件と制限 709
アーキテクチャ 709
ツール 710
エピック 710
関連リソース 715
追加情報 715
クロスリージョン S3 バケットからのイベント通知に Lambda 関数をサブスクライブする 719
概要 719

前提条件と制限 719
アーキテクチャ 719
ツール 720
エピック 721
関連リソース 724
データを変換するための 3 つの AWS Glue ジョブタイプ 726
概要 726
前提条件と制限 726
アーキテクチャ 726
ツール 727
エピック 728
関連リソース 730
追加情報 731
添付ファイル 737
Athena と QuickSight を使用して Amazon Redshift 監査ログを視覚化する 738
概要 738
前提条件と制限 738
アーキテクチャ 738
ツール 739
エピック 739
関連リソース 744
添付ファイル 744
Amazon QuickSight を使用して IAM 認証情報レポートを視覚化する 745
概要 745
前提条件と制限 745
アーキテクチャ 746
ツール 747
エピック 747
追加情報 753
その他のパターン 755
コンピューティング 757
コンテナとマイクロサービス 758
Amazon EKS から Amazon Neptune にアクセスする 760
Amazon ECS のコンテナアプリケーションにアクセス 772
AWS Fargate 起動タイプで Amazon ECS のコンテナアプリケーションにアクセス 788
Amazon EKS でコンテナアプリケーションにプライベートにアクセスします 803

Amazon EKS での App Mesh で mTL をアクティベートする 811
AWS Batch を使用して Amazon RDS for PostgreSQL DBインスタンスのバックアップを自動化します 819
ノード終了ハンドラーのデプロイを自動化 832
Amazon EKS へ Java アプリケーションを自動的にビルドし、デプロイする 847
Amazon EFS を使用して EC2 インスタンスに Amazon ECS タスク定義を作成する 872
コンテナイメージを使用して Lambda 関数をデプロイする 878
AWS Fargate を使用して Amazon ECS に Java マイクロサービスをデプロイする 887
Amazon EKS と Helm を使用して Kubernetes パッケージをデプロイする 893
Amazon EKS にサンプル Java マイクロサービスをデプロイし、Application Load Balancer で公開する 905
AWS Copilot を使用してクラスター化されたアプリケーションを Amazon ECS にデプロイする 918
gRPC ベースのアプリケーションを Amazon EKS にデプロイする 929
Amazon EKS クラスターをデプロイおよびデバッグ 942
Elastic Beanstalk を使用してコンテナをデプロイする 970
Lambda と Amazon VPC を使用して静的アウトバウンド IP アドレスを生成する 976
重複するコンテナイメージを自動的に識別する 991
Amazon EKS ワーカーノードに SSM エージェントのインストール 1022
prebootstrapコマンドを使用して、SSM エージェントと CloudWatch エージェントを Amazon EKS ワーカーノードにインストールします。 1028
EKS Auto Mode 用の NGINX Ingress Controller の移行 1035
コンテナワークロードを ARO から ROSA に移行する 1056
生成した Docker イメージを最適化する 1073
Kubernetes ポッドを Amazon EKS の互換性のあるノードに配置します。 1083
フィルタリングされた Amazon ECR コンテナイメージをアカウントまたはリージョン間で複製する 1100
コンテナを再起動せずに認証情報をローテーションする 1117
Amazon ECS タスクをAmazon WorkSpaces で実行する 1123
ASP.NET ウェブ API Docker コンテナを AWS で実行する 1133
AWS Fargate を使用してメッセージ駆動型のワークロードを実行する 1146
永続データストレージでステートフルワークロードを実行する 1155
Amazon EKS Pod Identity と KEDA を使用して自動スケーリングを設定する 1180
..... 1203
ALB で Amazon ECS で相互 TLS を使用する 1226
その他のパターン 1257

サーバーレス 1259
AWS Amplify を使用して React Native アプリを構築する 1260
複数のSaaS製品間でテナントを一元管理 1279
クロスアカウント Amazon EventBridge 接続を作成する 1291
Kinesis Data Streams と Firehose を使用して DynamoDB レコードを Amazon S3 に配信する 1306
API Gateway を使用してパスベースの API バージョニングを実装する 1313
psycopg2 ライブラリを にインポートする AWS Lambda 1325
API Gateway を Amazon SQS と統合する 1335
AWS Lambda で APIs非同期的に処理する 1353
Amazon DynamoDB Streams を使用して APIs非同期的に処理する 1362
Amazon SQS で APIs非同期的に処理する 1372
Step Functions から Systems Manager Automation タスクを同期的に実行する 1383
を使用して S3 オブジェクトの並列読み取りを実行する AWS Lambda 1397
Lambda から OpenSearch にテレメトリデータを送信する 1409
サーバーレスセルルーターのセットアップ 1424
Amazon S3 バケットへのプライベートアクセスを設定する 1443
Step Functions の状態のトラブルシューティング 1450
サーバーレスアプローチを使用して AWS サービスを連結する 1460
その他のパターン 1466
ネットワーク 1468
AWS Transit Gateway のピアリングを自動化する 1469
AWS Transit Gateway を使用してネットワーク接続を一元化 1476
Application Load Balancer を使用して Oracle JD Edwards EnterpriseOne の HTTPS 暗号化を設定します 1483
プライベートネットワーク経由でアプリケーション移行サービスのデータプレーンをコントロールプレーンに接続 1495
AWS CloudFormation カスタムリソースを使用して Infoblox オブジェクトを作成します。 1509
Network Firewall 用に CloudWatch アラートをカスタマイズ 1522
Terraform を使用して Wavelength Zone にリソースをデプロイする 1541
DNS レコードを Amazon Route 53 プライベートホストゾーンに一括で移行する 1549
F5 から AWS のApplication Load Balancer に移行するときの HTTP ヘッダーを変更 1559
複数の VPC から中央のAWS のサービスエンドポイントにプライベートにアクセスする . 1565
Network Access Analyzer の検出結果を複数の でレポートする AWS アカウント 1576
マルチアカウント AWS 環境でハイブリッドネットワークの DNS 解決を設定する 1607

.... 1621
Splunk を使用して AWS Network Firewall のログとメトリクスを表示する 1627
その他のパターン 1638
コンテンツ配信 1640
Firehose を使用して Splunk に AWS WAF ログを送信する 1641
CloudFront を使用して VPC 経由で S3 バケット内の静的コンテンツを提供する 1649
その他のパターン 1658
データベースとストレージ 1659
データベース 1660
リンクサーバーを使用してオンプレミス SQL Server データにアクセスする 1663
AWS で Oracle PeopleSoft に HA を追加 1669
SQL Server データベースを AWS 上の MongoDB Atlas に移行する際のクエリパフォーマンスを評価する 1693
IaC を使用して Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する 1703
AWS Lambda および Task Scheduler を使用してデータベースタスクを自動化する 1736
DR Orchestrator Framework を使用してフェイルオーバーとフェイルバックを自動化する 1748
全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント 1776
SAP HANA データベースの自動的にバックアップ 1791
DynamoDB の PynamoDB モデルと CRUD 関数を自動的に生成する DynamoDB 1799
Amazon RDS へのパブリックアクセスをブロック 1807
Amazon DynamoDB へのクロスアカウントアクセスを設定する 1814
Always On アベイラビリティグループで読み取り専用ルーティングを構成する 1828
pgAdmin の SSH トンネルを使用して接続 1837
JSON Oracleクエリを PostgreSQL データベース SQL に変換 1842
Amazon DynamoDB テーブルをアカウント間でコピー 1872
Amazon DynamoDB テーブルをアカウントからコピー 1879
Amazon RDS と Amazon Aurora のコストと使用状況のレポートを作成する 1891
Aurora PostgreSQL を使用して、Oracle RAC ワークロードをエミュレート 1898
Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする .. 1904
既存の Amazon RDS for PostgreSQL DB インスタンスを暗号化する 1913
起動時に Amazon RDS データベースの自動タグ付けを強制する 1921
DynamoDB のコストを見積る 1927
Amazon DynamoDB テーブルのストレージコストを推定 1938
AWR レポートを使用して Oracle データベースの Amazon RDS エンジンサイズを推定 ... 1942
Amazon RDS for SQL Server テーブルを S3 バケットにエクスポートする 1978
動的 SQL ステートメントの匿名ブロックを処理 1990

Aurora PostgreSQL 互換にオーバーロードされた Oracle関数を処理 1997
DynamoDB でのタグ付けの強制を支援 2004
クロスリージョンの DR を実装します 2010
100 個以上の引数を持つ Oracle 関数を PostgreSQL に移行 2027
Amazon RDS for Oracle DB インスタンスを AMS に移行する 2033
Oracle OUT バインド変数を PostgreSQL に移行 2043
HSR を使用して SAP HANA を AWS に移行する 2051
分散可用性グループを使用して SQL Server を AWS に移行する 2064
SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for Oracle に移行 2076
Amazon Aurora の暗号化をモニタリングする 2085
Amazon CloudWatch を使用して GoldenGate のログをモニタリング 2090
Oracle データベース EE を Amazon RDS for Oracle SE2 にリプラットフォームする 2105
Precisely Connect を使用してメインフレームデータベースを AWS にレプリケート 2115
Amazon RDS と Aurora PostgreSQL のジョブをスケジュールする 2135
オンプレミスの SMTP サーバーを使用して RDS for SQL Server の通知を送信 2141
AWS の IBM Db2 に SAP の DR をセットアップ 2155
Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する 2176
Amazon RDS Customで Oracle E-Business Suite の HA/DR アーキテクチャを設定する .. 2185
RDS for MySQL と Amazon EC2 上の MySQL との間のデータレプリケーションをセットアップする 2194
Oracle PeopleSoft アプリケーションの移行ロール 2200
アカウント間で Amazon Redshift からデータをアンロードする 2236
ワークロード別のデータベース移行パターン 2265
その他のパターン 2276
ストレージとバックアップ 2282
EC2 インスタンスを AMS の S3 バケットへの書き込みアクセスを許可 2283
Snowflake データベースへのデータストリームの取り込みを自動化します。 2289
EBS ボリュームを自動的に暗号化する 2300
AWS Charon-SSPエミュレーターでSun SPARCサーバーをバックアップします 2312
Veeam を使用して Amazon S3 にデータをバックアップおよびアーカイブする 2337
VMware Cloud on AWS 向け NetBackup の構成 2363
AWS CLI を使用してアカウントとリージョン間で S3 オブジェクトをコピーする 2370
S3 バッチレプリケーションを使用してアカウントとリージョン間で S3 オブジェクトをコピーする 2385
DistCp と AWS PrivateLink for Amazon S3 を使用して Hadoop データを移行します。 2399

その他のパターン 2415
デベロッパーツール 2417
DevOps 2418
AWS インフラストラクチャオペレーションを自動化する 2422
ロードバランサーエンドポイントの変更に関する CloudFront 更新を自動化する 2432
Python アプリケーションの CodeGuru AWS CDK レビューを自動化する 2441
AWS リソース評価を自動化する 2453
SAP システムのインストールを自動化 2467
AWS CDK を使用して Service Catalog ポートフォリオと製品のデプロイを自動化する ... 2481
AWS CodeCommit から Amazon S3 へのバックアップを自動化する 2500
CloudFormation スタックの削除を自動化する 2508
動的パイプライン管理を自動化してホットフィックスソリューションをデプロイする 2519
Amazon MWAA カスタムメトリクスの取り込みを自動化する 2548
AWS CDK と CloudFormation を使用して Amazon Lex リソースの管理を自動化する 2565
AWS CodePipeline および AWS CodeBuild を使用して、スタックセットデプロイを自動化 2584
Systems Manager のマネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする 2614
マイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する。 2630
マイクロサービスで疎結合アーキテクチャを構築する 2643
Docker イメージを構築して Amazon ECR にプッシュする 2655
AWS サービスで iOS アプリを構築してテストする 2661
AWS CDK アプリケーションまたは CloudFormation テンプレートのベストプラクティスをルールパックを使用して確認する 2667
Amazon EKS 上のアプリケーションに相互 TLS を設定する 2672
AWS CloudFormation を使用して AppStream 2.0 リソースを作成する 2684
Firelens を使用して Amazon ECS 用のカスタムログパーサーを作成する 2690
CodePipeline と HashiCorp Packer を使用してパイプラインと AMI を作成 2699
CodePipeline を使用してパイプラインを作成し、更新をオンプレミスの EC2 インスタンスにデプロイ 2706
Java および Python プロジェクト用の動的 CI パイプラインの作成 2715
CloudWatch Synthetics Canary をデプロイする 2731
Amazon ECS に Java マイクロサービス用の CI/CD パイプラインをデプロイ 2739
ChatOps ソリューションをデプロイして SAST スキャン結果を管理する 2748

AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイする 2766
..... 2780
EC2 インスタンスプロファイルを使用して AWS Cloud9 から Amazon EKS クラスターをデプロイする 2784
複数の AWS リージョンにコードをデプロイする 2796
Amazon Redshift SQL クエリを実行する 2809
AWS Backup レポートを CSV ファイルとしてエクスポートする 2822
Amazon EC2 インスタンスタグを CSV ファイルにエクスポートする 2832
AWS Config マネージドルールを含む AWS CloudFormation テンプレートを生成する 2839
SageMaker ノートブックインスタンスに CodeCommit リポジトリへのクロスアカウントアクセスを許可する 2846
GitHub フロー分岐戦略を実装する 2858
Gitflow 分岐戦略を実装する 2867
トランク分岐戦略を実装する 2879
一元化されたカスタム Checkov スキャンを実装する 2885
モノリポジトリ内の変更を検出した後、別の CI/CD パイプラインを開始する 2896
Bitbucket リポジトリを AWS Amplify と統合する 2913
Lambda を使用して AWS アカウント間で CodeBuild プロジェクトを起動する 2922
Application Recovery Controller を使用して EMR クラスターのマルチ AZ フェイルオーバーを管理する 2936
複数のアカウントやリージョンへのマイクロサービスのブルー/グリーンデプロイを管理 . 2953
ワイルドカードアクセス権限のため、Amazon ECR リポジトリをモニタリングします。 . 2987
マルチアカウントサーバーレスデプロイを最適化する 2992
AWS CodeCommit イベントからカスタムアクションを実行する 3013
GitHub Actions を使用して AWS Service Catalog 製品をプロビジョニングする 3018
最小特権 IAM ロールのプロビジョニング 3029
Amazon CloudWatch メトリクスを CSV ファイルに公開 3042
AWS アカウント から 全体の Amazon EC2 エントリを削除する AWS Managed Microsoft AD 3048
同じ の Amazon EC2 エントリを AWS アカウント から削除する AWS Managed Microsoft AD 3066
で Python ETL ジョブのユニットテストを実行する AWS Glue 3077
Amazon S3 で Helm v3 チャートを設定する 3088
CodePipeline を使用して CI/CD パイプラインをセットアップする 3097
Amazon EKS のアプリケーションのエンドツーエンド暗号化を設定 3113

Amazon EKS マルチテナントアプリケーションのデプロイを簡素化する 3127
複数の E メールエンドポイントを SNS トピックにサブスクライブする 3146
AWS Fargate WaitCondition フック構造を使用する 3151
AWS CodePipeline でサードパーティーの Git リポジトリを使用する 3164
AWS CodePipeline を使用して Terraform の構成を検証する 3174
その他のパターン 3190
インフラストラクチャ 3194
Session Manager と Amazon EC2 Instance Connect により踏み台ホストにアクセス 3196
を使用して DNS 解決を一元化する AWS Managed Microsoft AD 3212
オブザーバビリティアクセスマネージャーを使用してモニタリングを一元化 3224
起動時に EC2 インスタンスに必須タグが欠けていないか確認する 3238
Session Manager を使用して EC2 インスタンスに接続 3244
AWS CodePipeline に適用されない AWS リージョンにパイプラインを作成 3251
AWS CDK アスペクトとエスケープハッチを使用してデフォルトのロール名をカスタマイズする 3259
Amazon EC2 にプライベート静的 IP を使用して Cassandra クラスターをデプロイしてリバランスを回避する 3268
Transit Gateway Connect を使用して VRF を AWS に拡張する 3275
AWS KMS キーの状態が変更されたときの Amazon SNS 通知を受け取る 3292
非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約 3299
コードリポジトリから Service Catalog で Terraform 製品をプロビジョニングする 3305
単一メールアドレスで複数の AWS アカウントを登録する 3326
単一アカウントの AWS 環境でハイブリッドネットワークの DNS 解決を設定 3343
Amazon EC2 に UiPath RPA ボットを自動的にセットアップする 3348
AWS で可用性の高い PeopleSoft アーキテクチャを設定する 3365
Oracle JD Edwards EnterpriseOne のディザスタリカバリをセットアップする 3394
ドリフト検出とレポートの設定 3422
S3 バケットを CloudFormation スタックとして正常にインポートしました 3427
異なるリージョンで Amazon EFS ファイルシステムを同期する 3439
LocalStack および Terraform Tests を使用して AWS インフラストラクチャをテストする 3448
SAP ペースメーカークラスターを ENSA1 から ENSA2 にアップグレード 3456
異なるアカウントの VPC では一貫したアベイラビリティーゾーンを使用する 3478
IAM ポリシーでユーザー IDs 3484
Account Factory for Terraform コードをローカルで検証する 3495
その他のパターン 3515

ウェブおよびモバイルアプリ 3519
React Web アプリケーションのユーザーを認証する 3520
Amplify ウェブアプリケーションを継続的にデプロイ 3530
AWS Amplify とAmazon Cognito を使用して React アプリケーションを作成 3538
マイクロフロントエンド用のポータルを作成する 3554
React ベースの SPA を Amazon S3 と CloudFront にデプロイする 3586
プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API をデプロイする 3594
Amazon QuickSight ダッシュボードをローカルの Angular アプリケーションに埋め込みます。 3601
Green Boost を使用したウェブアプリ開発について探索する 3622
AWS CodeBuild を使用してユニットテストを実行する 3650
Python プロジェクトを六角形アーキテクチャで構築する 3659
その他のパターン 3686
IoT 3688
IoT 環境のセキュリティイベントのロギングとモニタリングを設定 3689
概要 3689
前提条件と制限 3689
アーキテクチャ 3690
ツール 3692
エピック 3693
関連リソース 3697
AWS IoT SiteWise メタデータ属性の抽出とクエリ 3698
概要 3698
前提条件と制限 3698
アーキテクチャ 3699
ツール 3699
エピック 3700
関連リソース 3703
追加情報 3703
..... 3706
概要 3706
前提条件と制限 3707
アーキテクチャ 3707
ツール 3708
ベストプラクティス 3708

エピック 3709
トラブルシューティング 3724
関連リソース 3726
追加情報 3726
その他のパターン 3728
移行とモダナイゼーション 3729
移行 3730
AWS DMS 用の AWS CloudFormation テンプレートの作成 3731
自動ポートフォリオ検出の開始方法 3735
オンプレミスの Cloudera ワークロードを AWS に移行する 3742
Microsoft SQL サーバーを AWS に移行した後に接続エラーを解決する 3757
RHEL ソースサーバーを再起動した後、SELinux を無効にせずに AWS レプリケーションエージェントを自動的に再起動する 3760
リアーキテクト 3767
リホスト 4168
リロケート 4432
リプラットフォーム 4525
ワークロード別の移行パターン 5146
その他のパターン 5155
モダナイゼーション 5157
CAST イメージングでのソフトウェアアーキテクチャの分析と視覚化 5158
CAST Highlight を使用して、AWS に移行する前に準備状況を評価 5167
Amazon S3 に期限切れの DynamoDB データを自動的にアーカイブする 5191
Amazon OpenSearch Service におけるマルチテナントサーバーレスアーキテクチャの構築 5208
マルチスタックのアプリケーションをデプロイ 5257
AWS SAM を使用してネストされたアプリケーションをデプロイ 5266
AWS Lambda TVM を使用して Amazon S3 の SaaS テナント分離を実装する 5275
AWS Step Functions を使用して、サーバーレス Saga パターンを実装する 5299
Amazon ECS Anywhere で、オンプレミスコンテナアプリケーションを管理します。 5313
ASP.NET ウェブフォームアプリケーションを AWS で最新化 5323
AWS Fargate を使用して、イベント駆動型の大規模なワークロードを実行する 5339
SaaS アーキテクチャにおけるテナントオンボーディング 5349
CQRS とイベントソーシングを使用する 5374
その他のパターン 5399
メインフレーム 5402

をインストールして IBM z/OS AWS のサービス からアクセスする AWS CLI 5404
メインフレームデータを Amazon S3 にバックアップおよびアーカイブ 5425
AWS Mainframe Modernization と を使用して COBOL Db2 プログラムを構築する AWS CodeBuild 5452
Micro Focus Enterprise Server を構築する 5476
AWS クラウドでメインフレームファイルビューアを構築 5496
最新の Blu Age アプリケーションをコンテナ化 5512
EBCDIC データを AWS 上の ASCII に変換する 5522
AWS Lambda を使用してメインフレーム EBCDIC ファイルを ASCII ファイルに変換する 5541
複雑なレコードレイアウトのメインフレームデータファイルを変換 5561
コンテナ化されたアプリの環境をデプロイする 5580
Db2 z/OS データインサイトの生成 5590
QuickSight で AWS Mainframe Modernization と Amazon Q を使用してインサイトを生成する 5642
Stonebranch ユニバーサルコントローラーを AWS と統合 5661
Precisely を使用して VSAM ファイルを AWS クラウドに移行および複製します 5696
でのメインフレーム出力管理のモダナイズ AWS 5714
Amazon Q を使用して CardDemo メインフレームアプリケーションをモダナイズする 5762
でメインフレームのバッチ印刷ワークロードをモダナイズする AWS 5778
Rocket Software Enterprise Suite を使用してメインフレーム環境をモダナイズする 5808
メインフレームのオンライン印刷ワークロードを AWS で最新化 5826
転送ファミリーを使用して、メインフレームファイルを Amazon S3 に移動する 5860
Db2 フェデレーションデータベース内のユーザーアクセスを保護する 5873
Db2 z/OS データを AWS に転送する 5882
その他のパターン 5909
管理 5910
コスト管理 5911
AWS Glue ジョブの詳細なコストと使用状況レポートを作成 5912
Amazon EMR クラスターの詳細なコストと使用状況レポートを作成する 5918
その他のパターン 5923
高性能コンピューティング 5924
Terraform と DRA を使用して Lustre ファイルシステムをデプロイする 5925
AWS ParallelCluster 用の Grafana モニタリングダッシュボードを設定する 5934
NICE DCV を使用して自動スケーリング VDI をセットアップする 5948
ハイブリッドクラウド 5961

VMware Cloud on AWS へのデータセンターの拡張を構成する 5962
VMware Cloud on AWS に VM をプロビジョニングするように vRealize Automation を設定 5967
VMware Cloud on AWS を使用して SDDC をデプロイする 5978
VMware vRealize Network Insight を VMware Cloud on AWS と統合する 5988
HCX OSAM を使用して、 VMware Cloud on AWS に VM を移行 5993
VMware Cloud on から Splunk AWS にログを送信する 5999
AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する 6006
その他のパターン 6027
管理とガバナンス 6028
Amazon Data Firehose リソースが暗号化されていない場合のアラート 6029
Windows レジストリエントリの追加または更新を自動化する 6034
Python を使用して RFC を自動的に作成する 6039
自動的に Amazon RDS DB インスタンスを停止して開始する 6047
Terraform を使用して AWS Organizations でソフトウェアパッケージの配布を一元化する 6061
.NET アプリケーションのロギングを CloudWatch Logs に設定する 6074
AWS Service Catalog 製品を AWS アカウントとリージョンにコピー 6084
クラウド運用の RACI マトリックスを作成 6093
CloudWatch により、カスタムメトリクスのアラームを作成 6099
デフォルトで暗号化された EBS ボリュームで AWS Cloud9 IDE を作成 6106
タグベースの CloudWatch ダッシュボードを自動的に作成する 6112
ランディングゾーンの設計を文書化する 6121
AWS CDK を使って組織全体で Amazon DevOps Guru を有効にしましょう 6124
ブートストラップパイプラインを使用して AFT を実装する 6151
複数の AWS アカウントとリージョンで AWS Service Catalog 製品を管理 6176
AWS アカウントを AWS Organizations から AWS Control Tower に移行する 6184
SAP RHEL Pacemaker クラスターのモニタリング 6198
全体で AMI の使用をモニタリングする AWS アカウント 6219
AWS Organizations のプログラムによるアカウント閉鎖のアラートを設定する 6236
AWS アカウントまたは組織の EBS スナップショットの詳細を表示 6246
その他のパターン 6253
メッセージとコミュニケーション 6255
Amazon MQ で RabbitMQ 設定を自動化する 6256
Amazon Connect のエージェントワークステーションの通話品質を向上 6264

その他のパターン 6279
セキュリティ、アイデンティティ、コンプライアンス 6280
Amazon Cognito を使用して ASP.NET から AWS のサービスにアクセスする 6283
概要 6283
前提条件と制限 6283
アーキテクチャ 6284
ツール 6284
エピック 6285
トラブルシューティング 6289
関連リソース 6289
添付ファイル 6290
AWS Directory Service を使用して Amazon EC2 の Microsoft SQL Server を認証する 6291
概要 6291
前提条件と制限 6291
アーキテクチャ 6291
ツール 6292
エピック 6292
関連リソース 6296
インシデント対応とフォレンジックを自動化する 6297
概要 6297
前提条件と制限 6297
アーキテクチャ 6298
ツール 6300
エピック 6301
関連リソース 6305
追加情報 6305
添付ファイル 6306
Security Hub 標準調査結果の修正を自動化 6307
概要 6307
前提条件と制限 6308
アーキテクチャ 6308
ツール 6309
ベストプラクティス 6309
エピック 6310
関連リソース 6312
添付ファイル 6313

Amazon Inspector を使用してクロスアカウントワークロードのセキュリティスキャンを自動化 6314
概要 6314
前提条件と制限 6314
アーキテクチャ 6315
ツール 6316
エピック 6317
関連リソース 6321
添付ファイル 6321
パブリック IP アドレスからのアクセスを自動的に監査する 6322
概要 6322
前提条件と制限 6322
アーキテクチャ 6323
ツール 6325
ベストプラクティス 6326
エピック 6326
トラブルシューティング 6330
関連リソース 6331
セキュリティのベストプラクティスを使用して AWS CloudTrail を自動的に再有効化する 6332
概要 6332
前提条件と制限 6332
アーキテクチャ 6333
ツール 6333
エピック 6334
関連リソース 6339
添付ファイル 6340
暗号化されていない Amazon RDS DB インスタンスとクラスターを自動的に修正する 6341
概要 6341
前提条件と制限 6341
アーキテクチャ 6343
ツール 6343
ベストプラクティス 6344
エピック 6345
関連リソース 6351
追加情報 6351
IAM ユーザーアクセスキーの自動ローテーション 6353

概要 6353
前提条件と制限 6354
アーキテクチャ 6355
ツール 6357
エピック 6358
関連リソース 6368
AWS アカウントの IAM ポリシーとロールを自動的に検証してデプロイする 6369
概要 6369
前提条件と制限 6370
アーキテクチャ 6371
ツール 6371
エピック 6372
関連リソース 6376
Security Hub と Jira を双方向に統合する 6377
概要 6377
前提条件と制限 6378
アーキテクチャ 6378
ツール 6380
エピック 6381
関連リソース 6391
追加情報 6391
強化されたコンテナイメージのパイプラインを構築する 6393
概要 6393
前提条件と制限 6393
アーキテクチャ 6394
ツール 6397
エピック 6398
トラブルシューティング 6405
関連リソース 6405
Terraform を使用して AWS Organizations の IAM アクセスキー管理を一元化する 6407
概要 6407
前提条件と制限 6408
アーキテクチャ 6408
ツール 6410
ベストプラクティス 6411
エピック 6411

トラブルシューティング 6420
関連リソース 6421
Amazon CloudFront ディストリビューションのアクセスロギング、HTTPS、TLS バージョンを確認する 6422
概要 6422
前提条件と制限 6422
アーキテクチャ 6423
ツール 6424
エピック 6424
関連リソース 6427
添付ファイル 6427
IPv4 および IPv6 対応セキュリティグループのイングレスルール内の単一ホストネットワークエントリを確認する 6428
概要 6428
前提条件と制限 6428
アーキテクチャ 6428
ツール 6429
エピック 6430
関連リソース 6433
添付ファイル 6433
Amazon Cognito 認証フローを選択する 6434
概要 6434
前提条件と制限 6434
アーキテクチャ 6435
ツール 6439
エピック 6440
関連リソース 6443
追加情報 6444
Guard を使用して AWS Config カスタムルールを作成する 6445
概要 6445
前提条件と制限 6446
アーキテクチャ 6446
ツール 6450
エピック 6451
トラブルシューティング 6453
関連リソース 6454

複数の から Prowler の検出結果のレポートを作成する AWS アカウント 6455
概要 6455
前提条件と制限 6456
アーキテクチャ 6457
ツール 6457
エピック 6459
トラブルシューティング 6482
関連リソース 6482
追加情報 6483
を使用して未使用の Amazon EBS ボリュームを削除する AWS Config 6485
概要 6485
前提条件と制限 6485
アーキテクチャ 6486
ツール 6486
エピック 6487
トラブルシューティング 6489
関連リソース 6490
を使用して AWS Control Tower コントロールをデプロイする AWS CDK 6491
概要 6491
前提条件と制限 6492
アーキテクチャ 6493
ツール 6494
ベストプラクティス 6495
エピック 6495
関連リソース 6502
追加情報 6503
Terraform を使用して AWS Control Tower コントロールをデプロイする 6506
概要 6506
前提条件と制限 6507
アーキテクチャ 6508
ツール 6508
ベストプラクティス 6509
エピック 6509
トラブルシューティング 6515
関連リソース 6517
追加情報 6517

コード内のセキュリティ問題を検出するパイプラインをデプロイする 6519
概要 6519
前提条件と制限 6519
アーキテクチャ 6519
ツール 6521
エピック 6521
トラブルシューティング 6524
関連リソース 6525
追加情報 6525
パブリックサブネットの検出コントロールをデプロイする 6527
概要 6527
前提条件と制限 6528
アーキテクチャ 6528
ツール 6529
ベストプラクティス 6530
エピック 6530
関連リソース 6538
追加情報 6539
パブリックサブネットの予防的コントロールをデプロイする 6542
概要 6542
前提条件と制限 6542
アーキテクチャ 6543
ツール 6544
エピック 6545
関連リソース 6551
追加情報 6551
Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする 6554
概要 6554
前提条件と制限 6554
アーキテクチャ 6555
ツール 6555
ベストプラクティス 6556
エピック 6556
トラブルシューティング 6559
関連リソース 6560

期限切れの CA 証明書を使用して Amazon RDS インスタンスを検出する 6561
概要 6561
前提条件と制限 6561
アーキテクチャ 6562
ツール 6563
ベストプラクティス 6564
エピック 6565
トラブルシューティング 6569
関連リソース 6570
IAM アクセスアナライザーで IAM ポリシーを動的に生成 6571
概要 6571
前提条件と制限 6571
アーキテクチャ 6572
ツール 6573
エピック 6575
関連リソース 6581
CloudFormation テンプレートを使用して GuardDuty を有効にする 6582
概要 6582
前提条件と制限 6582
アーキテクチャ 6582
ツール 6583
エピック 6584
関連リソース 6585
追加情報 6586
Amazon RDS for SQL Server で透過的なデータ暗号化を有効にする 6590
概要 6590
前提条件と制限 6590
アーキテクチャ 6590
ツール 6591
エピック 6591
関連リソース 6594
AWS ロードバランサーが安全なリスナープロトコルを使用していることを確認する 6595
概要 6595
前提条件と制限 6595
アーキテクチャ 6596
ツール 6596

ベストプラクティス 6597
エピック 6597
トラブルシューティング 6600
関連リソース 6601
添付ファイル 6601
Amazon EMR 保管中のデータの暗号化を確保 6602
概要 6602
前提条件と制限 6602
アーキテクチャ 6603
ツール 6603
エピック 6604
関連リソース 6607
添付ファイル 6607
IAM プロファイルが EC2 インスタンスと確実に関連付けられているようにします。 6608
概要 6608
前提条件と制限 6608
アーキテクチャ 6609
ツール 6609
エピック 6610
関連リソース 6613
添付ファイル 6613
新しい Amazon Redshift クラスターが暗号化されていることを確保 6614
概要 6614
前提条件と制限 6614
アーキテクチャ 6614
ツール 6615
エピック 6616
関連リソース 6618
添付ファイル 6618
IAM Identity Center のアイデンティティとその割り当てのレポートをエクスポートする 6619
概要 6619
前提条件と制限 6619
アーキテクチャ 6621
ツール 6621
エピック 6622
トラブルシューティング 6623

関連リソース 6625
追加情報 6625
予定されている KMS キーの削除を防ぐのに役立ちます。 6628
概要 6628
前提条件と制限 6628
アーキテクチャ 6629
ツール 6630
エピック 6631
関連リソース 6634
追加情報 6635
添付ファイル 6635
でパブリック Amazon S3 バケットを識別する AWS Organizations 6636
概要 6636
前提条件と制限 6636
アーキテクチャ 6637
ツール 6638
エピック 6639
トラブルシューティング 6643
関連リソース 6644
追加情報 6644
Microsoft Sentinel に AWS セキュリティログを取り込む 6645
概要 6645
前提条件と制限 6645
アーキテクチャ 6647
ツール 6648
ベストプラクティス 6649
エピック 6649
関連リソース 6659
CodePipeline で、IAM アイデンティティセンターのアクセス権限セットを管理 6661
概要 6661
前提条件と制限 6661
アーキテクチャ 6663
ツール 6664
ベストプラクティス 6665
エピック 6665
トラブルシューティング 6675

関連リソース 6675
AWS Secrets Manager で認証情報を管理 6676
概要 6676
前提条件と制限 6676
アーキテクチャ 6676
ツール 6677
エピック 6677
関連リソース 6679
追加情報 6679
..... 6682
概要 6682
前提条件と制限 6682
アーキテクチャ 6683
ツール 6683
エピック 6684
関連リソース 6687
添付ファイル 6687
Amazon EMR クラスターの起動時に転送中の暗号化をモニタリングする 6688
概要 6688
前提条件と制限 6689
アーキテクチャ 6689
ツール 6689
エピック 6691
関連リソース 6693
添付ファイル 6693
Amazon ElastiCache クラスターの保存時の暗号化をモニタリングする 6694
概要 6694
前提条件と制限 6695
アーキテクチャ 6695
ツール 6696
エピック 6697
関連リソース 6699
添付ファイル 6699
EC2 インスタンスキーペアを監視する 6700
概要 6700
前提条件と制限 6700

アーキテクチャ 6700
ツール 6701
エピック 6702
関連リソース 6705
添付ファイル 6706
IAM ルートユーザーのアクティビティを監視する 6707
概要 6707
前提条件と制限 6707
アーキテクチャ 6708
ツール 6708
エピック 6710
関連リソース 6715
追加情報 6715
IAM ユーザーが作成されたときに通知する 6716
概要 6716
前提条件と制限 6716
アーキテクチャ 6716
ツール 6717
エピック 6718
関連リソース 6720
添付ファイル 6720
SCP を使用してインターネットアクセスを防止する 6721
概要 6721
前提条件と制限 6721
ツール 6722
ベストプラクティス 6722
エピック 6722
関連リソース 6725
IP アドレスまたは位置情報に基づいてアクセスを制限する 6726
概要 6726
前提条件と制限 6726
ツール 6727
エピック 6728
関連リソース 6734
機密情報の Git リポジトリをスキャンする 6735
概要 6735

前提条件と制限 6735
アーキテクチャ 6735
ツール 6735
ベストプラクティス 6736
エピック 6736
関連リソース 6741
AWS Network Firewall から Slack チャネルにアラートを送信 6742
概要 6742
前提条件と制限 6742
アーキテクチャ 6743
ツール 6744
エピック 6745
関連リソース 6751
追加情報 6751
AWS Private CA と AWS RAM を使用してプライベート証明書の管理を簡素化する 6756
概要 6756
前提条件と制限 6756
アーキテクチャ 6757
ツール 6758
エピック 6759
関連リソース 6765
追加情報 6766
すべての Security Hub メンバーアカウントでセキュリティ標準コントロールをオフにする .. 6767
概要 6767
前提条件と制限 6767
アーキテクチャ 6768
ツール 6769
エピック 6770
関連リソース 6774
PowerShell を使用して IAM アイデンティティセンターから AWS CLI 認証情報を更新 6775
概要 6775
前提条件と制限 6775
アーキテクチャ 6776
ツール 6776
ベストプラクティス 6777
エピック 6777

トラブルシューティング 6779
関連リソース 6780
追加情報 6780
AWS Config を使用して Amazon Redshift をモニタリング 6783
概要 6783
前提条件と制限 6783
アーキテクチャ 6784
ツール 6784
エピック 6785
関連リソース 6788
追加情報 6789
Network Firewall を使用して、アウトバウンドネットワークトラフィックから DNS ドメイン名をキャプチャします 6790
概要 6790
前提条件と制限 6790
アーキテクチャ 6791
ツール 6791
エピック 6792
Terraform を使用して GuardDuty を自動的に有効化する 6807
概要 6807
前提条件と制限 6808
アーキテクチャ 6810
ツール 6811
エピック 6812
関連リソース 6821
追加情報 6822
PCI DSS 4.0 のベストプラクティスを検証する 6823
概要 6823
前提条件と制限 6823
ツール 6825
エピック 6825
関連リソース 6828
追加情報 6828
添付ファイル 6829
..... 6830
概要 6830

前提条件と制限 6830
アーキテクチャ 6831
ツール 6831
エピック 6832
関連リソース 6835
添付ファイル 6835
..... 6836
概要 6836
前提条件と制限 6836
アーキテクチャ 6837
ツール 6837
エピック 6838
関連リソース 6841
添付ファイル 6841
その他のパターン 6842
..... 6845

AWS 規範ガイダンスパターン

Amazon Web Services (AWS) の規範的ガイダンスパターンでは、特定のクラウド移行、モダナイゼーション、デプロイシナリオを実装するための手順説明、アーキテクチャ、ツール、およびコードが記載されています。これらのパターンは、対象分野の専門家によって精査され AWS、移行を計画している、または移行中のビルダーや実践的なユーザーを対象としています AWS。また、既に稼働していて AWS 、クラウド運用を最適化またはモダナイズする方法を探しているユーザーもサポートしています。

これらのパターンを使用して、複雑なオンプレミスまたはクラウドのワークロードを に移行 AWS し、プロジェクトの概念実証、計画、実装段階にあるかどうかにかかわらず、クラウドの導入、最適化、モダナイゼーションの取り組みを加速できます。たとえば、クラウド移行プロジェクトの場合：

- 計画段階では、 AWSへの移行に利用可能なさまざまなオプションを評価することができます。移転、リホスト、リプラットフォーム、再構築のいずれを行うかに応じて、ニーズに合った適切なパターンを選択できます。また、移行に使用できるさまざまなツールを理解し、ライセンスの調達を計画したり、ベンダーとの最初の話し合いを始めたりすることもできます。
- 概念実証と実装の段階では、パターンに記載されているステップバイステップの指示に従って、ワークロードを AWSに移行することができます。各パターンには、前提条件、ターゲットリファレンスアーキテクチャ、ツール、ステップバイステップタスク、ベストプラクティス、トラブルシューティング、コードなどの詳細が含まれています。
- を既に使用している場合は AWS クラウド、クラウドリソースの使用をモダナイズ、最適化、スケーリング、保護するのに役立つパターンを見つけることができます。

技術分野別のパターンのリストを表示するには、以下のリンクを使用するか、「AWS 規範的ガイダンスのホームページ」のフィルタリングと検索のオプションを使用してください。

- クラウド基盤
- AI と機械学習
- 分析
- コンピューティング
- データベースとストレージ
- デベロッパーツール
- IoT
- 移行とモダナイゼーション

- 管理
- セキュリティ、アイデンティティ、コンプライアンス

ガイド、戦略、パターンを含むすべての出版物を閲覧するには、「AWS 規範的ガイダンスのホームページ」をご覧ください。

クラウド基盤

トピック

- で Landing Zone Accelerator を使用してアカウントの作成を自動化する AWS
- 複数のアカウントとリージョンの AWS リソースを自動的にインベントリする
- AWS アカウント全体にわたり一元化するための VPC フローログを設定
- AWS Organizations を使用して Transit Gateway アタッチメントに自動的にタグを付ける
- その他のパターン

で Landing Zone Accelerator を使用してアカウントの作成を自動化する AWS

作成者: Justin Kuskowski (AWS)、Joe Behrens (AWS)、Nathan Scott (AWS)

概要

このパターンでは、Landing Zone Accelerator on AWS ソリューションを使用して、承認されたユーザーがリクエストを送信 AWS アカウント したときに新しい を自動的にデプロイする方法について説明します。を使用して、多数の AWS Lambda 関数をオーケストレーション AWS Step Functions します。Lambda 関数は、アカウント情報を Git リポジトリに追加し、 AWS CodePipeline パイプラインを開始し、必要な AWS リソースがプロビジョニングされていることを確認します。プロセスが完了すると、ユーザーはアカウントが作成されたという通知を受け取ります。

必要に応じて、Microsoft Entra ID グループを統合し、アカウント作成プロセス中に AWS IAM Identity Center アクセス許可セットを割り当てることができます。組織が ID ソースとして Microsoft Entra ID を使用している場合、このオプション機能は、新しいアカウントへのアクセスを自動的に管理および設定するのに役立ちます。

前提条件と制限

前提条件

- の管理アカウントへのアクセス AWS Organizations
- AWS Cloud Development Kit (AWS CDK) バージョン 2.118.0 以降、インストールおよび設定済み
- Python バージョン 3.9 以降、インストール済み
- AWS Command Line Interface （ AWS CLI) バージョン 2.13.19 以降、インストール済み
- Docker バージョン 24.0.6 以降、インストール済み
- 管理アカウントにデプロイされた Landing Zone Accelerator on AWS ソリューション
- （ オプション) Microsoft Entra ID と IAM アイデンティティセンター、統合

制約事項

アカウント作成ワークフローは、単一の をデプロイするためのシーケンシャル実行をサポートします AWS アカウント。この制限により、並列実行中にリソースを競合することなく、アカウント作成ワークフローが正常に完了します。

アーキテクチャ

ターゲット アーキテクチャ

次の図は、Landing Zone Accelerator on AWS. AWS Step Functions orchestrates AWS アカウントを使用して新しい の作成を自動化する高レベルのアーキテクチャを示しています。Step Functions ワークフローの各タスクは、1 つ以上の AWS Lambda 関数によって実行されます。

この図表は、次のワークフローを示しています：

1. ユーザーは、Python スクリプトを実行するか、Amazon API Gateway を使用してアカウントをリクエストします。
2. アカウント作成オーケストレーターのワークフローは から始まります AWS Step Functions。
3. ワークフローは、ソースコードリポジトリの `account-config.yaml` ファイルを更新します。また、 AWS パイプラインで Landing Zone Accelerator を開始し、パイプラインのステータスをチェックします。このパイプラインは、新しいアカウントを作成してセットアップします。この仕組みの詳細については、「 ランディングゾーンアクセラレーターのアーキテクチャの概要」を参照してください AWS。
4. （ オプション) パイプラインが完了すると、ワークフローは Microsoft Entra ID にグループが存在するかどうかを確認します。グループが Microsoft Entra ID に存在しない場合、ワークフローは Microsoft Entra ID にグループを追加します。
5. ワークフローは、Landing Zone Accelerator on AWS ソリューションでは実行できない追加のステップを実行します。デフォルトのステップは次のとおりです。
   - AWS Identity and Access Management (IAM) でのアカウントエイリアスの作成
   - のアカウントにタグをアタッチする AWS Organizations
   - アカウントに割り当てられたタグに基づいて AWS Systems Manager Parameter Store でパラメータを作成する
6. （ オプション) ワークフローは、以前に指定した Microsoft Entra ID グループに 1 つ以上のアクセス許可セットを割り当てます。アクセス許可セットにより、グループのユーザーは新しいアカウントにアクセスし、設定したアクションを実行できます。
7. AWS Lambda 関数は QA テストと検証テストを実行します。リソースの作成を検証し、タグが作成されたことを確認し、セキュリティリソースがデプロイされたことを確認します。
8. ワークフローはアカウントを解放し、Amazon Simple Email Service (Amazon SES) を使用して、プロセスが正常に完了したことをユーザーに通知します。

Step Functions ワークフローの詳細については、このパターンの???「追加情報」セクションの「Step Functions ワークフロー図」を参照してください。

Microsoft Entra ID アプリケーション

Microsoft Entra ID と統合することを選択した場合は、このパターンをデプロイするときに次の 2 つのアプリケーションを作成します。

- IAM Identity Center にリンクされ、Microsoft Entra ID グループが IAM Identity Center で使用可能であることを確認するアプリケーション。この例では、この Microsoft Entra ID アプリケーションの名前は ですLZA2。
- Lambda 関数が Microsoft Entra ID と通信し、Microsoft Graph APIs を呼び出すことを許可するアプリケーション。このパターンでは、このアプリケーションの名前は ですcreate_aws_account。

これらのアプリケーションは、Microsoft Entra ID グループを同期し、アクセス許可セットを割り当てるために使用されるデータを収集します。

ツール

AWS のサービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。このパターンでは、API Gateway を使用して、 AWS アカウント 名前の可用性をチェックし、 AWS Step Functions ワークフローを開始し、Step Functions 実行のステータスを確認します。
- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境をセットアップして管理するのに役立ちます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、他の のイベントバスなどです AWS アカウント。このソリューションでは、Step Functions ワークフローの状態が Failed、、Timed-outまたは に変わった場合に Lambda 関数を開始する EventBridge ルールを使用しますAborted。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。

- AWS IAM Identity Center は、すべての AWS アカウント およびクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理するのに役立ちます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。このパターンでは、Amazon Simple Storage Service (Amazon S3) に保存されているデータ、Lambda 環境変数、Step Functions のデータなどのデータを暗号化するために AWS KMS キーが使用されます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- Amazon Simple Email Service (Amazon SES) は、独自の E メールアドレスとドメインを使用して E メールを送受信するのに役立ちます。新しいアカウントが正常に作成されると、Amazon SES を通じて通知が送信されます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。アカウント作成プロセス中にエラーが発生した場合、Amazon SNS は設定した E メールアドレスに通知を送信します。
- AWS Step Functions はサーバーレスオーケストレーションサービスであり、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立ちます。
- AWS Systems Manager Parameter Store は、設定データ管理とシークレット管理のための安全な階層型ストレージを提供します。

その他のツール

- awscurl は AWS API リクエストの署名プロセスを自動化し、標準の curl コマンドとしてリクエストを行うのに役立ちます。
- 以前は Azure Active Directory と呼ばれていた Microsoft Entra ID は、クラウドベースの ID およびアクセス管理サービスです。
- Microsoft Graph APIs、Microsoft Entra や Microsoft 365 などの Microsoft クラウドサービスのデータやインテリジェンスへのアクセスに役立ちます。

コードリポジトリ

このパターンのコードは、GitHub lza-account-creation-workflow リポジトリで入手できます。

lambda_layer ディレクトリには、複数の Lambda 関数で参照される次のレイヤーが含まれています。

- account_creation_helper – このレイヤーには、ロールを引き受け、進行状況をチェックするためのモジュールが含まれています AWS Service Catalog。
- boto3 – このレイヤーには、 AWS Lambda が最新バージョンであることを確認するための AWS SDK for Python (Boto3)モジュールが含まれています。
- identity_center_helper – このレイヤーは IAM アイデンティティセンターへの呼び出しをサポートします。

lambda_src ディレクトリには、次の Lambda 関数が含まれています。

- AccountTagToSsmParameter – この関数は、 パラメータストアでパラメータを作成するために AWS Organizations 、 アカウントにアタッチされたタグを使用します。各パラメータは/account/tags/プレフィックスで始まります。
- AttachPermissionSet – この関数は、IAM Identity Center グループに権限セットを追加します。
- AzureADGroupSync – この関数は、ターゲット Microsoft Entra ID グループを IAM アイデンティティセンターに同期します。
- CheckForRunningProcesses – この関数は、AWSAccelerator-Pipelineパイプラインが現在実行されているかどうかを確認します。パイプラインが実行されている場合、関数は AWS Step Functions ワークフローを遅延させます。
- CreateAccount – この関数は AWS Service Catalog と AWS Control Tower を使用して新しい を作成します AWS アカウント。
- CreateAdditionalResources – この関数は、Landing Zone Accelerator によって管理されていない AWS リソース AWS CloudFormation、またはアカウントのエイリアスや AWS Service Catalog タグなどを作成します。
- GetAccountStatus – この関数は、 でプロビジョニングされた製品をスキャンし AWS Service Catalog て、アカウント作成プロセスが完了したかどうかを確認します。
- GetExecutionStatus – この関数は、実行中または完了した AWS Step Functions 実行のステータスを取得します。
- NameAvailability – この関数は、 AWS アカウント 名前がすでに存在するかどうかを確認します AWS Organizations。

- ReturnResponse – アカウントの作成が成功すると、この関数は新しいアカウントの ID を返します。アカウントの作成が成功しなかった場合は、エラーメッセージが返されます。
- RunStepFunction – この関数は、アカウントを作成する AWS Step Functions ワークフローを実行します。
- SendEmailWithSES – この関数は、アカウントの作成が終了するのを待っているユーザーに E メールを送信します。
- ValidateADGroupSyncToSSO – この関数は、指定された Microsoft Entra ID グループが IAM アイデンティティセンターと同期されているかどうかを確認します。
- ValidateResources – この関数は、すべての AWS Control Tower カスタマイズが正常に実行されたことを検証します。

ベストプラクティス

には、次の命名規則をお勧めします AWS CDK。

- p プレフィックスを使用してすべてのパラメータを開始します。
- c プレフィックスを使用してすべての条件を開始します。
- r プレフィックスを使用してすべてのリソースを起動します。
- o プレフィックスを使用してすべての出力を開始します。

エピック

検証とタグ付けのために IAM ロールをデプロイする

タスク	説明	必要なスキル
Landing Zone Accelerator をでカスタマイズ AWS 用に準備します。	1. AWS コードリポジトリの Landing Zone Accelerator で、 という名前のファイルを作成しますcustomizations-config.yaml 。このファイルを使用して、コアソリューションのカスタマイズを定義します。詳細については、「ソリュー	AWS DevOps

タスク	説明	必要なスキル
	ションのカスタマイズ」を参照してください。 2. customizations-config.yaml ファイルで、というセクションを作成しますcloudFormationStacks 。	

タスク	説明	必要なスキル
lza-account-creation-validation ロールをデプロイする準備をします。	次に、ソリューションをカスタマイズして、管理アカウント以外のすべてのアカウントに lza-account-creation-validation IAM ロールをデプロイします。このロールは、Lambda ValidateResources 関数に新しいアカウントへの読み取り専用アクセスを提供します。 1. GitHub から account-creation-validation-role.yaml ファイルをダウンロードします。 2. ファイルの テンプレートセクションに示されている場所にcustomizations-config.yaml ファイルを保存します。 3. customizations-config.yaml ファイルを開きます。 4. cloudFormationStacks セクションで、次のコードを追加します。ランディングゾーンの AWS リージョン 必要に応じてターゲットを更新します。 `- deploymentTargets: organizationalUnits:`	AWS DevOps

タスク	説明	必要なスキル

	<pre>- Root
	excludedA
	ccounts:
	-
	Management
	
	descripti
	on: IAM Role to allow
	Account Validation
	name: lza-
	account-creation-
	validation
	regions:
	- us-
	east-1
	template:
	cloudformation/acc
	ount-creation-vali
	dation-role.yaml
	runOrder: 1
	terminati
	onProtection: true
	parameters:
	- name:
	pManagementAccount
	Id
	value:
	"{{ account Managemen
	t }}"</pre> 5. customizations-con fig.yaml ファイルを保存して閉じます。	

タスク	説明	必要なスキル
account-tagging-to-ssm-parameter-role ロールをデプロイする準備をします。	次に、ソリューションをカスタマイズして、管理アカウント以外のすべてのアカウントに account-tagging-to-ssm-parameter-role IAM ロールをデプロイします。このロールは、Parameter Store AWS Systems Manager でパラメータを作成するために使用されます。 1. GitHub から account-tagging-to-ssm-parameter-role.yaml ファイルをダウンロードします。 2. ファイルの テンプレートセクションに示されている場所にcustomizations-config.yaml ファイルを保存します。 3. customizations-config.yaml ファイルを開きます。 4. cloudFormationStacks セクションで、次のコードを追加します。ランディングゾーンの AWS リージョン 必要に応じてターゲットを更新します。 `- deploymentTargets: organizationalUnits: - Root`	AWS DevOps

タスク	説明	必要なスキル

	<pre>excludedAccounts:
	- Management
	description: IAM Role to create SSM Parameters based on Account Tagging
	name: lza-account-tagging-to-ssm-parameter
	regions:
	- us-east-1
	template: cloudformation/account-tagging-to-ssm-parameter-role.yaml
	runOrder: 1
	terminationProtection: true
	
	parameters:
	- name: pManagementAccountId
	value: "{{ account Management }}"</pre>	

	5. customizations-config.yaml ファイルを保存して閉じます。	

タスク	説明	必要なスキル
config-log-validation-role ロールをデプロイする準備をします。	次に、ソリューションをカスタマイズして、IAM config-log-validation-role ロールをログアーカイブアカウントにデプロイします。このロールは、ValidateResources Lambda 関数が Amazon S3 バケットにアクセスして、ログ記録とアクセス AWS Config ルールを実行できるようにします。 1. GitHub から config-log-validation-role.yaml ファイルをダウンロードします。 2. ファイルの テンプレートセクションに示されている場所にcustomizations-config.yaml ファイルを保存します。 3. customizations-config.yaml ファイルを開きます。 4. cloudFormationStacks セクションで、次のコードを追加します。ランディングゾーンの AWS リージョン 必要に応じてターゲットを更新します。 `- deploymentTargets: accounts: - LogArchive`	AWS DevOps

タスク	説明	必要なスキル

	<pre>descripti
	on: IAM Role to
	 validate Config and
	 Logs
	name:
	 lza-config-log-val
	idation-role
	regions:
	- us-
	east-1
	template:
	 cloudformation/con
	fig-log-validation-
	role.yaml
	runOrder: 1
	terminati
	onProtection: true
	
	parameters:
	- name:
	 pManagementAccount
	Id
	value:
	 "{{ account Managemen
	t }}"</pre> 5. customizations-con fig.yaml ファイルに加えられた変更を保存、閉じ、コミットします。	

(オプション) Microsoft Entra ID からデータを取得する

タスク	説明	必要なスキル
Lambda 関数が Microsoft Entra ID と通信できるようにするアプリケーションを作成します。	1. Microsoft Entra ID 管理センターで、create_aw s_account アプリケーションを登録します。手順	Microsoft Entra ID

タスク	説明	必要なスキル

	については、Microsoft ドキュメントの「アプリケーションの登録」を参照してください。
	2. Microsoft ドキュメントの「アプリのリクエストされたアクセス許可を更新する」の手順に従って、`create_aws_account` アプリケーションの次の Microsoft Graph アクセス許可を設定します。
	• Application.Read.All
	  • タイプ: Application
	  • 管理者の同意が必要: はい
	• AppRoleAssignment.ReadWrite.All
	  • タイプ: Application
	  • 管理者の同意が必要: はい
	• Group.ReadWrite.All
	  • タイプ: Application
	  • 管理者の同意が必要: はい
	• ServicePrincipalEndpoint.Read.All
	  • タイプ: Application
	  • 管理者の同意が必要: はい
	• Synchronization.ReadWrite.All	

タスク	説明	必要なスキル
	• タイプ: Application • 管理者の同意が必要: はい • User.Read • タイプ: Delegated • 管理者の同意が必要: いいえ	

タスク	説明	必要なスキル
`create_aws_account` アプリケーションの値を取得します。	次に、`create_aws_account` アプリケーションに必要な値を取得します。 1. Microsoft Entra ID 管理センターで、アプリ登録に移動し、 を選択します`create_aws_account`。 2. 左側のペインで、概要を選択します。 3. 概要ページで、次の値を書き留めます。 • アプリケーション (クライアント) ID • ディレクトリ (テナント) ID 4. 左側のペインの「管理」で、「証明書とシークレット」を選択します。 5. 証明書とシークレットページで、クライアントシークレットタブを選択し、次の値を書き留めます。 • クライアントシークレット値 • クライアントシークレット ID	Microsoft Entra ID

タスク	説明	必要なスキル
Microsoft Entra ID を IAM アイデンティティセンターと統合するアプリケーションを作成します。	Microsoft Entra ID 管理センターで、LZA2アプリケーションを登録します。手順については、Microsoft ドキュメントの「アプリケーションの登録」を参照してください。	Microsoft Entra ID
LZA2 アプリケーションの値を取得します。	次に、LZA2アプリケーションに必要な値を取得します。 1. Microsoft Entra ID 管理センターで、エンタープライズアプリケーションに移動し、 を選択しますLZA2。 2. 左側のペインで、概要を選択します。 3. 概要ページで、次の値を書き留めます。 • 名前 • オブジェクト ID 4. 左側のペインの 管理 で マニフェスト を選択します。 5. JSON ファイル の appRolesセクションで、という名前のアプリケーションロールを見つけますUser。 6. このアプリロールidの値を書き留めます。	Microsoft Entra ID

タスク	説明	必要なスキル

シークレットを作成します。	1. で AWS CLI、次のコマンドを入力して変数を作成します。`create_aws_account` および LZA2アプリケーション用に取得した値を使用します。
	`# Variables for create_aws_account app`
	`TENANT_ID='<Directory ID>'`
	`CLIENT_ID='<Application ID>'`
	`SECRET_ID='<Client secret ID>'`
	`SECRET_VALUE='<Client secret value>'`
	
	`# Variables for LZA2 app`
	`OBJECT_ID='<Object ID>'`
	`APP_ROLE_ID='<App role ID>'`
	`ENTERPRISE_APP_NAME='<Name>'`
	2. 次のコマンドを入力して、`GraphApiSecret` という名前のシークレットを作成します AWS Secrets Manager。
	`aws secretsmanager create-secret \`
	`--name GraphApiSecret \`
		AWS DevOps

タスク	説明	必要なスキル

	<pre>--secret-string
	"{\"client_id\":
	\"${CLIENT_ID}\",
	\"tenant_id\":
	\"${TENANT_ID}\",
	\"object_id\":
	\"${OBJECT_ID}\",
	\"app_role_id\":
	\"${APP_ROLE_ID}\"
	, \"secret_value\":
	\"${SECRET_VALUE}\
	", \"secret_id\":
	\"${SECRET_ID}\"}"</pre> 今後シークレットを更新する必要がある場合は、変数を更新して次のコマンドを実行できます。 <pre>aws secretsmanager
	update-secret \
	--secret-id
	GraphApiSecret \
	--secret-string
	"{\"client_id\":
	\"${CLIENT_ID}\",
	\"tenant_id\":
	\"${TENANT_ID}\",
	\"object_id\":
	\"${OBJECT_ID}\",
	\"app_role_id\":
	\"${APP_ROLE_ID}\"
	, \"secret_value\":
	\"${SECRET_VALUE}\
	", \"secret_id\":
	\"${SECRET_ID}\"}"</pre>	

解決策をデプロイする

タスク	説明	必要なスキル

ソースコードを複製します。	1. 次のコマンドを入力して、lza-account-creation-workflow リポジトリのクローンを作成します。
	`git clone https://github.com/aws-samples/lza-account-creation-workflow`
	2. 次のコマンドを入力して、クローンされたリポジトリの config ディレクトリに移動します。
	`cd lza-account-creation-workflow/config`
		DevOps エンジニア

`deploy-config.yaml` ファイルを更新します。	1. `deploy-config.yaml` ファイルを開きます。 2. テンプレートを確認し、AWS 環境へのデプロイに必要な値を更新します。たとえば、次の値を更新します。 • `accountCreationFailure` • `accountCompletionFromEmail` • `ssoLoginUrl` • `rootEmailPrefix` • `rootEmailDomain`	AWS DevOps

タスク	説明	必要なスキル
	3. Microsoft Entra ID と統合する場合は、次の操作を行います。 • `enableAzureADIntegration` を `true` に設定します。 • `graphApiSecretName` 値には、前に作成したシークレット () を入力しますGraphApiSecret 。 4. `deploy-config.yaml` ファイルを保存して閉じます。	

タスク	説明	必要なスキル

ソリューションを AWS 環境にデプロイします。	1. 次のコマンドを入力します。
	`cdk bootstrap <account-number>/<Region>`
	詳細については、 AWS CDK ドキュメントの「ブートストラップ」を参照してください。
	2. CloudFormation テンプレートを合成するには、次のコマンドを入力します。
	`cdk synth`
	詳細については、 AWS CDK ドキュメントのAWS CDK 「スタック合成の設定と実行」を参照してください。
	3. 次のコマンドを入力してソリューションをデプロイします。
	`cdk deploy`
	詳細については、 AWS CDK ドキュメントのAWS CDK 「アプリケーションのデプロイ」を参照してください。
		AWS DevOps

タスク	説明	必要なスキル
	Note このソリューションでは、Amazon S3 バケットを使用して、このソリューションのソースコードを保存します。upload_to_source_bucket.py スクリプトを使用して、ソースコードのアーカイブを作成し、更新されたバージョンをアップロードできます。	

オプション 1 – Python を使用してアカウントを作成する

タスク	説明	必要なスキル
使用する引数を特定します。	Step Functions ワークフローを開始する Python スクリプトを実行するときに使用する引数を選択します。引数の完全なリストについては、このパターンの???「追加情報」セクションを参照してください。	AWS DevOps、Python
Python スクリプトを起動します。	1. 次のコマンドを入力して、クローンされたリポジトリに移動します。 `cd lza-account-creation-workflow`	DevOps エンジニア、Python

タスク	説明	必要なスキル
	2. Step Functions ワークフローを開始する Python スクリプトを実行します。以下は、引数と値のサンプルを含むコマンドの例です。 `python ./run-stepfunction.py \ --account-name "lza-test-01" \ --support-dl "johnsmith@example.com" \ --managed-org-unit "Workloads/Workload-1" \ --purpose "Testing new micro service" \ --force-update true \ --ad-integration "{\"CustomerAccountAdmin\": \"platform-admin\", \"CustomerAccountDev\": \"workload1-app1\"}" \ --bypass-creation true \ --tags APPLICATION=TestingMicroService` 3. アカウントが正常に作成されたという通知を受信するまで待ちます。	

タスク	説明	必要なスキル
	Note アカウント作成プロセスが失敗すると、Amazon SNS はdeploy-config.yaml 、 ファイルaccountCreationFailure で で定義した E メールアドレスに通知を送信します。アカウントリクエスタには通知されません。	

オプション 2 – API Gateway と awscurl を使用してアカウントを作成する

タスク	説明	必要なスキル
awscurl の変数を設定します。	1. 次のコマンドを入力して、ソースコードディレクトリに移動します。 `cd lza-account-creation-workflow` 2. 次のコマンドを入力して、AWS アクセスキー変数を設定します。AWS アクセスポータルから変数をコピーし、シェルに貼り付け	AWS DevOps

タスク	説明	必要なスキル

	ることができます。以下に例を示します。
	
	`export AWS_ACCESS_KEY_ID="<id>"`
	`export AWS_SECRET_ACCESS_KEY="<key>"`
	`export AWS_SESSION_TOKEN="<token>"`
	
	3. 次のコマンドを入力して、API コール AWS リージョン の を設定します。
	
	`export AWS_REGION=$(aws configure get region)`
	
	4. 次のコマンドを入力して、`lza-account-creation-workflow-application` CloudFormation 出力から API Gateway エンドポイントを取得します。
	
	`export API_GATEWAY_ENDPOINT=$(aws cloudformation describe-stacks --stack-name "lza-account-creation-workflow-application" --query 'Stacks[*].Outputs[?OutputKey==`oApiGatewayCreateAccountEndpoin`	

タスク	説明	必要なスキル
	`t`].OutputValue' --output text)`	
名前の可用性を確認します。	次のコマンドを入力して、の名前が使用可能であることを確認します AWS アカウント。をターゲットアカウントの名前<AWS_ACCOUNT_NAME>　に置き換えます。 `awscurl --service execute-api \ --region ${AWS_REGION} \ --access_key ${AWS_ACCESS_KEY_ID} \ --secret_key ${AWS_SECRET_ACCESS_KEY} \ --security_token ${AWS_SESSION_TOKEN} \ -X POST ${API_GATEWAY_ENDPOINT}check_name?account_name=<AWS_ACCOUNT_NAME>`	AWS DevOps

タスク	説明	必要なスキル
アカウント作成ワークフローを実行します。	1. クローンされたリポジトリのルートフォルダで、`api_example.json` ファイルを開きます。 2. 設定値でパラメータを更新します。 `{ "account_name": "lza-test-01", "account_email": "johnsmith+lzatest01@example.com", "support_dl": "johnsmith@example.com", "managed_org_unit": "Workloads/Workload-1", "ad_integration": [{ "PermissionSetName": "CustomerAccountAdmin", "ActiveDirectoryGroupName": "platform-admin" }, { "PermissionSetName": "CustomerAccountDev", "ActiveDirectoryGroupName": "workload1-app1" }],`	AWS DevOps

タスク	説明	必要なスキル

	<pre>"force_update": "true",
	"bypass_creation": "false",
	"account_tags": [
	{
	"Key": "Environment",
	"Value": "Dev"
	}, {
	"Key": "DeploymentMethod",
	"Value": "ApiGateway"
	}
	]
	}</pre>
	3. api_example.json ファイルを保存して閉じます。
	4. 次のコマンドを入力して、Step Functions ワークフローを開始します。
	<pre>awscurl --service execute-api \
	--data @api-example.json \
	--region ${AWS_REGION} \
	--access_key ${AWS_ACCESS_KEY_ID} \
	--secret_key ${AWS_SECRET_ACCESS_KEY} \
	--security_token ${AWS_SESSION_TOKEN} \</pre>	

タスク	説明	必要なスキル

	<pre>-X POST ${API_GATEWAY_ENDPOINT}execute</pre>
	5. 前のコマンドの出力で、Step Functions 実行 Amazon リソースネーム (ARN) を書き留めます。
	6. Step Functions ワークフローのステータスを確認する場合は、次のコマンドを入力します。を Step Functions 実行 ARN または名前<STEP_FUNCTION_EXECUTION_NAME> に置き換えます。
	<pre>awscurl --service execute-api \
	--region ${AWS_REGION} \
	--access_key ${AWS_ACCESS_KEY_ID} \
	--secret_key ${AWS_SECRET_ACCESS_KEY} \
	--security_token ${AWS_SESSION_TOKEN} \
	-X GET ${API_GATEWAY_ENDPOINT}get_execution_status?execution=<STEP_FUNCTION_EXECUTION_NAME></pre>	

タスク	説明	必要なスキル
	7. アカウントが正常に作成されたという通知を受信するまで待ちます。 **Note** アカウント作成プロセスが失敗すると、Amazon SNS は、 `deploy-config.yaml` ファイル`accountCreationFailure` で で定義した E メールアドレスに通知を送信します。アカウントリクエスタには通知されません。	

(オプション) ソリューションをクリーンアップする

タスク	説明	必要なスキル
Amazon S3 バケットからオブジェクトを削除します。	次の Amazon S3 バケット内のオブジェクトを削除します。 • `lza-account-creation-work-<CDK_UNIQUE_ID>` • `lza-account-creation-workflow-src-<A`	AWS DevOps

タスク	説明	必要なスキル
	WS_REGION>-<AWS_ACCOUNT> • lza-account-creation-workflow-<AWS_REGION>-<AWS_ACCOUNT>	

CloudFormation スタックを削除します。	次の コマンドを実行して CloudFormation スタックを削除します。
	`aws cloudformation delete-stack \ --stack-name lza-account-creation-workflow-application`
	`aws cloudformation wait stack-delete-complete \ --stack-name lza-account-creation-workflow-application`
		AWS DevOps

パイプラインを削除します。	lza-account-creation-workflow-pipeline パイプラインを削除するには、次のコマンドを入力します。 `cdk destroy lza-account-creation-workflow-pipeline --force`	AWS DevOps

関連リソース

- Landing Zone Accelerator on AWS (AWS ソリューションライブラリ)

- 一般的な AWS CDK 問題のトラブルシューティング (AWS CDK ドキュメント)

追加情報

Step Functions ワークフロー図

次の図は、Step Functions ワークフローの状態を示しています。

引数

Step Functions ワークフローを開始する Python スクリプトを実行するときに使用できる引数を次に示します。

以下の引数が必要です。

- `account-name` (`-a`) (文字列) – 新しい の名前 AWS アカウント。
- `support-dl` (`-s`) (文字列) – アカウント作成プロセスが完了したときに が通知を受け取る E メールアドレス。
- `managed-org-unit` (`-m`) (文字列) – 新しいアカウントを含むマネージド組織単位 (OU)。

以下の引数はオプションです。

- `ad-integration` (`-ad`) (文字列ディクショナリ) – Microsoft Entra ID グループと割り当てられたアクセス許可セット。この引数の使用方法の例を次に示します。

```
--ad-integration "{\"<PermissionSetName>\": \"<EntraIdGroupName>\"}"
```

- `account-email` (`-e`) (文字列) – 新しい のルートユーザーの E メールアドレス AWS アカウント。

> **Note**
>
> この引数を使用しない場合、 `configs/deploy-config.yaml` ファイル`rootEmailDomain`から `rootEmailPrefix`と の値を使用して E メールアドレスが生成されます。E メールアドレスが指定されていない場合、E メールアドレスは の形式を使用して生成されます`rootEmailPrefix+accountName@rootEmailDomain`。

- region（-r）(文字列) – Step Functions ワークフロー AWS リージョン がデプロイされた 。デフォルト値は us-east-1 です。
- force-update（-f）(文字列ブール値) – と入力trueすると、プロビジョニングされた製品 AWS Service Catalog の更新を強制します。
- bypass-creation（-b）(文字列ブール値) – とtrue入力すると、アカウントに accounts-config.yaml ファイルを追加したり、AWSAccelerator-Pipelineパイプラインを実行したりすることをバイパスできます。この引数は、通常、アカウント作成ワークフロープロセスをテストしたり、Landing Zone Acceleratorパイプラインでエラーが発生した場合に残りの Step Functions ステップを実行したりするために使用されます。
- tags（-t）(文字列) – に追加する追加のタグ AWS アカウント。デフォルトでは、、account-name、support-dlおよび のタグが追加されますpurpose。この引数の使用方法の例を次に示します。

```
--tags TEST1=VALUE1 TEST2=VALUE2
```

複数のアカウントとリージョンの AWS リソースを自動的にインベントリする

作成者: Matej Macek (AWS)

概要

このパターンは、複数のアカウントと の AWS リソースの包括的なインベントリを維持するための自動化されたアプローチの概要を示しています AWS リージョン。これは、インフラストラクチャエンジニアとセキュリティエンジニアがリソース管理プラクティスを改善するのに役立つように設計されています。を使用して AWS Config 、リソースの変更、クエリ用の Amazon Athena、インタラクティブダッシュボード用の Amazon QuickSight を追跡します。このソリューションを実装するには、 AWS CloudFormation スタックをデプロイします。

このソリューションは、Amazon Athena と Amazon QuickSight を使用した AWS Config データの視覚化」(AWS ブログ記事) で説明されているものと似ています。このパターンでは、このソリューションを拡張して、以下の一般的な要件に対処し、以下の主な利点を提供します。

- コンプライアンスに重点を置く – このアプローチは、PCI DSS、NIST SP 800-53、ISO/IEC 27001、HIPAA、GDPR、および正確なアセットインベントリを義務付けるその他の規制要件を満たすのに役立ちます。
- カスタマイズフレームワーク – さまざまな AWS リソースの QuickSight ダッシュボードを作成するための基盤を提供し、特定の要件に合わせてソリューションをカスタマイズできます。
- ユーザー主導の機能強化 – このアプローチでは、実際のユースケースからのフィードバックを取り入れ、より包括的なソリューションのリクエストに対処します。

インフラストラクチャ、セキュリティ、財務の各チームは、多くの場合、動的、マルチアカウント、またはマルチリージョン環境で可視性とコラボレーションの課題に直面します。このソリューションは、これらの課題に対処し、リソースインベントリの作成と維持に必要な時間と労力を大幅に削減するように設計されています。その結果、リソース割り当ての決定の改善、リスクの特定と軽減、コストの最適化、全体的な可視性とコラボレーションの向上に役立つリソースの一元的なビューが得られます。このアプローチは、セキュリティ、コンプライアンス、運用上の目的における概念的なソリューションと実際の実装ニーズとのギャップを埋めます。

前提条件と制限

前提条件

- 次のアクティブな AWS アカウント：
  - 管理アカウント - 請求、アカウントの作成、組織全体のアクセスの制御のための一元化されたアカウント
  - 監査アカウント – セキュリティモニタリング、コンプライアンスチェック、ドリフト通知のための一元化されたハブ
  - ログアーカイブアカウント – 収集されたデータを保存および分析するための一元化されたアカウント
- 監査アカウントでは、ターゲットアカウントとリージョンから設定データを収集して集計する AWS Config アグリゲータ
- ログアーカイブアカウントで、以下を設定します。
  - アグリゲータからデータを保存する Amazon Simple Storage Service (Amazon S3) AWS Config バケット
  - Amazon QuickSight サブスクリプション
  - QuickSight と Amazon Athena 間の認可された接続
  - Athena クエリを介して Amazon S3 バケットにアクセスするアクセス許可
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- 次のリソースをプロビジョニングする CloudFormation スタックをデプロイするアクセス許可。
  - AWS Lambda 関数
  - Amazon S3 通知設定
  - Athena データベース、テーブル、ビュー
  - QuickSight データセットとデータソース
- でオートメーションを実行するアクセス許可 AWS Systems Manager
- QuickSight へのアクセス許可

制約事項

- このソリューションは に依存します AWS Config。 AWS Config は通常、変更が検出された直後、または指定した頻度でリソースの設定変更を記録します。ただし、これはベストエフォートベースであり、場合によっては時間がかかることがあります。
- このソリューションは、 がAWS Config サポートするリソースタイプのみを追跡します。
- このソリューションは、他のクラウドプロバイダーやオンプレミス環境全体のリソースインベントリを追跡しません。

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、 AWS ドキュメントの「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、 AWS 組織内の複数のアカウントで設定およびコンプライアンスデータを収集、整理、分析、視覚化するための効率的なプロセスを示しています。

この図表は、次のワークフローを示しています：

1. ア AWS Config グリゲータは定期的にターゲットアカウントとリージョンのリソースに関する設定とコンプライアンスデータを収集し、ログアーカイブアカウントの Amazon S3 バケットにデータを配信します。
2. Amazon S3 バケットに新しい AWS Config データを追加すると、 AWS Lambda 関数が呼び出されます。
3. Lambda 関数は、各スナップショットファイルのリージョンと日付に対応する値を持つキーを設定することで、データをパーティション化します。これにより、設定とコンプライアンスデータを AWS Glue 効率的にクエリして処理できます。
4. Amazon Athena は AWS Glue スキーマを使用して、Amazon S3 バケットに保存されているデータに対して SQL クエリを実行します。のスキーマメタデータを使用して AWS Glue 、データの構造を理解します。
5. Athena のビューは、ターゲットデータセットを定義して抽出します。
6. Amazon QuickSight のダッシュボードは、データセットを視覚化および分析するのに役立ちます。

ツール

AWS のサービス

- 「Amazon Athena」はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接分析できます。

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- AWS Config は、 のリソース AWS アカウント とその設定方法の詳細ビューを提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。 AWS Config アグリゲータは、複数の および リージョンから AWS Config 設定 AWS アカウント およびコンプライアンスデータを収集します。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。このパターンでは、 AWS Glue データカタログとスキーマレジストリを使用します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- Amazon QuickSight は、単一ダッシュボードでデータを視覚化、分析、レポートできるクラウドスケールのビジネスインテリジェンス (BI) サービスです。
- Amazon Simple Storage Service (Amazon S3) は、データ量にかかわらず、保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間が短縮され、 AWS リソースを大規模に安全に管理できます。AWS Systems Manager 自動化により、多くの の一般的なメンテナンス、デプロイ、修復タスクが簡素化されます AWS のサービス。

コードリポジトリ

このパターンの AWS CloudFormation テンプレートは、AWS Config 視覚化 GitHub リポジトリで使用できます。この CloudFormation テンプレートは、Amazon Athena で使用する AWS Config ように を設定する AWS Systems Manager オートメーションランブックをデプロイします。この自動化は、指定された Amazon S3 バケットに接続する準備 AWS Glue をし、Amazon Athena でビューを作成し、ダッシュボードの視覚化用に Amazon QuickSight を設定します。

ベストプラクティス

- AWS 「 規範ガイダンス」の「 を使用した安全なマルチアカウント AWS 環境のセットアップと管理 AWS Control Tower」のベストプラクティスに従うことをお勧めします。
- AWS 組織全体の設定とコンプライアンスデータを収集する AWS Config アグリゲータを作成することをお勧めします。詳細については、 AWS Config ドキュメントの「マルチアカウントマルチリージョンデータ集約」を参照してください。
- このソリューションをデプロイする前に、Amazon S3、、AWS ConfigAthena、QuickSight の現在の料金情報を確認することをお勧めします。

エピック

CloudFormation スタックをデプロイする

タスク	説明	必要なスキル
CloudFormation テンプレートをダウンロードします。	Config-QuickSight-Visualization-SSM-Automation.yaml CloudFormation テンプレートをダウンロードします。	AWS 管理者、クラウド管理者、DevOps エンジニア
CloudFormation テンプレートを変更します。	を使用してAWS Control Towerいて、 AWS Config によって管理されている場合にのみ、このステップを完了します AWS Control Tower。CloudFormation テンプレートを変更する必要があります。 1. 管理アカウントにサインインします。 2. AWS Organizations コンソールを開きます。 3. 設定ページに移動します。このページには、組織 ID	DevOps エンジニア、AWS 管理者

タスク	説明	必要なスキル

	など、組織に関する詳細が表示されます。
	4. 組織 ID をコピーします。
	5. 任意のテキストエディタで、Config-QuickSight-Visualization-SSM-Automation.yaml ファイルを開きます。
	6. 次の行を見つけます。
	`return re.match( '^AWSLogs/(\d+)/Config/([\w-]+)/(\d+)/(\d+)/(\d+)/(\d+)/ConfigSnapshot/[^\\\]+$', object_key)`
	7. この行を以下に置き換えます。ここで、 <ORGANIZATION_ID> は以前にコピーした ID です。
	`return re.match( '^<ORGANIZATION_ID>/AWSLogs/(\d+)/Config/([\w-]+)/(\d+)/(\d+)/(\d+)/(\d+)/ConfigSnapshot/[^\\\]+$', object_key)`
	8. Config-QuickSight-Visualization-SSM-Automation.yaml ファイルを保存して閉じます。	

タスク	説明	必要なスキル
CloudFormation スタックを作成します。	CloudFormation コンソールからスタックを作成する」の手順に従います。次の点に注意してください: 1. テンプレートファイルをアップロードを選択し、ダウンロードした YAML ファイルを選択します。 2. [スタック名] に「`Config-QuickSight-Visualization-SSM-Automation` 」と入力します。 3. [Submit] を選択してください。	AWS 管理者、クラウド管理者、DevOps エンジニア

Systems Manager でオートメーションを実行する

タスク	説明	必要なスキル
QuickSight ユーザー名を見つけます。	1. QuickSight コンソールを開きます。 2. プロファイルメニューを開きます。 3. ユーザー名を書き留めます。この値は後で必要になります。	AWS 管理者、クラウド管理者、DevOps エンジニア
配信チャネル名と Amazon S3 バケット名を検索します。	1. で AWS CLI、次のコマンドを入力します。	AWS 管理者、クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	``` aws configservice describe-delivery- channels ``` 2. Amazon S3 バケット名とAWS Config 配信チャネルの名前を書き留めます。これらの値は後で必要になります。	

タスク	説明	必要なスキル
Systems Manager でオートメーションを実行します。	1. AWS Systems Manager コンソールを開きます。 2. ナビゲーションペインで、[ドキュメント] を選択します。 3. [自己所有] を選択します。 4. 「Config-QuickSight-Visualization」を選択します。 5. [Execute automation (自動化の実行)] を選択してください。 6. 入力パラメータ セクションに、次のパラメータの値を入力します。 • `ConfigDeliveryChannelName` – AWS Config 配信チャネルの名前を入力します。このパラメータは必須です。 • `ConfigS3BucketLocation` – AWS Config 設定データを保存する Amazon S3 バケットの名前を入力します。このパラメータは必須です。 • `QuickSightUserName` – QuickSight への管理アクセス権を持つユーザー名を入力します。このパラメータは必須です。	AWS 管理者、クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	• `AutomationAssumeRole` – Systems Manager Automation がユーザーに代わってアクションを実行できるようにする AWS Identity and Access Management (IAM) ロールの Amazon リソースネーム (ARN)。このパラメータはオプションです。このパラメータは空白のままにします。 • `DeleteConfigVisualization` – を選択しますfalse。 7. [実行] を選択してください。	

Amazon QuickSight でデータを視覚化する

タスク	説明	必要なスキル
データを更新します。	特定の要件に従ってデータセットの更新をスケジュールするには、「SPICE データの更新」の手順に従います。	AWS 管理者、DevOps エンジニア、クラウド管理者
分析を作成します。	リソースを視覚化するのに役立つダッシュボードを QuickSight で作成するには、Amazon QuickSight で分析を開始する」の手順に従います。	QuickSight 管理者

タスク	説明	必要なスキル
ダッシュボードを作成します。	1. QuickSight 分析の変更が完了したら、「ダッシュボードの公開」の手順に従ってダッシュボードを作成します。ダッシュボードは、他の QuickSight ユーザーと共有できる分析です。 2. 「ダッシュボードへのアクセスを許可する」の手順に従って、ダッシュボードをターゲット QuickSight ユーザーと共有します。	QuickSight 管理者

(オプション) クリーンアップする

タスク	説明	必要なスキル
Systems Manager オートメーションによって作成されたリソースを削除します。	1. AWS Systems Manager コンソールを開きます。 2. ナビゲーションペインで、[ドキュメント] を選択します。 3. [自己所有] を選択します。 4. 「Config-QuickSight-Visualization」を選択します。 5. [Execute automation (自動化の実行)] を選択してください。 6. 入力パラメータセクションで、 `DeleteConfigVisualization` パ	AWS 管理者、クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	ラメータに と入力しますtrue。 7. [実行] を選択してください。	
CloudFormation スタックを削除します。	Config-QuickSight-Visualization-SSM-Automation スタック内のリソースを削除するには、CloudFormation コンソールからスタックを削除する」の手順に従います。	AWS 管理者、クラウド管理者、DevOps エンジニア

トラブルシューティング

問題	ソリューション

Amazon QuickSight は への接続を試みていますがus-east-1 AWS リージョン、そのリージョンでのリソースの作成は許可されていません。	サービスコントロールポリシーは、このリージョンでサブスクリプションを Amazon QuickSight に制限しています。サービスコントロールポリシーで、ターゲットを手動で指定します AWS リージョン。を適切なリージョン識別子<REGION_ID> に置き換えます。
	`https://<REGION_ID>.quicksight.aws.amazon.com/sn/start/dashboards`
	以下に例を示します。
	`https://eu-central-1.quicksight.aws.amazon.com/sn/start/dashboards`

Amazon Athena では、次のメッセージが表示されます。	Amazon Athena からのクエリ結果を保存する Amazon S3 バケットが準備されていることを

問題	ソリューション
`Before you run your first query, you need to set up a query result location in Amazon S3.`	確認します。 Amazon Athena 次に、Amazon Athena コンソールを使用してクエリ結果の場所を指定する」の手順に従います。

関連リソース

AWS ドキュメント

- AWS Config ドキュメント
- Amazon QuickSight ドキュメント

AWS ブログ投稿

- による AWS Config データの視覚化の自動化 AWS Systems Manager
- を使用してリソース設定の変更を定期的に記録する方法 AWS Config

その他のリソース

- Amazon QuickSight コミュニティラーニングセンター
- Amazon QuickSight コミュニティギャラリー

AWS アカウント全体にわたり一元化するための VPC フローログを設定

作成者: Benjamin Morris (AWS) と Aman Kaur Gandhi (AWS)

概要

Amazon Web Services（AWS）の仮想プライベートクラウド（VPC）では、VPC フローログの特徴量が運用上およびセキュリティ上のトラブルシューティングに役立つデータを提供します。ただし、マルチアカウント環境での VPC フローログの使用には制限があります。具体的には、Amazon CloudWatch Logs からのクロスアカウントフローログが適用されません。代わりに、Amazon Simple Storage Service (Amazon S3) バケットに適切なバケットポリシーを設定して、ログを一元化できます。

> **Note**
>
> このパターンでは、フローログを一元管理された場所に送信するための要件について説明します。ただし、メンバーアカウントでローカルにもログを利用できるようにする場合、各 VPC ごとに複数のフローログを作成できます。ログアーカイブアカウントにアクセスできないユーザーは、トラブルシューティングのトラフィックログを見ることができます。また、VPC ごとに 1 つのフローログを設定して、CloudWatch ログにログを送信することもできます。その後、Amazon Data Firehose サブスクリプションフィルターを使用してログを S3 バケットに転送できます。詳細については、「関連のリソース」 を参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- ログの一元管理に使用されるアカウント (ログアーカイブなど) を持つ AWS Organizations の組織

機能制限

AWS Key Management Service (AWS KMS)のマネージドキー `aws/s3` を使用して、中央バケットを暗号化すると、別のアカウントからのログは受信されません。代わりに、エラー`Unsuccessful`コード 400 が表示され、特定の `"LogDestination: <bucketName> is undeliverable"`の などのメッセージが表示されます`ResourceId`。

これは、アカウントの AWS マネージドキーをアカウント間で共有できないためです。

解決策は、 Amazon S3 マネージド暗号化 (SSE-S3)、または メンバーアカウントに共有できるAWS KMS カスタマーマネージドキー (SSE-S3)を使用することです。

アーキテクチャ

ターゲットテクノロジースタック

以下の図表では、角VPC ごとに 2つのフローログがデプロイされています。 1 つはローカルの CloudWatch ロググループにログを送信します。もう 1 つは、一元化されたロギングアカウントの S3 バケットにログを送信します。バケットポリシーでは、ログ配信サービスがバケットにログを書き込むことを許可します。

> **Note**
>
> 2023 年 11 月 AWS 現在、 は aws:SourceOrgID 条件キーをサポートしています。この条件により、 AWS Organizations 組織外のアカウントの一元化されたバケットへの書き込みを拒否できます。

ターゲット アーキテクチャ

自動化とスケール

各 VPC は、セントラルロギングアカウントの S3 バケットにログを送信するように設定されます。フローログが適切に設定されていることを確認するには、以下の自動化ソリューションのいずれかを使用してください：

- 「AWS CloudFormation StackSets」
- 「AWS Control Tower Account Factory for Terraform (AFT)」
- 「修正 AWS Config ルール」

ツール

ツール

- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、 AWS からのログを一元化することを支援して、ログを監視して安全にアーカイブできるようにします。
- 「Amazon Simple Storage Service (Amazon S3)」 は、どのようなデータの量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。このパターンでは、「VPC フローログ」特徴量 を使用して、 VPC のネットワークインターフェイスとの間で行き来する IP トラフィック情報をキャプチャします。

ベストプラクティス

Infrastructure as Code (IaC)を使用して、VPC フローログのデプロイプロセスを大幅に簡素化できます。VPC デプロイ定義を抽象化してフローログリソース構造を含める場合、フローログを含む VPC が自動的にデプロイされます。これについては、次のセクションで説明します。

一元化されたフローログ

HashiCorp Terraform の VPC モジュールに一元化されたフローログを追加するための構文例

このコードは、VPC から一元管理された S3 バケットにログを送信するフローログを作成します。このパターンは S3 バケットの作成には適用されないことに注意してください。

推奨されるバケットポリシーステートメントについては、「追加情報」 セクションを参照してください。

```
variable "vpc_id" { type = string }
locals { custom_log_format_v5 = "$${version} $${account-id} $${interface-id} $
${srcaddr} $${dstaddr} $${srcport} $${dstport} $${protocol} $${packets} $${bytes}
 $${start} $${end} $${action} $${log-status} $${vpc-id} $${subnet-id} $${instance-
id} $${tcp-flags} $${type} $${pkt-srcaddr} $${pkt-dstaddr} $${region} $${az-id} $
${sublocation-type} $${sublocation-id} $${pkt-src-aws-service} $${pkt-dst-aws-service}
 $${flow-direction} $${traffic-path}" }
resource "aws_flow_log" "centralized_flow_log" {
  log_destination      = "arn:aws:s3:::centralized-vpc-flow-logs-
<log_archive_account_id>" # Optionally, a prefix can be added after the ARN.
  log_destination_type = "s3"
  traffic_type         = "ALL"
  vpc_id               = var.vpc_id
```

```
  log_format           = local.custom_log_format_v5 # If you want fields from VPC Flow
 Logs v3+, you will need to create a custom log format.
}
```

カスタムログ形式の詳細については、AWS ドキュメントを参照してください。

ローカルフローログ

必要な権限を持つ Terraform の VPC モジュールにローカルフローログを追加する構文例

このコードは、VPC からローカルの CloudWatch ロゴグループにログを送信するフローログを作成します。

```
data "aws_region" "current" {}
variable "vpc_id" { type = string }
resource "aws_iam_role" "local_flow_log_role" {
  name = "flow-logs-policy-${var.vpc_id}"
  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Principal": {"Service": "vpc-flow-logs.amazonaws.com"},
      "Action": "sts:AssumeRole"
  }]
}
EOF
}
resource "aws_iam_role_policy" "logs_permissions" {
  name = "flow-logs-policy-${var.vpc_id}"
  role = aws_iam_role.local_flow_log_role.id
  policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [{
      "Action": ["logs:CreateLog*", "logs:PutLogEvents", "logs:DescribeLog*",
 "logs:DeleteLogDelivery"],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:${data.aws_region.current.name}:*:log-group:vpc-flow-
logs*"
  }]
}
EOF
```

```
}
resource "aws_cloudwatch_log_group" "local_flow_logs" {
  name              = "vpc-flow-logs/${var.vpc_id}"
  retention_in_days = 30
}
resource "aws_flow_log" "local_flow_log" {
  iam_role_arn    = aws_iam_role.local_flow_log_role.arn
  log_destination = aws_cloudwatch_log_group.local_flow_logs.arn
  traffic_type    = "ALL"
  vpc_id          = var.vpc_id
}
```

エピック

VPC フローログインフラストラクチャのデプロイ

タスク	説明	必要なスキル
暗号化戦略を決定し、中央の S3 バケットのポリシーを作成します。	中央バケットには aws/s3 AWS KMS キーが適用されますので、SSE-S3 または AWS KMS カスタマー管理キーを使用する必要があります。AWS KMS キーを使用する場合、キーポリシーでメンバーアカウントにキーの使用を許可する必要があります。	コンプライアンス
中央フローログバケットを作成します。	フローログの送信先となる中央バケットを作成し、前のステップで選択した暗号化戦略を適用します。これはログアーカイブまたは同様の目的のアカウントに在る必要があります。 「追加情報」セクションからバケットポリシーを取得し、環境固有の値でプレースホル	AWS 全般

タスク	説明	必要なスキル
	ダーを更新してから中央バケットに適用します。	
VPC フローログを設定して、中央のフローログバケットにログを送信するようにします。	データの収集元の各 VPC にフローログを追加します。これを行う最もスケーラブルな方法は、AFT や AWS Cloud Development Kit (AWS CDK) などの IaC ツールを使用することです。たとえば、フローログと一緒に VPC をデプロイする Terraform モジュールを作成できます。必要に応じて、フローログを手動で追加します。	ネットワーク管理者
VPC フローログをローカル CloudWatch ログに送信するように設定します。	(オプション) ログが生成されているアカウントでフローログを表示する場合、別のフローログを作成して、ローカルアカウントの CloudWatch Logs にデータを送信します。あるいは、ローカルアカウントのアカウント固有の S3 バケットにデータを送信することもできます。	AWS 全般

関連リソース

- 「一元化されたフローログデータを使用してデータ分析を容易にし、セキュリティ要件を満たす方法」(ブログ記事)
- 「AWS Config ルールを使用して VPC フローログを自動的に有効にする方法」(ブログ記事)

追加情報

バケットポリシー

このバケットポリシーの例は、プレースホルダー名の値を追加した後に、フローログ用のセントラル S3 バケットに適用できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<BUCKET_NAME>/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceOrgID": "<ORG_ID>"
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::<BUCKET_NAME>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceOrgID": "<ORG_ID>"
                }
            }
        },
        {
            "Sid": "DenyUnencryptedTraffic",
            "Effect": "Deny",
            "Principal": {
```

```
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::<BUCKET_NAME>/*",
                "arn:aws:s3:::<BUCKET_NAME>"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

AWS Organizations を使用して Transit Gateway アタッチメントに自動的にタグを付ける

作成者:Richard Milner-Watts (AWS), Haris Bin Ayub (AWS), and John Capps (AWS)

概要

アマゾン ウェブ サービス (AWS) ではAWS Resource Access Manager、 を使用して AWS アカウント 境界AWS Transit Gatewayを越えて共有できます。ただし、アカウントの境界を越えて Transit Gateway アタッチメントを作成すると、アタッチメントは名前タグなしで作成されます。そのため、アタッチメントの識別に時間がかかることがあります。

このソリューションは、 によって管理されている組織内のアカウントの各 Transit Gateway アタッチメントに関する情報を収集する自動メカニズムを提供しますAWS Organizations。このプロセスには、Transit Gateway のルートテーブルから「Classless Inter-Domain Routing (CIDR)」範囲を検索することが含まれます。次に、このソリューションでは、トランジットゲートウェイを保有するアカウント内の添付ファイルに、`<CIDR-range>-<AccountName>` という形式の名前タグを適用します。

このソリューションは、 ソリューションライブラリのサーバーレストランジットネットワークオーケストレーターなどの AWS ソリューションと一緒に使用できます。Serverless Transit Network Orchestrator を使用すると、Transit Gateway Gateway アタッチメントを大規模に自動作成できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- すべての関連アカウントを含む AWS Organizations 組織
- 必要な AWS Identity and Access Management (IAM) ロールを作成するための、組織のルートにある組織管理アカウントへのアクセス
- 組織と共有され、添付ファイルがある 1 つ以上のトランジットゲートウェイを含む共有ネットワーキングメンバーアカウント

アーキテクチャ

次のスクリーンショットは、関連付けられた Name タグがない Transit Gateway アタッチメントと、このソリューションによって生成された Name タグがある 2 つの Transit Gateway アタッチ

メントの例 AWS Management Console を示しています。生成された名前タグの構造は <CIDR-range>-<AccountName> です。

このソリューションAWS CloudFormationでは、 を使用して、設定されたすべての で Transit Gateway Name タグの作成を管理するAWS Step Functionsワークフローをデプロイします AWS リージョン。ワークフローは、基盤となるタスクを実行する AWS Lambda関数を呼び出します。

ソリューションが からアカウント名を取得すると AWS Organizations、Step Functions ステートマシンはすべての Transit Gateway アタッチメント IDsを取得します。これらはリージョンごとに並行して処理されます。この処理には、各添付ファイルの CIDR 範囲の検索が含まれます。CIDR 範囲は、リージョン内の Transit Gateway ルートテーブルで一致するTransit Gateway アタッチメント ID を検索することで取得されます。必要な情報がすべて揃うと、ソリューションは添付ファイルに名前タグを適用します。ソリューションは既存の名前タグを上書きしません。

このソリューションは、「Amazon EventBridge」イベントによって制御されるスケジュールで実行されます。このイベントは、毎日午前 6 時 (UTC) にソリューションを開始します。

ターゲットテクノロジースタック

- Amazon EventBridge
- AWS Lambda
- AWS Organizations
- AWS Transit Gateway
- Amazon Virtual Private Cloud (Amazon VPC)
- AWS X-Ray

ターゲット アーキテクチャ

ソリューションのアーキテクチャとワークフローを次の図表に示しています。

1. スケジュールされたイベントによってルールが開始されます。
2. EventBridge ルールは Step Functions ステートマシンを起動します。
3. ステートマシンは tgw-tagger-organizations-account-query Lambda 関数を呼び出します。

4. tgw-tagger-organizations-account-query Lambda 関数は組織管理アカウントでの役割を引き受けます。
5. tgw-tagger-organizations-account-query Lambda 関数は Organizations API を呼び出して AWS アカウント メタデータを返します。
6. ステートマシンは tgw-tagger-attachment-query Lambda 関数を呼び出します。
7. 各リージョンについて、ステートマシンは並行に tgw-tagger-rtb-query Lambda 関数を呼び出して、各アタッチメントの CIDR 範囲を読み取ります。
8. 各リージョンについて、ステートマシンは並行に tgw-tagger-attachment-tagger Lambda 関数を呼び出します。
9. 名前タグは、共有ネットワークアカウントの Transit Gateway アタッチメントファイルに作成されます。

自動化とスケール

ソリューションは各リージョンを並行処理し、実行にかかる合計時間を短縮します。

ツール

AWS サービス

- AWS CloudFormation は、インフラストラクチャをコードとして扱うことで、関連 AWS リソースとサードパーティーリソースのコレクションをモデル化し、迅速かつ一貫してプロビジョニングし、ライフサイクルを通じて管理する方法を提供します。
- Amazon CloudWatch は、AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- Amazon EventBridge は、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。EventBridge は、環境の変化の指標であるイベントを受信し、イベントをターゲットにルーティングするルールを適用します。ルールは、イベントパターンと呼ばれるイベントの構造、またはスケジュールのいずれかに基づいて、イベントをターゲットにマッチングさせます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートするコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。支払いは、使用したコンピューティング時間に対する料金のみになります。コードが実行されていないときに料金は発生しません。

- AWS Organizations は、 AWS リソースの拡大とスケーリングに応じて、環境を一元的に管理および管理するのに役立ちます。Organizations を使用すると、プログラムで新しい を作成し、リソース AWS アカウント を割り当てたり、アカウントをグループ化してワークフローを整理したり、ガバナンスのためにアカウントまたはグループにポリシーを適用したり、すべてのアカウントに単一の支払い方法を使用して請求を簡素化したりできます。
- AWS Step Functions は、ビジネスプロセスのオーケストレーション AWS のサービス、自動化、サーバーレスアプリケーションの構築に使用されるローコードのビジュアルワークフローサービスです。ワークフローは障害、再試行、並列化、サービス統合、可観測性を管理するため、開発者はより価値の高いビジネスロジックに集中できます。
- AWS Transit Gateway は、中央のハブを介して VPC とオンプレミスネットワークを接続します。これにより、ネットワークが簡素化され、複雑なピアリング関係が終了します。クラウドルーターとして機能し、新しい接続は 1 回だけ行われます。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した論理的に分離された仮想ネットワークで AWS リソースを起動するためのサービスです。
- AWS X-Ray は、アプリケーションが処理するリクエストに関するデータを収集し、そのデータを表示、フィルタリング、インサイトを取得して、最適化の問題と機会を特定するために使用できるツールを提供します。

コード

このソリューションのソースコードは、「Transit Gateway アタッチメントタガー」GitHub リポジトリにあります。リポジトリには以下のファイルが含まれています。

- `tgw-attachment-tagger-main-stack.yaml` は、このソリューションをサポートするすべてのリソースを共有ネットワーキングアカウント内に作成します。
- `tgw-attachment-tagger-organizations-stack.yaml` は、組織の管理アカウントにロールを作成します。

エピック

メインソリューションスタックをデプロイ

タスク	説明	必要なスキル
必要な前提情報を収集する。	Lambda 関数から AWS Organizations API へのクロス	DevOps エンジニア

タスク	説明	必要なスキル
	アカウントアクセスを設定するには、組織の管理アカウントのアカウント ID が必要です。 **Note** 2 つの AWS CloudFormation スタックが作成される順序が重要です。最初に共有ネットワークアカウントにリソースをデプロイする必要があります。リソースを組織の管理アカウントに展開する前に、共有ネットワークアカウントのロールがすでに存在している必要があります。詳細については、AWS のドキュメントを参照してください。	

タスク	説明	必要なスキル
メインソリューションスタックの AWS CloudFormation テンプレートを起動します。	メインソリューションスタックのテンプレートは、IAM ロール、Step Functions ワークフロー、Lambda 関数、Amazon CloudWatch イベントをデプロイします。 AWS Management Console 共有ネットワーキングアカウントの を開き、:&CFN コンソールを開きます。 `tgw-attachment-tagger-main-stack.yaml` テンプレートと以下の値を使用してスタックを作成します。 • スタック名 — tgw-attachment-tagger-main-stack • AWS/OrganizationsRootAccountID — 組織の管理アカウントのアカウント ID • TGWRegions パラメータ – ソリューション AWS リージョン の場合は、カンマ区切りの文字列として入力します。 • TGWList パラメータ — ソリューションから除外するトランジットゲートウェイ ID。カンマ区切りの文字列で入力されます。 AWS CloudFormation スタックの起動の詳細については、	DevOps エンジニア

タスク	説明	必要なスキル
	AWS ドキュメントを参照してください。	
ソリューションが正常に起動したことを確認します。	CloudFormation スタックのステータスが [CREATE_COMPLETE] になるまでお待ちください。この所要時間は 1 分以内となります。 Step Functions コンソールを開き、tgw-attachment-tagger-state-machine という名前で新しいステートマシンが作成されていることを確認します。	DevOps エンジニア

AWS Organizations スタックをデプロイ

タスク	説明	必要なスキル
必要な前提情報を収集する。	Lambda 関数から AWS Organizations API へのクロスアカウントアクセスを設定するには、共有ネットワーキングアカウントのアカウント ID が必要です。	DevOps エンジニア
Organizations スタック用の CloudFormation テンプレートを起動します。	AWS Organizations スタックのテンプレートは、組織の管理アカウントにIAMロールをデプロイします。 組織の管理アカウントの AWS コンソールにアクセスし、CloudFormation コンソールを開きます。 tgw-attac	DevOps エンジニア

タスク	説明	必要なスキル
	hment-tagger-organizations-stack.yaml テンプレートと以下の値を使用してスタックを作成します。 • スタック名 — tgw-attachment-tagger-organizations-stack • NetworkingAccountID パラメーター — 共有ネットワークアカウントのアカウント ID その他のスタック作成オプションには、デフォルトを使用してください。	
ソリューションが正常に起動したことを確認します。	AWS CloudFormation スタックが CREATE_COMPLETE のステータスになるまで待ちます。この所要時間は 1 分以内となります。 AWS Identity and Access Management (IAM) コンソールを開き、tgw-attachment-tagger-organization-query-role という名前の新しいロールが作成されていることを確認します。	DevOps エンジニア

ソリューションを検証する

タスク	説明	必要なスキル
ステートマシンを実行します。	共有ネットワークアカウントの Step Functions コンソールを開き、ナビゲーションペインで [ステートマシン] を選択します。 ステートマシン [tgw-attachment-tagger-state-machine] を選択し、[実行開始] を選択します。 このステートマシンへの入力はソリューションでは使用されないため、既定値を使用できます。 `{ "Comment": "Insert your JSON here" }` [実行のスタート] を選択します。	DevOps エンジニア
ステートマシンが完了するまで監視してください。	開いた新しいページでは、ステートマシンの実行を確認できます。所要時間は、処理する Transit Gateway アタッチメントの数によって異なります。 このページでは、ステートマシンの各ステップを確認できます。ステートマシン	DevOps エンジニア

タスク	説明	必要なスキル
	内のさまざまなタスクを表示したり、Lambda 関数の CloudWatch ログへのリンクをたどったりできます。マップ内でparallel 実行されるタスクについては、[インデックス] ドロップダウンリストを使用して、各リージョンの特定の実装を表示できます。	
Transit Gateway アタッチメントタグを確認します。	共有ネットワークアカウントの VPC コンソールを開き、[Transit Gateway アタッチメント] を選択します。 コンソールでは、条件を満たす添付ファイルの名前タグが表示されます (添付ファイルは Transit Gateway のルートテーブルに伝達され、リソース所有者は組織のメンバーです)。	DevOps エンジニア

タスク	説明	必要なスキル
CloudWatch イベントの開始を確認します。	CloudWatch イベントが開始されるのを待ちます。これは 06:00 UTC に予定されています。 次に、共有ネットワークアカウントの Step Functions コンソールを開き、ナビゲーションペインで [ステートマシン] を選択します。 ステートマシン tgw-attachment-tagger-state-machine を選択します。ソリューションが 06:00 UTC に実行されたことを確認します。	DevOps エンジニア

関連リソース

- AWS Organizations
- AWS Resource Access Manager
- 「サーバーレストランジットネットワークオーケストレーター」
- 「IAM ロールの作成」
- AWS CloudFormation コンソールでのスタックの作成

その他のパターン

- GitHub Actions を使用して AWS CloudFormation テンプレートに基づいて AWS Service Catalog 製品をプロビジョニングする
- ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする

AI と機械学習

トピック

- Athena でのML 予測のため、Amazon DynamoDB 内のデータを集約
- ある の AWS CodeCommit リポジトリ AWS アカウント を別のアカウントの Amazon SageMaker AI Studio Classic に関連付ける
- Amazon Textract を使用して PDF ファイルからコンテンツを自動的に抽出する
- Amazon SageMaker AI Studio Lab の時系列に DeepAR を使用してコールドスタート予測モデルを構築する
- Amazon SageMaker AI と Azure DevOps を使用して MLOps ワークフローを構築する
- を使用して Amazon Bedrock でモデル呼び出しログ記録を設定する AWS CloudFormation
- SageMaker 用のカスタム Docker コンテナイメージを作成し、AWS Step Functions のモデルトレーニングに使用する
- Amazon Bedrock エージェントを使用して、テキストベースのプロンプトによる Amazon EKS でのアクセスエントリコントロールの作成を自動化する
- Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする
- Amazon SageMaker の推論パイプラインを使用して、前処理ロジックを単一エンドポイントの ML モデルにデプロイする
- RAG および ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する
- Amazon Bedrock エージェントとナレッジベースを使用して、完全に自動化されたチャットベースのアシスタントを開発する
- Amazon Bedrock と Amazon Transcribe を使用して、音声入力から組織の知識を文書化する
- Amazon Personalize を使用して、パーソナライズされ再ランク付けされたレコメンデーションを生成します
- SageMaker AI と Hybrid を使用して、ローカル開発からスケーラブルな実験まで機械学習ワークフローを合理化する
- GPU がサポートするカスタム ML モデルを Amazon SageMaker でトレーニングしてデプロイする
- 自然言語を OpenSearch クエリと Elasticsearch クエリのクエリ DSL に変換する
- Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める
- SageMaker Processing を使用して、テラバイト規模の ML データセットの分散型特徴量エンジニアリングを実現

- フラスコと AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化
- その他のパターン

Athena でのML 予測のため、Amazon DynamoDB 内のデータを集約

作成者：Sachin Doshi (AWS) と Peter Molnar (AWS)

概要

このパターンは、Amazon Athena で Amazon DynamoDB テーブル内のモノのインターネット (IoT) データの複雑な集約を構築する方法を示しています。また、Amazon SageMaker AI を使用して機械学習 (ML) 推論でデータを強化する方法と、Athena を使用して地理空間データをクエリする方法についても説明します。このパターンは、組織の要件を満たす ML 予測ソリューションを作成するための基礎として使用できます。

デモの目的で、このパターンでは、スクーターシェアリングを運営する企業が、さまざまな都市部の顧客に展開しなければならない最適なスクーターの台数を予測したいと考えているシナリオを例に挙げています。この企業では、過去 4 時間に基づき、次の 1 時間の顧客需要を予測する、事前にトレーニングされた ML モデルを使用しています。このシナリオでは、「ルイビル市市民イノベーション技術局」が公開しているデータセットを使用しています。このシナリオのリソースは GitHub リポジトリで利用できます。

前提条件と制限

- アクティブな AWS アカウント
- 以下の AWS Identity and Access Management (IAM) ロールを使用して AWS CloudFormation スタックを作成するアクセス許可。
  - Amazon Simple Storage Service (Amazon S3) バケット
  - Athena
  - DynamoDB
  - SageMaker AI
  - AWS Lambda

アーキテクチャ

テクノロジースタック

- Amazon QuickSight

- Amazon S3
- Athena
- DynamoDB
- Lambda
- SageMaker AI

ターゲット アーキテクチャ

次の図は、Athena、Lambda 関数、Amazon S3 ストレージ、SageMaker AI エンドポイント、QuickSight ダッシュボードのクエリ機能を使用して、DynamoDB でデータの複雑な集約を構築するためのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. DynamoDB テーブルは、複数のスクーターから送信される IoT データを取り込みます。
2. Lambda 関数は、取り込まれたデータを DynamoDB テーブルにロードします。
3. Athena クエリは、都市部を表す地理空間データ用の新しい DynamoDB テーブルを作成します。
4. クエリの位置は S3 バケットに保存されます。
5. Athena 関数は、事前トレーニング済みの ML モデルをホストする SageMaker AI エンドポイントから ML 推論をクエリします。
6. Athena は DynamoDB テーブルから直接データをクエリし、データを統合して分析します。
7. ユーザーは、分析されたデータの出力を QuickSight ダッシュボードに閲覧できます。

ツール

AWS のサービス

- 「Amazon Athena」標準 SQL を使用して Amazon S3 内のデータを直接分析するのに役立つインタラクティブなクエリサービスです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- Amazon SageMaker AI は、ML モデルを構築してトレーニングし、本番環境に対応したホスト環境にデプロイするのに役立つマネージド ML サービスです。

- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「Amazon QuickSight」は、クラウドスケールのビジネスインテリジェンス (BI) サービスで、データを単一のダッシュボードで可視化、分析、レポートすることができます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

コードリポジトリ

このパターンのコードは、 GitHub 内の「Amazon Athena ML で Amazon DynamoDB データに対する ML 予測を使用」リポジトリで利用できます。リポジトリから CloudFormation テンプレートを使用して、サンプルシナリオで使用される以下のリソースを作成できます。

- DynamoDB テーブル。
- テーブルに関連データをロードする Lambda 関数
- Amazon S3 に保存されている事前トレーニング済み XGBoost モデルを使用した、推論リクエスト用の SageMaker Amazon S3エンドポイント XGBoost
- `V2EngineWorkGroup` という Athena ワークグループ
- 名前付きの Athena クエリで地理空間シェープファイルを検索し、スクーターの需要を予測します。
- Amazon Athena DynamoDB Athena DynamoDB コネクタ DynamoDB AWS Serverless Application ModelAWS SAM

エピック

サンプルデータセットを取得

タスク	説明	必要なスキル
データセットとリソースをダウンロードします。	1. 「ドックレス車両レンタルの公開データセット」をダウンロードします。デモ目的では、このデータはユースケースの一部として	アプリ開発者、データサイエンティスト

タスク	説明	必要なスキル
	DynamoDB に事前入力されますが、本番環境ではIoTデバイスや Amazon Kinesis コンシューマーなどのさまざまなメカニズムを通じてこのデータを DynamoDB に送信します。これらのメカニズムは Lambda で DynamoDB にデータを導入します。 2. ケンタッキー州ルイビル市内の歴史地区と文化地区の境界を表す「GIS シェープファイル」をダウンロードします。公開データセットは、「Louisville and Jefferson County, KY Information Consortium」により提供されています。元のシェープファイルはすでに Athena でクエリできるテキストファイルに変換されていますが、シェープファイルを変換する Python コードは、GitHub で「Amazon Athenaを使ってGIS shapefilesを地理空間的に処理する」 Jupyter Notebookにあります。 3. SageMaker AI と Athena を使用して時間単位の予測の ML モデルをトレーニングする事前トレーニング済み	

タスク	説明	必要なスキル
	の Python コードをダウンロードします。 4. すべてをまとめて DynamoDB に保存されているデータからライブ予測を行う SQL クエリを Athena で取得します。 5. (オプション) QuickSight を使用して、「ケンタッキー州ルイビルのマップ上の地理空間データを視覚化します」。	

CloudFormation テンプレートで必要なリソースをデプロイします。

タスク	説明	必要なスキル
CloudFormation スタックを作成します。	1. CloudFormation テンプレートを GitHub「リポジトリ」からダウンロードします。 2. **Note** にサインインし AWS Management Console、 を選択しますus-east-1 。: ML モデルは の Amazon Elastic Container Registry (Amazon ECR) に保存されますがus-east-1 AWS リージョン、パターンはリージョンに依	AWS DevOps

タスク	説明	必要なスキル
	存しません。このパターン AWS のサービス で使用されるがサポートされている任意のリージョンでパターンをレプリケートできます。 3. 「CloudFormation コンソール」を開き、ナビゲーションペインで [スタック] を選択します。 4. [スタックを作成] を選択し、[既存のリソースを使用 (リソースのインポート)] を選択します。 5. リソースの識別ページで次へを選択します。 6. テンプレートを指定セクションのテンプレートソースで、テンプレートファイルをアップロードを選択します。 7. [ファイル] を選択し、先ほどダウンロードした CloudFormation テンプレートを選択します。 8. 次へを選択し、デフォルトのパラメータ値をそのまま使用し、次へを選択してセットアップウィザードの残りの部分を進めます。 9. がカスタム名で IAM リソースを作成する AWS	

タスク	説明	必要なスキル
	CloudFormation 可能性があることを確認するチェックボックスをオンにします。 10[スタックの作成] を選択してください。 **Note** CloudFormation スタックがこれらのリソースを作成するまでに 15～20 分かかる場合があります。	

タスク	説明	必要なスキル
CloudFormation のデプロイを確認します。	CloudFormation テンプレートからのサンプルデータが DynamoDB にロードされたことを確認するには、次の操作を実行します。 1. [DynamoDB コンソール] を開き、ナビゲーションペインから [テーブル] を選択します。 2. [テーブル] セクションで、`DynamoDBTableDocklessVehicles` テーブルを確認します。 3. リソースの作成が完了したら、「Athena コンソール」を開き、ナビゲーションペインからワークグループを選択します。 4. [`V2EngineWorkGroup` ワークグループ] を選択し、[ワークグループの切り替え] を選択します。 5. クエリ結果の位置を保存するように求めるプロンプトが表示されたら、書き込み権限がある Amazon S3 の位置を選択します。 6. [Save] を選択します。 7. ナビゲーションペインで、[クエリーエディター] を選	アプリ開発者

タスク	説明	必要なスキル
	択し、データベースを選択します。	

位置情報ファイルを Athena にロード

タスク	説明	必要なスキル
Athena テーブルを地理空間データにより作成します。	Athena に位置情報ファイルを読み込むには、次の操作を実行します。 1. 「Athena コンソール」を開き、ナビゲーションペインから [クエリエディター] を選択します。 2. [保存済みクエリ] タブを選択します。 3. Q1: Neighborhoods を検索して選択します。 4. クエリエディタに戻るには、 [Editor] タブを選択します。 5. [Run] (実行) を選択します。これにより、データベースにと `louisville_ky_neighborhoods` いうテーブルが作成されます。`athena-ml-db-<your-AWS-account-number>` データベースにテーブルが作成されていることを確認してください。	データエンジニア

タスク	説明	必要なスキル
	このクエリは、都市部を表す地理空間データ用の新しいテーブルを作成します。データテーブルは GIS シェープファイルから作成されます。`CREATE EXTERNAL TABLE` このステートメントは、テーブルのスキーマと、基になるデータファイルの位置と形式を定義します。 シェープファイルを処理し、このテーブルを生成する Python コードについては、 AWS 「サンプル」の「Geo-Spatial processing of GIS shapefiles with Amazon Athena」を参照してください。SQL コードの詳細については、GitHub の「create_neighborhood_table.sql」を参照してください。	

集約された DynamoDB データから地域別のスクーターの需要を予測

タスク	説明	必要なスキル
Athena で関数を宣言して SageMaker AI をクエリします。	1. 「Athena コンソール」を開き、ナビゲーションペインから [クエリエディター] と [エディター] を順に選択します。	データサイエンティスト、データエンジニア

タスク	説明	必要なスキル
	2. 次の SQL ステートメントをコピーしてクエリエディタに貼り付けます。 `USING EXTERNAL FUNCTION predict_demand ( location_id BIGINT, hr BIGINT , dow BIGINT, n_pickup_1 BIGINT, n_pickup_2 BIGINT, n_pickup_3 BIGINT, n_pickup_4 BIGINT, n_dropoff_1 BIGINT, n_dropoff_2 BIGINT, n_dropoff_3 BIGINT, n_dropoff_4 BIGINT ) RETURNS DOUBLE SAGEMAKER '<Your SageMaker endpoint>'` SQL ステートメントの最初の部分は、事前トレーニング済みモデルをホストする SageMaker AI エンドポイントから ML 推論をクエリする外部関数を宣言します。 3. 入力パラメータの順序とタイプと戻り値のタイプを定義します。 4. [Run] (実行) を選択します。	

タスク	説明	必要なスキル
集約された DynamoDB データから地域別のスクーターの需要を予測します。	これで、Athena により、DynamoDB から直接トランザクションデータをクエリし、データを集約して分析と予測を行うことができます。これは、DynamoDB NoSQL データベースに直接クエリを実行しても簡単には実現できません。 1. 「Athena コンソール」を開き、ナビゲーションペインから [クエリエディター] を選択します。 2. [保存済みクエリ] タブを選択します。 3. Q2：DynamodBathenAMLScooterPredict を検索して選択します。 4. クエリエディタに戻るには、[エディタ] タブを選択します。 5. [Run] (実行) を選択します。 SQL ステートメントは次のことを行います。 • 「Athena 横串検索」で、未加工のトリップデータで DynamoDB テーブルをクエリします。	アプリ開発者、データサイエンティスト

タスク	説明	必要なスキル
	• Athena の地理空間関数を使用して、地理座標を近傍に配置します。 • SageMaker AI を使用して ML 推論でデータを強化する SQL を使用して Athena で DynamoDB データおよび SageMaker AI 推論データを集約する方法については、GitHub の「athena_long.sql」を参照してください。	
出力の検証	出力テーブルには、近傍と近傍の重心の経度と緯度を含んでいます。また、次の 1 時間に予測される車両の数も含んでいます。 このクエリでは、選択した時点の予測が生成されます。ステートメントのあらゆる箇所で `TIMESTAMP '2019-09-07 15:00'` 式を変更することで、それ以外の時点の予測を行うことができます。 DynamoDB テーブルにリアルタイムデータフィードがある場合は、タイムスタンプを `NOW()` に変更します。	アプリ開発者、データサイエンティスト

環境をクリーンアップ

タスク	説明	必要なスキル
リソースの削除	1. 「Athena コンソール」を開き、CloudFormation スタックの一部として作成した「バケットを空に」します。 2. 「CloudFormation コンソール」を開き、`bdb-1462-athena-dynamodb-ml-stack` という「スタックを削除」します。 3. 「Amazon CloudWatch コンソール」を開き、`/aws/sagemaker/Endpoints/Sg-athena-ml-dynamodb-model-endpoint` という名前の「ログ グループを削除」します。	アプリ開発者、AWS DevOps

関連リソース

- 「Amazon Athena クエリフェデレーション SDK」(GitHub)
- 地理空間データのクエリ (AWS ドキュメント)
- Amazon Athena ML で Amazon DynamoDB データに対する ML 予測 Amazon Athenaを使用する (AWS ビッグデータブログ)
- Amazon ElastiCache (Redis OSS) (AWS ドキュメント)
- Amazon Neptune (AWS ドキュメント)

ある の AWS CodeCommit リポジトリ AWS アカウント を別のアカウントの Amazon SageMaker AI Studio Classic に関連付ける

ローレンス・ヴァン・デル・マース (AWS) とオーブリー・ウーストハイゼン (AWS) によって作成されました

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンでは、ある AWS アカウント （ アカウント A) の AWS CodeCommit リポジトリを別の (AWS アカウント アカウント B) の Amazon SageMaker AI Studio Classic に関連付ける方法の手順とコードを示します。関連付けを設定するには、アカウント A に AWS Identity and Access Management (IAM) ポリシーとロールを作成し、アカウント B に IAM インラインポリシーを作成する必要があります。次に、シェルスクリプトを使用して、アカウント A からアカウント B の Amazon SageMaker AI Classic に CodeCommit リポジトリのクローンを作成します。

前提条件と制限

前提条件

- 2 つの AWS アカウント。1 つは CodeCommit リポジトリを含み、もう 1 つはユーザーを含む SageMaker AI ドメインを含みます。
- プロビジョニングされた SageMaker AI ドメインとユーザー。インターネットアクセスまたは仮想プライベートネットワーク (VPC AWS STS) エンドポイントを介した CodeCommit および AWS Security Token Service () へのアクセスが可能
- 「IAM」の基本的な理解
- SageMaker AI Studio Classic の基本的な理解
- Git と CodeCommit の基本的な理解

機能制限

このパターンは SageMaker AI Studio Classic にのみ適用され、Amazon SageMaker AI の RStudio には適用されません。

アーキテクチャ

テクノロジースタック

- Amazon SageMaker AI
- Amazon SageMaker AI Studio Classic
- AWS CodeCommit
- AWS Identity and Access Management (IAM)
- Git

ターゲット アーキテクチャ

次の図は、アカウント A からアカウント B の SageMaker AI Studio Classic に CodeCommit リポジトリを関連付けるアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーは、アカウント B `MyCrossAccountRepositoryContributorRole`の SageMaker AI Studio Classic で SageMaker AI 実行ロールを使用しながら、`sts:AssumeRole`ロールを通じてアカウント A のロールを引き受けます。引き受けたロールには、指定されたリポジトリのクローンを作成して操作するための CodeCommit アクセス許可が含まれます。
2. ユーザーは SageMaker AI Studio Classic のシステムターミナルから Git コマンドを実行します。

自動化とスケール

このパターンは、AWS Cloud Development Kit (AWS CDK)、、AWS CloudFormationまたは Terraform を使用して自動化できる手動ステップで構成されます。

ツール

AWS ツール

- Amazon SageMaker AI は、ML モデルを構築してトレーニングし、本番環境に対応したホスト環境にデプロイするのに役立つマネージド機械学習 (ML) サービスです。
- Amazon SageMaker AI Studio Classic は、機械学習モデルの構築、トレーニング、デバッグ、デプロイ、モニタリングを可能にする、機械学習用のウェブベースの統合開発環境 (IDE) です。

- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。

  注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。

その他のツール

- 「Git」は、ソフトウェア開発中のソースコードの変更を追跡するための分散型バージョン管理システムです。

エピック

アカウント A で IAM ポリシーと IAM ロールを作成

タスク	説明	必要なスキル
Account A でリポジトリアクセスの IAM ポリシーを作成します。	1. にサインイン AWS Management Console し、IAM コンソールを開きます。 2. ナビゲーションペインで、[Policies] を選択し、次に [Create policy] を選択します。 3. [JSON] タブを選択します。 4. このパターンの「追加情報」セクションにある Example IAM policy からポリシーステートメントをコピーし、そのステートメントを JSON エディターに貼り付けます。ポリシー内のすべてのプレースホルダー	AWS DevOps

タスク	説明	必要なスキル
	値を必ず置き換えてください。 5. 次へ:タグを選択し、次へ:レビューを選択します。 6. [名前] にポリシーの名前を入力します。注:このパターンでは IAM ポリシーは「`CrossAccountAccessForMySharedDemoRepo`」と呼ばれますが、任意のポリシー名を選択できます。 7. [Create policy] を選択します。 Tip ベストプラクティスとして、IAM ポリシーの範囲をユースケースに必要な最小限のアクセス許可に制限します。	

タスク	説明	必要なスキル
Account A でリポジトリアクセスの IAM ロールを作成します。	1. IAM コンソールのナビゲーションペインで、ロール、ロールを作成の順に選択します。 2. 信頼できるエンティティタイプには、AWS アカウントを選択します。 3. AWS アカウントセクションで、別の AWS アカウントを選択します。 4. アカウント ID に対して、アカウント B のアカウント ID を入力します。 5. アクセス許可を追加ページで、以前に作成された `CrossAccountAccessForMySharedDemoRepo` ポリシーを検索し、選択します。 6. [Next (次へ)] を選択します。 7. [Role name] (ロール名) に名前を入力します。注:このパターンでは IAM ロール名は `MyCrossAccountRepositoryContributorRole` と呼ばれますが、好きなロール名を選択できます。 8. ロールの作成を選択し、新しいロールの Amazon リソースネーム (ARN) をコピーします。	AWS DevOps

アカウント B で IAM インラインポリシーを作成します。

タスク	説明	必要なスキル
アカウント B の SageMaker ドメインユーザーにアタッチされている実行ロールにインラインポリシーをアタッチします。	1. IAM コンソールのナビゲーションペインでロールを選択します。 2. アカウント B の SageMaker AI ドメインユーザーにアタッチされている実行ロールを検索して選択します。 3. アクセス許可を追加、インラインポリシーを作成の順に選択します。 4. [JSON] タブを選択します。 5. 次のポリシーステートメントをコピーして、 JSON エディタに貼り付けます。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "VisualEditor0", "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "arn:aws:iam::<Account_A_ID>:role/<Account_A_Role_Name>" } ] }`	AWS DevOps

タスク	説明	必要なスキル
	6. アカウント A のアカウント ID <Account_A_Role_Name> に置き換え、前に作成した IAM ロールの名前 <Account_A_ID> に置き換えます。 7. [ポリシーの確認] を選択します。 8. 名前に対して、インラインポリシーの名前を入力します。 9. [Create policy] を選択します。	

アカウント B の SageMaker AI Studio Classic でリポジトリのクローンを作成する

タスク	説明	必要なスキル
アカウント B の SageMaker AI Studio Classic でシェルスクリプトを作成します。	1. 「SageMaker コンソール」 のナビゲーションペインで、 Studio を選択します。 2. ユーザープロファイルを選択し、Open Studio を選択します。 3. ホームセクションでランチャーを開くを選択します。 4. ユーティリティとファイルセクションで、テキストファイルを選択します。 5. このパターンの「追加情報」セクションにある	AWS DevOps

タスク	説明	必要なスキル
	Example SageMaker シェルスクリプトからスクリプトをコピーし、そのステートメントを新しいファイルに貼り付けます。スクリプトのすべてのプレースホルダー値を置き換えます。 6. 新しいファイルの untitled.txt タブを右クリックし、テキストの名前を変更を選択します。新しい名前に cross_account_git_clone.sh と入力し、名前の変更を選択します。	

タスク	説明	必要なスキル
システムターミナルからシェルスクリプトを呼び出します。	1. SageMaker コンソールのホームセクションで、「ランチャーを開く」を選択します。 2. ユーティリティとファイルセクションで、システムターミナルを選択します。 3. ターミナルで、以下のコマンドを実行します。 `chmod u+x ./cross_account_git_clone.sh && ./cross_account_git_clone.sh` SageMaker AI Studio クロスアカウントで CodeCommit リポジトリをクローンしました。システムターミナルからすべての Git コマンドを実行できるようになりました。	AWS DevOps

追加情報

IAM ポリシーの例

この例のポリシーを使用するには、次を行います。

- をリポジトリ AWS リージョン の <CodeCommit_Repository_Region>に置き換えます。
- アカウント A のアカウント ID で <Account_A_ID> を置き換えます。
- アカウント A の CodeCommit リポジトリの名前に <CodeCommit_Repository_Name> 置き換えてください。

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:BatchGet*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Describe*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:Test*",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [

 "arn:aws:codecommit:<CodeCommit_Repository_Region>:<Account_A_ID>:<CodeCommit_Repository_Name>
        ]
    }
]
}
```

SageMaker AI シェルスクリプトの例

この例のスクリプトを使用するには、次を行います。

- アカウント A のアカウント ID で <Account_A_ID> を置き換えます。
- 前に作成した IAM ロールの名前で <Account_A_Role_Name> を置き換えます。
- をリポジトリ AWS リージョン の <CodeCommit_Repository_Region>に置き換えます。
- アカウント A の CodeCommit リポジトリの名前に <CodeCommit_Repository_Name> を置き換えてください。

```
#!/usr/bin/env bash
#Launch from system terminal
```

```
pip install --quiet git-remote-codecommit

mkdir -p ~/.aws
touch ~/.aws/config

echo "[profile CrossAccountAccessProfile]
region = <CodeCommit_Repository_Region>
credential_source=EcsContainer
role_arn = arn:aws:iam::<Account_A_ID>:role/<Account_A_Role_Name>
output = json" > ~/.aws/config

echo '[credential "https://git-
codecommit.<CodeCommit_Repository_Region>.amazonaws.com"]
        helper = !aws codecommit credential-helper $@ --profile
 CrossAccountAccessProfile
        UseHttpPath = true' > ~/.gitconfig

git clone codecommit::<CodeCommit_Repository_Region>://
CrossAccountAccessProfile@<CodeCommit_Repository_Name>
```

Amazon Textract を使用して PDF ファイルからコンテンツを自動的に抽出する

作成者: Tianxia Jia (AWS)

概要

多くの組織は、ビジネスアプリケーションにアップロードされた PDF ファイルから情報を抽出する必要があります。例えば、組織は税務分析や医療請求処理のために、税務または医療用 PDF ファイルから情報を正確に抽出する必要があるかもしれません。

Amazon Web Services (AWS) クラウドでは、Amazon Textract が PDF ファイルから情報 (印刷されたテキスト、フォーム、表など) を自動的に抽出し、元の PDF ファイルからの情報を含む JSON 形式のファイルを生成します。Amazon Textract は、AWS マネジメントコンソールまたは API コールを実装して使用できます。大量の PDF ファイルをスケーリングして自動的に処理するには、プログラムによる API 呼び出しを使用することをお勧めします。

Amazon Textract がファイルを処理すると、ページ、テキストの行と単語、フォーム (キーと値のペア)、テーブルとセル、選択要素の `Block` オブジェクトリストが作成されます。バウンディングボックス、信頼区間、ID、関係など、その他のオブジェクト情報も含まれます。Amazon Textract はコンテンツ情報を文字列として抽出します。データ値はダウンストリームアプリケーションでより簡単に使用できるため、正しく識別され変換されたデータ値が必要です。

このパターンは、Amazon Textract を使用して PDF ファイルからコンテンツを自動的に抽出し、クリーンな出力に処理するためのステップバイステップのワークフローを示しています。このパターンでは、テンプレートマッチング技術を使用して必要なフィールド、キー名、テーブルを正しく識別し、各データタイプに後処理による修正を適用します。このパターンを使用してさまざまな種類の PDF ファイルを処理し、このワークフローをスケーリングおよび自動化して同じ形式の PDF ファイルを処理できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon Textract で処理するために JPEG 形式に変換された PDF ファイルを保存するための既存の Amazon Simple Storage Service (Amazon S3) バケット。S3 バケットの詳細については、Amazon S3 ドキュメントの「バケットの概要」を参照してください。

- `Textract_PostProcessing.ipynb` Jupyter Notebook (添付)、インストール、設定。Jupyter Notebook の詳細については、Amazon SageMaker ドキュメントの「Jupyter Notebook の作成」を参照してください。
- 同じ形式の既存の PDF ファイル。
- Python について理解していること。

制限

- PDF ファイルは高品質で、読みやすいものでなければなりません。ネイティブ PDF ファイルを使用することをお勧めしますが、個々の単語がすべて鮮明であれば、スキャンして PDF 形式に変換した文書も使用できます。詳細については、AWS 機械学習のブログの「PDF document pre-processing with Amazon Textract: Visuals detection and removal」を参照してください。
- 複数ページのファイルの場合は、非同期操作を使用するか、PDF ファイルを 1 ページに分割して同期操作を使用できます。この 2 つのオプションの詳細については、Amazon Textract ドキュメントの「Detecting and analyzing text in multipage documents」と「Detecting and analyzing text in single-page documents」を参照してください。

アーキテクチャ

このパターンのワークフローは、最初にサンプル PDF ファイルで Amazon Textract を実行し (初回実行)、次に最初の PDF と同じ形式の PDF ファイルに対して実行します (繰り返し実行)。次の図は、同じ形式の PDF ファイルからコンテンツを自動的かつ繰り返し抽出する、初回実行と繰り返し実行を組み合わせたワークフローを示しています。

この図は、このパターンの次のワークフローを示しています。

1. PDF ファイルを JPEG 形式に変換し、S3 バケットに保存します。
2. Amazon Textract API を呼び出し、Amazon Textract レスポンス JSON ファイルを解析します。
3. JSON ファイルを編集して、各必須フィールドに正しい `KeyName:DataType` ペアを追加します。繰り返し実行ステージ用の `TemplateJSON` ファイルを作成します。
4. データタイプ (浮動小数点、整数、日付など) ごとに後処理補正関数を定義します。
5. 最初の PDF ファイルと同じ形式の PDF ファイルを準備します。
6. Amazon抽出API を呼び出し、Amazon Textract レスポンス JSON を解析します。
7. 解析した JSON ファイルを `TemplateJSON` ファイルと照合します。

8. 後処理による修正を実装します。

最終的な JSON 出力ファイルには、必須フィールドごとに正しい `KeyName` および `Value` が含まれています。

ターゲットテクノロジースタック

- Amazon SageMaker
- Amazon S3
- Amazon Textract

自動化とスケール

新しい PDF ファイルが Amazon S3 に追加されたときに Amazon Textract を開始する AWS Lambda 関数を使用することにより、繰り返し実行ワークフローを自動化できます。次に Amazon Textract が処理スクリプトを実行し、最終出力を保存場所に保存できます。詳細については、Lambda ドキュメントの Amazon S3 トリガーを使用して Lambda 関数を呼び出すを参照してください。

ツール

- Amazon SageMaker は、ML モデルを迅速かつ簡単に構築およびトレーニングし、本番環境に対応したホスト環境に直接デプロイするのに役立つフルマネージド ML サービスです。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Textract を使用すると、ドキュメントテキストの検出と分析をアプリケーションに簡単に追加できます。

エピック

初回の実行

タスク	説明	必要なスキル
PDF ファイルを変換します。	PDF ファイルを 1 ページに分割し、Amazon Textract 同期オペレーション (Syn API) 用	データサイエンティスト/デベロッパー

タスク	説明	必要なスキル
	に JPEG 形式に変換して、初回実行に備えて準備します。 **Note** マルチページ PDF ファイルには、Amazon Textract 非同期オペレーション (Asyn API) を使用することもできます。	

タスク	説明	必要なスキル

Amazon Textract レスポンス JSON を解析します。	`Textract_PostProcessing.ipynb` Jupyter Notebook (添付ファイル) を開き、次のコードを使用して Amazon Textract API を呼び出します。
	`response = textract.analyze_document(Document={ 'S3Object': { 'Bucket': BUCKET, 'Name': '{}'.format(filename) } }, FeatureTypes=["TABLES", "FORMS"])`
	次のコードを使用して、応答 JSON を解析してフォームとテーブルにします。
	`parseformKV=form_kv_from_JSON(response) parseformTables=get_tables_fromJSON(response)`
		データサイエンティスト/デベロッパー

タスク	説明	必要なスキル
TemplateJSON ファイルを編集します。	各 KeyName および対応する DataType (文字列、浮動小数点数、整数、日付など)、およびテーブルヘッダー (ColumnNames および RowNames など) の解析された JSON を編集します。 このテンプレートは個々の PDF ファイルタイプに使用されるため、同じ形式の PDF ファイルでもテンプレートを再利用できます。	データサイエンティスト/デベロッパー
後処理補正関数を定義します。	TemplateJSON ファイルに対する Amazon Textract のレスポンス内の値は文字列です。日付、浮動小数点数、整数、通貨には区別がありません。これらの値は、ダウンストリームのユースケースに適したデータ型に変換する必要があります。 次のコードを使用して、TemplateJSON ファイルに従って各データ型を修正します。 `finalJSON=postprocessingCorrection(parsedJSON,templateJSON)`	データサイエンティスト/デベロッパー

繰り返し実行

タスク	説明	必要なスキル
PDF ファイルを準備します。	PDF ファイルを 1 ページに分割し、Amazon Textract 同期オペレーション (Syn API) 用に JPEG 形式に変換して準備します。 **Note** マルチページ PDF ファイルには、Amazon Textract 非同期オペレーション (Asyn API) を使用することもできます。	データサイエンティスト/デベロッパー
Amazon Textract API を呼び出す	以下のコードを使用して Amazon Textract API を呼び出します。 `response = textract.analyze_document( Document={ 'S3Object': { 'Bucket': BUCKET, 'Name': '{}'.format(filename) } }, FeatureTypes=["TABLES", "FORMS"])`	データサイエンティスト/デベロッパー

タスク	説明	必要なスキル

Amazon Textract レスポンス JSON を解析します。	次のコードを使用して、応答 JSON を解析してフォームとテーブルにします。
	
	`parseformKV=form_kv_from_JSON(response)`
	`parseformTables=get_tables_fromJSON(response)`
		データサイエンティスト/デベロッパー

TemplateJSON ファイルをロードし、解析された JSON と照合します。	TemplateJSON ファイルを使用して、以下のコマンドを使用して正しいキーと値のペアとテーブルを抽出します。
	
	`form_kv_corrected=form_kv_correction(parseformKV,templateJSON)`
	`form_table_corrected=form_Table_correction(parseformTables, templateJSON)`
	`form_kv_table_corrected_final={**form_kv_corrected , **form_table_corrected}`
		データサイエンティスト/デベロッパー

タスク	説明	必要なスキル
後処理修正。	次のコードを使用して、TemplateJSON ファイルと後処理関数で DataType を使用してデータを修正します。 `finalJSON=postprocessingCorrection(form_kv_table_corrected_final,templateJSON)`	データサイエンティスト/デベロッパー

関連リソース

- Automatically extract text and structured data from documents with Amazon Textract
- Extract text and structured data with Amazon Textract
- Amazon Textract resources

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon SageMaker AI Studio Lab の時系列に DeepAR を使用してコールドスタート予測モデルを構築する

作成者: Ivan Cui (AWS) と Eyal Shacham (AWS)

概要

ウェブトラフィックのリソースをより効率的に割り当てる場合でも、スタッフのニーズに対する患者の需要を予測する場合でも、会社の製品の売上を予測する場合でも、予測は不可欠なツールです。コールドスタート予測は、小売市場に入ったばかりの新製品など、履歴データが少ない時系列の予測を構築します。このパターンでは、Amazon SageMaker AI DeepAR 予測アルゴリズムを使用してコールドスタート予測モデルをトレーニングし、コールドスタート項目に対して予測を実行する方法を示します。

DeepAR は、再帰型ニューラルネットワーク (RNN) を使用してスカラー (1 次元) 時系列を予測するための教師あり学習アルゴリズムです。DeepAR は、関連製品のすべての時系列にわたって 1 つのモデルを共同でトレーニングするアプローチを採用しています。

自己回帰統合移動平均 (ARIMA) や指数平滑法 (ETS) などの従来の時系列予測方法は、個々の製品の過去の時系列に大きく依存します。したがって、これらの方法はコールドスタート予測には効果的ではありません。データセットに何百もの関連する時系列データが含まれている場合、DeepAR は標準の ARIMA や ETS メソッドよりも優れています。トレーニング済みモデルを使用して、トレーニング済み時系列に似た新しい時系列の予測を生成することもできます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Amazon SageMaker AI ドメイン。
- Amazon SageMaker AI Studio Lab または Jupiter ラボアプリケーション。
- 読み取りおよび書き込みアクセス許可を持つ Amazon Simple Storage Service (Amazon S3) バケット。
- Python でのプログラミングに関する知識。
- Jupyter ノートブックの使用に関する知識。

制約事項

- 履歴データポイントなしで予測モデルを呼び出すと、エラーが返されます。最小限の履歴データポイントでモデルを呼び出すと、信頼度の高い不正確な予測が返されます。このパターンは、コールドスタート予測のこれらの既知の制限を解決するためのアプローチを提案します。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Python バージョン 3.10 以降。
- このパターンのノートブックは、Python 3 (データサイエンス) カーネルを搭載した ml.t3.medium インスタンスで Amazon SageMaker AI Studio でテストされました。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

ワークフローは次のタスクを実行します。

1. トレーニングデータとテストデータの入力ファイルは合成され、Amazon S3 バケットにアップロードされます。このデータには、ターゲット値 (予測対象) とともに、カテゴリ別特徴と動的特徴を持つ複数の時系列が含まれます。Jupyter ノートブックは、トレーニングデータの要件と予想される予測値をよりよく理解するためにデータを視覚化します。
2. ハイパーパラメータチューナージョブは、モデルをトレーニングし、事前定義されたメトリクスに基づいて最適なモデルを見つけるために作成されます。
3. 入力ファイルは、Amazon S3 バケットからハイパーパラメータ調整ジョブの各インスタンスにダウンロードされます。
4. チューナージョブがチューナーの事前定義されたしきい値に基づいて最適なモデルを選択すると、モデルは SageMaker AI エンドポイントとしてデプロイされます。
5. その後、デプロイされたモデルを呼び出す準備が整い、その予測がテストデータに対して検証されます。

このノートブックでは、十分な数の履歴データポイントが利用可能な場合に、モデルがターゲット値をどの程度予測できるかを示します。ただし、履歴データポイント (コールド積を表す) が少ないモデルを呼び出すと、モデルの予測は、モデルの信頼レベル内であっても元のテストデータと一致しない。このパターンでは、新しいモデルがコールド製品用に構築され、最初のコンテキストの長さ (予測ポイント) が使用可能な履歴ポイントの量として定義され、新しいモデルが新しいデータポイントが取得されるたびに繰り返しトレーニングされます。このノートブックは、履歴データポイントの量がコンテキストの長さに近づいている限り、モデルが正確な予測を持つことを示しています。

ツール

AWS のサービス

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon SageMaker AI は、ML モデルを構築してトレーニングし、本番環境に対応したホスト環境にデプロイするのに役立つマネージド機械学習 (ML) サービスです。
- Amazon SageMaker AI Studio は、ML モデルの構築、トレーニング、デバッグ、デプロイ、モニタリングを可能にする ML 用のウェブベースの統合開発環境 (IDE) です。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub DeepAR-ColdProduct-Pattern リポジトリで入手できます。

ベストプラクティス

- 仮想環境でモデルをトレーニングし、常にバージョン管理を使用して再現性を最大限に高めます。
- 最高の予測モデルを得るために、可能な限り多くの高品質のカテゴリ別特徴を含めます。
- モデルがコールドスタート製品の予測を適切に推測できるように、メタデータに同様のカテゴリ項目が含まれていることを確認してください。
- ハイパーパラメータ調整ジョブを実行して、最も高い予測モデルを取得します。

- このパターンでは、構築するモデルのコンテキストの長さは 24 時間です。つまり、次の 24 時間は予測されます。履歴データが 24 時間未満の次の 24 時間を予測しようとすると、モデルの予測精度は履歴データポイントの量に基づいて直線的に低下します。この問題を軽減するには、この数が目的の予測 (コンテキスト) の長さに達するまで、履歴データポイントのセットごとに新しいモデルを作成します。たとえば、コンテキスト長モデルを 2 時間から始め、モデルを 4 時間、8 時間、16 時間、24 時間に徐々に増やします。

エピック

SageMaker AI Studio Classic アプリケーションを起動する

タスク	説明	必要なスキル
ノートブック環境を起動します。	1. にサインインし AWS Management Console、SageMaker AI Studio ホームページを開きます。次に、Open Studio を選択します。 2. 左側のナビゲーションペインで、 アプリケーションの Studio Classic アイコンを選択します。次に、アプリケーションリストの Open ボタンを選択します。 詳細については、Amazon SageMakerドキュメントの「Amazon SageMaker AI Studio を起動する」を参照してください。 SageMaker	データサイエンティスト

ノートブックを作成してアクティブ化する

タスク	説明	必要なスキル
モデルトレーニング用の仮想環境を設定します。	モデルトレーニング用に仮想環境を設定するには、以下を実行します。 1. このパターンの GitHub リポジトリからローカルマシンに`deepar_synthetic.ipynb` ノートブックをダウンロードします。 2. Amazon SageMaker AI Studio Classic で、Studio Classic メニューバーからファイルのアップロードアイコンを選択し、ダウンロードしたノートブックを選択します。 3. 左側のナビゲーションペインのファイルブラウザでノートブックを選択します。プロンプトに従ってノートブック環境を設定します。データサイエンス 3.0 イメージと Python 3 カーネルを選択します。 詳細については、SageMaker AI ドキュメントの「SageMaker AI Studio Classic へのファイルのアップロード」を参照してください。SageMaker	データサイエンティスト

タスク	説明	必要なスキル
予測モデルを作成して検証します。	• ノートブックの指示に従って、トレーニングデータとテストデータを作成し、モデルをトレーニングしてから、モデルを呼び出します。 • 適切な履歴データポイントが提供されるときのモデルの予測の正確性を確認します。	データサイエンティスト

関連リソース

- DeepAR ハイパーパラメータ
- AWS 機械学習サービスを使用した新製品導入の需要の予測
- Amazon SageMaker AI Studio Classic を起動する
- SageMaker AI DeepAR 予測アルゴリズムを使用する

Amazon SageMaker AI と Azure DevOps を使用して MLOps ワークフローを構築する

作成者: Deepika Kumar (AWS)、Hiplate Kokoh Prasetyo (AWS)、Sara van de Moosdijk (AWS)

概要

機械学習オペレーション (MLOps) は、機械学習 (ML) ワークフローとデプロイを自動化および簡素化する一連のプラクティスです。MLOps は、ML ライフサイクルの自動化に焦点を当てています。これにより、モデルが開発されるだけでなく、体系的かつ繰り返しデプロイ、モニタリング、再トレーニングが行われるようになります。DevOps の原則を ML にもたらします。MLOps は、ML モデルのデプロイを高速化し、時間の経過とともに精度を向上させ、真のビジネス価値を提供するという保証を強化します。

多くの場合、組織は MLOps ジャーニーを開始する前に、既存の DevOps ツールとデータストレージソリューションを用意しています。このパターンは、Microsoft Azure と の両方の長所を活用する方法を示しています AWS。Azure DevOps を Amazon SageMaker AI と統合して MLOps ワークフローを作成するのに役立ちます。

このソリューションは、Azure と 間の作業を簡素化します AWS。Azure は、開発および機械学習 AWS に使用できます。これにより、データ処理、トレーニング、デプロイなど、機械学習モデルを最初から最後まで作成するための効果的なプロセスが促進されます AWS。効率を高めるために、Azure DevOps パイプラインを通じてこれらのプロセスを管理します。このソリューションは、ファインチューニング、ベクトルデータベース、プロンプト管理を含む生成 AI の基盤モデルオペレーション (FMOps) と大規模言語モデルオペレーション (LLMOps) に適用されます。

前提条件と制限

前提条件

- Azure サブスクリプション – 継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを設定するための Azure DevOps などの Azure サービスへのアクセス。
- アクティブな AWS アカウント – このパターン AWS のサービス で使用される を使用するアクセス許可。
- データ – 機械学習モデルをトレーニングするための履歴データへのアクセス。
- ML の概念に関する知識 — Python、Jupyter Notebooks、機械学習モデル開発の理解。
- セキュリティ設定 – Azure と AWS の両方にわたるロール、ポリシー、アクセス許可を適切に設定して、安全なデータ転送とアクセスを確保します。

- （オプション) ベクトルデータベース – ベクトルデータベースに検索拡張生成 (RAG) アプローチとサードパーティーサービスを使用している場合は、外部ベクトルデータベースにアクセスする必要があります。

制約事項

- このガイダンスでは、安全なクラウド間データ転送については説明していません。クラウド間のデータ転送の詳細については、AWS「ハイブリッドおよびマルチクラウドのソリューション」を参照してください。
- マルチクラウドソリューションは、リアルタイムデータ処理とモデル推論のレイテンシーを高める可能性があります。
- このガイダンスでは、マルチアカウント MLOps アーキテクチャの例を 1 つ示します。機械学習と AWS 戦略に基づいて調整する必要があります。
- このガイダンスでは、Amazon SageMaker AI 以外の AI/ML サービスの使用については説明していません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ターゲット アーキテクチャ

ターゲットアーキテクチャは Azure DevOps を Amazon SageMaker AI と統合し、クロスクラウド ML ワークフローを作成します。これは、CI/CD プロセスには Azure を使用し、ML モデルのトレーニングとデプロイには SageMaker AI を使用します。モデルの構築とデプロイを通じて (Amazon S3、Snowflake、Azure Data Lake などのソースから) データを取得するプロセスの概要を説明します。主なコンポーネントには、モデルの構築とデプロイ、データ準備、インフラストラクチャ管理のための CI/CD パイプライン、ML モデルのトレーニングと微調整、評価、デプロイのための Amazon SageMaker AI などがあります。このアーキテクチャは、クラウドプラットフォーム全体で効率的で自動化されたスケーラブルな ML ワークフローを提供するように設計されています。

アーキテクチャは以下のコンポーネントで構成されます。

1. データサイエンティストは、開発アカウントで ML 実験を実行して、さまざまなデータソースを使用して ML ユースケースのさまざまなアプローチを調べます。データサイエンティストはユニットテストとトライアルを実行し、実験を追跡するために、MLflow で Amazon SageMaker AI を使用できます。生成 AI モデル開発では、データサイエンティストは Amazon SageMaker AI JumpStart モデルハブから基盤モデルを微調整します。モデル評価の後、データサイエンティストはコードをプッシュして、Azure DevOps でホストされている Model Build リポジトリにマージします。このリポジトリには、複数ステップのモデル構築パイプラインのコードが含まれています。
2. Azure DevOps では、継続的インテグレーション (CI) を提供する Model Build パイプラインは、メインブランチへのコードマージ時に自動または手動でアクティブ化できます。自動化アカウントでは、これにより、データ前処理、モデルトレーニングとファインチューニング、モデル評価、精度に基づく条件付きモデル登録のための SageMaker AI パイプラインがアクティブ化されます。
3. Automation アカウントは、ML 環境 (Amazon ECR)、モデル (Amazon S3)、モデルメタデータ (SageMaker AI Model Registry)、機能 (SageMaker AI Feature Store)、自動パイプライン (SageMaker AI Pipelines)、ML ログインサイト (CloudWatch) をホストする ML プラットフォーム全体の中央アカウントです。生成 AI ワークロードの場合、ダウンストリームアプリケーションのプロンプトに対して追加の評価が必要になる場合があります。プロンプト管理アプリケーションは、プロセスの合理化と自動化に役立ちます。このアカウントは、ML アセットの再利用を可能にし、ベストプラクティスを適用して ML ユースケースの配信を高速化します。
4. 最新のモデルバージョンが SageMaker AI Model Registry に追加され、レビューされます。モデルバージョンとそれぞれのアーティファクト (系統とメタデータ) を追跡します。また、モデルのステータス (承認、拒否、または保留中) を管理し、ダウンストリームデプロイのバージョンを管理します。
5. Model Registry のトレーニング済みモデルがスタジオインターフェイスまたは API コールを通じて承認されると、イベントを Amazon EventBridge にディスパッチできます。EventBridge は Azure DevOps でモデルデプロイパイプラインを開始します。
6. 継続的デプロイ (CD) を提供する Model Deploy パイプラインは、Model Deploy リポジトリからソースをチェックします。ソースには、コード、モデルデプロイの設定、品質ベンチマークのテストスクリプトが含まれています。Model Deploy パイプラインは、推論タイプに合わせて調整できます。
7. 品質管理チェック後、Model Deploy パイプラインはモデルをステージングアカウントにデプロイします。ステージングアカウントは本番稼働用アカウントのコピーであり、統合テストと評価に使用されます。バッチ変換の場合、Model Deploy パイプラインは、承認された最新のモデルバー

ジョンを使用するようにバッチ推論プロセスを自動的に更新できます。リアルタイム、サーバーレス、または非同期推論の場合、それぞれのモデルエンドポイントを設定または更新します。

8. ステージングアカウントでのテストが成功すると、モデルデプロイパイプラインを通じて手動で承認することで、モデルを本番稼働用アカウントにデプロイできます。このパイプラインは、モデルモニタリングやデータフィードバックメカニズムなど、本番環境へのデプロイステップで本番環境エンドポイントをプロビジョニングします。
9. モデルが本番稼働状態になったら、SageMaker AI Model Monitor や SageMaker AI Clarify などのツールを使用してバイアスを特定し、ドリフトを検出して、モデルのパフォーマンスを継続的にモニタリングします。

自動化とスケール

Infrastructure as Code (IaC) を使用して、自動的に複数のアカウントと環境にデプロイします。MLOps ワークフローの設定プロセスを自動化することで、さまざまなプロジェクトに取り組む ML チームが使用する環境を分離できます。 は、インフラストラクチャをコードとして扱うことで、AWS リソースのモデル化、プロビジョニング、管理AWS CloudFormationを支援します。

ツール

AWS のサービス

- Amazon SageMaker AI は、ML モデルを構築してトレーニングし、本番環境に対応したホスト環境にデプロイするのに役立つマネージド ML サービスです。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。このパターンでは、Amazon S3 はデータストレージに使用され、モデルトレーニングとモデルオブジェクトの SageMaker AI と統合されています。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。このパターンでは、Lambda はデータの前処理タスクと後処理タスクに使用されます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。このパターンでは、SageMaker AI がトレーニング環境とデプロイ環境として使用する Docker コンテナを保存します。

- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。このパターンでは、EventBridge は、自動モデル再トレーニングまたはデプロイを開始するイベント駆動型または時間ベースのワークフローを調整します。
- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。 このパターンでは、SageMaker AI エンドポイントの外部向け単一エントリポイントを作成するために使用されます。
- RAG アプリケーションの場合、Amazon OpenSearch Service や Amazon RDS for PostgreSQL AWS のサービスなどの を使用して、LLM に内部データを提供するベクトル埋め込みを保存できます。

その他のツール

- Azure DevOps は、CI/CD パイプラインを管理し、コードの構築、テスト、デプロイを容易にするのに役立ちます。
- Azure Data Lake Storage または Snowflake は、ML モデルのトレーニングデータのサードパーティーソースである可能性があります。
- Pinecone、Milvus、または ChromaDB は、ベクトル埋め込みを保存できるサードパーティーのベクトルデータベースです。

ベストプラクティス

このマルチクラウド MLOps ワークフローのコンポーネントを実装する前に、以下のアクティビティを完了してください。

- 機械学習ワークフローとそのサポートに必要なツールを定義して理解します。ユースケースごとに異なるワークフローとコンポーネントが必要です。例えば、特徴量ストアは、パーソナライゼーションのユースケースで特徴量の再利用や低レイテンシーの推論に必要になる場合がありますが、他のユースケースでは必要ではない場合があります。アーキテクチャを正常にカスタマイズするには、データサイエンスチームのターゲットワークフロー、ユースケース要件、優先されるコラボレーション方法を理解する必要があります。
- アーキテクチャの各コンポーネントの責任を明確に分離します。Azure Data Lake Storage、Snowflake、Amazon S3 にデータストレージを分散すると、複雑さとコストが増大する可能性があります。可能であれば、一貫したストレージメカニズムを選択します。同様に、Azure と AWS DevOps サービスの組み合わせ、または Azure と AWS ML サービスの組み合わせを使用しないでください。

- MLOps ワークフローのend-to-endのテストを実行するには、既存のモデルとデータセットを 1 つ以上選択します。テストアーティファクトには、プラットフォームが本番稼働を開始するときにデータサイエンスチームが開発する実際のユースケースを反映する必要があります。

エピック

MLOps アーキテクチャの設計

タスク	説明	必要なスキル
データソースを特定します。	現在および将来のユースケース、利用可能なデータソース、およびデータの種類 (機密データなど) に基づいて、MLOps プラットフォームと統合する必要があるデータソースを文書化します。データは、Amazon S3、Azure Data Lake Storage、Snowflake、またはその他のソースに保存できます。生成 AI ワークロードの場合、データには生成されたレスポンスを根拠とするナレッジベースが含まれる場合もあります。このデータは、ベクトルデータベースにベクトル埋め込みとして保存されます。これらのソースをプラットフォームと統合し、正しいリソースへのアクセスを保護するための計画を作成します。	データエンジニア、データサイエンティスト、クラウドアーキテクト
該当するサービスを選択します。	データサイエンスチーム、該当するデータソース、既存のクラウドアーキテクチャの望	AWS 管理者、データエンジニア、データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル
	ましいワークフローに基づいてサービスを追加または削除することで、アーキテクチャをカスタマイズします。たとえば、データエンジニアやデータサイエンティストは、SageMaker AI AWS Glue、または Amazon EMR でデータ前処理と特徴量エンジニアリングを実行する場合があります。3 つのサービスすべてが必要になる可能性はほとんどありません。	

タスク	説明	必要なスキル
セキュリティ要件を分析します。	セキュリティ要件を収集して文書化します。これには、以下を決定することが含まれます。 • 特定のデータソースにアクセスできるチームまたはエンジニア • 事前トレーニング済みの基盤モデルにアクセスできるチームまたはエンジニア • チームが他のチームのコードとモデルにアクセスできるかどうか • 非開発アカウントに対してチームメンバーに付与する必要があるアクセス許可 (存在する場合) • クラウド間データ転送に実装する必要があるセキュリティ対策 生成 AI ワークロードの保護の詳細については、「生成 AI の保護: 生成 AI セキュリティスコープマトリックスの概要」(AWS ブログ記事) を参照してください。	AWS 管理者、クラウドアーキテクト

セットアップ AWS Organizations

タスク	説明	必要なスキル
セットアップします AWS Organizations。	ルート AWS Organizations でをセットアップします AWS アカウント。これにより、マルチアカウント MLOps 戦略の一部として作成する後続のアカウントを管理できます。詳細については、AWS Organizations のドキュメントを参照してください。	AWS 管理者

開発環境とバージョニングを設定する

タスク	説明	必要なスキル
AWS 開発アカウントを作成します。	データエンジニアとデータサイエンティスト AWS アカウント が ML モデルを試して作成するためのアクセス許可を持つ を作成します。手順については、 AWS Organizations ドキュメントの「組織内のメンバーアカウントの作成」を参照してください。	AWS 管理者
Model Build リポジトリを作成します。	実験フェーズの完了後にデータサイエンティストがモデルビルドとデプロイコードをプッシュできる Azure に Git リポジトリを作成します。手順については、Azure DevOps ドキュメントの「Git リポジト	DevOps エンジニア、ML エンジニア

タスク	説明	必要なスキル
	リのセットアップ」を参照してください。	
Model Deploy リポジトリを作成します。	標準のデプロイコードとテンプレートを保存する Git リポジトリを Azure に作成します。これには、設計フェーズで特定された、組織が使用するすべてのデプロイオプションのコードを含める必要があります。例えば、リアルタイムエンドポイント、非同期エンドポイント、サーバーレス推論、バッチ変換を含める必要があります。手順については、Azure DevOps ドキュメントの「Git リポジトリのセットアップ」を参照してください。	DevOps エンジニア、ML エンジニア
Amazon ECR リポジトリを作成します。	承認された ML 環境を Docker イメージとして保存する Amazon ECR リポジトリを設定します。データサイエンティストと ML エンジニアが新しい環境を定義できるようにします。手順については、Amazon ECR ドキュメントの「プライベートリポジトリの作成」を参照してください。	ML エンジニア

タスク	説明	必要なスキル
SageMaker AI Studio をセットアップします。	以前に定義したセキュリティ要件、優先データサイエンスツール (MLflow など)、優先統合開発環境 (IDE) に従って、開発アカウントに SageMaker AI Studio を設定します。ライフサイクル設定を使用して、主要な機能のインストールを自動化し、データサイエンティストのための統一された開発環境を作成します。詳細については、SageMaker AI ドキュメントの「Amazon SageMaker AI Studio と MLflow 追跡サーバー」を参照してください。SageMaker	データサイエンティスト、ML エンジニア、プロンプトエンジニア

CI/CD パイプラインの統合

タスク	説明	必要なスキル
オートメーションアカウントを作成します。	自動パイプラインとジョブ AWS アカウント が実行される を作成します。データサイエンスチームにこのアカウントへの読み取りアクセスを許可できます。手順については、 AWS Organizations ドキュメントの「組織内のメンバーアカウントの作成」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
モデルレジストリをセットアップします。	Automation アカウントで SageMaker AI Model Registry を設定します。このレジストリは ML モデルのメタデータを保存し、特定のデータサイエンティストやチームがモデルを承認または拒否するのに役立ちます。詳細については、SageMaker AI ドキュメントの「Model Registry でモデルを登録してデプロイする」を参照してください。	ML エンジニア
Model Build パイプラインを作成します。	コードが Model Build リポジトリにプッシュされたときに手動または自動で開始する CI/CD パイプラインを Azure に作成します。パイプラインはソースコードをチェックアウトし、オートメーションアカウントで SageMaker AI パイプラインを作成または更新する必要があります。パイプラインは、モデルレジストリに新しいモデルを追加する必要があります。パイプラインの作成の詳細については、Azure Pipelines ドキュメントを参照してください。	DevOps エンジニア、ML エンジニア

デプロイスタックを構築する

タスク	説明	必要なスキル
AWS ステージングアカウントとデプロイアカウントを作成します。	ML モデルのステージングとデプロイ AWS アカウント 用に を作成します。これらのアカウントは、本番環境に移行する前にステージング中のモデルを正確にテストできるように同一である必要があります。データサイエンスチームにステージングアカウントへの読み取りアクセスを許可できます。手順については、AWS Organizations ドキュメントの「組織内のメンバーアカウントの作成」を参照してください。	AWS 管理者
モデルモニタリング用に S3 バケットを設定します。	Model Deploy パイプラインによって作成されたデプロイ済みモデルのモデルモニタリングを有効にする場合は、このステップを完了します。入出力データを保存するための Amazon S3 バケットを作成します。S3 バケットの作成の詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。自動モデルモニタリングジョブが自動化アカウントで実行されるように、クロスアカウントアクセス許可を設定します。詳細については、SageMaker AI ドキュメントの「デー	ML エンジニア

タスク	説明	必要なスキル
	タおよびモデル品質のモニタリング」を参照してください。	
Model Deploy パイプラインを作成します。	モデルがモデルレジストリで承認されたときに開始する CI/CD パイプラインを Azure に作成します。パイプラインは、ソースコードとモデルアーティファクトをチェックし、ステージングアカウントと本番稼働用アカウントにモデルをデプロイするためのインフラストラクチャテンプレートを構築し、ステージングアカウントにモデルをデプロイし、自動テストを実行し、手動承認を待機し、承認されたモデルを本番稼働用アカウントにデプロイする必要があります。パイプラインの作成の詳細については、Azure Pipelines のドキュメントを参照してください。	DevOps エンジニア、ML エンジニア

(オプション) ML 環境インフラストラクチャを自動化する

タスク	説明	必要なスキル
ビルドテンプレート AWS CDK または CloudFormation テンプレート。	自動的にデプロイする必要があるすべての環境の AWS Cloud Development Kit (AWS CDK) または AWS CloudFormation テンプレートを定義します。これには、開発環境、	AWS DevOps

タスク	説明	必要なスキル
	オートメーション環境、ステージングおよびデプロイ環境が含まれます。詳細については、AWS CDK「」およびCloudFormation ドキュメント」を参照してください。	
インフラストラクチャパイプラインを作成します。	インフラストラクチャデプロイ用の CI/CD パイプラインを Azure に作成します。管理者は、このパイプラインを開始して新しい を作成し AWS アカウント 、ML チームが必要とする環境を設定できます。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
モニタリングとドリフトの検出が不十分 – モニタリングが不十分であると、モデルのパフォーマンスの問題やデータドリフトの検出が失敗する可能性があります。	Amazon CloudWatch、SageMaker AI Model Monitor、SageMaker AI Clarify などのツールを使用してモニタリングフレームワークを強化します。特定された問題に対する即時アクションのアラートを設定します。
CI パイプライントリガーエラー– Azure DevOps の CI パイプラインは、設定ミスによりコードマージ時にトリガーされない場合があります。	Azure DevOps プロジェクト設定をチェックして、ウェブフックが正しく設定され、正しい SageMaker AI エンドポイントを指していることを確認します。
ガバナンス – 中央オートメーションアカウントは、ML プラットフォーム間でベストプラクティスを適用しない可能性があり、ワークフローに一貫性がない可能性があります。	Automation アカウント設定を監査し、すべての ML 環境とモデルが定義済みのベストプラクティスとポリシーに準拠していることを確認します。

問題	ソリューション
モデルレジストリの承認の遅延 – これは、ユーザーがモデルを確認するのに時間がかかるか、技術的な問題が原因で、モデルのチェックと承認が遅れた場合に発生します。	承認保留中のモデルをステークホルダーに警告する通知システムを実装し、レビュープロセスを合理化します。
モデルデプロイイベントの失敗 – モデルデプロイパイプラインを開始するためにディスパッチされた イベントが失敗し、デプロイが遅延する可能性があります。	Amazon EventBridge に Azure DevOps パイプラインを正常に呼び出すための正しいアクセス許可とイベントパターンがあることを確認します。
本番稼働用デプロイのボトルネック – 手動の承認プロセスではボトルネックが発生し、モデルの本番稼働用デプロイが遅れる可能性があります。	モデルデプロイパイプライン内の承認ワークフローを最適化し、タイムリーなレビューと明確なコミュニケーションチャネルを促進します。

関連リソース

AWS ドキュメント

- Amazon SageMaker AI ドキュメント
- Machine Learningレンズ (AWS Well Architected Framework)
- MLOps を成功させるための計画 (AWS 規範ガイダンス)

その他の AWS リソース

- Amazon SageMaker AI を使用する企業の MLOps 基盤ロードマップ Amazon SageMaker (AWS ブログ記事)
- AWS Summit ANZ 2022 - アーキテクト向けのEnd-to-endの MLOps (YouTube 動画)
- FMOps/LLMOps: 生成 AI の運用と MLOps との違い (AWS ブログ記事)
- Amazon SageMaker AI Clarify と MLOps サービスを使用して LLM 評価を大規模に運用する (AWS ブログ記事)
- 生成 AI アプリケーションにおけるベクトルデータベースの役割 (AWS ブログ記事)

Azure ドキュメント

- Azure DevOps ドキュメント
- Azure Pipelines のドキュメント

を使用して Amazon Bedrock でモデル呼び出し口グ記録を設定する AWS CloudFormation

作成者: Vikramaditya Bhatnagar (AWS)

概要

Amazon Bedrock を設定して、 内のすべてのモデル呼び出しの呼び出し口グ、モデル入力データ、モデル出力データを収集できます AWS アカウント。これは、Amazon Bedrock を使用して堅牢な生成 AI アプリケーションを構築するためのベストプラクティスです。モデル呼び出し口グは、Amazon CloudWatch Logs 口ググループ、Amazon Simple Storage Service (Amazon S3) バケット、またはその両方に保存できます。CloudWatch Logs に口グデータがあると、カスタムメトリクスフィルター、アラーム、ダッシュボードを作成できます。Amazon S3 は、組織のポリシーに従って、 間でデータをレプリケート AWS リージョン したり、長期保存したりするのに理想的です。

このパターンは、Infrastructure as Code (IaC) アプローチを使用して Amazon Bedrock のモデル呼び出し口グ記録を設定するサンプル AWS CloudFormation テンプレートを提供します。テンプレートは、CloudWatch Logs と Amazon S3 の両方で口グストレージを設定します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 以下のアクセス許可:
  - CloudFormation スタックを作成するアクセス許可
  - Amazon Bedrock へのアクセス許可
  - Amazon S3 バケットを作成してアクセスするためのアクセス許可
  - CloudWatch Logs 口ググループを作成してアクセスするためのアクセス許可
  - AWS Lambda 関数を作成してアクセスするためのアクセス許可
  - AWS Key Management Service （ AWS KMS) キーを作成してアクセスするためのアクセス許可

制約事項

このパターンは、CloudWatch Logs と Amazon S3 の両方にモデル呼び出しを口グに記録します。この 2 つのサービスのうち 1 つのみを選択することはできません。

アーキテクチャ

ターゲット アーキテクチャ

CloudFormation テンプレートは、ターゲットに次のリソースをプロビジョニングします AWS アカウント。

- モデル呼び出しログを保存するための CloudWatch Logs ロググループ
- モデル呼び出しログと対応するバケットポリシーを保存するための Amazon S3 バケット
- サーバー側のアクセスログと対応するバケットポリシーを保存するための Amazon S3 バケット
- Amazon Bedrock でログ記録設定を構成する AWS Lambda 関数
- AWS KMS key および対応するキーエイリアス
- Amazon Bedrock の AWS Identity and Access Management (IAM) サービスロール

次の図は、このパターンに関連付けられた CloudFormation スタックをデプロイした後の呼び出しログの保存方法を示しています。Amazon Bedrock は、基盤モデルがテキスト、イメージ、ビデオ、または埋め込みデータを配信するときにログデータを公開します。図に示すように、Amazon S3 バケットと CloudWatch Logs ロググループは で暗号化されます AWS KMS key。

この図表は、次のワークフローを示しています：

1. ユーザーは Amazon Bedrock の基盤モデルにクエリを送信します。
2. Amazon Bedrock は IAM サービスロールを引き受けます。
3. Amazon Bedrock はログデータを生成し、CloudWatch Logs ロググループと Amazon S3 バケットに保存します。
4. ユーザーがモデル呼び出しログを含む Amazon S3 バケット内のファイルを読み取る、アップロードする、または削除すると、それらのアクティビティはサーバー側のアクセスログ用に別の Amazon S3 バケットに記録されます。

自動化とスケール

このソリューションをスケールするには、CloudFormation テンプレートをスタックセットとして複数の AWS リージョン と にデプロイします AWS アカウント。詳細については、CloudFormation ド

キュメントのStackSets を使用したアカウントとリージョン間のスタックの管理」を参照してください。

ツール

AWS のサービス

- Amazon Bedrock は、主要な AI 企業と Amazon の高性能基盤モデル (FMs) を統一された API を通じて使用できるようにするフルマネージドサービスです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- Amazon CloudWatch Logs は、すべてのシステム、アプリケーション、および からのログを一元化 AWS のサービス し、それらをモニタリングして安全にアーカイブするのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。

その他のツール

- Git はオープンソースの分散バージョン管理システムです。

コードリポジトリ

このパターンのコードは、GitHub enable-bedrock-logging-using-cloudformation リポジトリで入手できます。

エピック

CloudFormation スタックを作成する

タスク	説明	必要なスキル
CloudFormation テンプレートをダウンロードします。	CloudFormation テンプレートを GitHub「リポジトリ」からダウンロードします。	クラウドアーキテクト
テンプレートをデプロイします。	ターゲットアカウントとリージョンにスタックを作成します。Parameters セクションで、テンプレートで定義されているパラメータの値を指定します。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。	クラウドアーキテクト

ソリューションをテストする

タスク	説明	必要なスキル
モデルアクセスを有効にします。	Amazon Bedrock で、基盤モデルへのアクセスを追加します。手順については、「Amazon Bedrock ドキュメント」の「Amazon Bedrock 基盤モデルへのアクセスの追加または削除」を参照してください。	クラウドアーキテクト
サンプルプロンプトを実行します。	Amazon Bedrock プレイグラウンドで、サンプルプロンプトを実行します。手順については、Amazon Bedrock ド	クラウドアーキテクト

タスク	説明	必要なスキル
	キュメントの「プレイグラウンドを使用してコンソールでレスポンスを生成する」を参照してください。	
ログ記録設定を確認します。	1. Amazon Bedrock コンソールにサインインします。 2. ナビゲーションバーで、CloudFormation スタックをデプロイした AWS リージョン を選択します。 3. 左側のナビゲーションペインの Bedrock 設定で、設定を選択します。 4. 次の点を確認します。 • モデル呼び出しのログ記録が有効になっています。 • すべてのデータ型が選択されます。 • ログ記録先では、S3 ログと CloudWatch Logs の両方が選択されます。	クラウドアーキテクト
Amazon S3 バケットを確認します。	1. S3 設定セクションで、S3 を参照を選択します。これにより、Amazon S3 コンソールでターゲットバケットが開きます。 2. 以前に実行したサンプルプロンプトにログデータが存在することを確認します。	クラウドアーキテクト

タスク	説明	必要なスキル
ロググループを確認します。	1. Amazon Bedrock コンソールの設定ページに戻ります。 2. CloudWatch Logs 設定セクションで、CloudWatch Logs ロググループの設定を確認します。ロググループ名を書き留めます。 3. CloudWatch コンソールを開きます。 4. ナビゲーションペインで、[Logs] (ログ)、[Log groups] (ロググループ) の順に選択します。 5. Amazon Bedrock がログデータを発行するロググループの名前を選択します。 6. 以前に実行したサンプルプロンプトにログデータが存在することを確認します。	クラウドアーキテクト

関連リソース

AWS ドキュメント

- Amazon S3 バケットへのアクセス (Amazon S3 ドキュメント)
- スタックの作成と管理 (CloudFormation ドキュメント)
- モデルの呼び出しをモニタリングする (Amazon Bedrock ドキュメント)
- 「ロググループとログストリームの操作」(CloudWatch Logsドキュメント)

AWS ブログ投稿

- Amazon Bedrock と Amazon CloudWatch の統合を使用した生成 AI アプリケーションのモニタリング
- Amazon Bedrock エージェントを使用して堅牢な生成 AI アプリケーションを構築するためのベストプラクティス – パート 1
- Amazon Bedrock エージェントを使用して堅牢な生成 AI アプリケーションを構築するためのベストプラクティス – パート 2

SageMaker 用のカスタム Docker コンテナイメージを作成し、AWS Step Functions のモデルトレーニングに使用する

作成者: Julia Bluszcz (AWS)、Neha Sharma (AWS)、Aubrey Oosthuizen (AWS)、Mohan Gowda Purushothama (AWS)、Mateusz Zaremba (AWS)

概要

このパターンは、Amazon SageMaker 用の Docker コンテナイメージを作成し、AWS Step Functions のトレーニングモデルに使用する方法を示しています。アルゴリズムをコンテナにパッケージ化することにより、プログラミング言語、フレームワーク、依存関係に関係なく、ほぼすべてのコードを SageMaker 環境で実行できます。

提供されている SageMaker ノートブックの例では、カスタム Docker コンテナイメージは Amazon Elastic Container Registry (Amazon ECR) に保存されます。次に、Step Functions は Amazon ECR に保存されているコンテナを使用して SageMaker の Python 処理スクリプトを実行します。次に、コンテナはモデルを Amazon Simple Storage Service (Amazon S3) にエクスポートします。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon S3 アクセス許可を持つ SageMaker の AWS Identity and Access Management (IAM) ロール Amazon S3
- Step Functions の IAM ロール
- Python に精通
- Amazon SageMaker Python SDK に精通していること
- AWS Command Line Interface (AWS CLI) に精通していること
- AWS SDK for Python (Boto3) に精通していること
- Amazon ECR に精通している
- Docker に精通していること

製品バージョン

- AWS Step Functions データサイエンス SDK バージョン 2.3.0

- Amazon SageMaker Python SDK バージョン 2.78.0

アーキテクチャ

次の図は、SageMaker 用の Docker コンテナイメージを作成し、それを Step Functions のトレーニングモデルに使用するためのワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. データサイエンティストまたは DevOps エンジニアは、Amazon SageMaker ノートブックを使用してカスタム Docker コンテナイメージを作成します。
2. データサイエンティストまたは DevOps エンジニアは、Docker コンテナイメージをプライベートレジストリにおける Amazon ECR プライベートリポジトリに保存します。
3. データサイエンティストまたは DevOps エンジニアは、Docker コンテナで、Step Functions ワークフロー内の Python SageMaker 処理ジョブを実行します。

自動化とスケール

このパターンの SageMaker ノートブックの例としては、`ml.m5.xlarge` ノートブックインスタンスタイプを使用します。ユースケースに応じて、インスタンスタイプを変更することができます。SageMaker ノートブックインスタンスタイプの詳細については、「Amazon SageMaker の料金表」を参照してください。

ツール

- 「Amazon Elastic Container Registry (Amazon ECR)」は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- 「Amazon SageMaker」 はマネージド型の機械学習 (ML) サービスで、ML モデルの構築とトレーニングを行い、それらを本番稼働環境に対応したホスティング環境にデプロイします。
- Amazon SageMaker Python SDK は、SageMaker で機械学習モデルをトレーニングおよびデプロイするためのオープンソースライブラリです。
- AWS Step Functionsは、AWS Lambda関数と他のAWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。
- AWS Step Functions Data Science Python SDK は、機械学習モデルを処理して公開する Step Functions ワークフローの作成に役立つオープンソースライブラリです。

エピック

カスタム Docker コンテナイメージを作成して Amazon ECR に保存します。

タスク	説明	必要なスキル
Amazon ECR をセットアップし、新しいプライベートレジストリを作成します。	まだ Amazon ECR をセットアップしていない場合は、Amazon ECR ユーザーガイドの「Amazon ECR でセットアップ」における指示に従ってください。各 AWS アカウントには、デフォルトのプライベート Amazon ECR レジストリが提供されます。	DevOps エンジニア
Amazon ECR プライベートリポジトリを作成します。	Amazon ECR ユーザーガイドの「プライベートリポジトリの作成」における指示に従ってください。 **Note** 作成するリポジトリは、カスタム Docker コンテナイメージを保存する場所です。	DevOps エンジニア
SageMaker 処理ジョブの実行に必要な仕様を含む Docker ファイルを作成します。	Dockerfile を設定して、SageMaker 処理ジョブの実行に必要な仕様を含む Dockerfile を作成します。手順については、Amazon SageMaker 開発者ガイドの「独自のトレーニングコンテナの調整」を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル

	Dockerfiles の詳細については、Docker ドキュメントの Dockerfile リファレンスを参照してください。
	
	例：Jupyter Notebookのコードセルで [Dockerfile] を作成
	
	セル 1
	
	`# Make docker folder`
	`!mkdir -p docker`
	
	セル 2
	
	`%%writefile docker/Dockerfile`
	
	`FROM python:3.7-slim-buster`
	
	`RUN pip3 install pandas==0.25.3 scikit-learn==0.21.3`
	`ENV PYTHONUNBUFFERED=TRUE`
	
	`ENTRYPOINT ["python3"]`	

タスク	説明	必要なスキル
Docker コンテナイメージを構築し、Amazon ECR にプッシュします。	1. AWS CLI で `docker build` コマンドを実行して作成した Dockerfile でコンテナイメージを構築します。 2. `docker push` コマンドを実行して、コンテナイメージを Amazon ECR にプッシュします。 詳細については、GitHub での独自のアルゴリズムコンテナの構築の「コンテナの構築と登録」を参照してください。 Docker イメージを構築して登録する Jupyter Notebookのコードセルの例 **Important** 次のセルを実行する前に、Dockerfile を作成し、 というディレクトリに保存していることを確認してくださいdocker。また、Amazon ECR リポジトリが作成されたことを確認し、最初のセルの `ecr_repository` 値をリポジトリの名前に置き換えてください。	DevOps エンジニア

タスク	説明	必要なスキル

	セル 1
	`import boto3`
	`tag = ':latest'`
	`account_id = boto3.client('sts').get_caller_identity().get('Account')`
	`region = boto3.Session().region_name`
	`ecr_repository = 'byoc'`
	
	`image_uri = '{}.dkr.ecr.{}.amazonaws.com/{}'.format(account_id, region, ecr_repository + tag)`
	
	セル 2
	`# Build docker image`
	`!docker build -t $image_uri docker`
	
	セル 3
	`# Authenticate to ECR`
	`!aws ecr get-login-password --region {region} | docker login --username AWS --password-stdin {account_id}.dkr.ecr.{region}.amazonaws.com`
	
	セル 4
	`# Push docker image`	

タスク	説明	必要なスキル
	`!docker push $image_uri` **Note** `docker push` および `docker pull` コマンドを使用できるように、Docker クライアントをプライベートレジストリに対して認証する必要があります。これらのコマンドは、レジストリー内のリポジトリにイメージをプッシュし、またはレジストリーのリポジトリからイメージをプルします。	

カスタム Docker コンテナイメージを使用する Step Functions ワークフローを作成

タスク	説明	必要なスキル
カスタム処理とモデルトレーニングロジックを含む Python スクリプトを作成します。	カスタム処理ロジックを書き込んでデータ処理スクリプトで実行します。その後、これを Python スクリプトとして、`training.py` という名前で保存します。 詳細については、GitHub の「SageMaker スクリプトモー	データサイエンティスト

タスク	説明	必要なスキル

	ドでモデルを自作」を参照してください。
	
	カスタム処理とモデルトレーニングロジックを含む Python スクリプトの例
	
	`%%writefile training.py`
	`from numpy import empty`
	`import pandas as pd`
	`import os`
	`from sklearn import datasets, svm`
	`from joblib import dump, load`
	
	`if __name__ == '__main__':`
	`    digits = datasets.load_digits()`
	`    #create classifier object`
	`    clf = svm.SVC(gamma=0.001, C=100.)`
	
	`    #fit the model`
	`    clf.fit(digits.data[:-1], digits.target[:-1])`
	
	`    #model output in binary format`
	`    output_path = os.path.join('/opt/ml/processing/model', "model.joblib")`
	`    dump(clf, output_path)`	

タスク	説明	必要なスキル

SageMaker 処理ジョブをステップの 1 つとして含むステップファンクションワークフローを作成します。	「AWS Step Functions データサイエンス SDK」をインストールしてインポートし、training.py ファイルを Amazon S3 にアップロードします。その後、「Amazon SageMaker Python SDK」でステップファンクションの処理ステップを定義します。
	
	⚠ Important
	AWS アカウントで Step Functions の IAM 実行ロールが作成されていることを確認します。
	
	Amazon S3 にアップロードする環境設定例とカスタムトレーニングスクリプト
	
	`!pip install stepfunct ions`
	
	`import boto3`
	`import stepfunctions`
	`import sagemaker`
	`import datetime`
	
	`from stepfunctions import steps`
	`from stepfunct ions.inputs import ExecutionInput`
		データサイエンティスト

タスク	説明	必要なスキル

	from stepfunct
	ions.steps import (
	    Chain
	)
	from stepfunct
	ions.workflow import
	 Workflow
	from sagemaker
	.processing import
	 ScriptProcessor,
	 ProcessingInput,
	 ProcessingOutput
	
	sagemaker_session =
	 sagemaker.Session()
	bucket = sagemaker
	_session.default_b
	ucket()
	role = sagemaker
	.get_execution_role()
	prefix = 'byoc-tra
	ining-model'
	
	# See prerequisites
	 section to create this
	 role
	workflow_execution_rol
	e = f"arn:aws:iam::
	{account_id}:role/Ama
	zonSageMaker-StepF
	unctionsWorkflowEx
	ecutionRole"
	
	execution_input =
	 ExecutionInput(
	    schema={
	        "Preproce
	ssingJobName": str})
	
	input_code = sagemaker
	_session.upload_data(	

タスク	説明	必要なスキル
	`"training.py", bucket=bucket, key_prefix="preprocessing.py", )` カスタム Amazon ECR イメージと Python スクリプトを使用する SageMaker 処理ステップ定義の例 **Note** 必ず `execution_input` パラメータを使用してジョブ名を指定します。パラメータの値は、ジョブを実行するたびに一意でなければなりません。また、[training.py] ファイルのコードは `input` パラメータとして `ProcessingStep` に渡されます。即ち、このコードはコンテナ内にコピーされます。`ProcessingInput` コードの宛先は、`container_entrypoint` 内の 2 番目の引数と同じです。	

タスク	説明	必要なスキル
	script_processor = ScriptProcessor(command=['python3'], image_uri=image_uri, role=role, instance_count=1, instance_type='ml.m5.xlarge') processing_step = steps.ProcessingStep("training-step", processor=script_processor, job_name=execution_input["PreprocessingJobName"], inputs=[ProcessingInput(source=input_code, destination="/opt/ml/processing/input/code", input_name="code",),], outputs=[ProcessingOutput(source='/opt/ml/processing/model',	

タスク	説明	必要なスキル

	<pre>destinati
	on="s3://{}/{}".fo
	rmat(bucket, prefix),
	output_na
	me='byoc-example')
	],
	container_entrypoi
	nt=["python3", "/opt/
	ml/processing/input/c
	ode/training.py"],
	)</pre>
	SageMaker 処理ジョブを実行するStep Functions ワークフローの例
	
	Note
	このサンプルワークフローには、完全な Step Functions ワークフローではなく、SageMaker 処理ジョブステップのみが含まれます。完全なワークフローの例については、「AWS Step Functions データサイエンス SDK ドキュメント」の「SageMaker のノートブック例」を参照してください。
	
	<pre>workflow_graph =
	Chain([processing_
	step])</pre>	

タスク	説明	必要なスキル

	<pre>workflow = Workflow(
	    name="ProcessingWo
	rkflow",
	    definition=workflo
	w_graph,
	    role=workflow_exec
	ution_role
	)
	
	workflow.create()
	# Execute workflow
	execution = workflow.
	execute(
	    inputs={
	        "Preproce
	ssingJobName":
	 str(datetime.datet
	ime.now().strftime
	("%Y%m%d%H%M-%SS")),
	  # Each pre processin
	g job (SageMaker
	 processing job)
	 requires a unique name,
	    }
	)
	execution_output =
	 execution.get_outp
	ut(wait=True)</pre>	

関連リソース

- 「データの処理」(Amazon SageMaker デベロッパーガイド)
- 「独自のトレーニングコンテナを調整」(Amazon SageMaker 開発者ガイド)

Amazon Bedrock エージェントを使用して、テキストベースのプロンプトによる Amazon EKS でのアクセスエントリコントロールの作成を自動化する

作成者: Keshav Ganesh (AWS) と Sudhanshu Saurav (AWS)

概要

複数のチームが共有 Amazon Elastic Kubernetes Service (Amazon EKS) クラスターを操作する必要がある場合、組織はアクセスコントロールとリソースプロビジョニングを管理する上で課題に直面します。Amazon EKS などのマネージド Kubernetes サービスでは、クラスターオペレーションが簡素化されています。ただし、チームのアクセスとリソースのアクセス許可を管理するための管理オーバーヘッドは、複雑で時間がかかります。

このパターンは、Amazon Bedrock エージェントが Amazon EKS クラスターアクセス管理の自動化にどのように役立つかを示しています。この自動化により、開発チームはアクセスコントロールの設定と管理を処理するのではなく、コアアプリケーション開発に集中できます。Amazon Bedrock エージェントをカスタマイズして、シンプルな自然言語プロンプトを使用してさまざまなタスクのアクションを実行できます。

AWS Lambda 関数をアクショングループとして使用することで、Amazon Bedrock エージェントはユーザーアクセスエントリの作成やアクセスポリシーの管理などのタスクを処理できます。さらに、Amazon Bedrock エージェントは、クラスターで実行されているポッドの AWS Identity and Access Management (IAM) リソースへのアクセスを許可するポッド ID の関連付けを設定できます。このソリューションを使用すると、組織はシンプルなテキストベースのプロンプトで Amazon EKS クラスター管理を合理化し、手動のオーバーヘッドを減らし、全体的な開発効率を向上させることができます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- デプロイプロセスの IAM ロールとアクセス許可を確立しました。これには、Amazon Bedrock 基盤モデル (FM) へのアクセス、Lambda 関数の作成、ターゲット全体のその他の必要なリソースへのアクセス許可が含まれます AWS アカウント。
- Amazon Bedrock FMs AWS アカウント に対してアクティブな で有効になっているアクセス: Amazon Titan Text Embeddings V2 および Anthropic Claude 3 Haiku。

- AWS Command Line Interface （ AWS CLI) バージョン 2.9.11 以降、インストールおよび設定済み。
- eksctl 0.194.0 以降がインストールされている。

制約事項

- これらの手法をスムーズに導入し、効果的に使用するために、トレーニングとドキュメントが必要になる場合があります。Amazon Bedrock、Amazon EKS、Lambda、Amazon OpenSearch Service、OpenAPI を使用すると、デベロッパーと DevOps チームにとって重要な学習曲線になります。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

このソリューションでは、次のステップを実行します。

1. ユーザーは、エージェントが処理してアクションを実行するための入力として機能するプロンプトまたはクエリを送信することで、Amazon Bedrock エージェントとやり取りします。
2. プロンプトに基づいて、Amazon Bedrock エージェントは OpenAPI スキーマをチェックして、ターゲットとする正しい API を特定します。Amazon Bedrock エージェントが正しい API コールを見つけた場合、リクエストはこれらのアクションを実装する Lambda 関数に関連付けられているアクショングループに送信されます。
3. 関連する API が見つからない場合、Amazon Bedrock エージェントは OpenSearch コレクションにクエリを実行します。OpenSearch コレクションは、Amazon EKS ユーザーガイドを含む Amazon S3 バケットから取得されたインデックス付きナレッジベースコンテンツを使用します。
4. OpenSearch コレクションは、関連するコンテキスト情報を Amazon Bedrock エージェントに返します。

5. 実行可能なリクエスト (API オペレーションに一致するリクエスト) の場合、Amazon Bedrock エージェントは Virtual Private Cloud (VPC) 内で実行され、Lambda 関数をトリガーします。
6. Lambda 関数は、Amazon EKS クラスター内のユーザー入力に基づいてアクションを実行します。
7. Lambda コードの Amazon S3 バケットには、Lambda 関数用に記述されたコードとロジックを持つアーティファクトが保存されます。

ツール

AWS のサービス

- Amazon Bedrock は、主要な AI スタートアップと Amazon からの高性能な基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- Amazon Elastic Kubernetes Service (Amazon EKS) を使用すると、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行できます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を認可するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon OpenSearch Service は、 で OpenSearch クラスターをデプロイ、運用、スケーリングするのに役立つマネージドサービスです AWS クラウド。そのコレクション機能は、データを整理し、Amazon Bedrock エージェントなどの AI アシスタントが使用できる包括的なナレッジベースを構築するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- eksctl – これは Amazon EKS で Kubernetes クラスターを作成および管理するコマンドラインユーティリティです。

コードリポジトリ

このパターンのコードは、GitHub eks-access-controls-bedrock-agent リポジトリで入手できます。

ベストプラクティス

- このパターンを実装するときは、可能な限り高いセキュリティを維持します。Amazon EKS クラスターがプライベートであり、アクセス許可が制限されており、すべてのリソースが Virtual Private Cloud (VPC) 内にあることを確認します。詳細については、Amazon EKS ドキュメントの「セキュリティのベストプラクティス」を参照してください。
- 可能な限り AWS KMS カスタマーマネージドキーを使用し、制限付きアクセス許可を付与します。
- 最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	このパターンのリポジトリのクローンを作成するには、ローカルワークステーションで次のコマンドを実行します。 `git clone https://github.com/aws-samples/eks-access-controls-bedrock-agent.git`	AWS DevOps
AWS アカウント ID を取得します。	AWS アカウント ID を取得するには、次の手順を実行します。	AWS DevOps

タスク	説明	必要なスキル
	1. クローンされたリポジトリのルートフォルダ でシェルを開きますeks-access-controls-bedrock-agent。 2. AWS アカウント ID を取得するには、クローンされたディレクトリに移動し、次のコマンドを実行します。 `AWS_ACCOUNT=$(aws sts get-caller-identity --query "Account" --output text)` このコマンドは、 AWS アカウント ID を AWS_ACCOUNT 変数に保存します。	

タスク	説明	必要なスキル

Lambda コードの S3 バケットを作成します。	このソリューションを実装するには、アーキテクチャ図に示すように、さまざまな目的に対応する 3 つの Amazon S3 バケットを作成する必要があります。S3 バケットは、Lambda コード、ナレッジベース、OpenAPI スキーマ用です。
	
	Lambda コードバケットを作成するには、次の手順を実行します。
	
	1. Lambda コード用の S3 バケットを作成するには、次のコマンドを実行します。
	
	`aws s3 mb s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} --region us-east-1`
	
	2. Lambda コード依存関係をインストールするには、次のコマンドを実行します。
	
	`cd eks-lambda`
	`npm install`
	`tsc`
	`cd .. && cd opensearch-lambda`
	`npm install`
	`tsc`
	`cd ..`
		AWS DevOps

タスク	説明	必要なスキル
	3. コードをパッケージ化し、Lambda の S3 バケットにアップロードするには、次のコマンドを実行します。 `aws cloudformation package \ --template-file eks-access-controls.yaml \ --s3-bucket bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} \ --output-template-file eks-access-controls-template.yaml \ --region us-east-1` パッケージコマンドは、以下を含む新しい CloudFormation テンプレート (`eks-access-controls-template.yaml`) を作成します。 • S3 バケットに保存されている Lambda 関数コードへの参照。 • VPC、サブネット、Amazon Bedrock エージェント、OpenSearch コレクションなど、必要なすべての AWS インフラストラクチャの定義。このテンプレートを使用して	

タスク	説明	必要なスキル
	、CloudFormation を使用して完全なソリューションをデプロイできます。	

タスク	説明	必要なスキル

ナレッジベースの S3 バケットを作成します。	ナレッジベースの Amazon S3 バケットを作成するには、次の手順を実行します。
	
	1. ナレッジベースの Amazon S3 バケットを作成するには、次のコマンドを実行します。
	
	`aws s3 mb s3://eks-knowledge-base-${AWS_ACCOUNT} --region us-east-1`
	
	2. Amazon EKS ユーザーガイドをダウンロードしてディレクトリに保存するには、次のコマンドを実行します。
	
	`mkdir dataSource`
	`cd dataSource`
	`curl https://docs.aws.amazon.com/pdfs/eks/latest/userguide/eks-ug.pdf -o eks-user-guide.pdf`
	
	3. ステップ 1 で作成した S3 バケットにユーザーガイドをアップロードするには、次のコマンドを実行します。
	
	`aws s3 cp eks-user-guide.pdf s3://eks-`
		AWS DevOps

タスク	説明	必要なスキル
	knowledge-base-${AWS_ACCOUNT} \ --region us-east-1 \ 4. ルートディレクトリに戻るには、次のコマンドを実行します。 cd ..	
OpenAPI スキーマの S3 バケットを作成します。	OpenAPI スキーマの Amazon S3 バケットを作成するには、次の手順を実行します。 1. S3 バケットを作成するには、次のコマンドを実行します。 aws s3 mb s3://eks-openapi-schema-${AWS_ACCOUNT} --region us-east-1 2. OpenAPI スキーマを S3 バケットにアップロードするには、次のコマンドを実行します。 aws s3 cp openapi-schema.yaml s3://eks-openapi-schema-${AWS_ACCOUNT} --region us-east-1	AWS DevOps

CloudFormation スタックをデプロイする

タスク	説明	必要なスキル
CloudFormation スタックをデプロイします。	CloudFormation スタックをデプロイするには、前に`eks-access-controls-template.yaml` 作成した CloudFormation テンプレートファイルを使用します。詳細については、CloudFormation ドキュメントの「CloudFormation コンソールからスタックを作成する」を参照してください。 **Note** CloudFormation テンプレートを使用して OpenSearch インデックスをプロビジョニングするには、約 10 分かかります。 スタックを作成したら、 `VPC_ID`と を書き留めます`PRIVATE_SUBNET ID` 。	AWS DevOps
Amazon EKS クラスターを作成します。	VPC 内に Amazon EKS クラスターを作成するには、次の手順を実行します。 1. `eks-config.yaml` 設定ファイルのコピーを作成し、コピーに という名	AWS DevOps

タスク	説明	必要なスキル

	前を付けますeks-deploy.yaml 。
	2. テキストエディタで eks-deploy.yaml を開きます。次に、次のプレースホルダー値をデプロイされたスタックの値に置き換えます: VPC_ID、PRIVATE_SUBNET1 、および PRIVATE_SUBNET2
	3. eksctl ユーティリティを使用してクラスターを作成するには、次のコマンドを実行します。
	`eksctl create cluster -f eks-deploy.yaml`
	Note
	このクラスター作成プロセスが完了するまでに最大 15～20 分かかる場合があります。
	4. クラスターが正常に作成されたことを確認するには、次のコマンドを実行します。
	`aws eks describe-cluster --name --query "cluster.status"`	

タスク	説明	必要なスキル
	`aws eks update-ku beconfig --name -- region kubectl get nodes` 期待される結果は次のとおりです。 • クラスターのステータスはですACTIVE。 • コマンドkubectl get nodesは、すべてのノードが Ready状態であることを示します。	

Lambda 関数と Amazon EKS クラスターを接続する

タスク	説明	必要なスキル
Amazon EKS クラスターと Lambda 関数間の接続を作成します。	Lambda 関数が Amazon EKS クラスターと通信できるようにするネットワークおよび IAM アクセス許可を設定するには、次の手順を実行します。 1. Lambda 関数にアタッチされている IAM ロールを特定するには、 を開き、 という名前の Lambda 関数 AWS Management Console を見つけますbedrock-agent-eks-access-control 。IAM ロールの	AWS DevOps

タスク	説明	必要なスキル

	Amazon リソースネーム (ARN) を書き留めます。
	2. Lambda 関数の IAM ロールの Amazon EKS クラスターにアクセスエントリを作成するには、次のコマンドを実行します。
	`aws eks create-access-entry --cluster-name eks-testing-cluster --principal-arn <principal-Role-ARN>`
	3. このロールにアクセスAmazonEKSClusterAdminPolicy 許可を割り当てるには、次のコマンドを実行します。
	`aws eks associate-access-policy --cluster-name eks-testing-cluster --principal-arn <principal-Role-ARN> --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy --access-scope type=cluster`
	詳細については、Amazon EKS ドキュメントの「アクセスポリシーをアクセ	

タスク	説明	必要なスキル
	スエントリに関連付ける」および「AmazonEKSClusterAdminPolicy」を参照してください。 AmazonEKSClusterAdminPolicy 4. Amazon EKS クラスターのセキュリティグループを見つけます。Lambda 関数から Amazon EKS クラスターへの受信ネットワークトラフィックを許可するインバウンドルールを追加します。 インバウンドルールには次の値を使用します。 • タイプ – HTTPS • ポート範囲 – 443 • ソース – Lambda セキュリティグループ 詳細については、Amazon VPC ドキュメントの「セキュリティグループルールを設定する」を参照してください。	

ソリューションをテストする

タスク	説明	必要なスキル
Amazon Bedrock エージェントをテストします。	Amazon Bedrock エージェントをテストする前に、以下を実行してください。	AWS DevOps

タスク	説明	必要なスキル
	• 最初に非本番稼働用ロールでテストします。 • クラスターアクセスに加えられた変更を文書化します。 • 必要に応じて変更を元に戻す計画を立てます。 Amazon Bedrock エージェントにアクセスするには、次の手順を実行します。 1. Amazon Bedrock アクセス許可を持つ IAM ロール AWS Management Console を使用して にサインインし、https://console.aws.amazon.com/bedrock/ で Amazon Bedrock コンソールを開きます。 2. 左側のナビゲーションペインで [エージェント] を選択します。次に、 エージェント セクションで設定済みエージェントを選択します。 3. エージェントをテストするには、 を実際の IAM ロール ARN `Principal-ARN-OF-ROLE` に置き換える次のサンプルプロンプトを試してください。	

タスク	説明	必要なスキル
	• EKS クラスターへのアクセスを許可する IAM ロールのアクセスエントリを作成するには、次のプロンプトを使用します。 `Create an access entry in cluster eks-testing-new for a role whose principal arn is <Principal-ARN-OF-ROLE> with access policy as AmazonEKSAdminPolicy` 期待される結果： • エージェントはアクセスエントリの作成を確認する必要があります。 • 確認するには、 を使用する AWS Management Console か、Amazon EKS API を使用して次のコマンドを実行します。 `aws eks list-access-entries --cluster-name ekscluster` • 作成したアクセスエントリを記述するには、次のプロンプトを使用します。 `Describe an access entry in cluster eks-testing-new whose principal arn is`	

タスク	説明	必要なスキル
	<Principal-ARN-OF-ROLE> 期待される結果： • エージェントは、アクセスエントリに関する詳細を返す必要があります。 • 詳細は、アクセスエントリに対して以前に設定した内容と一致する必要があります。 • 作成したアクセスエントリを削除するには、次のプロンプトを使用します。 Delete the access entry in cluster eks-testing-new whose principal arn is <Principal-ARN-OF-ROLE> 期待される結果： • エージェントはアクセスエントリの削除を確認する必要があります。 • 確認するには、 を使用する AWS Management Console か、Amazon EKS API を使用して次のコマンドを実行します。 aws eks list-access-entries --cluster-name ekscluster	

タスク	説明	必要なスキル
	EKS Pod Identity の関連付けに対してアクションを実行するようにエージェントに依頼することもできます。詳細については、Amazon EKS ドキュメントの「EKS Pod Identity がポッドに へのアクセスを許可する方法 AWS のサービス」を参照してください。	

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップします。	このパターンが作成したリソースをクリーンアップするには、次の手順を使用します。各削除ステップが完了するまで待ってから、次のステップに進みます。 **Warning** この手順では、これらのスタックによって作成されたすべてのリソースが完全に削除されます。先に進む前に、重要なデータをバックアップしていることを確認してください。	AWS DevOps

タスク	説明	必要なスキル

	1. Amazon EKS クラスターを削除するには、次のコマンドを実行します。
	
	`eksctl delete cluster -f eks-deploy.yaml`
	
	Note
	このオペレーションが完了するまでに 15～20 分かかる場合があります。
	
	2. Amazon S3 バケットを削除するには、次のコマンドを実行します。
	• Lambda バケットを空にするには：
	
	`aws s3 rm s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} --recursive`
	
	• ナレッジベースバケットを空にするには：
	
	`aws s3 rm s3://eks-knowledge-base-${AWS_ACCOUNT} --recursive`
	
	• OpenAPI スキーマバケットを空にするには：	

タスク	説明	必要なスキル
	``` aws s3 rm s3://bedrock-agent-openapi-schema-${AWS_ACCOUNT} -recursive ``` • 空のバケットを削除するには： ``` aws s3 rb s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} aws s3 rb s3://eks-knowledge-base-${AWS_ACCOUNT} aws s3 rb s3://bedrock-agent-openapi-schema-${AWS_ACCOUNT} ``` 3. CloudFormation スタックを削除するには、次のコマンドを実行します。 ``` aws cloudformation delete-stack \ --stack-name ``` 4. Amazon EKS クラスターの削除を確認するには、次のコマンドを実行します。 ``` eksctl get clusters ``` 5. Amazon S3 バケットの削除を確認するには、次のコマンドを実行します。	

タスク	説明	必要なスキル

	• Lambda バケットの削除を確認するには：
	
	`aws s3 ls | grep "bedrock-agent-lambda-artifacts"`
	
	• ナレッジベースバケットの削除を確認するには：
	
	`aws s3 ls | grep "eks-knowledge-base"`
	
	• OpenAPI スキーマバケットの削除を確認するには：
	
	`aws s3 ls | grep "bedrock-agent-openapi-schema"`
	
	6. スタックの削除を確認するには、次のコマンドを実行します。
	
	`aws cloudformation list-stacks \--query 'StackSummaries[?StackName==``]'`
	
	スタックの削除に失敗した場合は、「トラブルシューティング」を参照してください。	

トラブルシューティング

問題	ソリューション
環境のセットアップ中にゼロ以外のエラーコードが返されます。	コマンドを実行してこのソリューションをデプロイするときに、正しいフォルダを使用していることを確認します。詳細については、このパターンのリポジトリにある FIRST_DEPLOY.md ファイルを参照してください。
Lambda 関数はタスクを実行できません。	Lambda 関数から Amazon EKS クラスターへの接続が正しく設定されていることを確認します。
エージェントプロンプトは APIsを認識しません。	ソリューションをデプロイする 詳細については、このパターンのリポジトリの RE_DEPLOY.md ファイルを参照してください。
スタックの削除に失敗しました。	スタックの削除を最初に試みると、失敗する可能性があります。この障害は、ナレッジベースのインデックスを作成する OpenSearch コレクション用に作成されたカスタムリソースの依存関係の問題が原因で発生する可能性があります。スタックを削除するには、カスタムリソースを保持して削除オペレーションを再試行します。

関連リソース

AWS ブログ

- 簡素化された Amazon EKS アクセス管理コントロールの詳細

Amazon Bedrock ドキュメント

- AI エージェントを使用してアプリケーションのタスクを自動化する

- Amazon Bedrock エージェントの仕組み
- エージェントの動作のテストとトラブルシューティング
- アクショングループを使用して、エージェントが実行するアクションを定義する

Amazon EKS ドキュメント

- Amazon EKS でのアクセスコントロールの仕組みについて説明します。

Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする

作成者: Martin Maritsch (AWS)、Alice Morano (AWS)、Julian Ferdinand Grueber (AWS)、Nicolas Jacob Baer (AWS)、Olivier Brique (AWS)、Nicola D Orazio (AWS)

概要

AWS には、検索拡張生成 (RAG) 対応の生成 AI ユースケースを構築するためのさまざまなオプションが用意されています。このパターンは、LangChain および Amazon Aurora PostgreSQL 互換をベクトルストアとする RAG ベースのアプリケーション向けのソリューションを提供します。Terraform を使用してこのソリューションを に直接デプロイ AWS アカウント し、次のシンプルな RAG ユースケースを実装できます。

1. ユーザーは、Microsoft Excel ファイルや PDF ドキュメントなどの Amazon Simple Storage Service (Amazon S3) バケットにファイルを手動でアップロードします。（サポートされているファイルタイプの詳細については、Unstructured ドキュメントを参照してください）。
2. ファイルの内容が抽出され、サーバーレス Aurora PostgreSQL 互換のナレッジデータベースに埋め込まれます。これは、ベクトルストアへのドキュメントのほぼリアルタイムの取り込みをサポートします。このアプローチにより、RAG モデルは低レイテンシーが重要なユースケースの関連情報にアクセスして取得できます。
3. ユーザーがテキスト生成モデルを使用する場合、以前にアップロードされたファイルから関連コンテンツを取得して拡張することで、インタラクションを強化します。

このパターンでは、Amazon Bedrock で利用可能な埋め込みモデルとして Amazon Titan Text Embeddings v2 を使用し、テキスト生成モデルとして Anthropic Claude 3 Sonnet を使用します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface（AWS CLI) がインストールされ、 で設定されています AWS アカウント。インストール手順については、 AWS CLI ドキュメントの「最新バージョンの のインストールまたは更新 AWS CLI」を参照してください。 AWS 認証情報とアカウントへのアクセスを確認するには、 AWS CLI ドキュメントの「設定と認証情報ファイルの設定」を参照してください。

- の Amazon Bedrock コンソールで必要な大規模言語モデル (LLMs) に対して有効になっているモデルアクセス AWS アカウント。このパターンには、次の LLMsが必要です。
  - `amazon.titan-embed-text-v2:0`
  - `anthropic.claude-3-sonnet-20240229-v1:0`

制約事項

- このサンプルアーキテクチャには、ベクトルデータベースを使用したプログラムによる質問への回答のためのインターフェイスは含まれていません。ユースケースで API が必要な場合は、取得タスクと質疑応答タスクを実行する AWS Lambda 関数で Amazon API Gateway を追加することを検討してください。
- このサンプルアーキテクチャには、デプロイされたインフラストラクチャのモニタリング機能は含まれていません。ユースケースでモニタリングが必要な場合は、AWS モニタリングサービスの追加を検討してください。
- 短期間に大量のドキュメントを Amazon S3 バケットにアップロードすると、Lambda 関数でレート制限が発生する可能性があります。ソリューションとして、Lambda 呼び出しのレートを制御できる Amazon Simple Queue Service (Amazon SQS) キューを使用して Lambda 関数を切り離すことができます。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- AWS CLI バージョン 2 以降
- Docker バージョン 26.0.0 以降
- Poetry バージョン 1.7.1 以降
- Python バージョン 3.10 以降
- Terraform バージョン 1.8.4 以降

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

この図は、以下を示しています。

1. Amazon S3 バケット にオブジェクトが作成されるとbedrock-rag-template-<account_id>、Amazon S3 通知は Lambda 関数 を呼び出しますdata-ingestion-processor。
2. Lambda 関数data-ingestion-processorは、Amazon Elastic Container Registry (Amazon ECR) リポジトリ に保存されている Docker イメージに基づいていますbedrock-rag-template。

   関数は LangChain S3FileLoader を使用して、ファイルを LangChain ドキュメントとして読み取ります。次に、LangChain RecursiveCharacterTextSplitter は、Amazon Titan Text Embedding V2 埋め込みモデルの最大トークンサイズに依存する CHUNK_SIZEと を指定CHUNK_OVERLAPして、各ドキュメントをチャンク化します。次に、Lambda 関数は Amazon Bedrock の埋め込みモデルを呼び出して、チャンクを数値ベクトル表現に埋め込みます。最後に、これらのベクトルは Aurora PostgreSQL データベースに保存されます。データベースにアクセスするには、Lambda 関数がまずユーザー名とパスワードを取得します AWS Secrets Manager。
3. Amazon SageMaker AI ノートブックインスタンス ではaws-sample-bedrock-rag-template、ユーザーは質問プロンプトを記述できます。このコードは Amazon Bedrock で Claude 3 を呼び出し、ナレッジベース情報をプロンプトのコンテキストに追加します。その結果、Claude 3 はドキュメント内の情報を使用してレスポンスを提供します。

このパターンのネットワークとセキュリティへのアプローチは次のとおりです。

- Lambda 関数data-ingestion-processorは、仮想プライベートクラウド (VPC) 内のプライベートサブネットにあります。Lambda 関数は、セキュリティグループのため、パブリックインターネットにトラフィックを送信することはできません。その結果、Amazon S3 と Amazon Bedrock へのトラフィックは VPC エンドポイントを介してのみルーティングされます。したがって、トラフィックはパブリックインターネットを経由しないため、レイテンシーが短縮され、ネットワークレベルでセキュリティレイヤーが追加されます。
- すべてのリソースとデータは、該当する場合、エイリアス で AWS Key Management Service (AWS KMS) キーを使用して暗号化されますaws-sample/bedrock-rag-template。

自動化とスケール

このパターンでは、Terraform を使用してコードリポジトリから にインフラストラクチャをデプロイします AWS アカウント。

ツール

AWS のサービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。このパターンでは、Aurora PostgreSQL 互換は pgvector プラグインをベクトルデータベースとして使用します。
- Amazon Bedrock は、主要な AI スタートアップと Amazon からの高性能な基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を使用して を操作するのに役立つオープンソースツールです。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。このパターンでは、Amazon ECR は `data-ingestion-processor` Lambda 関数の Docker イメージをホストします。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を認可するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Key Management Service (AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。このパターンでは、Lambda はデータをベクトルストアに取り込みます。
- Amazon SageMaker AI は、ML モデルを構築およびトレーニングし、本番環境対応のホスト環境にデプロイするのに役立つマネージド機械学習 (ML) サービスです。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。VPC には、トラフィックフローを制御するためのサブネットとルーティングテーブルが含まれています。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。
- Poetry は、Python での依存関係の管理とパッケージングのためのツールです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub terraform-rag-template-using-amazon-bedrock リポジトリで入手できます。

ベストプラクティス

- このコードサンプルは任意の にデプロイできますが AWS リージョン、米国東部 (バージニア北部) – `us-east-1`または米国西部 (北カリフォルニア) – を使用することをお勧めしますus-west-1。このレコメンデーションは、このパターンの公開時点での Amazon Bedrock での基盤モデルと埋め込みモデルの可用性に基づいています。での Amazon Bedrock 基盤モデルサポートup-to-dateリストについては AWS リージョン、Amazon Bedrock ドキュメントの「 によるモデルサポート AWS リージョン」を参照してください。このコードサンプルを他のリージョンにデプロイする方法については、「追加情報」を参照してください。
- このパターンは、proof-of-concept (PoC) またはパイロットデモのみを提供します。コードを本番環境に移行する場合は、次のベストプラクティスを使用してください。
  - Amazon S3 のサーバーアクセスログ記録を有効にします。
  - Lambda 関数のモニタリングとアラートを設定します。
  - ユースケースで API が必要な場合は、取得タスクと質疑応答タスクを実行する Lambda 関数で Amazon API Gateway を追加することを検討してください。

- 最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

ソリューションを にデプロイする AWS アカウント

タスク	説明	必要なスキル
リポジトリをクローン作成します。	このパターンで提供されている GitHub リポジトリのクローンを作成するには、次のコマンドを使用します。 `git clone https://github.com/aws-samples/terraform-rag-template-using-amazon-bedrock`	AWS DevOps
変数を設定します。	このパターンのパラメータを設定するには、以下を実行します。 1. コンピュータの GitHub リポジトリで、次のコマンドを使用して `terraform` フォルダを開きます。 `cd terraform` 2. `commons.tfvars` ファイルを開き、必要に応じてパラメータをカスタマイズします。	AWS DevOps

タスク	説明	必要なスキル
ソリューションをデプロイします。	ソリューションを展開するには、次を実行します。 1. `terraform` フォルダで、次のコマンドを使用して Terraform を実行し、カスタマイズした変数を渡します。 ``` terraform init terraform apply -var-file=commons.tfvars ``` 2. アーキテクチャ図に示されているリソースが正常にデプロイされたことを確認します。 インフラストラクチャデプロイは、VPC 内に SageMaker AI インスタンスをプロビジョニングし、Aurora PostgreSQL データベースにアクセスするアクセス許可を付与します。	AWS DevOps

ソリューションをテストする

タスク	説明	必要なスキル
デモを実行します。	前回のインフラストラクチャのデプロイが成功したら、次のステップを使用して Jupyter	AWS 全般

タスク	説明	必要なスキル
	ノートブックでデモを実行します。 1. インフラストラクチャがデプロイ AWS アカウント されている AWS Management Console の にサインインします。 2. SageMaker AI ノートブックインスタンス を開きますaws-sample-bedrock-rag-template 。 3. ドラッグアンドドロップを使用して、rag_demo.ipynb Jupyter ノートブックを SageMaker AI ノートブックインスタンスに移動します。 4. SageMaker AI ノートブックインスタンスrag_demo.ipynb でを開き、conda_python3 カーネルを選択します。 5. デモを実行するには、ノートブックのセルを実行します。 Jupyter ノートブックでは、以下のプロセスについて説明します。 • インストール要件 • 定義の埋め込み	

タスク	説明	必要なスキル
	• データベース接続 • データ取り込み • 拡張テキスト生成の取得 • 関連するドキュメントクエリ	

インフラストラクチャをクリーンアップする

タスク	説明	必要なスキル
インフラストラクチャをクリーンアップします。	不要になったときに作成したすべてのリソースを削除するには、次のコマンドを使用します。 `terraform destroy -var-file=commons.tfvars`	AWS DevOps

関連リソース

AWS リソース

- Python で Lambda 関数を構築する
- 基盤モデルの推論パラメータ
- Amazon Bedrock 基盤モデルへのアクセス
- 生成 AI アプリケーションにおけるベクトルデータベースの役割 (AWS データベースブログ)
- Amazon Aurora PostgreSQL の操作

その他のリソース

- pgvector ドキュメント

追加情報

ベクトルデータベースの実装

このパターンでは、Aurora PostgreSQL 互換を使用して RAG のベクトルデータベースを実装します。Aurora PostgreSQL の代わりに、 は Amazon Bedrock ナレッジベースや Amazon OpenSearch Service など、RAG の他の機能やサービス AWS を提供します。特定の要件に最適なソリューションを選択できます。

- Amazon OpenSearch Service は、大量のデータを保存およびクエリするために使用できる分散検索および分析エンジンを提供します。
- Amazon Bedrock ナレッジベースは、RAG の取り込みおよび取得プロセスを簡素化するための追加の抽象化としてナレッジベースを構築およびデプロイするように設計されています。Amazon Bedrock ナレッジベースは、Aurora PostgreSQL と Amazon OpenSearch Service の両方を操作できます。

他の へのデプロイ AWS リージョン

アーキテクチャで説明されているように、このコードサンプルをus-west-1デプロイするには、米国東部 (バージニア北部) リージョンus-east-1または米国西部 (北カリフォルニア) リージョンを使用することをお勧めします。ただし、このコードサンプルを us-east-1および 以外のリージョンにデプロイするには、2 つの方法がありますus-west-1。ファイルでデプロイリージョンを設定できますcommons.tfvars。クロスリージョン基盤モデルにアクセスするには、次のオプションを検討してください。

- パブリックインターネットを経由する – トラフィックがパブリックインターネットを経由できる場合は、インターネットゲートウェイを VPC に追加します。次に、Lambda 関数data-ingestion-processorと SageMaker AI ノートブックインスタンスに割り当てられたセキュリティグループを調整して、パブリックインターネットへのアウトバウンドトラフィックを許可します。
- パブリックインターネットを経由しない – このサンプルを us-east-1または 以外のリージョンにデプロイするにはus-west-1、次の手順を実行します。

1. us-east-1 または us-west-1リージョンのいずれかで、 の VPC エンドポイントを含む追加の VPC を作成しますbedrock-runtime。
2. VPC ピアリングまたはトランジットゲートウェイを使用して、アプリケーション VPC へのピアリング接続を作成します。 https://docs.aws.amazon.com/vpc/latest/tgw/tgw-peering.html

3. us-east-1 または 以外の Lambda 関数で bedrock-runtime boto3 クライアントを設定する場合はus-west-1、 us-east-1 または us-west-1 bedrock-runtimeの の VPC エンドポイントのプライベート DNS 名endpoint_urlを boto3 クライアントに渡します。

Amazon SageMaker の推論パイプラインを使用して、前処理ロジックを単一エンドポイントの ML モデルにデプロイする

作成者: Mohan Gowda Purushothama (AWS)、Gabriel Rodriguez Garcia (AWS)、Mateusz Zaremba (AWS)

概要

このパターンでは、Amazon SageMaker の推論パイプラインを使用して 1 つのエンドポイントに複数のパイプラインモデルオブジェクトをデプロイする方法を説明します。パイプラインモデルオブジェクトは、前処理、モデル推論、後処理など、さまざまな機械学習 (ML) ワークフローステージを表します。このパターンでは、シリアル接続されたパイプラインモデルオブジェクトのデプロイを説明するために、SageMaker に組み込まれている線形学習アルゴリズムに基づく前処理の Scikit-learn コンテナとリグレッションモデルをデプロイする方法を示します。デプロイメントは SageMaker の単一のエンドポイントの背後でホストされます。

> **Note**
>
> このパターンのデプロイでは、ml.m4.2xlarge インスタンスタイプを使用します。データサイズの要件とワークフローの複雑さに合ったインスタンスタイプを使用することをお勧めします。詳細については、「Amazon SageMaker の料金」を参照してください。このパターンでは Scikit-Learn 用にビルド済みの Docker イメージを使用しますが、独自の Docker コンテナを使用してワークフローに統合することもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Python 3.9
- Amazon SageMaker Python SDK と Boto3 ライブラリ
- 基本的な SageMaker 権限と Amazon Simple Storage Service (Amazon S3) 権限を持つ AWS Identity and Access Management (AWS IAM) ロール

製品バージョン

- Amazon SageMaker Python SDK 2.49.2

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Elastic Container Registry (Amazon ECR)
- Amazon SageMaker
- Amazon SageMaker Studio
- Amazon Simple Storage Service (Amazon S3)
- Amazon SageMaker 用のリアルタイム推論エンドポイント

ターゲットアーキテクチャ

次の図表は、Amazon SageMaker Pipelines モデルオブジェクトをデプロイするためのアーキテクチャを示します。

この図表は、次のワークフローを示しています：

1. SageMaker ノートブックはパイプラインモデルをデプロイします。
2. S3 バケットにはモデルアーティファクトが格納されます。
3. Amazon ECR は S3 バケットからソースコンテナイメージを取得します。

ツール

AWS ツール

- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- 「Amazon SageMaker」 は、ML モデルの構築とトレーニングを行うことができ、それらを稼働準備が整ったホストされている環境にデプロイするのに役立つマネージド型 ML サービスです。
- Amazon SageMaker Studio は、ML モデルの構築、トレーニング、デバッグ、デプロイ、モニタリングを可能にする、ウェブベースの ML 用統合開発環境 (IDE) です。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コード

このパターンのコードは、GitHub 内の「Inference Pipeline with Scikit-learn and Linear Learner」リポジトリで利用できます。

エピック

データセットの準備

タスク	説明	必要なスキル

リグレッションタスク用のデータセットを準備します。	Amazon SageMaker Studio でノートブックを開きます。
	
	必要なライブラリをすべてインポートして作業環境を初期化するには、ノートブックで以下のサンプルコードを使用してください。
	
	`import sagemaker`
	`from sagemaker import get_execution_role`
	
	`sagemaker_session = sagemaker.Session()`
	
	`# Get a SageMaker-compatible role used by this Notebook Instance.`
	`role = get_execution_role()`
	
	`# S3 prefix`
	`bucket = sagemaker_session.default_bucket()`
		データサイエンティスト

タスク	説明	必要なスキル

	`prefix = "Scikit-LinearLearner-pipeline-abalone-example"`
	
	サンプルデータセットをダウンロードするには、以下のコードをノートブックに追加します。
	
	`! mkdir abalone_data`
	`! aws s3 cp s3://sagemaker-sample-files/datasets/tabular/uci_abalone/abalone.csv ./abalone_data`
	
	Note
	このパターンの例では、UCI Machine Learningリポジトリのアワビデータセットを使用します。	

タスク	説明	必要なスキル

データセットを S3 バケットにアップロードします。	先にデータセットを準備したノートブックに、以下のコードを追加してサンプルデータを S3 バケットにアップロードします。
	`WORK_DIRECTORY = "abalone_data"`
	`train_input = sagemaker_session.upload_data( path="{}/{}".format(WORK_DIRECTORY, "abalone.csv"), bucket=bucket, key_prefix="{}/{}".format(prefix, "train"), )`
		データサイエンティスト

SKLearn を使用してデータプリプロセッサを作成します。

タスク	説明	必要なスキル
preprocessor.py スクリプトを準備します。	1. GitHub sklearn_abalone_featurizer.py リポジトリの Python ファイルから前処理ロジックをコピーし、そのコードを `sklearn_abalone_featurizer.py` という別の Python ファイルに貼り付けます。カスタムデータセットとカスタムワークフローに合うようにコードを変更できます。	データサイエンティスト

タスク	説明	必要なスキル
	2. `sklearn_abalone_featurizer.py` ファイルをプロジェクトのルートディレクトリ (つまり、SageMaker ノートブックを実行するのと同じ場所) に保存します。	

タスク	説明	必要なスキル

SKLearn プリプロセッサオブジェクトを作成します。	最終的な推論パイプラインに組み込むことができる SKLearn プリプロセッサオブジェクト (SKLearn Estimator と呼ばれる) を作成するには、SageMaker ノートブックで次のコードを実行します。
	
	`from sagemaker.sklearn.estimator import SKLearn`
	
	`FRAMEWORK_VERSION = "0.23-1"`
	`script_path = "sklearn_abalone_featurizer.py"`
	
	`sklearn_preprocessor = SKLearn(`
	`entry_point=script_path,`
	`role=role,`
	`framework_version=FRAMEWORK_VERSION,`
	`instance_type="ml.c4.xlarge",`
	`sagemaker_session=sagemaker_session,`
	`)`
	`sklearn_preprocessor.fit({"train": train_input})`
		データサイエンティスト

タスク	説明	必要なスキル

プリプロセッサの推論をテストします。	プリプロセッサが正しく定義されていることを確認するには、SageMaker ノートブックに次のコードを入力してバッチ変換ジョブを起動します。
	
	`# Define a SKLearn Transformer from the trained SKLearn Estimator`
	`transformer = sklearn_preprocessor.transformer(`
	`instance_count=1, instance_type="ml.m5.xlarge", assemble_with="Line", accept="text/csv"`
	`)`
	
	`# Preprocess training input`
	`transformer.transform(train_input, content_type="text/csv")`
	`print("Waiting for transform job: " + transformer.latest_transform_job.job_name)`
	`transformer.wait()`
	`preprocessed_train = transformer.output_path`	

機械学習モデルを作成する

タスク	説明	必要なスキル
モデルオブジェクトを作成します。	線形学習器アルゴリズムに基づいてモデルオブジェクトを作成するには、SageMaker ノートブックに次のコードを入力します。 `import boto3 from sagemaker .image_uris import retrieve ll_image = retrieve("linear-learner", boto3.Session().region_name) s3_ll_output_key_prefix = "ll_training_output" s3_ll_output_location = "s3://{}/{}/{}/{}" .format(bucket, prefix, s3_ll_output_key_prefix, "ll_model") ll_estimator = sagemaker.estimator.Estimator(ll_image, role, instance_count=1, instance_type="ml.m4.2xlarge", volume_size=20, max_run=3600, input_mode="File",`	データサイエンティスト

タスク	説明	必要なスキル

	<pre> output_path=s3_ll_
	output_location,
	    sagemaker_session=
	sagemaker_session,
	)
	
	ll_estimator.s
	et_hyperparameters
	(feature_dim=10,
	 predictor_type="re
	gressor", mini_batc
	h_size=32)
	
	ll_train_data =
	 sagemaker.inputs.T
	rainingInput(
	    preprocessed_train
	,
	    distribution="Full
	yReplicated",
	    content_type="text
	/csv",
	    s3_data_type="S3Pr
	efix",
	)
	
	data_channels =
	 {"train": ll_train_
	data}
	ll_estimator.fit(inpu
	ts=data_channels,
	 logs=True)</pre>
	上記のコードでは、パブリック Amazon ECR レジストリからモデルの関連する Amazon ECR Docker イメージを取得し、推定オブジェクトを作成し、そのオブジェクトを使用	

タスク	説明	必要なスキル
	してリグレッションモデルをトレーニングします。	

最後のパイプラインをデプロイする

タスク	説明	必要なスキル

パイプラインモデルをデプロイします。	パイプラインモデルオブジェクト (つまり、プリプロセッサオブジェクト) を作成してオブジェクトをデプロイするには、SageMaker ノートブックに次のコードを入力します。
	`from sagemaker.model import Model`
	`from sagemaker.pipeline import PipelineModel`
	`import boto3`
	`from time import gmtime, strftime`
	
	`timestamp_prefix = strftime("%Y-%m-%d-%H-%M-%S", gmtime())`
	
	`scikit_learn_inferencee_model = sklearn_preprocessor.create_model()`
	`linear_learner_model = ll_estimator.create_model()`
	
	`model_name = "inferenc e-pipeline-" + timestamp_prefix`
		データサイエンティスト

タスク	説明	必要なスキル
	endpoint_name = "inference-pipeline-ep-" + timestamp_prefix sm_model = PipelineModel(name=model_name, role=role, models=[scikit_learn_inferencee_model, linear_learner_model]) sm_model.deploy(initial_instance_count=1, instance_type="ml.c4.xlarge", endpoint_name=endpoint_name) **Note** モデルオブジェクトで使用されるインスタンスタイプは、ニーズに合わせて調整できます。	

タスク	説明	必要なスキル

推論をテストします。	エンドポイントが正しく動作していることを確認するには、SageMaker ノートブックで以下のサンプル推論コードを実行します。
	
	`from sagemaker.predictor import Predictor`
	`from sagemaker.serializers import CSVSerializer`
	
	`payload = "M, 0.44, 0.365, 0.125, 0.516, 0.2155, 0.114, 0.155"`
	`actual_rings = 10`
	`predictor = Predictor(`
	`endpoint_name=endpoint_name, sagemaker_session=sagemaker_session, serializer=CSVSerializer()`
	`)`
	
	`print(predictor.predict(payload))`
		データサイエンティスト

関連リソース

- Preprocess input data before making predictions using Amazon SageMaker inference pipelines and Scikit-learn (AWS 機械学習のブログ)
- End to end Machine Learning with Amazon SageMaker (GitHub)

RAG および ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する

作成者: Praveen Kumar Jeyarajan (AWS)、Jundong Qiao (AWS)、Kara Yang (AWS)、Kioua Jackson (AWS)、Noah Hamilton (AWS)、Shuai Cao (AWS)

概要

一般的な企業では、データの 70% がサイロ化されたシステムにトラップされています。生成 AI を活用したチャットベースのアシスタントを使用して、自然言語のやり取りを通じて、これらのデータサイロ間のインサイトと関係性を解放できます。生成 AI を最大限に活用するには、出力が信頼性が高く、正確で、利用可能な企業データが含まれている必要があります。成功するチャットベースのアシスタントは、以下によって異なります。

- 生成 AI モデル (Anthropic Claude 2 など)
- データソースのベクトル化
- ReAct フレームワークなどの高度な推論手法でモデルをプロンプトする

このパターンは、Amazon Simple Storage Service (Amazon S3) バケット、AWS Glue、Amazon Relational Database Service (Amazon RDS) などのデータソースからのデータ取得アプローチを提供します。値は、chain-of-thought法で検索拡張生成 (RAG) をインターリーブすることで、そのデータから取得されます。その結果、企業の保存されたデータ全体を利用する、チャットベースの複雑なアシスタントの会話がサポートされます。

このパターンでは、Amazon SageMaker のマニュアルと料金データテーブルを例として使用して、生成 AI チャットベースのアシスタントの機能を調べます。料金とサービスの機能に関する質問に回答することで、顧客が SageMaker サービスを評価するのに役立つチャットベースのアシスタントを構築します。このソリューションは、フロントエンドアプリケーションを構築するための Streamlit ライブラリと、大規模言語モデル (LLM) を搭載したアプリケーションバックエンドを開発するための LangChain フレームワークを使用します。

チャットベースのアシスタントへの問い合わせは、3 つの可能なワークフローのいずれかにルーティングするための初期インテント分類で満たされます。最も洗練されたワークフローは、一般的なアドバイザリガイダンスと複雑な料金分析を組み合わせています。このパターンは、エンタープライズ、企業、産業のユースケースに合わせて調整できます。

前提条件と制限

前提条件

- AWS コマンドラインインターフェイス (AWS CLI) のインストールと設定
- AWS Cloud Development Kit (AWS CDK) Toolkit 2.114.1 以降がインストールされ、設定されている
- Python と AWS CDK の基本的な知識
- インストール済みGit
- Docker がインストールされました
- Python 3.11 以降がインストールされ、設定されている (詳細については、「 ツール」セクションを参照してください)
- AWS CDK を使用してブートストラップされたアクティブな AWS アカウント https://docs.aws.amazon.com/cdk/v2/guide/bootstrapping.html
- Amazon Bedrock サービスで有効になっている Amazon Titan と Anthropic Claude のモデルアクセス
- ターミナル環境における AWS セキュリティ認証情報 (`AWS_ACCESS_KEY_ID` を含む) の適切な設定

制約事項

- LangChain は、ストリーミングのすべての LLM をサポートしているわけではありません。Anthropic Claude モデルはサポートされていますが、AI21 Labs のモデルはサポートされていません。
- このソリューションは単一 AWS アカウントにデプロイされます。
- このソリューションは、Amazon Bedrock と Amazon Kendra が利用可能な AWS リージョンでのみデプロイできます。可用性の詳細については、Amazon Bedrock と Amazon Kendra のドキュメントを参照してください。

製品バージョン

- Python バージョン 3.11 以降
- Streamlit バージョン 1.30.0 以降
- Streamlit-chat バージョン 0.1.1 以降

- LangChain バージョン 0.1.12 以降
- AWS CDK バージョン 2.132.1 以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Athena
- Amazon Bedrock
- Amazon Elastic Container Service (Amazon ECS)
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon Kendra
- エラスティックロードバランシング

ターゲット アーキテクチャ

AWS CDK コードは、AWS アカウントでチャットベースのアシスタントアプリケーションをセットアップするために必要なすべてのリソースをデプロイします。次の図に示すチャットベースのアシスタントアプリケーションは、ユーザーからの SageMaker 関連のクエリに応答するように設計されています。ユーザーは Application Load Balancer を介して、Streamlit アプリケーションをホストする Amazon ECS クラスターを含む VPC に接続します。オーケストレーション Lambda 関数はアプリケーションに接続します。S3 バケットデータソースは、Amazon Kendra と AWS Glue を介して Lambda 関数にデータを提供します。Lambda 関数は Amazon Bedrock に接続して、チャットベースのアシスタントユーザーからのクエリ (質問) に応答します。

1. オーケストレーション Lambda 関数は、LLM プロンプトリクエストを Amazon Bedrock モデル (Claude 2) に送信します。
2. Amazon Bedrock は、LLM レスポンスをオーケストレーション Lambda 関数に送り返します。

オーケストレーション Lambda 関数内のロジックフロー

ユーザーが Streamlit アプリケーションを通じて質問すると、オーケストレーション Lambda 関数を直接呼び出します。次の図は、Lambda 関数が呼び出されたときのロジックフローを示しています。

- ステップ 1 – 入力 query (質問) は 3 つのインテントのいずれかに分類されます。
  - 一般的な SageMaker ガイダンスの質問
  - 一般的な SageMaker 料金 (トレーニング/推論) に関する質問
  - SageMaker と料金に関する複雑な質問
- ステップ 2 – 入力は 3 つのサービスのいずれかqueryを開始します。
  - `RAG Retrieval service`。Amazon Kendra ベクトルデータベースから関連するコンテキストを取得し、Amazon Bedrock を介して LLM を呼び出して、取得したコンテキストをレスポンスとして要約します。
  - `Database Query service`。 は、LLM、データベースメタデータ、および関連するテーブルのサンプル行を使用して、入力を SQL クエリに変換queryします。Database Query サービスは、Amazon Athena を介して SageMaker 料金データベースに対して SQL クエリを実行し、クエリ結果をレスポンスとして要約します。
  - `In-context ReACT Agent service`。レスポンスを提供する前に、入力を複数のステップqueryに分割します。エージェントは、推論プロセス中に関連情報を取得するためのツールDatabase Query serviceとして RAG Retrieval serviceと を使用します。推論とアクションのプロセスが完了すると、エージェントはレスポンスとして最終的な回答を生成します。
- ステップ 3 – オーケストレーション Lambda 関数からのレスポンスは、出力として Streamlit アプリケーションに送信されます。

ツール

AWS サービス

- 「Amazon Athena」は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析できるようにするインタラクティブなクエリサービスです。
- Amazon Bedrock は、主要な AI スタートアップと Amazon の高性能基盤モデル (FMs) を統一された API を通じて使用できるようにするフルマネージドサービスです。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの確実な分類、整理、強化、移動をサポートできます。このパターンでは、AWS Glue クローラーと AWS Glue データカタログテーブルを使用します。
- Amazon Kendra は、自然言語処理と高度な機械学習アルゴリズムを使用して、データからの検索質問に対する特定の回答を返すインテリジェントな検索サービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- 「Elastic Load Balancing (ELB)」は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。例えば、1 つ以上のアベイラビリティゾーンにある Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレス間でトラフィックを分散できます。

コードリポジトリ

このパターンのコードは、GitHub genai-bedrock-chatbot リポジトリで入手できます。

コードリポジトリには以下のファイルとフォルダが含まれています。

- `assets` フォルダ – アーキテクチャ図とパブリックデータセットの静的アセット
- `code/lambda-container` フォルダ – Lambda 関数で実行される Python コード
- `code/streamlit-app` フォルダ – Amazon ECS のコンテナイメージとして実行される Python コード
- `tests` フォルダ – AWS CDK コンストラクトをユニットテストするために実行される Python ファイル
- `code/code_stack.py` – AWS CDK コンストラクト AWS リソースの作成に使用される Python ファイル
- `app.py` – ターゲット AWS アカウントに AWS リソースをデプロイするために使用される AWS CDK スタック Python ファイル

- `requirements.txt` – AWS CDK にインストールする必要があるすべての Python 依存関係のリスト
- `requirements-dev.txt` – AWS CDK がユニットテストスイートを実行するためにインストールする必要があるすべての Python 依存関係のリスト
- `cdk.json` — リソースの起動に必要な値を提供する入力ファイル

> **Note**
>
> AWS CDK コードは、L3 (レイヤー 3) コンストラクトと AWS が管理する AWS Identity and Access Management (IAM) ポリシーを使用してソリューションをデプロイします。

ベストプラクティス

- ここで提供されるコード例は、proof-of-concept (PoC) またはパイロットデモのみを目的としています。コードを Production に持ち込む場合は、必ず次のベストプラクティスを使用してください。
  - Amazon S3 アクセスログ記録が有効になっています。
  - VPC フローログが有効になっています。
  - Amazon Kendra Enterprise Edition インデックスが有効になっています。
- Lambda 関数のモニタリングとアラートを設定します。詳細については、Lambda 関数をモニタリングおよびトラブルシューティングするを参照してください。Lambda 関数を使用する際の一般的なベストプラクティスについては、AWS ドキュメントを参照してください。

エピック

ローカルマシンで AWS 認証情報をセットアップする

タスク	説明	必要なスキル
スタックがデプロイされるアカウントと AWS リージョンの変数をエクスポートします。	環境変数を使用して AWS CDK の AWS 認証情報を提供には、次のコマンドを実行します。	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	`export CDK_DEFAU LT_ACCOUNT=<12 Digit AWS Account Number> export CDK_DEFAU LT_REGION=<region>`	
AWS CLI プロファイルをセットアップします。	アカウントの AWS CLI プロファイルを設定するには、AWS ドキュメントの指示に従います。	DevOps エンジニア、AWS DevOps

環境をセットアップします。

タスク	説明	必要なスキル
ローカルマシンにリポジトリを複製します。	リポジトリのクローンを作成するには、ターミナルで次のコマンドを実行します。 `git clone https://github.com/awslabs/genai-bedrock-chatbot.git`	DevOps エンジニア、AWS DevOps
Python 仮想環境を設定し、必要な依存関係をインストールします。	Python 仮想環境をセットアップするには、次のコマンドを実行します。 `cd genai-bedrock-chatbot python3 -m venv .venv source .venv/bin/activate`	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	必要な依存関係を設定するには、次のコマンドを実行します。 `pip3 install -r requirements.txt`	
AWS CDK 環境をセットアップし、AWS CDK コードを合成します。	1. AWS アカウントで AWS CDK 環境を設定するには、次のコマンドを実行します。 `cdk bootstrap aws://ACCOUNT-NUMBER/REGION` 2. コードを AWS CloudFormation スタック設定に変換するには、`cdk synth` コマンドを実行します。	DevOps エンジニア、AWS DevOps

チャットベースのアシスタントアプリケーションを設定してデプロイする

タスク	説明	必要なスキル
Claude モデルアクセスをプロビジョニングします。	AWS アカウントの Anthropic Claude モデルアクセスを有効にするには、Amazon Bedrock ドキュメントの指示に従います。	AWS DevOps
アカウントにリソースをデプロイします。	AWS CDK を使用して AWS アカウントにリソースをデプロイするには、次の手順を実行します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	1. クローンされたリポジトリのルートの `cdk.json` ファイルで、`logging`パラメータの入力を指定します。値の例は、`INFO`、`DEBUG`、`WARN`、および ですERROR。 これらの値は、Lambda 関数と Streamlit アプリケーションのログレベルのメッセージを定義します。 2. クローンされたリポジトリのルートにある `app.py` ファイルには、デプロイに使用される AWS CloudFormation スタック名が含まれています。デフォルトのスタック名は ですchatbot-stack 。 3. リソースをデプロイするには、`cdk deploy` コマンドを実行します。 `cdk deploy` コマンドは L3 コンストラクトを使用して、ドキュメントと CSV データセットファイルを S3 バケットにコピーするための複数の Lambda 関数を作成します。 4. コマンドが完了したら、AWS マネジメントコンソールにサインインし、CloudFormation コンソー	

タスク	説明	必要なスキル
	ルを開き、スタックが正常にデプロイされたことを確認します。 デプロイが成功したら、CloudFormation 出力セクションで提供されている URL を使用して、チャットベースのアシスタントアプリケーションにアクセスできます。	

タスク	説明	必要なスキル
AWS Glue クローラーを実行し、データカタログテーブルを作成します。	AWS Glue クローラーは、データスキーマを動的に保つために使用されます。このソリューションは、クローラーをオンデマンドで実行することで、AWS Glue データカタログテーブルにパーティションを作成および更新します。CSV データセットファイルが S3 バケットにコピーされたら、AWS Glue クローラを実行し、テスト用の Data Catalog テーブルスキーマを作成します。 1. AWS Glue コンソールに移動します。 2. ナビゲーションペインの [Data Catalog (データカタログ)] で、[Crawlers (クローラー)] を選択します。 3. サフィックス が付いたクローラを選択しますsagemaker-pricing-crawler 。 4. クローラーを実行します。 5. クローラーが正常に実行されると、AWS Glue データカタログテーブルが作成されます。	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	Note AWS CDK コードは、AWS Glue クローラをオンデマンドで実行するように設定しますが、定期的に実行するようにスケジュールすることもできます。	

タスク	説明	必要なスキル
ドキュメントのインデックス作成を開始します。	ファイルが S3 バケットにコピーされたら、Amazon Kendra を使用してクロールし、インデックスを作成します。 1. Amazon Kendra コンソールに移動します。 2. サフィックス が付いたインデックスを選択しますchatbot-index 。 3. ナビゲーションペインで、データソースを選択し、サフィックス が付いたデータソースコネクタを選択しますchatbot-index 。 4. 今すぐ同期を選択して、インデックス作成プロセスを開始します。 **Note** AWS CDK コードは、Amazon Kendra インデックス同期をオンデマンドで実行するように設定しますが、Schedule パラメータを使用して定期的に実行することもできます。	AWS DevOps、DevOps エンジニア

ソリューション内のすべての AWS リソースをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	ソリューションをテストしたら、リソースをクリーンアップします。 1. ソリューションによってデプロイされた AWS リソースを削除するには、`cdk destroy` コマンドを実行します。 2. 2 つの S3 バケットからすべてのオブジェクトを削除してから、バケットを削除します。 詳細については、バケットを削除するを参照してください。	DevOps エンジニア、AWS DevOps

トラブルシューティング

問題	ソリューション
AWS CDK はエラーを返します。	AWS CDK の問題に関するヘルプは、一般的な AWS CDK 問題をトラブルシューティングするを参照してください。

関連リソース

- Amazon Bedrock:
  - モデルアクセス
  - 基盤モデルの推論パラメータ

- Python で Lambda 関数を構築する
- AWS CDK の使用を開始する
- Python での AWS CDK の使用
- AWS で生成 AI Application Builder
- LangChain ドキュメント
- Streamlit ドキュメント

追加情報

AWS CDK コマンド

AWS CDK を使用する際は、以下の便利なコマンドに注意してください。

- アプリ内のすべてのスタックを一覧表示

```
cdk ls
```

- 合成された AWS CloudFormation テンプレートを発行

```
cdk synth
```

- スタックをデフォルトの AWS アカウントとリージョンにデプロイ

```
cdk deploy
```

- デプロイされたスタックを現在の状態と比較

```
cdk diff
```

- AWS CDK ドキュメントを開く

```
cdk docs
```

- CloudFormation スタックを削除し、AWS にデプロイされたリソースを削除します。

```
cdk destroy
```

Amazon Bedrock エージェントとナレッジベースを使用して、完全に自動化されたチャットベースのアシスタントを開発する

作成者: Jundong Qiao (AWS)、Kara Yang (AWS)、Kioua Jackson (AWS)、Noah Hamilton (AWS)、Praveen Kumar Jeyarajan (AWS)、Shuai Cao (AWS)

概要

多くの組織は、包括的な回答を提供するために多様なデータソースをオーケストレーションできるチャットベースのアシスタントを作成する際に課題に直面しています。このパターンは、ドキュメントとデータベースの両方からのクエリに簡単に回答できるチャットベースのアシスタントを開発するためのソリューションを示しています。

Amazon Bedrock 以降、このフルマネージド生成人工知能 (AI) サービスは、さまざまな高度な基盤モデル (FMsを提供します。これにより、プライバシーとセキュリティに重点を置いた生成 AI アプリケーションの効率的な作成が容易になります。ドキュメントの取得では、取得拡張生成 (RAG) が重要な機能です。ナレッジベースを使用して、外部ソースからのコンテキストに関連する情報で FM プロンプトを補強します。Amazon OpenSearch Serverless インデックスは、Amazon Bedrock のナレッジベースの背後にあるベクトルデータベースとして機能します。この統合は、不正確さを最小限に抑え、応答が事実関係のドキュメントに確実に固定されるように、慎重なプロンプトエンジニアリングによって強化されています。データベースクエリの場合、Amazon Bedrock の FMs は、テキストによる問い合わせを構造化された SQL クエリに変換し、特定のパラメータを組み込みます。これにより、AWS Glue データベースによって管理されるデータベースからデータを正確に取得できます。Amazon Athena は、これらのクエリに使用されます。

より複雑なクエリを処理するには、包括的な回答を得るには、ドキュメントとデータベースの両方から取得した情報が必要です。Agents for Amazon Bedrock は生成 AI 機能であり、複雑なタスクを理解し、オーケストレーションのためによりシンプルなタスクに分解できる自律型エージェントを構築できます。Amazon Bedrock の自律エージェントによって容易になる、簡素化されたタスクから取得したインサイトの組み合わせにより、情報の合成が強化され、より徹底的かつ網羅的な回答が得られます。このパターンは、Amazon Bedrock および関連する生成 AI サービスおよび自動ソリューション内の機能を使用してチャットベースのアシスタントを構築する方法を示しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント

- Docker、インストール済み
- AWS Cloud Development Kit (AWS CDK)、 `us-east-1`または AWS `us-west-2` リージョンにインストールおよびブートストラップ
- AWS CDK Toolkit バージョン 2.114.1 以降、インストール済み
- インストールされ、構成済みの AWS コマンドラインインターフェイス (AWS CLI)
- Python バージョン 3.11 以降、インストール済み
- Amazon Bedrock で、Claude 2、Claude 2.1、Claude Instant、Titan Embeddings G1 へのアクセスを有効にする – テキスト

制限

- このソリューションは単一 AWS アカウントにデプロイされます。
- このソリューションは、Amazon Bedrock と Amazon OpenSearch Serverless がサポートされている AWS リージョンでのみデプロイできます。詳細については、Amazon Bedrock と Amazon OpenSearch Serverless のドキュメントを参照してください。

製品バージョン

- Llama-index バージョン 0.10.6 以降
- Sqlalchemy バージョン 2.0.23 以降
- Opensearch-py バージョン 2.4.2 以降
- Requests_aws4auth バージョン 1.2.3 以降
- AWS SDK for Python (Boto3) バージョン 1.34.57 以降

アーキテクチャ

ターゲットテクノロジースタック

AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、AWS CloudFormation を介してプロビジョニングするためのオープンソースのソフトウェア開発フレームワークです。このパターンで使用される AWS CDK スタックは、次の AWS リソースをデプロイします。

- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

- AWS Glue データベースコンポーネントの AWS Glue データカタログ
- AWS Lambda
- AWS Identity and Access Management (IAM)
- Amazon OpenSearch Serverless
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- AWS Fargate
- Amazon Virtual Private Cloud (Amazon VPC)
- Application Load Balancer

ターゲット アーキテクチャ

この図は、複数の AWS サービスを使用した、単一の AWS リージョン内の包括的な AWS クラウドネイティブセットアップを示しています。チャットベースのアシスタントの主なインターフェイスは、Amazon ECS クラスターでホストされている Streamlit アプリケーションです。Application Load Balancer はアクセシビリティを管理します。このインターフェイスを介して行われたクエリは`Invocation`、Lambda 関数をアクティブ化し、Amazon Bedrock のエージェントとインターフェイスします。このエージェントは、Amazon Bedrock のナレッジベースを参照するか、`Agent executor`Lambda 関数を呼び出して、ユーザーの問い合わせに応答します。この関数は、事前定義された API スキーマに従って、エージェントに関連付けられた一連のアクションをトリガーします。Amazon Bedrock のナレッジベースは、ベクトルデータベース基盤として OpenSearch Serverless インデックスを使用します。さらに、`Agent executor`関数は Amazon Athena を介して AWS Glue データベースに対して実行される SQL クエリを生成します。

ツール

AWS サービス

- 「Amazon Athena」は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析できるようにするインタラクティブなクエリサービスです。
- Amazon Bedrock は、主要な AI スタートアップと Amazon の高性能基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。

- AWS コマンドラインインターフェイス (AWS CLI) は、コマンドラインシェルのコマンドを通じて AWS のサービスを操作するのに役立つオープンソースツールです。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- Elastic Load Balancing (ELB) は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。例えば、1 つ以上のアベイラビリティゾーンにある Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレス間でトラフィックを分散できます。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの確実な分類、整理、強化、移動をサポートできます。このパターンでは、AWS Glue クローラーと AWS Glue データカタログテーブルを使用します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon OpenSearch Serverless は、Amazon OpenSearch Service のオンデマンドサーバーレス設定です。このパターンでは、OpenSearch Serverless インデックスは Amazon Bedrock のナレッジベースのベクトルデータベースとして機能します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- Streamlit は、データアプリケーションを作成するためのオープンソースの Python フレームワークです。

コードリポジトリ

このパターンのコードは、GitHub genai-bedrock-agent-chatbot リポジトリで入手できます。コードリポジトリには以下のファイルとフォルダが含まれています。

- `assets` フォルダ – アーキテクチャ図やパブリックデータセットなどの静的アセット。
- `code/lambdas/action-lambda` フォルダ – Amazon Bedrock エージェントのアクションとして機能する Lambda 関数の Python コード。
- `code/lambdas/create-index-lambda` フォルダ – OpenSearch Serverless インデックスを作成する Lambda 関数の Python コード。

- `code/lambdas/invoke-lambda` フォルダ – Amazon Bedrock エージェントを呼び出す Lambda 関数の Python コード。Streamlit アプリケーションから直接呼び出されます。
- `code/lambdas/update-lambda` フォルダ – AWS CDK を介して AWS リソースがデプロイされた後にリソースを更新または削除する Lambda 関数の Python コード。
- `code/layers/boto3_layer` フォルダ – すべての Lambda 関数間で共有される Boto3 レイヤーを作成する AWS CDK スタック。
- `code/layers/opensearch_layer` フォルダ – インデックスを作成するためのすべての依存関係をインストールする OpenSearch Serverless レイヤーを作成する AWS CDK スタック。
- `code/streamlit-app` フォルダ – Amazon ECS のコンテナイメージとして実行される Python コード
- `code/code_stack.py` – AWS CDK は、AWS リソースを作成する Python ファイルを構築します。
- `app.py` – ターゲット AWS アカウントに AWS リソースをデプロイする AWS CDK スタック Python ファイル。
- `requirements.txt` – AWS CDK にインストールする必要があるすべての Python 依存関係のリスト。
- `cdk.json` – リソースの作成に必要な値を提供する入力ファイル。また、 `context/config`フィールドでは、それに応じてソリューションをカスタマイズできます。カスタマイズの詳細については、「追加情報」セクションを参照してください。

ベストプラクティス

- ここで提供されるコード例は、proof-of-concept (PoC) またはパイロットのみを目的としています。コードを本番環境に移行する場合は、次のベストプラクティスを使用してください。
  - Amazon S3 アクセスログ記録を有効にする
  - VPC フローログを有効にする
- Lambda 関数のモニタリングとアラートを設定します。詳細については、「Lambda 関数をモニタリングおよびトラブルシューティングする」を参照してください。ベストプラクティスについては、AWS Lambda 関数を使用するためのベストプラクティス」を参照してください。

エピック

ローカルワークステーションで AWS 認証情報を設定する

タスク	説明	必要なスキル
アカウントとリージョンの変数をエクスポートします。	環境変数を使用して AWS CDK の AWS 認証情報を提供するには、次のコマンドを実行します。 `export CDK_DEFAU LT_ACCOUNT=<12-digit AWS account number> export CDK_DEFAU LT_REGION=<Region>`	AWS DevOps、DevOps エンジニア
AWS CLI の名前付きプロファイルを設定します。	アカウントの AWS CLI 名前付きプロファイルを設定するには、「設定と認証情報ファイルの設定」の手順に従います。	AWS DevOps、DevOps エンジニア

環境をセットアップします。

タスク	説明	必要なスキル
リポジトリをローカルワークステーションにクローンします。	リポジトリのクローンを作成するには、ターミナルで次のコマンドを実行します。 `git clone https://g ithub.com/awslabs/ genai-bedrock-agent- chatbot.git`	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル

Python 仮想環境をセットアップします。	Python 仮想環境をセットアップするには、次のコマンドを実行します。
	`cd genai-bedrock-agent-chatbot`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	必要な依存関係を設定するには、次のコマンドを実行します。
	`pip3 install -r requirements.txt`
		DevOps エンジニア、AWS DevOps

AWS CDK 環境を設定します。	コードを AWS CloudFormation テンプレートに変換するには、 コマンドを実行しますcdk synth。	AWS DevOps、DevOps エンジニア

アプリケーションの設定とデプロイ

タスク	説明	必要なスキル
アカウントにリソースをデプロイします。	AWS CDK を使用して AWS アカウントにリソースをデプロイするには、以下を実行します。 1. クローンされたリポジトリのルートの cdk.json ファイルで、ログ記録パラメータの入力	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	を指定します。値の例は、INFO、DEBUG、WARN、および ですERROR。 これらの値は、Lambda 関数と Streamlit アプリケーションのログレベルのメッセージを定義します。 2. クローンされたリポジトリのルートにある cdk.json ファイルには、デプロイに使用される AWS CloudFormation スタック名が含まれています。デフォルトのスタック名はですchatbot-stack 。デフォルトの Amazon Bedrock エージェント名はでChatbotBedrockAgent 、デフォルトの Amazon Bedrock エージェントエイリアスは ですChatbot_Agent 。 3. リソースをデプロイするには、cdk deploy コマンドを実行します。 cdk deploy コマンドは Layer-3 コンストラクトを使用して、ドキュメントと CSV データセットファイルを S3 バケットにコピーするための複数の Lambda 関数を作成します。また、Amazon	

タスク	説明	必要なスキル
	Bedrock エージェント、ナレッジベース、および Amazon Bedrock エージェント用の `Action group` Lambda 関数もデプロイします。 4. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/cloudformation/://www.com で CloudFormation コンソールを開きます。 5. スタックが正常にデプロイされたことを確認します。手順については、AWS CloudFormation コンソールでのスタックの確認」を参照してください。 デプロイが成功したら、CloudFormation コンソールの出力タブにある URL を使用して、チャットベースのアシスタントアプリケーションにアクセスできます。	

ソリューション内のすべての AWS リソースをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	ソリューションをテストした後、リソースをクリーンアッ	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	プするには、 コマンド を実行しますcdk destroy。	

関連リソース

AWS ドキュメント

- Amazon Bedrock リソース：
  - モデルアクセス
  - 基盤モデルの推論パラメータ
  - Amazon Bedrock のエージェント
  - Amazon Bedrock のナレッジベース
- Python で Lambda 関数を構築する
- AWS CDK リソース：
  - AWS CDK の使用を開始する
  - AWS CDK の一般的な問題のトラブルシューティング
  - Python での AWS CDK の使用
- AWS で生成 AI Application Builder

その他の AWS リソース

- Amazon OpenSearch Serverless のベクトルエンジン

その他のリソース

- LlamaIndex ドキュメント
- Streamlit ドキュメント

追加情報

チャットベースのアシスタントを独自のデータでカスタマイズする

ソリューションをデプロイするためにカスタムデータを統合するには、以下の構造化されたガイドラインに従ってください。これらのステップは、シームレスで効率的な統合プロセスを実現するように設計されており、カスタムデータを使用してソリューションを効果的にデプロイできます。

ナレッジベースのデータ統合の場合

データ準備

1. assets/knowledgebase_data_source/ ディレクトリを見つけます。
2. このフォルダ内にデータセットを配置します。

設定の調整

1. cdk.json ファイルを開きます。
2. context/configure/paths/knowledgebase_file_name フィールドに移動し、それに応じて更新します。
3. bedrock_instructions/knowledgebase_instruction フィールドに移動し、新しいデータセットのニュアンスとコンテキストを正確に反映するように更新します。

構造データ統合の場合

データ組織

1. assets/data_query_data_source/ ディレクトリ内で、 などのサブディレクトリを作成しますtabular_data。
2. 構造化データセット (CSV、JSON、ORC、Parquet などの許容される形式) をこの新しく作成されたサブフォルダに配置します。
3. 既存のデータベースに接続する場合は、 の 関数を更新create_sql_engine()code/lambda/action-lambda/build_query_engine.pyしてデータベースに接続します。

設定とコードの更新

1. cdk.json ファイルで、新しいデータパスに合わせて context/configure/paths/athena_table_data_prefixフィールドを更新します。
2. データセットに対応する新しい text-to-SQL の例code/lambda/action-lambda/dynamic_examples.csvを組み込んで修正します。

3. 構造データセットの属性をミラーリングcode/lambda/action-lambda/prompt_templates.pyするように修正します。
4. cdk.json ファイルで、 context/configure/bedrock_instructions/action_group_descriptionフィールドを更新して Lambda Action group 関数の目的と機能について説明します。
5. assets/agent_api_schema/artifacts_schema.json ファイルで、Action groupLambda 関数の新しい機能について説明します。

一般的な更新

cdk.json ファイルで、 context/configure/bedrock_instructions/agent_instructionセクションに、新しく統合されたデータを考慮して、Amazon Bedrock エージェントの意図した機能と設計目的を包括的に説明します。

Amazon Bedrock と Amazon Transcribe を使用して、音声入力から組織の知識を文書化する

作成者: Praveen Kumar Jeyarajan (AWS)、Jundong Qiao (AWS)、Megan Wu (AWS)、Rajiv Upadhyay (AWS)

概要

組織の成功とレジリエンスを確保するには、組織の知識をキャプチャすることが最優先事項です。知識の蓄積は、従業員が蓄積した共通の知恵、インサイト、経験を表し、多くの場合、本質的に暗黙的で非公式に受け継がれます。この豊富な情報には、他には文書化されていない複雑な問題に対する独自のアプローチ、ベストプラクティス、およびソリューションが含まれます。この知識を形式化して文書化することで、企業は組織の記憶を保持し、イノベーションを促進し、意思決定プロセスを強化し、新しい従業員の学習曲線を加速できます。さらに、コラボレーションを促進し、個人に権限を与え、継続的な改善の文化を育みます。最終的には、組織の知識を活用することで、企業はワークフォースの集合的インテリジェンスである最も重要なアセットを使用して、課題を乗り越え、成長を促進し、動的なビジネス環境で競争上の優位性を維持できます。

このパターンでは、上級従業員からの音声録音を通じて組織の知識をキャプチャする方法について説明します。Amazon Amazon Transcribe と Amazon Bedrock を使用して、体系的なドキュメントと検証を行います。この非公式な知識を文書化することで、それを保存し、後続の従業員のグループと共有できます。この取り組みは、運用上の優秀性をサポートし、直接的な経験を通じて取得した実践的な知識を組み込むことで、トレーニングプログラムの有効性を向上させます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Docker、インストール済み
- AWS Cloud Development Kit (AWS CDK) バージョン 2.114.1 以降、 `us-east-1`または AWS `us-west-2` リージョンにインストールおよびブートストラップ
- AWS CDK Toolkit バージョン 2.114.1 以降、インストール済み
- インストールされ、構成済みの AWS コマンドラインインターフェイス (AWS CLI)
- Python バージョン 3.12 以降、インストール済み
- Amazon Transcribe、Amazon Bedrock、Amazon Simple Storage Service (Amazon S3)、および AWS Lambda リソースを作成するアクセス許可

制限

- このソリューションは単一 AWS アカウントにデプロイされます。
- このソリューションは、Amazon Bedrock と Amazon Transcribe が利用可能な AWS リージョンでのみデプロイできます。可用性の詳細については、Amazon Bedrock と Amazon Transcribe のドキュメントを参照してください。
- オーディオファイルは、Amazon Transcribe がサポートする形式である必要があります。サポートされている形式のリストについては、Transcribe ドキュメントの「メディア形式」を参照してください。

製品バージョン

- AWS SDK for Python (Boto3) バージョン 1.34.57 以降
- LangChain バージョン 0.1.12 以降

アーキテクチャ

アーキテクチャは、AWS でのサーバーレスワークフローを表します。AWS Step Functions は、オーディオ処理、テキスト分析、ドキュメント生成のために Lambda 関数をオーケストレーションします。次の図は、ステートマシンとも呼ばれる Step Functions ワークフローを示しています。

ステートマシンの各ステップは、個別の Lambda 関数によって処理されます。以下は、ドキュメント生成プロセスのステップです。

1. `preprocess` Lambda 関数は、Step Functions に渡された入力を検証し、指定された Amazon S3 URI フォルダパスに存在するすべてのオーディオファイルを一覧表示します。ワークフローのダウンストリーム Lambda 関数は、 ファイルリストを使用してドキュメントを検証、要約、生成します。
2. `transcribe` Lambda 関数は Amazon Transcribe を使用してオーディオファイルをテキストトランスクリプトに変換します。この Lambda 関数は、文字起こしプロセスを開始し、音声をテキストに正確に変換し、後続の処理のために保存します。
3. `validate` Lambda 関数はテキストトランスクリプトを分析し、最初の質問に対するレスポンスの関連性を判断します。Amazon Bedrock を介して大規模言語モデル (LLM) を使用することで、トピック上の回答を識別し、トピック外の回答から分離します。

4. `summarize` Lambda 関数は、Amazon Bedrock を使用して、トピック上の回答の一貫した簡潔な概要を生成します。
5. `generate` Lambda 関数は、概要を構造化されたドキュメントにアセンブルします。事前定義されたテンプレートに従ってドキュメントをフォーマットし、追加の必要なコンテンツやデータを含めることができます。
6. Lambda 関数のいずれかが失敗した場合、Amazon Simple Notification Service (Amazon SNS) を通じて E メール通知が送信されます。

このプロセス全体を通して、AWS Step Functions は各 Lambda 関数が正しい順序で開始されていることを確認します。このステートマシンには、効率を向上させるための並列処理の容量があります。Amazon S3 バケットは中央ストレージリポジトリとして機能し、関連するさまざまなメディアとドキュメントの形式を管理することでワークフローをサポートします。

ツール

AWS サービス

- Amazon Bedrock は、主要な AI スタートアップと Amazon の高性能基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functionsは、AWS Lambda関数と他のAWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。
- Amazon Transcribe は、機械学習モデルを使用して音声をテキストに変換する自動音声認識サービスです。

その他のツール

- LangChain は、大規模言語モデル (LLMs。

コードリポジトリ

このパターンのコードは、GitHub genai-knowledge-capture リポジトリで入手できます。

コードリポジトリには以下のファイルとフォルダが含まれています。

- `assets` フォルダ – アーキテクチャ図やパブリックデータセットなど、ソリューションの静的アセット
- `code/lambdas` フォルダ – すべての Lambda 関数の Python コード
  - `code/lambdas/generate` フォルダ - S3 バケット内の要約データからドキュメントを生成する Python コード
  - `code/lambdas/preprocess` フォルダ - Step Functions ステートマシンの入力を処理する Python コード
  - `code/lambdas/summarize` フォルダ - Amazon Bedrock サービスを使用して文字起こしされたデータを要約する Python コード
  - `code/lambdas/transcribe` フォルダ - Amazon Transcribe を使用して音声データ (オーディオファイル) をテキストに変換する Python コード
  - `code/lambdas/validate` フォルダ - すべての回答が同じトピックに関連しているかどうかを検証する Python コード
- `code/code_stack.py` – AWS リソースの作成に使用される AWS CDK コンストラクト Python ファイル
- `app.py` – ターゲット AWS アカウントに AWS リソースをデプロイするために使用される AWS CDK アプリの Python ファイル
- `requirements.txt` – AWS CDK にインストールする必要があるすべての Python 依存関係のリスト
- `cdk.json` – リソースの作成に必要な値を提供する入力ファイル

ベストプラクティス

提供されているコード例は、proof-of-concept (PoC) またはパイロットのみを目的としています。ソリューションを本稼働環境に移行する場合は、次のベストプラクティスを使用します。

- Amazon S3 アクセスログ記録を有効にする
- VPC フローログを有効にする

エピック

ローカルワークステーションで AWS 認証情報を設定する

タスク	説明	必要なスキル

アカウントと AWS リージョンの変数をエクスポートします。	環境変数を使用して AWS CDK の AWS 認証情報を提供するには、次のコマンドを実行します。
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	`export CDK_DEFAULT_REGION=<Region>`
		AWS DevOps、DevOps エンジニア

AWS CLI の名前付きプロファイルを設定します。	アカウントの AWS CLI 名前付きプロファイルを設定するには、「設定と認証情報ファイルの設定」の手順に従います。	AWS DevOps、DevOps エンジニア

環境をセットアップします。

タスク	説明	必要なスキル
リポジトリをローカルワークステーションにクローンします。	genai-knowledge-capture リポジトリのクローンを作成するには、ターミナルで次のコマンドを実行します。 `git clone https://github.com/aws-samples/genai-knowledge-capture`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
(オプション) オーディオファイルを置き換えます。	サンプルアプリケーションをカスタマイズして独自のデータを組み込むには、次の手順を実行します。 1. クローンされたリポジトリの `assets/audio_samples` フォルダに移動します。 2. サンプルオーディオファイルを含むフォルダを削除します。 3. 分析するトピックごとにフォルダを作成します。 4. オーディオファイルをそれぞれのフォルダに転送します。	AWS DevOps、DevOps エンジニア

Python 仮想環境をセットアップします。	Python 仮想環境をセットアップするには、次のコマンドを実行します。
	`cd genai-knowledge-capture`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	`pip install -r requirements.txt`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
AWS CDK コードを合成します。	コードを AWS CloudFormation スタック設定に変換するには、次のコマンドを実行します。 `cdk synth`	AWS DevOps、DevOps エンジニア

ソリューションの設定とデプロイ

タスク	説明	必要なスキル
基盤モデルアクセスをプロビジョニングします。	AWS アカウントの Anthropic Claude 3 Sonnet モデルへのアクセスを有効にします。手順については、Bedrock ドキュメントの「モデルアクセスの追加」を参照してください。	AWS DevOps
アカウントにリソースをデプロイします。	AWS CDK を使用して AWS アカウントにリソースをデプロイするには、次の手順を実行します。 1. (オプション) クローンされたリポジトリのルートで、 `app.py` ファイルで AWS CloudFormation スタック名を更新します。デフォルトのスタック名は ですgenai-knowledge-capture-stack 。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	2. リソースをデプロイするには、`cdk deploy` コマンドを実行します。 `cdk deploy` コマンドは Layer-3 コンストラクトを使用して、一連の Lambda 関数、S3 バケット、Amazon SNS トピック、Step Functions ステートマシンを作成します。`assets/audio_samples` フォルダ内のオーディオファイルは、デプロイ中に S3 バケットにコピーされます。 3. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/cloudformation/://www.com で CloudFormation コンソールを開きます。 4. スタックが正常にデプロイされたことを確認します。手順については、AWS CloudFormation コンソールでのスタックの確認」を参照してください。	

タスク	説明	必要なスキル
Amazon SNS トピックを購読します。	通知用に Amazon SNS トピックをサブスクライブするには、次の手順を実行します。 1. CloudFormation コンソールのナビゲーションペインで、スタックを選択します。 2. `genai-knowledge-capture-stack` スタックを選択します。 3. [出力] タブを選択します。 4. キー を使用して Amazon SNS トピック名を検索します`SNSTopicName` 。 5. Amazon SNSトピックに E メールアドレスをサブスクライブする」の手順に従って、通知を受信するように E メールアドレスを設定します。	AWS 全般

ソリューションをテストする

タスク	説明	必要なスキル
ステートマシンを実行します。	1. Step Functions コンソールを開きます。 2. ステートマシンページで、genai-knowledge-capture-stack-state-machine を選択します。	アプリ開発者、AWS 全般

タスク	説明	必要なスキル
	3. [実行のスタート] を選択します。 4. (オプション) 名前 ボックスに、実行の名前を入力します。 5. 入力エリアで、プレースホルダーテキストを置き換えて次の JSON オブジェクトを入力します。 • <Name> は、ドキュメントに名前を付けるものです。 • <S3 bucket name> は、オーディオファイルを含む Amazon S3 バケットの名前です。 • <Folder path> は、オーディオファイルを含むディレクトリです。 `{ "documentName": "<Name>", "audioFileFolderUri": "s3://<S3 bucket name>/<Folder path>" }` 6. [実行のスタート] を選択します。 7. 実行の詳細ページで、結果を確認し、実行が完了するまで待ちます。	

ソリューション内のすべての AWS リソースをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	ソリューションをテストしたら、リソースをクリーンアップします。 1. S3 バケットからすべてのオブジェクトを削除し、バケットを削除します。詳細については、バケットを削除するを参照してください。 2. クローンされたリポジトリから、 コマンドを実行しますcdk destroy。	AWS DevOps、DevOps エンジニア

関連リソース

AWS ドキュメント

- Amazon Bedrock リソース：
  - モデルアクセス
  - 基盤モデルの推論パラメータ
- AWS CDK リソース：
  - AWS CDK の使用を開始する
  - Python での AWS CDK の使用
  - AWS CDK の一般的な問題のトラブルシューティング
  - ツールキットコマンド
- AWS Step Functions リソース：
  - AWS Step Functions の開始方法
  - トラブルシューティング
- Python で Lambda 関数を構築する

- AWS で生成 AI Application Builder

その他のリソース

- LangChain ドキュメント

Amazon Personalize を使用して、パーソナライズされ再ランク付けされたレコメンデーションを生成します

作成者: メイソン・ケイヒル (AWS)、マシュー・シャッセ (AWS)、タヨ・オラジデ (AWS)

概要

このパターンは、Amazon Personalize を使用して、ユーザーからのリアルタイムのユーザーインタラクションデータの取り込みに基づいて、ユーザー向けにパーソナライズされたレコメンデーション (再ランク付けされたレコメンデーションを含む) を生成する方法を示しています。このパターンで使用されるシナリオの例は、ペット養子縁組ウェブサイトに基づいています。このウェブサイトでは、ユーザーとのやり取り (たとえば、ユーザーが訪問したペットなど) に基づいてユーザー向けのレコメンデーションが生成されます。シナリオ例に従うことで、Amazon Kinesis Data Streams を使用してインタラクションデータを取り込む方法、AWS Lambda を使用してレコメンデーションを生成してレコメンデーションを再ランク付けする方法、Amazon Data Firehose を使用して Amazon Simple Storage Service (Amazon S3) バケットにデータを保存する方法を学びます。また、AWS Step Functions を使用して、レコメンデーションを生成するソリューションバージョン (つまり、トレーニング済みモデル) を管理するステートマシンを構築する方法も学びます。

前提条件と制限

前提条件

- 「ブートストラップ」された AWS Cloud Development Kit (AWS CDK) を使用したアクティブな「AWS アカウント」
- 認証情報が設定された「AWS コマンドラインインターフェイス(AWS CLI)」
- 「Python 3.9」

製品バージョン

- Python 3.9
- AWS CDK 2.23.0 以降
- AWS CLI 2.7.27 以降

アーキテクチャ

テクノロジースタック

- Amazon Data Firehose
- Amazon Kinesis Data Streams
- Amazon Personalize
- Amazon Simple Storage Service (Amazon S3)
- AWS Cloud Development Kit (AWS CDK)
- AWS コマンドラインインターフェイス (AWS CLI)
- AWS Lambda
- AWS Step Functions

ターゲットアーキテクチャ

次の図は、Amazon Personalize にリアルタイムデータを取り込むためのパイプラインを示しています。次に、パイプラインはそのデータを使用して、ユーザー向けにパーソナライズされ、ランクが変更されたレコメンデーションを生成します。

この図表は、次のワークフローを示しています：

1. Kinesis Data Streams は、Lambda と Firehose による処理のために、リアルタイムのユーザーデータ (訪問したペットなどのイベント) を取り込みます。
2. Lambda 関数は Kinesis データストリームからのレコードを処理し、レコード内のユーザーインタラクションを Amazon Personalize のイベントトラッカーに追加するための API コールを行います。
3. 時間ベースのルールは Step Functions ステートマシンを呼び出し、Amazon Personalize のイベントトラッカーからのイベントを使用して、レコメンデーションモデルと再ランク付けモデルの新しいソリューションバージョンを生成します。
4. Amazon Personalize「キャンペーン」は、ステートマシンによって新しいソ「リューションバージョン」を使用するように更新されます。
5. Lambda は Amazon Personalize 再ランキングキャンペーンを呼び出して、おすすめ商品のリストを再ランク付けします。

6. Lambda は Amazon Personalize レコメンデーションキャンペーンを呼び出して、おすすめ商品のリストを取得します。
7. Firehose は、イベントを履歴データとしてアクセスできる S3 バケットに保存します。

ツール

AWS ツール

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Amazon Data Firehose は、サポートされているサードパーティーサービスプロバイダーが所有する他の AWS のサービス、カスタム HTTP エンドポイント、HTTP エンドポイントにリアルタイムのストリーミングデータを配信するのに役立ちます。
- 「Amazon Kinesis Data Streams」は、データレコードの大量のストリームをリアルタイムで収集し、処理するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Personalize」はフルマネージド型の機械学習 (ML) サービスで、データに基づいてユーザー向けの商品レコメンデーションを作成できます。
- 「AWS Step Functions」は、Lambda 関数と他の AWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。

その他のツール

- 「pytest」は、小さくて読みやすいテストを書くための Python フレームワークです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コード

このパターンのコードは、GitHub 内の「Animal Recommender」リポジトリで利用できます。このリポジトリの AWS CloudFormation テンプレートを使用して、サンプルソリューションのリソースをデプロイできます。

> **Note**
>
> Amazon Personalize ソリューションバージョン、イベントトラッカー、キャンペーンは、ネイティブ CloudFormation リソースを拡張するカスタムリソース (インフラストラクチャ内) によってバックアップされます。

エピック

インフラストラクチャを作成する

タスク	説明	必要なスキル
隔離された Python 環境を作成します。	Mac/Linux セットアップ 1. 仮想環境を手動で作成するには、ターミナルから `$ python3 -m venv .venv` コマンドを実行します。 2. init プロセスが完了したら、`$ source .venv/bin/activate` コマンドを実行して仮想環境をアクティブ化します。 Windows セットアップ 仮想環境を手動で作成するには、ターミナルから `% .venv\Scripts\activate.bat` コマンドを実行します。	DevOps エンジニア
CloudFormation のテンプレートを合成します。	1. 必要な従属関係をインストールするには、ターミナルから `$ pip install -`	DevOps エンジニア

タスク	説明	必要なスキル
	r requirements.txt コマンドを実行します。 2. AWS CLI で、次の環境変数を設定します。 • export ACCOUNT_ID=123456789 • export CDK_DEPLOY_REGION=us-east-1 • export CDK_ENVIRONMENT=dev 3. config/{env}.yml ファイル内で、仮想プライベートクラウド (VPC) ID と一致するように vpcId を更新します。 4. このコードの CloudFormation テンプレートを合成するには、$ cdk synth コマンドを実行します。 Note ステップ 2 では、 は config/{env}.yml ファイルCDK_ENVIRONMENT を参照します。	

タスク	説明	必要なスキル
リソースをデプロイし、インフラストラクチャーを構築します。	ソリューションリソースをデプロイするには、ターミナルから `./deploy.sh` コマンドを実行します。 このコマンドは、必要な Python 依存関係をインストールします。Python スクリプトは、S3 バケットと AWS Key Management Service (AWS KMS) キーを作成し、最初にモデルを作成するためのシードデータを追加します。最後に、スクリプトは `cdk deploy` を実行し、残りのインフラストラクチャを作成します。 **Note** 初期モデルトレーニングは、スタックの作成中に行われます。スタックの作成が完了するまでに、最大 2 時間かかることがあります。	DevOps エンジニア

関連リソース

- 「アニマルレコメンダー」(GitHub)
- 「AWS CDK リファレンスドキュメント」
- 「Boto3 ドキュメント」

- 「Amazon Personalize で選択したビジネス指標に最適化されたパーソナライズされたレコメンデーションを最適化」(AWS 機械学習 ブログ)。

追加情報

ペイロードとレスポンスの例

Lambda 関数の推奨事項

レコメンデーションを取得するには、以下の形式のペイロードを含むレコメンデーション Lambda 関数にリクエストを送信します。

```
{
  "userId": "3578196281679609099",
  "limit": 6
}
```

次のレスポンス例には、アニマルグループのリストが含まれています。

```
[{"id": "1-domestic short hair-1-1"},
{"id": "1-domestic short hair-3-3"},
{"id": "1-domestic short hair-3-2"},
{"id": "1-domestic short hair-1-2"},
{"id": "1-domestic short hair-3-1"},
{"id": "2-beagle-3-3"},
```

`userId` フィールドを省略すると、関数は一般的な推奨事項を返します。

Lambda 関数の再ランク付け

再ランク付けを使用するには、再ランク付けの Lambda 関数にリクエストを送信します。ペイロードには、再ランク付けの対象となるすべてのアイテム ID の `userId` とそのメタデータが含まれます。以下のデータ例では、`animal_species_id` (1=猫、2=犬) には Oxford Pets クラスを使用し、`animal_age_id` および `animal_size_id` には 1 ~ 5 の整数を使用しています。

```
{
   "userId":"12345",
   "itemMetadataList":[
      {
         "itemId":"1",
```

```
            "animalMetadata":{
               "animal_species_id":"2",
               "animal_primary_breed_id":"Saint_Bernard",
               "animal_size_id":"3",
               "animal_age_id":"2"
            }
         },
         {
            "itemId":"2",
            "animalMetadata":{
               "animal_species_id":"1",
               "animal_primary_breed_id":"Egyptian_Mau",
               "animal_size_id":"1",
               "animal_age_id":"1"
            }
         },
         {
            "itemId":"3",
            "animalMetadata":{
               "animal_species_id":"2",
               "animal_primary_breed_id":"Saint_Bernard",
               "animal_size_id":"3",
               "animal_age_id":"2"
            }
         }
      ]
}
```

Lambda 関数はこれらの項目を再ランク付けし、項目 ID と Amazon Personalize からのダイレクトレスポンスを含む順序付きリストを返します。これは、商品が属するアニマルグループとそのスコアをランク付けしたリストです。Amazon Personalize は、「ユーザーパーソナライズ」と「パーソナライズランキング」のレシピを使用して、各項目のスコアをレコメンデーションに含めます。これらのスコアは、ユーザーが次にどのアイテムを選ぶかについて、Amazon Personalize の相対的な確実性を表します。スコアが高いほど、確実性が高くなります。

```
{
   "ranking":[
      "1",
      "3",
      "2"
   ],
   "personalizeResponse":{
```

```
      "ResponseMetadata":{
         "RequestId":"a2ec0417-9dcd-4986-8341-a3b3d26cd694",
         "HTTPStatusCode":200,
         "HTTPHeaders":{
            "date":"Thu, 16 Jun 2022 22:23:33 GMT",
            "content-type":"application/json",
            "content-length":"243",
            "connection":"keep-alive",
            "x-amzn-requestid":"a2ec0417-9dcd-4986-8341-a3b3d26cd694"
         },
         "RetryAttempts":0
      },
      "personalizedRanking":[
         {
            "itemId":"2-Saint_Bernard-3-2",
            "score":0.8947961
         },
         {
            "itemId":"1-Siamese-1-1",
            "score":0.105204
         }
      ],
      "recommendationId":"RID-d97c7a87-bd4e-47b5-a89b-ac1d19386aec"
   }
}
```

Amazon Kinesis スペイロード

Amazon Kinesis に送信するペイロードの形式は次のとおりです。

```
{
    "Partitionkey": "randomstring",
    "Data": {
        "userId": "12345",
        "sessionId": "sessionId4545454",
        "eventType": "DetailView",
        "animalMetadata": {
            "animal_species_id": "1",
            "animal_primary_breed_id": "Russian_Blue",
            "animal_size_id": "1",
            "animal_age_id": "2"
        },
        "animal_id": "98765"
```

```
    }
}
```

> **Note**
>
> 認証されていないユーザーの `userId` フィールドは削除されます。

SageMaker AI と Hybrid を使用して、ローカル開発からスケーラブルな実験まで機械学習ワークフローを合理化する

作成者: David Sauerwein (AWS)、Julian Ferdinand Grueber (AWS)、Marco Geiger (AWS)

概要

このパターンは、ローカルテストから Amazon SageMaker AI での本番稼働まで、機械学習 (ML) アルゴリズムを設定および実行する統一されたアプローチを提供します。ML アルゴリズムはこのパターンの焦点ですが、そのアプローチは特徴量エンジニアリング、推論、ML パイプライン全体に及びます。このパターンは、サンプルユースケースを通じて、ローカルスクリプト開発から SageMaker AI トレーニングジョブへの移行を示しています。

一般的な ML ワークフローは、ローカルマシンでソリューションを開発およびテストし、クラウドで大規模な実験 (異なるパラメータなど) を実行し、承認されたソリューションをクラウドにデプロイすることです。次に、デプロイされたソリューションをモニタリングして維持する必要があります。このワークフローに対する統一されたアプローチがない場合、デベロッパーは多くの場合、各段階でコードをリファクタリングする必要があります。ソリューションがこのワークフローのどの段階でも変更される可能性のある多数のパラメータに依存する場合、整理と一貫性を維持することがますます難しくなる可能性があります。

このパターンは、これらの課題に対処します。まず、ローカルマシン、コンテナ、または SageMaker AI のいずれで実行しても一貫性を保つ統一されたワークフローを提供することで、環境間でコードリファクタリングを行う必要がなくなります。次に、各実行の設定の自動ログ記録を使用して、パラメータを簡単に変更および結合できる個別の設定ファイルでパラメータを定義する、Hydroid の設定システムによるパラメータ管理を簡素化します。このパターンがこれらの課題にどのように対処するかの詳細については、「追加情報」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- SageMaker AI トレーニングジョブをデプロイおよび開始するための AWS Identity and Access Management (IAM) ユーザーロール
- AWS Command Line Interface （ AWS CLI) バージョン 2.0 以降がインストールされ、設定されている
- Poetry バージョン 1.8 以降、2.0 より前、インストール済み

- Docker がインストールされました
- Python バージョン 3.10.x

制約事項

- コードは現在、SageMaker AI トレーニングジョブのみを対象としています。処理ジョブや SageMaker AI パイプライン全体に拡張するのは簡単です。
- 完全に実稼働化された SageMaker AI のセットアップでは、追加の詳細を設定する必要があります。例としては、コンピューティングとストレージの custom AWS Key Management Service (AWS KMS) キーや、ネットワーク設定などがあります。これらの追加オプションは、フォルダの専用サブフォルダで、Hyraid を使用して設定することもできますconfig。
- 一部の AWS のサービス は では使用できません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、 ソリューションのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. データサイエンティストは、Docker や SageMaker AI を必要とせずに、ローカル環境でアルゴリズムを小規模に反復処理し、パラメータを調整し、トレーニングスクリプトを迅速にテストできます。(詳細については、「エピックで」タスクの「クイックテストをローカルで実行する」を参照してください。)
2. アルゴリズムに満足すると、データサイエンティストは Docker イメージを構築し、 という名前の Amazon Elastic Container Registry (Amazon ECR) リポジトリにプッシュしますhydra-sm-artifact。(詳細については、「エピック」のSageMaker AI でワークフローを実行する」を参照してください) 。 ???
3. データサイエンティストは、Python スクリプトを使用して SageMaker AI トレーニングジョブまたはハイパーパラメータ最適化 (HPO) ジョブを開始します。通常のトレーニングジョブの場合、調整された設定は という名前の Amazon Simple Storage Service (Amazon S3) バケットに書き込まれますhydra-sample-config。HPO ジョブの場合、 configフォルダにあるデフォルト設定セットが適用されます。

4. SageMaker AI トレーニングジョブは Docker イメージをプルし、Amazon S3 バケット から入力データを読み取りhydra-sample-data、Amazon S3 バケットから設定を取得するhydra-sample-configか、デフォルト設定を使用します。トレーニング後、ジョブは出力データを Amazon S3 バケット に保存しますhydra-sample-data。

自動化とスケール

- 自動トレーニング、再トレーニング、推論では、 AWS CLI コードを AWS Lambda、、AWS CodePipelineAmazon EventBridge などのサービスと統合できます。
- スケーリングは、インスタンスサイズの設定を変更するか、分散トレーニングの設定を追加することで実現できます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を使用して を操作するのに役立つオープンソースツールです。このパターンでは、 AWS CLI は初期リソース設定とテストの両方に役立ちます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon SageMaker AI は、ML モデルを構築およびトレーニングし、本番環境対応のホスト環境にデプロイするのに役立つマネージド機械学習 (ML) サービスです。SageMaker AI Training は、大規模な ML モデルのトレーニングを可能にする SageMaker AI 内のフルマネージド ML サービスです。このツールは、組み込みのスケーラビリティと他の との統合を利用して、トレーニングモデルの計算上の需要を効率的に処理できます AWS のサービス。SageMaker AI Training はカスタムアルゴリズムとコンテナもサポートしているため、幅広い ML ワークフローに柔軟に対応できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。このパターンでは、開発からデプロイまで、さまざまなステージにわたって一貫した環境を確保し、依存関係とコードを確実にパッケージ化するために使用されました。Docker のコンテナ化により、ワークフロー全体のスケーリングとバージョン管理が容易になります。
- Invalid は、複数の設定と動的リソース管理を柔軟に処理できる設定管理ツールです。環境設定の管理に役立ち、さまざまな環境にシームレスにデプロイできます。詳細については、「追加情報」を参照してください。
- 「Python」は汎用のコンピュータープログラミング言語です。Python を使用して ML コードとデプロイワークフローを記述しました。
- Poetry は、Python での依存関係の管理とパッケージングのためのツールです。

コードリポジトリ

このパターンのコードは、GitHub configuring-sagemaker-training-jobs-with-hydra リポジトリで入手できます。

ベストプラクティス

- 最小権限の原則に従って SageMaker AI トレーニングジョブをデプロイおよび開始するための IAM ロールを選択し、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。
- 一時的な認証情報を使用して、ターミナルの IAM ロールにアクセスします。

エピック

環境をセットアップする

タスク	説明	必要なスキル
仮想環境を作成してアクティブ化します。	仮想環境を作成してアクティブ化するには、リポジトリのルートで次のコマンドを実行します。	AWS 全般

タスク	説明	必要なスキル
	`poetry install` `poetry shell`	
インフラストラクチャを準備します。	CloudFormation を使用してインフラストラクチャをデプロイするには、次のコマンドを実行します。 `aws cloudformation deploy --template-file infra/hydra-sagemaker-setup.yaml --stack-name hydra-sagemaker-setup --capabilities CAPABILITY_NAMED_IAM`	AWS 全般、DevOps エンジニア
サンプルデータをダウンロードします。	openml からローカルマシンに入力データをダウンロードするには、次のコマンドを実行します。 `python scripts/download_data.py`	AWS 全般

タスク	説明	必要なスキル
クイックテストのためにローカルで を実行します。	テスト用にトレーニングコードをローカルで実行するには、次のコマンドを実行します。 `python mypackage/train.py data.train_data_path=data/train.csv evaluation.base_dir_path=data` すべての実行のログは、実行時間ごとに というフォルダに保存されますoutputs。詳細については、GitHub リポジトリの「出力」セクションを参照してください。 `--multirun` 機能を使用して、パラメータが異なる複数のトレーニングを並行して実行することもできます。詳細については、「 のドキュメント」を参照してください。	データサイエンティスト

SageMaker AI でワークフローを実行する

タスク	説明	必要なスキル
環境変数を設定します。	SageMaker AI でジョブを実行するには、 AWS リージョンと AWS アカウント ID を指定して、次の環境変数を設定します。	AWS 全般

タスク	説明	必要なスキル

	`export ECR_REPO_NAME=hydra-sm-artifact`
	`export image_tag=latest`
	`export AWS_REGION="<your_aws_region>" # for instance, us-east-1`
	`export ACCOUNT_ID="<your_account_id>"`
	`export BUCKET_NAME_DATA=hydra-sample-data-$ACCOUNT_ID`
	`export BUCKET_NAME_CONFIG=hydra-sample-config-$ACCOUNT_ID`
	`export AWS_DEFAULT_REGION=$AWS_REGION`
	`export ROLE_ARN=arn:aws:iam::${ACCOUNT_ID}:role/hydra-sample-sagemaker`
	`export INPUT_DATA_S3_PATH=s3://$BUCKET_NAME_DATA/hydra-on-sm/input/`
	`export OUTPUT_DATA_S3_PATH=s3://$BUCKET_NAME_DATA/hydra-on-sm/output/`	

タスク	説明	必要なスキル

Docker イメージを作成してプッシュします。	Docker イメージを作成して Amazon ECR リポジトリにプッシュするには、次のコマンドを実行します。
	
	`chmod +x scripts/create_and_push_image.sh`
	`scripts/create_and_push_image.sh $ECR_REPO_NAME $image_tag $AWS_REGION $ACCOUNT_ID`
	
	このタスクは、 環境に有効な認証情報があることを前提としています。Docker イメージは、前のタスクの環境変数で指定された Amazon ECR リポジトリにプッシュされ、トレーニングジョブが実行される SageMaker AI コンテナをアクティブ化するために使用されます。
		ML エンジニア、AWS 全般

入力データを Amazon S3 にコピーします。	SageMaker AI トレーニングジョブは、入力データを取得する必要があります。データの入力データを Amazon S3 バケットにコピーするには、次のコマンドを実行します。 `aws s3 cp data/train.csv "${INPUT_DATA_S3_PATH}train.csv"`	データエンジニア、AWS 全般

タスク	説明	必要なスキル
SageMaker AI トレーニングジョブを送信します。	スクリプトの実行を簡素化するには、 `default.yaml` ファイルでデフォルトの設定パラメータを指定します。このアプローチは、実行間の一貫性を確保するだけでなく、必要に応じてデフォルト設定を簡単に上書きする柔軟性も提供します。次の例を参照してください。 `python scripts/start_sagemaker_training_job.py sagemaker.role_arn=$ROLE_ARN sagemaker.config_s3_bucket=$BUCKET_NAME_CONFIG sagemaker.input_data_s3_path=$INPUT_DATA_S3_PATH sagemaker.output_data_s3_path=$OUTPUT_DATA_S3_PATH`	AWS 全般、ML エンジニア、データサイエンティスト

タスク	説明	必要なスキル
SageMaker AI ハイパーパラメータチューニングを実行します。	SageMaker AI ハイパーパラメータチューニングの実行は、SageMaker AII トレーニングジョブの送信と似ています。ただし、実行スクリプトは、start_sagemaker_hpo_job.py ファイルに表示されるように、いくつかの重要な点で異なります。チューニングするハイパーパラメータは、トレーニングジョブへのチャネルではなく、boto3 ペイロードを通過する必要があります。 ハイパーパラメータ最適化 (HPO) ジョブを開始するには、次のコマンドを実行します。 `python scripts/start_sagemaker_hpo_job.py sagemaker.role_arn=$ROLE_ARN sagemaker.config_s3_bucket=$BUCKET_NAME_CONFIG sagemaker.input_data_s3_path=$INPUT_DATA_S3_PATH sagemaker.output_data_s3_path=$OUTPUT_DATA_S3_PATH`	データサイエンティスト

トラブルシューティング

問題	ソリューション
期限切れのトークン	新しい AWS 認証情報をエクスポートします。
IAM アクセス許可の欠如	CloudFormation テンプレートをデプロイし、SageMaker AI トレーニングジョブを開始するために必要なすべての IAM アクセス許可を持つ IAM ロールの認証情報をエクスポートしてください。

関連リソース

- Amazon SageMaker AI を使用してモデルをトレーニングする (AWS ドキュメント)
- ハイパーパラメータチューニングとは

追加情報

このパターンでは、以下の課題に対処します。

ローカル開発から大規模なデプロイまでの一貫性 – このパターンでは、開発者はローカル Python スクリプトを使用しているか、ローカル Docker コンテナを実行しているか、SageMaker AI で大規模な実験を実行しているか、SageMaker AI で本番環境にデプロイしているかに関係なく、同じワークフローを使用できます。この一貫性は、以下の理由で重要です。

- イテレーションの高速化 – スケールアップ時に大幅な調整を必要とせずに、ローカルでの実験を高速化できます。
- リファクタリングなし – SageMaker AI での大規模な実験への移行はシームレスで、既存のセットアップの見直しは必要ありません。
- 継続的な改善 – コードは環境間で同じであるため、新機能の開発とアルゴリズムの継続的な改善は簡単です。

設定管理 - このパターンでは、設定管理ツールであるhttps://hydra.cc/「Hyrady」を活用して、次の利点を提供します。

- パラメータは、コードとは別に設定ファイルで定義されます。
- さまざまなパラメータセットを簡単にスワップまたは組み合わせることができます。
- 実験の追跡は、各実行の設定が自動的にログに記録されるため、簡素化されます。
- クラウド実験では、ローカル実行と同じ設定構造を使用して一貫性を確保できます。

Invalid を使用すると、設定を効果的に管理し、次の機能を有効にできます。

- 設定を分割する – プロジェクト設定を小さく管理しやすい部分に分割し、個別に変更できます。このアプローチにより、複雑なプロジェクトを簡単に処理できます。
- デフォルトを簡単に調整 – ベースライン設定をすばやく変更できるため、新しいアイデアをより簡単にテストできます。
- CLI 入力と設定ファイルの整列 – コマンドライン入力を設定ファイルとスムーズに組み合わせます。このアプローチにより、乱雑な作業や混乱が軽減され、時間の経過とともにプロジェクトをより管理しやすくなります。

GPU がサポートするカスタム ML モデルを Amazon SageMaker でトレーニングしてデプロイする

作成者:Ankur Shukla (AWS)

概要

グラフィックプロセッシングユニット (GPU) がサポートする機械学習 (ML) モデルをトレーニングしてデプロイするには、NVIDIA GPU の利点を最大限に引き出すために、特定の環境変数の初期設定と初期化が必要です。ただし、環境をセットアップし、Amazon Web Services (AWS) クラウド上の Amazon SageMaker アーキテクチャと互換性を持たせるには時間がかかる場合があります。

このパターンは、Amazon SageMaker を使用して GPU をサポートするカスタム ML モデルをトレーニングおよび構築するのに役立ちます。オープンソースの Amazon Reviews データセットに基づいて構築されたカスタム CatBoost モデルをトレーニングしてデプロイする手順を示します。その後、`p3.16xlarge` Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでパフォーマンスをベンチマークできます。

このパターンは、組織が GPU 対応の既存の ML モデルを SageMaker にデプロイしたい場合に役立ちます。データサイエンティストは、このパターンの手順に従って NVIDIA GPU 対応コンテナを作成し、それらのコンテナに ML モデルをデプロイできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- モデルのアーティファクトと予測を保存する Amazon Simple Storage Service (Amazon S3) ソースバケット。
- SageMaker ノートブックインスタンスと Jupyter notebook についての理解。
- 基本的な SageMaker ロール権限、S3 バケットへのアクセス権と更新権限、および Amazon Elastic Container Registry (Amazon ECR) の追加権限を含む AWS アイデンティティおよびアクセス管理 (IAM) ロールを作成する方法の理解。

機能制限

- このパターンは、Python で記述されたコードをトレインしてデプロイする教師付き ML ワークロードを対象としています。

アーキテクチャ

テクノロジースタック

- SageMaker
- Amazon ECR

ツール

ツール

- 「Amazon ECR」— Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。
- 「Amazon SageMaker」— Amazon SageMaker は、フルマネージド型の ML サービスです。
- 「Docker」— Docker は、アプリケーションを迅速に構築、テスト、デプロイするためのソフトウェアプラットフォームです。
- 「Python」— Python はプログラミング言語です。

コード

このパターンのコードは、GitHub 内の「Catboost と SageMaker によるレビュー分類モデルの実装」リポジトリで利用できます。

エピック

データを準備する

タスク	説明	必要なスキル
IAM ロールを作成して、必要なポリシーをアタッチします。	AWS マネジメントコンソールにサインインし、IAM コンソール を開き、新しいIAM ロールを作成します。次のポリシーを IAM ロールにアタッチします。	データサイエンティスト

タスク	説明	必要なスキル
	• AmazonEC2ContainerRegistryFullAccess • AmazonS3FullAccess • AmazonSageMakerFullAccess 詳細については、Amazon SageMaker のドキュメントの「ノートブックインスタンスの作成」を参照してください。	
SageMaker ノートブックインスタンスを作成します。	SageMaker コンソールを開き、[ノートブックインスタンス] を選択し、[ノートブックインスタンスの作成] を選択します。[IAM ロール] には、前に作成したIAM ロールを選択します。要件に従ってノートブックインスタンスを設定し、[ノートブックインスタンスを作成] を選択します。 詳細な手順と手順については、Amazon SageMaker ドキュメントの「ノートブックインスタンスの作成」を参照してください。	データサイエンティスト

タスク	説明	必要なスキル
リポジトリをクローン作成します。	SageMaker ノートブックインスタンスでターミナルを開き、以下のコマンドを実行して GitHub の「Catboost と SageMaker リポジトリによるによるレビュー分類モデルの実装」リポジトリをクローンします。 `git clone https://github.com/aws-samples/review-classification-using-catboost-sagemaker.git`	
Jupyter Notebook を開始します。	定義済みのステップが含まれている `Review classification model with Catboost and SageMaker.ipynb` Jupyter notebook を起動します。	データサイエンティスト

特徴量エンジニアリング

タスク	説明	必要なスキル
Jupyter notebook でコマンドを実行します。	Jupyter notebook を開き、以下のストーリーのコマンドを実行して ML モデルをトレーニングするためのデータを準備します。	データサイエンティスト
S3 バケットからデータを読み取ります。	`import pandas as pd` `import csv`	データサイエンティスト

タスク	説明	必要なスキル
	fname = 's3://amazon-reviews-pds/tsv/amazon_reviews_us_Digital_Video_Download_v1_00.tsv.gz' df = pd.read_csv(fname, sep='\t',delimiter='\t',error_bad_lines=False)	

タスク	説明	必要なスキル

データを前処理します。	`import numpy as np`
	`def pre_process(df):`
	
	`  df.fillna(value={'`
	`review_body': '',`
	`'review_headline':`
	`''}, inplace=True)`
	`df.fillna( value={'v`
	`erified_purchase':`
	`'Unk'}, inplace=True)`
	
	`df.fillna(0,`
	`inplace=True)`
	`return df`
	`df = pre_process(df)`
	`df.review_date =`
	`pd.to_datetime(df.`
	`review_date)`
	`df['target'] =`
	`np.where(df['star_`
	`rating']>=4,1,0)`
	
	Note
	このコードは、 の null `'review_body'` 値を空の文字列に置き換え、`'verified_purchase'` 列を「不明`'Unk'`」という意味の に置き換えます。
		データサイエンティスト

タスク	説明	必要なスキル

データをトレーニング、検証、テストデータセットに分割します。	ターゲットラベルの分布を分割セット全体で同じに保つには、「scikit-learn ライブラリ」を使用してサンプリングを層別化する必要があります。
	
	`from sklearn.model_selection import StratifiedShuffleSplit`
	`sss = StratifiedShuffleSplit(n_splits=2, test_size=0.10, random_state=0)`
	`sss.get_n_splits(df, df['target'])`
	`for train_index, test_index in sss.split(df, df['target']):`
	`  X_train_vallid , X_test = df.iloc[train_index], df.iloc[test_index]`
	
	`sss.get_n_splits(X_train_vallid, X_train_vallid['target'])`
	`for train_index, test_index in sss.split(X_train_vallid, X_train_vallid['target']):`
	`  X_train , X_valid = X_train_vallid.iloc[train_index],`
		データサイエンティスト

タスク	説明	必要なスキル
	`X_train_vallid.ilo c[test_index]`	

Docker イメージをビルド、実行し、Amazon ECR にプッシュします。

タスク	説明	必要なスキル
Docker イメージを準備してプッシュします。	Jupyter notebook で、以下のストーリーのコマンドを実行して Docker イメージを準備し、Amazon ECR にプッシュします。	ML エンジニア

Amazon ECR でリポジトリを作成します。	`%%sh`
	
	`algorithm_name=sagemaker-catboost-github-gpu-img`
	
	`chmod +x code/train`
	`chmod +x code/serve`
	
	`account=$(aws sts get-caller-identity --query Account --output text)`
	
	`# Get the region defined in the current configuration (default to us-west-2 if none defined)`
	`region=$(aws configure get region)`
	`region=${region:-us-east-1}`
		ML エンジニア

タスク	説明	必要なスキル

	`fullname="${accou nt}.dkr.ecr.${regi on}.amazonaws.com/ ${algorithm_name}: latest"`
	
	`aws ecr create-re pository --repository- name "${algorithm_name} " > /dev/nul`	

Docker イメージをローカルで構築します。	`docker build -t "${algorithm_name}" . docker tag ${algorit hm_name} ${fullname}`	ML エンジニア
Docker イメージを実行し、Amazon ECR にプッシュします。	`docker push ${fullname}`	ML エンジニア

トレーニング

タスク	説明	必要なスキル
SageMaker ハイパーパラメータ調整ジョブを作成します。	Jupyter notebook で、以下のストーリーのコマンドを実行して、Docker イメージを使用して SageMaker ハイパーパラメータチューニングジョブを作成します。	データサイエンティスト
SageMaker 見積りを作成します。	Docker イメージの名前を使用して「SageMaker エスティメータ」作成します。 `import sagemaker as sage`	データサイエンティスト

タスク	説明	必要なスキル
	from time import gmtime, strftime sess = sage.Session() from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, HyperparameterTuner account = sess.boto_session.client('sts').get_caller_identity()['Account'] region = sess.boto_session.region_name image = '{}.dkr.ecr.{}.amazonaws.com/sagemaker-catboost-github-gpu-img:latest'.format(account, region) tree_hpo = sage.estimator.Estimator(image, role, 1, 'ml.p3.16xlarge', train_volume_size = 100, output_path="s3://{}/sagemaker/DEMO-GPU-Catboost/output".format(bucket), sagemaker_session=sess)	

タスク	説明	必要なスキル

HPO ジョブを作成します。	パラメータ範囲を含むハイパーパラメーター最適化 (HPO) チューニングジョブを作成し、トレインセットと検証セットをパラメーターとして関数に渡します。
	
	`hyperparameter_ranges = {'iterations': IntegerParameter(80000, 130000),`
	
	`'max_depth': IntegerParameter(6, 10),`
	
	`'max_ctr_complexity': IntegerParameter(4, 10),`
	
	`'learning_rate': ContinuousParameter(0.01, 0.5)}`
	
	`objective_metric_name = 'auc'`
	`metric_definitions = [{'Name': 'auc',`
	
	`'Regex': 'auc: ([0-9\.]+)'}]`
	
	`tuner = HyperparameterTuner(tree_hpo,`
	
	`objective_metric_name,`
	
	`hyperparameter_ranges,`
		データサイエンティスト

タスク	説明	必要なスキル

	`metric_definitions,`
	
	`objective_type='Maximize',`
	
	`max_jobs=50,`
	
	`max_parallel_jobs=2)`	

HPO ジョブを実行します。	`train_location = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/train/'`
	`valid_location = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/valid/'`
	
	`tuner.fit({'train': train_location,`
	`'validation': valid_location`
	`})`
		データサイエンティスト

最もパフォーマンスの高いトレーニングジョブを受けます。	`import sagemaker as sage`
	`from time import gmtime, strftime`
	`sess = sage.Session()`
	
	`best_job =tuner.best_training_job()`
		データサイエンティスト

バッチ変換

タスク	説明	必要なスキル
モデル予測用のテストデータで SageMaker バッチ変換ジョブを作成します。	Jupyter notebook で、以下のストーリーのコマンドを実行して SageMaker ハイパーパラメータチューニングジョブからモデルを作成し、モデル予測用のテストデータに SageMaker バッチ変換ジョブを送信します。	データサイエンティスト

SageMaker モデルを作成します。	最適なトレーニングジョブを使用して、SageMaker モデルでモデルを作成します。
	
	`attached_estimator = sage.estimator.Estimator.attach(best_job)`
	
	`output_path ='s3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test-predictions/'`
	`input_path ='s3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test/'`
	
	`transformer = attached_estimator.transformer(instance_count=1,`
	
	`instance_type='ml.p3.16xlarge',`
		データサイエンティスト

タスク	説明	必要なスキル

	<pre>assemble_with='Lin
	e',
	
	accept='text/csv',
	
	max_payload=1,
	
	output_path=output
	_path,
	
	env
	= {'SAGEMAKER_MODEL_
	SERVER_TIMEOUT' :
	'3600' })</pre>	

バッチ変換ジョブを作成します。	テストデータセットにバッチ変換ジョブを作成します。
	<pre>transformer.transf
	orm(input_path,
	
	content_type='text/
	csv',
	
	split_type='Line')</pre>
		データサイエンティスト

結果を分析する

タスク	説明	必要なスキル
結果を読み込み、モデルのパフォーマンスを評価する。	Jupyter notebook では、以下のストーリーのコマンドを実行して結果を読み取り、ROC	データサイエンティスト

タスク	説明	必要なスキル
	曲線下面積 (ROC-AUC) と精度再現曲線下面積 (PR-AUC) モデルメトリクスでモデルのパフォーマンスを評価します。 詳細については、Amazon Machine Learning (Amazon ML) ドキュメントの「Amazon 機械学習の主要概念」を参照してください。	

バッチ変換ジョブの結果をお読みください。	バッチ変換ジョブの結果をデータフレームに読み込みます。
	
	`file_name = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test-predictions/file_1.out'`
	
	`results = pd.read_csv(file_name, names=['review_id','target','score'] ,sep='\t',escapechar ='\\' , quoting=csv.QUOTE_NONE,`
	
	`lineterminator='\n',quotechar='"').dropna()`
		データサイエンティスト

タスク	説明	必要なスキル

パフォーマンスメトリクスを評価します。	ROC-AUC と PR-AUC でのモデルのパフォーマンスを評価します。
	
	`from sklearn import metrics`
	`import matplotlib`
	`import pandas as pd`
	`matplotlib.use('agg', warn=False, force=True)`
	`from matplotlib import pyplot as plt`
	
	`%matplotlib inline`
	
	`def analyze_results(labels, predictions):`
	`precision, recall, thresholds = metrics.precision_recall_curve(labels, predictions)`
	`auc = metrics.auc(recall, precision)`
	
	`fpr, tpr, _ = metrics.roc_curve(labels, predictions)`
	`roc_auc_score = metrics.roc_auc_score(labels, predictions)`
	
	`print('Neural-Nets: ROC auc=%.3f' % ( roc_auc_score))`
	
	`plt.plot(fpr, tpr, label="data 1, auc=" + str(roc_auc_score))`
		データサイエンティスト

タスク	説明	必要なスキル

	`    plt.xlabel('1-Specificity')`
	`    plt.ylabel('Sensitivity')`
	`    plt.legend(loc=4)`
	`    plt.show()`
	
	`    lr_precision, lr_recall, _ = metrics.precision_recall_curve(labels, predictions)`
	`    lr_auc = metrics.auc(lr_recall, lr_precision)`
	`    # summarize scores`
	`    print('Neural-Nets: PR auc=%.3f' % ( lr_auc))`
	`    # plot the precision-recall curves`
	`    no_skill = len(labels[labels==1.0]) / len(labels)`
	`    plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')`
	
	`    plt.plot(lr_recall, lr_precision, marker='.', label='Neural-Nets')`
	`    # axis labels`
	`    plt.xlabel('Recall')`
	`    plt.ylabel('Precision')`
	`    # show the legend`
	`    plt.legend()`
	`    # show the plot`	

タスク	説明	必要なスキル
	plt.show() return auc analyze_results(results['target'].values,results['score'].values)	

関連リソース

- 「Scikit Docker コンテナを構築して Amazon SageMaker で Scikit-Learn モデルをトレーニングしてホストする」

追加情報

以下のリストは、Docker イメージの [ビルド、実行、および Amazon ECR エピックへのプッシュ] で実行される Dockerfile のさまざまな要素を示しています。

aws-cli を使用して Python をインストールします。

```
FROM amazonlinux:1

 RUN yum update -y && yum install -y python36 python36-devel python36-libs python36-tools python36-pip && \
 yum install gcc tar make wget util-linux kmod man sudo git -y && \
 yum install wget -y && \
 yum install aws-cli -y && \
 yum install nginx -y && \
yum install gcc-c++.noarch -y && yum clean all
```

Python パッケージのインストール

```
RUN pip-3.6 install --no-cache-dir --upgrade pip && \pip3 install --no-cache-dir --upgrade setuptools && \
```

```
pip3 install Cython && \
pip3 install --no-cache-dir numpy==1.16.0 scipy==1.4.1 scikit-learn==0.20.3
 pandas==0.24.2 \
flask gevent gunicorn boto3 s3fs  matplotlib joblib catboost==0.20.2
```

CUDA と CuDNN のインストー

```
RUN wget https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/
cuda_9.0.176_384.81_linux-run \
&& chmod u+x cuda_9.0.176_384.81_linux-run \
&& ./cuda_9.0.176_384.81_linux-run --tmpdir=/data --silent --toolkit --override \
&& wget https://custom-gpu-sagemaker-image.s3.amazonaws.com/installation/cudnn-9.0-
linux-x64-v7.tgz \
&& tar -xvzf cudnn-9.0-linux-x64-v7.tgz \
&& cp /data/cuda/include/cudnn.h /usr/local/cuda/include \
&& cp /data/cuda/lib64/libcudnn* /usr/local/cuda/lib64 \

&& chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn* \
&&  rm -rf /data/*
```

SageMaker に必要なディレクトリ構造を作成します。

```
RUN mkdir /opt/ml /opt/ml/input /opt/ml/input/config /opt/ml/input/data /opt/ml/input/
data/training /opt/ml/model /opt/ml/output /opt/program
```

NVIDIA 環境変数を設定します。

```
ENV PYTHONPATH=/opt/program
ENV PYTHONUNBUFFERED=TRUE
ENV PYTHONDONTWRITEBYTECODE=TRUE
ENV PATH="/opt/program:${PATH}"

# Set NVIDIA mount environments
ENV LD_LIBRARY_PATH=/usr/local/nvidia/lib:/usr/local/nvidia/lib64:$LD_LIBRARY_PATH
ENV NVIDIA_VISIBLE_DEVICES="all"
ENV NVIDIA_DRIVER_CAPABILITIES="compute,utility"
ENV NVIDIA_REQUIRE_CUDA "cuda>=9.0"
```

トレーニングファイルと推論ファイルを Docker イメージにコピーする

```
COPY code/* /opt/program/
```

```
WORKDIR /opt/program
```

自然言語を OpenSearch クエリと Elasticsearch クエリのクエリ DSL に変換する

作成者: Tabby Ward (AWS)、Nicholas Switzer (AWS)、Breanne Warner (AWS)

概要

このパターンは、大規模言語モデル (LLMs) を使用して自然言語クエリをクエリドメイン固有の言語 (クエリ DSL) に変換する方法を示しています。これにより、ユーザーはクエリ言語の広範な知識を持たずに OpenSearch や Elasticsearch などの検索サービスとやり取りしやすくなります。このリソースは、自然言語クエリ機能を使用して検索ベースのアプリケーションを強化し、最終的にユーザーエクスペリエンスと検索機能を向上させたい開発者やデータサイエンティストにとって特に重要です。

このパターンは、プロンプトエンジニアリング、反復的な改良、専門知識の組み込みの手法を示しています。これらはすべて合成データ生成に不可欠です。このアプローチは主にクエリ変換に焦点を当てていますが、データ拡張とスケーラブルな合成データ生成の可能性を暗黙的に示しています。この基盤は、より包括的な合成データ生成タスクに拡張して、構造化されたアプリケーション固有の出力で非構造化自然言語入力をブリッジする LLMs の能力を強調できます。

このソリューションには、従来の意味での移行ツールやデプロイツールは含まれません。代わりに、LLM を使用して自然言語クエリをクエリ DSL に変換するための概念実証 (PoC) のデモンストレーションに焦点を当てています。 LLMs

- このパターンでは、Jupyter Notebook を環境を設定し、text-to-queryへの変換を実装するための step-by-stepガイドとして使用します。
- Amazon Bedrock を使用して LLMs にアクセスします。これは、自然言語を解釈し、適切なクエリを生成するために不可欠です。
- このソリューションは、Amazon OpenSearch Service と連携するように設計されています。Elasticsearch でも同様のプロセスに従うことができ、生成されたクエリを同様の検索エンジンに適用できる可能性があります。

Query DSL は、Elasticsearch と OpenSearch の両方で複雑なクエリを構築するために使用される柔軟な JSON ベースの検索言語です。これにより、検索オペレーションのクエリパラメータでクエリを指定でき、さまざまなクエリタイプがサポートされます。DSL クエリには、リーフクエリと複合クエリが含まれます。リーフクエリは、特定のフィールドの特定の値を検索し、全文、用語レベル、地理的、結合、スパン、および特殊なクエリを含みます。複合クエリは、複数のリーフ句または複

合句のラッパーとして機能し、結果を結合するか、動作を変更します。クエリ DSL は、シンプルでマッチオールなクエリから、非常に具体的な結果を生成する複雑な複数句のクエリまで、高度な検索の作成をサポートします。クエリ DSL は、高度な検索機能、柔軟なクエリ構築、JSON ベースのクエリ構造を必要とするプロジェクトに特に役立ちます。

このパターンでは、text-to-queryへの DSL 変換に、数ショットのプロンプト、システムプロンプト、構造化出力、プロンプトの連鎖、コンテキストのプロビジョニング、タスク固有のプロンプトなどの手法を使用します。これらの手法の定義と例については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

Jupyter ノートブックを効果的に使用して自然言語クエリをクエリ DSL クエリに変換するには、以下が必要です。

- Jupyter ノートブックに精通していること。Jupyter ノートブック環境でコードを移動および実行する方法に関する基本的な理解。
- Python 環境。必要なライブラリがインストールされた、動作中の Python 環境、できれば Python 3.x。
- Elasticsearch または OpenSearch の知識。アーキテクチャやクエリの実行方法など、Elasticsearch または OpenSearch の基本知識。
- AWS アカウント。 Amazon Bedrock およびその他の関連サービスにアクセス AWS アカウント するためのアクティブな 。
- ライブラリと依存関係。 AWS インタラクションboto3など、ノートブックに記載されている特定のライブラリのインストール、および LLM 統合に必要なその他の依存関係。
- Amazon Bedrock 内のモデルアクセス。このパターンでは、Anthropic LLMs を使用します。Amazon Bedrock コンソールを開き、モデルアクセスを選択します。次の画面で、特定のモデルを有効化を選択し、次の 3 つのモデルを選択します。
  - Claude 3 サブネット
  - Claude 3.5 サブネット
  - クロード 3 ハイク
- 適切な IAM ポリシーと IAM ロール。でノートブックを実行するには AWS アカウント、 AWS Identity and Access Management (IAM) ロールに `SagemakerFullAccess`ポリシーと、「追加情報」セクションで提供されているポリシーが必要です。このポリシーには、 という名前を付け

ることができますAPGtext2querydslpolicy。このポリシーには、リストされている 3 つの Claude モデルへのサブスクライブが含まれます。

これらの前提条件を設定すると、ノートブックを操作してtext-to-query機能を実装するときに、スムーズなエクスペリエンスが得られます。

制約事項

- 概念実証のステータス。このプロジェクトは主に概念実証 (PoC) を目的としています。これは、LLMs を使用して自然言語クエリをクエリ DSL に変換する可能性を示していますが、完全に最適化されていないか、本番稼働に対応していない可能性があります。
- モデルの制限：

  コンテキストウィンドウの制約。Amazon Bedrock で使用できる LLMs を使用する場合は、コンテキストウィンドウの制限に注意してください。

  Claude モデル (2024 年 9 月現在):

  - Claude 3 Opus: 200,000 トークン
  - Claude 3 Sonnet: 200,000 トークン
  - Claude 3 Haiku: 200,000 トークン

  Amazon Bedrock の他のモデルでは、コンテキストウィンドウのサイズが異なる場合があります。常に最新のドキュメントで最新情報を確認してください。

  モデルの可用性。 Amazon Bedrock での特定のモデルの可用性は異なる場合があります。このソリューションを実装する前に、必要なモデルにアクセスできることを確認してください。
- その他の制限事項
  - クエリの複雑さ。DSL 変換をクエリするための自然言語の有効性は、入力クエリの複雑さによって異なります。
  - バージョンの互換性。生成されたクエリ DSL では、使用する Elasticsearch または OpenSearch の特定のバージョンに基づいて調整が必要になる場合があります。
  - パフォーマンス。このパターンは PoC 実装を提供するため、クエリ生成の速度と精度は大規模な本番環境での使用には最適ではない可能性があります。
  - 料金。Amazon Bedrock で LLMs を使用すると、コストが発生します。選択したモデルの料金構造に注意してください。詳細については、「Amazon Bedrock の料金」を参照してください。

- メンテナンス。LLM テクノロジーの進歩やクエリ DSL 構文の変更に対応するために、プロンプトとモデルの選択を定期的に更新する必要がある場合があります。

製品バージョン

このソリューションは Amazon OpenSearch Service でテストされています。Elasticsearch を使用する場合は、このパターンの正確な機能をレプリケートするためにいくつかの変更が必要になる場合があります。

- OpenSearch バージョンの互換性。 OpenSearch は、メジャーバージョン内で下位互換性を維持します。以下に例を示します。
  - OpenSearch 1.x クライアントは、一般的に OpenSearch 1.x クラスターと互換性があります。
  - OpenSearch 2.x クライアントは、一般的に OpenSearch 2.x クラスターと互換性があります。

  ただし、可能な限り、クライアントとクラスターの両方に同じマイナーバージョンを使用することをお勧めします。
- OpenSearch API の互換性。 OpenSearch は、ほとんどのオペレーションで Elasticsearch OSS 7.10.2 との API 互換性を維持します。ただし、特に新しいバージョンでは、いくつかの違いがあります。
- OpenSearch のアップグレードに関する考慮事項：
  - 直接ダウングレードはサポートされていません。必要に応じて、ロールバックにスナップショットを使用します。
  - アップグレードするときは、互換性マトリックスとリリースノートに重大な変更がないか確認してください。

Elasticsearch に関する考慮事項

- Elasticsearch バージョン。クエリ構文と機能はメジャーバージョン間で変更される可能性があるため、使用している Elasticsearch のメジャーバージョンが重要です。現在、最新の安定バージョンは Elasticsearch 8.x です。クエリが特定の Elasticsearch バージョンと互換性があることを確認してください。
- Elasticsearch クエリ DSL ライブラリのバージョン。Elasticsearch クエリ DSL Python ライブラリを使用している場合は、そのバージョンが Elasticsearch のバージョンと一致していることを確認してください。以下に例を示します。

  - Elasticsearch 8.x では、8.0.0 以上 9.0.0 未満の`elasticsearch-dsl`バージョンを使用します。
  - Elasticsearch 7.x では、7.0.0 以上 8.0.0 未満の`elasticsearch-dsl`バージョンを使用します。
- クライアントライブラリのバージョン。公式の Elasticsearch クライアントを使用しているか、言語固有のクライアントを使用しているかにかかわらず、Elasticsearch バージョンと互換性があることを確認してください。
- DSL バージョンをクエリします。クエリ DSL は Elasticsearch バージョンで進化します。一部のクエリタイプまたはパラメータは、非推奨になったり、異なるバージョンで導入されたりする可能性があります。
- マッピングバージョン。インデックスのマッピングを定義し、バージョン間で変更する方法。特定の Elasticsearch バージョンのマッピングドキュメントを必ず確認してください。
- 分析ツールのバージョン。アナライザー、トークナイザー、またはその他のテキスト分析ツールを使用している場合、バージョンによって動作や可用性が変わる可能性があります。

アーキテクチャ

ターゲット アーキテクチャ

このパターンのアーキテクチャを以下に図で示します。

各パラメータの意味は次のとおりです。

1. ユーザー入力とシステムプロンプトと、数ショットプロンプトの例。このプロセスは、自然言語クエリまたはスキーマ生成のリクエストを提供するユーザーから始まります。
2. Amazon Bedrock 入力は Amazon Bedrock に送信されます。Amazon Bedrock は Claude LLM にアクセスするためのインターフェイスとして機能します。
3. Claude 3 Sonnet LLM。Amazon Bedrock は、LLMs の Claude 3 ファミリーの Claude 3 Sonnet を使用して入力を処理します。適切な Elasticsearch または OpenSearch クエリ DSL を解釈して生成します。スキーマリクエストの場合、合成 Elasticsearch または OpenSearch マッピングを生成します。
4. DSL 生成をクエリします。自然言語クエリの場合、アプリケーションは LLM の出力を受け取り、有効な Elasticsearch または OpenSearch Service クエリ DSL にフォーマットします。

5. 合成データ生成。また、アプリケーションはスキーマを使用して、テスト用に OpenSearch Serverless コレクションにロードする合成 Elasticsearch または OpenSearch データを作成します。
6. OpenSearch または Elasticsearch。生成されたクエリ DSL は、すべてのインデックスの OpenSearch Serverless コレクションに対してクエリされます。JSON 出力には、OpenSearch Serverless コレクションに存在するデータからの関連データとヒット数が含まれます。

自動化とスケール

このパターンで提供されるコードは、PoC 専用に構築されています。次のリストでは、ソリューションの自動化とスケーリングをさらに進め、コードを本番環境に移行するためのいくつかの提案を示します。これらの機能強化は、このパターンの範囲外です。

- コンテナ化：
  - アプリケーションを Dockerize して、異なる環境間で整合性を確保します。
  - スケーラブルなデプロイには、Amazon Elastic Container Service (Amazon ECS) や Kubernetes などのコンテナオーケストレーションプラットフォームを使用します。
- サーバーレスアーキテクチャ：
  - コア機能を AWS Lambda 関数に変換します。
  - Amazon API Gateway を使用して、自然言語クエリ入力用の RESTful エンドポイントを作成します。
- 非同期処理：
  - Amazon Simple Queue Service (Amazon SQS) を実装して、受信クエリをキューに入れます。
  - を使用して AWS Step Functions 、クエリを処理し、クエリ DSL を生成するワークフローを調整します。
- キャッシュ：
  - プロンプトをキャッシュするメカニズムを実装します。
- モニタリングとログ記録：
  - Amazon CloudWatch をモニタリングとアラートに使用します。
  - ログ分析のために Amazon CloudWatch Logs または Amazon OpenSearch Service を使用して一元的なログ記録を実装します。
- セキュリティの強化：
  - きめ細かなアクセスコントロールのための IAM ロールを実装します。
  - を使用して AWS Secrets Manager 、API キーと認証情報を安全に保存および管理します。

- マルチリージョンデプロイ：
  - レイテンシーとディザスタリカバリを改善する AWS リージョン ために、ソリューションを複数の にデプロイすることを検討してください。
  - インテリジェントなリクエストルーティングには Amazon Route 53 を使用します。

これらの提案を実装することで、この PoC を堅牢でスケーラブル、本番環境に対応したソリューションに変換できます。完全なデプロイの前に、各コンポーネントとシステム全体を徹底的にテストすることをお勧めします。

ツール

ツール

- Amazon SageMaker AI ノートブックは、機械学習開発用のフルマネージド Jupyter ノートブックです。このパターンでは、Amazon SageMaker AI でのデータ探索、モデル開発、実験のためのインタラクティブな環境としてノートブックを使用します。ノートブックは、他の SageMaker AI 機能や とシームレスに統合できます AWS のサービス。
- 「Python」は汎用のコンピュータープログラミング言語です。このパターンでは、Python をコア言語として使用してソリューションを実装します。
- Amazon Bedrock は、主要な AI スタートアップと Amazon からの高性能な基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。Amazon Bedrock は、自然言語処理のための LLMs へのアクセスを提供します。このパターンでは、Anthropic Claude 3 モデルを使用します。
- AWS SDK for Python (Boto3) は、Amazon Bedrock AWS のサービスを含む Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです。
- 「Amazon OpenSearch Service」は、AWS クラウドにおける OpenSearch クラスターのデプロイ、オペレーション、スケーリングを支援するマネージドサービスです。このパターンでは、クエリ DSL を生成するためのターゲットシステムとして OpenSearch Service を使用します。

コードリポジトリ

このパターンのコードは、GitHub Prompt Engineering Text-to-QueryDSL Using Claude 3 Models リポジトリにあります。この例では、ヘルスアプリケーションに関連付けられたユーザーとユーザープロファイルの投稿を作成するヘルスソーシャルメディアアプリを使用しています。

ベストプラクティス

このソリューションを使用する場合は、次の点を考慮してください。

- Amazon Bedrock にアクセスするための適切な AWS 認証情報とアクセス許可の必要性
- AWS のサービス および LLMs の使用に関連する潜在的なコスト
- クエリ DSL を理解して、生成されたクエリを検証して変更することの重要性

エピック

環境をセットアップし、データを準備する

タスク	説明	必要なスキル
開発環境を設定します。	**Note** この詳細な手順とコード、およびこのパターンの他のステップについては、GitHub リポジトリの包括的なチュートリアルを参照してください。 1. pip `requests-aws4auth` を使用して、、`boto3`、`awscli`、、などopensearch-py 、必要な numpyPython パッケージをインストールします。 2. `boto3`、、`json`、 などの必要なモジュールを osopensearch から`opensearchpy` 、	Python、pip、AWS SDK

タスク	説明	必要なスキル
	`RequestsHttpConnection` から`Opensearchpy` 、、`bulkopensearchpy.helpers` 、`re`、、`sagemaker` `time` `randomAWS4Auth`からインポートします`requests_aws4auth` 。	
AWS アクセスを設定します。	Amazon Bedrock クライアントと SageMaker AI セッションを設定します。SageMaker AI 実行ロールの Amazon リソースネーム (ARN) を取得し、後で OpenSearch Serverless コレクションを作成するときに使用します。	IAM、AWS CLI、Amazon Bedrock、Amazon SageMaker
ヘルスアプリスキーマをロードします。	事前定義されたファイルからヘルスポストとユーザープロファイルの JSON スキーマを読み取って解析します。後でプロンプトで使用できるようにスキーマを文字列に変換します。	DevOps エンジニア、AWS 全般、Python、JSON

合成データの生成

タスク	説明	必要なスキル
LLM ベースのデータジェネレーターを作成します。	generate_data() 関数を実装して、Claude 3 モデルで Amazon Bedrock Converse API を呼び出します。Sonnet、Sonnet 3.5、および	Python、Amazon Bedrock API、LLM プロンプト

タスク	説明	必要なスキル

	Haiku のモデル IDs を設定します。
	
	`model_id_sonnet3_5 = "anthropic.claude-3-5-sonnet-20240620-v1:0"`
	`model_id_sonnet = "anthropic.claude-3-sonnet-20240229-v1:0"`
	`model_id_haiku = "anthropic.claude-3-haiku-20240307-v1:0"`	

合成ヘルスポストを作成します。	generate_data() 関数を特定のメッセージプロンプトとともに使用して、指定されたスキーマに基づいて合成ヘルスポストエントリを作成します。関数呼び出しは次のようになります。 `health_post_data = generate_data(bedrock_rt, model_id_sonnet, system_prompt, message_healthpost, inference_config)`	Python、JSON

タスク	説明	必要なスキル
合成ユーザープロファイルを作成します。	generate_data() 関数を特定のメッセージプロンプトとともに使用して、指定されたスキーマに基づいて合成ユーザープロファイルエントリを作成します。これはヘルスポストの生成に似ていますが、別のプロンプトを使用します。	Python、JSON

OpenSearch をセットアップしてデータを取り込む

タスク	説明	必要なスキル
OpenSearch Serverless コレクションを設定します。	Boto3 を使用して、適切な暗号化、ネットワーク、アクセスポリシーを持つ OpenSearch Serverless コレクションを作成します。コレクションの作成は次のようになります。 `collection = aoss_client.create_collection(name=es_name, type='SEARCH')` OpenSearch Serverless の詳細については、AWS ドキュメントを参照してください。	OpenSearch Serverless、IAM
OpenSearch インデックスを定義します。	事前定義されたスキーママッピングに基づいて、OpenSearch クライアントを使用してヘルスポストとユーザープロファイルのインダイエック	OpenSearch、JSON

タスク	説明	必要なスキル
	スを作成します。インデックスの作成は次のようになります。 `response_health = oss_client.indices.create(healthpost_index, body=healthpost_body)`	
OpenSearch にデータをロードします。	inest_data() 関数を実行して、合成ヘルスポストとユーザープロファイルをそれぞれの OpenSearch インデックスに一括挿入します。関数は、 のバルクヘルパーを使用しますopensearch-py 。 `success, failed = bulk(oss_client, actions)`	Python、OpenSearch API、バルクデータオペレーション

クエリを生成して実行する

タスク	説明	必要なスキル
数ショットプロンプトの例を設計します。	Claude 3 モデルを使用してクエリ生成の少数の例を提供し、クエリの例と対応する自然言語の質問を生成します。システムプロンプトには、次の例が含まれます。 `system_prompt_query_generation =`	LLM プロンプト、クエリ DSL

タスク	説明	必要なスキル
	`[{"text": f"""You are an expert query dsl generator. ... Examples: {example_prompt} ..."""}]`	
text-to-queryDSL コンバーターを作成します。	クエリ生成用のスキーマ、データ、数ショットの例を含むシステムプロンプトを実装します。システムプロンプトを使用して、自然言語クエリをクエリ DSL に変換します。関数呼び出しは次のようになります。 `query_response = generate_data(bedrock_rt, model_id, system_prompt_query_generation, query, inference_config)`	Python、Amazon Bedrock API、LLM プロンプト
OpenSearch でクエリ DSL をテストします。	query_oss() 関数を実行して、生成されたクエリ DSL を OpenSearch Serverless コレクションに対して実行し、結果を返します。関数は OpenSearch クライアントの検索方法を使用します。 `response = oss_client.search(index="_all", body=temp)`	Python、OpenSearch API、クエリ DSL

テストと評価

タスク	説明	必要なスキル
テストクエリセットを作成します。	Claude 3 を使用して、合成データとスキーマに基づいてさまざまなテスト質問のセットを生成します。 `test_queries = generate_data(bedrock_rt, model_id_sonnet, query_system_prompt, query_prompt, inference_config)`	LLM プロンプト
クエリ DSL 変換の精度を評価します。	OpenSearch に対してクエリを実行し、返された結果の関連性と正確性を分析して、生成されたクエリ DSL をテストします。これには、クエリの実行とヒットの検査が含まれます。 `output = query_oss(response1) print("Response after running query against Opensearch") print(output)`	Python、データ分析、クエリ DSL
Claude 3 モデルをベンチマークします。	精度とレイテンシーの観点から、クエリ生成のためのさまざまな Claude 3 モデル (Haiku、Sonnet、Sonnet 3.5) のパフォーマンスを比較します。比較するには、 generate_data() を呼び出して実行時間を測定す	Python、パフォーマンスベンチマーク

タスク	説明	必要なスキル
	るmodel_idときに を変更します。	

クリーンアップとドキュメント化

タスク	説明	必要なスキル
クリーンアッププロセスを開発します。	使用後に OpenSearch Serverless コレクションからすべてのインデックスを削除します。	Python、AWS SDK、OpenSearch API

関連リソース

- クエリ DSL (OpenSearch ドキュメント)
- 「Amazon OpenSearch Service」
- OpenSearch Serverless コレクション
- Amazon Bedrock ドキュメント
- Amazon SageMaker AI ドキュメント
- AWS SDK for Python (Boto3) ドキュメント

追加情報

IAM ポリシー

このパターンで使用される IAM ロールのAPGtext2querydslpolicyポリシーは次のとおりです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream"
```

```
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::sagemaker-*",
        "arn:aws:s3:::sagemaker-*/*"
      ]
    },
    { "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
    },
    { "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": [
        "aoss:*"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "sagemaker:*"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*", "*"
      ],
```

```
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
          }
      }
    },
    { "Effect": "Allow",
      "Action": [
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "codecommit:ListRepositories"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": [
        "aws-marketplace:Subscribe"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws-marketplace:ProductId": [
            "prod-6dw3qvchef7zy",
            "prod-m5ilt4siql27k",
            "prod-ozonys2hmmpeu"
          ]
        }
      }
    },
    { "Effect": "Allow",
      "Action": [
        "aws-marketplace:Unsubscribe",
        "aws-marketplace:ViewSubscriptions"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
```

```
}
```

Anthropic Claude 3 モデルを使用したプロンプト手法

このパターンは、Claude 3 モデルを使用したtext-to-queryへの DSL 変換に関する以下のプロンプト手法を示しています。

- 数ショットプロンプト： 数ショットプロンプトは、Amazon Bedrock での Claude 3 モデルのパフォーマンスを向上させるための強力な手法です。このアプローチでは、モデルに同様のタスクの実行を求める前に、必要な入出力動作を示す少数の例を提供します。Amazon Bedrock で Claude 3 モデルを使用すると、特定のフォーマット、推論パターン、またはドメインの知識を必要とするタスクに対して、数ショットプロンプトが特に効果的になります。この手法を実装するには、通常、プロンプトをサンプルセクションと実際のクエリの 2 つの主要コンポーネントで構造化します。サンプルセクションには、タスクを説明する 1 つ以上の入力/出力ペアが含まれており、クエリセクションにはレスポンスが必要な新しい入力が表示されます。この方法は、Claude 3 がコンテキストと予想される出力形式を理解するのに役立ちます。多くの場合、より正確で一貫したレスポンスが得られます。

  例:

```
"query": {
  "bool": {
    "must": [
      {"match": {"post_type": "recipe"}},
      {"range": {"likes_count": {"gte": 100}}},
      {"exists": {"field": "media_urls"}}
    ]
  }
}
Question: Find all recipe posts that have at least 100 likes and include media URLs.
```

- システムプロンプト: Amazon Bedrock の Claude 3 モデルは、数ショットプロンプトに加えて、システムプロンプトの使用もサポートしています。システムプロンプトは、特定のユーザー入力を提示する前に、全体的なコンテキスト、指示、またはガイドラインをモデルに提供する方法です。トーンの設定、モデルの役割の定義、会話全体の制約の確立に特に役立ちます。Amazon Bedrock で Claude 3 でシステムプロンプトを使用するには、それを API リクエストの `system`パラメータに含めます。これはユーザーメッセージとは別のもので、インタラクション全体に適用されます。詳細なシステムプロンプトは、コンテキストを設定し、モデルのガイドラインを提供するために使用されます。

例:

```
You are an expert query dsl generator. Your task is to take an input question and
 generate a query dsl to answer the question. Use the schemas and data below to
 generate the query.

Schemas: [schema details]
Data: [sample data]
Guidelines:
- Ensure the generated query adheres to DSL query syntax
- Do not create new mappings or other items that aren't included in the provided
 schemas.
```

- 構造化出力: JSON や XML タグ内など、特定の形式で出力を提供するようにモデルに指示できます。

  例:

  ```
  Put the query in json tags
  ```

- プロンプトの連鎖: ノートブックは、生成された合成データを使用してサンプル質問を作成するなど、ある LLM 呼び出しの出力を別の の入力として使用します。
- コンテキストプロビジョニング: スキーマやサンプルデータなどの関連コンテキストがプロンプトに表示されます。

  例:

  ```
  Schemas: [schema details]
  Data: [sample data]
  ```

- タスク固有のプロンプト: 合成データの生成、質問例の作成、自然言語クエリの DSL クエリへの変換など、特定のタスクに対して異なるプロンプトが作成されます。

  テストの質問を生成する例 :

  ```
  Your task is to generate 5 example questions users can ask the health app based on
   provided schemas and data. Only include the questions generated in the response.
  ```

Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める

作成者: Ram Kandaswamy (AWS)

概要

このパターンでは、tic-tac-toe ゲームを使用して、さまざまな開発タスクに Amazon Q Developer を適用する方法を示します。tic-tac-toe ゲームのコードをシングルページアプリケーション (SPA) として生成し、UI を強化して、アプリケーションをデプロイするスクリプトを作成します AWS。

Amazon Q Developer は、ソフトウェア開発ワークフローを加速し、開発者と非開発者の両方の生産性を向上させるコーディングアシスタントとして機能します。技術的な専門知識に関係なく、ビジネス問題のアーキテクチャと設計ソリューションの作成、作業環境のブートストラップ、新機能の実装、検証用のテストケースの生成に役立ちます。自然言語の指示と AI 機能を使用して、一貫性のある高品質のコードを確保し、プログラミングスキルに関係なくコーディングの課題を軽減します。

Amazon Q Developer の主な利点は、反復的なコーディングタスクから解放できることです。`@workspace` 注釈を使用すると、Amazon Q Developer は統合開発環境 (IDE) 内のすべてのコードファイル、設定、プロジェクト構造を取り込み、インデックスを作成し、クリエイティブな問題解決に集中できるようにカスタマイズされたレスポンスを提供します。革新的なソリューションを設計し、ユーザーエクスペリエンスを向上させるためにより多くの時間を費やすことができます。技術系でない場合は、Amazon Q Developer を使用してワークフローを合理化し、開発チームとより効果的にコラボレーションできます。Amazon Q Developer Explain コード機能には詳細な手順と概要が用意されているため、複雑なコードベースをナビゲートできます。

さらに、Amazon Q Developer は、言語に依存しないアプローチを提供し、下位および中間レベルの開発者がスキルセットを拡張するのに役立ちます。言語固有の構文の代わりに、主要な概念とビジネスロジックに集中できます。これにより、テクノロジーを切り替える際の学習曲線が短縮されます。

前提条件と制限

前提条件

- Amazon Q Developer プラグインがインストールされた IDE (WebStorm や Visual Studio Code など)。手順については、Amazon Q Developer ドキュメントの「IDE での Amazon Q Developer 拡張機能またはプラグインのインストール」を参照してください。
- Amazon Q Developer でのアクティブな AWS アカウント セットアップ。手順については、Amazon Q Developer ドキュメントの「開始方法」を参照してください。

- npm がインストールされました。手順については、npm ドキュメントを参照してください。このパターンは npm バージョン 10.8 でテストされました。
- AWS Command Line Interface （ AWS CLI) がインストールされています。手順については、AWS CLI ドキュメントを参照してください。

制約事項

- Amazon Q Developer は、一度に 1 つの開発タスクのみを実行できます。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

ツール

- このパターンには、Visual Studio Code や WebStorm などの IDE が必要です。サポートされている IDEs のリストについては、Amazon Q Developer のドキュメントを参照してください。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

ベストプラクティス

AWS 「 規範ガイダンス」の「Amazon Q Developer のベストプラクティス」を参照してください。加えて:

- Amazon Q Developer にプロンプトを提供するときは、指示が明確であいまいでないことを確認してください。プロンプトにより多くのコンテキストを提供するために、プロンプト`@workspace`になどのコードスニペットと注釈を追加します。
- 関連するライブラリを含めてインポートし、システムによる競合や誤った推測を回避します。
- 生成されたコードが正確でない、または期待どおりにない場合は、フィードバックと再生成の提供オプションを使用します。プロンプトを小さな手順に分割してみてください。

エピック

作業環境のセットアップ

タスク	説明	必要なスキル
新しいプロジェクトを作成します。	作業環境で新しいプロジェクトを作成するには、次のコマンドを実行し、すべての質問のデフォルト設定を受け入れます。 `npx create-next-app@latest`	アプリ開発者、プログラマー、ソフトウェア開発者
ベースアプリケーションをテストします。	次のコマンドを実行し、ベースアプリケーションがブラウザに正常にロードされていることを確認します。 `npm run dev`	アプリ開発者、プログラマー、ソフトウェア開発者
ベースコードをクリーンアップします。	`src/app` フォルダ内の `page.tsx` ファイルに移動し、デフォルトのコンテンツを削除して空白のページを取得します。削除後、ファイルは次のようになります。 `export default function Home() { return (<div></div>); }`	アプリ開発者、プログラマー、ソフトウェア開発者

Amazon Q Developer を使用して tic-tac-toe ゲームプロジェクトを設計する

タスク	説明	必要なスキル
ステップの概要について説明します。	1. IDE でプロジェクトを開き、Amazon Q アイコンを選択してチャットパネルを開きます。 2. Amazon Q Developer チャットパネルで、単一ページアプリケーション (SPA) の作成の概要を尋ねます。以下に例を示します。 `I would like to create a single-page application involving Next.js React framework for tic-tac-toe game. What are the steps?`	アプリ開発者、プログラマー、ソフトウェア開発者
tic-tac-toe のコードを生成します。	チャットパネルで、 /dev コマンドとタスクの説明を使用して開発タスクを開始します。以下に例を示します。 `/dev Create a React-based single-page application written in TypeScript for a tic-tac-toe game with the following specifications: 1. Design an aesthetically pleasing interface with the game grid`	アプリ開発者、プログラマー、ソフトウェア開発者

タスク	説明	必要なスキル
	centered vertically and horizontally on the page. 2. Include a heading and clear instructions on how to play the game. 3. Implement color-coding for X and O marks to distinguish them easily. Amazon Q Developer は、指示に基づいてコードを生成します。	
生成されたコードを検査して受け入れます。	コードを視覚的に検査し、コードを受け入れるを選択してpage.tsxファイルを自動的に置き換えます。 問題が発生した場合は、フィードバックの提供と再生成を選択し、発生した問題を記述します。	アプリ開発者、プログラマー、ソフトウェア開発者

タスク	説明	必要なスキル
lint エラーを修正しました。	tic-tac-toe ゲームの例にはグリッドが含まれています。Amazon Q Developer が生成するコードは、デフォルトのタイプ を使用する場合がありますany。次のように Amazon Q Developer にプロンプトを表示することで、型の安全性を追加できます。 `/dev Ensure proper TypeScript typing for the onSquare Click event handler to resolve any 'any' type issues.`	アプリ開発者、プログラマー、ソフトウェア開発者

タスク	説明	必要なスキル

視覚的な魅力を追加します。	元の要件を小さなフラグメントに分割できます。たとえば、開発タスクで次のプロンプトを使用してゲーム UI を改善できます。このプロンプトは、カスケードスタイルシート (CSS) スタイルを強化し、デプロイ用にアプリケーションをエクスポートします。
	
	`/dev Debug and fix any CSS issues to correctly display the game grid and overall layout.`
	
	`Simplify the code by removing game history functionality and related components.`
	
	`Implement static file export to an 'out' directory for easy deployment. The solution should be fully functional, visually appealing, and free of typing errors or layout issues.`
		アプリ開発者、プログラマー、ソフトウェア開発者

タスク	説明	必要なスキル

もう一度テストします。	1. 開発ライフサイクルを完了したので、コードをテストして、期待どおりに動作することを確認します。アプリケーションをローカルで実行するには、 コマンドを使用します。
	`npm run dev`
	2. アプリケーションが正常に動作する場合は、 `build` コマンドを使用して、デプロイの準備としてアプリケーション全体を出力フォルダにエクスポートします。
	`npm run build`
		アプリ開発者、プログラマー、ソフトウェア開発者

アプリケーションを にデプロイする AWS クラウド

タスク	説明	必要なスキル

デプロイ用のフォルダとファイルを作成します。	作業環境のプロジェクトで、デプロイフォルダと、その中に `pushtos3.sh` と の 2 つのファイルを作成しますcloudformation.yml 。
	`mkdir deployment && cd deployment`
	`touch pushtos3.sh && chmod +x pushtos3.sh`
		アプリ開発者、プログラマー、ソフトウェア開発者

タスク	説明	必要なスキル
	`touch cloudform ation.yml`	
自動化コードを生成します。	1. Amazon Q Developer のチャットパネルで、次のプロンプトを指定します。 `/dev Generate a Cloudformation template that creates two resources: tictactoe artifact bucket and CloudFron t. CloudFront should be configure d with this bucket as origin. Add cache policy appropriate for Amazon S3 and default root object as index.html. Ensure that origin access control is used and no public bucket is created. Output all the resources and their ARNs.` 2. 生成されたコードを確認して受け入れます。これで、以前に作成した `cloudformation.yml` ファイルに、 のリソースを作成する AWS CloudForm ation スクリプトが入力されます AWS クラウド。	AWS 管理者、AWS DevOps、アプリ開発者

タスク	説明	必要なスキル
スクリプトコンテンツを生成します。	デプロイスクリプトを作成するには、次のプロンプトを使用します。 `/dev Modify the pushtos3 shell script so that it can use AWS CLI commands to create a CloudFormation stack named tictactoe-stack if it does not exist already, and use cloudformation.yml as the source template. Wait for the stack to complete and sync the contents from the out folder to the S3 bucket. Perform invalidation of the CloudFront origin.`	アプリ開発者、プログラマー、ソフトウェア開発者
アプリケーションを にデプロイします AWS クラウド。	1. 有効な AWS 認証情報を使用して作業環境を設定します。 2. シェルスクリプトを実行して、完全に機能する tic-tac-toe ゲームを にデプロイします AWS クラウド。	AWS 管理者、AWS DevOps、クラウドアーキテクト、アプリ開発者

トラブルシューティング

問題	ソリューション

ビルドでは、単一ページのアプリケーションを作成したり、出力フォルダにエクスポートしたりすることはありません。	`next.config.mjs` ファイルの内容を確認します。
	
	コードに次のデフォルト設定がある場合：
	
	`const nextConfig = {};`
	
	次のように変更します。
	
	`const nextConfig = {`
	`  output: 'export',`
	`  distDir: 'out',`
	`};`

関連リソース

- 新しい React プロジェクトの作成 (React ドキュメント)
- Amazon Q Developer の概要 (AWS ドキュメント)
- Amazon Q Developer のベストプラクティス (AWS 規範ガイダンス)
- JetBrains IDEs での Amazon Q Developer のインストール、設定、使用 (YouTube 動画)
- コマンドライン用の Amazon Q のインストール (AWS ドキュメント)

SageMaker Processing を使用して、テラバイト規模の ML データセットの分散型特徴量エンジニアリングを実現

クリス・ブームハワー (AWS) によって作成された

概要

テラバイト規模またはそれ以上のデータセットの多くは階層的なフォルダ構造で構成されており、データセット内のファイルは相互に依存している場合があります。このため、機械学習 (ML) エンジニアとデータサイエンティストは、モデルトレーニングや推論のためにデータを準備するために、慎重に設計上の決定を下さなければなりません。このパターンは、手動のマクロシャーディングとマイクロシャーディングの手法を Amazon SageMaker Processing と仮想 CPU (vCPU) 並列化と組み合わせて使用して、複雑なビッグデータ ML データセットの特徴量エンジニアリングプロセスを効率的にスケーリングする方法を示しています。

このパターンでは、データディレクトリを複数のマシンに分割して処理することを「マクロシャーディング」と定義し、「マイクロシャーディング」は各マシンのデータを複数の処理スレッドに分割することと定義されています。このパターンでは、Amazon SageMaker と「PhysioNet MIMIC-III」データセットのサンプル時系列波形レコードを使用して、これらの手法を実証しています。このパターンの手法を実装することで、リソース利用率とスループット効率を最大化しながら、特徴量エンジニアリングの処理時間とコストを最小限に抑えることができます。これらの最適化では、データタイプに関係なく、類似の大規模データセットで、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスと vCPUs で、分散された SageMaker 処理を使用します。

前提条件と制限

前提条件

- 独自のデータセットにこのパターンを実装する場合は、SageMaker ノートブックインスタンスまたは SageMaker Studio にアクセスできます。Amazon SageMaker を初めて使用する場合は、AWS ドキュメントの「Amazon SageMaker の入門」を参照してください。
- SageMaker Studio、「PhysioNet MIMIC-III」のサンプルデータを使用してこのパターンを実装したい場合。
- このパターンは SageMaker 処理を使用しますが、SageMaker 処理ジョブを実行した経験は必要ありません。

制約事項

- このパターンは、相互に依存するファイルを含む ML データセットに非常に適しています。このような相互依存関係は、手動によるマクロシャーディングと、複数の単一インスタンスの SageMaker Processing ジョブをparallel 実行することから最もメリットがあります。そのような相互依存関係が存在しないデータセットの場合、SageMaker Processing のShardedByS3Key機能は、同じ処理ジョブによって管理される複数のインスタンスにシャーディングされたデータを送信するため、マクロシャーディングのより良い代替となる可能性があります。ただし、このパターンのマイクロシャーディング戦略をどちらのシナリオでも実装して、インスタンス vCPUs を最大限に活用できます。

製品バージョン

- Amazon SageMaker Python SDK バージョン 2

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Simple Storage Service (Amazon S3)
- Amazon SageMaker

ターゲットアーキテクチャ

マクロシャーディングと分散型 EC2 インスタンス

このアーキテクチャで表される 10 個のparallel プロセスは、MIMIC-III データセットの構造を反映しています。(図を簡略化するため、プロセスは楕円で示されています)。手動マクロシャーディングを使用する場合も、同様のアーキテクチャがどのデータセットにも適用されます。MIMIC-III の場合、各患者グループフォルダーを最小の労力で個別に処理することで、データセットの未加工の構造を活用できます。以下の図では、レコードグループブロックが左側に表示されています (1)。データが分散されていることを考えると、患者グループごとにシャードするのは理にかなっています。

ただし、患者グループごとに手動でシャーディングを行うと、図 (2) の中央のセクションでわかるように、複数の EC2 インスタンスによる単一の処理ジョブではなく、患者グループフォルダごとに個別の処理ジョブが必要になります。MIMIC-III のデータには、バイナリ波形ファイルと対応するテキストベースのヘッダーファイルの両方が含まれており、バイ

ナリデータ抽出には「wfdb ライブラリ」に依存する必要があるため、特定の患者のすべてのレコードを同じインスタンスで利用できるようにする必要があります。各バイナリ波形ファイルの関連ヘッダーファイルも確実に存在させる唯一の方法は、各シャードを独自の処理ジョブ内で実行する手動シャーディングを実装し、処理ジョブの入力を定義するときにs3_data_distribution_type='FullyReplicated'を指定することです。あるいは、すべてのデータが単一のディレクトリにあり、ファイル間に依存関係がない場合、より適切なオプションは、複数の EC2 インスタンスとs3_data_distribution_type='ShardedByS3Key'を指定して単一の処理ジョブを起動することかもしれません。Amazon S3 データ分散タイプとしてShardedByS3Key を指定すると、SageMaker はインスタンス間のデータシャーディングを自動的に管理するようになります。

複数のインスタンスを同時に実行すると時間を節約できるため、データを前処理するにはフォルダごとに処理ジョブを起動するのがコスト効率の高い方法です。コストと時間をさらに節約するために、各処理ジョブ内でマイクロシャーディングを使用することもできます。

マイクロシャーディングとparallel vCPUs

各処理ジョブ内で、グループ化されたデータはさらに分割され、SageMaker のフルマネージド EC2 インスタンスで使用可能なすべての vCPUs を最大限に活用します。図の中央のセクション (2) のブロックは、各主要処理ジョブ内で何が起こるかを表しています。患者記録フォルダーの内容は、インスタンスで使用可能なvCPUs の数に基づいてフラット化され、均等に分割されます。フォルダの内容が分割されると、同じサイズのファイルセットがすべての vCPUs に分散されて処理されます。処理が完了すると、各 vCPU の結果は、処理ジョブごとに 1 つのデータファイルにまとめられます。

添付のコードでは、これらの概念がsrc/feature-engineering-pass1/preprocessing.pyファイルの次のセクションに示されています。

```
def chunks(lst, n):
    """
    Yield successive n-sized chunks from lst.

    :param lst: list of elements to be divided
    :param n: number of elements per chunk
    :type lst: list
    :type n: int
    :return: generator comprising evenly sized chunks
    :rtype: class 'generator'
    """
    for i in range(0, len(lst), n):
        yield lst[i:i + n]
```

```
# Generate list of data files on machine
data_dir = input_dir
d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]
file_list = []
for ds in d_subs:
    file_list.extend(os.listdir(os.path.join(data_dir, ds, '.')))
dat_list = [os.path.join(re.split('_|\.', f)[0].replace('n', ''), f[:-4]) for f in
 file_list if f[-4:] == '.dat']

# Split list of files into sub-lists
cpu_count = multiprocessing.cpu_count()
splits = int(len(dat_list) / cpu_count)
if splits == 0: splits = 1
dat_chunks = list(chunks(dat_list, splits))

# Parallelize processing of sub-lists across CPUs
ws_df_list = Parallel(n_jobs=-1, verbose=0)(delayed(run_process)(dc) for dc in
 dat_chunks)

# Compile and pickle patient group dataframe
ws_df_group = pd.concat(ws_df_list)
ws_df_group = ws_df_group.reset_index().rename(columns={'index': 'signal'})
ws_df_group.to_json(os.path.join(output_dir, group_data_out))
```

最初に関数chunksを定義し、与えられたリストをn の長さの均一な大きさのブロックに分割し、その結果をジェネレータとして返すことにより、与えられたリストを使用します。次に、存在するすべてのバイナリ波形ファイルのリストをコンパイルして、データを患者フォルダー全体でフラット化します。これが完了すると、EC2 インスタンスで使用可能な vCPUs の数が取得されます。バイナリ波形ファイルのリストは、chunksを呼び出すことによってこれらのvCPUに均等に分割され、その後、「joblibのParallelクラス」を使用することによって、各波形サブリストがそれぞれのvCPUで処理されます。処理ジョブによって結果は自動的に 1 つのデータフレームリストにまとめられ、SageMaker はさらに処理してからジョブ完了時に Amazon S3 に書き込まれます。この例では、処理ジョブによって Amazon S3 に書き込まれるファイルが 10 個あります (ジョブごとに 1 つ)。

最初の処理ジョブがすべて完了すると、図 (3) の右側のブロックに示されているセカンダリ処理ジョブが、各プライマリ処理ジョブによって生成された出力ファイルを結合し、結合された出力を Amazon S3 (4) に書き込みます。

ツール

ツール

- Python — このパターンに使用されるサンプルコードは Python (バージョン 3) です。
- SageMaker Studio – Amazon SageMaker Studio は、ウェブベースの機械学習用の統合開発環境 (IDE) です。この IDE を使うと、機械学習モデルを構築、トレーニング、デバッグ、デプロイ、モニタリングできます。SageMaker Processing ジョブは、SageMaker Studio 内の Jupyter Notebookを使用して実行します。
- SageMaker Processing — Amazon SageMaker Processingを使用すると、データ処理ワークロードを簡単に実行できます。このパターンでは、特徴量エンジニアリングコードは SageMaker Processing ジョブを使用して大規模に実装されます。

コード

添付の.zip ファイルには、このパターンの完全なコードが記載されています。次のセクションでは、このパターンのアーキテクチャを構築する手順について説明します。各ステップは、添付ファイルのサンプルコードで説明されています。

エピック

SageMaker Studio 環境をセットアップする

タスク	説明	必要なスキル
Amazon SageMaker Studio へアクセスします。	「Amazon SageMaker ドキュメント」に記載されている指示に従って、AWS アカウントの SageMaker Studio にオンボーディングします。	データサイエンティスト、ML エンジニア
wget ユーティリティをインストールします。	新しい SageMaker Studio 構成をオンボーディングした場合や、SageMaker Studio でこれらのユーティリティをこれまで使用したことがない場合は、「wget」をインストールしてください。	データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル
	インストールするには、SageMaker Studio コンソールでターミナルウィンドウを開き、以下のコマンドを実行します。 `sudo yum install wget`	

サンプルコードをダウンロードして解凍します。	「添付ファイル」セクションからattachments.zip ファイルをダウンロードします。ターミナルウィンドウで、ファイルをダウンロードしたフォルダに移動し、その内容を抽出します。
	`unzip attachment.zip`
	Scaled-Processing.zip ファイルを抽出したフォルダに移動し、ファイルの内容を抽出します。
	`unzip Scaled-Processing.zip`
		データサイエンティスト、MLエンジニア

タスク	説明	必要なスキル
physionet.org からサンプルデータセットをダウンロードし、Amazon S3 にアップロードします。	`Scaled-Processing` ファイルが含まれているフォルダ内で `get_data.ipynb` Jupyter Notebookを実行します。このノートブックは「physionet.org」からサンプル MIMIC-III データセットをダウンロードし、Amazon S3 の SageMaker Studio セッションバケットにアップロードします。	データサイエンティスト、ML エンジニア

1 つ目の前処理スクリプトを設定します。

タスク	説明	必要なスキル

すべてのサブディレクトリにわたってファイル階層をフラット化する。	MIMIC-III のような大規模なデータセットでは、論理的な親グループ内であってもファイルが複数のサブディレクトリに分散されることがよくあります。次のコードが示すように、スクリプトはすべてのサブディレクトリにあるすべてのグループファイルをフラット化するように設定する必要があります。
	`# Generate list of .dat files on machine`
	`data_dir = input_dir`
	`d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]`
	`file_list = []`
		データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル

	``
	for ds in d_subs:
	    file_list.extend(o
	s.listdir(os.path.
	join(data_dir, ds,
	 '.')))
	dat_list = [os.path.
	join(re.split('_|\
	.', f)[0].replace('n',
	 ''), f[:-4]) for f in
	 file_list if f[-4:] ==
	 '.dat']
	``
	
	Note
	このエピックのコードス二ペットの例は、添付ファイルに提供されている `src/feature-engineering-pass1/preprocessing.py` ファイルからのものです。	

タスク	説明	必要なスキル

vCPU 数に基づいてファイルをサブグループに分割します。	ファイルは、スクリプトを実行するインスタンスに存在するvCPUs の数に応じて、同じサイズのサブグループまたはチャンクに分割する必要があります。このステップでは、次のようなコードを実装できます。
	`# Split list of files into sub-lists`
	`cpu_count = multiprocessing.cpu_count()`
	`splits = int(len(dat_list) / cpu_count)`
	`if splits == 0: splits = 1`
	`dat_chunks = list(chunks(dat_list, splits))`
		データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル

vCPUs 間のサブグループの処理を並列化します。	スクリプトロジックは、すべてのサブグループを並行して処理するように設定する必要があります。そのためには、Joblib ライブラリのParallel クラスとdelayed メソッドを次のように使用します。
	`# Parallelize processing of sub-lists across CPUs`
	`ws_df_list = Parallel(n_jobs=-1, verbose=0)(delayed(run_process)(dc) for dc in dat_chunks)`
		データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル

Amazon S3 に単一ファイルグループの出力を保存します。	並列 vCPU 処理が完了したら、各 vCPU の結果を組み合わせて、ファイルグループの S3 バケットパスにアップロードする必要があります。このステップでは、次のようなコードを実行できます。
	
	`# Compile and pickle patient group dataframe`
	`ws_df_group = pd.concat(ws_df_list)`
	`ws_df_group = ws_df_group.reset_index().rename(columns={'index': 'signal'})`
	`ws_df_group.to_json(os.path.join(output_dir, group_data_out))`
		データサイエンティスト、ML エンジニア

2 つ目の前処理スクリプトを設定します。

タスク	説明	必要なスキル
最初のスクリプトを実行したすべての処理ジョブで生成されたデータファイルを結合します。	前述のスクリプトは、データセットのファイルグループを処理する SageMaker Processing ジョブごとに 1 つのファイルを出力します。 次に、これらの出力ファイルを 1 つのオブジェクトに結合し、1 つの出力データセットを Amazon S3 に書き込む必要があります。これは、添付	データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル

	ファイルにあるsrc/feature-engineering-pass1p5/preprocessing.py ファイルに次のように示されています。
	
	`def write_parquet(wavs_df, path):`
	`    """`
	`    Write waveform summary dataframe to S3 in parquet format.`
	
	`    :param wavs_df: waveform summary dataframe`
	`    :param path: S3 directory prefix`
	`    :type wavs_df: pandas dataframe`
	`    :type path: str`
	`    :return: None`
	`    """`
	`    extra_args = {"ServerSideEncryption": "aws:kms"}`
	`    wr.s3.to_parquet(`
	`        df=wavs_df,`
	`        path=path,`
	`        compression='snappy',`
	`        s3_additional_kwargs=extra_args)`
	
	`def combine_data():`
	`    """`
	`    Get combined data and write to parquet.`	

タスク	説明	必要なスキル
	:return: waveform summary dataframe :rtype: pandas dataframe """ wavs_df = get_data() wavs_df = normalize_signal_names(wavs_df) write_parquet(wavs_df, "s3://{}/{}/{}".format(bucket_xform, dataset_prefix, pass1p5out_data)) return wavs_df wavs_df = combine_data()	

処理ジョブの実行

タスク	説明	必要なスキル
最初の処理ジョブを実行します。	マクロシャーディングを実行するには、ファイルグループごとに個別の処理ジョブを実行します。マイクロシャーディングは各処理ジョブ内で実行されます。これは、各ジョブで最初のスクリプトが実行されるためです。次のコードは、次のスニペット (notebooks/FeatExtr	データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル

	act_Pass1.ipynb に含まれている) 内の各ファイルグループディレクトリの処理ジョブを起動する方法を示しています。
	
	`pat_groups = list(range(30,40))`
	`ts = str(int(time.time()))`
	
	`for group in pat_groups:`
	`  sklearn_processor = SKLearnProcessor(framework_version='0.20.0',`
	`  role=role,`
	`  instance_type='ml.m5.4xlarge',`
	`  instance_count=1,`
	`  volume_size_in_gb=5)`
	`  sklearn_processor.run(`
	`  code='../src/feature-engineering-pass1/preprocessing.py',`
	`  job_name='-'.join(['scaled-processing-p1', str(group), ts]),`
	`  arguments=[`	

タスク	説明	必要なスキル
	"input_path", "/opt/ml/processing/input", "output_path", "/opt/ml/processing/output", "group_data_out", "ws_df_group.json"], inputs= [ProcessingInput(source=f's3://{sess.default_bucket()}/data_inputs/{group}', destination='/opt/ml/processing/input', s3_data_distribution_type='FullyReplicated')], outputs= [ProcessingOutput(source='/opt/ml/processing/output', destination=f's3://{sess.default_bucket()}/data_outputs/{group}')], wait=False	

タスク	説明	必要なスキル
	)	

タスク	説明	必要なスキル

2 つ目の処理ジョブを実行します。	最初の処理ジョブセットで生成された出力を結合し、前処理のために追加の計算を実行するには、1 つの SageMaker 処理ジョブを使用して 2 番目のスクリプトを実行します。次のコードはこれを示しています (notebooks/FeatExtract_Pass1p5.ipynb に含まれている)。
	
	`ts = str(int(time.time()))`
	`bucket = sess.default_bucket()`
	
	`sklearn_processor = SKLearnProcessor(framework_version='0.20.0',`
	
	`role=role,`
	
	`instance_type='ml.t3.2xlarge',`
	
	`instance_count=1,`
	
	`volume_size_in_gb=5)`
	`sklearn_processor.run(`
	`code='../src/feature-engineering-pass1p5/preprocessing.py',`
		データサイエンティスト、ML エンジニア

タスク	説明	必要なスキル
	job_name='-'.join(['scaled-processing', 'p1p5', ts]), arguments=['bucket', bucket, 'pass1out_prefix', 'data_outputs', 'pass1out_data', 'ws_df_group.json', 'pass1p5out_data', 'waveform_summary.parquet', 'statsdata_name', 'signal_stats.csv'], wait=True)	

関連リソース

- クイックスタートを使用して Amazon SageMaker Studio にオンボーディングする (SageMaker ドキュメント)
- プロセスデータ (SageMaker ドキュメンテーション)
- scikit-learn を使ってデータを処理する (SageMaker ドキュメンテーション)
- JobLib. パラレルドキュメンテーション
- ムーディ、B.、ムーディ、G.、ビジャロエル、M.、クリフォード、G.D.、シルバ、I. (2020)。MIMIC-III 波形データベース (バージョン 1.0)。フィジオネット。
- ジョンソン、A. E. W.、ポラード、T.J.、シェン、L.、リーマン、L.H.、フェン、M.、ガセミ、M.、ムーディ、B.、ゾロビッツ、P.、セリ、L.A.、マーク、R.G. (2016) 。「MIMIC-III」は、無料でアクセスできる救命救急データベースです。科学データ、3、160035。
- MIMIC-III 波形データベースライセンス

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

フラスコと AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化

作成者：Chris Caudill (AWS) と Durga Sury (AWS)

概要

人工知能と機械学習 (AI/ML) サービスからの出力を視覚化するには、開発者やエンジニアがカスタマイズする必要な複雑な API 呼び出しが必要になることがよくあります。アナリストが新しいデータセットを迅速に探索する場合、これは欠点になる可能性があります。

ウェブベースのユーザーインターフェイス (UI) を使用することで、サービスのアクセシビリティを高め、よりインタラクティブなデータ分析を実現できます。これにより、ユーザーが独自のデータをアップロードし、ダッシュボードでモデルの結果を視覚化できます。

このパターンでは、Flask と Plotly を使用して、 Amazon Comprehend をカスタムウェブアプリケーションと統合し、ユーザーが提供したデータからセンチメントとエンティティを視覚化します。このパターンでは、AWS Elastic Beanstalk を使用してアプリケーションをデプロイするステップも記載されています。Amazon Web Services (AWS) の AI サービスを使用するか、エンドポイント (Amazon SageMaker エンドポイントなど) でホストされるカスタムトレーニング済みモデルを使用してアプリケーションを適応させることができます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ローカルマシンにインストールされ、設定されたAWS コマンドラインインターフェイス (AWS CLI) 詳細については、JupyterHub ドキュメントのConfiguration Basics を参照してください。AWS Cloud9 統合開発環境 (IDE) を使用することもできます。詳細については、AWS Cloud9 ドキュメントのAWS Cloud9 の Python チュートリアル」とAWS Cloud9 IDE で実行中のアプリケーションのプレビュー を参照してください。

  注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら
- Flask のウェブアプリケーションフレームワークについての理解。 へのサインアップの詳細については、 ドキュメント内のクイックスタート を参照してください。

- インストールされ、設定されたPython バージョン 3.6 以降。Python をインストールするには、AWS Elastic Beanstalk ドキュメントのPython 開発環境のセットアップ の指示に従います。
- Elastic Beanstalk コマンドラインインターフェイス (EB CLI) これに関する詳細については、AWS Elastic Beanstalk ドキュメントからEB CLI をインストール とEB CLI を設定 を参照してください。

制約事項

- このパターンの Flask アプリケーションは、単一のテキスト列を使用し、200 行に制限されている.csv ファイルで動作するように設計されています。アプリケーションコードは他のファイルタイプやデータ量を処理するように調整できます。
- アプリケーションはデータの保持を考慮せず、アップロードされたユーザーファイルは手動で削除されるまで集計を続けます。アプリケーションを Amazon Simple Storage Service (Amazon S3) と統合して永続的なオブジェクトストレージを作成したり、Amazon DynamoDB などのデータベースを使用してサーバーレスのキーバリューストレージを作成したりできます。
- このアプリケーションは英語のドキュメントのみを考慮します。ただし、Amazon Comprehend を使用してドキュメントの主要言語を検出することはできます。各アクションに適用される言語の詳細については、Amazon Comprehend ドキュメントのAPI リファレンス を参照してください。
- 一般的なエラーとその解決策を含むトラブルシューティングリストは、追加情報セクションにあります。

アーキテクチャ

アプリケーションのアーキテクチャ

Flaskは、Pythonでウェブアプリケーションを開発するための軽量フレームワークです。Python の強力なデータ処理と豊富な ウェブ UI を組み合わせるように設計されています。パターンの Flask アプリケーションでは、ユーザーがデータをアップロードし、そのデータを Amazon Comprehend に送信して推論し、結果を視覚化するウェブアプリケーションの構築方法を示しています。 マッピングドキュメントの構造は次のとおりです。

- `static` — ウェブ UI が適用されるすべての静的ファイル (JavaScript、CSS、画像など) が含まれています。
- `templates` — アプリケーションのすべての HTML ページが含まれます。
- `userData` — アップロードされたユーザーデータを格納します。
- `application.py` — Flask アプリケーションファイル

- comprehend_helper.py — Amazon Comprehend に API 呼び出しを行う関数
- アプリケーション設定ファイル。
- requirements.txt — アプリケーションに必要な Python の依存関係

application.py このスクリプトには、4 つの Flask ルートで構成されるウェブアプリケーションのコア機能が含まれています。以下の図に、これらの Flask ルートを示します。

- / はアプリケーションのルートで、ユーザーを upload.html ページ (templates ディレクトリに格納) にガイドします。
- /saveFile は、ユーザーがファイルをアップロードした後に呼び出されるルートです。このルートは、ユーザーがアップロードしたファイルを含む HTML フォームを介して POST リクエストを受け取ります。userData ファイルがディレクトリに保存され、ルートはユーザーを / dashboard ルートにガイドします。
- /dashboard がユーザーを dashboard.html ページにがガイドします。このページの HTML、/ data ルートからデータを読み取る static/js/core.js に JavaScript コードを実行して、ページの視覚化を構築します。
- /data は、ダッシュボードに視覚化されるデータを表示する JSON API です。このルートでは、ユーザーが提供したデータを読み取り、comprehend_helper.py の関数を使用してユーザーデータを Amazon Comprehend にガイドして、センチメント分析と名前付けのエンティティ認識 (NER) を行います。Amazon Comprehend のレスポンスがフォーマットされ、JSON オブジェクトとして返されます。

デプロイアーキテクチャ

AWS クラウドで Elastic Beanstalk を使用してデプロイされたアプリケーションの設計上の考慮事項の詳細については、AWS Elastic Beanstalk ドキュメントの「」を参照してください。

設計上の考慮事項

テクノロジースタック

- Amazon Comprehend
- Elastic Beanstalk
- Flask

自動化とスケール

Elastic Beanstalk のデプロイは、ロードバランサーとauto スケーリンググループを使用して自動的にセットアップされます。その他の設定オプションについては、AWS Elastic Beanstalk ドキュメントのElastic Beanstalk 環境設定 を参照してください。

ツール

- AWS コマンドラインインターフェイス (AWS CLI) は、AWS のすべての部分とやり取りするための一貫したインターフェイスを提供する統合ツールです。
- Amazon Comprehend は、特別な前処理を必要とせずに、自然言語処理 (NLP) を使用してドキュメントの内容に関するインサイトを抽出します。
- AWS Elastic Beanstalk を使用すると、アプリケーションを実行するインフラストラクチャについて知ることなく、AWS クラウドでアプリケーションをすばやくデプロイおよび管理できます。
- Elastic Beanstalk CLI (EB CLI) は、AWS Elastic Beanstalk のコマンドラインインターフェイスです。インタラクティブなコマンドを提供し、ローカルリポジトリからの環境の作成、更新、モニタリングを簡素化します。
- Flask フレームワークは、Python を使用してデータ処理と API コールを実行し、Plotly によるインタラクティブなウェブビジュアライゼーションを提供します。

コード

このパターンのコードは、GitHub 内の「Flask と AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化」リポジトリで利用できます。

エピック

Flask アプリケーションをセットアップします。

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	次のコマンドを実行して、GitHub の Flask と AWS Elastic Beanstalk を使用して、AI/ML モデル結果を可視化 を取得します。	開発者

タスク	説明	必要なスキル
	`git clone git@github.com:aws-samples/aws-comprehend-elasticbeanstalk-for-flask.git` **Note** GitHub で SSH キーを設定してください。	

Python モジュールをインストールします。	リポジトリをクローンすると、新しいローカル `aws-comprehend-elasticbeanstalk-for-flask` ディレクトリが作成されます。そのディレクトリでは、`requirements.txt` ファイルには、アプリケーションを実行する Python モジュールとバージョンが含まれています。 と をインストールするには、次のコマンドを使用します。
	
	`cd aws-comprehend-elasticbeanstalk-for-flask`
	
	`pip install -r requirements.txt`
		Python 開発

タスク	説明	必要なスキル

アプリケーションのローカルテスト	次のコマンドを実行して、Weka サーバーを作成します。
	
	`python application.py`
	
	これにより、実行中のサーバーに関する情報が返されます。ブラウザを開いて http://localhost:5000 にアクセスすると、アプリケーションにアクセスできます。
	
	Note
	AWS Cloud9 IDE でアプリケーションを実行している場合は、`application.py` ファイルの `application.run()` コマンドを次の行に置き換える必要があります。
	
	`application.run(host=os.getenv('IP', '0.0.0.0'),port=int(os.getenv('PORT', 8080)))`
	
	デプロイする前にこの変更を元に戻す必要があります。
		Python 開発

アプリケーションを Elastic Beanstalk にデプロイします。

タスク	説明	必要なスキル
AWS Elastic Beanstalk アプリケーション	Elastic Beanstalk アプリケーションとしてプロジェクトを起動するには、アプリケーションのルートディレクトリから以下のコマンドを実行します。 `eb init -p python-3.6 comprehend_flask --region us-east-1` ⚠ Important - `comprehend_flask` は Elastic Beanstalk アプリケーションの名前で、必要に応じて変更できます。 - AWS リージョンを任意のリージョンに置き換えることができます。リージョンを指定していない場合は、AWS CLI のデフォルトリージョンが使用されます。 - このアプリケーションは Python バージョン 3.6 でビルドされました。他の Python バージョンを使用すると、エラーが発生する可能性があります。	アーキテクト、開発者

タスク	説明	必要なスキル
	`eb init -i` コマンドを実行する場合、より多くのデプロイ設定オプションが表示されます。	
Elastic Beanstalk 環境をデプロイします。	プロジェクトルートディレクトリから以下のコマンドを実行すると、アプリケーションが構築されます。 `eb create comprehend-flask-env` **Note** `comprehend-flask-env` は Elastic Beanstalk 環境の名前であり、要件に応じて変更できます。含むことができるのは、英文字、数字、およびダッシュのみです。	アーキテクト、開発者

タスク	説明	必要なスキル

Amazon Comprehend を使用するようにデプロイを承認します。	アプリケーションが正常にデプロイされるかもしれませんが、デプロイに Amazon Comprehend へのアクセス権も提供する必要があります。`ComprehendFullAccess` は、デプロイされたアプリケーションに Amazon Comprehend への API 呼び出しを行うためのアクセス権限を提供する AWS 管理ポリシーです。
	
	次のコマンドを実行して、`aws-elasticbeanstalk-ec2-role` に `ComprehendFullAccess` ポリシーを添付します (このロールがデプロイの Amazon Elastic Compute Cloud (Amazon EC2 インスタンスに自動的に作成)。
	
	`aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ComprehendFullAccess --role-name aws-elasticbeanstalk-ec2-role`
	
	Important
	`aws-elasticbeanstalk-`
		開発者、セキュリティアーキテクト

タスク	説明	必要なスキル
	`ec2-role` は、アプリケーションのデプロイ時に作成されます。AWS Identity and Access Management (IAM) ポリシーをアタッチする前に、デプロイプロセスを完了する必要があります。	
デプロイしたアプリケーションにアクセスします。	アプリケーションが正常にデプロイされた後、 `eb open` コマンドを実行して、そのアプリケーションにアクセスできます。 `eb status` コマンドを実行して、デプロイに関する詳細を取得することもできます。デプロイ URL は `CNAME` に記載されています。	アーキテクト、開発者

(オプション) ML モデルに合わせてアプリケーションをカスタマイズ

タスク	説明	必要なスキル
Elastic Beanstalk が新しいモデルにアクセスすることを許可します。	Elastic Beanstalk に新しいモデルエンドポイントに必要なアクセス権限があることを保証します。たとえば、Amazon SageMaker エンドポイントを使用する場合、デプロイにはエンドポイントを呼び出すアクセス権限が必要です。	開発者、セキュリティアーキテクト

タスク	説明	必要なスキル
	これに関する詳細については、Amazon SageMaker ドキュメントのInvokeEndpointを参照してください。	

タスク	説明	必要なスキル
ユーザーデータを新しいモデルに送信します。	このアプリケーションの基になる ML モデルを変更するには、以下のファイルを変更する必要があります。 • `comprehend_helper.py` — これは Amazon Comprehend に接続してレスポンスを処理し、最終結果をアプリケーションに返す Python スクリプトです。このスクリプトでは、データを AWS クラウド上の別の AI サービスにルーティングするか、カスタムモデルのエンドポイントにデータを送信できます。このパターンを論理的に分離して再利用できるように、結果をこのスクリプトでフォーマットすることも推薦します。 • `application.py` — `comprehend_helper.py` スクリプトまたは関数の名前を変更した場合、その変更を反映するために、アプリケーション `application.py` スクリプトを更新する必要があります。	データサイエンティスト

タスク	説明	必要なスキル
ダッシュボードのビジュアライゼーションを更新します。	通常、新しい ML モデルを組み込むと、新しい結果を反映するようにビジュアライゼーションを更新する必要があります。これらのフラグは、次のファイルで定義されています。 • `templates/dashboard.html` — ビルド済みのアプリケーションでは、2 つの基本的なビジュアライゼーションしか考慮されていません。このファイルでは、ページのレイアウト全体を調整できます。 • `static/js/core.js` — このスクリプトは Flask サーバーの `/data` ルートのフォーマットされた出力をキャプチャし、Plotly を使用してビジュアライゼーションを作成します。ページのチャートを追加または更新できます。	ウェブ開発者

更新したアプリケーションをデプロイします。

タスク	説明	必要なスキル
アプリケーションの要件ファイルを更新します。	Elastic Beanstalk に変更を送信する前に、アプリケーションのルートディレクトリで以下のコマンドを実行して、	Python 開発

タスク	説明	必要なスキル
	新しい Pythonモジュールを反映するように `requirements.txt` ファイルを更新します。 `pip freeze > requirements.txt`	
Elastic Beanstalk 環境をセットアップします。	アプリケーションの変更が Elastic Beanstalk デプロイに確実に反映されるようにするには、アプリケーションのルートディレクトリに移動し、以下のコマンドを実行します。 `eb deploy` これにより、アプリケーションのコードの最新バージョンが既存の Elastic Beanstalk デプロイに送信されます。	システム管理者、アーキテクト

関連リソース

- Amazon API Gatewayと AWS Lambda を使用して Amazon SageMaker モデルエンドポイントを呼び出す
- Elastic Beanstalk への Flask アプリケーションのデプロイ
- EB CLI コマンドリファレンス
- Python 開発環境をセットアップします

追加情報

トラブルシューティングのリスト

一般的な原因とその解決策を以下に示します。

エラー 1

```
Unable to assume role "arn:aws:iam::xxxxxxxxxx:role/aws-elasticbeanstalk-ec2-role".
 Verify that the role exists and is configured correctly.
```

解決策： `eb create` を実行する時、このエラーが発生した場合、Elastic Beanstalk コンソールでサンプルアプリケーションを作成して、デフォルトのインスタンスプロファイルを作成します。詳細については、AWS Elastic Beanstalk ドキュメントのElastic Beanstalk 環境を作成 を参照してください。

エラー 2

```
Your WSGIPath refers to a file that does not exist.
```

解決策：このエラーがデプロイログで発生されます。Elastic Beanstalk が `application.py` という名前のFlask コードを想定しているためです。別の名前を選択した場合は、次のコードサンプルに示すように WSGIPath `eb config` を実行して編集します。

```
aws:elasticbeanstalk:container:python:
     NumProcesses: '1'
     NumThreads: '15'
     StaticFiles: /static/=static/
     WSGIPath: application.py
```

自分のファイル名で `application.py` を置き換えます。

Gunicorn と Procfile を利用することもできます。このアプローチの詳細については、AWS Elastic Beanstalk ドキュメントのプロファイルファイルによる WSGI サーバーの設定 を参照してください。

エラー 3

```
Target WSGI script '/opt/python/current/app/application.py' does not contain WSGI
 application 'application'.
```

解決策：Elastic Beanstalk では、Flask アプリケーションを表す変数に `application` という名前が付けられていることを想定しています。`application.py` ファイルが変数名として `application` を使用していることを確認します。

```
application = Flask(__name__)
```

エラー 4

```
The EB CLI cannot find your SSH key file for keyname
```

解決策：EB CLI を使用して、使用するキーペアを指定するか、デプロイの EC2 インスタンス用のキーペアを作成します。エラーを解決するには、`eb init -i` を実行します。一つのオプションでは、下記のように提示されます。

```
Do you want to set up SSH for your instances?
```

Y と回答して、キーペアを作成するか、既存のキーペアを指定します。

エラー 5

コードを更新して再デプロイしましたが、デプロイに変更が反映されていません。

解決策：デプロイに Git リポジトリを使用している場合、再デプロイする前に変更を追加してコミットする必要があります。

エラー 6

AWS Cloud9 IDE から Flask アプリケーションをプレビューしていて、エラーが発生しました。

解決策：詳細については、AWS Cloud9 ドキュメントの「AWS Cloud9 IDE での実行中のアプリケーションのプレビュー」を参照してください。

自然言語処理に Amazon Comprehend を使用

Amazon Comprehend を使用することを選択すると、リアルタイム分析または非同期バッチジョブを実行して、個々のテキストドキュメント内のカスタムエンティティを検出できます。Amazon Comprehend では、エンドポイントを作成することでリアルタイムで使用できるカスタムエンティティ認識モデルとテキスト分類モデルをトレーニングすることもできます。

このパターンでは、非同期バッチジョブを使用して、複数のドキュメントを含む入力ファイルからセンチメントとエンティティを検出します。このパターンで提供されるサンプルアプリケーションは、ユーザーが 1 つの列と 1 行に 1 つのテキストドキュメントを含む.csv ファイルをアップロードでき

るように設計されています。GitHub のFlask と AWS Elastic Beanstalk を使用して、Visualize AI/ML モデルの結果を視覚化 リポジトリの comprehend_helper.py ファイルが、入力ファイルを読み取り、その入力を Amazon Comprehend に送信して処理します。

BatchDetectEntities

Amazon Comprehend は、一連のドキュメントのテキストに指定されたエンティティがないか検査し、検出されたエンティティ、場所、エンティティのタイプ、および Amazon Comprehend の信頼度を示すスコアを返します。1 つの API 呼び出しで送信できるドキュメントは最大 25 個で、各ドキュメントのサイズは 5,000 バイト未満です。結果をフィルタリングして、ユースケースに基づいて特定のエンティティのみを表示できます。たとえば、'quantity' エンティティタイプをスキップして、検出されたエンティティのしきい値スコア (0.75 など) を設定できます。しきい値を選択する前に、特定のユースケースの結果を調べることを推薦します。これに関する詳細については、Amazon Comprehend ドキュメントのBatchDetectEntities を参照してください。

BatchDetectSentiment

Amazon Comprehend は、受信した複数のドキュメントを検査し、各ドキュメントの一般的なセンチメント(POSITIVE、NEUTRAL、MIXED、または NEGATIVE)を返します。1 つの API 呼び出しで送信できるドキュメントは最大 25 個で、各ドキュメントのサイズは 5,000 バイト未満です。センチメントの分析は簡単で、スコアが最も高いセンチメントを選択して最終結果に表示します。これに関する詳細については、Amazon Comprehend ドキュメントのBatchDetectSentiment を参照してください。

フラスコの設定処理

Flask サーバーは一連の設定変数 を使用して、サーバーの実行方法を制御します。これらの変数には、デバッグ出力、セッショントークン、その他のアプリケーション設定を含めることができます。アプリケーションの実行中にアクセスできるカスタム変数を定義することもできます。設定変数を設定する方法は複数あります。

このパターンでは、設定が config.py で定義され、application.py の内部で継承されます。

- **Note**

  config.py には、アプリケーションの起動時に設定される設定変数が含まれます。このアプリケーションでは、サーバーをデバッグモード で実行するようにアプリケーションに

指示する DEBUG 変数が定義されています。: 本番稼働環境でアプリケーションを実行するときは、デバッグモードを使用しないでください。 はUPLOAD_FOLDER、アプリケーションの後半で参照するように定義されたカスタム変数であり、アップロードされたユーザーデータの保存先を通知します。

- application.py はFlask アプリケーションを初期化し、config.py で定義されている設定を継承します。これは以下のコードによって実行されます。

```
application = Flask(__name__)
application.config.from_pyfile('config.py')
```

その他のパターン

- Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する
- ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用して Db2 z/OS データインサイトを生成する
- SageMaker ノートブックインスタンスに、別の AWS アカウントの CodeCommit リポジトリへの一時的なアクセスを許可する
- AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する
- Amazon Q Developer を使用して CardDemo メインフレームアプリケーションをモダナイズする
- Amazon Redshift ML機械学習を使用して高度な分析を実行する
- 自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する
- Amazon Bedrock AWS Step Functions を使用して の状態をトラブルシューティングする

分析

トピック

- Microsoft SQL Server Analysis Services の Amazon Redshift データを分析
- Amazon Athena と Amazon QuickSight を使用してネストされた JSON データを分析および視覚化する
- から Amazon S3 AWS Data Exchange へのデータ取り込みを自動化する
- AWS CloudFormation のテンプレートを使用して、AWS Glue での暗号化の適用を自動化
- AWS DataOps Development Kit を使用して Google Analytics データを取り込み、変換、分析するためのデータパイプラインを構築する
- Amazon Kinesis Video Streamsと AWS Fargate を使用してビデオ処理パイプラインを構築する
- AWS Glue を使用して Amazon S3 から Amazon Redshift にデータを段階的にロードする ETL サービスパイプラインを構築
- Amazon DataZone を使用してエンタープライズデータメッシュを構築する AWS CDKAWS CloudFormation
- AWS のサービスを使用してバリューアットリスク (VaR) を計算
- Amazon Athena を使用して共有 AWS Glue データカタログへのクロスアカウントアクセスを構成する
- Teradata NORMALIZE 時間的特徴量を Amazon Redshift SQL に変換
- Teradata RESET WHEN 特徴量を Amazon Redshift SQL に変換
- インフラストラクチャをコードとして使用して、AWS クラウドにサーバーレスデータレイクをデプロイして管理する
- 起動時に Amazon EMR クラスターのタグ付けを強制する
- Amazon S3 へのAmazon EMR ロギングが有効になっていることを確認する
- AWS Glue ジョブと Python を使用してテストデータを生成します
- AWS IoT Greengrass を使用して IoT データをコスト効率よく直接 Amazon S3 に取り込む
- Lambda 関数を使用して一時的な EMR クラスターで Spark ジョブを起動する
- AWS Glue を使用して Apache Cassandra ワークロードを Amazon Keyspaces に移行する
- WANdisco LiveData Migrator を使用して Hadoop データを Amazon S3 に移行する
- Oracle ビジネスインテリジェンス 12c をオンプレミスサーバーから AWS クラウドに移行
- MirrorMaker を使用して、オンプレミスの Apache Kafka クラスターを Amazon MSK に移行する

- ELK スタックを AWS 上のElasticクラウドに移行する
- Starburst AWS クラウド を使用してデータを に移行する
- AWS での入力ファイルサイズの ETL 取り込みを最適化する
- AWS Step Functions を使用して ETL パイプラインを検証、変換、パーティショニングでオーケストレーションします
- Amazon Redshift ML機械学習を使用して高度な分析を実行する
- Amazon Athena を使用して SQL で Amazon DynamoDB テーブルをクエリする Amazon Athena
- Athena による Amazon DynamoDB テーブルへのアクセス、クエリ、結合
- 組織間でデータを共有するための最小限の実行可能なデータスペースを設定する
- スカラー Python UDF を使用した Amazon Redshift クエリー結果の言語固有のソートの設定
- さまざまな AWS リージョンの S3 バケットからのイベント通知に Lambda 関数をサブスクライブする
- データを Apache Parquet に変換するための 3 つの AWS Glue ETL ジョブタイプ
- Amazon Athena および Amazon QuickSight を使用して Amazon Redshift 監査ログを視覚化
- Amazon QuickSight を使用してすべての AWS アカウントの IAM 認証情報レポートを視覚化します
- その他のパターン

Microsoft SQL Server Analysis Services の Amazon Redshift データを分析

作成者：Sunil Vora (AWS)

概要

このパターンは、データベースアクセスには Intellisoft OLE DB プロバイダーまたは CData ADO.NET プロバイダーにより、Microsoft SQL Server Analysis Servicesの Amazon Redshift データに接続して分析する方法を説明しています。

Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。SQL Server Analysis Services は、データマートや Amazon Redshift などのデータウェアハウスからのデータを分析するために使用できるオンライン分析処理 (OLAP) ツールです。SQL Server Analysis Services を使用してデータから OLAP キューブを作成し、迅速で高度なデータ分析を行うことができます。

前提条件と制限

引き受け

- このパターンは Amazon Redshift の SQL Server Analysis Services と Intellisoft OLE DB Provider または CData ADO.NET Provider を Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに設定する方法を説明しています。または、企業データセンターのホストに両方をインストールすることもできます。

前提条件

- アクティブな AWS アカウント
- Amazon Redshift クラスターと認証情報

アーキテクチャ

ソーステクノロジースタック

- Amazon Redshift クラスター

ターゲットテクノロジースタック

- Microsoft SQL Server Analysis Services

ソースアーキテクチャとターゲットアーキテクチャ

ツール

- 「Microsoft Visual Studio 2019 (コミュニティエディション)」
- 「Amazon Redshift 用インテリソフト OLE DB プロバイダー (試用版)」 または「Amazon Redshift 用 CData ADO.NETプロバイダー (試用版)」

エピック

テーブルを分析

タスク	説明	必要なスキル
インポートするテーブルとデータを分析します。	インポートする Amazon Redshift テーブルとそのサイズを指定します。	DBA

EC2 インスタンスを設定してツールをインストール

タスク	説明	必要なスキル
EC2 インスタンスをセットアップします。	AWS アカウントで、プライベートサブネットまたはパブリックサブネットに EC2 インスタンスを作成します。	システム管理者
データベースアクセス用のツールをインストールします。	「Amazon Redshift 用インテリソフト OLE DB プロバイダー (試用版)」(または「Amazon Redshift 用 CData ADO.NETプロバイダー」)を	システム管理者

タスク	説明	必要なスキル
	ダウンロードして、インストールします。	
Visual Studio をインストールします。	「Visual Studio 2019 (コミュニティエディション)」をダウンロードしてインストールします。	システム管理者
拡張機能をインストールします。	Microsoft Analysis Services Projects の拡張機能をVisual Studio にインストールします。	システム管理者
プロジェクトを作成します。	Amazon Redshift データを保存するための新しい表形式モデルプロジェクトを Visual Studio で作成します。Visual Studio では、プロジェクトを作成するときに Analysis Services Tabular Project オプションを選択します。	DBA

データソースの作成とテーブルのインポート

タスク	説明	必要なスキル
Amazon Redshift データソースを作成します。	Amazon Redshift 用の Intelliso ft OLE DB プロバイダー (または Amazon Redshift 用の CData ADO.NET プロバイダー) と Amazon Redshift 認証情報により、 Amazon Redshift データソースを作成します。	Amazon Redshift、DBA

タスク	説明	必要なスキル
テーブルをインポートします。	Amazon Redshift からテーブルとビューを選択し、SQL Server Analysis Services プロジェクトにインポートします。	Amazon Redshift、DBA

移行後のクリーンアップ

タスク	説明	必要なスキル
EC2 インスタンスを削除します。	以前に起動した EC2 インスタンスを削除します。	システム管理者

関連リソース

- 「Amazon Redshift」(AWS のドキュメント)
- 「SQL Server Analysis Servicesのインストール」(Microsoft のドキュメント)
- 「Tabular Model Designer」(Microsoft のドキュメント)
- 「高度な分析用の OLAP キューブの概要」(Microsoft のドキュメント)
- 「Microsoft Visual Studio 2019 (コミュニティエディション)」
- 「Amazon Redshift 用インテリソフト OLE DB プロバイダー (試用版)」
- 「Amazon Redshift 用 CData ADO.NET プロバイダー (試用版)」

Amazon Athena と Amazon QuickSight を使用してネストされた JSON データを分析および視覚化する

作成者: Anoop Singh (AWS)

概要

このパターンでは、Amazon Athena を使用してネストされた JSON 形式のデータ構造を表形式ビューに変換し、Amazon QuickSight でデータを視覚化する方法について説明します。

運用システムからの API を活用したデータフィードに JSON 形式のデータを使用して、データ製品を作成できます。このデータは、顧客と製品とのやり取りをよりよく理解するのに役立つため、ユーザーエクスペリエンスを調整し、結果を予測できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ネストされたデータ構造を表す JSON ファイル (このパターンはサンプルファイルを提供します)

[Limitations:] (制限:)

- JSON 機能は、Athena の既存の SQL 指向関数とうまく統合されています。ただし、これらは ANSI SQL 互換ではなく、JSON ファイルは各レコードを別々の行に格納することが期待されます。Athena の `ignore.malformed.json`プロパティを使用して、不正な形式の JSON レコードを null 文字に変換するか、エラーを生成するかを示す必要がある場合があります。詳細については、Athena ドキュメントの「JSON データを読み取るためのベストプラクティス」を参照してください。
- このパターンでは、JSON 形式のデータを単純かつ少量のみ考慮します。これらの概念を大規模に使用する場合は、データパーティショニングを適用し、データをより大きなファイルに統合することを検討してください。

アーキテクチャ

次の図は、このパターンのアーキテクチャとワークフローを示しています。ネストされたデータ構造は、JSON 形式で Amazon Simple Storage Service (Amazon S3) に保存されます。Athena で

は、JSON データは Athena データ構造にマッピングされます。次に、データを分析するためのビューを作成し、QuickSight でデータ構造を視覚化します。

ツール

AWS サービス

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。このパターンでは、Amazon S3 を使用して JSON ファイルを保存します。
- Amazon Athena は、標準 SQL を使用して Amazon S3 でデータを直接分析するのに役立つ対話型のクエリサービスです。このパターンでは、Athena を使用して JSON データをクエリおよび変換します。でいくつかのアクションを使用すると AWS Management Console、Amazon S3 のデータに Athena をポイントし、標準 SQL を使用して 1 回限りのクエリを実行できます。Athena はサーバーレスであるため、設定や管理を行うインフラストラクチャはなく、実行したクエリに対してのみ料金が発生します。Athena は自動的にスケーリングされ、クエリが並行して実行されるため、大規模なデータセットや複雑なクエリでも結果が高速になります。
- Amazon QuickSight は、単一のダッシュボードでデータを視覚化、分析、レポートするのに役立つクラウドスケールのビジネスインテリジェンス (BI) サービスです。QuickSight を使用すると、機械学習 (ML) インサイトを含むインタラクティブなダッシュボードを簡単に作成して公開できます。これらのダッシュボードには任意のデバイスからアクセスし、アプリケーション、ポータル、ウェブサイトに埋め込むことができます。

コードの例

次の JSON ファイルは、このパターンで使用できるネストされたデータ構造を提供します。

```
{
  "symbol": "AAPL",
  "financials": [
    {
      "reportDate": "2017-03-31",
      "grossProfit": 20591000000,
      "costOfRevenue": 32305000000,
      "operatingRevenue": 52896000000,
      "totalRevenue": 52896000000,
      "operatingIncome": 14097000000,
      "netIncome": 11029000000,
```

```
      "researchAndDevelopment": 2776000000,
      "operatingExpense": 6494000000,
      "currentAssets": 101990000000,
      "totalAssets": 334532000000,
      "totalLiabilities": 200450000000,
      "currentCash": 15157000000,
      "currentDebt": 13991000000,
      "totalCash": 67101000000,
      "totalDebt": 98522000000,
      "shareholderEquity": 134082000000,
      "cashChange": -1214000000,
      "cashFlow": 12523000000,
      "operatingGainsLosses": null
    }
  ]
}
```

エピック

S3 バケットをセットアップする

タスク	説明	必要なスキル
S3 バケットを作成する。	JSON ファイルを保存するバケットを作成するには、 にサインインし AWS Management Console、Amazon S3 コンソールを開き、バケットの作成を選択します。詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	システム管理者
ネストされた JSON データを追加します。	JSON ファイルを S3 バケットにアップロードします。サンプル JSON ファイルについては、前のセクションを参照してください。手順については、Amazon S3 ドキュメントの「オブジェクトのアッ	システム管理者

タスク	説明	必要なスキル
	プロード」を参照してください。	

Athena でデータを分析する

タスク	説明	必要なスキル
JSON データをマッピングするためのテーブルを作成します。	1. Athena コンソールを開きます。 2. Athena ドキュメントの指示に従ってデータベースを作成します。 3. データベースメニューから、作成したデータベースを選択します。 4. クエリエディタで、次のようなCREATE TABLEステートメントを入力します。 `CREATE EXTERNAL TABLE financials_json ( symbol string, financials array< struct<reportdate: string, grossprofit: bigint, totalrevenue: bigint, totalcash: bigint, totaldebt: bigint, researchanddevelopment: bigint>> )`	開発者

タスク	説明	必要なスキル

	``
	ROW FORMAT SERDE
	  'org.openx.data.js
	onserde.JsonSerDe'
	  LOCATION 's3://s3b
	ucket-for-athena/'
	``
	ここで、 は JSON ファイルを含む S3 バケットの場所LOCATIONを指定します。
	5. Run を選択してテーブルを作成します。
	
	テーブルの作成の詳細については、Athena のドキュメントを参照してください。	

タスク	説明	必要なスキル
データ分析用のビューを作成します。	1. Athena コンソールを開きます。 2. Athena ドキュメントの指示に従ってデータベースを作成します。 3. データベースメニューから、作成したデータベースを選択します。 4. クエリエディタで、次のようなCREATE VIEWステートメントを入力します。 5. [Run] (実行) をクリックしてビューを作成します。	開発者

```
CREATE OR REPLACE VIEW
 financial_json_view
 AS
  SELECT
  symbol,
  financials[1].repo
rtdate one_repor
t_date, -- indexes
 start with 1
  financials[1].tota
lrevenue one_total
_revenue,
  financials[1].repo
rtdate another_r
eport_date,
  financials[1].tota
lrevenue another_t
otal_revenue
FROM financials_json
 where symbol='AAPL'
ORDER BY
  1
```

タスク	説明	必要なスキル
	ビューの作成の詳細については、Athena のドキュメントを参照してください。	
データを分析して検証します。	1. Athena コンソールを開きます。 2. クエリエディタで、前のステップで作成したビューを使用してクエリを実行します。 3. JSON ファイルに対してデータを検証し、列名とデータ型が正しくマッピングされていることを確認します。	開発者

QuickSight でデータを視覚化する

タスク	説明	必要なスキル
QuickSight でデータソースとして Athena を設定します。	1. QuickSight コンソールを開きます。 2. [Datasets (データセット)]、[New dataset (新しいデータセット)] の順に選択します。 3. データソースとして Athena を選択します。 4. 作成したビューを含むデータベースを選択します。 5. データセットを作成するビューを選択します。	システム管理者

タスク	説明	必要なスキル
	6. データセット作成の完了ページで、データの直接クエリを選択します。 7. [Visualize (視覚化する)] を選択します。	
QuickSight でデータを視覚化します。	1. データセットを視覚化したら、左側のペインからビジュアルを選択し、データセットのフィールドを選択します。詳細については、QuickSight ドキュメントのチュートリアルを参照してください。 2. 変更を分析に保存します。 3. ダッシュボードの発行 を選択して、作成したビジュアルを公開します。	データアナリスト

関連リソース

- Amazon Athena ドキュメント
- Amazon QuickSight チュートリアル
- ネストされた JSON の使用 (ブログ記事)

から Amazon S3 AWS Data Exchange へのデータ取り込みを自動化する

作成者: Adnan Alvee (AWS) と Manikanta Gona (AWS)

概要

このパターンは、Amazon Simple Storage Service (Amazon S3) のデータレイク AWS Data Exchange に からデータを自動的に取り込むことができる AWS CloudFormation テンプレートを提供します。

AWS Data Exchange は、AWS Cloud 内のファイルベースのデータセットを安全に交換できるサービスです。 AWS Data Exchange データセットはサブスクリプションベースです。サブスクライバーは、プロバイダーが新しいデータをパブリッシュしたときに、データセットの改訂版にアクセスすることもできます。

AWS CloudFormation テンプレートは、Amazon CloudWatch Events と AWS Lambda 関数にイベントを作成します。このイベントは、お客様がサブスクライブしているデータセットの更新を監視します。更新がある場合、CloudWatch は Lambda 関数を開始し、指定した S3 バケットにデータをコピーします。データが正常にコピーされると、Lambda から Amazon Simple Notification Service (Amazon SNS) 通知が送信されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- でのデータセットへのサブスクリプション AWS Data Exchange

機能制限

- AWS CloudFormation テンプレートは、サブスクライブされたデータセットごとに個別にデプロイする必要があります AWS Data Exchange。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Lambda
- Amazon S3
- AWS Data Exchange
- Amazon CloudWatch
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

AWS CloudFormation テンプレートは、データレイクに取り込むデータセットに複数回使用できます。

ツール

- AWS Data Exchange を使用すると、 AWS お客様は でファイルベースのデータセットを安全に交換できます AWS クラウド。サブスクライバーは、認定されたデータプロバイダーからの何千もの製品を検索してサブスクライブすることができます。その後、データセットをすばやくダウンロードするかAmazon S3 にコピーして、さまざまな AWS 分析および機械学習サービスで使用できます。を持つユーザーは誰でもサブ AWS Data Exchange スクライバーになる AWS アカウント ことができます。
- AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。使用したコンピューティング時間に対してのみお支払いいただきます。コードが実行中でなければ料金はかかりません。Lambda を使用すると、ほぼすべてのタイプのアプリケーションまたはバックエンドサービスのコードを管理なしで実行できます。Lambda は、高可用性コンピューティングインフラストラクチャでコードを実行し、サーバーとオペレーティングシステムのメンテナンス、容量のプロビジョニングと自動スケーリング、コードのモニタリング、ログ記録など、すべてのコンピューティングリソースを管理します。
- Amazon S3 はインターネット用のストレージを提供します。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができま

す。CloudWatch Events が発生すると、運用上の変更が認識されます。オペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、修正アクションを実行します。CloudWatch Events を使用して、cron式 や rate式により特定の時間に自己トリガーする自動アクションをスケジュールすることもできます。

- Amazon Simple Notification Service (Amazon SNS) を使用すると、アプリケーション、エンドユーザー、デバイスはクラウドから通知を即座に送受信できます。Amazon SNS は、高スループットのプッシュベースの多対多メッセージング用のトピック (通信チャネル) を提供します。Amazon SNS トピックを使用すると、パブリッシャーは Amazon Simple Queue Service (Amazon SQS) キュー、Lambda 関数、HTTP/S ウェブフックなど、並列処理のために多数のサブスクライバーにメッセージを配信できます。Amazon SNS を使用して、モバイルプッシュ、SMS、E メールを使用してエンドユーザーに通知を送信することもできます。

エピック

データセットをサブスクライブする

タスク	説明	必要なスキル
データセットをサブスクライブする	AWS Data Exchange コンソールで、データセットをサブスクライブします。手順については、 AWS ドキュメントの「 でのデータ製品のサブスクライブ AWS Data Exchange」を参照してください。	AWS 全般
データセットの属性に注意してください。	データセットの AWS リージョン、ID、リビジョン ID を書き留めます。これは、次のステップで AWS CloudFormation テンプレートに必要になります。	AWS 全般

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
S3 バケットとフォルダを作成する。	Amazon S3 に既にデータレイクがある場合は、取り込むデータを保存するフォルダを作成します AWS Data Exchange。テスト目的でテンプレートをデプロイする場合は、新しい S3 バケットを作成し、次のステップのためにバケット名とフォルダプレフィックスを書き留めておきます。	AWS 全般
AWS CloudFormation テンプレートをデプロイします。	このパターンの添付ファイルとして提供されている AWS CloudFormation テンプレートをデプロイします。手順については、 AWS CloudFormation ドキュメントを参照してください。 AWS アカウント、データセット、および S3 バケット設定に対応するように、次のパラメータを設定します。データセット AWS リージョン、データセット ID、リビジョン ID、S3 バケット名 (などDOC-EXAMPLE-BUCKET)、フォルダプレフィックス (などmyfolder/)、SNS 通知の E メール。データセット名パラメータは任意の名前に設定できま	AWS 全般

タスク	説明	必要なスキル
	す。テンプレートをデプロイすると、Lambda 関数が実行され、データセットで使用可能な最初のデータセットが自動的に取り込まれます。その後、データセットに新しいデータが到着すると、自動的に取り込まれます。	

関連リソース

- でのデータ製品のサブスクライブ AWS Data Exchange (AWS Data Exchange ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CloudFormation のテンプレートを使用して、AWS Glue での暗号化の適用を自動化

ディオゴ・ゲデス (AWS) により作成

概要

このパターンは、AWS CloudFormation テンプレートを使用して AWS Glue で暗号化の適用をセットアップして自動化する方法を示しています。このテンプレートは、暗号化を実施するために必要なすべての設定とリソースを作成します。これらのリソースには、初期設定、Amazon EventBridge ルールによって作成された予防コントロール、および AWS Lambda 関数が含まれます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- CloudFormation テンプレートとそのリソースをデプロイするための権限

機能制限

このセキュリティコントロールは地域ごとに行われます。AWS Glue で暗号化の適用を設定する各 AWS リージョンに、セキュリティコントロールをデプロイする必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon CloudWatch Logs (AWS Lambda から)
- Amazon EventBridge ルール
- AWS CloudFormation スタック
- AWS CloudTrail
- AWS Identity and Access Management (IAM) 管理ロールとポリシー
- AWS Key Management Service (AWS KMS)
- AWS KMS alias
- AWS Lambda 関数

- Systems Manager パラメータストア

ターゲットアーキテクチャ

次の図は、AWS Glue で暗号化の適用を自動化する方法を示しています。

この図表は、次のワークフローを示しています：

1. 「CloudFormation テンプレート」は、AWS Glue での暗号化実施のための初期設定や検出制御を含むすべてのリソースを作成します。
2. EventBridge ルールは、暗号化設定の状態変化を検出します。
3. Lambda 関数が呼び出され、CloudWatch ログを通じて評価とロギングが行われます。非準拠の検出では、パラメータストアは AWS KMS キーの Amazon リソースネーム (ARN) で復元されます。サービスは暗号化が有効になった状態で準拠状態に修正されます。

自動化とスケール

「AWS Organizations」を使用している場合は、「AWS CloudFormation StackSets」を使用して、AWS Glue で暗号化の強制を有効にしたい複数のアカウントにこのテンプレートをデプロイできます。

ツール

- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- 「Amazon EventBridge」は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。たとえば、Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「AWS CloudTrail」は、AWS アカウントの運用とリスクの監査、ガバナンスとコンプライアンスを支援します。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。

- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。

コード

このパターンのコードは、GitHub 内の「aws-custom-guardrail-event-driven」リポジトリで利用できます。

ベストプラクティス

AWS Glue は、「AWS Glue」でジョブを作成したり、「開発エンドポイントを使用してスクリプトを開発したりするための保存データ暗号化」をサポートしています。

以下のベストプラクティスを考慮します。

- AWS KMS キーを使用して暗号化されたデータを残りの部分に書き込むように、ETL ジョブと開発エンドポイントを設定します。
- AWS KMS で管理するキーを使用して、「AWS Glue データカタログ」に保存されているメタデータを暗号化します。
- AWS KMS キーを使用して、ジョブのブックマークや、「クローラー」および ETL ジョブで生成されたログを暗号化します。

エピック

CloudFormation のテンプレートを起動

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	GitHub `aws-custom-guardrail-event-`	クラウドアーキテクト

タスク	説明	必要なスキル
	driven.yaml 「リポジトリ」からテンプレートをダウンロードし、テンプレートを「デプロイ」します。CREATE_COMPLETE ステータスはテンプレートが正常にデプロイされたことを示します。 **Note** テンプレートには入力パラメータは必要ありません。	

AWS Glue の暗号化設定を確認します。

タスク	説明	必要なスキル
AWS KMS キー設定を確認します。	1. AWS マネジメントコンソールにサインインし、「AWS Glue コンソール」を開きます。 2. ナビゲーションペインのデータカタログで、カタログ設定を選択します。 3. メタデータの暗号化と接続パスワードの暗号化の設定にフラグが付けられ、KMSKeyGlue 使用するように設定されていることを確認します。	クラウドアーキテクト

暗号化の適用をテスト

タスク	説明	必要なスキル
クラウドフォーメーションの暗号化設定を確認します。	1. AWS マネジメントコンソールにサインインし、次に「CloudFront コンソール」を開きます。 2. ナビゲーションペインでスタックを選択し、相応しいスタックを選択します。 3. [リソース] タブを選択してください。 4. リソーステーブルで、論理 ID による暗号化設定を探します。	クラウドアーキテクト
プロビジョニングされたインフラストラクチャーを非準拠状態に切り替えます。	1. AWS マネジメントコンソールにサインインし、「AWS Glue コンソール」を開きます。 2. ナビゲーションペインのデータカタログで、カタログ設定を選択します。 3. メタデータの暗号化チェックボックスをオフにします。 4. 接続パスワードを暗号化チェックボックスをオフにします。 5. [Save] を選択します。 6. AWS Glue コンソールを更新します。	クラウドアーキテクト

タスク	説明	必要なスキル
	チェックボックスをオフにすると、ガードレールは AWS Glue の非準拠状態を検出し、暗号化の設定ミスを自動的に修正することでコンプライアンスを強制します。そのため、ページを更新した後は、暗号化のチェックボックスが再びオンになっているはずです。	

関連リソース

- 「AWS CloudFormation コンソールでのスタックの作成」(AWS CloudFormation ドキュメント)
- 「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成」(Amazon CloudWatch ドキュメント)
- 「AWS Glue での暗号化のセットアップ」(AWS Glue ドキュメント)

AWS DataOps Development Kit を使用して Google Analytics データを取り込み、変換、分析するためのデータパイプラインを構築する

作成者: Anton Kukushkin (AWS)、Rudy Puig (AWS)

概要

このパターンでは、 AWS DataOps Development Kit (AWS DDK) などを使用して Google Analytics データを取り込み、変換、分析するためのデータパイプラインを構築する方法について説明します AWS のサービス。 AWS DDK は、データワークフローと最新のデータアーキテクチャの構築に役立つオープンソースの開発フレームワークです AWS。 AWS DDK の主な目的の 1 つは、パイプラインのオーケストレーション、インフラストラクチャの構築、そのインフラストラクチャの背後にある DevOps の作成など、通常、労力のかかるデータパイプラインタスクに費やす時間と労力を節約することです。これらの労力のかかるタスクを DDK AWS にオフロードして、コードやその他の価値の高いアクティビティの作成に集中できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 設定済みの Google アナリティクス用の Amazon AppFlow コネクタ
- Python と pip (Python のパッケージマネージャ)
- インストールおよび設定済みの Git
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- AWS Cloud Development Kit (AWS CDK)、インストール済み

製品バージョン

- Python 3.7 以降
- pip 9.0.3 以降

アーキテクチャ

テクノロジースタック

- Amazon AppFlow
- Amazon Athena
- Amazon CloudWatch
- Amazon EventBridge
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Queue Service (Amazon SQS)
- AWS DataOps 開発キット (AWS DDK)
- AWS Lambda

ターゲット アーキテクチャ

次の図は、Google アナリティクスのデータを取り込み、変換、分析するイベント駆動型のプロセスを示しています。

この図表は、次のワークフローを示しています：

1. Amazon CloudWatch のスケジュールされたイベントルールは Amazon AppFlow を呼び出します。
2. Amazon AppFlow は Google アナリティクスのデータを S3 バケットに取り込みます。
3. データが S3 バケットに取り込まれると、EventBridge のイベント通知が生成され、CloudWatch イベントルールによってキャプチャされ、Amazon SQS キューに入れられます。
4. Lambda 関数は、Amazon SQS キューからのイベントを消費し、それぞれの S3 オブジェクトを読み取り、オブジェクトを Apache Parquet 形式に変換し、変換されたオブジェクトを S3 バケットに書き込み、 AWS Glue Data Catalog テーブル定義を作成または更新します。
5. Athena クエリがテーブルに対して実行されます。

ツール

AWS ツール

- Amazon AppFlow は、Software as a Service (SaaS) アプリケーション間でデータを安全に交換できるようにするフルマネージド型の統合サービスです。
- Amazon Athena は、標準 SQL を使用して Amazon S3 でデータを直接分析するのに役立つ対話型のクエリサービスです。

- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Simple Queue Service (Amazon SQS) は、分散ソフトウェアシステムとコンポーネントの統合と分離に役立つ、安全で耐久性があり、利用可能なホスト型キューを提供します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それをプロビジョニングするためのフレームワークです AWS CloudFormation。
- AWS DataOps Development Kit (AWS DDK) は、データワークフローと最新のデータアーキテクチャの構築に役立つオープンソースの開発フレームワークです AWS。

コード

このパターンのコードは、GitHub AWS DataOps Development Kit (AWS DDK) および Analyzing Google Analytics data with Amazon AppFlow、Amazon Athena、 AWS DataOps Development Kit リポジトリで入手できます。

エピック

環境の準備

タスク	説明	必要なスキル
ソースコードを複製します。	ソースコードのクローンを作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samp`	DevOps エンジニア

タスク	説明	必要なスキル
	`les/aws-ddk-exampl es.git`	
仮想環境を作成します。	ソースコードディレクトリに移動し、以下のコマンドを実行して仮想環境を作成します。 `cd google-analytics-data-using-appflow/ python && python3 -m venv .venv`	DevOps エンジニア
依存関係をインストールします。	次のコマンドを実行して、仮想環境を有効にし、依存関係をインストールします。 `source .venv/bin/ activate && pip install -r requirements.txt`	DevOps エンジニア

データパイプラインを使用するアプリケーションをデプロイ

タスク	説明	必要なスキル
環境を起動します。	1. AWS CLI が の有効な認証情報で設定されていることを確認します AWS アカウント。詳細については、 AWS CLI ドキュメントの「名前付きプロファイルの使用」を参照してください。 2. `cdk bootstrap --profile [AWS_PROF`	DevOps エンジニア

タスク	説明	必要なスキル
	ILE] コマンドを実行します。	
データをデプロイします。	データパイプラインをデプロイするには、cdk deploy --profile [AWS_PROFILE] コマンドを実行します。	DevOps エンジニア

デプロイをテストする

タスク	説明	必要なスキル
スタックのステータスを検証します。	1. AWS CloudFormation コンソールを開きます。 2. Stacks ページで、DdkAppflowAthenaStack スタックのステータスが CREATE_COMPLETE であることを確認します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
AWS::AppFlow::Flow リソースの作成中にデプロイが失敗し、次のエラーが表示されます。Connector Profile with name ga-connection does not exist	Google アナリティクス用の Amazon AppFlow コネクタを作成して ga-connection 名前を付けたことを確認します。 手順については、Amazon AppFlow ドキュメントの「Google アナリティクス」を参照してください。

関連リソース

- AWS DataOps 開発キット (AWS DDK) (GitHub)
- AWS DDK の例 (GitHub)

追加情報

AWS DDK データパイプラインは、1 つ以上のステージで構成されます。次のコード例では、AppFlowIngestionStage を使用して Google Analytics からデータを取り込み、SqsToLambdaStage を使用してデータ変換を処理し、AthenaSQLStage を使用して Athena クエリを実行します。

まず、次のコード例に示すように、データ変換ステージと取り込みステージを作成します。

```
        appflow_stage = AppFlowIngestionStage(
            self,
            id="appflow-stage",
            flow_name=flow.flow_name,
        )
        sqs_lambda_stage = SqsToLambdaStage(
            self,
            id="lambda-stage",
            lambda_function_props={
                "code": Code.from_asset("./ddk_app/lambda_handlers"),
                "handler": "handler.lambda_handler",
                "layers": [
                    LayerVersion.from_layer_version_arn(
                        self,
                        id="layer",
                        layer_version_arn=f"arn:aws:lambda:
{self.region}:336392948345:layer:AWSDataWrangler-Python39:1",
                    )
                ],
                "runtime": Runtime.PYTHON_3_9,
            },
        )
        # Grant lambda function S3 read & write permissions
        bucket.grant_read_write(sqs_lambda_stage.function)
        # Grant Glue database & table permissions
        sqs_lambda_stage.function.add_to_role_policy(
            self._get_glue_db_iam_policy(database_name=database.database_name)
```

```
        )
        athena_stage = AthenaSQLStage(
            self,
            id="athena-sql",
            query_string=[
                (
                    "SELECT year, month, day, device, count(user_count) as cnt "
                    f"FROM {database.database_name}.ga_sample "
                    "GROUP BY year, month, day, device "
                    "ORDER BY cnt DESC "
                    "LIMIT 10; "
                )
            ],
            output_location=Location(
                bucket_name=bucket.bucket_name, object_key="query-results/"
            ),
            additional_role_policy_statements=[
                self._get_glue_db_iam_policy(database_name=database.database_name)
            ],
        )
```

次に、次のコード例に示すように、DataPipeline コンストラクトを使用して、EventBridge ルールを使用してステージを「接続」します。

```
        (
            DataPipeline(self, id="ingestion-pipeline")
            .add_stage(
                stage=appflow_stage,
                override_rule=Rule(
                    self,
                    "schedule-rule",
                    schedule=Schedule.rate(Duration.hours(1)),
                    targets=appflow_stage.targets,
                ),
            )
            .add_stage(
                stage=sqs_lambda_stage,
                # By default, AppFlowIngestionStage stage emits an event after the flow
run finishes successfully
                # Override rule below changes that behavior to call the the stage when
data lands in the bucket instead
                override_rule=Rule(
                    self,
```

```
                "s3-object-created-rule",
                event_pattern=EventPattern(
                    source=["aws.s3"],
                    detail={
                        "bucket": {"name": [bucket.bucket_name]},
                        "object": {"key": [{"prefix": "ga-data"}]},
                    },
                    detail_type=["Object Created"],
                ),
                targets=sqs_lambda_stage.targets,
            ),
        )
        .add_stage(stage=athena_stage)
    )
```

その他のコード例については、GitHub [Analyzing Google Analytics data with Amazon AppFlow、Amazon Athena、 AWS DataOps Development Kit] リポジトリを参照してください。

Amazon Kinesis Video Streamsと AWS Fargate を使用してビデオ処理パイプラインを構築する

作成者: Piotr Chotkowski (AWS)、Pushparaju Thangavel (AWS)

概要

Amazon Kinesis Video Streams と AWS Fargate を使用してビデオストリームからフレームを抽出し、さらに処理できるように Amazon Simple Storage Service (Amazon S3) にイメージファイルとして保存します。

このパターンは Java Maven プロジェクト形式でサンプルアプリケーションを提供します。このアプリケーションは、AWS Cloud Development Kit (AWS CDK) を使用して AWS インフラストラクチャを定義します。フレーム処理ロジックとインフラストラクチャ定義は Java プログラミング言語で記述されています。このサンプルアプリケーションを、独自のリアルタイムビデオ処理パイプラインを開発したり、機械学習パイプラインのビデオ前処理ステップを構築するための基礎として使用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Java SE Development Kit (JDK) 11 をインストール済み
- Apache Maven をインストール済み
- AWS Cloud Development Kit (AWS CDK) をインストール済み
- AWS コマンドラインインターフェイス (AWS CLI) バージョン 2
- Docker (AWS Fargate タスク定義で使用する Docker イメージを構築するために必要) をインストール済み

機能制限

このパターンは、概念実証として、または今後の開発の基礎となることを目的としています。本稼働環境では、現行の形式を使用しないでください。

製品バージョン

- このパターンは AWS CDK バージョン 1.77.0 でテストされました (AWS CDK バージョンを参照)
- JDK 11
- AWS CLI バージョン 2

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Kinesis Video Streams
- AWS Fargate タスク
- Amazon Simple Queue Service (Amazon SQS) キュー
- Amazon S3 バケット

ターゲット アーキテクチャ

ユーザーは Kinesis ビデオストリームを作成し、動画をアップロードして、入力 Kinesis ビデオストリームと出力 S3 バケットの詳細を含む JSON メッセージを SQS キューに送信します。コンテナ内でメインアプリケーションを実行している AWS Fargate は、SQS キューからメッセージを取得し、フレームの抽出を開始します。各フレームはイメージファイルに保存され、ターゲット S3 バケットに格納されます。

自動化とスケール

サンプルアプリケーションは、単一の AWS リージョン内で水平方向と垂直方向の両方にスケーリングできます。水平スケーリングは、SQS キューから読み取る、デプロイされた AWS Fargate タスクの数を増やすことで実現できます。垂直スケーリングは、アプリケーションのフレーム分割スレッドと画像公開スレッドの数を増やすことで実現できます。これらの設定は、AWS CDK の QueueProcessingFarGateService リソースの定義で環境変数としてアプリケーションに渡されます。AWS CDK スタックのデプロイの性質上、このアプリケーションは追加作業なしで複数の AWS リージョンとアカウントにデプロイできます。

ツール

ツール

- AWS CDK は、TypeScript、JavaScript、Python、Java、C#/.Net などのプログラミング言語を使用してクラウドインフラストラクチャとリソースを定義するためのソフトウェア開発フレームワークです。
- Amazon Kinesis Video Streams は、デバイスから AWS クラウドへの動画のライブストリーミングに使用したり、あるいはリアルタイムの動画処理やバッチ指向の動画分析のためのアプリケーションを構築できる完全管理の AWS のサービスです。
- AWS Fargate は、コンテナ用のサーバーレスコンピューティングエンジンです。Fargate を使用すると、サーバーのプロビジョニングと管理が不必要になるため、アプリケーションの開発への集中が容易になります。
- Amazon S3 は、スケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。
- Amazon SQS は、完全マネージド型のメッセージキューイングサービスであり、マイクロサービス、分散システム、およびサーバーレスアプリケーションのデカップリングとスケーリングを容易にします。

コード

- サンプルアプリケーションプロジェクト (`frame-splitter-code.zip`) の.zip ファイルが添付されます。

エピック

インフラストラクチャをデプロイする

タスク	説明	必要なスキル
Docker デーモンを開始します。	ローカルシステムで Docker デーモンを開始します。AWS CDK は Docker を使用して AWS Fargate タスクの使用イメージを構築します。次の手順に進む前に Docker を実行する必要があります。	開発者、DevOps エンジニア
プロジェクトをビルドします。	`frame-splitter-code` サンプルアプリケーション (添	開発者、DevOps エンジニア

タスク	説明	必要なスキル
	付) をダウンロードし、ローカルマシンのフォルダに抽出します。インフラストラクチャをデプロイする前に、Java Maven プロジェクトをビルドする必要があります。コマンドプロンプトで、プロジェクトのルートディレクトリに移動し、次のコマンドを実行してプロジェクトをビルドします。 `mvn clean install`	
AWS CDK をブートストラップします。	(AWS CDK を初めて使用するユーザーのみ) AWS CDK を初めて使用する場合は、AWS CLI コマンドを実行して環境をブートストラップする必要があります。 `cdk bootstrap --profile "$AWS_PROFILE_NAME"` `$AWS_PROFILE_NAME` は、AWS 認証情報から AWS プロファイルの名前を保持します。または、このパラメータを削除してデフォルトのプロファイルを使用します。詳細については、「AWS CDK ドキュメント」を参照してください。	開発者、DevOps エンジニア

タスク	説明	必要なスキル
AWS CDK スタックをデプロイします。	このステップでは、必要なインフラストラクチャリソース (SQS キュー、S3 バケット、AWS Fargate タスク定義) を AWS アカウントで作成し、AWS Fargate タスクに必要な Docker イメージを構築して、アプリケーションをデプロイします。　コマンドプロンプトで、プロジェクトのルートディレクトリに移動し、次のコマンドを実行します。 `cdk deploy --profile "$AWS_PROFILE_NAME" --all` `$AWS_PROFILE_NAME` は、AWS 認証情報から AWS プロファイルの名前を保持します。または、このパラメータを削除してデフォルトのプロファイルを使用します。デプロイを確認します。CDK デプロイ出力の QueueUrl と バケット の値を書き留めておきます。これらは後のステップで必要になります。AWS CDK はアセットを作成し、AWS アカウントにアップロードして、すべてのインフラストラクチャリソースを作成します。リソースの作成プロセスは、AWS CloudForm	開発者、DevOps エンジニア

タスク	説明	必要なスキル
	ation コンソールで確認できます。詳細については、AWS CloudFormation のドキュメントと AWS CDK のドキュメント を参照してください。	

タスク	説明	必要なスキル
ビデオストリームを作成します。	このステップでは、ビデオ処理の入力ストリームとして機能する Kinesis ビデオストリームを作成します。AWS CLI がインストールされ、設定されていることを確認します。AWS CLI で、以下の手順を実行します。 `aws kinesisvideo --profile "$AWS_PROFILE_NAME" create-stream --stream-name "$STREAM_NAME" --data-retention-in-hours "24"` `$AWS_PROFILE_NAME` は AWS 認証情報の AWS 構成ファイルの名前を保存します (または、デフォルトの構成ファイルを使用するには、このパラメータを削除します)。`$STREAM_NAME` は任意の有効なストリーム名です。 代わりに、Kinesis コンソールを使用して「Kinesis ビデオストリームのドキュメント」に記載されている手順に従って Kinesis Video Streams を作成することもできます。作成したストリームの AWS リソースネーム (ARN) を書き留めて	開発者、DevOps エンジニア

タスク	説明	必要なスキル
	ください。後で必要になります。	

例の実行

タスク	説明	必要なスキル
動画をストリームにアップロードします。	サンプル `frame-splitter-code` アプリケーションのプロジェクトフォルダで、`src/test/java/amazon/awscdk/examples/splitter` フォルダにある `ProcessingTaskTest.java` ファイルを開きます。`profileName` および `streamName` 変数を、前のステップで使用した値に置き換えます。前のステップで作成した Kinesis ビデオストリームにサンプルビデオをアップロードするには、以下の手順を実行します。 `amazon.awscdk.examples.splitter.ProcessingTaskTest#testExample test` 代わりに、「Kinesis ビデオストリームのドキュメント」に記載されている方法のいずれかを使用して動画をアップロードすることもできます。	開発者、DevOps エンジニア

タスク	説明	必要なスキル

ビデオ処理を開始します。	これで、Kinesis ビデオストリームへのビデオのアップロードが完了したので、処理を開始できます。処理ロジックを開始するには、AWS CDK がデプロイ中に作成した SQS キューに詳細を含むメッセージを送信する必要があります。 AWS CLI を使用してメッセージを送信するには、以下の手順を実行してください。
	`aws sqs --profile "$AWS_PROFILE_NAME" send-message --queue-url QUEUE_URL --message-body MESSAGE`
	ここで、`$AWS_PROFILE_NAME` は AWS 認証情報から AWS 構成ファイルの名前を保存します (デフォルトのプロファイルを使用するには、このパラメータを削除します)。`QUEUE_URL` は AWS CDK 出力の QueueUrl 値、`MESSAGE` は次の形式の JSON 文字列です：
	`{ "streamARN": "STREAM_ARN", "bucket": "BUCKET_NAME", "s3Directory": "test-output" }`
		開発者、DevOps エンジニア

タスク	説明	必要なスキル
	ここで、STREAM_ARN は前の手順で作成したビデオストリームの ARN、BUCKET_NAME は AWS CDK 出力のバケット値です。 このメッセージを送信すると、動画処理が開始されます。または、「Amazon SQS ドキュメント」で説明されているように、Amazon SQS コンソールを使用してメッセージを送信することもできます。	
ビデオフレームの画像を表示します。	結果の画像は S3 出力バケット s3://BUCKET_NAME/test-output で確認できます。ここで、BUCKET_NAME は AWS CDK 出力からのバケット値です。	開発者、DevOps エンジニア

関連リソース

- AWS SDK ドキュメント
- 「AWS CDK API リファレンス」
- 「AWS CDK 入門ワークショップ」
- 「Amazon Kinesis Video Streams のドキュメント」
- 例: SageMaker を使用してビデオストリーム内のオブジェクトを識別する
- 例: Kinesis Video Streams フラグメントの解析およびレンダリング
- 「Amazon Kinesis Video Streams と Amazon SageMaker を使用してライブビデオをスケーリングし、リアルタイムで分析する」 (AWS 機械学習 ブログ記事)
- 「AWS Fargate のご利用開始にあたって」

追加情報

IDE の選択

お気に入りの Java IDE を使用してこのプロジェクトを構築し、探索することをお勧めします。

クリーンアップ

この例を実行した後、追加の AWS インフラストラクチャにコストが発生しないように、デプロイしたリソースをすべて削除してください。

インフラストラクチャとビデオストリームを削除するには、AWS CLI で次の 2 つのコマンドを使用します。

```
cdk destroy --profile "$AWS_PROFILE_NAME" --all
```

```
aws kinesisvideo --profile "$AWS_PROFILE_NAME" delete-stream --stream-arn "$STREAM_ARN"
```

代わりに、AWS CloudFormation コンソールを使用して AWS CloudFormation スタックを削除し、Kinesis コンソールを使用して Kinesis ビデオストリームを削除することで、リソースを手動で削除することもできます。ただし、`cdk destroy` は出力 S3 バケットや Amazon Elastic Container Registry (Amazon ECR) リポジトリ内のイメージを削除しないことに注意してください (`aws-cdk/assets`)。これらは手動で削除してください。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Glue を使用して Amazon S3 から Amazon Redshift にデータを段階的にロードする ETL サービスパイプラインを構築

ローハン・ジャマダーニ (AWS) とアルナバ・ダッタ (AWS) によって作成されました

概要

このパターンは、Amazon Simple Storage Service (Amazon S3) を設定してデータレイクのパフォーマンスを最適化し、AWS Glue を使用して差分データ変更を Amazon S3 から Amazon Redshift に読み込み、抽出、変換、ロード (ETL) オペレーションを実行する方法に関するガイダンスを提供します。

Amazon S3 のソースファイルには、カンマ区切り値 (CSV)、XML、JSON ファイルなど、さまざまな形式を使用できます。このパターンは、AWS Glue を使用してソースファイルを Apache Parquet のようなコストとパフォーマンスが最適化された形式に変換する方法を示しています。Amazon Athena と Amazon Redshift Spectrum からパーケットファイルを直接クエリできます。Parquet ファイルを Amazon Redshift に読み込んで集約し、集約したデータをコンシューマーと共有したり、Amazon QuickSight を使用してデータを視覚化したりすることもできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 適切な権限を持ち、CSV、XML、または JSON ファイルを含む S3 ソースバケット。

前提

- CSV、XML、または JSON ソースファイルはすでに Amazon S3 に読み込まれており、AWS Glue と Amazon Redshift が設定されているアカウントからアクセスできます。
- 「Amazon Redshift」のドキュメントで説明されているように、ファイルのロード、ファイルの分割、圧縮、マニフェストの使用に関するベストプラクティスに従います。
- ソースファイル構造は変更されません。
- ソースシステムは、Amazon S3 で定義されたフォルダ構造に従って Amazon S3 にデータを取り込むことができます。

- Amazon Redshift クラスターは 1 つのアベイラビリティーゾーンにまたがっています。(AWS Lambda、AWS Glue、Amazon Athena はサーバーレスであるため、このアーキテクチャは適切です)。高可用性を実現するため、クラスターのスナップショットは定期的に作成されます。

制約事項

- ファイル形式は、「現在 AWS Glue でサポートされている形式に限定されています」。
- リアルタイムのダウンストリームレポートはサポートされていません。

アーキテクチャ

ソーステクノロジースタック

- CSV、XML、または JSON ファイルを含む S3 バケット

ターゲットテクノロジースタック

- S3 データレイク (パーティション分割された Parquet ファイルストレージを使用)
- Amazon Redshift

ターゲットアーキテクチャ

データフロー

ツール

- 「Amazon S3」 – Amazon Simple Storage Service (Amazon S3) では、拡張性の高いオブジェクトストレージサービスです。Amazon S3 は、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- 「AWS Lambda」 – AWS Lambda を使用して、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。AWS Lambda はイベント駆動型サービスです。他の AWS サービスから自動的に開始するようにコードを設定できます。

- 「Amazon Redshift」 – Amazon Redshift はフルマネージド型で、ペタバイト規模のデータウェアハウスサービスです。Amazon Redshift では、標準 SQL を使用して、データウェアハウスとデータレイク全体でペタバイトの構造化データおよび半構造化データをクエリできます。
- 「AWS Glue」 — AWS Glue は、分析のためにデータを簡単に準備してロードできるフルマネージドのETLサービスです。AWS Glue がデータを検出し、関連するメタデータ (テーブル定義やスキーマなど) を AWS Glue データカタログに保存します。カタログ化されたデータは、すぐに検索およびクエリが可能になり、ETL で使用できるようになります。
- 「AWS Secrets Manager」 — AWS Secrets Manager は、アプリケーションやサービスへのアクセスに必要なシークレットの保護と一元管理を容易にします。このサービスでは、データベース認証情報、API キー、その他のシークレットが保存され、機密情報をプレーンテキスト形式でハードコーディングする必要がなくなります。Secrets Manager は、セキュリティとコンプライアンスのニーズを満たすためのキーローテーションも提供しています。Amazon Redshift、Amazon Relational Database Service (Amazon RDS)、および Amazon DocumentDB の統合が組み込まれています。Secrets Manager コンソール、コマンドラインインターフェイス (CLI)、または Secrets Manager API と SDK を使用して、シークレットを保存して一元管理できます。
- 「Amazon Athena」 – Amazon Athena は、Amazon S3 に保存されたデータを簡単に分析できるインタラクティブなクエリサービスです。Athena はサーバーレスで AWS Glue と統合されているため、AWS Glue を使用してカタログ化されたデータを直接クエリできます。Athena は、インタラクティブなクエリパフォーマンスを実現するように伸縮自在にスケールされています。

エピック

S3 バケットとフォルダ構造を作成します。

タスク	説明	必要なスキル
ソースシステムのデータ構造と属性を分析する。	Amazon S3 データレイクに寄与する各データソースに対してこのタスクを実行します。	データエンジニア
パーティションとアクセス戦略を定義します。	この戦略は、データキャプチャの頻度、差分処理、消費ニーズに基づいて策定する必要があります。S3 バケットが一般に公開されていないこと、アクセスが特定のサービ	データエンジニア

タスク	説明	必要なスキル
	スロールベースのポリシーのみによって制御されていることを確認してください。詳細については、Amazon S3 のドキュメント を参照してください。	
データソースタイプごとに個別の S3 バケットを作成し、処理された (Parquet) データ用にソースごとに個別の S3 バケットを作成します。	ソースごとに個別のバケットを作成し、ソースシステムのデータインジェスト頻度に基づいたフォルダ構造を作成します (例:`s3://source-system-name/date/hour`)。処理された (Parquet 形式に変換された) ファイルについては、同様の構造 (例:`s3://source-processed-bucket/date/hour`) を作成します。S3 バケットの作成について詳しくは、「Amazon S3 ドキュメント」 を参照してください。	データエンジニア

Amazon Redshift でデータウェアハウスを作成

タスク	説明	必要なスキル
適切なパラメータグループとメンテナンスおよびバックアップ戦略を使用して Amazon Redshift クラスターを起動します。	Amazon Redshift クラスターを作成するときに、Secrets Manager のデータベースシークレットを管理者ユーザーの認証情報に使用します。Amazon Redshift クラスターの作成とサイジングについては、	データエンジニア

タスク	説明	必要なスキル
	「Amazon Redshift ドキュメント」とクラウドデータウェアハウスの「サイジング」ホワイトペーパーを参照してください。	
IAM サービスロールを作成して Amazon Redshift クラスターにアタッチします。	AWS Identity and Access Management (IAM) サービスロールは、Secrets Manager とソース S3 バケットへのアクセスを保証します。詳細については、「承認」と「ロールの追加」に関するAWS ドキュメントを参照してください。	データエンジニア
データベーススキーマを作成します。	テーブル設計の Amazon Redshift ベストプラクティスユースケースに基づいて、適切なソートキーと分散キー、および可能な限り最適な圧縮エンコーディングを選択します。ベストプラクティスについては、「AWS のドキュメンテーション」を参照ください。	データエンジニア

タスク	説明	必要なスキル
ワークロード管理の設定	要件に応じて、ワークロード管理 (WLM) キュー、ショートクエリアクセラレーション (SQA)、または同時実行スケーリングを設定します。詳細については、Amazon Redshift ドキュメントの「ワークロード管理の実装」を参照してください。	データエンジニア

Secrets Manager でのシークレットの作成

タスク	説明	必要なスキル
Amazon Redshift サインイン認証情報をSecrets Managerに保存するための新しいシークレットを作成します。	このシークレットには、管理者ユーザーだけでなく、個々のデータベースサービスユーザーの認証情報も保存されます。手順については、「Secrets Manager のドキュメント」を参照してください。シークレットタイプとして Amazon Redshift クラスターを選択します。さらに、シークレットローテーションページでローテーションを有効にします。これにより、Amazon Redshift クラスターに適切なユーザーが作成され、定義された間隔でキーシークレットがローテーションされます。	データエンジニア

タスク	説明	必要なスキル
Secrets Manager へのアクセスを制限する IAM ポリシーを作成します。	Secrets Manager へのアクセスを Amazon Redshift 管理者と AWS Glue のみに制限します。	データエンジニア

AWS Glue の設定

タスク	説明	必要なスキル
AWS Glue データカタログで、Amazon Redshift への接続を追加します。	手順については、「AWS Glue のドキュメント」を参照してください。	データエンジニア
Secrets Manager、Amazon Redshift、S3 バケットにアクセスするための AWS Glue の IAM サービスロールを作成してアタッチします。	詳細については、「AWS Glue のドキュメント」を参照してください。	データエンジニア
ソースの AWS Glue データカタログを定義します。	このステップでは、AWS Glue データカタログにデータベースと必要なテーブルを作成します。クローラーを使用して AWS Glue データベースのテーブルをカタログ化することも、Amazon Athena 外部テーブルとして定義することもできます。AWS Glue データカタログから Athena で定義されている外部テーブルにアクセスすることもできます。Athena での「データカタログの定義」と「外部テーブルの作成の詳細」について	データエンジニア

タスク	説明	必要なスキル
	は、AWS ドキュメントを参照してください。	
ソースデータを処理する AWS Glue ジョブを作成します。	AWS Glue ジョブは Python シェルでも PySpark でもかまいません。これにより、ソースデータファイルを標準化、重複排除、およびクレンジングできます。パフォーマンスを最適化し、S3 ソースバケット全体をクエリする必要がないようにするには、AWS Glue ジョブのプッシュダウン述語として S3 バケットを日付別にパーティション化し、年、月、日、時間ごとに分割します。詳細については、「AWS Glue のドキュメント」を参照してください。処理され変換されたデータを、処理済みの S3 バケットパーティションに Parquet 形式でロードします。Athena からパーケットファイルをクエリできます。	データエンジニア
Amazon Redshift にデータをロードする AWS Glue ジョブを作成します。	AWS Glue ジョブは Python シェルでも、データを更新して完全に更新することでデータをロードする PySpark でもかまいません。詳細については、「AWS Glue のドキュメント」と追加情報セクションを参照してください。	データエンジニア

タスク	説明	必要なスキル
(オプション) 必要に応じてトリガーを使用して AWS Glue ジョブをスケジュールします。	増分データロードは主に、AWS Lambda 関数が AWS Glue ジョブを呼び出す Amazon S3 イベントによって駆動されます。イベントベースのスケジューリングではなく、時間ベースを必要とするデータロードには、AWS Glue のトリガーベースのスケジューリングを使用してください。	データエンジニア

「Lambda 関数の作成」

タスク	説明	必要なスキル
S3 バケットと AWS Glue ジョブにアクセスするための AWS Lambda 用の IAM サービスにリンクされたロールを作成してアタッチします。	Amazon S3 オブジェクトとバケットを読み取るポリシーと、AWS Glue API にアクセスして AWS Glue ジョブを開始するポリシーを含む AWS Lambda 用の IAM サービスにリンクされたロールを作成します。詳細については、「ナレッジセンター」 を参照してください。	データエンジニア
定義された Amazon S3 イベントに基づいて AWS Glue ジョブを実行する Lambda 関数を作成します。	Lambda 関数は Amazon S3 マニフェストファイルの作成によって開始する必要があります。Lambda 関数は Amazon S3 フォルダの場所 (たとえば、ソース_バケット/年/月/日付/時間) をパラメータとして	データエンジニア

タスク	説明	必要なスキル
	AWS Glue ジョブに渡す必要があります。AWS Glue ジョブは、このパラメータをプッシュダウン述語として使用して、ファイルアクセスとジョブ処理パフォーマンスを最適化します。詳細については、「AWS Glue のドキュメント」を参照してください。	
Amazon S3 PUT オブジェクトイベントを作成してオブジェクトの作成を検出し、それぞれの Lambda 関数を呼び出します。	Amazon S3 PUT オブジェクトイベントは、マニフェストファイルの作成によってのみ開始する必要があります。マニフェストファイルは Lambda 関数と AWS Glue ジョブの同時実行を制御し、S3 ソースバケットの特定のパーティションに到着する個々のファイルを処理する代わりに、ロードをバッチとして処理します。詳細については、「Lambda ドキュメント」を参照してください。	データエンジニア

関連リソース

- 「Amazon S3 ドキュメント」
- 「AWS Glue ドキュメンテーション」
- 「Amazon Redshift ドキュメンテーション」
- AWS Lambda
- Amazon Athena
- AWS Secrets Manager

追加情報

アップサートと完全更新の詳細なアプローチ

アップサート:ビジネスユースケースによっては、履歴の集計が必要なデータセットを対象としています。ビジネスニーズに基づいて、「新しいデータの更新と挿入」(Amazon Redshift ドキュメント)で説明されているアプローチのいずれかに従ってください。

完全更新:これは、履歴集計を必要としない小規模なデータセット向けです。以下のいずれかの方法に従ってください。

1. Amazon Redshift テーブルを切り捨てます。
2. ステージングエリアから現在のパーティションをロードします。

または

1. 現在のパーティションデータを含む一時テーブルを作成します。
2. ターゲットのAmazon Redshift テーブルをドロップします。
3. テンポラリテーブルの名前をターゲットテーブルに変更します。

Amazon DataZone を使用してエンタープライズデータメッシュを構築する AWS CDKAWS CloudFormation

作成者: Dhrubajyoti Mukherjee (AWS)、Adjoa Taylor (AWS)、Ravi Kumar (AWS)、Weizhou Sun (AWS)

概要

アマゾン ウェブ サービス (AWS) では、お客様は、イノベーションを加速し、企業のビジネス価値を促進するための鍵がデータであることを理解しています。この大量のデータを管理するには、データメッシュなどの分散アーキテクチャを採用できます。データメッシュアーキテクチャは、顧客、目標、市場を考慮する考え方である製品思考を容易にします。データメッシュは、データへの高速で安全なアクセスを提供するフェデレーティッドガバナンスモデルを確立するのにも役立ちます。

でデータメッシュベースのエンタープライズソリューションを構築するための戦略 AWSでは、Data Mesh Strategy Framework を使用して組織のデータメッシュ戦略を策定および実装する方法について説明します。Data Mesh Strategy Framework を使用すると、チームの編成とそのやり取りを最適化して、データメッシュジャーニーを加速できます。

このドキュメントでは、Amazon DataZone でエンタープライズデータメッシュを構築する方法に関するガイダンスを提供します。Amazon DataZone は、 、オンプレミス、およびサードパーティーのソースに保存されたデータをカタログ化、検出 AWS、共有、管理するためのデータ管理サービスです。このパターンには、 AWS Cloud Development Kit (AWS CDK) と を使用してデータメッシュベースのデータソリューションインフラストラクチャをデプロイするのに役立つコードアーティファクトが含まれています AWS CloudFormation。このパターンは、クラウドアーキテクトと DevOps エンジニアを対象としています。

このパターンの目的とソリューションの範囲については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- 少なくとも 2 つのアクティブ AWS アカウント: 1 つは中央ガバナンスアカウント用、もう 1 つはメンバーアカウント用
- AWS 開発環境の中央ガバナンスアカウントの管理者認証情報

- AWS Command Line Interface コマンドライン AWS のサービス から を管理するためにインストールされた (AWS CLI)
- AWS CDK アプリケーションを管理するために Node.js と Node Package Manager (npm) がインストールされている
- AWS CDK npm を使用して開発環境にグローバルにインストールされたツールキットで、 AWS CDK アプリケーションを合成してデプロイする

```
npm install -g aws-cdk
```

- 開発環境にインストールされている Python バージョン 3.12
- TypeScript は、開発環境にインストールされるか、npm コンパイラを使用してグローバルにインストールされます。

```
npm install -g typescript
```

- 開発環境にインストールされた Docker
- ソリューションのソースコードを維持するための Git などのバージョン管理システム (推奨)
- Python と TypeScript をサポートする統合開発環境 (IDE) またはテキストエディタ (強く推奨)

制約事項

- このソリューションは、Linux または macOS を実行しているマシンでのみテストされています。
- 現在のバージョンでは、ソリューションは AWS IAM Identity Center デフォルトで Amazon DataZone と の統合をサポートしていません。ただし、この統合をサポートするように設定できます。

製品バージョン

- Python バージョン 3.12

アーキテクチャ

次の図は、データメッシュリファレンスアーキテクチャを示しています。このアーキテクチャは Amazon DataZone に基づいており、Amazon Simple Storage Service (Amazon S3) と をデータソース AWS Glue Data Catalog として使用します。データメッシュの実装で Amazon DataZone AWS のサービス で使用する は、組織の要件によって異なる場合があります。

1. プロデューサーアカウントでは、raw データは現在の形式で消費に適しているか、 を使用して消費するために変換されます AWS Glue。データの技術メタデータは Amazon S3 に保存され、AWS Glue データクローラーを使用して評価されます。データ品質はAWS Glue 、Data Quality を使用して測定されます。Data Catalog のソースデータベースは、Amazon DataZone カタログにアセットとして登録されます。Amazon DataZone カタログは、Amazon DataZone データソースジョブを使用して中央ガバナンスアカウントでホストされます。
2. 中央ガバナンスアカウントは、Amazon DataZone ドメインと Amazon DataZone データポータルをホストします。データプロデューサーとコンシューマー AWS アカウント の は、Amazon DataZone ドメインに関連付けられています。データプロデューサーとコンシューマーの Amazon DataZone プロジェクトは、対応する Amazon DataZone ドメイン単位で編成されます。
3. データアセットのエンドユーザーは、 AWS Identity and Access Management (IAM) 認証情報またはシングルサインオン (IAM Identity Center を介した統合) を使用して Amazon DataZone データポータルにログインします。Amazon DataZone データカタログ内のアセット情報 (データ品質情報、ビジネスメタデータ、技術メタデータなど) を検索、フィルタリング、表示します。
4. エンドユーザーが目的のデータアセットを見つけたら、Amazon DataZone サブスクリプション機能を使用してアクセスをリクエストします。プロデューサーチームのデータ所有者は通知を受け取り、Amazon DataZone データポータルでサブスクリプションリクエストを評価します。データ所有者は、その有効性に基づいてサブスクリプションリクエストを承認または拒否します。
5. サブスクリプションリクエストが付与および受理されると、アセットはコンシューマーアカウントで以下のアクティビティのためにアクセスされます。
   - Amazon SageMaker AI を使用した AI/ML モデル開発
   - Amazon Athena と Amazon QuickSight を使用した分析とレポート

ツール

AWS のサービス

- Amazon Athenaは、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析できるようにするインタラクティブなクエリサービスです。
- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。

- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体にわたるリソースの管理に役立ちます AWS リージョン。
- Amazon DataZone は、、オンプレミス、およびサードパーティーソースに保存されたデータのカタログ化、検出 AWS、共有、管理を支援するデータ管理サービスです。
- Amazon QuickSight は、視覚化、分析、レポート生成に使用できるクラウドスケールのビジネスインテリジェンス (BI) サービスです。
- Amazon SageMaker AI は、ML モデルを構築してトレーニングし、本番環境に対応したホスト環境にデプロイするのに役立つマネージド機械学習 (ML) サービスです。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「Amazon Simple Queue Service (Amazon SQS) 」は、安全で耐久性があり、配信ソフトウェアシステムとコンポーネントを統合および分離できる利用可能なホスト型キューを提供します。
- 「Amazon Simple Storage Service (Amazon S3) 」は、どのようなデータの量であっても、保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このソリューションは、GitHub data-mesh-datazone-cdk-cloudformation リポジトリで利用できます。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリのクローンを作成するには、ローカル開発環境 (Linux または macOS) で次のコマンドを実行します。 `git clone https://github.com/aws-samples/data-mesh-data`	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	`zone-cdk-cloudform ation`	

環境を作成します。	Python 仮想環境を作成するには、次のコマンドを実行します。
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	`pip install -r requirements.txt`
		クラウドアーキテクト、DevOps エンジニア

アカウントをブートストラップします。	を使用して中央ガバナンスアカウントをブートストラップするには AWS CDK、次のコマンドを実行します。 `cdk bootstrap aws://<GOVERNANCE_ACCOUNT_ID>/<AWS_REGION>` にサインインし AWS Management Console、中央ガバナンスアカウントコンソールを開き、 AWS CDK 実行ロールの Amazon リソースネーム (ARN) を取得します。	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
DzDataMeshMemberStackSet.yaml ファイルを作成します。	DzDataMeshMemberStackSet.yaml ファイルを作成するには、リポジトリのルートディレクトリから次の bash スクリプトを開始します。 `./lib/scripts/create_dz_data_mesh_member_stack_set.sh`	クラウドアーキテクト、DevOps エンジニア
テンプレートの作成を確認します。	AWS CloudFormation テンプレートファイルが のlib/cfn-templates/DzDataMeshMemberStackSet.yaml 場所に作成されていることを確認します。	クラウドアーキテクト、DevOps エンジニア

中央ガバナンスアカウントにリソースをデプロイする

タスク	説明	必要なスキル

設定を変更します。	config/Config.ts ファイルで、次のパラメータを変更します。
	`DZ_APPLICATION_NAME - Name of the application.`
	`DZ_STAGE_NAME - Name of the stage.`
	`DZ_DOMAIN_NAME - Name of the Amazon DataZone domain`
		クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	DZ_DOMAIN_DESCRIPTIO N - Description of the Amazon DataZone domain DZ_DOMAIN_TAG - Tag of the Amazon DataZone domain DZ_ADMIN_PROJECT_NAM E - Name of the Amazon DataZone project for administrators DZ_ADMIN_PROJECT_DESC RIPTION - Description of the Amazon DataZone project for administr ators CDK_EXEC_ROLE_ARN - ARN of the cdk execution role DZ_ADMIN_ROLE_ARN - ARN of the administr ator role 残りのパラメータは空のままにします。	

タスク	説明	必要なスキル
Amazon DataZone 用語集の設定を更新します。	`lib/utils/glossary_config.json` ファイルで Amazon DataZone 用語集設定を更新するには、次の設定例を使用します。 `{ "GlossaryName": "PII Data", "GlossaryDescription": "If data source contains PII attributes", "GlossaryTerms": [{ "Name": "Yes", "ShortDescription": "Yes", "LongDescription": "Yes Glossary Term" }, { "Name": "No", "ShortDescription": "No", "LongDescription": "No Glossary Term" } ] }`	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
Amazon DataZone メタデータフォームの設定を更新します。	で Amazon DataZone メタデータフォーム設定を更新するには`lib/utils/metadata_form_config.json file` 、次の設定例を使用します。 `{ "FormName": "ScheduleDataRefresh", "FormDescription": "Form for data refresh schedule", "FormSmithyModel": "@amazon.datazone#displayname(defaultName: \"Data Refresh Schedule\")\nstructure ScheduleDataRefresh {\n @documentation(\"Schedule of Data Refresh\")\n @required\n @amazon.datazone#searchable\n @amazon.datazone#displayname(defaultName: \"Data Refresh Schedule\")\n data_refresh_schedule: String\n}" }`	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
AWS 認証情報をエクスポートします。	管理者権限を持つ IAM ロールの開発環境に AWS 認証情報をエクスポートするには、次の形式を使用します。 `export AWS_ACCES S_KEY_ID= export AWS_SECRE T_ACCESS_KEY= export AWS_SESSI ON_TOKEN=`	クラウドアーキテクト、DevOps エンジニア
テンプレートを合成します。	AWS CloudFormation テンプレートを合成するには、次のコマンドを実行します。 `npx cdk synth`	クラウドアーキテクト、DevOps エンジニア
ソリューションをデプロイします。	ソリューションをデプロイするには、次のコマンドを実行します。 `npx cdk deploy --all`	クラウドアーキテクト、DevOps エンジニア

新しいメンバーアカウントを設定する

タスク	説明	必要なスキル
テンプレートをデプロイします。	次の入力パラメータを使用して、メンバーアカウントの `lib/cfn-templates/ DzDataMeshCfnStack SetExecutionRole.y aml` にある AWS CloudForm	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	ation テンプレートをデプロイします。 • `GovernanceAccountID` – ガバナンスアカウントのアカウント ID • `DataZoneKMSKeyID` – Amazon DataZone メタデータを暗号化する AWS Key Management Service (AWS KMS) キーの ID • `NotificationQueueName` – ガバナンスアカウントの Amazon SQS 通知キューの名前	
ARNs を更新します。	メンバーアカウントの AWS CloudFormation StackSet 実行ロール ARNs のリストを更新するには、次のコードを使用します。 `DZ_MEMBER_STACK_SET_EXEC_ROLE_LIST - List of Stack set execution role arns for the member accounts.`	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル

合成してデプロイします。	AWS CloudFormation テンプレートを合成してソリューションをデプロイするには、次のコマンドを実行します。
	
	`npx cdk synth`
	`npx cdk deploy --all`
		クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
メンバーアカウントを関連付けます。	メンバーアカウントを中央ガバナンスアカウントに関連付けるには、次の手順を実行します。 1. 中央ガバナンスアカウントの コンソールにサインインし、「https://https://console.aws.amazon.com/datazone/.com で Amazon DataZone コンソールを開きます。 2. 作成したドメインを選択します。 3. 関連アカウントタブまでスクロールし、関連付けのリクエストを選択します。 4. AWS アカウント ID を指定し、RAM ポリシーとして AWSRAMPermissionDataZonePortalReadWrite を選択します。 5. リクエストの関連付けを選択します。 6. アカウントが正常にブートストラップされたという E メール通知を受信するまで待ちます。	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
パラメータを更新します。	の設定ファイルのメンバーアカウント固有のパラメータを更新するにはconfig/Config.ts 、次の形式を使用します。 `export const DZ_MEMBER_ACCOUNT_CONFIG: memberAccountConfig = { '123456789012' : { PROJECT_NAME: 'TEST-PROJECT-123456789012', PROJECT_DESCRIPTION: 'TEST-PROJECT-123456789012', PROJECT_EMAIL: 'user@xyz.com' } }`	クラウドアーキテクト、DevOps エンジニア

テンプレートを合成してデプロイします。	AWS CloudFormation テンプレートを合成してソリューションをデプロイするには、次のコマンドを実行します。
	
	`npx cdk synth`
	`npx cdk deploy --all`
		クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
メンバーアカウントを追加します。	データソリューションで追加のメンバーアカウントを作成して設定するには、各メンバーアカウントに対して前のステップを繰り返します。 このソリューションは、データプロデューサーとコンシューマーを区別しません。	クラウドアーキテクト、DevOps エンジニア

クリーンアップ

タスク	説明	必要なスキル
メンバーアカウントの関連付けを解除します。	アカウントの関連付けを解除するには、次の手順を実行します。 1. コンソールにサインインし、Amazon DataZone コンソールを開きます。 2. ドメインの表示 を選択します。 3. 作成したドメインを選択します。 4. アカウントの関連付けタブを選択します。 5. 関連付けを解除するメンバーアカウントを選択します。 6. 関連付け解除を選択し、`disassociate` を入力して確認します。	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	7. すべてのメンバーアカウントに対してステップ 3～6 を繰り返します。	

タスク	説明	必要なスキル
スタックインスタンスを削除します。	AWS CloudFormation スタックインスタンスを削除するには、次の手順を実行します。 1. AWS CloudFormation コンソールを https://console.aws.amazon.com/cloudformation/.com で開きます。 2. ナビゲーションペインから [StackSets] を選択します。 3. StackSet-DataZone-DataMesh-Member という名前のスタックセットを選択し、スタックインスタンスタブを選択します。 4. メンバーシップから削除するメンバーアカウントの AWS アカウント ID をコピーします。 5. アクションを選択し、StackSetからスタックを削除を選択し、デフォルトのオプションを維持します。 6. アカウント番号 フィールドに、アカウント ID を入力します。 7. リージョンの指定ドロップダウンリストで、 を選択します AWS リージョン。 8. Next を選択し、Submit を選択します。	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	9. オペレーションタブで、オペレーションが成功したことを確認します。スタックの削除には時間がかかる場合があります。 10.すべてのメンバーアカウントに対してステップ 2～9 を繰り返します。	
すべてのリソースを破棄します。	リソースを破棄するには、ローカル開発環境 (Linux または macOS) に次の手順を実装します。 1. リポジトリのルートディレクトリに移動します。 2. AWS CDK スタックの作成に使用した IAM ロールの AWS 認証情報をエクスポートします。 3. クラウドリソースを破棄するには、次のコマンドを実行します。 `npx cdk destroy --all`	クラウドアーキテクト、DevOps エンジニア

関連リソース

- AWS Glue データを使用した Amazon DataZone クイックスタート
- チュートリアル: 最初の AWS CDK アプリを作成する
- の開始方法 AWS CloudFormation
- でデータメッシュベースのエンタープライズソリューションを構築するための戦略 AWS

追加情報

目的

このパターンを実装すると、次のことが可能になります。

- データの所有権の分散 - データの所有権を中央チームから、組織のソースシステム、ビジネスユニット、またはユースケースを表すチームに移行します。
- 製品思考 - 組織内のデータアセットを検討するときに、顧客、市場、その他の要因を含む製品ベースの考え方を導入します。
- フェデレーティッドガバナンス - 組織のデータ製品全体のセキュリティガードレール、コントロール、コンプライアンスを改善します。
- マルチアカウントおよびマルチプロジェクトサポート – 組織のビジネスユニットまたはプロジェクト全体で、効率的で安全なデータ共有とコラボレーションをサポートします。
- 一元的なモニタリングと通知 - Amazon CloudWatch を使用してデータメッシュのクラウドリソースをモニタリングし、新しいメンバーアカウントが関連付けられたときにユーザーに通知します。
- スケーラビリティと拡張性 - 組織の進化に合わせて、データメッシュに新しいユースケースを追加します。

ソリューションスコープ

このソリューションを使用すると、小規模から始めて、データメッシュジャーニーの進行に合わせてスケールできます。多くの場合、メンバーアカウントがデータソリューションを採用すると、組織、プロジェクト、またはビジネスユニットに固有のアカウント設定が含まれます。このソリューションは、以下の機能をサポートすることで、これらの多様な AWS アカウント 設定に対応します。

- Amazon DataZone のデータソースとしての AWS Glue データカタログ DataZone
- Amazon DataZone データドメインと関連するデータポータルの管理
- データメッシュベースのデータソリューションにメンバーアカウントを追加する管理
- Amazon DataZone プロジェクトと環境の管理
- Amazon DataZone 用語集とメタデータフォームの管理
- データメッシュベースのデータソリューションユーザーに対応する IAM ロールの管理
- データメッシュベースのデータソリューションユーザーの通知
- プロビジョニングされたクラウドインフラストラクチャのモニタリング

このソリューションでは、 AWS CDK と AWS CloudFormation を使用してクラウドインフラストラクチャをデプロイします。を使用して以下 AWS CloudFormation を実行します。

- クラウドリソースをより低い抽象化レベルで定義してデプロイします。
- からクラウドリソースをデプロイします AWS Management Console。このアプローチを使用すると、開発環境なしでインフラストラクチャをデプロイできます。

データメッシュソリューションは、 AWS CDK を使用して、より高い抽象化レベルでリソースを定義します。その結果、このソリューションは、クラウドリソースをデプロイする関連ツールを選択することで、分離されたモジュール式のスケーラブルなアプローチを提供します。

次のステップ

Amazon DataZone でデータメッシュを構築する際のガイダンスについては、 AWSエキスパートにお問い合わせください。

このソリューションのモジュラー性は、データファブリックやデータレイクなど、さまざまなアーキテクチャでデータ管理ソリューションを構築することをサポートします。さらに、組織の要件に基づいて、ソリューションを他の Amazon DataZone データソースに拡張できます。

AWS のサービスを使用してバリューアットリスク (VaR) を計算

作成者: Sumon Samanta (AWS)

概要

このパターンは、AWS のサービスを使用してバリューアットリスク (VaR) 計算システムを実装する方法を示します。オンプレミス環境では、ほとんどの VaR システムが大規模な専用インフラストラクチャ、および社内または商用のグリッドスケジューリングソフトウェアを使用してバッチ処理を実行します。このパターンは、AWS クラウドで VaR 処理を行うための、シンプルで信頼性が高く、スケーラブルなアーキテクチャを提示します。ストリーミングサービスとして Amazon Kinesis Data Streams、マネージドキューサービスとして Amazon Simple Queue Service (Amazon SQS)、マネージドキューサービスとして Amazon ElastiCache、注文の処理とリスク計算に AWS Lambda を使用するサーバーレスアーキテクチャを構築しています。

VaR は統計的尺度であり、トレーダーやリスクマネージャーが、一定の信頼水準を超えるとポートフォリオの潜在的な損失を推定するために使用します。ほとんどの VaR システムでは、数学計算や統計計算を多数実行し、その結果を保存しています。これらの計算には大量の計算リソースが必要となるため、VaR バッチ処理はより小さな計算タスクに分割する必要があります。大きなバッチを小さなタスクに分割することは可能です。これらのタスクはほとんど独立している (つまり、あるタスクの計算が他のタスクには依存しない) ためです。

VaR アーキテクチャのもう1つの重要な要件は、計算のスケーラビリティです。このパターンでは、計算負荷に基づいて自動的にスケールインまたはスケールアウトするサーバーレスアーキテクチャを使用しています。バッチ処理やオンライン計算の需要を予測することは難しいため、サービスレベルアグリーメント (SLA) で定められたスケジュール内でプロセスを完了するには動的なスケーリングが必要です。また、コストが最適化されたアーキテクチャでは各コンピュートリソース上のタスクが完了したら、すぐにそのリソースをスケールダウンできるはずです。

AWS のサービスは、スケーラブルな計算能力とストレージ容量、コストを最適化した方法で処理できる分析サービス、リスク管理ワークフローを実行するさまざまなタイプのスケジューラーを備えているため、VaR 計算に最適です。また、AWS で使用したコンピューティングとストレージリソースに対してのみ料金を支払います。

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- 入力ファイルは、ビジネス要件によって異なります。一般的な使用例には以下の入力ファイルが含まれます：
  - マーケットデータファイル (VaR 計算エンジンへの入力)
  - 取引データファイル (取引データがストリーム経由で送られる場合を除く)。
  - 構成データファイル (モデルとその他の静的構成データ)
  - 計算エンジンモデルファイル (定量ライブラリー)
  - 時系列データファイル (過去 5 年間の株価などの履歴データ用)
- マーケットデータやその他の入力がストリームを通じて受信される場合、Amazon Kinesis Data Streams がセットアップし、ストリームに書き込むように Amazon 識別とアクセス管理 (IAM) の権限が設定されます。

このパターンでは、取引データを取引システムから Kinesis データストリームに書き込むアーキテクチャを構築します。ストリーミングサービスを使用する代わりに、取引データをスモールバッチファイルに保存できます。それらは Amazon Simple Storage Service (Amazon S3) バケットに保管し、そしてイベントを呼び出してデータの処理を開始します。

制約事項

- Kinesis データストリームの順序付けは各シャードで保証されています。そのため複数のシャードに書き込まれた取引注文が、書き込み操作と同じ順序で配信されることは保証されません。
- AWS Lambda ランタイムの制限は、現在は 15 分です。詳細については、「Lambda FAQ」を参照してください。

アーキテクチャ

ターゲットアーキテクチャ

次のアーキテクチャ図は、リスク評価システムの AWS サービスとワークフローを示しています。

この図表は、以下を示すものです：

1. 取引は注文管理システムからストリームインされます。
2. チケットポジションネッティング Lambda 関数は、注文を処理し、各ティッカーの統合メッセージを Amazon SQS のリスクキューに書き込みます。

3. リスク計算エンジン の Lambda 関数は、Amazon SQS からのメッセージを処理し、リスク計算を実行し、Amazon ElastiCache のリスクキャッシュにある VaR 損益 (PnL) 情報を更新します。
4. ElastiCache データの読み取りのLambda 関数は、ElastiCache からリスク結果を取得し、データベースと S3 バケットに保管します。

これらの手順の詳細については、エピックセクションを参照してください。

自動化とスケール

AWS Cloud Development Kit (AWS CDK) または AWS CloudFormation テンプレートを使用して、アーキテクチャ全体をデプロイできます。このアーキテクチャは、バッチ処理と日中 (リアルタイム) 処理の両方のサポートができます。

このアーキテクチャにはスケーリングがビルドインされています。Kinesis データストリームに書き込まれて処理を待つ取引が増えたら、追加の Lambda 関数を呼び出してそれらの取引を処理し、処理が完了したらスケールダウンができます。複数の Amazon SQS リスク計算キューによる処理もオプションです。キュー全体で厳密な順序付けや統合が必要な場合、処理を並列化することはできません。ただし、一日の終わりのバッチや日中のミニバッチの場合、Lambda 関数が並行で処理され、最終結果を ElastiCache に保存できます。

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。このパターンでは、例として MySQL を使用していますが、いずれかの RDBMS システムを使用してデータを保存できます。
- 「Amazon ElastiCache」 は、AWS クラウド上でのインメモリ分散キャッシュ環境のセットアップ、管理、スケーリングに役立ちます。
- 「Amazon Kinesis Data Streams」は、データレコードの大量のストリームをリアルタイムで収集し、処理するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Queue Service (Amazon SQS) 」は、安全で耐久性があり、配信ソフトウェアシステムとコンポーネントを統合および分離できる利用可能なホスト型キューを提供します。

- 「Amazon Simple Storage Service (Amazon S3) 」は、どのようなデータの量であっても、保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

コード

このパターンは、AWS クラウドの VaR システムのアーキテクチャの例を示し、Lambda 関数を VaR 計算に使用する方法を示しています。Lambda 関数を作成するには、「Lambdaドキュメント」のコード例を参照してください。サポートが必要な場合、「AWS プロフェッショナルサービス」 にお問い合わせください。

ベストプラクティス

- 各 VaR 計算タスクはできるだけ小さく、軽量にして保持します。各コンピュートタスクでさまざまなトレード数を試して、どれが計算時間とコストについて最適化されているかを確認します。
- 再利用可能なオブジェクトを Amazon ElastiCache に保存します。Apache Arrow などのフレームワークを使用して、シリアル化と逆シリアル化を減らすことができます。
- Lambda の時間制限を考慮します。計算タスクが 15 分を超える可能性があると思われる場合、Lambda タイムアウトを回避するためにタスクをより小さなタスクに分割してみてください。これが不可能な場合、AWS Fargate、Amazon Elastic Container Service (Amazon ECS)、および Amazon Elastic Kubernetes Service (Amazon EKS) を使用したコンテナオーケストレーションソリューションを検討します。

エピック

リスクシステムへのトレードフロー

タスク	説明	必要なスキル
トレードの書き込みを開始します。	新規取引、決済取引、または一部決済取引は、注文管理システムからリスクストリームに書き込まれます。このパターンでは、マネージドストリーミングサービスとして Amazon Kinesis を使用します。取引注文ティッカーのハ	Amazon Kinesis

タスク	説明	必要なスキル
	ッシュは、複数のシャードに取引注文を出すために使用されます。	

注文処理のためのLambda 関数を実行

タスク	説明	必要なスキル
Lambda でリスク処理を開始します。	新しい注文に対して AWS Lambda 関数を実行します。保留中の取引注文の数に基づいて、Lambda は自動的にスケーリングします。各 Lambda インスタンスには 1 つ以上の注文があり、Amazon ElastiCache から各ティッカーの最新の位置を取得します。（ElasticCache からデータを保存および取得するためのキーとして、CUSIP ID、カーブ名、または他の金融デリバティブ商品のインデックス名が使用できます）。ElastiCache では、合計ポジション (数量) とキーと値のペア <ティッカー、ネットポジション > ここでネットポジションがスケーリングファクターですが、ティッカーごとに 1 回更新されます。	Amazon Kinesis、AWS Lambda、Amazon ElastiCache

各ティッカーのメッセージをキューに書き込む

タスク	説明	必要なスキル
統合メッセージをリスクキューに書き込みます。	キューへメッセージを送信します。このパターンでは、Amazon SQS をマネージドキューサービスとして使用します。1 つの Lambda インスタンスでいつでも取引注文のミニバッチを取得できますが、Amazon SQS にはティッカーごとに 1 つのメッセージしか書き込まれません。スケーリングファクターは (古いネットポジション + 現在のポジション) / 古いネットポジション で計算されます。	Amazon SQS, AWS Lambda

リスクエンジンを呼び出す

タスク	説明	必要なスキル
リスク計算を開始します。	リスクエンジンラムダの Lambda 関数が呼び出されます。各ポジションは 1 つの Lambda 関数によって処理されます。ただし、最適化のため、各 Lambda 関数は Amazon SQS からの複数のメッセージを処理できます。	Amazon SQS, AWS Lambda

リスク結果をキャッシュから取得

タスク	説明	必要なスキル
リスクキャッシュを取得して更新します。	Lambda は、各ティッカーの現在のネットポジションを ElastiCache から取得します。また、各ティッカーの Var 利益と損失 (PnL) 配列を ElastiCache から取得します。 PnL 配列がすでに存在する場合、Lambda 関数は配列と Var をスケールで更新します。スケールは、ネッティング Lambda 関数によって書き込まれた Amazon SQS メッセージから取得されます。PnL 配列が ElasticCache にない場合、シミュレートされたティッカー価格シリーズデータを使用して新しい PnL と Var が計算されます。	Amazon SQS, AWS Lambda、Amazon ElastiCache

Elastic Cache のデータを更新してデータベースに保存

タスク	説明	必要なスキル
リスク結果を保存します。	ElastiCache で Var と PnL の数値が更新されると、5 分ごとに新しい Lambda 関数が呼び出されます。この関数は、保存されているすべてのデータを ElastiCache から読み取り、Aurora MySQL 互換デー	AWS Lambda、Amazon ElastiCache

タスク	説明	必要なスキル
	タベースと S3 バケットに保存します。	

関連リソース

- 「バーゼル Var フレームワーク」

Amazon Athena を使用して共有 AWS Glue データカタログへのクロスアカウントアクセスを構成する

作成者: Denis Avdonin (AWS)

概要

このパターンでは、AWS Identity and Access Management (IAM) ポリシーサンプルを含め、AWS Glue データカタログを使用して Amazon Simple Storage Service (Amazon S3) バケットに保管されたデータセットをクロスアカウントで共有できるように構成する手順を順を追って説明しています。データセットは S3 バケットに保存できます。メタデータは AWS Glue クローラーによって収集され、AWS Glue データカタログに格納されます。S3 バケットと AWS Glue データカタログは、データアカウントと呼ばれる AWS アカウントにあります。IAM プリンシパルには、コンシューマーアカウントと呼ばれる別の AWS アカウントでアクセスを提供できます。ユーザーは Amazon Athena サーバーレスクエリエンジンを使用して、コンシューマーアカウントのデータをクエリできます。

前提条件と制限

前提条件

- 2 つのアクティブな アカウント。
- AWS アカウントのいずれかにある S3 バケット
- Athena エンジンバージョン 2
- あらかじめインストールおよび構成されている AWS コマンドラインインターフェイス (AWS CLI) (AWS CLI コマンドを実行する場合は AWS CloudShell)

製品バージョン

このパターンは Athena エンジンバージョン 2 と Athena エンジンバージョン 3 でのみ機能します。Athena エンジンバージョン 3 にアップグレードすることをお勧めします。Athena エンジンバージョン 1 から Athena エンジンバージョン 3 にアップグレードできない場合は、AWS ビッグデータブログの「Amazon Athena でのクロスアカウント AWS Glue データカタログアクセス」で説明されているアプローチに従ってください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Athena
- Amazon Simple Storage Service (Amazon S3)
- AWS Glue
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)

次の図は、IAM 権限を使用して、ある AWS アカウント (データアカウント) の S3 バケットのデータを、AWS Glue データカタログを介して別の AWS アカウント (コンシューマーアカウント) と共有するアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. データアカウントの S3 バケットポリシーは、コンシューマーアカウントの IAM ロールとデータアカウントの AWS Glue クローラーサービスロールにアクセス権限を付与します。
2. データアカウントの AWS KMS キーポリシーは、コンシューマーアカウントの IAM ロールとデータアカウントの AWS Glue クローラーサービスロールにアクセス権限を付与します。
3. データアカウントの AWS Glue クローラーは、S3 バケットに保存されているデータのスキーマを検出します。
4. データアカウントの AWS Glue データカタログのリソースポリシーは、コンシューマーアカウントの IAM ロールへのアクセスを許可します。
5. ユーザーは AWS CLI コマンドを使用して、コンシューマーアカウントに名前付きカタログレファレンスを作成します。
6. IAM ポリシーは、コンシューマーアカウントの IAM ロールに、データアカウント内のリソースへのアクセスを許可します。IAM ロールの信頼ポリシーは、コンシューマーアカウントのユーザーが IAM ロールを引き受けることを許可します。
7. コンシューマーアカウントのユーザーは IAM ロールを引き受け、SQL クエリを使用してデータカタログ内のオブジェクトにアクセスします。
8. Athena サーバーレスエンジンは SQL クエリを実行します。

> **Note**
>
> IAM ベストプラクティスでは、IAM ロールにアクセス許可を付与し、ID フェデレーションを使用することをお勧めします。

ツール

- Amazon Athena はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接分析できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、ユーザーのデータを保護するために使用される、暗号化キーの作成と制御を容易にするマネージドサービスです。

エピック

データアカウントに権限を設定する

タスク	説明	必要なスキル
S3 バケットへのアクセス許可の付与	次のテンプレートに基づいて S3 バケットポリシーを作成し、データが保存されているバケットにそのポリシーを割り当てます。 `{ "Version": "2012-10-17", "Statement": [ {`	クラウド管理者

タスク	説明	必要なスキル
	"Effect": "Allow", "Principal": { "AWS": ["arn:aws:iam::<consumer account id>:role/<role name>", "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler"] }, "Action": "s3:GetObject", "Resource": "arn:aws:s3:::data-bucket/*" }, { "Effect": "Allow", "Principal": { "AWS": ["arn:aws:iam::<consumer account id>:role/<role name>", "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler"] },	

タスク	説明	必要なスキル

	<pre>"Action":
	"s3:ListBucket",
	"Resource":
	"arn:aws:s3:::data-
	bucket"
	}
	]
	}</pre>
	バケットポリシーは、コンシューマーアカウントの IAM ロールとデータアカウントの AWS Glue クローラーサービスロールにアクセス権限を付与します。	

タスク	説明	必要なスキル
(必要な場合) データ暗号化キーへのアクセスを付与します。	S3 バケットが AWS KMS キーで暗号化されている場合は、コンシューマーアカウントの IAM ロールとデータアカウントの AWS Glue クローラーサービスロールにキーの `kms:Decrypt` 許可を与えます。 キーポリシーを次のステートメントで更新します。 `{ "Effect": "Allow", "Principal": { "AWS": [ "arn:aws:iam::<consumer account id>:role/<role name>", "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler" ] }, "Action": "kms:Decrypt", "Resource": "arn:aws:kms:<region>:<data account id>:key/<key id>" }`	クラウド管理者

タスク	説明	必要なスキル

クローラーにデータへのアクセスを許可します。	以下の IAM ポリシーをクローラーのサービスロールにアタッチします。
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": "s3:GetObject",
	"Resource": "arn:aws:s3:::data-bucket/*"
	},
	{
	"Effect": "Allow",
	"Action": "s3:ListBucket",
	"Resource": "arn:aws:s3:::data-bucket"
	}
	]
	}
		クラウド管理者

タスク	説明	必要なスキル
(必要な場合) クローラーにデータ暗号化キーへのアクセス権限を付与します。	S3 バケットが AWS KMS キーで暗号化されている場合は、次のポリシーをアタッチして、キーに対する kms:Decrypt 許可をクローラーのサービスロールに与えます。 `{ "Effect": "Allow", "Action": "kms:Decrypt", "Resource": "arn:aws:kms:<region>:<data account id>:key/<key id>" }`	クラウド管理者

タスク	説明	必要なスキル
コンシューマーアカウントの IAM ロールとクローラーにデータカタログへのアクセス権限を付与します。	1. AWS マネジメントコンソールにサインインし、AWS Glueコンソールを開きます。 2. ナビゲーションペインの [Data catalog] で、[Settings] (設定) を選択します。 3. [権限] セクションで、次のステートメントを追加し、[保存] を選択します。 `{ "Version" : "2012-10-17", "Statement" : [ { "Effect" : "Allow", "Principal" : { "AWS" : [ "arn:aws:iam::<consumer account id>:role/<role name>", "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler" ] }, "Action" : "glue:*", "Resource" : [`	クラウド管理者

タスク	説明	必要なスキル

	<pre>"arn:aws:glue:<region>:<data account id>:catalog",
	
	"arn:aws:glue:<region>:<data account id>:database/*",
	
	"arn:aws:glue:<region>:<data account id>:table/*"
	]
	}
	]
	}</pre>
	このポリシーは、データアカウントのすべてのデータベースとテーブルに対するすべての AWS Glue アクションを許可します。ポリシーをカスタマイズして、必要なアクセス権限のみをコンシューマープリンシパルに付与することができます。たとえば、データベース内の特定のテーブルまたはビューへの読み取り専用アクセスなどです。	

コンシューマーアカウントからデータにアクセスする

タスク	説明	必要なスキル
データカタログの名前付きレファレンスを作成する。	名前付きデータカタログリファレンスを作成するには、CloudShell またはローカ	クラウド管理者

タスク	説明	必要なスキル
	ルにインストールされた AWS CLI を使用して次のコマンドを実行します。 `aws athena create-data-catalog --name <shared catalog name> --type GLUE --parameters catalog-id=<data account id>`	

タスク	説明	必要なスキル
コンシューマーアカウントの IAM ロールにデータへのアクセスを許可する。	次のポリシーをコンシューマーアカウントの IAM ロールにアタッチして、そのロールにデータへのクロスアカウントアクセスを許可します。 `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::data-bucket/*" }, { "Effect": "Allow", "Action": "s3:ListBucket", "Resource": "arn:aws:s3:::data-bucket" }, { "Effect": "Allow", "Action": "glue:*", "Resource": ["arn:aws:glue:<region>:<data account id>:catalog",`	クラウド管理者

タスク	説明	必要なスキル

	<pre>"arn:aws:glue:<region>:<data account id>:database/*",
	
	"arn:aws:glue:<region>:<data account id>:table/*"
	]
	}
	]
	}</pre>
	次に、以下のテンプレートを使用して、どのユーザーが IAM ロールを受け入れることができるかを信頼ポリシーで指定します。
	
	<pre>{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Principal": {
	"AWS": "arn:aws:iam::<consumer account id>:user/<IAM user>"
	},
	"Action": "sts:AssumeRole"
	}
	]
	}</pre>	

タスク	説明	必要なスキル
	最後に、同じポリシーをユーザーが属するユーザーグループにアタッチして、IAM ロールを引き受ける権限をユーザーに付与します。	
(必要な場合) コンシューマーアカウントの IAM ロールにデータ暗号化キーへのアクセス権限を付与します。	S3 バケットが AWS KMS キーで暗号化されている場合は、次のポリシーをアタッチして、キーに対する `kms:Decrypt` 許可をコンシューマーアカウントの IAM ロールに与えます。 `{ "Effect": "Allow", "Action": "kms:Decrypt", "Resource": "arn:aws:kms:<region>:<data account id>:key/<key id>" }`	クラウド管理者
コンシューマーアカウントの IAM ロールに切り替えてデータにアクセスします。	データコンシューマーは IAM ロールに切り替えてデータアカウントのデータにアクセスします。	データコンシューマー

タスク	説明	必要なスキル

データにアクセスします。	Athena を使用したクエリデータ。たとえば、Athena クエリエディタを開き、次のクエリを実行します。
	
	`SELECT * FROM <shared catalog name>.<database name>.<table name>`
	
	名前付きのカタログレファレンスを使用する代わりに、Amazon リソースネーム (ARN) でカタログを参照することもできます。
	
	Note
	クエリまたはビューで動的カタログ参照を使用する場合は、参照をエスケープされた二重引用符 (\") で囲みます。以下に例を示します。
	
	`SELECT * FROM \"glue:arn:aws:glue:<region>:<data account id>:catalog\".<database name>.<table name>`
		データコンシューマー

タスク	説明	必要なスキル
	詳細については、Amazon Athena ユーザーガイドの「AWS Glue データカタログへのクロスアカウントアクセス」を参照してください。	

関連リソース

- AWS Glue データカタログへのクロスアカウントアクセス (Athena ドキュメント)
- (CLI) create-data-catalog (AWS CLI コマンドリファレンス)
- Amazon Athena を使ったAWS Glue データカタログへのクロスアカウントアクセス (AWS ビッグデータブログ)
- IAM でのセキュリティのベストプラクティス (IAM ドキュメント)

追加情報

クロスアカウント共有の代替手段として Lake Formation を使用する

AWS Lake Formation を使用して、AWS Glue カタログオブジェクトへのアクセスをアカウント間で共有することもできます。Lake Formation は、列レベルと行レベルでのきめ細かいアクセス制御、タグベースのアクセス制御、ACID トランザクションの管理テーブル、その他の機能を提供します。Lake Formation は Athena と十分統合されていますが、このパターンの IAM のみのアプローチと比較すると、追加の構成が必要になります。Lake Formation を使用するか、IAM のみのアクセス制御を使用するかの決定は、ソリューションアーキテクチャ全体という広い視点で検討することをお勧めします。考慮すべき点には、他にどのようなサービスが関与しているか、またそれらを両方のアプローチとどのように統合するかなどがあります。

Teradata NORMALIZE 時間的特徴量を Amazon Redshift SQL に変換

作成者: Po Hong (AWS)

概要

NORMALIZE は ANSI SQL 標準に対するTeradata 拡張です。SQL テーブルに PERIOD データ型の列が含まれている場合、NORMALIZE はその列と一致するか重複する値を組み合わせて、複数の個別の期間値を統合する 1 つの期間を形成します。NORMARISEを使用するには、SQL の選択リストに少なくとも 1 つの列がテラデータの一時的な期間データ型である必要があります。NORMALIZEの詳細については、「Teradata のドキュメント」を参照してください。

Amazon Redshift にNORMALIZEは適用されませんが、ネイティブ SQL 構文と Amazon Redshift の LAG ウィンドウ関数を使用することでこの機能を実装できます。このパターンは、最も一般的な形式である ON MEETS OR OVERLAPS 条件で、Teradata NORMALIZEの拡張機能を使用することにフォーカスします。この特徴量が Teradata でどのように機能するかについて、また Amazon Redshift ネイティブ SQL 構文に変換する方法についても説明しています。

前提条件と制限

前提条件

- Teradata SQL の基本的な知識と経験
- Amazon Redshift の知識と経験

アーキテクチャ

ソーステクノロジースタック

- Teradataデータウェアハウス

ターゲットテクノロジースタック

- Amazon Redshift

ターゲットアーキテクチャ

Teradata データベースを Amazon Redshift に移行するための高レベルのアーキテクチャについては、「AWS SCT データ抽出エージェントを使用して、Teradataのデータベースを Amazon Redshift に移行する」というパターンを参照してください。移行しても、TeradataのNORMALIZEフレーズが Amazon Redshift SQL に自動的に変換されるわけではありません。このTeradata 拡張は、このパターンのガイドラインに従って変換できます。

ツール

Code

NORMALIZEの概念と機能を説明するために、Teradata で以下のテーブル定義を考慮します:

```
CREATE TABLE systest.project
     (    emp_id        INTEGER,
          project_name  VARCHAR(20),
          dept_id       INTEGER,
          duration      PERIOD(DATE)
     );
```

次の SQL コードを実行して、サンプルデータをテーブルに挿入します :

```
BEGIN TRANSACTION;

INSERT INTO systest.project VALUES (10, 'First Phase', 1000,  PERIOD(DATE '2010-01-10',
 DATE '2010-03-20') );
INSERT INTO systest.project VALUES (10, 'First Phase', 2000,  PERIOD(DATE '2010-03-20',
 DATE '2010-07-15') );

INSERT INTO systest.project VALUES (10, 'Second Phase', 2000,  PERIOD(DATE
 '2010-06-15', DATE '2010-08-18') );
INSERT INTO systest.project VALUES (20, 'First Phase', 2000,  PERIOD(DATE '2010-03-10',
 DATE '2010-07-20') );

INSERT INTO systest.project VALUES (20, 'Second Phase', 1000,  PERIOD(DATE
 '2020-05-10', DATE '2020-09-20') );

END TRANSACTION;
```

結果:

```
select * from systest.project order by 1,2,3;
```

```
 *** Query completed. 4 rows found. 4 columns returned.
 *** Total elapsed time was 1 second.

     emp_id  project_name              dept_id  duration
-----------  --------------------  -----------  ------------------------
         10  First Phase                  1000  ('10/01/10', '10/03/20')
         10  First Phase                  2000  ('10/03/20', '10/07/15')
         10  Second Phase                 2000  ('10/06/15', '10/08/18')
         20  First Phase                  2000  ('10/03/10', '10/07/20')
         20  Second Phase                 1000  ('20/05/10', '20/09/20')
```

Teradata NORMALIZEのユースケース

ここで、Teradata NORMALIZE SQL 句を SELECT ステートメントに追加します:

```
SELECT NORMALIZE ON MEETS OR OVERLAPS emp_id, duration
FROM systest.project
ORDER BY 1,2;
```

このNORMALIZE のオペレーションは、単一の列 (emp_id)で実行されます。emp_id=10 の場合、次のように、期間内で重複する三つの期間値が合体して 1 つの期間値になります:

```
     emp_id  duration
-----------  ------------------------
         10  ('10/01/10', '10/08/18')
         20  ('10/03/10', '10/07/20')
         20  ('20/05/10', '20/09/20')
```

次のSELECTステートメントは、 project_name と dept_id でNORMALIZEオペレーションを実行します。SELECT リストには、期間という 1 つの期間列しか含まれていないことに注意してください。

```
SELECT NORMALIZE project_name, dept_id, duration
FROM systest.project;
```

出力:

```
project_name              dept_id  duration
--------------------  -----------  ------------------------
```

```
First Phase                  1000  ('10/01/10', '10/03/20')
Second Phase                 1000  ('20/05/10', '20/09/20')
First Phase                  2000  ('10/03/10', '10/07/20')
Second Phase                 2000  ('10/06/15', '10/08/18')
```

Amazon Redshift と同等の SQL

Amazon Redshift には現在、テーブルの期間データタイプが適用されません。代わりに、Teradata の 期間 データフィールドを次のように start_date と end_date の二つの部分に分割する必要があります:

```
CREATE TABLE systest.project
     (    emp_id        INTEGER,
          project_name  VARCHAR(20),
          dept_id       INTEGER,
          start_date  DATE,
          end_date    DATE
     );
```

テーブルにデータ行を挿入します:

```
BEGIN TRANSACTION;

INSERT INTO systest.project VALUES (10, 'First Phase', 1000,  DATE '2010-01-10', DATE
 '2010-03-20' );
INSERT INTO systest.project VALUES (10, 'First Phase', 2000,  DATE '2010-03-20', DATE
 '2010-07-15');

INSERT INTO systest.project VALUES (10, 'Second Phase', 2000,  DATE '2010-06-15', DATE
 '2010-08-18' );
INSERT INTO systest.project VALUES (20, 'First Phase', 2000,  DATE '2010-03-10', DATE
 '2010-07-20' );

INSERT INTO systest.project VALUES (20, 'Second Phase', 1000,  DATE '2020-05-10', DATE
 '2020-09-20' );

END TRANSACTION;
```

出力:

```
 emp_id | project_name | dept_id | start_date |  end_date
```

```
--------+--------------+---------+------------+------------
     10 | First Phase  |    1000 | 2010-01-10 | 2010-03-20
     10 | First Phase  |    2000 | 2010-03-20 | 2010-07-15
     10 | Second Phase |    2000 | 2010-06-15 | 2010-08-18
     20 | First Phase  |    2000 | 2010-03-10 | 2010-07-20
     20 | Second Phase |    1000 | 2020-05-10 | 2020-09-20
(5 rows)
```

TeradataのNORMALIZE 句を書き直すには、Amazon Redshift の 「LAG ウィンドウ関数」 を使用できます。この機能では、パーティションの現在の行より上 (以前) の指定されたオフセットの行の値を返します。

LAG 関数を使用して、ある期間が前の期間と一致するか重複しているかどうか (「はい」の場合は 0、「いいえ」の場合は 1) により、新しい期間を開始する各行を識別できます。このフラグを累積的に合計して、Amazon Redshift で望ましい結果を得るために、外部の グループ分け 句で使用できるグループ ID を提供します。

LAG () を使用する Amazon Redshift SQL ステートメントのサンプルは次のとおりです:

```
SELECT emp_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY
 start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project
ORDER BY 1,2;
```

出力:

```
 emp_id | start_date |  end_date  | groupstartflag
--------+------------+------------+----------------
     10 | 2010-01-10 | 2010-03-20 |              1
     10 | 2010-03-20 | 2010-07-15 |              0
     10 | 2010-06-15 | 2010-08-18 |              0
     20 | 2010-03-10 | 2010-07-20 |              1
     20 | 2020-05-10 | 2020-09-20 |              1
(5 rows)
```

次の Amazon Redshift SQL ステートメントは、emp_id 列でのみ正規化されます:

```
SELECT T2.emp_id, MIN(T2.start_date) as new_start_date, MAX(T2.end_date) as
 new_end_date
FROM
```

```
( SELECT T1.*, SUM(GroupStartFlag) OVER (PARTITION BY emp_id ORDER BY start_date ROWS
 UNBOUNDED PRECEDING) As GroupID
FROM ( SELECT emp_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY
 start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project ) T1
) T2
GROUP BY T2.emp_id, T2.GroupID
ORDER BY 1,2;
```

出力:

```
 emp_id | new_start_date | new_end_date
--------+----------------+-----------------------------------
     10 | 2010-01-10     | 2010-08-18
     20 | 2010-03-10     | 2010-07-20
     20 | 2020-05-10     | 2020-09-20
(3 rows)
```

次の Amazon Redshift SQL ステートメントは、project_name 列と dept_id 列の両方を正規化します:

```
SELECT T2.project_name, T2.dept_id, MIN(T2.start_date) as new_start_date,
 MAX(T2.end_date) as new_end_date
FROM
( SELECT T1.*, SUM(GroupStartFlag) OVER (PARTITION BY project_name, dept_id ORDER BY
 start_date ROWS UNBOUNDED PRECEDING) As GroupID
FROM ( SELECT project_name, dept_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY project_name,
 dept_id ORDER BY start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project ) T1
) T2
GROUP BY T2.project_name, T2.dept_id, T2.GroupID
ORDER BY 1,2,3;
```

出力:

```
 project_name | dept_id | new_start_date | new_end_date
--------------+---------+----------------+--------------
 First Phase  |    1000 | 2010-01-10     | 2010-03-20
 First Phase  |    2000 | 2010-03-10     | 2010-07-20
```

```
 Second Phase |    1000 | 2020-05-10     | 2020-09-20
 Second Phase |    2000 | 2010-06-15     | 2010-08-18
(4 rows)
```

エピック

NORMALIZE を Amazon Redshift SQL に変換

タスク	説明	必要なスキル
Teradata SQL コードを作成します。	必要に応じて NORMALIZE フレーズを使用してください。	SQL Developer
コードを Amazon Redshift SQL に変換します。	コードを変換するには、このパターンの「ツール」セクションのガイドラインに従います。	SQL Developer
Amazon Redshift でコードを実行します。	テーブルを作成し、テーブルにデータをロードして、Amazon Redshift でコードを実行します。	SQL Developer

関連リソース

リファレンス

- 「Teradata NORMALIZE 時間的特徴量」 (テラデータのドキュメント)
- 「LAG ウィンドウ関数」 (Amazon Redshift ドキュメント)
- 「Amazon Redshiftに移行」(AWS ウェブサイト)
- 「AWS SCT データ抽出エージェントを使用してTeradata　データベースを Amazon Redshift に移行する」(AWS 規範ガイダンス)
- 「Teradata RESET WHEN 特徴量を Amazon Redshift SQL」 (AWS 規範ガイダンス) に変換

ツール

- 「AWS Schema Conversion Tool (AWS SCT)」

パートナー

- 「AWS 移行コンピテンシーパートナー」

Teradata RESET WHEN 特徴量を Amazon Redshift SQL に変換

作成者: Po Hong (AWS)

概要

RESET WHEN は、SQL 分析ウィンドウ関数で使用されるテラデータの特徴量です。それは ANSI SQL 標準の拡張です。RESET WHEN は、特定の条件に基づいてSQL ウィンドウ関数が動作するパーティションを決定します。条件が TRUEと評価されると、既存のウィンドウパーティションの中に新しい動的サブパーティションが作成されます。RESET WHENの詳細については、「テラデータのドキュメント」を参照してください。

Amazon Redshift は SQL ウィンドウ関数で RESET WHEN をサポートしていません。この機能を実装するには、 RESET WHEN を Amazon Redshift でネイティブ SQL 構文に変換し、複数の入れ子関数を使用する必要があります。このパターンは、Teradata RESET WHEN 特徴量を使用する方法と、 Amazon Redshift SQL 構文に変換する方法を示しています。

前提条件と制限

前提条件

- Teradata のデータウェアハウスとその SQL 構文の基本的な知識
- Amazon Redshift とその SQL 構文の十分な理解

アーキテクチャ

ソーステクノロジースタック

- Teradataデータウェアハウス

ターゲットテクノロジースタック

- Amazon Redshift

アーキテクチャ

Teradataデータベースを Amazon Redshift に移行するための高レベルのアーキテクチャについては、「AWS SCT データ抽出エージェントを使用して、 Teradataのデータベースを Amazon Redshift に移行する」というパターンを参照してください。移行しても、Teradata RESET WHEN フレーズ

は Amazon Redshift SQL に自動的に変換されません。このTeradata拡張は、次のセクションのガイドラインに従って変換できます。

ツール

Code

RESET WHEN の概念を説明するには、テラデータにおける以下のテーブル定義を考慮します:

```
create table systest.f_account_balance
( account_id integer NOT NULL,
  month_id integer,
  balance integer )
unique primary index (account_id, month_id);
```

次の SQL コードを実行して、サンプルデータをテーブルに挿入します :

```
BEGIN TRANSACTION;
Insert Into systest.f_account_balance values (1,1,60);
Insert Into systest.f_account_balance values (1,2,99);
Insert Into systest.f_account_balance values (1,3,94);
Insert Into systest.f_account_balance values (1,4,90);
Insert Into systest.f_account_balance values (1,5,80);
Insert Into systest.f_account_balance values (1,6,88);
Insert Into systest.f_account_balance values (1,7,90);
Insert Into systest.f_account_balance values (1,8,92);
Insert Into systest.f_account_balance values (1,9,10);
Insert Into systest.f_account_balance values (1,10,60);
Insert Into systest.f_account_balance values (1,11,80);
Insert Into systest.f_account_balance values (1,12,10);
END TRANSACTION;
```

このサンプルテーブルには、次のデータがあります :

account_id	month_id	balance
1	1	60
1	2	99
1	3	94

1	4	90
1	5	80
1	6	88
1	7	90
1	8	92
1	9	10
1	10	60
1	11	80
1	12	10

アカウントごとに、連続的な月次残高の増加のシーケンスを分析したいものとします。ある月の残高が前月の残高以下の場合、必要なことはカウンターをゼロにリセットして再起動します。

Teradata RESET WHENのユースケース

このデータを分析するためにTeradata SQL は、ネストされた集計ウィンドウ関数と RESET WHEN フレーズを次のように使用します:

```
SELECT account_id, month_id, balance,
 ( ROW_NUMBER() OVER (PARTITION BY account_id ORDER BY month_id
RESET WHEN balance <= SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS
 BETWEEN 1 PRECEDING AND 1 PRECEDING) ) -1 ) as balance_increase
FROM systest.f_account_balance
ORDER BY 1,2;
```

出力:

account_id	month_id	balance	balance_increase
1	1	60	0
1	2	99	1

1	3	94	0
1	4	90	0
1	5	80	0
1	6	88	1
1	7	90	2
1	8	92	3
1	9	10	0
1	10	60	1
1	11	80	2
1	12	10	0

Teradataにクエリが次のように処理されます:

1. SUM (残高)集計関数は、特定の口座の特定の月のすべての残高の合計を計算します。
2. 特定の月の（特定の口座の）残高が前月残高を超えているかどうかを確認します。
3. 残高が増加した場合、累積のカウント値を追跡します。RESET WHEN の条件がfalse と評価された場合、つまり、残高が連続する月について増加した場合、引き続きカウントを増加します。
4. ROW_NUMBER ()で順序付けされた分析関数がカウントの値を計算します。残高が前月の残高以下になると、RESET WHEN 条件で true と評価します。その場合、新しいパーティションを開始し、ROW_NUMBER ()でカウントを 1 から再開します。ROWS BETWEEN 1 PRECEDING AND 1 PRECEDINGを使用して、前の行の値にアクセスします。
5. 1 を差し引くことで、カウント値が 必ず0 から始まるようにします。

Amazon Redshift と同等の SQL

SQL 分析ウィンドウ関数の RESET WHEN フレーズに Amazon Redshift が適用されません。 同じ結果を生むために、 Amazon Redshift ネイティブ SQL 構文とネストされたサブクエリを使用して、テラデータの SQL を次のように書き直す必要があります:

```
SELECT account_id, month_id, balance,
   (ROW_NUMBER() OVER(PARTITION BY account_id, new_dynamic_part ORDER BY month_id) -1)
 as balance_increase
FROM
( SELECT account_id, month_id, balance, prev_balance,
SUM(dynamic_part) OVER (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN
 UNBOUNDED PRECEDING AND CURRENT ROW) As new_dynamic_part
FROM ( SELECT account_id, month_id, balance,
SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN 1 PRECEDING
 AND 1 PRECEDING) as prev_balance,
(CASE When balance <= prev_balance Then 1 Else 0 END) as dynamic_part
FROM systest.f_account_balance ) A
) B
ORDER BY 1,2;
```

単一の SQL ステートメントのSELECT 句にネストされたウィンドウ関数に、Amazon Redshift が適用されないため、2つのネストされたサブクエリを使用する必要があります。

- 内部サブクエリ (エイリアス A) では、動的パーティションインジケータ (dynamic_part) が作成され、入力されます。ある月の残高が前月残高の以下の場合、 dynamic_part が 1 に設定され、それ以外の場合は 0 に設定されます。
- 次のレイヤー (エイリアス B) では、new_dynamic_part 属性が、SUM ウィンドウ関数の結果として、生成されます。
- 最後に、新しいパーティション属性 (動的パーティション) として、 new_dynamic_part を既存のパーティション属性 (account_id) に追加します。そしてテラデータに同じ ROW_NUMBER()ウィンドウ関数を適用します(1 を差し引いて) 。

こうした変更後、Amazon Redshift SQL がTeradataと同じように出力を生成します。

エピック

RESET WHEN を Amazon Redshift SQL に変換

タスク	説明	必要なスキル
Teradataのウィンドウ関数を作成します。	必要に応じて、ネストされた集計と RESET WHEN フレーズを使用します。	SQL Developer

タスク	説明	必要なスキル
コードを Amazon Redshift SQL に変換します。	コードを変換するには、このパターンの「ツール」セクションのガイドラインに従います。	SQL Developer
Amazon Redshift でコードを実行します。	テーブルを作成し、テーブルにデータをロードして、Amazon Redshift でコードを実行します。	SQL Developer

関連リソース

リファレンス

- 「RESET WHEN フレーズ」（Teradataのドキュメント）
- 「RESET WHEN の説明」（スタックオーバーフロー）
- 「Amazon Redshiftに移行」（AWS ウェブサイト）
- 「AWS SCT データ抽出エージェントを使用して、Teradataデータベースを Amazon Redshift に移行する」(AWS 規範ガイダンス)
- 「Teradata NORMALIZE 時間的特徴量を Amazon Redshift SQL に変換」(AWS 規範ガイダンス)

ツール

- 「AWS Schema Conversion Tool (AWS SCT)」

パートナー

- 「AWS 移行コンピテンシーパートナー」

インフラストラクチャをコードとして使用して、AWS クラウドにサーバーレスデータレイクをデプロイして管理する

作成者: Kirankumar Chandrashekar (AWS)、Abdel Jaidi (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

このパターンでは、サーバーレスコンピューティングと Infrastructure as Code (IaC) を使用して、Amazon Web Services (AWS) クラウドにデータレイクを実装および管理する方法を説明します。このパターンは、AWS が開発したサーバーレスデータレイクフレームワーク (SDLF) ワークショップに基づいています。

SDLF は、AWS クラウドでのエンタープライズデータレイクの配信を加速し、本番環境への迅速なデプロイに役立つ再利用可能なリソースの集まりです。ベストプラクティスに従ってデータレイクの基本構造を実装するために使用されます。

SDLF は、AWS CodePipeline、AWS CodeBuild、AWS CodeCommit などの AWS サービスを使用して、コードとインフラストラクチャのデプロイ全体にわたって継続的インテグレーション/継続的デプロイ (CI/CD) プロセスを実装します。

このパターンでは、複数の AWS サーバーレスサービスを使用してデータレイク管理を簡素化します。これらには、ストレージ用に Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB、コンピューティング用に AWS Lambda と AWS Glue、オーケストレーション用に Amazon CloudWatch Events、Amazon Simple Queue Service (Amazon SQS)、AWS Step Functions などがあります。

AWS CloudFormation と AWS コードサービスは IaC レイヤーとして機能し、運用と管理が容易で再現性のある迅速なデプロイを実現します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- インストールおよび設定済みの AWS コマンドラインインターフェイス (AWS CLI)。
- インストールおよび設定済みの Git クライアント。

- SDLF ワークショップはウェブブラウザウィンドウで開き、すぐに使用できます。

アーキテクチャ

このアーキテクチャ図は、イベント駆動型プロセスを以下のステップで示しています。

1. ローデータの S3 バケットにファイルを追加すると、Amazon S3 イベント通知が SQS キューに配置されます。各通知は、S3 バケット名、オブジェクトキー、タイムスタンプなどのメタデータを含む JSON ファイルとして配信されます。
2. この通知は、メタデータに基づいてイベントを正しい抽出、変換、ロード (ETL) プロセスにルーティングする Lambda 関数によって使用されます。Lambda 関数は Amazon DynamoDB テーブルに保存されているコンテキスト設定を使用することもできます。このステップにより、データレイク内の複数のアプリケーションを切り離してスケーリングできるようになります。
3. イベントは、ETL プロセスの最初の Lambda 関数にルーティングされます。この関数はデータを未加工データ領域からデータレイクのステージング領域に変換して移動します。最初のステップでは、総合カタログを更新します。これは、データレイクのすべてのファイルメタデータを含む DynamoDB テーブルです。このテーブルの各行には、Amazon S3 に保存されている 1 つのオブジェクトに関する運用メタデータが格納されています。S3 オブジェクトに対して軽い変換を行う Lambda 関数に対して同期呼び出しが行われます。この変換は、計算コストのかからない操作 (ファイルをある形式から別の形式に変換するなど) です。ステージング S3 バケットに新しいオブジェクトが追加されたため、包括的なカタログが更新され、ETL の次のフェーズを待つメッセージが SQS キューに送信されます。
4. CloudWatch イベントルールは 5 分ごとに Lambda 関数をトリガーします。この関数は、メッセージが前の ETL フェーズから SQS キューに配信されたかどうかを確認します。メッセージが配信されると、Lambda 関数は ETL プロセスの AWS Step Functions から 2 番目の関数を開始します。
5. その後、大量の変換がファイルのバッチに適用されます。この大規模な変換は、AWS Glue ジョブ、AWS Fargate タスク、Amazon EMR ステップ、または Amazon SageMaker ノートブックへの同期呼び出しなど、計算負荷の高い操作です。テーブルメタデータは、AWS Glue カタログを更新する AWS Glue クローラーを使用して出力ファイルから抽出されます。ファイルメタデータは DynamoDB の包括的なカタログテーブルにも追加されます。最後に、Deequ を活用したデータ品質ステップも実行されます。

テクノロジースタック

- Amazon CloudWatch Events
- AWS CloudFormation
- CodePipeline
- CodeBuild
- AWS CodeCommit
- Amazon DynamoDB
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SQS
- AWS Step Functions

ツール

- Amazon CloudWatch Events – CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのストリームをほぼリアルタイムで配信します。
- AWS CloudFormation – CloudFormation は、AWS インフラストラクチャのデプロイメントを予測どおりに繰り返し作成およびプロビジョニングするのに役立ちます。
- AWS CodeBuild – CodeBuild は、ソースコードをコンパイルし、ユニットテストを実行し、すぐにデプロイできるアーティファクトを生成するフルマネージドビルドサービスです。
- AWS CodeCommit – CodeCommit は、AWS がホストするバージョン管理サービスで、資産 (ソースコードやバイナリファイルなど) をプライベートに保存および管理するために使用できます。
- AWS CodePipeline – CodePipeline は、ソフトウェアの変更を継続的にリリースするために必要なステップをモデル化し、視覚化し、自動化するために使用できる継続的配信サービスです。
- Amazon DynamoDB – DynamoDB は、高速で予測可能なパフォーマンスとスケーラビリティを提供するフルマネージド NoSQL データベースサービスです。
- AWS Glue – AWS Glue は、分析のためにデータを簡単に準備してロードできるフルマネージドの ETL サービスです。
- AWS Lambda – Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートします。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスです。Amazon S3 は、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- AWS Step Functions – AWS Step Functions はサーバーレスの関数オーケストレーターで、AWS Lambda 関数と複数の AWS サービスをビジネスクリティカルなアプリケーションに簡単にシーケンスできます。
- Amazon SQS – Amazon Simple Queue Service (Amazon SQS) は、マイクロサービス、分散システム、サーバーレスアプリケーションの分離と拡張に役立つフルマネージドのメッセージキューイングサービスです。
- Deequ – Deequ は、大規模なデータセットのデータ品質メトリクスの計算、データ品質制約の定義と検証、データ分布の変化に関する最新情報の把握に役立つツールです。

コードリポジトリ

SDLF のソースコードとリソースは、AWS ラボの GitHub リポジトリにあります。

エピック

IaC をプロビジョニングするため CI/CD パイプラインを設定する

タスク	説明	必要なスキル
データレイクの IaC を管理するための CI/CD パイプラインを設定します。	AWS マネジメントコンソールにサインインし、SDLF ワークショップの「Initial setup」セクションの手順に従います。これにより、データレイク用の IaC をプロビジョニングおよび管理する CodeCommit リポジトリ、CodeBuild 環境、CodePipeline パイプラインなどの初期の CI/CD リソースが作成されます。	DevOps エンジニア

IaC のバージョン管理

タスク	説明	必要なスキル
ローカルマシンに CodeCommit リポジトリをクローンします。	SDLF ワークショップの「Deploying the Foundations」セクションの手順に従ってください。これにより、IaC をホストする Git リポジトリをローカル環境に複製できます。 詳細については、CodeCommit ドキュメントの「Connect to an AWS CodeCommit repository」を参照してください。	DevOps エンジニア
CloudFormation テンプレートを変更します。	ローカルワークステーションとコードエディタを使用して、ユースケースや要件に応じて CloudFormation テンプレートを変更します。それらをローカルにクローンされた Git リポジトリにコミットします。 詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormation テンプレートの使用」を参照してください。	DevOps エンジニア
変更を CodeCommit リポジトリにプッシュします。	インフラストラクチャコードはバージョン管理下に置かれ、コードベースへの変更が追跡されるようになりました。CodeCommit リポジトリ	DevOps エンジニア

タスク	説明	必要なスキル
	に変更をプッシュすると、CodePipeline はその変更をインフラストラクチャに自動的に適用し、CodeBuild に配信します。 **Important** CodeBuild で AWS SAM CLI を使用する場合は、 コマンド`sam package`と `sam deploy` コマンドを実行します。AWS CLI を使用する場合は、`aws cloudformation package` コマンドおよび `aws cloudformation deploy` コマンドを実行します。	

関連リソース

IaC をプロビジョニングするため CI/CD パイプラインを設定する

- SDLF workshop – Initial setup

IaC のバージョン管理

- SDLF workshop – Deploying the foundations
- AWS CodeCommit リポジトリに接続する
- AWS CloudFormation テンプレートの使用

その他のリソース

- AWS serverless data analytics pipeline reference architecture
- SDLF documentation

起動時に Amazon EMR クラスターのタグ付けを強制する

作成者: Priyanka Chaudhary (AWS)

概要

このパターンは、Amazon EMR クラスターによる作成時のタグ付けを保証するセキュリティコントロールを提供します。

Amazon EMR は、膨大な量のデータを処理して分析するためのAmazon Web Services (AWS) サービスです。Amazon EMR は、社内でクラスターコンピューティングを実行する代わりに、拡張可能な低構成のサービスを提供します。タグ付けを使用し、目的、所有者、環境などのさまざまな方法で AWS リソースを分類できます。たとえば、各クラスターにカスタムメタデータを割り当てることで、Amazon EMR クラスターにタグを付けることができます。タグは、定義するキーと値で構成されます。組織の要件に適合する一貫したタグのセットを作成することをお勧めします。Amazon EMR クラスターにタグを追加するとき、タグはクラスターに関連するアクティブな Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにもそれぞれ伝達されます。同様に、Amazon EMR クラスターからタグを削除すると、そのタグは関連するアクティブな EC2 インスタンスそれぞれから削除されます。

検出コントロールは、API コールをモニタリングし、RunJobFlow、AddTags、RemoveTags、および CreateTags API の Amazon CloudWatch Events イベントを起動します。このイベントは、Python スクリプトを実行する AWS Lambda 関数を呼び出します。Python 関数は、イベントの JSON 入力から Amazon EMR クラスター ID を取得し、以下のチェックを実行します。

- Amazon EMR クラスターが、指定したタグ名で設定されていることを確認します。
- そうでない場合は、Amazon EMR クラスター名、違反の詳細、AWS リージョン、AWS アカウント、およびこの通知の送信元である Lambda の Amazon リソースネーム (ARN) などの関連情報 を含む Amazon Simple Notification Service (Amazon SNS) 通知をユーザーに送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 指定の Lambda コードをアップロードする Amazon Simple Storage Service (Amazon S3) バケット。または、「エピック」セクションで説明されているように、この目的で S3 バケットを作成することもできます。

- 違反の通知を受信するアクティブなメールアドレス
- 確認したい必須タグのリスト。

機能制限

- このセキュリティコントロールは地域ごとに行われます。モニタリングする AWS リージョンごとにデプロイする必要があります。

製品バージョン

- Amazon EMR リリース 4.8.0 以降

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、モニタリングする複数のアカウントにこのテンプレートをデプロイできます。

ツール

AWS サービス

- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理し、プロビジョニングすることが可能です。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを配信します。
- Amazon EMR - Amazon EMR は、ビッグデータフレームワークの実行と膨大な量のデータの効率的な処理を簡素化するウェブサービスです。

- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、オブジェクトストレージサービスです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージ配信または送信を調整して管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、以下の添付ファイルが含まれます。

- `EMRTagValidation.zip` — セキュリティコントロール用の Lambda コード。
- `EMRTagValidation.yml` — イベントとLambda 関数をセットアップする CloudFormation テンプレート。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを定義します。	Amazon S3 コンソールで、Lambda コードの.zip ファイルをホストする S3 バケットを選択または作成します。この S3 バケットが、モニタリングする Amazon EMR クラスターと同じ AWS リージョンに存在する必要があります。Amazon S3 バケット名はグローバルに一意であり、名前空間はすべての AWS ア	クラウドアーキテクト

タスク	説明	必要なスキル
	カウントによって共有されています。S3 バケット名には、先頭にスラッシュを含めることはできません。	
Lambda コードをアップロードします。	「添付ファイル」セクションにある Lambda コードの.zip ファイルを S3 バケットにアップロードします。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートを起動します。	同じAWS リージョンで S3 バケットとして AWS CloudFormation コンソール を開き、テンプレートをデプロイします。AWS CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントの「AWS CloudFormation コンソールでのスタックの作成」を参照してください。	クラウドアーキテクト
テンプレートのパラメータを入力します。	テンプレートを起動すると、次の情報の入力を求められます。 • S3 バケット: 最初のエピックで作成または選択したバケットを指定します。ここで添付の Lambda コー	クラウドアーキテクト

タスク	説明	必要なスキル
	ド (.zip ファイル) をアップロードしました。 • S3 キー: S3 バケット内の Lambda .zip ファイルの場所を指定します (たとえば、<ファイル名>.zip または <コントロール/ファイル名>.zip)。先頭にスラッシュを含めないでください。 • 通知 メールアドレス: Amazon SNS 通知を受け取るメールアドレスを入力します。 • キー名のタグ付け: チェックするタグ (`ApplicationID` 、`Environment` 、`Owner` など)をカンマで区切ったリストで指定します。CloudWatch Events イベントは、クラスター内のこれらのタグをモニタリングし、見つからない場合は通知を送信します。 • Lamba ログ記録レベル: Lambda 関数のログ記録のレベルと頻度を指定します。[情報] を使用して、進行状況に関する詳細な情報メッセージをログに記録し、引き続きデプロイを続行できるエラーイベントには「エラー」、潜在的に有害な状況の場合は「警告」	

タスク	説明	必要なスキル
	を使用します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定したメールアドレスにサブスクリプションメールが送信されます。違反通知の受信を開始するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS Lambda 開発者ガイド
- 「Amazon EMR クラスターでのタグ付け」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon S3 へのAmazon EMR 口ギングが有効になっていることを確認する

作成者: Priyanka Chaudhary (AWS)

概要

このパターンは、Amazon Web Services (AWS) で実行されている Amazon EMR クラスターの口ギング設定を監視するセキュリティコントロールを提供します。

Amazon EMR は、ビッグデータの処理と分析のための AWS ツールです。Amazon EMR は、社内でクラスターコンピューティングを実行する代わりに、拡張可能な低構成のサービスを提供します。Amazon EMR には 2 種類の EMR クラスターが用意されています。

- 一時的な Amazon EMR クラスター:一時的な Amazon EMR クラスターは、処理が終了すると自動的にシャットダウンされ、コストが発生しなくなります。
- 永続的な Amazon EMR クラスター:永続的な Amazon EMR クラスターは、データ処理ジョブが完了した後も引き続き実行されます。

Amazon EMR と Hadoop はいずれも、クラスターの状態を報告するログファイルを生成します。デフォルトで、ログファイルは /mnt/var/log/ ディレクトリのマスターノードに出力されます。クラスターの起動時の設定方法によっては、Amazon Simple Storage Service (Amazon S3) にログを保存できます。Amazon S3 口ギングは、クラスターの起動時にのみ指定できることに注意してください。この設定では、口グは 5 分ごとにプライマリノードから Amazon S3 の場所に送信されます。一時的なクラスターでは、処理が完了するとクラスターが消え、これらの口グファイルを使用して失敗したジョブをデバッグできるため、Amazon S3 口ギングは重要です。

このパターンでは、AWS CloudFormation テンプレートを使用して API 呼び出しをモニタリングし、「RunJobFlow」で Amazon CloudWatch Events を開始するセキュリティコントロールをデプロイします。トリガーは、Python スクリプトを実行する AWS Lambda を呼び出します。Lambda 関数はイベント JSON 入力から EMR クラスター ID を取得し、Amazon S3 口グ URI もチェックします。Amazon S3 URI が見つからない場合、Lambda 関数は、EMR クラスター名、違反の詳細、AWS リージョン、AWS アカウント、および通知の送信元である Lambda Amazon リソースネーム (ARN) を詳述した Amazon Simple Notification Service (Amazon SNS) 通知を送信します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Lambda コードの .zip ファイル用の S3 バケット
- 違反の通知を受信する E メールアドレス

制約事項

- この検出統制はリージョンごとに適用されるため、監視対象の AWS リージョンに導入する必要があります。

製品バージョン

- Amazon EMR 5.16.0 以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon CloudWatch Events イベント
- Amazon EMR
- Lambda function
- S3 バケット
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、監視したい複数のアカウントにこのテンプレートをデプロイできます。

ツール

ツール

- 「AWS CloudFormation」— AWS CloudFormation は、インフラストラクチャをコードとして使用して AWS リソースをモデル化およびセットアップするのに役立ちます。
- 「AWS Cloudwatch Events」– AWS CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「Amazon EMR」— Amazon EMR は、ビッグデータフレームワークの実行を簡素化するマネージド型クラスタープラットフォームです。
- 「AWS Lambda」— AWS Lambda は、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- 「Amazon S3」– Amazon S3 はウェブサービスインターフェイスで、ウェブのどこからでも容量に関係なくデータを格納および取得できます。
- 「Amazon SNS」— Amazon SNS は、ウェブサーバーや E メールアドレスを含む、パブリッシャーとクライアント間のメッセージの配信または送信を、調整および管理するウェブサービスです。

コード

- プロジェクトの.zip ファイルは添付ファイルとして入手できます。

エピック

S3 バケットを定義

タスク	説明	必要なスキル
S3 バケットを定義します。	Lambda コードの .zip ファイルをホストするには、先頭にスラッシュを含まない一意の名前で S3 バケットを選択または作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケットは、評価されている Amazon EMR	クラウドアーキテクト

タスク	説明	必要なスキル
	クラスターと同じ AWS リージョンに存在する必要があります。	

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
S3 バケットに Lambda コードをアップロードします。	「添付ファイル」セクションで提供されている Lambda コードの.zip ファイルを S3 バケットにアップロードします。S3 バケットは、評価されている Amazon EMR クラスターと同じリージョンに存在する必要があります。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをデプロイします。	AWS CloudFormation コンソールで、S3 バケットと同じリージョンに、このパターンへのアタッチメントとして提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を指定します。AWS CloudFormation テンプレートのデプロイの詳細について	クラウドアーキテクト

タスク	説明	必要なスキル
	は、「関連リソース」セクションを参照してください。	

AWS CloudFormation テンプレートのパラメータを入力します

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成した S3 バケットの名前を入力します。	クラウドアーキテクト
Amazon S3 キーを指定します。	S3 バケット内の Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例、<directory>/<file-name>.zip)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ロギングのレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。「Info」はアプリケーションの進行状況に関する詳細な情報メッセージを表します。「Error」は、それでもアプリケーションの実行を継続できるエラーイベントを示します。「警告」は潜在的に有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定したメールアドレスに購読メールメッセージが送信されます。違反通知を受信するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS Lambda
- 「Amazon EMR ロギング」
- 「AWS CloudFormation テンプレートをデプロイ」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Glue ジョブと Python を使用してテストデータを生成します

作成者: Moinul Al-Mamun (AWS)

概要

このパターンは、Python で記述された AWS Glue ジョブを作成することで、何百万ものサンプルファイルをすばやく簡単に同時に生成する方法を示しています。サンプルファイルは Amazon Simple Storage Service (Amazon S3) のバケットに保存されています。大量のサンプルファイルをすばやく生成できることは、AWS クラウド内のサービスをテストまたは評価する上で重要です。たとえば、Amazon S3 プレフィックス内の何百万もの小さなファイルに対してデータ分析を実行することで、AWS Glue Studio または AWS Glue DataBrew ジョブのパフォーマンスをテストできます。

他の AWS のサービスを使用してサンプルデータセットを生成できますが、AWS Glue を使用することをお勧めします。AWS Glue はサーバーレスのデータ処理サービスであるため、インフラストラクチャを管理する必要はありません。コードを持ち込んで AWS Glue クラスターで実行するだけで済みます。さらに、AWS Glue はジョブの実行に必要なリソースをプロビジョニング、設定、スケーリングします。ジョブの実行中に使用したリソースに対してのみ料金を支払います。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS コマンドラインインターフェイス(AWS CLI)、AWS アカウントと連携するように「インストール」および「設定」されている

製品バージョン

- Python 3.9
- AWS CLI バージョン 2

機能制限

トリガーあたりの AWS Glue ジョブの最大数は 50 です。詳細については、「AWS Glue のエンドポイントとクォータ」を参照してください。

アーキテクチャ

次の図は、出力 (つまりサンプルファイル) を S3 バケットに書き込む AWS Glue ジョブを中心としたアーキテクチャの例を示しています。

この図には、次のワークフローが含まれている。

1. AWS CLI、AWS マネジメントコンソール、または API を使用して、AWS Glue ジョブの開始を行います。AWS CLI または API を使用すると、呼び出されたジョブの並列化を自動化し、サンプルファイルを生成するためのランタイムを短縮できます。
2. AWS Glue ジョブは、ファイルコンテンツをランダムに生成し、そのコンテンツを CSV 形式に変換して、共通のプレフィックスの下で Amazon S3 オブジェクトとして保存します。各ファイルは 1 KB 未満です。AWS Glue ジョブは、`START_RANGE` と `END_RANGE` の 2 つのユーザー定義ジョブパラメータを受け入れます。これらのパラメータを使用して、ジョブを実行するたびに Amazon S3 で生成されるファイル名とファイル数を設定できます。このジョブの複数のインスタンス (たとえば、100 インスタンス) を並行実行できます。

ツール

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

ベストプラクティス

このパターンを実装する際には、次の AWS Glue のベストプラクティスを検討してください。

- 適切な AWS Glue ワーカータイプを使用してコストを削減します。ワーカータイプのさまざまなプロパティを理解し、CPU とメモリの要件に基づいてワークロードに適したワーカータイプを選択することをお勧めします。このパターンでは、DPU を最小限に抑えてコストを削減するた

めに、Python シェルジョブをジョブタイプとして使用することをお勧めします。詳細については、AWS Glue 開発者ガイドの「AWS Glue でのジョブの追加」を参照してください。

- 同時実行数の上限を適切に設定してジョブをスケールしてください。AWS Glue ジョブの最大同時実行数は、時間要件と必要なファイル数に基づいて設定することをお勧めします。
- 最初は、少数のファイルの生成から始めてください。AWS Glue ジョブを構築する際のコスト削減と時間の節約のため、最初は少数のファイル (1,000 など) から始めてください。これにより、トラブルシューティングが容易になります。少数のファイルの生成に成功すれば、より多くのファイル数に拡張できます。
- 最初にローカルで実行します。AWS Glue ジョブを構築する際のコストを削減し、時間を節約するには、ローカルで開発を開始し、コードをテストしてください。シェルと統合開発環境 (IDE) の両方で AWS Glue の抽出、変換、ロード (ETL) ジョブを作成するのに役立つ Docker コンテナの設定方法については、AWS ビッグデータブログの「コンテナを使用して AWS Glue ETL ジョブをローカルで開発する」を参照してください。

AWS Glue のその他のベストプラクティスについては、AWS Glue ドキュメントの「ベストプラクティス」を参照してください。

エピック

送信先 S3 バケットと IAM ロールの作成

タスク	説明	必要なスキル
ファイルを保存する S3 バケットを作成します。	「S3 バケット」とその中に「プレフィックス」を作成します。 **Note** このパターンでは、デモンストレーションの目的で `s3://{your-s3-bucket-name}/small-files/` の場所を使用します。	アプリ開発者

タスク	説明	必要なスキル
IAM ロールを作成して設定します。	AWS Glue ジョブが S3 バケットへの書き込みに使用できる IAM ロールを作成する必要があります。 1. 「IAM ロール」(たとえば、"AWSGlueServiceRole-smallfiles" というロール) を作成します。 2. ポリシーの信頼できるエンティティとして AWS Glue を選択します。 3. "AWSGlueServiceRole" という名前の「AWS マネージドポリシー」をロールにアタッチします。 4. 以下の設定に基づいて、"s3-small-file-access" というインラインポリシーまたは「カスタマー管理ポリシー」を作成します。"{bucket}" をバケット名に置き換えます。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [`	アプリ開発者

タスク	説明	必要なスキル

	<pre>"s3:GetObject",
	
	"s3:PutObject"
	],
	"Resource
	": [
	
	"arn:aws:s3:::{buc
	ket}/small-files/i
	nput/*"
	]
	}
	]
	}</pre> 5. "s3-small-file-acc ess" ポリシーをロールにアタッチします。	

同時実行を処理するように AWS Glue ジョブを作成して設定する

タスク	説明	必要なスキル
AWS Glue ジョブの作成	コンテンツを生成して S3 バケットに保存する AWS Glue ジョブを作成する必要があります。 「AWS Glue ジョブを作成」し、次のステップを完了してジョブを設定します。 1. AWS マネジメントコンソールにサインインし、「AWS Glue コンソール」を開きます。	アプリ開発者

タスク	説明	必要なスキル
	2. ナビゲーションペインの [データ統合と ETL] で、[ジョブ] を選択します。 3. 「ジョブの作成」セクションで、[Python シェルスクリプトエディタ] を選択します。 4. 「オプション」セクションで、[ボイラープレートコードで新しいスクリプトを作成] を選択し、[作成] を選択します。 5. [ジョブの詳細] を選択します。 6. [名前] には「create_small_files」と入力します。 7. [IAM ロール] で、前に作成したIAM ロールを選択します。 8. 「このジョブは実行」セクションで、[自分で作成する新しいスクリプト] を選択します。 9. [詳細プロパティ] を展開します。 10[最大同時実行数] には、デモンストレーション用に「100」と入力します。注: 最大同時実行数は、並行実行できるジョブのインスタンス数を定義します。 11[Save] を選択します。	

タスク	説明	必要なスキル

ジョブのコードを更新する。	1. 「AWS Glue コンソール」を開きます。
	2. ナビゲーションペインでジョブを選択します。
	3. 「あなたのジョブ」セクションで、前に作成したジョブを選択します。
	4. [スクリプト] タブを選択し、次のコードに基づいてスクリプトを更新します。BUCKET_NAME 、PREFIX、および text_str 変数を指定した値で更新します。
	`from awsglue.utils import getResolvedOptions`
	`import sys`
	`import boto3`
	`from random import randrange`
	
	`# Two arguments`
	`args = getResolvedOptions(sys.argv, ['START_RANGE', 'END_RANGE'])`
	
	`START_RANGE = int(args['START_RANGE'])`
	`END_RANGE = int(args['END_RANGE'])`
	
	`BUCKET_NAME = '{BUCKET_NAME}'`
		アプリ開発者

タスク	説明	必要なスキル

	PREFIX = 'small-files/input/'
	s3 = boto3.resource('s3')
	for x in range(START_RANGE, END_RANGE):
	    # generate file name
	    file_name = f"input_{x}.txt"
	    # generate text
	    text_str = str(randrange(100000))+","+str(randrange(100000))+", " + str(randrange(10000000)) + "," + str(randrange(10000))
	    # write in s3
	s3.Object(BUCKET_NAME, PREFIX + file_name).put(Body=text_str)
	5. [Save] を選択します。	

コマンドラインまたはコンソールから AWS Glue ジョブを実行する

タスク	説明	必要なスキル
コマンドラインから AWS Glue ジョブの実行を行います。	AWS Glue ジョブの実行を AWS CLI から実行するには、指定した値を使用して次のコマンドを実行します。 cmd:~$ aws glue start-job-run --job-name	アプリ開発者

タスク	説明	必要なスキル
	create_small_files --arguments '{"--STAR T_RANGE":"0","--EN D_RANGE":"1000000"}' cmd:~$ aws glue start- job-run --job-name create_small_files --arguments '{"--STAR T_RANGE":"1000000" ,"--END_RANGE":"20 00000"}' **Note** AWS マネジメントコンソールから AWS Glue ジョブを実行する手順については、このパターンの「AWS マネジメントコンソールで AWS Glue ジョブを実行する」を参照してください。 **Tip** 上記の例に示すように、異なるパラメータで一度に複数の実行を実行する場合は、AWS CLI を使用して AWS Glue ジョブを実行することをお勧めします。	

タスク	説明	必要なスキル

	特定の並列化係数を使用して定義済みの数のファイルを生成するために必要なすべての AWS CLI コマンドを生成するには、次の bash コードを (指定した値を使用して) 実行します。
	
	`# define parameters`
	`NUMBER_OF_FILES=`
	`10000000;`
	`PARALLELIZATION=50;`
	
	`# initialize`
	`_SB=0;`
	
	`# generate commands`
	`for i in $(seq 1`
	` $PARALLELIZATION);`
	`do`
	`  echo aws glue`
	` start-job-run --`
	`job-name create_sm`
	`all_files --argumen`
	`ts "'"'{"--START_RANG`
	`E":"'$(((NUMBER_OF`
	`_FILES/PARALLELIZA`
	`TION) * (i-1) +`
	` _SB))'","--END_RAN`
	`GE":"'$(((NUMBER_O`
	`F_FILES/PARALLELIZ`
	`ATION) * (i)))'"}'"'";`
	`  _SB=1;`
	`done`
	
	上記のスクリプトを使用する場合は、次の点を考慮してください。	

タスク	説明	必要なスキル
	• このスクリプトを使うと、小規模なファイルの呼び出しと生成を大規模に簡略化できます。 • `NUMBER_OF_FILES` と `PARALLELIZATION` を任意の値で更新します。 • 上のスクリプトは、実行する必要のあるコマンドのリストを出力します。これらの出力コマンドをコピーして、ターミナルで実行します。 • スクリプト内から直接コマンドを実行する場合は、11 行目の echo ステートメントを削除してください。 **Note** 上記のスクリプトからの出力例を確認するには、このパターンの「追加情報」セクションの「シェルスクリプト出力」を参照してください。	

タスク	説明	必要なスキル
AWS マネジメントコンソールで AWS Glue ジョブを実行します。	1. AWS マネジメントコンソールにサインインし、「AWS Glue コンソール」を開きます。 2. ナビゲーションペインの [データ統合と ETL] で、[ジョブ] を選択します。 3. 「あなたのジョブ」セクションで、あなたのジョブを選択します。 4. 「パラメーター (オプション)」セクションで、パラメーターを更新します。 5. [アクション]、[ジョブの実行] の順に選択します。 6. ステップ 3 ~ 5 を必要な回数繰り返します。たとえば、1,000 万個のファイルを作成するには、この処理を 10 回繰り返します。	アプリ開発者

タスク	説明	必要なスキル
AWS Glue ジョブのステータスを確認します。	1. 「AWS Glue コンソール」を開きます。 2. ナビゲーションペインでジョブを選択します。 3. 「あなたのジョブ」セクションで、前に作成したジョブ (つまり、`create_small_files`) を選択します。 4. ファイルの進行状況と生成状況を確認するには、「実行 ID」、「実行ステータス」などの列を確認してください。	アプリ開発者

関連リソース

リファレンス

- 「Registry of Open Data on AWS」
- 「分析用データセット」
- 「AWS のオープンデータ」
- 「AWS Glue でのジョブ追加」
- 「AWS Glue の使用開始」

ガイドとパターン

- 「AWS Glue のベストプラクティス」
- 「アプリケーションの負荷テスト」

追加情報

ベンチマークテスト

このパターンを使用して、ベンチマークテストの一環として、さまざまな並列化パラメーターを使用して 1,000 万個のファイルを生成しました。次のテーブルは、テストの出力を示しています。

並列化	1 回のジョブ実行で生成されるファイルの数	ジョブ所要時間	[Speed] (スピード)
10	1,000,000	6 時間、40 分	とても遅い
50	200,000 件の	80 分	中
100	100,000	40 分	高速

処理を速くしたい場合は、ジョブ設定で同時実行数を増やすことができます。要件に基づいてジョブ設定を簡単に調整できますが、AWS Glue サービスのクォータ制限があることに注意してください。詳細については、「AWS Glue のエンドポイントとクォータ」を参照してください。

シェルスクリプト出力

次の例は、このパターンの [コマンドラインから AWS Glue ジョブを実行する] ストーリーのシェルスクリプトの出力を示しています。

```
user@MUC-1234567890 MINGW64 ~
  $ # define parameters
  NUMBER_OF_FILES=10000000;
  PARALLELIZATION=50;
  # initialize
  _SB=0;

  # generate commands
  for i in $(seq 1 $PARALLELIZATION);
   do
         echo aws glue start-job-run --job-name create_small_files --arguments
 "'"'{"--START_RANGE":"'$(((NUMBER_OF_FILES/PARALLELIZATION)  (i-1) + SB))'","--
ENDRANGE":"'$(((NUMBER_OF_FILES/PARALLELIZATION)  (i)))'"}'"'";
         _SB=1;
   done
```

```
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"0","--END_RANGE":"200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"200001","--END_RANGE":"400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"400001","--END_RANGE":"600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"600001","--END_RANGE":"800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"800001","--END_RANGE":"1000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1000001","--END_RANGE":"1200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1200001","--END_RANGE":"1400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1400001","--END_RANGE":"1600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1600001","--END_RANGE":"1800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1800001","--END_RANGE":"2000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2000001","--END_RANGE":"2200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2200001","--END_RANGE":"2400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2400001","--END_RANGE":"2600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2600001","--END_RANGE":"2800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2800001","--END_RANGE":"3000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3000001","--END_RANGE":"3200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3200001","--END_RANGE":"3400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3400001","--END_RANGE":"3600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3600001","--END_RANGE":"3800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3800001","--END_RANGE":"4000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4000001","--END_RANGE":"4200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4200001","--END_RANGE":"4400000"}'
```

```
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4400001","--END_RANGE":"4600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4600001","--END_RANGE":"4800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4800001","--END_RANGE":"5000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5000001","--END_RANGE":"5200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5200001","--END_RANGE":"5400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5400001","--END_RANGE":"5600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5600001","--END_RANGE":"5800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5800001","--END_RANGE":"6000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6000001","--END_RANGE":"6200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6200001","--END_RANGE":"6400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6400001","--END_RANGE":"6600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6600001","--END_RANGE":"6800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6800001","--END_RANGE":"7000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7000001","--END_RANGE":"7200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7200001","--END_RANGE":"7400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7400001","--END_RANGE":"7600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7600001","--END_RANGE":"7800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7800001","--END_RANGE":"8000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8000001","--END_RANGE":"8200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8200001","--END_RANGE":"8400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8400001","--END_RANGE":"8600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8600001","--END_RANGE":"8800000"}'
```

```
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8800001","--END_RANGE":"9000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9000001","--END_RANGE":"9200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9200001","--END_RANGE":"9400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9400001","--END_RANGE":"9600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9600001","--END_RANGE":"9800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9800001","--END_RANGE":"10000000"}'

  user@MUC-1234567890 MINGW64 ~
```

よくある質問

同時実行または並列ジョブはいくつ使用すべきですか?

同時実行数と並行ジョブ数は、所要時間と必要なテストファイル数によって異なります。作成するファイルのサイズを確認することをお勧めします。まず、AWS Glue ジョブが希望する数のファイルを生成するのにかかる時間を確認します。次に、目標に合わせて適切な数の同時実行を行います。たとえば、100,000 ファイルの実行が完了するまでに 40 分かかるが、目標時間が 30 分であると仮定した場合、AWS Glue ジョブの同時実行設定を増やす必要があります。

このパターンではどのようなコンテンツを作成できますか?

区切り文字が異なるテキストファイル (PIPE、JSON、CSV など) など、あらゆる種類のコンテンツを作成できます。このパターンでは、Boto3 を使用してファイルに書き込み、そのファイルを S3 バケットに保存します。

S3 バケットにはどのレベルの IAM 権限が必要ですか?

S3 バケット内のオブジェクトへの `Write` アクセスを許可する ID ベースのポリシーが必要です。詳細については、Amazon S3 ドキュメントの「Amazon S3: S3 バケットのオブジェクトへの読み取りおよび書き込みのアクセス許可」を参照してください。

AWS IoT Greengrass を使用して IoT データをコスト効率よく直接 Amazon S3 に取り込む

セバスチャン・ヴィヴィアーニ (AWS) とリズワン・サイード (AWS) が制作

概要

このパターンは、AWS IoT Greengrass バージョン 2 デバイスを使用して、コスト効率よくモノのインターネット (IoT) データを Amazon Simple Storage Service (Amazon S3) バケットに直接取り込む方法を示しています。デバイスは、IoT データを読み取り、データを永続ストレージ (つまり、ローカルディスクまたはボリューム) に保存するカスタムコンポーネントを実行します。次に、デバイスは IoT データを Apache Parquet ファイルに圧縮し、そのデータを定期的に S3 バケットにアップロードします。

取り込むIoTデータの量と速度は、エッジハードウェアの機能とネットワーク帯域幅によってのみ制限されます。Amazon Athena を使用すれば、取り込んだデータをコスト効率よく分析することができます。Athena は、圧縮された Apache Parquet ファイルと「Amazon Managed Grafana」によるデータの視覚化をサポートしています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 「AWS IoT Greengrass Version 2」上で動作し、センサーからデータを収集する「エッジゲートウェイ」(データソースとデータ収集プロセスはこのパターンの範囲外ですが、ほぼすべての種類のセンサーデータを使用できます。 このパターンでは、センサーを備えたローカル「MQTT」ブローカー、またはデータをローカルに公開するゲートウェイを使用します。)
- AWS IoT Greengrass 「コンポーネント」、「ロール」、「SDK の依存関係」
- S3 バケットにデータをアップロードする「ストリームマネージャーコンポーネント」
- API を実行するには「AWS SDK for Java」の AWS SDK、「JavaScript 用の AWS SDK」、または 「AWS SDK for Python (Boto3)」

制約事項

- このパターンのデータは S3 バケットにリアルタイムでアップロードされません。遅延期間があり、遅延期間を設定できます。データは一時的にエッジデバイスにバッファされ、期間が終了するとアップロードされます。
- SDK は、Java、Node.js、Python で使用できます。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon S3
- AWS IoT Greengrass
- MQTT ブローカー
- ストリームマネージャーコンポーネント

ターゲットアーキテクチャ

次の図は、IoT センサーデータを取り込み、そのデータを S3 バケットに保存するように設計されたアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. 複数のセンサー (温度やバルブなど) の更新がローカルの MQTT ブローカーに公開されます。
2. これらのセンサーにサブスクライブされている Parquet ファイルコンプレッサーは、トピックを更新し、更新を受信します。
3. Parquet ファイルコンプレッサーは更新をローカルに保存します。
4. 期間が経過すると、保存されたファイルは Parquet ファイルに圧縮され、ストリームマネージャーに渡され、指定された S3 バケットにアップロードされます。
5. ストリームマネージャーは Parquet ファイルを S3 バケットにアップロードします。

> **Note**
>
> ストリームマネージャー (`StreamManager`) はマネージドコンポーネントです。Amazon S3 にデータをエクスポートする方法の例については、AWS IoT Greengrass ドキュメントの「ストリームマネージャー」を参照してください。ローカルの MQTT ブローカーをコンポー

ネントとして使用することも、「Eclipse Mosquitto」のような別のブローカーを使用することもできます。

ツール

AWS ツール

- Amazon Athena はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接、簡単に分析します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS IoT Greengrass は、デバイス上で IoT アプリケーションを構築、デプロイ、管理するのに役立つオープンソースの IoT エッジランタイムおよびクラウドサービスです。

その他のツール

- 「Apache Parquet」は、ストレージとデータの取得を目的として設計されたオープンソースの列指向データファイル形式です。
- MQTT (メッセージキューテレメトリートランスポート) は、制約のあるデバイス向けに設計された軽量のメッセージングプロトコルです。

ベストプラクティス

アップロードされたデータには適切なパーティション形式を使用してください。

S3 バケットのルートプレフィックス名 (たとえば、"myAwesomeDataSet/"や"dataFromSource") には特定の要件はありませんが、データセットの目的がわかりやすいように、わかりやすいパーティションとプレフィックスを使用することをお勧めします。

また、クエリがデータセットで最適に実行されるように、Amazon S3 では適切なパーティション分割を使用することをお勧めします。次の例では、各 Athena クエリでスキャンされるデータ量が最適化されるように、データを HIVE 形式で分割しています。これにより、パフォーマンスを向上させ、コストを削減できます。

```
s3://<ingestionBucket>/<rootPrefix>/year=YY/month=MM/day=DD/
HHMM_<suffix>.parquet
```

エピック

環境をセットアップします。

タスク	説明	必要なスキル
S3 バケットを作成する。	1. S3 バケットを作成するか、既存のバケットを使用します。 2. IoT データを取り込む S3 バケットにわかりやすい「プレフィックス」を作成します (例：`s3:\\<bucket>\<prefix>`)。 3. 後で使用するためにプレフィックスを記録します。	アプリ開発者
IAM を追加して、S3 バケットへの許可を提供します。	以前に作成した S3 バケットとプレフィックスへの書き込みアクセスをユーザーに許可するには、次の IAM ポリシーを AWS IoT Greengrass ロールに追加します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "S3DataUpload", "Effect": "Allow", "Action": [ "s3:List*", "s3:Put*" ],`	アプリ開発者

タスク	説明	必要なスキル
	"Resource": ["arn:aws:s3:::<ingestionBucket>", "arn:aws:s3:::<ingestionBucket>/<prefix>/*"] }] } 詳細については、Aurora ドキュメントの「Amazon S3 リソースにアクセスするための IAM ポリシーの作成」を参照してください。 次に、S3 バケットのリソースポリシー (必要な場合) を更新して、正しい AWS 「プリンシパル」による書き込みアクセスを許可します。	

AWS IoT Greengrass コンポーネントを開発して構築します。

タスク	説明	必要なスキル
コンポーネントのレシピを更新します。	「デプロイを作成」するときは、次の例に基づいて「コンポーネント設定を更新」します。 { "region": "<region>",	アプリ開発者

タスク	説明	必要なスキル
	<pre>"parquet_period": <period>, "s3_bucket": "<s3Bucket>", "s3_key_prefix": "<s3prefix>" }</pre> `<region>`をAWSリージョン、`<period>`を周期間隔、`<s3Bucket>` をS3バケット、`<s3prefix>` をプレフィックスに置き換えます。	

タスク	説明	必要なスキル
コンポーネントを作成します。	次のいずれかを行います： • コンポーネントを作成します。 • CI/CD パイプライン (存在する場合) にコンポーネントを追加します。アーティファクトは必ず、アーティファクトリポジトリから AWS IoT Greengrass アーティファクトバケットにコピーしてください。次に、AWS IoT Greengrass コンポーネントを作成または更新します。 • **Note** MQTT ブローカーをコンポーネントとして追加するか、後で手動で追加します。: この決定は、ブローカーで使用できる認証スキームに影響します。ブローカーを手動で追加すると、そのブローカーは AWS IoT Greengrass から切り離され、サポートされているすべてのブローカーの認証スキームが有効になります。AWS	アプリ開発者

タスク	説明	必要なスキル
	が提供するブローカーコンポーネントには、認証スキームが事前定義されています。詳細については、「MQTT 3.1.1 ブローカー (Moquette)」および「MQTT 5 ブローカー (EMQX)」を参照してください。	
MQTT クライアントを更新してください。	このサンプルコードでは、コンポーネントはブローカーにローカルに接続するため、認証を使用していません。シナリオが異なる場合は、必要に応じて MQTT クライアントセクションを更新してください。さらに、以下の作業を行います。 1. サブスクリプションの MQTT トピックを更新します。 2. 各ソースからのメッセージは異なる場合があるため、必要に応じて MQTT メッセージパーサーを更新してください。	アプリ開発者

AWS IoT Greengrass バージョン 2 コアデバイスにコンポーネントを追加します。

タスク	説明	必要なスキル
コアデバイスのデプロイを更新します。	AWS IoT Greengrass バージョン 2 コアデバイスのデプロイがすでに存在する場合は、「デプロイを修正」します。デプロイが存在しない場合は、「新しいデプロイを作成」します。 コンポーネントに正しい名前を付けるには、以下に基づいて新しいコンポーネントの「ログマネージャー設定を更新」します (必要に応じて) 。 `{ "logsUploaderConfiguration": { "systemLogsConfiguration": { ... }, "componentLogsConfigurationMap": { "<com.iot.ingest.parquet>": { "minimumLogLevel": "INFO", "diskSpaceLimit": "20", "diskSpaceLimitUnit": "MB", "deleteLogFileAfterCloudUpload": "false" } ... }`	アプリ開発者

タスク	説明	必要なスキル
	<pre> }, "periodicUploadInt ervalSec": "300" }</pre> 最後に、お使いの AWS IoT Greengrass コアデバイスのデプロイのリビジョンを完了します。	

S3 バケットへのデータの取り込みを確認します。

タスク	説明	必要なスキル
AWS IoT Greengrass ボリュームのログを確認してください。	以下を確認します。 • MQTT クライアントはローカル MQTT ブローカーに正常に接続されました。 • MQTT クライアントは正しいトピックにサブスクライブされています。 • MQTT トピックに関するセンサー更新メッセージがブローカーに届いています。 • Parquet の圧縮は定期的に行われます。	アプリ開発者
S3 バケットを確認します。	S3 バケットにデータがアップロードされているかどうかを確認します。各期間にアップロードされているファイルを確認できます。	アプリ開発者

タスク	説明	必要なスキル
	次のセクションでデータをクエリすることで、データが S3 バケットにアップロードされたかどうかを確認することもできます。	

Athena からのクエリを設定する

タスク	説明	必要なスキル
データベースとテーブルを作成します。	1. AWS Glue データベースを作成します (必要な場合)。 2. AWS Glue でテーブルを「手動」で作成するか、AWS Glue で「クローラー」を実行して作成します。	アプリ開発者
Athena にデータへのアクセスを許可します。	1. Athena が S3 バケットにアクセスすることを許可する権限を更新します。詳細については、Athenaドキュメントの「AWS Glue データカタログのデータベースとテーブルへのきめ細かなアクセス」を参照してください。 2. データベース内のテーブルをクエリします。	アプリ開発者

トラブルシューティング

問題	ソリューション
MQTT クライアントは接続に失敗します。	• MQTT ブローカーの権限を検証してください。AWS の MQTT ブローカーをお持ちの場合は、「MQTT 3.1.1 ブローカー (モケット)」と「MQTT 5 ブローカー (EMQX)」を参照してください。 • MQTT クライアントで認証情報を検証します。AWS の MQTT ブローカーをお持ちの場合は、「MQTT 3.1.1 ブローカー (モケット)」と「MQTT 5 ブローカー (EMQX)」を参照してください。
MQTT クライアントがサブスクライブに失敗する	MQTT ブローカーの権限を検証してください。AWS の MQTT ブローカーをお持ちの場合は、「MQTT 3.1.1 ブローカー (モケット)」と「MQTT 5 ブローカー (EMQX)」を参照してください。
パーケットファイルは作成されません。	• MQTT トピックが正しいことを確認してください。 • センサーからの MQTT メッセージが正しい形式であることを確認します。
オブジェクトは S3 バケットにアップロードされません。	• インターネット接続とエンドポイント接続があることを確認します。 • S3 バケットのリソースポリシーが正しいことを確認します。 • AWS IoT Greengrass バージョン 2 コアデバイスロールのアクセス権限を確認してください。

関連リソース

- DataFrame (Pandas ドキュメンテーション)
- Apache パーケットドキュメンテーション (パーケットドキュメント)
- AWS IoT Greengrass コンポーネントの開発 (AWS IoT Greengrass 開発者ガイド、バージョン 2)
- AWS IoT Greengrass コンポーネントをデバイスにデプロイ (AWS IoT Greengrass 開発者ガイド、バージョン 2)
- ローカルの IoT デバイスとのやり取り (AWS IoT Greengrass 開発者ガイド、バージョン 2)
- MQTT 3.1.1 ブローカー (Moquette) (AWS IoT Greengrass 開発者ガイド、バージョン 2)
- MQTT 5 ブローカー (EMQX) (AWS IoT Greengrass 開発者ガイド、バージョン 2)

追加情報

コスト分析

次のコスト分析シナリオは、このパターンで取り上げられているデータ取り込みアプローチが AWS クラウドのデータ取り込みコストにどのように影響するかを示しています。このシナリオの料金例は、公開時の価格に基づいています。料金は変更される可能性があります。さらに、費用は AWS リージョン、AWS Service Quotas、およびクラウド環境に関連するその他の要因によって異なる場合があります。

入力信号セット

この分析では、IoT の取り込みコストを他の利用可能な代替手段と比較するための基礎として、以下の入力信号セットを使用します。

シグナル数。	[Frequency] (頻度)	1 信号あたりのデータ
125	25 ヘルツ	8 バイト

このシナリオでは、システムは 125 個の信号を受信します。各信号は 8 バイトで、40 ミリ秒 (25 Hz) ごとに発生します。これらの信号は、個別に受信することも、共通のペイロードにまとめて送信することもできます。これらの信号は、必要に応じて分割してパックすることができます。レイテンシーも決定できます。レイテンシーは、データの受信、蓄積、および取り込みにかかる時間で構成されます。

比較のため、このシナリオの取り込み操作は us-east-1 AWS リージョンをベースにしています。コスト比較は AWS サービスにのみ適用されます。ハードウェアや接続などの他のコストは、分析には考慮されません。

コスト比較

次の表は、各取り込み方法の月額費用を米ドル (USD) で示しています。

方法	月額コスト
AWS IoT SiteWise*	331.77 米ドル
データ処理パック付きの AWS IoT SiteWise Edge (すべてのデータをエッジに保持)	200 米ドル
未加工データにアクセスするための AWS IoT Core と Amazon S3 のルール	84.54 米ドル
エッジでの Parquet ファイルの圧縮と Amazon S3 へのアップロード	0.5米ドル

*サービスクォータを満たすには、データをダウンサンプリングする必要があります。つまり、この方法ではデータの一部が失われるということです。

代替方法

このセクションでは、以下の代替方法の等価コストを示します。

- AWS IoT SiteWise — 各シグナルは個別のメッセージでアップロードする必要があります。したがって、1 か月あたりのメッセージの総数は 125×25×3600×24×30、つまり 1 か月あたり 81 億メッセージになります。ただし、AWS IoT SiteWise は、プロパティごとに 1 秒あたり 10 データポイントしか処理できません。データが 10 Hz にダウンサンプリングされると仮定すると、1 か月あたりのメッセージ数は 125×10×3600×24×30、つまり 32.4 億に減少します。測定値を 10 件グループ (100 万メッセージあたり 1 USD) にまとめるパブリッシャーコンポーネントを使用すると、1 か月あたり 324 USD の月額料金が発生します。各メッセージが 8 バイト (1 Kb/125) であると仮定すると、25.92 GB のデータストレージになります。これにより、1 か月あたり 7.77 USD の月額コストが加算されます。初月の総費用は 331.77 米ドルで、毎月 7.77 米ドルずつ増加します。

- エッジで完全に処理されたすべてのモデルと信号を含む、データ処理パック付きのAWS IoT SiteWise Edge (つまり、クラウドインジェストなし) — コストを削減し、エッジで計算されるすべてのモデルを設定するための代替手段としてデータ処理パックを使用できます。これは、実際の計算を行わなくても、保存と視覚化のためだけに機能します。この場合、エッジゲートウェイには強力なハードウェアを使用する必要があります。1 か月あたり 200 米ドルの固定費がかかります。
- MQTT と Amazon S3 に生データを保存するための IoT ルールによる AWS IoT Core への直接取り込み — すべてのシグナルが共通のペイロードでパブリッシュされると仮定すると、AWS IoT Core にパブリッシュされるメッセージの総数は 25×3600×24×30、つまり 1 か月あたり 6,480 万件になります。100 万メッセージあたり 1 米ドルとすると、1 か月あたり 64.8 米ドルの月額料金になります。100 万回のルール有効化あたり 0.15 USD で、メッセージごとに 1 つのルールを設定すると、1 か月あたり 19.44 USD の月額料金が加算されます。Amazon S3 のストレージ 1 GB あたり 0.023 USD のコストで、1 か月あたり 1.5 USD が追加されます (新しいデータを反映して毎月増加しています)。最初の 1 か月の総コストは 84.54 米ドルで、毎月 1.5 米ドルずつ増加します。
- Parquetファイルの端でデータを圧縮してAmazon S3 にアップロードする（提案方法）— 圧縮率はデータの種類によって異なります。同じ産業用データを MQTT でテストした場合、1 か月分の出力データの合計は 1.2 Gb になります。これには 1 か月あたり 0.03 米ドルかかります。他のベンチマークで説明されている圧縮率 (ランダムデータを使用) は、約 66% (最悪のシナリオに近い) です。データの合計は 21 Gb で、1 か月あたり 0.5 米ドルかかります。

Parquetファイルジェネレーター

次のコード例は、Python で記述された Parquet ファイルジェネレーターの構造を示しています。このコード例は説明のみを目的としており、ご使用の環境に貼り付けても動作しません。

```
import queue
import paho.mqtt.client as mqtt
import pandas as pd

#queue for decoupling the MQTT thread
messageQueue = queue.Queue()
client = mqtt.Client()
streammanager = StreamManagerClient()

def feederListener(topic, message):
    payload = {
        "topic" : topic,
        "payload" : message,
    }
```

```
    messageQueue.put_nowait(payload)

def on_connect(client_instance, userdata, flags, rc):
    client.subscribe("#",qos=0)

def on_message(client, userdata, message):
    feederListener(topic=str(message.topic),
 message=str(message.payload.decode("utf-8")))


filename = "tempfile.parquet"
streamname = "mystream"
destination_bucket= "amzn-s3-demo-bucket"
keyname="mykey"
period= 60

client.on_connect = on_connect
client.on_message = on_message
streammanager.create_message_stream(
            MessageStreamDefinition(name=streamname,
 strategy_on_full=StrategyOnFull.OverwriteOldestData)
        )


while True:
   try:
       message = messageQueue.get(timeout=myArgs.mqtt_timeout)
   except (queue.Empty):
       logger.warning("MQTT message reception timed out")

   currentTimestamp = getCurrentTime()
   if currentTimestamp >= nextUploadTimestamp:
       df = pd.DataFrame.from_dict(accumulator)
       df.to_parquet(filename)
       s3_export_task_definition = S3ExportTaskDefinition(input_url=filename,
 bucket=destination_bucket, key=key_name)
       streammanager.append_message(streamname,
 Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
       accumulator = {}
       nextUploadTimestamp += period
   else:
        accumulator.append(message)
```

Lambda 関数を使用して一時的な EMR クラスターで Spark ジョブを起動する

作成者:Dhrubajyoti Mukherjee (AWS)

概要

このパターンでは、Amazon EMR RunJobFlow API アクションを使用して一時的なクラスターを起動し、Lambda 関数から Spark ジョブを実行します。一時的な EMR クラスターは、ジョブが完了したり、エラーが発生したりするとすぐに終了するように設計されています。一時的なクラスターは計算時間中にのみ実行されるため、コストを節約でき、クラウド環境でのスケーラビリティと柔軟性も高まります。

一時的な EMR クラスターは、Boto3 API と Lambda 関数内の Python プログラミング言語を使用して起動されます。Python で記述された Lambda 関数を使用すると、必要なときにクラスターを柔軟に開始できます。

バッチ計算と出力の例を示すために、このパターンでは、Lambda 関数から EMR クラスター内の Spark ジョブを起動し、架空の会社のサンプル売上データに対してバッチ計算を実行します。Spark ジョブの出力は、Amazon Simple Storage Service (Amazon S3) のカンマ区切り値 (CSV) ファイルになります。入力データファイル、Spark.jarファイル、コードスニペット、仮想プライベートクラウド（VPC）用の AWS CloudFormation テンプレート、計算を実行するための AWS Identity and Access Management（IAM）ロールが添付ファイルとして提供されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント

制約事項

- 一度にコードから開始できる Spark ジョブは 1 つだけです。

製品バージョン

- Amazon EMR 6.0.0 でテスト済み

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EMR
- AWS Lambda
- Amazon S3
- Apache Spark

ターゲットアーキテクチャ

自動化とスケール

Spark-EMR のバッチ計算を自動化するには、次のいずれかのオプションを使用できます。

- cron スケジュールで Lambda 関数を開始できる Amazon EventBridge ルールを実装する。詳細については、「チュートリアル: EventBridge を使用して AWS Lambda 関数をスケジュールする」を参照してください。
- ファイルの到着時に Lambda 関数を開始するように Amazon S3 イベント通知を構成する。
- Event bodyと Lambda 関数を使用して、入力パラメータを AWS Lambda 関数に渡します。

ツール

サービス

- Amazon EMR は、 でビッグデータフレームワーク の実行を簡素化して、大量のデータを処理および分析するマネージドクラスタープラットフォームです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

その他のツール

- Apache Spark は、大規模データ処理のための複数言語分析エンジンです。

エピック

Amazon EMR とLambdaの IAM ロール、および VPC を作成する

タスク	説明	必要なスキル
IAM ロールと VPC を作成します。	AWS Lambda と Amazon EMR の IAM ロールと VPC がすでにある場合は、このステップはスキップできます。コードを実行するには、EMR クラスターと Lambda 関数の両方に IAM ロールが必要です。EMR クラスターには、パブリックサブネットを使用する VPC または NAT ゲートウェイを使用するプライベートサブネットも必要です。すべての IAM ロールと VPC を自動的に作成するには、添付された AWS CloudFormation テンプレートをそのままデプロイするか、[追加情報] セクションの指定に従ってロールと VPC を手動で作成できます。	クラウドアーキテクト
AWS CloudFormation テンプレートの出力キーを書き留めます。	CloudFormation テンプレートが正常にデプロイされたら、AWS CloudFormation コンソールの [出力] タブに移動します。5 つの出力キーを書き留めます。 • `S3Bucket`	クラウドアーキテクト

タスク	説明	必要なスキル
	• LambdaExecutionRole • ServiceRole • JobFlowRole • Ec2SubnetId Lambda 関数の作成時に、これらのキーの値を使用します。	

Spark .jar ファイルのアップロード

タスク	説明	必要なスキル
Spark .jar ファイルをアップロードします。	AWS CloudFormation スタックが作成した S3 バケットに Spark .jar ファイルをアップロードします。バケットの名前は出力キー S3Bucket と同じです。	AWS 全般

EMR クラスターを起動する Lambda 関数の作成

タスク	説明	必要なスキル
Lambda 関数を作成する。	Lambda コンソールで、実行ロールを含む Python 3.9 以降の Lambda 関数を作成します。実行ロールポリシーでは、Lambda が EMR クラスターを起動できるようにする必要があります。(添付の	データエンジニア、クラウドエンジニア

タスク	説明	必要なスキル
	AWS CloudFormation テンプレートを参照してください。)	
コードをコピーして貼り付けます。	`lambda_function.py` ファイル内のコードを、このパターンの [追加情報] セクションのコードに置き換える。	データエンジニア、クラウドエンジニア
コード内のパラメータの変更。	コード内のコメントに従って、AWS アカウントに合わせてパラメータ値を変更します。	データエンジニア、クラウドエンジニア
関数を起動してクラスターを開始します。	関数を起動し、指定された Spark .jar ファイルを使用して一時的な EMR クラスターの作成を開始します。Spark ジョブが実行され、ジョブが完了すると自動的に終了します。	データエンジニア、クラウドエンジニア
EMR クラスターのステータスを確認します。	EMR クラスターが開始されると、Amazon EMR コンソールの [クラスター] タブに表示されます。クラスターの起動中またはジョブの実行中に発生したエラーは、それに応じて確認できます。	データエンジニア、クラウドエンジニア

サンプルデモをセットアップして実行する

タスク	説明	必要なスキル
Spark .jar ファイルをアップロードします。	[添付ファイル] セクションから Spark .jar ファイルをダウンロードし、S3 バケットにアップロードします。	データエンジニア、クラウドエンジニア
入力データをアップロードします。	添付された `fake_sales_data.csv` ファイルを S3 バケットにアップロードします。	データエンジニア、クラウドエンジニア
Lambda コードを貼り付けて、パラメータを変更する。	[ツール] セクションからコードをコピーし、そのコードを Lambda 関数に貼り付け、コード `lambda_function.py` ファイルを置き換えます。アカウントに合わせてパラメータ値を変更します。	データエンジニア、クラウドエンジニア
関数を起動し、出力を確認する。	Lambda 関数は、指定された Spark ジョブでクラスターを開始すると、S3 バケットに .csv ファイルが生成されます。	データエンジニア、クラウドエンジニア

関連リソース

- Spark の構築
- Apache Spark と Amazon EMR
- Boto3 Docs run_job_flow ドキュメント
- Apache Spark の情報とドキュメント

追加情報

コード

```
"""
Copy paste the following code in your Lambda function. Make sure to change the
 following key parameters for the API as per your account

-Name (Name of Spark cluster)
-LogUri (S3 bucket to store EMR logs)
-Ec2SubnetId (The subnet to launch the cluster into)
-JobFlowRole (Service role for EC2)
-ServiceRole (Service role for Amazon EMR)

The following parameters are additional parameters for the Spark job itself. Change the
 bucket name and prefix for the Spark job (located at the bottom).

-s3://your-bucket-name/prefix/lambda-emr/SparkProfitCalc.jar (Spark jar file)
-s3://your-bucket-name/prefix/fake_sales_data.csv (Input data file in S3)
-s3://your-bucket-name/prefix/outputs/report_1/ (Output location in S3)
"""
import boto3

client = boto3.client('emr')


def lambda_handler(event, context):
    response = client.run_job_flow(
        Name='spark_job_cluster',
        LogUri='s3://your-bucket-name/prefix/logs',
        ReleaseLabel='emr-6.0.0',
        Instances={
            'MasterInstanceType': 'm5.xlarge',
            'SlaveInstanceType': 'm5.large',
            'InstanceCount': 1,
            'KeepJobFlowAliveWhenNoSteps': False,
            'TerminationProtected': False,
            'Ec2SubnetId': 'subnet-XXXXXXXXXXXXX'
        },
        Applications=[{'Name': 'Spark'}],
        Configurations=[
            {'Classification': 'spark-hive-site',
             'Properties': {
```

```
                'hive.metastore.client.factory.class':
'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'}
            }
        ],
        VisibleToAllUsers=True,
        JobFlowRole='EMRLambda-EMREC2InstanceProfile-XXXXXXXXX',
        ServiceRole='EMRLambda-EMRRole-XXXXXXXXX',
        Steps=[
            {
                'Name': 'flow-log-analysis',
                'ActionOnFailure': 'TERMINATE_CLUSTER',
                'HadoopJarStep': {
                    'Jar': 'command-runner.jar',
                    'Args': [
                        'spark-submit',
                        '--deploy-mode', 'cluster',
                        '--executor-memory', '6G',
                        '--num-executors', '1',
                        '--executor-cores', '2',
                        '--class', 'com.aws.emr.ProfitCalc',
                        's3://your-bucket-name/prefix/lambda-emr/SparkProfitCalc.jar',
                        's3://your-bucket-name/prefix/fake_sales_data.csv',
                        's3://your-bucket-name/prefix/outputs/report_1/'
                    ]
                }
            }
        ]
    )
```

IAM ロールと VPC の作成

Lambda 関数で EMR クラスターを起動するには、VPC と IAM ロールが必要です。VPC と IAM ロールは、このパターンの添付ファイルセクションにある AWS CloudFormation テンプレートを使用してセットアップできます。または、以下のリンクを使用して手動で作成することもできます。

Lambda と Amazon EMR を実行するには、次の IAM ロールが必要です。

Lambda 実行ロール

AWS Lambda 関数の実行ロールでは、AWS サービスおよびリソースにアクセスするためのアクセス許可を付与します。

Amazon EMR のサービスロール

Amazon EMR ロールは、Amazon EMR がリソースをプロビジョニングし、クラスター内で実行されている Amazon Elastic Compute Cloud (Amazon EC2) のコンテキストでは実行されないサービスレベルのタスクを実行するときに Amazon EMR に対して許可されるアクションを定義します。たとえば、このサービスロールを使用してクラスターの起動時に EC2 インスタンスをプロビジョニングします。

EC2 インスタンスの サービスロール

クラスター EC2 インスタンスのサービスロール (Amazon EMR の EC2 インスタンスプロファイルとも呼ばれます) は、インスタンスの起動時に Amazon EMR クラスター内のすべての EC2 インスタンスに割り当てられる特殊なサービスロールです。Apache Hadoop 上で実行されるアプリケーションプロセスは、このロールを引き受けることで、AWS の他のサービスとやり取りするアクセス許可を取得します。

VPC とサブネットの作成

VPC コンソールを使用して VPC を作成できます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Glue を使用して Apache Cassandra ワークロードを Amazon Keyspaces に移行する

作成者: Nikolai Kolesnikov (AWS)、Karthiga Priya Chandran (AWS)、Samir Patel (AWS)

概要

このパターンは、AWS Glue で CQLReplicator を使用して、既存の Apache Cassandra ワークロードを Amazon Keyspaces (Apache Cassandra 用) に移行する方法を示しています。AWS Glue で CQLReplicator を使用すると、ワークロードの移行に伴うレプリケーションの遅延を数分に抑えることができます。Amazon Simple Storage Service (Amazon S3) バケットを使用して、Apache Parquet ファイル、設定ファイル、スクリプトなどの移行に必要なデータを保存する方法についても学びます。このパターンは、Cassandra ワークロードが Virtual Private Cloud (VPC) の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされていることを前提としています。

前提条件と制限

前提条件

- ソーステーブルを含む Cassandra クラスター
- ワークロードをレプリケートするための Amazon Keyspaces ターゲットテーブル
- データの増分変更を含む中間 Parquet ファイルを保存する S3 バケット
- ジョブ設定ファイルとスクリプトを保存する S3 バケット

制限

- AWS Glue の CQLReplicator では、Cassandra ワークロードのデータ処理ユニット (DPUs) をプロビジョニングするために少し時間がかかります。Cassandra クラスターと Amazon キースペース内のターゲットキースペースおよびテーブル間のレプリケーションラグは、ほんの数分しか続かない可能性があります。

アーキテクチャ

ソーステクノロジースタック

- Apache Cassandra
- DataStax サーバー

- ScyllaDB

ターゲットテクノロジースタック

- Amazon Keyspaces

移行アーキテクチャ

次の図は、Cassandra クラスターが EC2 インスタンスでホストされ、3 つのアベイラビリティーゾーンにまたがるアーキテクチャの例を示しています。Cassandra ノードは、プライベートサブネットでホストされます。

この図表は、次のワークフローを示しています：

1. カスタムサービスロールは、Amazon Keyspaces と S3 バケットへのアクセスを提供します。
2. AWS Glue ジョブは、S3 バケット内のジョブ設定とスクリプトを読み取ります。
3. AWS Glue ジョブはポート 9042 を介して接続し、Cassandra クラスターからデータを読み取ります。
4. AWS Glue ジョブはポート 9142 を介して接続し、Amazon Keyspaces にデータを書き込みます。

ツール

AWS サービスとツール

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS CloudShell」はブラウザベースのシェルで、AWS コマンドラインインターフェイス (AWS CLI) とプリインストールされたさまざまな開発ツールを使用して、AWS のサービスを管理できます。
- AWS Glue は完全マネージド型の ETL サービスで、データストアとデータストリーム間でデータを確実に分類、クリーニング、強化、移動できます。
- Amazon Keyspaces (Apache Cassandra に向け) は、AWS クラウドの Cassandra ワークロードの移行、実行、スケーリングを支援するマネージド型データベースサービスです。

コード

このパターンのコードは、GitHub CQLReplicator リポジトリで入手できます。

ベストプラクティス

- 移行に必要な AWS Glue リソースを決定するには、ソース Cassandra テーブルの行数を見積もります。たとえば、84 GB のディスクを持つ 0.25 DPU (2 vCPUs4 GB のメモリ) あたり 250 K の行です。
- CQLReplicator を実行する前に Amazon Keyspaces テーブルを事前ウォーミングします。たとえば、8 つの CQLReplicator タイル (AWS Glue ジョブ) は 1 秒あたり最大 22 K WCUs書き込むことができるため、ターゲットは 1 秒あたり最大 25～30 K WCUs を事前にウォーミングする必要があります。
- AWS Glue コンポーネント間の通信を有効にするには、セキュリティグループ内のすべての TCP ポートに対して自己参照インバウンドルールを使用します。
- 増分トラフィック戦略を使用して、移行ワークロードを経時的に分散します。

エピック

CQLReplicator をデプロイする

タスク	説明	必要なスキル
ターゲットのキースペースとテーブルを作成します。	1. Amazon Keyspaces でキースペースとテーブルを作成します。 書き込み容量の詳細については、このパターンの「追加情報」セクションの「単位計算の書き込み」を参照してください。 Cassandra Query Language (CQL) を使用してキースペースを作成することもできます。詳細に	アプリ所有者、AWS 管理者、DBA、アプリ開発者

タスク	説明	必要なスキル
	ついては、このパターンの「追加情報」セクションの「CQL を使用してキースペースを作成する」を参照してください。 **Note** テーブルを作成したら、不要な料金が発生しないように、テーブルをオンデマンドキャパシティモードに切り替えることを検討してください。 2. スループットモードに更新するには、次のスクリプトを実行します。 `ALTER TABLE target_keyspace.target_table WITH CUSTOM_PROPERTIES = { 'capacity_mode': { 'throughput_mode': 'PAY_PER_REQUEST'} }`	

タスク	説明	必要なスキル
カサンドラに接続するようにカサンドラドライバーを設定します。	次の設定スクリプトを使用します。 `Datastax-java-driver { basic.request.consistency = “LOCAL_QUORUM” basic.contact-points = [“127.0.0.1:9042”] advanced.reconnect-on-init = true basic.load-balancing-policy { local-datacenter = “datacenter1” } advanced.auth-provider = { class = PlainTextAuthProvider username = “user-at-sample” password = “S@MPLE=PASSWORD=” } }` **Note** 前述のスクリプトでは、Spark Cassandra コネクタを使用します。詳細については、Cassandra のリファレンス設定を参照してください。	DBA

タスク	説明	必要なスキル
Amazon Keyspaces に接続するように Cassandra ドライバーを設定します。	次の設定スクリプトを使用します。 `datastax-java-driver { basic { load-balancing-policy { local-datacenter = us-west-2 } contact-points = ["cassandra.us-west-2.amazonaws.com:9142"] request { page-size = 2500 timeout = 360 seconds consistency = LOCAL_QUORUM } } advanced { control-connection { timeout = 360 seconds } session-leak.threshold = 6 connection { connect-timeout = 360 seconds init-query-timeout = 360 seconds warn-on-init-error = false } auth-provider = { class = software.aws.mcs.auth.SigV4AuthProvider`	DBA

タスク	説明	必要なスキル

	<pre> aws-region = us-
	west-2
	 }
	
	 ssl-engine-factory {
	  class = DefaultSs
	lEngineFactory
	  }
	  }
	}</pre>
	Note
	前述のスクリプトでは、Spark Cassandra コネクタを使用します。詳細については、Cassandra のリファレンス設定を参照してください。	

タスク	説明	必要なスキル
AWS Glue ジョブ用に IAM ロールを作成します。	AWS Glue `glue-cassandra-migration` を信頼されたエンティティとして という名前の新しい AWS サービスロールを作成します。AWS Glue **Note** は、S3 バケットと Amazon Keyspaces への読み取りおよび書き込みアクセスを提供する`glue-cassandra-migration` 必要があります。S3 バケットには、.jar ファイル、Amazon Keyspaces と Cassandra の設定ファイル、および中間 Parquet ファイルが含まれています。たとえば、、`AWSGlueServiceRole` `AmazonS3FullAccess` 、 `AmazonKeyspacesFullAccess` 管理ポリシーが含まれます。	AWS DevOps

タスク	説明	必要なスキル

AWS CloudShell で CQLReplicator をダウンロードします。	次のコマンドを実行して、プロジェクトをホームフォルダにダウンロードします。
	
	`git clone https://github.com/aws-samples/cql-replicator.git`
	`cd cql-replicator/glue`
	`# Only for AWS CloudShell, the bc package includes bc and dc. Bc is an arbitrary precision numeric processing arithmetic language`
	`sudo yum install bc -y`	

参照設定ファイルを変更します。	`CassandraConnector.conf` と `KeyspacesConnector.conf` をプロジェクトフォルダの `../glue/conf` ディレクトリにコピーします。	AWS DevOps

タスク	説明	必要なスキル

移行プロセスを開始します。	次のコマンドは、CQLReplicator 環境を初期化します。初期化には、.jar アーティファクトのコピーと、AWS Glue コネクタ、S3 バケット、AWS Glue ジョブ、`migration` キースペース、テーブルの作成が含まれます`ledger`。
	
	`cd cql-replicator/glue/bin`
	`./cqlreplicator --state init --sg '"sg-1","sg-2"' \ --subnet "subnet-XXXXXXXXXXXX" \ --az us-west-2a --region us-west-2 \ --glue-iam-role glue-cassandra-migration \ --landing-zone s3://cql-replicator-1234567890-us-west-2`
	
	このスクリプトには次のパラメータが含まれます。
	
	• `--sg` – AWS Glue から Cassandra クラスターへのアクセスを許可し、すべてのトラフィックの自己参照インバウンドルールを含むセキュリティグループ
		AWS DevOps

タスク	説明	必要なスキル
	• `--subnet` – Cassandra クラスターが属するサブネット • `--az` – サブネットのアベイラビリティーゾーン • `--region` – Cassandra クラスターがデプロイされている AWS リージョン • `--glue-iam-role` – AWS Glue がユーザーに代わって Amazon Keyspaces と Amazon S3 を呼び出すときに引き受けることができる IAM ロールのアクセス許可 • `--landing zone` – S3 バケットを再利用するためのオプションのパラメータ (`--landing zone` パラメータの値を指定しない場合、`init`プロセスは設定ファイル、.jar アーティファクト、および中間ファイルを保存するための新しいバケットを作成しようとします) 。	

タスク	説明	必要なスキル
デプロイを検証します。	前のコマンドを実行した後、AWS アカウントには以下が含まれている必要があります。 • AWS Glue の CQLReplicator AWS Glue ジョブと AWS Glue AWS Glue コネクタ • アーティファクトを保存する S3 バケット • Amazon Keyspaces のターゲットキースペース`migration` と`ledger`テーブル	AWS DevOps

CQLReplicator を実行する

タスク	説明	必要なスキル
移行プロセスを開始します。	AWS Glue で CQLReplicator を操作するには、 `--state run` コマンドを使用し、その後に一連のパラメータを使用する必要があります。これらのパラメータの正確な設定は、主に一意の移行要件によって決まります。例えば、有効期限 (TTL) 値と更新をレプリケートすることを選択した場合、または 1 MB を超えるオブジェクトを Amazon S3 にオフロードする場合、これ	AWS DevOps

タスク	説明	必要なスキル
	らの設定は異なる場合があります。 Cassandra クラスターから Amazon Keyspaces にワークロードをレプリケートするには、次のコマンドを実行します。 `./cqlreplicator --state run --tiles 8 \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --region us-west-2 \ --src-keyspace source_keyspace \ --src-table source_table \ --trg-keyspace taget_keyspace \ --writetime-column column_name \ --trg-table target_table --inc-traffic` ソースキースペースとテーブルは Cassandra クラスターsource_keyspace.source_table にあります。ターゲットキースペースとテーブルは Amazon	

タスク	説明	必要なスキル
	Keyspaces `target_keyspace.target_table` にあります。パラメータは、リクエスト数が多い Cassandra クラスターと Amazon Keyspaces に増分トラフィックが過負荷にならないようにする`--inc-traffic` のに役立ちます。 更新をレプリケートするには、コマンドライン`--writetime-column regular_column_name` にを追加します。通常の列は、書き込みタイムスタンプのソースとして使用されます。	

移行プロセスをモニタリングする

タスク	説明	必要なスキル
移行の履歴フェーズで、移行された Cassandra 行を検証します。	バックフィルフェーズでレプリケートされた行数を取得するには、次のコマンドを実行します。 `./cqlreplicator --state stats \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --src-keyspace source_keyspace --src-table`	AWS DevOps

タスク	説明	必要なスキル
	source_table --region us-west-2	

移行プロセスの停止

タスク	説明	必要なスキル
cqlreplicator コマンドまたは AWS Glue コンソールを使用します。	移行プロセスを適切に停止するには、次のコマンドを実行します。 `./cqlreplicator --state request-stop --tiles 8 \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --region us-west-2 \ --src-keyspace source_keyspace --src-table source_table` 移行プロセスをすぐに停止するには、AWS Glue コンソールを使用します。	AWS DevOps

クリーンアップ

タスク	説明	必要なスキル
デプロイされたリソースを削除します。	次のコマンドは、AWS Glue ジョブ、コネクタ、S3 バケット、および Keyspaces テーブル を削除しますledger。 `./cqlreplicator --state cleanup --landing-zone s3://cql-replicator-1234567890-us-west-2`	AWS DevOps

トラブルシューティング

問題	ソリューション
AWS Glue ジョブが失敗し、メモリ不足 (OOM) エラーが返されました。	1. ワーカータイプを変更します (スケールアップ) 。たとえば、 G0.25X を G.1Xまたはに変更G.1XしますG.2X。または、CQLReplicator で AWS Glue ジョブあたりの DPUs の数を増やします (スケールアウト) 。 2. 中断された時点から移行プロセスを開始します。失敗した CQLReplicator ジョブを再起動するには、同じパラメータで --state run コマンドを再実行します。

関連リソース

- AWS Glue を使用した CQLReplicator README.MD://www.
- AWS Glue ドキュメント
- Amazon Keyspaces ドキュメント

- Apache Cassandra

追加情報

移行に関する考慮事項

AWS Glue を使用して Cassandra ワークロードを Amazon Keyspaces に移行できます。移行プロセス中も Cassandra ソースデータベースは完全に機能したままです。レプリケーションが完了したら、Cassandra クラスターと Amazon Keyspaces 間のレプリケーション遅延を最小限 (数分未満) に抑えて、アプリケーションを Amazon Keyspaces に切り替えることができます。データ整合性を維持するために、同様のパイプラインを使用して Amazon Keyspaces から Cassandra クラスターにデータを複製して戻すこともできます。

単位計算の書き込み

例として、1 時間の間に行サイズ 1 KiB で 500,000,000 を書き込むつもりだとします。必要な Amazon Keyspaces 書き込みユニット (WCU) の総数は、次の計算に基づいています。

```
(number of rows/60 mins 60s) 1 WCU per row = (500,000,000/(60*60s) * 1 WCU)
= 69,444 WCUs required
```

1 秒あたり 69,444 WCU は 1 時間のレートですが、オーバーヘッドをいくらか抑えることもできます。 例えば、`69,444 * 1.10 = 76,388` WCUs は10% のオーバーヘッドがあります。

CQL を使用してキースペースを作成する

CQL を使用してキースペースを作成するには、次のコマンドを実行します。

```
CREATE KEYSPACE target_keyspace WITH replication = {'class': 'SingleRegionStrategy'}
CREATE TABLE target_keyspace.target_table ( userid uuid, level text, gameid int,
 description text, nickname text, zip text, email text, updatetime text, PRIMARY KEY
 (userid, level, gameid) ) WITH default_time_to_live = 0 AND CUSTOM_PROPERTIES =
 {'capacity_mode':{ 'throughput_mode':'PROVISIONED', 'write_capacity_units':76388,
 'read_capacity_units':3612 }} AND CLUSTERING ORDER BY (level ASC, gameid ASC)
```

WANdisco LiveData Migrator を使用して Hadoop データを Amazon S3 に移行する

作成者: Tony Velcich

概要

このパターンは、Apache Hadoop データを Hadoop 分散ファイルシステム (HDFS) から Amazon Simple Storage Service (Amazon S3) に移行するプロセスを説明しています。ここでは、WANdisco LiveData Migrator を使用してデータ移行プロセスを自動化します。

前提条件と制限

前提条件

- LiveData Migrator がインストールされる Hadoop クラスターエッジノード。ノードは、以下の要件を満たしている必要があります。
  - 最低限の仕様: 4 つの CPU、16 GB の RAM、100 GB のストレージ。
  - 最低 2 Gbps のネットワーク。
  - WANdisco UI にアクセスするため、エッジノードでポート 8081 にアクセスできること。
  - Java 1.8 64 ビット。
  - Hadoop クライアントライブラリがエッジノードにインストールされていること。
  - HDFS スーパーユーザーとして認証できること (「hdfs」など)。
  - Hadoop クラスターで Kerberos が有効になっている場合は、HDFS スーパーユーザーに適したプリンシパルを含む有効なキータブがエッジノードで使用できる必要があります。
- S3 バケットにアクセスできるアクティブな AWS アカウント。
- オンプレミスの Hadoop クラスター (特にエッジノード) と AWS の間に確立された AWS Direct Connect リンク。

製品バージョン

- LiveData Migrator 1.8.6
- WANdisco UI (OneUI) 5.8.0

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Hadoop クラスター

ターゲットテクノロジースタック

- Amazon S3

アーキテクチャ

次のダイアグラムは、LiveData Migrator アーキテクチャを示しています。

このワークフローは、オンプレミスの HDFS から Amazon S3 へのデータ移行に使用する 4 つの主要コンポーネントで構成されています。

- LiveData Migrator — HDFS から Amazon S3 へのデータ移行を自動化し、Hadoop クラスターのエッジノードに配置されます。
- HDFS — アプリケーションデータへの高スループットアクセスを提供する分散ファイルシステム。
- Amazon S3 – スケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。
- AWS Direct Connect — オンプレミスのデータセンターから AWS への専用ネットワーク接続を確立するサービスです。

自動化とスケール

通常、ソースファイルシステムからパスまたはディレクトリで特定のコンテンツを選択できるように、複数の移行を作成します。また、複数の移行リソースを定義することで、データを複数の独立したファイルシステムに同時に移行することもできます。

エピック

Amazon S3 ストレージを AWS アカウントに構成する

タスク	説明	必要なスキル
AWS アカウントにサインインします。	AWS マネジメントコンソールにサインインして Amazon S3 コンソール 「https://console.aws.amazon.com/s3/」 を開きます。	AWS の使用経験
S3 バケットを作成する。	ターゲットストレージとして使用する既存の S3 バケットがまだない場合は、Amazon S3 コンソールで [バケットの作成] オプションを選択し、パブリックアクセスをブロックするバケット名、AWS リージョン、バケット設定を指定します。AWS と WANdisco は、S3 バケットの「パブリックアクセスのブロックオプション」を有効にし、組織の要件に合わせてバケットアクセスポリシーとユーザー権限ポリシーを設定することを推奨しています。AWS の例は次の場所にあります: https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkthroughs-managing-access-example1.html	AWS の使用経験

LiveData Migrator をインストールする

タスク	説明	必要なスキル
LiveData Migrator インストーラをダウンロードします。	LiveData Migrator インストーラをダウンロードし、Hadoop エッジノードにアップロードします。LiveData Migrator の無料試用版は https://www2.wandisco.com/ldm-trial からダウンロードできます。LiveData Migrator には AWS Marketplace https://aws.amazon.com/marketplace/pp/B07B8SZND9 からアクセスすることもできます。	Hadoop 管理者、アプリ所有者
LiveData Migrator をインストールします。	ダウンロードしたインストーラーを使用して、Hadoop クラスターのエッジノードに HDFS スーパーユーザーとして LiveData Migrator をインストールします。インストールコマンドについては、「追加情報」セクションを参照してください。	Hadoop 管理者、アプリ所有者
LiveData Migrator やその他のサービスのステータスを確認します。	「追加情報」セクションに記載されているコマンドを使用して、LiveData Migrator および WANdisco UI のステータスを確認します。	Hadoop 管理者、アプリ所有者

WANdisco UI を使用してストレージを構成する

タスク	説明	必要なスキル
LiveData Migratorアカウントを登録します。	ウェブブラウザでポート 8081 (Hadoop エッジノード) から WANdisco UI にログインし、登録に必要な情報を入力します。たとえば、myldmhost.example.com という名前のホストで LiveData Migrator を実行している場合、URL は http://myldmhost.example.com:8081 になります。	アプリ所有者
ソース HDFS ストレージを構成します。	ソース HDFS ストレージに必要な構成の詳細を指定します。これには「fs.DefaultFS」値とユーザー定義のストレージ名が含まれます。Kerberos が有効になっている場合は、LiveData Migrator が使用するプリンシパルとキータブの場所を指定します。クラスターで NameNode HA が有効になっている場合は、エッジノードの core-site.xml ファイルと hdfs-site.xml ファイルへのパスを指定します。	Hadoop 管理者、アプリ所有者
ターゲット Amazon S3 ストレージを構成します。	ターゲットストレージを S3a タイプとして追加します。ユーザー定義のストレージ名と、S3 バケット名を指定します。認証情報プロバイダーオプションに「org.apache.hadoop.fs.s3a.simpleaws	AWS、アプリ所有者

タスク	説明	必要なスキル
	CredentialsProvider」と入力し、S3 バケットの AWS アクセスキーとシークレットキーを指定します。その他の S3a プロパティも必要になります。詳細については、https://docs.wandisco.com/live-data-migrator/docs/command-reference/#filesystem-add-s3a にある LiveData Migrator ドキュメントの「S3a プロパティ」セクションを参照してください。	

移行の準備をする

タスク	説明	必要なスキル
除外を追加 (必要な場合)。	特定のデータセットを移行から除外したい場合は、ソース HDFS ストレージを除外対象として追加します。これらの除外は、ファイルサイズ、ファイル名 (正規表現パターンに基づく)、および変更日に基づいて設定できます。	Hadoop 管理者、アプリ所有者

移行を作成して開始する

タスク	説明	必要なスキル
移行を作成して構成します。	WANdisco UI のダッシュボードで移行を作成します。ソース (HDFS) とターゲット (S3	Hadoop 管理者、アプリ所有者

タスク	説明	必要なスキル
	バケット) を選択します。前のステップで定義した新しい除外を追加します。[上書き] または [サイズが一致した場合はスキップ] オプションのいずれかを選択します。すべてのフィールドに入力したら、移行を作成します。	
移行を開始します。	ダッシュボードで、作成した移行を選択します。クリックして移行を開始します。また、移行の作成時に自動開始オプションを選択して、移行を自動的に開始することもできます。	アプリ所有者

帯域幅の管理 (オプション)

タスク	説明	必要なスキル
送信元とターゲット間のネットワーク帯域幅の制限を設定します。	ダッシュボードのストレージリストでソースストレージを選択し、グループリストで [帯域幅管理] を選択します。無制限オプションを解除して、最大帯域幅制限と単位を指定します。[Apply (適用)] を選択します。	アプリ所有者、ネットワーク

移行のモニタリングと管理

タスク	説明	必要なスキル
WANdisco UI を使用して移行情報を表示します。	WANdisco UI を使用して、ライセンス、帯域幅、ストレージ、移行情報を表示します。この UI には通知システムも装備されているため、エラー、警告、使用状況における重要なマイルストーンに関する通知を受け取ることができます。	Hadoop 管理者、アプリ所有者
移行を停止、再開、削除します。	移行を STOPPED 状態にすることで、移行によるターゲットへのコンテンツの転送を停止できます。停止した移行は再開できます。STOPPED 状態の移行も削除できます。	Hadoop 管理者、アプリ所有者

関連リソース

- LiveData Migrator ドキュメント
- AWS Marketplace の LiveData Migrator
- WANdisco LiveData Migrator のデモンストレーション (ビデオ)

追加情報

LiveData Migrator をインストールする

インストーラが作業ディレクトリ内にあることを前提として、次のコマンドを使用して LiveData Migrator をインストールできます。

```
su - hdfs
chmod +x livedata-migrator.sh && sudo ./livedata-migrator.sh
```

インストール後の LiveData Migrator とその他のサービスのステータスの確認

LiveData Migrator、Hive Migrator、WANdisco UI のステータスを確認するには、次のコマンドを使用します。

```
service livedata-migrator status
service hivemigrator status
service livedata-ui status
```

Oracle ビジネスインテリジェンス 12c をオンプレミスサーバーから AWS クラウドに移行

作成者: Lanre (Lan-Ray) showunmi (AWS) と Patrick Huang (AWS)

概要

このパターンは、AWS CloudFormation を使用して「Oracle ビジネスインテリジェンスエンタープライズエディション 12c」をオンプレミスサーバーから AWS クラウドに移行する方法を示しています。また、他の AWS サービスを使用して、高可用性、セキュリティ、柔軟性、および動的なスケーリング機能を提供する Oracle BI 12c コンポーネントを実装する方法についても説明します。

Oracle BI 12c の AWS クラウドへの移行に関連するベストプラクティスのリストについては、このパターンの「追加情報」セクションを参照してください。

> **Note**
>
> 既存の Oracle BI 12c データをクラウドに転送する前に、複数のテスト移行を実行することがベストプラクティスです。これらのテストは、移行アプローチを微調整し、潜在的な問題を特定して修正し、ダウンタイム要件をより正確に見積もるのに役立ちます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 「AWS Virtual Private Network (AWS VPN)」サービスまたは「AWS Direct Connect」のいずれかを介した、オンプレミスサーバーと AWS の間の安全なネットワーク接続
- お使いの Oracle オペレーティングシステム、Oracle BI 12c、Oracle データベース、Oracle WebLogic Server、および Oracle HTTP Server 用のソフトウェアライセンス

機能制限

ストレージサイズの制限については、「Amazon Relational Database Service (Amazon RDS) for Oracle」のドキュメントを参照してください。

製品バージョン

- Oracle ビジネスインテリジェンスエンタープライズエディション 12c
- Oracle WebLogic サーバー 12c
- Oracle HTTP サーバー 12c
- Oracle Database 12c (またはそれ以降)
- Oracle Java SE 8

アーキテクチャ

次の図は、AWS クラウドで Oracle BI 12c コンポーネントを実行するためのアーキテクチャの例を示しています。

この図は次のようなアーキテクチャを示しています。

1. Amazon Route 53 はドメインネームサービス (DNS) 設定を提供します。
2. Elastic Load Balancing (ELB) は、ネットワークトラフィックを分散して、複数のアベイラビリティーゾーンにわたる Oracle BI 12c コンポーネントのスケーラビリティと可用性を向上させます。
3. Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling グループは、複数のアベイラビリティーゾーンにわたって Oracle HTTP サーバー、Weblogic Admin Server、マネージド BI サーバーをホストします。
4. Oracle データベースの Amazon Relational Database Service (Amazon RDS) は、複数のアベイラビリティーゾーンの BI サーバーメタデータを格納します。
5. Amazon Elastic File System (Amazon EFS) は、共有ファイルストレージ用のすべての Oracle BI 12c コンポーネントにマウントされています。

テクノロジースタック

- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic File System (Amazon EFS)
- 「Amazon RDS for Oracle」
- AWS Certificate Manager (ACM)
- Elastic Load Balancing (ELB)

- Oracle BI 12c
- Oracle WebLogic サーバー 12c
- Oracle HTTP サーバー (OHS)

ツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「AWS Certificate Manager (ACM)」は、AWS ウェブサイトとアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書とキーの作成、保存、更新に役立ちます。
- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。仮想サーバーを必要な数だけ起動して、迅速にスケールアップまたはスケールダウンができます。
- 「Amazon EC2 Auto Scaling」は、アプリケーションの可用性を維持するのに役立ち、定義した条件に従って、Amazon EC2 インスタンスのインスタンスを自動的に追加または削除できます。
- 「Amazon Elastic File System (Amazon EFS)」は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- 受信したアプリケーションまたはネットワークトラフィックを複数のターゲットに分散するためには、「Elastic Load Balancing」を使用します。例えば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および 1 つまたは複数のアベイラビリティーゾーンの IP アドレスにトラフィックを分散できます。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。
- 「Oracle Data Pump」を使用すると、データやメタデータをあるデータベースから別のデータベースに高速で移動できます。

- 「Oracle Fusion Middleware」は、ID管理、コラボレーション、ビジネスインテリジェンスレポート作成のためのアプリケーション開発ツールと統合ソリューションのスイートです。
- 「Oracle GoldenGate」は、Oracle Cloud Infrastructure におけるデータレプリケーションおよびストリームデータ処理ソリューションの設計、実行、調整、監視を支援します。
- 「Oracle WebLogicスクリプトツール（WLST）」には、WebLogic クラスタを水平方向にスケールアウトするのに役立つコマンドライン・インタフェースが用意されています。

エピック

ソース環境を評価してください。

タスク	説明	必要なスキル
ソフトウェアインベントリ情報を収集する。	以下を含む、ソーステクノロジースタックの各ソフトウェアコンポーネントのバージョンとパッチレベルを特定します。 • Oracle オペレーティングシステム • Oracle Database • Oracle BI 12c • Oracle WebLogic サーバー • Oracle HTTP サーバー • Java	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者
コンピューティングとストレージのインベントリ情報を収集します。	ソース環境で、以下について現在および過去の使用状況メトリクスを確認します。 • CPU の使用 • メモリ使用量 • ストレージ使用量	移行アーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者、システム管理者

タスク	説明	必要なスキル
	Important 使用量の過去の急増を考慮してください。	
ソース環境のアーキテクチャと要件に関する情報を収集します。	以下の知識を含め、ソース環境のアーキテクチャとその要件を十分に理解してください。 • Oracle WebLogic Server のドメイン設定 • クラスタリング • 負荷分散 • 接続 • 可用性 • ディザスタリカバリの要件	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者
Java データベース接続 (JDBC) データソースを特定します。	ソース環境が使用するデータベースエンジンごとに、ソース環境の JDBC データソースとドライバーに関する情報を収集します。	移行アーキテクト、アプリ所有者、Oracle BI 管理者、データベースエンジニア、または管理者
環境固有の設定に関する情報を収集します。	以下を含む、ソース環境に固有の設定や構成に関する情報を収集します。 • カスタムの起動スクリプトとシャットダウンスクリプト • Java とその他の環境変数 • 証明書	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者

タスク	説明	必要なスキル
他のアプリケーションとの依存関係を特定します。	他のアプリケーションとの依存関係を引き起こすソース環境の統合に関する情報を収集します。 **Important** Lightweight Directory Access Protocol (LDAP) 統合やその他のネットワーク要件を必ず特定してください。	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者

ターゲット環境の設計

タスク	説明	必要なスキル
高レベルの設計ドキュメントを作成します。	ターゲットアーキテクチャ設計文書を作成する。ソース環境を評価する際に収集した情報を設計文書の作成に使用するようにしてください。	ソリューションアーキテクト、アプリケーションアーキテクト、データベースエンジニア、マイグレーションアーキテクト
設計文書の承認を得る。	利害関係者と設計文書を検討し、必要な承認を得る。	アプリケーションまたはサービスオーナー、ソリューションアーキテクト、アプリケーションアーキテクト

インフラストラクチャをデプロイする

タスク	説明	必要なスキル
CloudFormation でインフラストラクチャーコードを準備します。	CloudFormation テンプレートを作成して、Oracle BI 12c インフラストラクチャを AWS クラウドにプロビジョニングします。 詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormation テンプレートの操作」を参照してください。 **Note** すべてのリソースに対して 1 つの大きなテンプレートを作成するのではなく、Oracle BI 12c 階層ごとにモジュラー CloudFormation テンプレートを作成するのがベストプラクティスです。CloudFormation のベストプラクティスの詳細については、AWS ブログの「AWS CloudFormation を使用してデプロイを自動化する際の 8 つのベストプラクテ	クラウドインフラストラクチャアーキテクト、ソリューションアーキテクト、アプリケーションアーキテクト

タスク	説明	必要なスキル
	ィス」を参照してください。	
必要なソフトウェアをダウンロードします。	「Oracle ウェブサイト」から、次のソフトウェアと必要なバージョンとパッチをダウンロードします。 • Java JDK8 • Oracle WebLogic サーバー 12c • Oracle BI 12c	マイグレーションアーキテクト、データベースエンジニア、アプリケーションアーキテクト
インストールスクリプトを準備します。	サイレントインストールを実行するソフトウェアインストールスクリプトを作成します。これらのスクリプトは導入の自動化を簡素化します。 詳細については、Oracle Support サイトにある「OBIEE 12c:サイレント・インストールを実行する方法?」を参照してください。ドキュメントを閲覧するには、Oracle Supportアカウントが必要です。	マイグレーションアーキテクト、データベースエンジニア、アプリケーションアーキテクト

タスク	説明	必要なスキル
ウェブ層とアプリケーション層の Amazon EBS-backed Linux AMI を作成してください。	1. ウェブ層とアプリケーション層用に「Amazon EC2 インスタンスをデプロイして設定します」。インスタンスが以下を実行するための前提条件を満たしていることを確認してください。 • Oracle オペレーティングシステム環境のセットアップ • Oracle オペレーティングシステムのユーザーアカウントの設定 • Java ソフトウェアのインストール 2. インスタンスの Amazon マシンイメージ (AMI) を作成し、将来の使用に備えてコピーを保存します。詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドの「Amazon EBS-backed Linux AMI の作成」を参照してください。	マイグレーションアーキテクト、データベースエンジニア、アプリケーションアーキテクト

タスク	説明	必要なスキル
CloudFormation を使用して AWS インフラストラクチャを起動します。	作成したCloudFormationテンプレートを使用して、Oracle BI 12cのウェブ層とアプリケーション層をモジュールにデプロイします。 手順については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation の使用開始」を参照してください。	クラウドインフラストラクチャアーキテクト、ソリューションアーキテクト、アプリケーションアーキテクト

新規インストールを使用して Oracle BI 12c を AWS に移行してください

タスク	説明	必要なスキル
必要なソフトウェアをを準備します。	Amazon EC2 インスタンスにアクセスできる場所に、必要なソフトウェアをステージングします。たとえば、ウェブサーバーやアプリケーションサーバーにアクセスできるAmazon S3 または別のAmazon EC2 インスタンスにソフトウェアをステージングできます。	移行アーキテクト、Oracle BI アーキテクト、クラウドインフラストラクチャアーキテクト、ソリューションアーキテクト、アプリケーションアーキテクト
Oracle BI 12c をインストールするためのリポジトリデータベースを準備します。	新しい「Amazon RDS for Oracle」データベースインスタンスに対して「Oracle リポジトリ作成ユーティリティ (RCU)」を実行して Oracle BI 12c スキーマを作成します。	クラウドインフラストラクチャアーキテクト、ソリューションアーキテクト、アプリケーションアーキテクト、移行アーキテクト、Oracle BI アーキテクト

タスク	説明	必要なスキル
Oracle Fusion Middleware 12c と Oracle BI 12c をインストールします。	1. 1 つの Amazon EC2 インスタンスから始めて、Oracle Fusion Middleware 12c インフラストラクチャと OBIEE 12c をインストールします。詳細については、「Oracle Business Intelligence の Oracle Fusion ミドルウェア エンタープライズ・デプロイメント・ガイド」の以下のセクションを参照してください。 • 「BIHOST1 でのインフラストラクチャーインストーラの起動」 • 「エンタープライズへの導入に備えて Oracle ビジネスインテリジェンスをインストールする」 **Note** Amazon EFS を使用して、Oracle BI 12c クラスターノード間で共有されるディレクトリをホストします。 2. 必要なパッチをインストールに適用します。 3. インスタンスの AMI を作成し、後で使用できるようにコピーを保存します。	マイグレーション・アーキテクト、Oracle BI アーキテクト

タスク	説明	必要なスキル
Oracle BI 12c 用に Oracle WebLogic Server ドメインを設定してください。	Oracle BI 12c ドメインを非クラスタ・デプロイメントとして構成してください。 詳細は、「Oracle ビジネスインテリジェンス向け Oracle Fusion ミドルウェアエンタープライズデプロイメントガイド」の「BI ドメインの設定」を参照してください。	マイグレーション・アーキテクト、Oracle BI アーキテクト
Oracle BI 12c の水平スケールアウトを実行します。	単一ノードを目的のノード数まで水平方向にスケールアウトします。 詳細は、「Oracle Fusion ミドルウェアエンタープライズデプロイメントガイド」の「Oracleビジネスインテリジェンスのスケールアウト」を参照してください。	マイグレーション・アーキテクト、Oracle BI アーキテクト

タスク	説明	必要なスキル
Oracle HTTP Server 12c をインストールします。	1. Oracle HTTP Server 12c をOracleのウェブ層の Amazon EC2 インスタンスにインストールします。手順については、「Oracle アクセス管理 12c 用 Oracle HTTP Server のインストールと設定」の「Oracle HTTP Server 12c のインストール」を参照してください。 2. 必要なパッチをインストールに適用します。 3. インスタンスの AMI を作成し、後で使用できるようにコピーを保存します。	マイグレーション・アーキテクト、Oracle BI アーキテクト
SSL ターミネーション用のロードバランサーを設定します。	1. ACM で SSL 証明書を作成またはインポートします。 2. 「SSL 証明書を ELB に関連付けます」。	クラウド・インフラストラクチャー・アーキテクト、移行アーキテクト

タスク	説明	必要なスキル
ビジネスインテリジェンスのメタデータアーティファクトを AWS に移行します。	1. オンプレミスの Oracle BI 12c インストールから Oracle ビジネスインテリジェンスアプリケーションアーカイブ (BAR) ファイルをエクスポートします。BAR ファイルをエクスポートするには、「WebLogic スクリプトツール (WLST)」を使用して「exportServiceInstance」コマンドを実行します。 2. オンプレミスの BAR ファイルを AWS Oracle BI 12c インストールにインポートします。BAR ファイルをインポートするには、[ImportServiceInstance] WLST コマンドを実行します。	マイグレーション・アーキテクト、Oracle BI アーキテクト

タスク	説明	必要なスキル
移行後のタスクを実行する。	BAR ファイルをインポートしたら、次の操作を行います。 • その他の「JDBC データソース」を設定します。 • PostgreSQL や Amazon Redshift などの他のデータソース用のドライバーをインストールします。 • Oracle「LDAP」、「SSL」、「シングルサインオン (SSO)」、および「WebLogic セキュリティストア」を設定します。 • AWS Identity and Access Management (IAM) ポリシーを設定します。 • 「使用状況追跡の有効化」。 • 他のシステムとのインテグレーションを設定します。 • 任意のカスタムスクリプトを移行します。	マイグレーション・アーキテクト、Oracle BI アーキテクト

新しい環境のテスト

タスク	説明	必要なスキル
新しい Oracle BI 12c 環境をテストしてください。	新しい Oracle BI 12c 環境で エンドツーエンドのテストを実施します。可能な限り自動化を利用します。	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者

タスク	説明	必要なスキル
	テストアクティビティの例には、以下のようなものがあります。 • ダッシュボード、レポート、URL の検証 • ユーザー受け入れテスト (UAT) • 運用承認テスト (OAT) **Note** 必要に応じて追加のテストと検証を実施します。	

新しい環境に切り替える

タスク	説明	必要なスキル
オンプレミスの Oracle BI 12c 環境へのトラフィックを切断します。	指定されたカットオーバー期間に、オンプレミスの Oracle BI 12c 環境へのトラフィックをすべて停止します。	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者
新しい Oracle BI 12c リポジトリデータベースをソースデータベースと再同期します。	Amazon RDS Oracle Oracle BI 12c リポジトリデータベースをオンプレミスデータベースと再同期します。 データベースを同期するには、「Oracle データポンプの更新」または「AWS DMS 変	Oracle BI 管理者、データベースエンジニア/管理者

タスク	説明	必要なスキル
	更データキャプチャ (CDC)」を使用できます。	
Oracle BI 12c の URL を新しい AWS 環境を指すように切り替えてください。	内部 DNS サーバーの Oracle BI 12c URL を更新して、新しい AWS インストールを指すようにします。	マイグレーションアーキテクト、ソリューションアーキテクト、アプリ所有者、Oracle BI 管理者
新しい環境をモニタリングします。	以下のツールのいずれかを使用して、新しい Oracle BI 12c 環境を監視します。 • Amazon CloudWatch • 「Amazon RDS Performance Insights」 • Oracle Enterprise Manager	Oracle BI 管理者、データベースエンジニア/管理者、アプリケーション管理者
プロジェクトの承認を得てください。	テスト結果を利害関係者と確認し、移行を完了するために必要な承認を得てください。	アプリ所有者、サービスオーナー、クラウドインフラストラクチャーアーキテクト、マイグレーションアーキテクト、Oracle BI アーキテクト

関連リソース

- RDS for Oracle での Oracle リポジトリ作成ユーティリティの使用 (Amazon RDSユーザーガイド)
- 「Amazon RDS 上のOracle」(Amazon RDS ユーザーガイド)
- 「AWS 上のOracle WebLogic Server 12c」(AWS ホワイトペーパー)
- 「高可用性のための Oracle ビジネスインテリジェンスのデポロイ」(Oracle ヘルプセンター)
- 「Oracle ビジネスインテリジェンスアプリケーションアーカイブ (BAR) ファイル」(Oracle ヘルプセンター)
- 「OBI 12c を環境間で移行する方法」(Oracle Support)

追加情報

以下は、Oracle BI 12c を AWS クラウドに移行することに関連するベストプラクティスのリストです。

リポジトリデータベース

Oracle BI 12c データベーススキーマを Amazon RDS for Oracle インスタンスでホストするのがベストプラクティスです。このインスタンスタイプは、ハードウェアのプロビジョニング、データベースのセットアップ、パッチ適用、バックアップなどの管理タスクを自動化しながら、コスト効率が高くサイズ変更可能な容量を提供します。

詳細については、[Amazon RDS ユーザーガイド] の「RDS for Oracle での Oracle リポジトリ作成ユーティリティの使用」を参照してください。

ウェブ層とアプリケーション層

多くの場合、「メモリ最適化された Amazon EC2 インスタンス」は Oracle BI 12c サーバーに適しています。どのインスタンスタイプを選択する場合でも、プロビジョニングするインスタンスがシステムのメモリ使用要件を満たしていることを確認してください。また、Amazon EC2 インスタンスの使用可能なメモリに基づいて「十分な WebLogic Java 仮想マシン (JVM) ヒープサイズ」を設定してください。

ローカルストレージ

I/O は Oracle BI 12c アプリケーションの全体的なパフォーマンスにおいて重要な役割を果たします。Amazon Elastic Block Store (Amazon EBS) は、さまざまなワークロードパターンに最適化されたさまざまなストレージクラスを提供しています。必ずユースケースに合った Amazon EBS ボリュームタイプを選択してください。

詳細については、EBS ドキュメントの「Amazon EBS 機能」を参照してください。

共有ストレージ

クラスタ化された Oracle BI 12c ドメインには、以下のリソース用の共有ストレージが必要です。

- 設定ファイル
- Oracle BI 12c シングルトンデータディレクトリ (SDD)
- Oracle グローバルキャッシュ
- Oracle BI スケジューラースクリプト
- Oracle WebLogic サーバーバイナリ

この共有ストレージ要件は、スケーラブルでフルマネージド型の伸縮自在なネットワークファイルシステム (NFS) ファイルシステムを提供する「Amazon EFS」を使用することで満たすことができます。

共有ストレージのパフォーマンスを微調整する

Amazon EFS には、[プロビジョニング型] と [バースト型] の 2 つの「スループットモード」があります。このサービスには、[汎用] モードと [最大 I/O] モードの 2 つの「パフォーマンスモード」もあります。

パフォーマンスを微調整するには、まず [汎用] パフォーマンスモードと [プロビジョンド] スループットモードでワークロードをテストします。これらのテストを行うと、これらのベースラインモードが希望するサービスレベルを満たすのに十分かどうかを判断するのに役立ちます。

詳細については、Amazon EFS ユーザーガイドの「Amazon EFS パフォーマンス」を参照してください。

可用性とディザスタリカバリ

アベイラビリティーゾーンに障害が発生した場合にそれらのリソースを保護するために、Oracle BI 12c コンポーネントを複数のアベイラビリティーゾーンにデプロイするのがベストプラクティスです。以下は、AWS クラウドでホストされている特定の Oracle BI 12c リソースの可用性とディザスタリカバリのベストプラクティスのリストです。

- Oracle BI 12c リポジトリデータベース:マルチ AZ Amazon RDS データベースインスタンスを Oracle BI 12c リポジトリデータベースにデプロイします。マルチ AZ 配置では、Amazon RDS は自動的に別の AZ に同期スタンバイレプリカをプロビジョニングし、維持します。Oracle BI 12c リポジトリデータベースインスタンスをアベイラビリティゾーン間で実行すると、計画的なシステム保守時の可用性が向上し、インスタンスとアベイラビリティゾーンの障害からデータベースを保護できます。
- Oracle BI 12c マネージドサーバー:耐障害性を実現するには、複数のアベイラビリティーゾーンにまたがるように設定された Amazon EC2 Auto Scaling グループのマネージドサーバーに Oracle BI 12c システムコンポーネントをデプロイするのがベストプラクティスです。Auto Scaling は、「Amazon EC2 ヘルスチェック」に基づいて障害のあるインスタンスを置き換えます。アベイラビリティーゾーンに障害が発生した場合、Oracle HTTP Servers は引き続き機能しているアベイラビリティーゾーン内のマネージドサーバーにトラフィックを転送します。次に、自動スケーリングはホスト数の要件に合わせてインスタンスを起動します。機能しているマネージドサーバーに既存のセッションをスムーズにフェイルオーバーできるように、HTTP セッション状態の複製を有効化することをお勧めします。

- Oracle BI 12c 管理サーバー:管理サーバーの可用性を高めるには、複数のアベイラビリティーゾーンにまたがるように設定された Amazon EC2 Auto Scaling グループで管理サーバーをホストします。次に、グループセットの最小サイズおよび最大サイズを 1 に設定します。アベイラビリティーゾーンに障害が発生した場合、Amazon EC2 Auto Scaling は代替アベイラビリティーゾーンで代替管理サーバーを起動します。同じアベイラビリティーゾーン内の障害が発生した基盤となるホストを復旧するには、「Amazon EC2 Auto Recovery」を有効化できます。
- Oracle Web Tier サーバー: Oracle HTTP サーバーを Oracle WebLogic Server ドメインに関連付けるのがベストプラクティスです。高可用性のためには、複数のアベイラビリティーゾーンを処理するように設定された Amazon EC2 Auto Scaling グループに Oracle HTTP サーバーをデプロイします。次に、サーバーを ELB エラスティックロードバランサーの背後に配置します。ホスト障害に対する保護を強化するために、Amazon EC2 Auto Recovery を有効化できます。

スケーラビリティ

AWS クラウドの伸縮性により、ワークロードの要件に応じてアプリケーションを水平または垂直にスケーリングできます。

垂直スケーリング

アプリケーションを垂直方向にスケーリングするには、Oracle BI 12c コンポーネントを実行している Amazon EC2 インスタンスのサイズとタイプを変更できます。デプロイの開始時にインスタンスを過剰にプロビジョニングする必要はなく、不要なコストも発生しません。

水平スケーリング

Amazon EC2 Auto Scaling は、ワークロード要件に基づいてマネージドサーバーを自動的に追加または削除することで、アプリケーションを水平方向にスケーリングするのに役立ちます。

> **Note**
>
> Amazon EC2 Auto Scaling による水平スケーリングには、スクリプティングスキルと実装のための徹底的なテストが必要です。

バックアップとリカバリ

以下は、AWS クラウドでホストされている特定の Oracle BI 12c リソースのバックアップとリカバリのベストプラクティスのリストです。

- Oracle ビジネスインテリジェンスのメタデータリポジトリ: Amazon RDS はデータベースインスタンスのバックアップを自動的に作成して保存します。これらのバックアップは、指定した期間保持されます。Amazon RDS のバックアップ期間と保持設定は、必ずデータ保護要件に基づいて設定してください。詳細については「Amazon RDS のバックアップと復元」を参照してください。
- マネージドサーバー、管理サーバー、ウェブ階層サーバー:データ保護と保持の要件に基づいて「Amazon EBS スナップショット」を設定してください。
- 共有ストレージ: Amazon EFS に保存されているファイルのバックアップと復元は、「AWS Backup」を使用して管理できます。AWS Backup サービスをデプロイして、Amazon EC2、Amazon EBS、Amazon RDS などの他のサービスのバックアップとリカバリを一元管理することもできます。詳細については、「AWS Backup とは?」を参照してください。「AWS Backup 開発者ガイド」にあります。

セキュリティとコンプライアンス

以下は、AWS クラウド内の Oracle BI 12c アプリケーションを保護するのに役立つセキュリティのベストプラクティスと AWS サービスのリストです。

- 保存時の暗号化: Amazon RDS、Amazon EFS、Amazon EBS はすべて、業界標準の暗号化アルゴリズムをサポートしています。「AWS Key Management Service (AWS KMS)」を使用して、暗号化キーを作成および管理し、AWS サービス全体およびアプリケーション内でのそれらの使用を管理できます。Oracle BI 12c リポジトリデータベースをホストしている Amazon RDS for Oracle データベースインスタンスで「Oracle 透過的データ暗号化 (TDE)」を設定することもできます。
- 転送中の暗号化: Oracle BI 12c インストールのさまざまなレイヤー間で転送中のデータを保護するために、SSL プロトコルまたは TLS プロトコルのいずれかをアクティブ化することがベストプラクティスです。「AWS Certificate Manager (ACM)」を使用して、Oracle BI 12c リソース用のパブリック SSL 証明書とプライベート SSL 証明書、および TLS 証明書のプロビジョニング、管理、およびデポロイを行うことができます。
- ネットワークセキュリティ: Oracle BI 12c リソースは、ユースケースに適したアクセス制御が設定された Amazon VPC にデプロイするようにしてください。インストールを実行している Amazon EC2 インスタンスからのインバウンドトラフィックとアウトバウンドトラフィックをフィルタリングするようにセキュリティグループを設定します。また、定義されたルールに基づいてトラフィックを許可または拒否する「ネットワークアクセスコントロールリスト (NACL)」を必ず設定してください。
- モニタリングとロギング:「AWS CloudTrail」を使用して、Oracle BI 12c リソースを含む AWS インフラストラクチャへの API 呼び出しを追跡できます。この機能は、インフラストラクチャの変更を追跡したり、セキュリティ分析を行う場合に役立ちます。「Amazon CloudWatch」を使用

して運用データを表示することもできます。これにより、Oracle BI 12c アプリケーションのパフォーマンスと状態に関する実用的な洞察を得ることができます。アラームを設定し、そのアラームに基づいて自動アクションを実行することもできます。Amazon RDS には、「拡張モニタリング」や「Performance Insights」など、追加のモニタリングツールが用意されています。

MirrorMaker を使用して、オンプレミスの Apache Kafka クラスターを Amazon MSK に移行する

ハン・チャン（AWS）とタナー・プラット（AWS）によって作成されました

概要

このパターンは、オンプレミス、セルフマネージド、またはホスト型の Apache Kafka クラスターを、Amazon Managed Streaming for Apache Kafka (Amazon MSK) に移行するガイダンスを提供します。このパターンを使用して、ある Amazon MSK クラスターから別の Amazon MSK クラスターに移行することもできます。

Apache Kafka には、2 つの Kafka クラスター間でデータをレプリケートするMirrorMaker 特徴量が含まれています。MirrorMaker は、コンシューマーグループの一部であるコンシューマーの集まりで構成されています。コンシューマーはソースクラスターのトピックからデータを読み取り、そのデータをプロデューサーに渡し、プロデューサーはデータをターゲットクラスターに書き込みます。

Amazon MSK ドキュメントには、MirrorMaker バージョン 1.0 を使用してオンプレミスの Kafka クラスターを Amazon MSK に移行するプロセスの 大まかな概要 が記載されています。このパターンは、MirrorMaker バージョン 2.0 の使用に関する包括的な手順を提供することで、この情報を補足します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 以下のいずれかのKafka ソースクラスター：
  - オンプレミスのデータセンターで
  - クラウドで管理されます
  - パートナーを通じてホストされています

制約事項

- MirrorMaker バージョン 2.0 を使用するには、ソースクラスターが Apache Kafka バージョン 2.4.0 以降を実行している必要があります。以前のバージョンでは、MirrorMaker バージョン 1.0 を使用するためにAmazon MSK ドキュメント の手順を参照してください。

製品バージョン

- MirrorMaker バージョン 2.0
- Apache Kafka バージョン 2.8.0 Amazon MSK がサポートする Apache Kafka のバージョンに関する詳細については、サポートされている Apache Kafka のバージョン を参照してください。

アーキテクチャ

ソーステクノロジースタック

- ソース：オンプレミスまたはセルフマネージド Apache Kafka クラスター

ターゲットテクノロジースタック

- Amazon MSK クラスター

ターゲット アーキテクチャ

図表に示す内容は以下のステップです。

1. MirrorMaker は、ソース Kafka クラスター内のトピックとコンシューマーグループからデータを読み取ります。
2. MirrorMaker は、データとコンシューマー情報をターゲットの Amazon MSK クラスターに複製します。

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「Amazon Managed Streaming for Apache Kafka (Amazon MSK)」 は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションを、構築および実行することを支援するフルマネージドサービスです。

- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内でリソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

その他のツール

- 「Apache Kafka」 は、オープンソースのイベントストリーミングプラットフォームです。このパターンでは、Kafka の「MirrorMaker」 特徴量を使用してクラスター間の移行を実行します。

ベストプラクティス

MirrorMaker はソース環境でもターゲット環境でも実行できますが、ターゲットクラスターのできるだけ近くで実行することをお勧めします。詳細については、Apache Kafka ドキュメントの「ベストプラクティス：リモートからコンシューマを使用し、ローカルに生成する」 を参照してください。

エピック

VPC を作成し、Amazon MSK クラスターをターゲットにします。

タスク	説明	必要なスキル
VPC を作成します。	1. ターゲット AWS アカウントに新しい VPC とサブネットグループを作成します。手順については、Create a role を参照してください。 2. これらは、同じ VPC 内の 2 つの異なるアベイラビリティーゾーンにあるプライベートサブネットである必要があります。手順については、Create a role を参照してください。異なるアベイラビリティーゾーンを使	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

タスク	説明	必要なスキル
	用すると、高い可用性と耐障害性が実現します。 **Note** パブリックインターネット接続を使用して Kafka クラスターを移行する場合は、パブリックサブネットを作成し、Amazon MSK クラスターへのパブリックアクセスを有効にします。	
Amazon MSK クラスターを作成します。	Amazon ECS クラスターを作成します。手順については、「AWS マネジメントコンソールを使用してクラスターを作成する」または「AWS CLI を使用してクラスターを作成する」を参照してください。以前に作成した VPC とサブネットを使用するようにクラスターを設定します。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

MirrorMaker のセットアップ

タスク	説明	必要なスキル
MirrorMaker をインストールします。	1. EC2 インスタンスを起動します。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

タスク	説明	必要なスキル
	2. EC2 インスタンスに接続します。 3. EC2 インスタンスで、最新の Kafka リリースをダウンロードして抽出します。手順については、 ドキュメントの「クイックスタート」を参照してください。 **Note** このパターンでは、MirrorMaker2.0 を専用の MirrorMaker クラスターとして Amazon EC2 インスタンスにインストールします。このオプションは開発環境には適しており、このパターンで使用されているアプローチでもあります。MirrorMaker 2.0 のその他のデプロイオプションの詳細については、このパターンの「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル
Kafka クラスター情報を指定します。	Kafka bin クライアントのインストールフォルダーに mm2.properties ファイルを作成し、ソース Kafka クラスター用に設定します。手順については、「専用の MirrorMaker クラスターの実行」(Kafka ドキュメント) を参照してください。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア
MirrorMaker を起動します。	次のコマンドを入力して MirrorMaker を起動し、mm2.properties ファイルを渡します。 `$ ./bin/connect-mirror-maker.sh mm2.properties`	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア
進行状況をモニタリングする	ミラーリングの進行状況をチェックするには、各トピックの最後のオフセットと、MirrorMaker が消費する現在のオフセットの間の遅延を調べます。手順については、Kafka ドキュメントの「ジオレプリケーションの監視」 を参照してください。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

カットオーバー

タスク	説明	必要なスキル
コンシューマアプリケーションを停止します。	ソースクラスターのデータを消費するコンシューマーアプ	アプリ開発者

タスク	説明	必要なスキル
	リケーションをすべて停止します。	
コンシューマアプリケーションを停止します。	アプリケーションのブートストラップ構成を、デスティネーションクラスターを指すように変更します。次に、ターゲットクラスターでコンシューマを開始します。	アプリ開発者
ソースクラスターですべてのETL プロセスを停止します。	コンシューマーアプリケーションがターゲットクラスターで正常に消費できるようになったら、ソースクラスターのプロデューサーを停止します。	アプリ開発者
ターゲットクラスターでプロデューサーを起動します。	プロデューサーの設定ブートストラップサーバーを変更し、ターゲットクラスターを指定します。MirrorMaker がソースクラスターからのすべてのデータのミラーリングを終了するのを待ってから、プロデューサを起動します。	アプリ開発者
MirrorMaker を起動します。	プロデューサーがターゲットクラスターに移動したら、MirrorMaker を停止します。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

関連リソース

「AWS リソース」

- 「MirrorMaker を使用したクラスターの移行」 (Amazon MSK ドキュメント)
- 「Amazon MSK 移行ラボ」 (AWS Workshop Studio)

その他のリソース

- 「MirrorMaker 2.0」(アパッチカフカ改善提案)
- 「ジオレプリケーション：クラスター間のデータミラーリング」(Apache Kafka ドキュメント)

追加情報

注：このパターンでは、MirrorMaker 2.0 を専用の MirrorMaker クラスターとして Amazon EC2 インスタンスにインストールします。このアカウントの SSH 開発環境 このパターンでは説明していませんが、MirrorMaker 2.0 を Kafka Connect クラスターで実行することもできます。このデプロイオプションでは、スケーリングとメンテナンスを改善する Kafka エコシステム内のフレームワークを使用します。アプリケーションを実行するための関連する設定を含む Kafka Connect クラスターにコネクターをデプロイします。コネクタは、開発やテストではスタンドアロンモードで、本番環境では分散モードで実行できます。詳細については、「Connect クラスターでの MirrorMaker の実行」(Apache Kafka ドキュメント) を参照してください。その他の MirrorMaker 2.0 デプロイオプションの詳細については、「チュートリアル：MirrorMaker 2.0 の実行」(Kafka ドキュメント) を参照してください。

ELK スタックを AWS 上のElasticクラウドに移行する

Battulga Purevragchaa (AWS)、uday reddy、Antony Prasad Thevaraj (AWS) によって作成されました

概要

Elastic は長年にわたりサービスを提供してきました。通常、ユーザーや顧客は自社でオンプレミスで Elastic を管理しています。マネージド型の Elasticsearchサービス である Elastic Cloud は、Elastic Stack (ELK Stack) を利用する方法と、エンタープライズサーチ、オブザーバビリティ、セキュリティ のためのソリューションを提供します。Elasticのソリューションには、ログ、メトリクス、APM(アプリケーションパフォーマンスモニタリング)、SIEM(セキュリティ情報とイベント管理)などのアプリからアクセスできます。機械学習、インデックスライフサイクル管理、Kibana Lens (ドラッグアンドドロップによる視覚化用) などの統合機能を使用できます。

セルフマネージド型のElasticsearchからElastic Cloudに移行すると、Elasticsearchサービスによって以下の処理が行われます。

- 基盤となるインフラストラクチャーのプロビジョニングと管理
- Elasticsearch クラスターの作成と管理
- クラスターのスケールアップとスケールダウン
- アップグレード、パッチ、スナップショットの作成

これにより、他の課題の解決に集中できる時間が増えます。

このパターンは、オンプレミスのElasticsearch 7.13をAmazon Web Services(AWS)のElasticCloud上のElasticsearchに移行する方法を定義します。他のバージョンでは、このパターンで説明されているプロセスを若干変更する必要がある場合があります。詳細については、Elastic 担当者にお問い合わせください。

前提条件と制限

前提条件

- スナップショット用の Amazon Simple Storage Service (Amazon S3) へのアクセス権を持つアクティブな AWS アカウント
- スナップショットデータファイルを Amazon S3 にコピーするための、安全で十分な広帯域幅のプライベートリンク

- Amazon S3 Transfer Acceleration
- データインジェストが、十分な大きさのローカルデータストアまたはリモートストレージ (Amazon S3) に定期的にアーカイブされるようにする Elastic Snapshotポリシー

移行を開始する前に、スナップショットとそれに付随するインデックスの ライフサイクルポリシー のサイズを把握しておく必要があります。詳細については、Elasticにお問い合わせください。

役割とスキル

移行プロセスには、次の表に示す役割と専門知識も必要です。

ロール	専門知識	責任
App Runter サポート	ElasticクラウドとElasticオンプレミスに精通していること	Elasticに関連するすべてのタスク
システム管理者またはデータベース管理者	オンプレミスのElastic環境とその設定に関する深い知識	ストレージのプロビジョニング、AWS コマンドラインインターフェイス (AWS CLI) のインストールと使用、Elasticにオンプレミスでフィードされるすべてのデータソースの特定を行う機能
ネットワーク管理者	オンプレミスと AWS のネットワーク接続、セキュリティ、パフォーマンスに関する知識	接続帯域幅を理解した上で、オンプレミスから Amazon S3 へのネットワークリンクを確立する

機能制限

- Elasticクラウド上のElasticサーチは、サポートされている AWS リージョン (2021 年 9 月) でのみご利用いただけます。

Product versions

- Elasticsearch 7.10

アーキテクチャ

ソーステクノロジースタック

オンプレミスのElasticサーチ 7.13 以降：

- クラススナップショット
- インデックススナップショット
- Beats 設定

ソーステクノロジーアーキテクチャ

次の図は、さまざまな取り込み方法、ノードタイプ、Kibana を使用した一般的なオンプレミスアーキテクチャを示しています。さまざまなノードタイプには、Elasticsearch クラスター、認証、可視化の役割が反映されています。

1. Beats から Logstash へのインジェスト
2. Beats から Apache Kafka メッセージキューへのインジェスト
3. ファイルビートから Logstash への取り込み
4. Beats から Apache Kafka メッセージキューへのインジェスト
5. Logstash から Elasticsearch クラスターへのインジェスト
6. ターゲット Elasticsearch クラスター
7. 認証および通知ノード
8. Kibana ノードとブロブノード

ターゲットテクノロジースタック

Elastic Cloudは、クラスター間のレプリケーションにより、複数のAWS リージョンのサービスとしてのソフトウェア(SaaS)アカウントにデプロイされます。

- クラススナップショット
- インデックススナップショット
- ビートの設定

- Elasticクラウド
- Network Load Balancer
- Amazon Route 53
- Amazon S3

ターゲット アーキテクチャ

マネージド型のElasticクラウドインフラストラクチャは以下のとおりです。

- 可用性が高く、複数の アベイラビリティーゾーン と複数の AWS リージョンに存在します。
- データ (インデックスとスナップショット) は Elastic Cloud のクロスクラスターレプリケーション (CCR) を使用してレプリケートされるため、リージョンに障害が発生しにくい
- アーカイブ (スナップショットはAmazon S3 にアーカイブされるため)
- Network Load Balancer とRoute 53 の組み合わせによるネットワーク分断耐性
- Elastic APM、Beats、Logstashからのデータインジェスト (ただしこれらに限定されない)

高レベル移行ステップ

Elasticは、オンプレミスのElasticクラスターをElasticクラウドに移行するための規範的な方法を独自に開発しました。Elasticの手法は、Well-Architected FrameworkやAWS Migration Acceleration Program(MAP)など、AWS 移行ガイダンスやベストプラクティスと直接連携し、補完するものです。通常、AWS 移行の 3 つのフェーズは次のとおりです。

- 評価
- 準備
- 移行とモダナイズ

Elasticは同様の移行段階を経ていますが、用語は補完的です。

- ジョブの開始*
- 計画
- を実装する
- 配信

- 閉じる

Elasticは、Elastic インプリメンテーション メソドロジーを用いて、プロジェクト成果の実現を促進します。これは、Elastic、コンサルティングチーム、顧客チームが明確に連携して、意図した成果を共同で達成できるようにするために設計上インクルーシブになっています。

Elasticの手法は、従来のウォーターフォールフェーズと実装フェーズのスクラムを組み合わせたものです。技術要件の設定は、リスクを最小限に抑えながら、協調的な方法で繰り返し提供されます。

ツール

AWS サービス

- Amazon Route 53 は、可用性と拡張性に優れたドメインネームシステム (DNS) のウェブサービスです。Route 53 を使用すると、ドメイン登録、DNS ルーティング、ヘルスチェックの 3 つの主要な機能を任意の組み合わせで実行できます。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、オブジェクトストレージサービスです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。このパターンでは、S3 バケットと Amazon S3 Transfer Acceleration を使用します。
- Elastic Load Balancing – Elastic Load Balancing は、受信したトラフィックを複数のアベイラビリティーゾーンの複数のターゲット (EC2 インスタンス、コンテナ、IP アドレスなど) に自動的に分散させます。

その他のツール

- Beats – Beats は Logstash または Elasticsearch からデータを送信します。
- Elastic Cloud – Elasticクラウドは、Elasticsearchをホスティングするためのマネージドサービスです。
- Elasticsearch — Elasticsearch は、Elastic Stack を使用してデータを一元的に保存する検索および分析エンジンで、拡張性の高い検索と分析を可能にします。このパターンでは、スナップショットの作成とクラスター間のレプリケーションも使用されます。
- Logstash — Logstashは、複数のソースからデータを取り込んで変換し、データストレージに送信するサーバー側のデータ処理パイプラインです。

エピック

移行の準備をする

タスク	説明	必要なスキル
オンプレミスのElasticソリューションを実行しているサーバーを特定する。	Elastic への移行がサポートされていることを確認してください。	アプリ所有者
オンプレミスサーバーの構成を理解します。	オンプレミスでワークロードを正常に処理するために必要なサーバー構成を理解するには、現在使用されているサーバーのハードウェアフットプリント、ネットワーク構成、およびストレージ特性を調べてください。	App Runter サポート
ユーザーとアプリのアカウント情報を収集します。	オンプレミスのElastic環境で使用されるユーザー名とアプリ名を特定します。	システム管理者、アプリケーションサポート
Beats とデータシッパーの設定を文書化します。	設定を文書化するには、既存のデータソースとシンクを見てください。詳細については、Elastic Load Balancing ドキュメントを参照してください。	App Runter サポート
データの速度と量を決定します。	クラスターが処理するデータ量のベースラインを設定します。	システム管理者、アプリケーションサポート
RPO と RTO のシナリオを文書化します。	障害およびサービスレベルアグリーメント (SLA) の観点から、目標復旧時点 (RPO) と目	システム管理者、アプリケーションサポート

タスク	説明	必要なスキル
	標復旧時間 (RTO) のシナリオを文書化します。	
最適なスナップショットライフサイクル設定を決定します。	移行中および移行後に Elastic スナップショットを使用してデータを保護する必要がある頻度を定義します。	システム管理者、アプリケーションサポート
移行後のパフォーマンスの期待値を定義します。	現在および予想される画面更新、クエリのランタイム、ユーザーインターフェイスの動作に関するメトリクスを生成します。	システム管理者、アプリケーションサポート
インターネットアクセスの転送、帯域幅、可用性の要件を文書化します。	Amazon S3 にスナップショットをコピーするためのインターネット接続の速度、レイテンシー、耐障害性を確認します。	ネットワーク管理者
Elastic のオンプレミスランタイムの現在のコストを文書化してください。	AWS の対象となる環境のサイジングが、パフォーマンスと費用対効果の両方を考慮して設計されていることを確認してください。	DBA、システム管理者、アプリケーションサポート
認証と認可のニーズを特定します。	Elastic Stackのセキュリティ機能には、ライトウェイトディレクトリアクセスプロトコル(LDAP)、セキュリティアサーションマークアップ言語(SAML)、OpenID Connect(OIDC)などのビルトインレルムが用意されています。	DBA、システム管理者、アプリケーションサポート

タスク	説明	必要なスキル
地理的位置に基づく具体的な規制要件を理解してください。	データは必ず、お客様の要件および関連する国の要件に従ってエクスポートおよび暗号化してください。	システム管理者、アプリケーションサポート

移行を実装する

タスク	説明	必要なスキル
Amazon S3 でステージングエリアを準備します。	Amazon S3 でスナップショットを受け取るには、S3 バケット と、新しく作成したバケットへのフルアクセス権を持つ一時的な AWS Identity and Access Management (IAM) ロールを作成します。詳細については、のIAM ユーザーにアクセス許可を委任するロールの作成を参照してください。または、AWS Security Token Service を使用して一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。アクセスキー ID、シークレットアクセスキー、セッショントークンを安全な場所に保管してください。 バケットで Amazon S3 Transfer Acceleration を有効にする	AWS 管理者

タスク	説明	必要なスキル
AWS CLI と Amazon S3 プラグインをオンプレミスにインストールします。	Elasticsearch の各ノードで、次のコマンドを実行します。 `sudo bin/elasticsearch-plugin install repository-s3` その後、キットが再起動します。	AWS 管理者

Amazon S3 クライアントアクセスを設定します。	以下のコマンドを実行して、以前に作成したキーを追加します。
	`elasticsearch-keystore add s3.client.default.access_key`
	`elasticsearch-keystore add s3.client.default.secret_key`
	`elasticsearch-keystore add s3.client.default.session_token`
		AWS 管理者

Elastic データのスナップショットリポジトリを登録します。	Kibana Dev Tools を使用して、どのリモート S3 バケットに書き込むかをオンプレミスのローカルクラスターに指示します。	AWS 管理者

タスク	説明	必要なスキル
スナップショットポリシーを設定します。	スナップショットのライフサイクル管理を設定するには、Kibana ポリシー タブで SLM ポリシーを選択し、含める時間、データストリーム、インデックス、使用する名前を定義します。 スナップショットを頻繁に作成するポリシーを設定します。スナップショットは段階的に作成され、ストレージを効率的に使用します。適性評価の決定に合わせてください。ポリシーでは、保存ポリシー を指定して、不要になったスナップショットを自動的に削除することもできます。	App Runter サポート
スナップショットが機能することを確認します。	Kibana Dev Tools で、次のコマンドを実行します。 `GET _snapshot/<your_repo_name>/_all`	システム管理者、アプリケーションサポート
新しいクラスターをElasticクラウドにデプロイします。	Elasticにログイン し、レディネスアセスメントで得たビジネス上の検出結果から導き出された「オブザーバビリティ、サーチ、セキュリティ」の対象となるクラスターを選択します。	システム管理者、アプリケーションサポート

タスク	説明	必要なスキル
クラスターキーストアへのアクセスを設定します。	新しいクラスターは、スナップショットを保存する S3 バケットにアクセスする必要があります。Elasticsearch サービスコンソールでセキュリティを選択し、先ほど作成したアクセスキーとシークレット IAM キーを入力します。	AWS 管理者
Amazon S3 にアクセスするようにElastic Cloudホストクラスターを設定します。	Amazon S3 に以前に作成したスナップショットリポジトリへの新しいクラスターアクセスを設定します。Kibana を使用して、次の操作を実行します。 1. Stack Management、Snapshot Settings、RegisterRepoを選択します。 2. エイリアスフィールドに、リポジトリの名前を入力します。 3. S3 クライアント名 には secondary を選択します。 4. 前に作成した S3 バケットをリポジトリに追加します。 5. スナップショットを圧縮を選択します。 6. Encryption 設定(暗号化設定) については、デフォルトのままにしておきます。	システム管理者、アプリケーションサポート

タスク	説明	必要なスキル
新しい Amazon S3 リポジトリを確認します。	Elastic Cloud クラスターでホストされている新しいリポジトリにアクセスできることを確認します。	AWS 管理者

Elasticsearch サービスクラスターを初期化します。	Elasticsearch サービスコンソールで、S3 スナップショットから Elasticsearch サービスクラスターを初期化します。
	以下のいずれかのコマンドを実行します。
	`*/_close?expand_wildcards=all`
	`/_snapshot/<your-repo-name>/ <your-snapshot-name>/_restore`
	`*/_open?expand_wildcards=all`
		App Runter サポート

移行を完了してください。

タスク	説明	必要なスキル
スナップショットの復元が成功したことを確認します。	Kibana Dev Tools で、次のコマンドを実行します。 `GET _cat/indices`	App Runter サポート
取り込みサービスを再デプロイ。	BeatsとLogstashのエンドポイントを新しいElasticsearch	App Runter サポート

タスク	説明	必要なスキル
	サービスのエンドポイント Connect。	

クラスター環境のテストとクリーンアップ

タスク	説明	必要なスキル
クラスター環境を検証します。	オンプレミスの Elastic クラスター環境を AWS に移行したら、その環境に接続し、独自のユーザー承認テスト (UAT) ツールを使用して新しい環境を検証できます。	App Runter サポート
リソースをクリーンアップします。	クラスターが正常に移行されたことを確認したら、S3 バケットと、移行に使用した IAM ロールを削除します。	AWS 管理者

関連リソース

Elastic Inference

- Elasticクラウド
- AWS でのマネージド型Elasticサーチとキバナ
- Elastic エンタープライズサーチ
- Elastic インテグレーション
- 弾性オブザーバビリティ
- Elastic セキュリティ
- ビート
- ElasticAPM
- インデックスライフサイクル管理への移行

- Elasticサブスクリプション
- Elasticにお問い合わせください

Elasticのブログ投稿

- セルフマネージド型のElasticsearchからAWS 上のElasticCloudに移行する方法 (ブログ記事)
- Elasticクラウドへの移行 (ブログ記事)

Elasticドキュメント

- チュートリアル：SLM によるバックアップの自動化
- ILM： インデックスのライフサイクルの管理
- Logstash
- クラスター間レプリケーション(CCR)
- インジェストパイプライン
- Elasticサーチ API リクエストの実行
- スナップショット保持期限

Elasticビデオとウェビナー

- 伸縮自在なクラウド移行
- Elastic Cloud： 顧客が移行する理由 (オンラインセミナー)

AWS リファレンス

- AWS Marketplace でのElasticクラウド
- AWS コマンドラインインターフェイス
- AWS Direct Connect
- Migration Acceleration Program (MAP)
- Network Load Balancers
- リージョンとアベイラビリティーゾーン
- Amazon Route 53
- Amazon Simple Storage Service

- Amazon S3 Transfer Acceleration
- VPN 接続
- Well-Architected フレームワーク

追加情報

複雑なワークロードの移行を計画している場合は、Elasticコンサルティングサービスをご利用ください。設定やサービスに関する基本的な質問がある場合は、ElasticSupport チームにお問い合わせください。

Starburst AWS クラウド を使用してデータを に移行する

作成者: Antony Prasad Thevaraj (AWS)、Shaun Van Staden、Suresh Veeragoni (AWS)

概要

Starburst は、既存のデータソースを 1 つのアクセスポイントにまとめるエンタープライズクエリエンジンを提供することで、Amazon Web Services (AWS) へのデータ移行ジャーニーを加速します。移行計画を最終決定する前に、複数のデータ ソースにわたって分析を実行して、貴重なインサイトを得ることができます。通常どおりの分析を中断することなく、Starburst エンジンまたは専用の抽出、変換、ロード (ETL) アプリケーションを使用してデータを移行できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 仮想プライベートクラウド (VPC)
- Amazon Elastic Kubernetes Service (Amazon EKS) クラスター
- Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling グループ
- 移行する必要のある現行システムワークロードのリスト
- からオンプレミス環境 AWS へのネットワーク接続

アーキテクチャ

リファレンスアーキテクチャ

次の大まかなアーキテクチャ図は、 での Starburst Enterprise の一般的なデプロイを示しています AWS クラウド。

1. Starburst Enterprise クラスターは 内で実行されます AWS アカウント。
2. ユーザーは Lightweight Directory Access Protocol (LDAP) または Open Authorization (OAuth) を使用して認証し、Starburst クラスターと直接やり取りします。
3. Starburst は、Amazon Simple Storage Service (Amazon S3)、Amazon Relational Database Service (Amazon RDS) AWS Glue、Amazon Redshift など、複数の AWS データソースに接続できます。Starburst は、、オンプレミス AWS クラウド、またはその他のクラウド環境のデータソース間でフェデレーティッドクエリ機能を提供します。

4. Starburst Enterprise は、Helm チャートを使用して Amazon EKS クラスターで起動します。
5. Starburst Enterprise は、Amazon EC2 Auto Scaling グループと Amazon EC2 スポットインスタンスを使用してインフラストラクチャを最適化します。
6. Starburst Enterprise は既存のオンプレミスデータソースに直接接続し、データをリアルタイムで読み取ります。さらに、この環境に既存の Starburst Enterprise デプロイがある場合は、 の新しい Starburst クラスター AWS クラウド をこの既存のクラスターに直接接続できます。

以下の点に注意してください。

- Starburst は、データ仮想化プラットフォームではありません。分析用のデータメッシュ戦略の基盤となる SQL ベースの超並列処理 (MPP) クエリエンジンです。
- Starburst を移行の一環としてデプロイすると、既存のオンプレミスインフラストラクチャに直接接続できます。
- Starburst には、さまざまなレガシーシステムへの接続を容易にするエンタープライズコネクタやオープンソースコネクタがいくつか組み込まれています。コネクターとその機能の一覧については、Starburst Enterprise ユーザーガイドの「コネクター」を参照してください。
- Starburst は、オンプレミスのデータソースからデータをリアルタイムでクエリできます。 これにより、データの移行中に通常の業務が中断することを防止できます。
- 既存のオンプレミス Starburst Enterprise デプロイから移行する場合は、特殊なコネクタである Starburst Stargate を使用して、 の Starburst Enterprise クラスターをオンプレミスクラスター AWS に直接接続できます。これにより、ビジネスユーザーとデータアナリストがクエリを からオンプレミス環境にフェデレーションする場合 AWS クラウド に、さらにパフォーマンス上の利点が得られます。

大まかなプロセスの概要

Starburst では、移行前にすべてのデータに関するインサイトを得ることができるため、データ移行プロジェクトを加速できます。次の図は、Starburst を使用してデータを移行する一般的なプロセスを示しています。

ロール

通常、Starburst を使用して移行を完了するには、以下のロールが必要です。

- クラウド管理者 — Starburst Enterprise アプリケーションを実行するためにクラウドリソースを利用できるようにする責任があります。
- Starburst 管理者 — Starburst アプリケーションのインストール、設定、管理、サポートを担当します。
- データエンジニア — 以下の責任を負います。
  - クラウドにレガシーデータを移行する
  - 分析をサポートするセマンティックビューの構築
- ソリューションオーナーまたはシステムオーナー — ソリューション全体の実装を担当

ツール

AWS のサービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Elastic Kubernetes Service (Amazon EKS) を使用すると、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行できます。

その他のツール

- Helm – Helm は、Kubernetes クラスター上でアプリケーションをインストールおよび管理するのに役立つ Kubernetes のパッケージマネージャです。
- Starburst Enterprise – 分析用のデータメッシュ戦略の基盤となる SQL ベースの超parallel 処理 (MPP) クエリエンジンです。
- Starburst Stargate – Starburst Stargate は、オンプレミスデータセンターのクラスターなど、ある Starburst Enterprise 環境のカタログとデータソースを、 のクラスターなど、別の Starburst Enterprise 環境のカタログとデータソースにリンクします AWS クラウド。

エピック

データを評価する

タスク	説明	必要なスキル
データを特定して優先順位を付けます。	移動するデータを特定します。大規模なオンプレミスのレガシーシステムには、移行したくないデータやコンプライアンス上の理由で移動できないデータに加え、移行したいコアデータが含まれる場合があります。データインベントリで、どのデータを最初にターゲットにするか決めるのに役立ちます。詳細については、「自動ポートフォリオ検出の開始方法」を参照してください。	データエンジニア、DBA
データの探索、インベントリ、バックアップを行います。	ユースケースに応じたデータの質、量、関連性を検証します。必要に応じて、データのバックアップまたはスナップショットを作成し、データのターゲット環境を確定します。	データエンジニア、DBA

Starburst Enterprise 環境を設定する

タスク	説明	必要なスキル
で Starburst Enterprise を設定します AWS クラウド。	データをカタログ化している間に、マネージド型の Amazon EKS クラスターに	AWS 管理者、アプリ開発者

タスク	説明	必要なスキル
	Starburst Enterprise を設定します。詳細については、Starburst Enterprise リファレンスドキュメントの「Kubernetes によるデプロイ」を参照してください。これにより、データ移行中も通常どおりの分析が可能になります。	
Starburst をデータソースに接続します。	データを特定し、Starburst Enterprise を設定した後、Starburst をデータソースに接続します。Starburst は SQL クエリとしてデータソースから直接データを読み取ります。 詳細については、Starburst Enterprise リファレンスドキュメントを参照してください。	AWS 管理者、アプリ開発者

データを移行する

タスク	説明	必要なスキル
ETL パイプラインを構築して実行します。	データ移行プロセスを開始します。このアクティビティは、通常どおりの分析と同時に実施できます。移行には、サードパーティ製の製品または Starburst を使用できます。Starburst には、さまざまなソースのデータを読み書きする機能があります。詳細については、Starburst Enterpris	データエンジニア

タスク	説明	必要なスキル
	e リファレンスドキュメントを参照してください。	
データを検証します。	データを移行したら、データを検証して、必要なデータがすべて移動され、変更がないことを確認します。	データエンジニア、DevOps エンジニア

カットオーバー

タスク	説明	必要なスキル
データをカットオーバーします。	データの移行と検証が完了した後、データをカットオーバーできます。これには、Starburst のデータ接続リンクの変更が含まれます。オンプレミスのソースを指定する代わりに、新しいクラウドソースを指定してセマンティックビューを更新します。 詳細については、Starburst Enterprise リファレンスドキュメントの「コネクタ」を参照してください。	データエンジニア、カットオーバーのリーダー
ユーザーにロールアウトします。	データコンシューマーは、移行したデータソースから作業を開始します。このプロセスは、分析のエンドユーザーには表示されません。	カットオーバーのリーダー、データエンジニア

関連リソース

AWS Marketplace

- Starburst Galaxy
- Starburst Enterprise
- Starburst Data JumpStart
- Starburst Enterprise with Graviton

Starburst ドキュメント

- Starburst Enterprise ユーザーガイド
- Starburst Enterprise リファレンスドキュメント

その他の AWS ドキュメント

- 自動ポートフォリオ検出の開始方法 (AWS 規範ガイダンス)
- Starburst を使用したクラウドインフラストラクチャのコストとパフォーマンスの最適化 AWS (AWS ブログ記事)

AWS での入力ファイルサイズの ETL 取り込みを最適化する

Apoorva Patrikar (AWS) によって作成されました

概要

このパターンは、データを処理する前にファイルサイズを最適化することで、AWS Glue 上のビッグデータと Apache Spark ワークロードの抽出、変換、ロード (ETL) プロセスの取り込みステップを最適化する方法を示しています。このパターンを使用して、小さなファイルの問題を防止または解決してください。つまり、サイズの小さいファイルが多数あると、ファイルの合計サイズが原因でデータ処理が遅くなる場合です。たとえば、それぞれがわずか数百キロバイトのファイルが数百個あると、AWS Glue ジョブのデータ処理速度が大幅に低下する可能性があります。これは、AWS Glue が Amazon Simple Storage Service (Amazon S3) で内部リスト機能を実行する必要があり、YARN (さらに別のリソースネゴシエーター) は大量のメタデータを保存する必要があるためです。データ処理速度を向上させるには、グループ化を使用すると、ETL タスクでは入力ファイルのグループを単一のインメモリパーティションに読み取ることができます。このパーティションは、小さいファイルを自動的にグループ化します。または、カスタムコードを使用して既存のファイルにバッチロジックを追加することもできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 1 つ以上の AWS Glue ジョブ
- 1 つ以上のビッグデータまたは Apache Spark ワークロード
- S3 バケット

アーキテクチャ

次のパターンは、さまざまな形式のデータを AWS Glue ジョブで処理し、S3 バケットに保存してパフォーマンスを可視化する方法を示しています。

この図表は、次のワークフローを示しています：

1.
> **Note**
>
> AWS Glue ジョブは、CSV、JSON、Parquet 形式の小さなファイルを動的フレームに変換します。: 入力ファイルのサイズは、AWS Glue ジョブのパフォーマンスに最も大きな影響を与えます。

2. AWS Glue ジョブは S3 バケットの内部リスト機能を実行します。

ツール

- AWS Glue はフルマネージド型の ETL サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

エピック

グループ化を使用して読み取り中の ETL 取り込みを最適化する

タスク	説明	必要なスキル
グループサイズを指定します。	50,000 個を超えるファイルがある場合、デフォルトでグループ化されます。ただし、`connectionOptions` パラメーターでグループサイズを指定することで、50,000 ファイル未満のファイルでもグループ化できます。`connectionOptions` パラメーターは `create_dynamic_frame.from_options` メソッド内にあります。	データエンジニア

タスク	説明	必要なスキル
グループ化コードを記述します。	create_dynamic_frame メソッドを使用して動的フレームを作成します。以下に例を示します。 `S3bucket_node1 = glueContext.create_dynamic_frame.from_options(format_options={"multiline": False}, connection_type="s3", format="json", connection_options={ "paths": ["s3://bucket/prefix/file.json"], "recurse": True, "groupFiles": 'inPartition', "groupSize": 1048576 }, transformation_ctx="S3bucket_node1",)` **Note** groupFiles を使用して、Amazon S3 パーティショングループのファイルをグループ化します。groupSize を	データエンジニア

タスク	説明	必要なスキル
	使用して、メモリに読み込まれるグループのターゲットサイズを設定します。`groupSize` バイト単位で指定します (1048576 = 1 MB)。	
コードをワークフローに追加します。	AWS Glue のジョブワークフロー にグループコードを追加します。	データエンジニア

カスタムロジックを使用して ETL の取り込みを最適化します。

タスク	説明	必要なスキル
言語と処理プラットフォームを選択してください。	ユースケースに合わせたスクリプト言語と処理プラットフォームを選択してください。	クラウドアーキテクト
コードを書き込む	ファイルをまとめてバッチ処理するカスタムロジックを記述します。	クラウドアーキテクト
コードをワークフローに追加します。	AWS Glue のジョブ ワークフロー にコードを追加します。これにより、ジョブを実行するたびにカスタムロジックを適用できます。	データエンジニア

変換後にデータを書き込む際の再パーティション

タスク	説明	必要なスキル
消費パターンを分析します。	ダウンストリームアプリケーションが、書き込むデータをどのように使用するかを確認します。例えば、毎日データをクエリし、リージョンごとにデータをパーティション分割するか、ファイルあたり 2.5 KB などの非常に小さな出力ファイルがある場合、これは消費には適していません。	DBA
書き込み前にデータを再パーティション化します。	処理中 (処理ロジックに基づく) および処理後 (消費に基づく) の結合またはクエリに基づく再パーティション。たとえば、 などのバイトサイズに基づく再パーティション`.repartition(100000)`、または などの列に基づく再パーティションなどです`.repartition("column_name")` 。	データエンジニア

関連リソース

- 大きなグループの入力ファイルの読み取り
- AWS Glue のモニタリング
- Amazon CloudWatch メトリクスを使用した AWS Glue のモニタリング
- ジョブのモニタリングとデバッグ
- AWS Glue でのサーバーレス ETL の使用を開始する

追加情報

ファイルサイズの決定

ファイルサイズが大きすぎるか小さすぎるかを判断する簡単な方法はありません。ファイルサイズが処理パフォーマンスに与える影響は、クラスターの構成によって異なります。コア Hadoop では、ブロックサイズを最大限に活用するために 128 MB または 256 MB のファイルを使用することをお勧めします。

AWS Glue のほとんどのテキストファイルワークロードでは、5 ~ 10 DPU クラスターのファイルサイズを 100 MB から 1 GB の範囲にすることを推奨しています。入力ファイルの最適なサイズを判断するには、AWS Glue ジョブの前処理セクションを監視し、ジョブの CPU 使用率とメモリ使用率を確認します。

追加の考慮事項

ETL の初期段階でのパフォーマンスがボトルネックである場合は、処理前にデータファイルをグループ化またはマージすることを検討してください。ファイル生成プロセスを完全に制御できれば、未加工データを AWS に送信する前に、ソースシステム自体にデータポイントを集約する方がさらに効率的です。

AWS Step Functions を使用して ETL パイプラインを検証、変換、パーティショニングでオーケストレーションします

サンディップ・ガンガパディヤイ (AWS) によって作成されました

概要

このパターンは、パフォーマンスとコストを最適化するために大規模な CSV データセットを検証、変換、圧縮、およびパーティション化するサーバーレスの抽出、変換、ロード (ETL) パイプラインを構築する方法を示しています。パイプラインは AWS Step Functions によってオーケストレートされ、自動再試行、エラー処理、ユーザー通知特徴量が含まれています。

CSV ファイルが Amazon Simple Storage Service (Amazon S3) バケットの出典フォルダにアップロードされると、ETL パイプラインが実行を開始します。パイプラインは、ソース CSV ファイルの内容とスキーマを検証し、CSV ファイルを圧縮された Apache Parquet 形式に変換し、データセットを年、月、日ごとにパーティション化し、分析ツールが処理できるように別のフォルダに保存します。

このパターンを自動化するコードは、GitHub の ETL パイプラインと AWS Step Functions リポジトリにあります。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS アカウントにインストールおよび設定されている AWS コマンドラインインターフェイス (AWS CLI)。これにより、AWS CloudFormation スタックをデプロイして AWS リソースを作成できます。AWS CLI バージョン 2 が推奨されます。(AWS CLI のドキュメントのAWS CLI のインストール、更新とアンインストールを参照してください)。AWS CLI の設定手順については、AWS CLI ドキュメントの 設定と認証情報ファイルの設定 を参照してください。
- Amazon S3 バケット。
- 正しいスキーマの CSV データセット。(このパターンに含まれる コードリポジトリ には、使用できる正しいスキーマとデータ型が記載されたサンプル CSV ファイルが用意されています)。
- AWS マネジメントコンソールでの使用がサポートされているウェブブラウザ。 でサポートされるブラウザのリストについては、 を参照してください。

- AWS Glue; コンソールへのアクセス。
- AWS Step Functions コンソールにアクセスします。

制約事項

- AWS Step Functions では、履歴ログの保持の上限は 90 日間です。詳細については、AWS Step Functions ドキュメントのクォータ と 標準ワークフローのクォータとクォータ を参照してください。

製品バージョン

- AWS Lambda 用の Python 3.11
- AWS Glue バージョン 2.0

アーキテクチャ

この図に示されているワークフローは、以下の大まかなステップで構成されています。

1. ユーザーは CSV ファイルを Amazon S3 のソースフォルダにアップロードします。
2. Amazon S3 通知イベントは、ステップファンクションステートマシンを起動する AWS Lambda 関数を開始します。
3. Lambda 関数は、未加工の CSV ファイルのスキーマとデータ型を検証します。
4. 検証結果に応じて：
   a. ソースファイルの検証が成功すると、ファイルはステージフォルダーに移動してさらに処理されます。
   b. 検証に失敗すると、ファイルはエラーフォルダに移動し、Amazon Simple Notification Service (Amazon SNS) を通じてエラー通知が送信されます。
5. AWS Glue クローラーは Amazon S3 のステージフォルダから未加工ファイルのスキーマを作成します。
6. AWS Glue ジョブは、未加工ファイルを Parquet 形式に変換、圧縮、およびパーティション化します。
7. また、AWS Glue ジョブはファイルを Amazon S3 のトランスフォームフォルダに移動します。

8. AWS Glue クローラーは、変換されたファイルからスキーマを作成します。生成されたスキーマは、どの分析ジョブでも使用できます。 を使用して、Amazon Athena でクエリを実行することができます。
9. パイプラインがエラーなしで完了すると、スキーマファイルはアーカイブフォルダーに移動されます。エラーが発生した場合、ファイルは代わりにエラーフォルダーに移動されます。
10. Amazon SNS は、パイプラインの完了ステータスに基づいて成功または失敗を示す通知を送信します。

このパターンで使用されるすべての AWS リソースはサーバーレスです。管理するサーバーはありません。

ツール

AWS サービス

- AWS Glue — AWS Glue は、分析のためにデータを簡単に準備してロードできるフルマネージドのETLサービスです。
- AWS Step Functionsは、AWS Lambda関数と他のサービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。AWS Step Functions のグラフィカルコンソールでは、アプリケーションのワークフローを一連のイベント駆動型ステップとして確認できます。
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、およびパフォーマンスを提供するオブジェクトストレージサービスです。
- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、マイクロサービス、分散システム、サーバーレスアプリケーションを分離できる、可用性が高く、耐久性があり、安全な、完全マネージド型のパブ/サブメッセージングサービスです。
- AWS Lambda – AWS Lambda はサーバーをプロビジョニングしたり管理しなくてもコードを実行できるコンピューティングサービスです。AWS Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケーリングします。

コード

このパターンのコードは、GitHub 内の「ETL パイプラインと AWS Step Functions」リポジトリで利用できます。コードリポジトリには以下のファイルとフォルダが含まれています。

- `template.yml` — AWS Step Functionsを使用して ETL パイプラインを作成するための AWS CloudFormation テンプレート。
- `parameter.json` — すべてのパラメータとパラメータ値が含まれます。エピック セクションで説明されているように、このファイルを更新してパラメータ値を変更します。
- `myLayer/python` フォルダー — このプロジェクトに必要な AWS Lambda レイヤーを作成するために必要な Python パッケージが含まれています。
- `lambda` フォルダー — 次の Lambda 関数が含まれます。
  - `move_file.py` — ソースデータセットをアーカイブフォルダー、トランスフォームフォルダー、またはエラーフォルダーに移動します。
  - `check_crawler.py` — 失敗メッセージを送信する前に、`RETRYLIMIT` 環境変数で設定された回数だけ AWS Glue クローラーのステータスを確認します。
  - `start_crawler.py` — AWS Glue クローラーを起動します。
  - `start_step_function.py` — AWS Step Functions を開始します。
  - `start_codebuild.py` — AWS CodeBuild プロジェクトを開始します。
  - `validation.py` — 入力された未加工データセットを検証します。
  - `s3object.py` — S3 バケット内に必要なディレクトリ構造を作成します。
  - `notification.py` — パイプラインの最後に成功通知またはエラー通知を送信します。

これらのファイルを使用するには、エピックセクションの指示に従ってください。

エピック

出典ファイルの準備

タスク	説明	必要なスキル
AWS SAM コードリポジトリを複製します。	1. AWS Step Functions リポジトリで ETL パイプラインを開きます。 2. メインリポジトリページのファイルリストの上にある コード を選択し、Clone with HTTPS の下にある URL をコピーします。	開発者

タスク	説明	必要なスキル
	3. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 4. ターミナルまたはコマンドラインプロンプトで、コマンドを実行します。 `git clone <repoURL>` where はステップ 2 でコピーした URL <repoURL> を指します。	

タスク	説明	必要なスキル
パラメータ値を更新します。	リポジトリのローカルコピーで、parameter.json ファイルを編集し、以下のようにデフォルトのパラメータ値を更新します。 • pS3BucketName — データセットを保存する S3 バケットの名前。このバケットはテンプレートによって自動的に作成されます。バケット名はグローバルに一意である必要があります。 • pSourceFolder — ソース CSV ファイルのアップロードに使用される S3 バケット内のフォルダの名前。 • pStageFolder — 処理中にステージング領域として使用される S3 バケット内のフォルダの名前。 • pTransformFolder — 変換および分割されたデータセットの保存に使用される S3 バケット内のフォルダの名前。 • pErrorFolder — 検証できない場合にソース CSV ファイルの移動先となる S3 バケット内のフォルダー。 • pArchiveFolder — ソース CSV ファイルのアップロードに使用される S3	開発者

タスク	説明	必要なスキル
	バケット内のフォルダの名前。 • `pEmailforNotification` — 成功/エラー通知を受信するための有効な E メールアドレス。 • `pPrefix` – AWS Glue クローラ名で使用されるプレフィックス文字列。 • `pDatasetSchema` — ソースファイルの検証対象となるデータセットスキーマ。Cerberus Python パッケージは、ソースデータセットの検証に使用されます。詳細については、Cerberus のウェブサイトを参照してください。	

タスク	説明	必要なスキル
S3 バケットにソースコードをアップロードする。	ETL パイプラインを自動化する CloudFormation テンプレートをデプロイする前に、CloudFormation テンプレートのソースファイルをパッケージ化して S3 バケットにアップロードする必要があります。そのためには、設定済みのプロファイルで以下の AWS CLI コマンドを次のように実行します。 `aws cloudformation package --template-file template.yml --s3-bucket <bucket_name> --output-template-file packaged.template --profile <profile_name>` 各パラメータの意味は次のとおりです。 • `<bucket_name>` は、スタックをデプロイする AWS リージョンにある既存の S3 バケット名です。このバケットは、CloudFormation テンプレートのソースコードパッケージを保存するために使用されます。 • `<profile_name>` は、AWS CLI をセットアップしたときに事前設定した	開発者

タスク	説明	必要なスキル
	有効な AWS CLI プロファイルです。	

スタックを作成します。

タスク	説明	必要なスキル
CloudFormation テンプレートをデプロイする。	CloudFormation テンプレートをデプロイするには、AWS CLI のコマンドを次のように実行します。 `aws cloudformation deploy --stack-name <stack_name> --template-file packaged.template --parameter-overrides file://parameter.json --capabilities CAPABILITY_IAM --profile <profile_name>` 各パラメータの意味は次のとおりです。 • `<stack_name>` は CloudFormation スタックのユニークな識別子です。 • `<profile-name>` は事前設定された AWS CLI プロファイルです。	開発者
進捗確認。	AWS CloudFormation コンソール で、スタック開発の進捗状況を確認します。ステー	開発者

タスク	説明	必要なスキル
	タスが CREATE_COMPLETE の場合、スタックは正常にデプロイされています。	
AWS Glue データベース名を書き留めておきます。	スタックの Outputs タブには、AWS Glue データベースの名前が表示されます。キー名は、GlueDBOutput です。	開発者

パイプラインを削除します。

タスク	説明	必要なスキル
ETL パイプラインを開始します。	1. S3 source バケット内のソースフォルダ (または parameter.json ファイルに設定したフォルダ名) に移動します。 2. サンプル CSV ファイルをこのフォルダにアップロードします。(コードリポジトリには、使用できる Sample_Bank_Transaction_Raw_Dataset.csv というサンプルファイルが用意されています)。ファイルをアップロードすると、Step Functions を通じて ETL パイプラインが開始されます。 3. Step Functions コンソールで ETL パイプラインのステータスを確認します。	開発者

タスク	説明	必要なスキル
パーティション化されたデータセットを確認します。	ETL パイプラインが完了したら、パーティション化されたデータセットが Amazon S3 トランスフォームフォルダ (`transform` または `parameter.json` ファイルに設定したフォルダ名) にあることを確認します。	開発者
パーティション化された AWS Glue データベースを確認します。	1. AWS Glue コンソール で、スタックによって作成された AWS Glue データベース (これは前のエピックでメモしたデータベースです) を選択します。 2. AWS Glue データカタログでパーティションテーブルが使用できることを確認します。	開発者
クエリを実行する。	(オプション) Amazon Athena を使用して、パーティション分割され変換されたデータベースでアドホッククエリを実行します。手順については、AWS ドキュメントの Amazon Athena を使用した SQL クエリの実行 を参照してください。	データベースアナリスト

トラブルシューティング

問題	ソリューション
AWS Glue ジョブとクローラの AWS Identity and Access Management (IAM) アクセス許可 AWS Glue	AWS Glue ジョブまたはクローラをさらにカスタマイズする場合は、AWS Glue ジョブで使用される IAM ロールで適切な IAM アクセス許可を付与するか、AWS Lake Formation にデータアクセス許可を付与してください。詳細については、AWS ドキュメントを参照してください。

関連リソース

AWS のドキュメント

- AWS Step Functions
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SNS

追加情報

次の図は、Step Functions Inspectorパネル からの ETL パイプラインを成功させるための AWS Step Functions ワークフローを示しています。

以下の図は、入力検証エラーが原因で失敗した ETL パイプラインの AWS Step Functions ワークフローを、Step Functions Inspector パネルから示しています。

Amazon Redshift ML機械学習を使用して高度な分析を実行する

作成者: Po Hong (AWS) と Chyanna Antonio (AWS)

概要

Amazon Web Services (AWS) クラウドでは、Amazon Redshift 機械学習 (Amazon Redshift ML) を使用して、Amazon Redshift クラスターまたは Amazon Simple Storage Service (Amazon S3) に保存されているデータに対して ML 分析を実行できます。Amazon Redshift ML は、高度な分析によく使用される教師あり学習をサポートしています。Amazon Redshift ML のユースケースには、収益予測、クレジットカード詐欺検知、顧客生涯価値 (CLV) または顧客解約予測などがあります。

Amazon Redshift ML を使用すると、データベース ユーザーは、使い慣れた 標準SQLコマンドを使用して、ML モデルを簡単に作成、トレーニング、デプロイできます。Amazon Redshift ML は Amazon SageMaker Autopilot を使用して、制御と可視性を保持しながら、データに基づいてリグレッションまたは回帰に最適な機械学習モデルを自動的にトレーニングおよび調整します。

Amazon Redshift、Amazon S3、Amazon SageMaker 間のやり取りはすべて抽象化され、自動化されています。ML モデルをトレーニングしてデプロイすると、Amazon Redshift で ユーザー定義関数 (UDF) として使用できるようになり、SQL クエリで使用できるようになります。

このパターンは、AWS ブログのAmazon Redshift ML で SQL を使用して Amazon Redshift で ML モデルを作成、トレーニング、デプロイする、および入門リソースセンター の Amazon SageMaker による ML モデルの構築、トレーニング、デプロイ チュートリアルを補完するものです。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon Redshift テーブル内の既存のデータ

スキル

- 機械学習、トレーニング や 予測など、Amazon Redshift ML で使用される用語や概念に精通している 詳細については、Amazon Machine Learning (Amazon ML) ドキュメントの tトレーニングML モデル を参照してください。

- Amazon Redshift のユーザーセットアップ、アクセス管理、および標準 SQL 構文に関する経験。こちらに関する詳細については、Amazon Redshift のドキュメントの Getting started with Amazon Redshift を参照下さい。
- Amazon S3 および AWS Identity and Access Management (IAM) に関する知識と経験
- AWS コマンドラインインターフェイス (AWS CLI) でコマンドを実行した経験も役立ちますが、必須ではありません。

機能制限

- Amazon Redshift クラスターおよび Amazon S3 バケットが、同じ AWS リージョンに存在する必要があります。
- このパターンのアプローチは、リグレッション、二項分類、多クラス分類などの教師あり学習モデルのみをサポートします。

アーキテクチャ

以下のステップでは、Amazon Redshift ML が SageMaker と連携して ML モデルを構築、トレーニング、デプロイする方法について説明します。

1. Amazon Redshift はトレーニングデータを S3 バケットにエクスポートします。
2. SageMaker Autopilot は、トレーニングデータを自動的に前処理します。
3. `CREATE MODEL` ステートメントが呼び出されると、Amazon Redshift ML はトレーニングに SageMaker を使用します。
4. SageMaker Autopilot は、評価メトリクスを最適化する ML アルゴリズムと最適なハイパーパラメータを検索して推奨します。
5. Amazon Redshift は、予測関数を SQL 関数として Amazon Redshift クラスターに登録します。
6. ML モデルの関数は、SQL ステートメントで使用できます。

テクノロジースタック

- Amazon Redshift
- SageMaker
- Amazon S3

ツール

- Amazon Redshift – Amazon Redshift は、エンタープライズレベル、ペタバイト規模、フルマネージド型のデータウェアハウスサービスです。
- Amazon Redshift ML – Amazon Redshift 機械学習 (Amazon Redshift ML) は、どの技術レベルのアナリストやデータサイエンティストでも、機械学習のテクノロジーを簡単に使用できる堅牢なクラウドベースのサービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。
- Amazon SageMaker – Amazon SageMaker は、フルマネージド型の機械学習サービスです。
- Amazon SageMaker Autopilot – Amazon SageMaker Autopilot は、自動機械学習 (AutoML) プロセスの主要タスクを自動化する機能セットです。

コード

Amazon Redshift では、以下のコードを使用してスーパーバイズド ML モデルを作成できます。

```
“CREATE MODEL customer_churn_auto_model
FROM (SELECT state,
             account_length,
             area_code,
             total_charge/account_length AS average_daily_spend,
             cust_serv_calls/account_length AS average_daily_cases,
             churn
      FROM customer_activity
      WHERE  record_date < '2020-01-01'
     )
TARGET churn
FUNCTION ml_fn_customer_churn_auto
IAM_ROLE 'arn:aws:iam::XXXXXXXXXXXX:role/Redshift-ML'
SETTINGS (
  S3_BUCKET 'your-bucket'
);”)
```

> **Note**
>
> SELECT 状態は、Amazon Redshift の通常のテーブル、Amazon Redshift Spectrum の外部テーブル、またはその両方を参照できます。

エピック

トレーニングデータセットとテストデータセットの準備

タスク	説明	必要なスキル
トレーニングデータセットとテストデータセットの準備	AWS マネジメントコンソールにサインインして、Amazon SageMaker コンソールを開きます。機械学習モデルの構築、トレーニング、デプロイ チュートリアルの指示に従って、ラベル列 (教師ありトレーニング) があり、ヘッダーがない.csv ファイルまたは Apache Parquet ファイルを作成します。 **Note** raw データセットをシャッフルして、モデルのトレーニング用のトレーニングセット (70%) とモデルのパフォーマンス評価用のテストセット (30%) に分割することをお勧めします。	データサイエンティスト

テクノロジースタックの準備と設定を行います。

タスク	説明	必要なスキル
Amazon Redshift クラスターを作成します。	Amazon Redshift コンソールで、要件に従ってクラスターを作成します。詳細については、Amazon Redshift ドキュメントのクラスターの作成を参照してください。 **Important** Amazon Redshift クラスターは、`SQL_PREVIEW` メンテナンストラックで作成する必要があります。トラックプレビューの詳細については、クラスターメンテナンストラックの選択Amazon Redshiftドキュメント内のを参照してください。	DBA、クラウドアーキテクト
トレーニングデータとモデルアーティファクトを保存する S3 バケットを作成する。	Amazon S3 コンソールで、トレーニングデータとテストデータ用の S3 バケットを作成します。S3 バケットを作成する方法の詳細については、AWS Quick StartsからS3 バケットを作成するを参照してください。	DBA、クラウドアーキテクト

タスク	説明	必要なスキル
	Important Amazon Redshift クラスターと S3 バケットが同じリージョンにあることを確認します。	
IAM サービスロールを作成して Amazon Redshift クラスターにアタッチします。	Amazon Redshift クラスターが SageMaker および Amazon S3 にアクセスできるようにする IAM ポリシーを作成します。手順と手順については、Amazon Redshift ドキュメントの Amazon Redshift ML を使用するためのクラスターセットアップ を参照してください。	DBA、クラウドアーキテクト
Amazon Redshift ユーザーとグループがスキーマとテーブルにアクセスできるようにします。	Amazon Redshift のユーザーとグループが内部および外部のスキーマとテーブルにアクセスできるようにするアクセス権限を付与します。手順と手順については、Amazon Redshift ドキュメントの アクセス権限と所有権の管理 を参照してください。	DBA

Amazon Redshift で ML モデルを作成してトレーニングする

タスク	説明	必要なスキル
Amazon Redshift で ML モデルを作成してトレーニングする	Amazon Redshift で ML モデルを作成してトレーニングする 詳細については、Amazon Redshift `CREATE MODEL` のドキュメントのステートメントを参照してください。	アプリ開発者、データサイエンティスト

Amazon Redshift でバッチ推論と予測を実行する

タスク	説明	必要なスキル
生成された ML モデル関数を使用して推論を行います。	生成された ML モデル関数を使用して推論を実行する方法の詳細については、Amazon Redshift ドキュメントの 予測 を参照してください。	データサイエンティスト、ビジネスインテリジェンスユーザー

関連リソース

トレーニングデータセットとテストデータセットの準備

- Amazon SageMaker による機械学習モデルの構築、トレーニング、デプロイ

テクノロジースタックの準備と設定を行います

- Amazon Redshift クラスターを作成します
- Amazon Redshift クラスターメンテナンストラックの選択
- 「S3 バケットの作成」
- Amazon Redshift ML 用に Amazon Redshift クラスターをセットアップする必要があります
- Amazon Redshift でのアクセス権限と所有権の管理

Amazon Redshift で ML モデルを作成してトレーニングする

- Amazon Redshift でのモデルステートメントの作成

Amazon Redshift でバッチ推論と予測を実行する

- Amazon Redshift での予測

その他のリソース

- Amazon Redshift 機械学習の開始方法
- SQL と Amazon Redshift ML を使用して Amazon Redshift で ML モデルを作成、トレーニング、デプロイする
- Amazon Redshift パートナー
- AWS 機械学習コンピテンシーパートナー

Amazon Athena を使用して SQL で Amazon DynamoDB テーブルをクエリする Amazon Athena

作成者: Gavin Perrie (AWS)、Ajit Ambike (AWS)、Brad Yates (AWS)

概要

データに Amazon Simple Storage Service (Amazon S3) 以外のソースが含まれている場合は、フェデレーティッドクエリを使用して、それらのリレーショナル、非リレーショナル、オブジェクト、またはカスタムデータソースにアクセスできます。このパターンは、SQL データソースコネクタを使用して Amazon Athena から Amazon DynamoDB へのフェデレーションクエリアクセスを設定する方法を示しています。

このパターンを使用すると、次のことができます。

- SQL を使用して DynamoDB テーブルをクエリします。
- Athena でフェデレーティッド SQL クエリを実行し、DynamoDB テーブルをサポートされている他のデータソースに結合します。

前提条件と制限

前提条件

- DynamoDB 表。
- Athena エンジンバージョン 2 を使用するように設定された Athena ワークグループ。手順については、Athena のドキュメントを参照してください。
- `AthenaDynamoDBConnector` AWS Lambda 関数がデータを流出できる S3 バケット。S3 バケットと Lambda 関数は、同じ AWS リージョンに存在する必要があります。

Athena に初めてアクセスする場合は、クエリ結果の場所として使用する追加の S3 バケットが必要です。手順については、Athena のドキュメントを参照してください。

制約事項

- INSERT INTOなどの書き込みオペレーションはサポートされていません。

製品バージョン

- GitHub での Athena クエリフェデレーションのリリース

アーキテクチャ

ターゲット アーキテクチャ

次の図は、パターンが確立された後の接続フローを示しています。ユーザーは Amazon Athena に接続してクエリを提供します。Athena はクエリとターゲットを DynamoDB データソースコネクタ Lambda 関数に渡します。Lambda 関数はデータを取得して Athena に返します。大量のデータが返された場合、Athena はデータセット全体をパッケージ化して返す前に、一時的な結果をスピルバケットに保存します。

ツール

AWS のサービス

- Amazon Athenaは、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析できるようにするインタラクティブなクエリサービスです。このパターンでは、Amazon Athena クエリフェデレーション SDK を使用して構築され、 を介してアプリケーションとしてインストールされたツールである Amazon Athena DynamoDB Connector を使用します AWS Serverless Application Repository。 Amazon Athena AWS Lambda
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このパターンのコードは、GitHub Athena クエリフェデレーションリポジトリで入手できます。

エピック

DynamoDB データソースコネクタのセットアップとテスト

タスク	説明	必要なスキル
AthenaDynamoDBConnector アプリケーションをデプロイします。	AthenaDynamoDBConnecter をデプロイするには、次の手順を実行します。 1. にサインインし AWS Management Console、DynamoDB テーブルとスピルバケット AWS リージョン に使用している を選択します。 2. サーバーレスアプリケーションリポジトリを「https://https://console.aws.amazon.com/serverlessrepo/.com」で開きます。 3. ナビゲーションペインで、使用可能なアプリケーションを選択します。 4. AWS Identity and Access Management (IAM) アクセスの場合は、検索バーで、カスタム IAM ロールまたはリソースポリシーを作成するアプリを表示するチェックボックスをオンにします。 5. AthenaDynamoDBConnector を検索して選択し、リストされている作成者が	AWS DevOps

タスク	説明	必要なスキル
	Amazon Athena フェデレーションであることを確認します。 6. アプリケーション設定で、次の値を入力します。 • SpillBucket – 関数がデータをスピルできる場所。 • AthenaCatalogName – 作成される Lambda 関数の名前。この名前は、Athena のデータソース名としても使用されます。 7. チェックボックスをオンにして、IAM ロールとポリシーの作成を承認します。 8. [デプロイ] を選択します。	

タスク	説明	必要なスキル
Athena のデータソースを作成します。	データソースを作成するには、次の手順を実行します。 1. で Athena コンソールを開きます。 https://console.aws.amazon.com/athena/ 「」。 2. ナビゲーションペインを展開し、データソースを選択します。 3. [データソースを作成] を選択します。 4. Amazon DynamoDB を選択します。 5. データソース名を入力します。 6. 作成した Lambda 関数を選択します。 7. 詳細を確認し、データソースの作成を選択します。	AWS DevOps

タスク	説明	必要なスキル
Athena を使用して DynamoDB テーブルをクエリします。	DynamoDB テーブルをクエリするには、以下を実行します。 1. Athena コンソールで、ナビゲーションペインを展開し、クエリエディタを選択します。 2. データソースドロップダウンリストで、作成したデータソースを選択します。 3. DynamoDB テーブルがテーブルにリストされていることを確認します。 4. クエリを実行します。	アプリ開発者

トラブルシューティング

問題	ソリューション
クエリは で失敗しますGENERIC_INTERNAL_ERROR: The bucket is in this region: <region>。	Athena スピルバケットと Lambda 関数が同じに作成されていることを確認します AWS リージョン。
新しく作成されたデータソースは Athena コンソールに表示されません。	Athena データカタログはリージョン別です。AthenaDynamoDBConnector が Athena を使用しようとしているリージョンにデプロイされていることを確認します。
新しく作成されたデータソースに対してクエリを実行できません。	クエリ結果の場所が設定されていることを確認します。

関連リソース

- Amazon Athena DynamoDB コネクタ
- Amazon Athena フェデレーティッドクエリ

Athena による Amazon DynamoDB テーブルへのアクセス、クエリ、結合

作成者: Moinul Al-Mamun (AWS)

概要

このパターンは、Amazon Athena DynamoDB コネクタを使用して Amazon Athena と Amazon DynamoDB 間の接続をセットアップする方法を示しています。コネクタは AWS Lambda 関数を使用して DynamoDB 内のデータをクエリします。接続を設定するコードはありません。接続が確立されたら、「Athena フェデレーテッドクエリ」を使用して Athena から SQL コマンドを実行することで、DynamoDB テーブルにすばやくアクセスして分析できます。また、1 つ以上の DynamoDB テーブルを相互に結合したり、Amazon Redshift や Amazon Aurora などの他のデータソースに結合したりすることもできます。

前提条件と制限

前提条件

- DynamoDB テーブル、Athena データソース、Lambda、AWS Identity and Access Management (IAM) ロールを管理するアクセス許可があるアクティブな AWS アカウント
- Athena がクエリ結果を保存できる Amazon Simple Storage Service (Amazon S3) バケット
- Athena DynamoDB コネクタがデータを短期的に保存できる S3 バケット
- 「Athena エンジンバージョン 2」をサポートする AWS リージョン
- Athena と必要な S3 バケットにアクセスするための IAM 権限
- 「Amazon Athena DynamoDB コネクタ」、インストール済み

制約事項

DynamoDB テーブルのクエリにはコストがかかります。テーブルサイズが数ギガバイト (GB) を超えると、高いコストが発生する可能性があります。テーブル全体の SCAN 操作を実行する前に、コストを考慮することをお勧めします。詳細については、「Amazon DynamoDB 料金」を参照してください。コストを削減して高いパフォーマンスを実現するには、クエリでは常に LIMIT を使用することをお勧めします (例: `SELECT * FROM table1 LIMIT 10`)。また、運用環境で JOIN または GROUP BY クエリを実行する前に、テーブルのサイズを検討してください。テーブルが大きすぎる場合は、「テーブルを Amazon S3 に移行する」などの代替オプションを検討してください。

アーキテクチャ

次の図は、ユーザーが Athena から DynamoDB テーブルで SQL クエリを実行する方法を示しています。

この図表は、次のワークフローを示しています：

1. DynamoDB テーブルにクエリを実行するには、ユーザーは Athena から SQL クエリを実行します。
2. Athena は Lambda 関数を開始します。
3. Lambda 関数は、DynamoDB テーブル内のリクエストされたデータをクエリを行います。
4. DynamoDB はリクエストされたデータを Lambda 関数に返します。次に、この関数は Athena を介してクエリ結果をユーザーに転送します。
5. Lambda 関数は S3 バケットにデータを保存します。

テクノロジースタック

- Amazon Athena
- Amazon DynamoDB
- Amazon S3
- AWS Lambda

ツール

- 「Amazon Athena」はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接分析できます。
- 「Amazon Athena DynamoDB コネクタ」は、Athena テナが DynamoDB に接続し、SQL クエリを使用してテーブルにアクセスできるようにする AWS ツールです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

エピック

サンプル DynamoDB テーブルを作成する

タスク	説明	必要なスキル
1 つ目のサンプルテーブルを作成します。	1. AWS マネジメントコンソールにサインインし、「DynamoDB コンソール」を開きます。 2. [Create table (テーブルの作成)] を選択します。 3. [テーブル名] に「dydbtable1」と入力します。 4. パーティションキーには「PK1」と入力します。 5. [ソートキー] には「SK1」と入力します。 6. [Table settings (テーブルの設定)] セクションで、[Customize settings (設定のカスタマイズ)] を選択します。 7. 「テーブルクラス」セクションで、[DynamoDB 標準] を選択します。 8. 「読み取り/書き込みキャパシティ設定」セクションの [キャパシティモード] で [オンデマンド] を選択します。 9. 「保存時の暗号化」セクションで、[Amazon DynamoDB が所有] を選択します。	開発者

タスク	説明	必要なスキル
	10[Create table (テーブルの作成)] を選択します。	

最初のテーブルにサンプルデータを挿入します。	1. DynamoDB コンソールを開きます。
	2. ナビゲーションペインで [テーブル] を選択し、[名前] 列でテーブルを選択します。
	3. [アクション]、[アイテムの作成] の順に選択します。
	4. [JSON ビュー] を選択します。
	5. [属性] エディターのタイトルバーで、[DynamoDB JSON を表示] をオフにします。
	6. [属性] エディタで、次のサンプルデータを 1 つずつ入力します。
	
	`{ "PK1": "1234", "SK1": "info", "Salary": "5000" }`
	
	`{ "PK1": "1235", "SK1": "info", "Salary": "5200" }`
		開発者

タスク	説明	必要なスキル
2 番目のサンプルテーブルを作成します。	1. DynamoDB コンソールを開きます。 2. [Create table] を選択します。 3. [テーブル名] に「dydbtable2」と入力します。 4. パーティションキーには「PK2」と入力します。 5. [ソートキー] には「SK2」と入力します。 6. [Table settings (テーブルの設定)] セクションで、[Customize settings (設定のカスタマイズ)] を選択します。 7. 「テーブルクラス」セクションで、[DynamoDB 標準] を選択します。 8. 「読み取り/書き込みキャパシティ設定」セクションの [キャパシティモード] で [オンデマンド] を選択します。 9. 「保存時の暗号化」セクションで、[Amazon DynamoDB が所有] を選択します。 10. [Create table (テーブルの作成)] を選択します。	開発者

タスク	説明	必要なスキル

2 番目のテーブルにサンプルデータを挿入します。	1. DynamoDB コンソールを開きます。
	2. ナビゲーションペインで [テーブル] を選択し、[名前] 列でテーブルを選択します。
	3. [アクション]、[アイテムの作成] の順に選択します。
	4. [属性] エディターのタイトルバーで、[DynamoDB JSON を表示] をオフにします。
	5. [属性] エディタで、次のサンプルデータを 1 つずつ入力します。
	
	`{ "PK2": "1234", "SK2": "bonus", "Bonus": "500" }`
	
	`{ "PK2": "1235", "SK2": "bonus", "Bonus": "1000" }`
		開発者

Athena で DynamoDB 用のデータソースを作成する

タスク	説明	必要なスキル
データソースコネクタを設定します。	DynamoDB のデータソースを作成し、そのデータソースに接続する Lambda 関数を作成します。 1. AWS マネジメントコンソールにサインインして、「Athena コンソール」を開きます。 2. ナビゲーションペインで [データソース] を選択してから、[データソースの作成] を選択します。 3. [Amazon DynamoDB] データソースを選択し、[次へ] を選択します。 4. 「データソースの詳細」セクションの [データソース名] に「TestDynamoDB」と入力します。 5. 接続の詳細セクションで、すでにデプロイされている Lambda 関数を選択するか、このパターンに使用する Lambda 関数がない場合は [Lambda 関数を作成] を選択します。注: Lambda 関数の作成の詳細については、Lambda 開発者ガイドの「Lambda の使用開始」を参照してください。	開発者

タスク	説明	必要なスキル
	6. (オプション) [Lambda 関数の作成] を選択した場合は、スタックをデプロイする前に Java アプリケーションに含まれている AWS CloudFormation テンプレートを設定する必要があります。テンプレートには、ApplicationName、SpillBucket、AthenaCatalogName、およびその他のアプリケーション設定が含まれています。注:この Java ベースのアプリケーションをデプロイすると、スタックは Athena が DynamoDB と通信できるようにする Lambda 関数を作成します。これにより、SQL コマンドを使用してテーブルにアクセスできるようになります。 7. Lambda 関数をデプロイします。 8. [Next (次へ)] を選択します。	

タスク	説明	必要なスキル
Lambda 関数が S3 スピルバケットにアクセスできることを確認します。	1. Lambdaのコンソールを開きます。 2. ナビゲーションペインで [関数] を選択し、先ほど作成した関数を選択する。 3. [設定] タブを選択します。 4. 左側のペインで [環境変数] を選択し、キーの値が `spill_bucket` であることを確認します。 5. 左側のペインで [権限] を選択し、次に「実行ロール」セクションで、アタッチされている IAM ロールを選択します。注: IAM コンソールの Lambda 関数にアタッチされている IAM ロールに誘導されます。 6. `spill_bucket` バケットへの書き込み権限があることを確認します。 エラーが発生した場合は、このパターンの「追加情報」セクションでガイダンスを参照してください。	開発者

Athena から DynamoDB テーブルにアクセスする

タスク	説明	必要なスキル
DynamoDB テーブルに対してクエリを実行します。	1. AWS マネジメントコンソールにサインインして、	開発者

タスク	説明	必要なスキル
	「Athena コンソール」を開きます。 2. ナビゲーションペインで [データソース] を選択してから、[データソースの作成] を選択します。 3. ナビゲーションペイン内で [Query editor (クエリエディタ)] を選択します。 4. [エディター] タブの「データ」セクションの [データソース] で、お客様のデータソースを選択します。 5. [Database] (データベース) で、データベースを選択します。 6. クエリ 1 には、次のクエリを入力します。`SELECT * FROM dydbtable1 t1;` 7. [実行] を選択し、テーブル内の出力を確認します。 8. クエリ 2 には、次のクエリを入力します。`SELECT * FROM dydbtable2 t2;` 9. [実行] を選択し、テーブル内の出力を確認します。	

タスク	説明	必要なスキル
2 つの DynamoDB テーブルを結合します。	DynamoDB は NoSQL データストアであり、SQL 結合オペレーションをサポートしていません。そのため、次の 2 つの DynamoDB テーブルで結合操作を実行する必要があります。 1. 新しいクエリを作成するには、プラスアイコンを選択します。 2. クエリ 3 には、次のクエリを入力します。 `SELECT pk1, salary, bonus FROM dydbtable1 t1 JOIN dydbtable2 t2 ON t1.pk1 = t2.pk2;`	開発者

関連リソース

- 「Amazon Athena DynamoDB コネクタ」(AWS ラボ)
- 「Amazon Athena の新しいフェデレーテッドクエリ」(AWS ビッグデータブログ) であらゆるデータソースをクエリできます
- 「Athena エンジンバージョンリファレンス」(Athena ユーザーガイド)
- 「AWS Glue と Amazon Athena を使用して Amazon DynamoDB データの抽出と分析を簡素化」(AWS データベースブログ)

追加情報

Athena のクエリで `spill_bucket` を `{bucket_name}/folder_name/` という形式で実行すると、次のエラーメッセージが表示されることがあります。

```
"GENERIC_USER_ERROR: Encountered an exception[java.lang.RuntimeException] from your
 LambdaFunction[arn:aws:lambda:us-east-1:xxxxxx:function:testdynamodb] executed in
 context[retrieving meta-data] with message[You do NOT own the spill bucket with the
 name: s3://amzn-s3-demo-bucket/athena_dynamodb_spill_data/]
This query ran against the "default" database, unless qualified by the query. Please
 post the error message on our forum  or contact customer support with Query Id:
 [query-id]"
```

このエラーを解決するには、Lambda 関数の環境変数 spill_bucket を {bucket_name_only} に更新し、バケット書き込みアクセス用の次の Lambda IAM ポリシーを更新します。

```
{
             "Action": [
                 "s3:GetObject",
                 "s3:ListBucket",
                 "s3:GetBucketLocation",
                 "s3:GetObjectVersion",
                 "s3:PutObject",
                 "s3:PutObjectAcl",
                 "s3:GetLifecycleConfiguration",
                 "s3:PutLifecycleConfiguration",
                 "s3:DeleteObject"
             ],
             "Resource": [
                 "arn:aws:s3:::spill_bucket",
                 "arn:aws:s3:::spill_bucket/*"
             ],
             "Effect": "Allow"
         }
```

または、前に作成した Athena データソースコネクタを削除し、spill_bucket のために {bucket_name} のみを使用して再作成することもできます。

組織間でデータを共有するための最小限の実行可能なデータスペースを設定する

作成者: Ramy Hcini (Think-it)、Ismail Abdellaoui (Think-it)、Malte Gasseling (Think-it)、Jorge Hernandez Suarez (AWS)、Michael Miller (AWS)

概要

データスペースは、データ交換のためのフェデレーティッドネットワークであり、データの中核となる原則として信頼と制御を行います。これにより、組織は費用対効果が高く、テクノロジーに依存しないソリューションを提供することで、大規模なデータの共有、交換、コラボレーションが可能になります。

データスペースは、関連するすべての利害関係者が関与するend-to-endのアプローチでデータ駆動型の問題解決を使用することで、持続可能な未来に向けた取り組みを大幅に推進できる可能性があります。

このパターンは、2 つの企業が Amazon Web Services (AWS) のデータスペーステクノロジーを使用して炭素排出量削減戦略を前進させる方法の例を示しています。このシナリオでは、X 社は Y 社が消費する炭素排出量データを提供します。以下のデータスペース仕様の詳細については、「追加情報」セクションを参照してください。

- 参加者
- ビジネスケース
- データスペースの権限
- データスペースコンポーネント
- データスペースサービス
- 交換するデータ
- データモデル
- Tractus-X " コネクタ

このパターンには、以下のステップが含まれます。

- 2 人の参加者が実行されている基本的なデータスペースに必要なインフラストラクチャをデプロイします AWS。

- コネクタを安全な方法で使用して、炭素排出量と強度のデータを交換します。

このパターンは、Amazon Elastic Kubernetes Service (Amazon EKS) を介してデータスペースコネクタとそのサービスをホストする Kubernetes クラスターをデプロイします。

Eclipse Dataspace Components (") コントロールプレーンとデータプレーンはどちらも Amazon EKS にデプロイされます。公式の Tractus-X Helm チャートでは、PostgreSQL および HashiCorp Vault サービスを依存関係としてデプロイします。

さらに、ID サービスは Amazon Elastic Compute Cloud (Amazon EC2) にデプロイされ、最小実行可能なデータスペース (MVDS) の実際のシナリオをレプリケートします。

前提条件と制限

前提条件

- 選択した にインフラストラクチャをデプロイ AWS アカウント するアクティブな AWS リージョン
- 技術ユーザーとして一時的に使用される Amazon S3 にアクセスできる AWS Identity and Access Management (IAM) ユーザー (現在、 の " コネクタはロールの使用をサポートしていません。 このデモ専用の IAM ユーザーを 1 人作成し、このユーザーには制限されたアクセス許可が関連付けられることをお勧めします) 。
- 選択した に AWS Command Line Interface （ AWS CLI ） がインストールされ、設定されている AWS リージョン
- AWS セキュリティ認証情報
- ワークステーションの eksctl
- ワークステーションの Git
- kubectl
- Helm
- ポストマン
- AWS Certificate Manager (ACM) SSL/TLS 証明書
- Application Load Balancer を指す DNS 名 (DNS 名は ACM 証明書でカバーされている必要があります)
- HashiCorp Vault (AWS Secrets Manager を使用してシークレットを管理する方法については、「追加情報」セクションを参照してください) 。

製品バージョン

- AWS CLI バージョン 2 以降
- Postman Collection v2.1

制約事項

- コネクタの選択 – このデプロイでは、「" ベースのコネクタ」を使用します。ただし、デプロイの特定のニーズに合った情報に基づいた決定を行うために、必ず " コネクタと FIWARE True コネクタの両方の長所と機能を考慮してください。
- 「」 のコネクタ構築 - 選択したデプロイソリューションは、十分に確立され、広範囲にテストされたデプロイオプションである Tractus-X の「コネクタ Helm チャート」に依存しています。このグラフを使用するかどうかの決定は、一般的な使用法と、提供されたビルドに不可欠な拡張機能を含めることによって決まります。PostgreSQL と HashiCorp Vault はデフォルトのコンポーネントですが、必要に応じて独自のコネクタビルドをカスタマイズできます。
- プライベートクラスターアクセス - デプロイされた EKS クラスターへのアクセスはプライベートチャネルに制限されます。クラスターとのやり取りは、 `kubectl`および IAM の使用によってのみ実行されます。クラスターリソースへのパブリックアクセスは、ロードバランサーとドメイン名を使用して有効にできます。ロードバランサーとドメイン名は、特定のサービスをより広範なネットワークに公開するために選択的に実装する必要があります。ただし、パブリックアクセスを提供することはお勧めしません。
- セキュリティの焦点 – セキュリティ設定をデフォルトの仕様に抽象化することに重点を置いているため、 の " コネクタのデータ交換に関連するステップに集中できます。デフォルトのセキュリティ設定は維持されますが、クラスターをパブリックネットワークに公開する前に、安全な通信を有効にすることが不可欠です。この予防策により、安全なデータ処理のための堅牢な基盤が確保されます。
- インフラストラクチャコスト - インフラストラクチャのコストの見積もりは、 を使用して確認できますAWS 料金見積りツール。単純な計算では、デプロイされたインフラストラクチャのコストは 1 か月あたり最大 162.92 USD になる可能性があります。

アーキテクチャ

MVDS アーキテクチャは 2 つの仮想プライベートクラウド (VPCs) で構成されます。1 つは動的属性プロビジョニングシステム (DAPS) ID サービス用、もう 1 つは Amazon EKS 用です。

DAPS アーキテクチャ

次の図は、Auto Scaling グループによって制御される EC2 インスタンスで実行されている DAPS を示しています。Application Load Balancer とルートテーブルは DAPS サーバーを公開します。Amazon Elastic File System (Amazon EFS) は、DAPS インスタンス間でデータを同期します。

Amazon EKS アーキテクチャ

データスペースはテクノロジーに依存しないソリューションとして設計されており、複数の実装があります。このパターンでは、Amazon EKS クラスターを使用してデータスペースの技術コンポーネントをデプロイします。次の図は、EKS クラスターのデプロイを示しています。ワーカーノードはプライベートサブネットにインストールされます。Kubernetes ポッドは、プライベートサブネットにも存在する Amazon Relational Database Service (Amazon RDS) for PostgreSQL インスタンスにアクセスします。Kubernetes ポッドは、共有データを Amazon S3 に保存します。

ツール

AWS サービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Elastic File System (Amazon EFS) は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Elastic Load Balancing (ELB) は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。たとえば、1 つ以上のアベイラビリティーゾーンの EC2 インスタンス、コンテナ、IP アドレスにトラフィックを分散できます。

その他のツール

- eksctl – これは Amazon EKS で Kubernetes クラスターを作成および管理するコマンドラインユーティリティです。
- Git はオープンソースの分散バージョン管理システムです。
- HashiCorp Vault は、認証情報やその他の機密情報へのアクセスを制御できる安全なストレージを提供します。
- Helm は、Kubernetes クラスターへのアプリケーションのインストールと管理に役立つ Kubernetes のオープンソースパッケージマネージャーです。
- kubectlは、Kubernetes クラスターに対してコマンドを実行するためのコマンドラインインターフェイスです。
- Postman は API プラットフォームです。

コードリポジトリ

このパターンの Kubernetes 設定 YAML ファイルと Python スクリプトは、GitHub aws-patterns-edc リポジトリで利用できます。このパターンでは、Tractus-X の " リポジトリも使用します。

ベストプラクティス

Amazon EKS と参加者のインフラストラクチャの分離

Kubernetes の名前空間は、このパターンで X プロバイダーのインフラストラクチャと Y コンシューマーのインフラストラクチャを分離します。詳細については、「EKS ベストプラクティスガイド」を参照してください。

より現実的な状況では、各参加者には個別の Kubernetes クラスターが独自の 内で実行されています AWS アカウント。共有インフラストラクチャ (このパターンの DAPS) は、データスペースの参加者がアクセスでき、参加者のインフラストラクチャから完全に分離されます。

エピック

環境を設定し、EKS クラスターと EC2 インスタンスをプロビジョニングする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリをワークステーションにクローンするには、次のコマンドを実行します。 `git clone https://github.com/Think-iT-Labs/aws-patterns-edc` ワークステーションは にアクセスできる必要がありますAWS アカウント。	DevOps エンジニア
Kubernetes クラスターをプロビジョニングし、名前空間を設定します。	アカウントに簡略化されたデフォルトの EKS クラスターをデプロイするには、リポジトリをクローンしたワークステーションで次のeksctlコマンドを実行します。 `eksctl create cluster` コマンドは、VPC と、3 つの異なるアベイラビリティーゾーンにまたがるプライベートサブネットとパブリックサブネットを作成します。ネットワークレイヤーが作成されると、 コマンドは Auto Scaling グループ内に 2 つの`m5.large` EC2 インスタンスを作成します。	DevOps エンジニア

タスク	説明	必要なスキル

	詳細と出力例については、eksctl ガイドを参照してください。
	
	プライベートクラスターをプロビジョニングしたら、次のコマンドを実行して、新しい EKS クラスターをローカル Kubernetes 設定に追加します。
	
	`aws eks update-kubeconfig --name <EKS CLUSTER NAME> --region <AWS REGION>`
	
	このパターンでは、 `eu-west-1` AWS リージョン を使用してすべてのコマンドを実行します。ただし、任意の で同じコマンドを実行できます AWS リージョン。
	
	EKS ノードが実行されていて、準備完了状態であることを確認するには、次のコマンドを実行します。
	
	`kubectl get nodes`	

タスク	説明	必要なスキル

名前空間を設定します。	プロバイダーとコンシューマーの名前空間を作成するには、次のコマンドを実行します。
	
	`kubectl create ns provider`
	`kubectl create ns consumer`
	
	このパターンでは、次のステップの設定に合わせて、名前空間`consumer`として`provider`と を使用することが重要です。
		DevOps エンジニア

ID サービスをデプロイする

タスク	説明	必要なスキル
を使用して DAPS をデプロイします AWS CloudFormation。	DAPS オペレーションの管理を容易にするために、DAPS サーバーは EC2 インスタンスにインストールされます。 DAPS をインストールするには、 AWS CloudFormation テンプレートを使用します。前提条件セクションの ACM 証明書と DNS 名が必要です。テンプレートは以下をデプロイして設定します。 • Application Load Balancer • Auto Scaling グループ	DevOps エンジニア

タスク	説明	必要なスキル
	• 必要なすべてのパッケージをインストールするためのユーザーデータで設定された EC2 インスタンス • IAM ロール • DAPS テンプレートをデプロイするには、 に AWS CloudFormation サインイン AWS Management Console し、AWS CloudFormation コンソールを使用します。テンプレートは、次のような AWS CLI コマンドを使用してデプロイすることもできます。 `aws cloudformation create-stack --stack-name daps \ --template-body file://aws-patterns-edc/cloudformation.yml --parameters \ ParameterKey=CertificateARN,ParameterValue=<ACM Certificate ARN> \ ParameterKey=DNSName,ParameterValue=<DNS name> \ ParameterKey=InstanceType,ParameterValue=<EC2 instance type> \ ParameterKey=EnvironmentName,ParameterValue=<Environ`	

タスク	説明	必要なスキル
	ment Name> --capabil ities CAPABILIT Y_NAMED_IAM 環境名は独自の選択です。AWS リソースタグに反映 DapsInfrastructure されるため、 などの意味のある用語を使用することをお勧めします。 このパターンでは、t3.smallは 3 つの Docker コンテナを持つ DAPS ワークフローを実行するのに十分な大きさです。 テンプレートは EC2 インスタンスをプライベートサブネットにデプロイします。つまり、インスタンスはインターネットから SSH (Secure Shell) 経由で直接アクセスすることはできません。インスタンスは、 の一機能である Session Manager を介して実行中のインスタンスへのアクセスを有効にするために必要な IAM ロールと AWS Systems Manager エージェントでプロビジョニングされます AWS Systems Manager。 アクセスには Session Manager を使用することをお勧めします。または、踏み	

タスク	説明	必要なスキル

	台ホストをプロビジョニングして、インターネットからのSSH アクセスを許可することもできます。踏み台ホストアプローチを使用する場合、EC2 インスタンスの実行開始に数分かかることがあります。
	
	AWS CloudFormation テンプレートが正常にデプロイされたら、DNS 名を Application Load Balancer の DNS 名にポイントします。これを確認するには、次のコマンドを実行します。
	
	`dig <DNS NAME>`
	
	出力は次の例のようになります:
	
	`; <<>> DiG 9.16.1-Ubuntu <<>> edc-pattern.think-it.io`
	`;; global options: +cmd`
	`;; Got answer:`
	`;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42344`
	`;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1`
	
	`;; OPT PSEUDOSECTION:`
	`; EDNS: version: 0, flags:; udp: 65494`	

タスク	説明	必要なスキル
	;; QUESTION SECTION: ;edc-pattern.think- it.io. IN A ;; ANSWER SECTION: edc-pattern.think- it.io. 276 IN CNAME daps- alb-iap9zmwy3kn8-13287 73120.eu-west-1.el b.amazonaws.com. daps-alb-iap9zmwy3k n8-1328773120.eu-w est-1.elb.amazonaw s.com. 36 IN A 52.208.240.129 daps-alb-iap9zmwy3kn8 -1328773120.eu-wes t-1.elb.amazonaws. com. 36 IN A 52.210.15 5.124	

タスク	説明	必要なスキル
参加者のコネクタを DAPS サービスに登録します。	DAPS 用にプロビジョニングされた EC2 インスタンス内から、参加者を登録します。 1. ルートユーザーを使用して、EC2 インスタンスで使用可能なスクリプトを実行します。 ``` cd /srv/mvds/omejdn-daps ``` 2. プロバイダーを登録します。 ``` bash scripts/register_connector.sh <provider_name> ``` 3. コンシューマーを登録します。 ``` bash scripts/register_connector.sh <consumer_name> ``` 名前の選択は次のステップには影響しません。と providerconsumerまたは companyxと を使用することをお勧めしますcompanyy。 また、登録コマンドは、作成された証明書とキーから取得した必要な情報を使用して	DevOps エンジニア

タスク	説明	必要なスキル
	DAPS サービスを自動的に設定します。 DAPS サーバーにログインしている間、インストールの後のステップで必要な情報を収集します。 1. プロバイダーとコンシューマーclient idの omejdn-daps/config /clients.yml を取得します。client id 値は 16 進数の長い文字列です。 2. omejdn-daps/keys ディレクトリから、consumer. cert 、、consumer. key 、provider. cert および provider. key ファイルの内容をコピーします。 テキストをコピーして、ワークステーションdaps-の というプレフィックスが付いた同様の名前のファイルに貼り付けることをお勧めします。 プロバイダーとコンシューマーIDs があり、ワークステーションの作業ディレクトリに 4 つのファイルがある必要があります。	

タスク	説明	必要なスキル
	• ソースファイル名はワークステーションファイル名 consumer.cert になりますdaps-consumer.cert 。 • ソースファイル名はワークステーションファイル名 consumer.key になりますdaps-consumer.key 。 • ソースファイル名はワークステーションファイル名 provider.cert になりますdaps-provider.cert 。 • ソースファイル名はワークステーションファイル名 provider.key になりますdaps-provider.key 。	

参加者のコネクタをデプロイする

タスク	説明	必要なスキル
Tractus-X リポジトリのクローンを作成し、0.4.1 バージョンを使用します。	Tractus-X " コネクタのビルドでは、PostgreSQL (アセットデータベース) および HashiCorp Vault (シークレット管理) サービスをデプロイして使用できるようにする必要があります。	DevOps エンジニア

タスク	説明	必要なスキル

	Tractus-X の " Helm チャートにはさまざまなバージョンがあります。このパターンは、DAPS サーバーを使用するため、バージョン 0.4.1 を指定します。
	
	最新バージョンでは、ID サービスの分散実装で Managed Identity Wallet (MIW) を使用します。
	
	2 つの Kubernetes 名前空間を作成したワークステーションで、tractusx-edc リポジトリのクローンを作成し、`release/0.4.1` ブランチを確認します。
	
	`git clone https://github.com/eclipse-tractusx/tractusx-edc`
	
	`cd tractusx-edc`
	
	`git checkout release/0.4.1`	

タスク	説明	必要なスキル

Tractus-X " Helm チャートを設定します。	Tractus-X Helm チャートテンプレート設定を変更して、両方のコネクタが相互にやり取りできるようにします。
	
	これを行うには、名前空間をサービスの DNS 名に追加して、クラスター内の他のサービスで解決できるようにします。これらの変更は、 `charts/tractusx-connector/templates/_helpers.tpl` ファイルに対して行う必要があります。このパターンでは、このファイルの最終変更バージョンを使用できます。コピーして、ファイル の `daps`セクションに配置します`charts/tractusx-connector/templates/_helpers.tpl` 。
	
	ですべての DAPS 依存関係をコメントしてください`charts/tractusx-connector/Chart.yaml` 。
	
	`dependencies:`
	`  # IDS Dynamic Attribute Provisioning Service (IAM)`
	`  #  - name: daps`
	`  # version: 0.0.1`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre> # repository:
	 "file://./subcharts/
	omejdn"
	  # alias: daps
	  # condition:
	 install.daps</pre>	

タスク	説明	必要なスキル

Amazon RDS で PostgreSQL を使用するようにコネクタを設定します。	(オプション) このパターンではAmazon Relational Database Service (Amazon RDS) インスタンスは必要ありません。ただし、Amazon RDS または Amazon Aurora は高可用性、バックアップ、リカバリなどの機能を提供するため、使用を強くお勧めします。
	
	PostgreSQL on Kubernetes を Amazon RDS に置き換えるには、次の手順を実行します。
	
	1. Amazon RDS for PostgreSQL インスタンスをプロビジョニングします。
	2. でChart.yaml 、PostgreSQL セクションをコメントします。
	3. provider_values.yml および でconsumer_values.yml 、postgresql セクションを次のように設定します。
	
	`postgresql:`
	`  auth:`
	`  database: edc`
	`  password: <RDS PASSWORD>`
	`  username: <RDS Username>`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre> jdbcUrl: jdbc:post
	gresql://<RDS DNS
	 NAME>:5432/edc
	  username: <RDS
	 Username>
	  password: <RDS
	 PASSWORD>
	  primary:
	  persistence:
	  enabled: false
	  readReplicas:
	  persistence:
	  enabled: false</pre>	

タスク	説明	必要なスキル
プロバイダーコネクタとそのサービスを設定してデプロイします。	プロバイダーコネクタとそのサービスを設定するには、次の手順を実行します。 1. edc_helm_configs ディレクトリから現在の Helm チャートフォルダにprovider_edc.yaml ファイルをダウンロードするには、次のコマンドを実行します。 `wget -q https://raw.githubusercontent.com/Think-iT-Labs/aws-patterns-edc/main/edc_helm_configs/provider_edc.yaml> -P charts/tractusx-connector/` 2. 次の変数 (ファイルにもマークされています) を値に置き換えます。 • CLIENT_ID – DAPS によって生成された ID。は DAPS サーバー/srv/mvds/omejdn-daps/config/clients.yml/config/clients.yml 上の にあるCLIENT_ID 必要があります。16 進数の文字列である必要があります。	DevOps エンジニア

タスク	説明	必要なスキル

	• DAPS_URL – DAPS サーバーの URL。 AWS CloudFormation テンプレートの実行時に設定した DNS 名`https://{DNS name}` を使用している必要があります。
	• `VAULT_TOKEN` – ボールト認可に使用されるトークン。任意の値を選択します。
	• `vault.fullnameOverride` – `vault-provider` 。
	• `vault.hashicorp.url` – `http://vault-provider:8200/` 。
	前の値は、デプロイ名と名前空間名がプロバイダーであることを前提としています。
	3. ワークステーションから Helm チャートを実行するには、次のコマンドを使用します。
	`cd charts/tractusx-connector`
	`helm dependency build`
	`helm upgrade --install provider ./ -f`	

タスク	説明	必要なスキル
	`provider_edc.yaml -n provider`	

タスク	説明	必要なスキル

証明書とキーをプロバイダーボールトに追加します。	混乱を避けるために、tractusx-edc/charts ディレクトリの外部で次の証明書を作成します。
	
	たとえば、次のコマンドを実行してホームディレクトリに変更します。
	
	`cd ~`
	
	これで、プロバイダーが必要とするシークレットをボールトに追加する必要があります。
	
	ボールト内のシークレットの名前は、 provider_edc.yml ファイルの secretNames: セクションにあるキーの値です。デフォルトでは、次のように設定されます。
	
	`secretNames:`
	`  transferProxyTokenSignerPrivateKey: transfer-proxy-token-signer-private-key`
	`  transferProxyTokenSignerPublicKey: transfer-proxy-token-signer-public-key`
	`  transferProxyTokenEncryptio`
		DevOps エンジニア

タスク	説明	必要なスキル
	nAesKey: transfer-proxy-token-encryption-aes-key dapsPrivateKey: daps-private-key dapsPublicKey: daps-public-key Advanced Encryption Standard (AES) キー、プライベートキー、パブリックキー、および自己署名証明書が最初に生成されます。その後、これらはシークレットとしてボールトに追加されます。 さらに、このディレクトリには、DAPS サーバーからコピーした daps-provider.cert および daps-provider.key ファイルが含まれている必要があります。 1. 以下の コマンドを実行します。 ``` # generate a private key openssl ecparam -name prime256v1 -genkey -noout -out provider-private-key.pem # generate corresponding public key	

タスク	説明	必要なスキル
	``` openssl ec -in provider-private-k ey.pem -pubout -out provider-public-ke y.pem # create a self-sign ed certificate openssl req -new - x509 -key provider- private-key.pem -out provider-cert.pem - days 360 # generate aes key openssl rand -base64 32 > provider- aes.key ``` 2. ボールトにシークレットを追加する前に、改行を に置き換えて、シークレットを複数行から単一行に変換します\n。 ``` cat provider-private- key.pem | sed 's/$/\ \\\n/'\|tr -d '\\n' > provider-private-k ey.pem.line cat provider-public- key.pem \| sed 's/$/\ \\\n/'|tr -d '\\n' > provider-public-ke y.pem.line cat provider-cert.pem | sed 's/$/\\\\ n/'\|tr -d '\\n' > provider-cert.pem. line cat provider-aes.key \| sed 's/$/\\\\ n/'|tr -d '\\n' > ```	

タスク	説明	必要なスキル
	provider-aes.key.l ine ## The following block is for daps certifica te and key openssl x509 -in daps-provider.cert - outform PEM | sed 's/ $/\\\\n/'\|tr -d '\\n' > daps-provider.cert .line cat daps-provider.key \| sed 's/$/\\\\n/'| tr -d '\\n' > daps- provider.key.line 3. Vault に追加するシークレットをフォーマットするには、次のコマンドを実行します。 JSONFORMAT='{"cont ent": "%s"}' #create a single line in JSON format printf "${JSONFO RMAT}\\n" "`cat  provider-private- key.pem.line`" > provider-private-k ey.json printf "${JSONFO RMAT}\\n" "`cat  provider-public- key.pem.line`" > provider-public-ke y.json printf "${JSONFO RMAT}\\n" "`cat provider-cert.pem.	

タスク	説明	必要なスキル
	line`" > provider-cert.json printf "${JSONFORMAT}\\n" "`cat provider-aes.key.line`" > provider-aes.json  printf "${JSONFORMAT}\\n" "`cat daps-provider.key.line`" > daps-provider.key.json printf "${JSONFORMAT}\\n" "`cat daps-provider.cert.line`" > daps-provider.cert.json これでシークレットは JSON 形式で、ボールトに追加される準備が整いました。 4. ボールトのポッド名を取得するには、次のコマンドを実行します。 kubectl get pods -n provider|egrep "vault|NAME" ポッド名は に似ています"vault-provider-0" 。この名前は、ボールトに転送されるポートを作成するときに使用されます。ポートフォワードを使用すると、ボールトにアクセス	

タスク	説明	必要なスキル
	してシークレットを追加できます。これは、AWS 認証情報が設定されたワークステーションから実行する必要があります。 5. ボールトにアクセスするには、 kubectl を使用してポートフォワードを設定します。 `kubectl port-forward <VAULT_POD_NAME> 8200:8200 -n provider` これで、ブラウザまたは CLI からボールトにアクセスできるはずです。 ブラウザ 1. ブラウザを使用して、設定したポートフォワードを使用する http://127.0.0.1:8200://https://https://https://https://https://https://https://https://https://https://https://https://https://https://https 2. で以前に設定したトークンを使用してログインしますprovider_edc.yml 。シークレットエンジンで、3 つのシークレットを作成します。各シークレットには、次のリストに示	

タスク	説明	必要なスキル
	すシークレット名である Path for this secret 値があります。secret data セクション内では、キーの名前は contentになり、値は という名前のそれぞれのファイルの 1 行のテキストになります.line。 3. シークレット名は、provider_edc.yml ファイルの secretNames セクションから取得されます。 4. 次のシークレットを作成します。 • ファイル名transfer-proxy-token-signer-private-key のシークレット provider-private-key.pem.line • ファイル名transfer-proxy-token-signer-public-key のシークレット provider-cert.pem.line • ファイル名transfer-proxy-token-encryption-aes-key のシークレット provider-aes.key.line	

タスク	説明	必要なスキル
	• ファイル名daps-private-key のシークレット daps-provider.key.line • ファイル名daps-public-key のシークレット daps-provider.cert.line ボールト CLI CLI は、設定したポートフォワードも使用します。 1. ワークステーションで、HashiCorp Vault ドキュメントの指示に従って Vault CLI をインストールします。 2. で設定したトークンを使用してボールトにログインするにはprovider_edc.yml 、次のコマンドを実行します。 `vault login -address=http://127.0.0.1:8200` 正しいトークンで、メッセージが表示されます。"Success! You are now authenticated." 3. 以前に作成した JSON 形式のファイルを使用してシー	

タスク	説明	必要なスキル

	クレットを作成するには、次のコードを実行します。
	
	`vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-signer-private-key @provider-private-key.json`
	`vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-signer-public-key @provider-cert.json`
	`vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-encryption-aes-key @provider-aes.json`
	
	`vault kv put -address=http://127.0.0.1:8200 secret/daps-private-key @daps-provider.key.json`
	`vault kv put -address=http://127.0.0.1:8200 secret/daps-public-key @daps-provider.cert.json`	

タスク	説明	必要なスキル

コンシューマーコネクタとそのサービスを設定してデプロイします。	コンシューマーを設定およびデプロイする手順は、プロバイダーに対して完了した手順と似ています。
	
	1. aws-patterns-edc リポジトリconsumer_edc.yaml から tractusx-edc/charts/tractusx-connector フォルダに をコピーするには、次のコマンドを実行します。
	
	`cd tractusx-edc`
	
	`wget -q https://raw.githubusercontent.com/Think-iT-Labs/aws-patterns-edc/main/edc_helm_configs/consumer_edc.yaml -P charts/tractusx-connector/`
	
	2. 次の変数を実際の値で更新します。
	
	• CONSUMER_CLIENT_ID – DAPS によって生成された ID。は DAPS サーバーconfig/clients.yml 上のにあるCONSUMER_CLIENT_ID 必要があります。	

タスク	説明	必要なスキル

	• DAPS_URL – プロバイダーに使用したのと同じ DAPS URL。
	• VAULT_TOKEN – ボールト認可に使用されるトークン。任意の値を選択します。
	• vault.fullnameOverride – vault-consumer
	• vault.hashicorp.url – http://vault-provider:8200/
	前の値は、デプロイ名と名前空間名が consumer であることを前提としています。
	3. Helm チャートを実行するには、次のコマンドを使用します。
	`cd charts/tractusx-connector`
	`helm upgrade --install consumer ./ -f consumer_edc.yaml -n consumer`	

タスク	説明	必要なスキル

証明書とキーをコンシューマーボールトに追加します。	セキュリティの観点から、各データスペース参加者の証明書とキーを再生成することをお勧めします。このパターンは、コンシューマーの証明書とキーを再生成します。
	
	ステップは、プロバイダーのステップとよく似ています。ファイル内のシークレット名を確認できますconsumer_edc.yml 。
	
	ボールト内のシークレットの名前は、 の secretNames: セクションのキーの値ですconsumer_edc.yml file。デフォルトでは、次のように設定されます。
	
	`secretNames:`
	`  transferProxyTokenSignerPrivateKey: transfer-proxy-token-signer-private-key`
	`  transferProxyTokenSignerPublicKey: transfer-proxy-token-signer-public-key`
	`  transferProxyTokenEncryptionAesKey: transfer-proxy-token-encryption-aes-key`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre>dapsPrivateKey: daps-private-key
	dapsPublicKey: daps-public-key</pre>
	DAPS サーバーからコピーした daps-consumer.cert および daps-consumer.key ファイルは、このディレクトリに既に存在している必要があります。
	1. 以下の コマンドを実行します。
	<pre># generate a private key
	openssl ecparam -name prime256v1 -genkey -noout -out consumer-private-key.pem
	# generate corresponding public key
	openssl ec -in consumer-private-key.pem -pubout -out consumer-public-key.pem
	# create a self-signed certificate
	openssl req -new -x509 -key consumer-private-key.pem -out consumer-cert.pem -days 360
	# generate aes key</pre>	

タスク	説明	必要なスキル

	`openssl rand -base64 32 > consumer-aes.key`
	2. ファイルを手動で編集して改行を に置き換えるか\n、次のような 3 つのコマンドを使用します。
	`cat consumer-private-key.pem | sed 's/$/\\\n/'|tr -d '\n' > consumer-private-key.pem.line`
	`cat consumer-public-key.pem | sed 's/$/\\\n/'|tr -d '\n' > consumer-public-key.pem.line`
	`cat consumer-cert.pem | sed 's/$/\\\\n/'|tr -d '\n' > consumer-cert.pem.line`
	`cat consumer-aes.key | sed 's/$/\\\\n/'|tr -d '\n' > consumer-aes.key.line`
	`cat daps-consumer.cert | sed 's/$/\\\\n/'|tr -d '\n' > daps-consumer.cert.line`
	`cat daps-consumer.key | sed 's/$/\\\\n/'| tr -d '\n' > daps-consumer.key.line`
	3. Vault に追加するシークレットをフォーマットするに	

タスク	説明	必要なスキル

	は、次のコマンドを実行します。
	
	`JSONFORMAT='{"content": "%s"}'`
	
	`#create a single line in JSON format`
	
	`printf "${JSONFORMAT}\\n" "`cat consumer-private-key.pem.line`" > consumer-private-key.json`
	`printf "${JSONFORMAT}\\n" "`cat consumer-public-key.pem.line`" > consumer-public-key.json`
	`printf "${JSONFORMAT}\\n" "`cat consumer-cert.pem.line`" > consumer-cert.json`
	`printf "${JSONFORMAT}\\n" "`cat consumer-aes.key.line`" > consumer-aes.json`
	
	`printf "${JSONFORMAT}\\n" "`cat daps-consumer.key.line`" > daps-consumer.key.json`
	`printf "${JSONFORMAT}\\n" "`cat daps-consumer.cert.line`"	

タスク	説明	必要なスキル
	> daps-consumer.cert .json これでシークレットは JSON 形式で、ボールトに追加される準備が整いました。 4. コンシューマーボールトのポッド名を取得するには、次のコマンドを実行します。 kubectl get pods - n consumer | egrep "vault|NAME" ポッド名は に似ています"vault-consumer-0"。この名前は、ボールトに転送されるポートを作成するときに使用されます。ポートフォワードを使用すると、ボールトにアクセスしてシークレットを追加できます。これは、 AWS 認証情報が設定されたワークステーションから実行する必要があります。 5. ボールトにアクセスするには、 kubectl を使用してポートフォワードを設定します。	

タスク	説明	必要なスキル
	`kubectl port-forward <VAULT_POD_NAME> 8201:8200 -n consumer` プロデューサーとコンシューマーの両方にポート転送を設定できるように、ローカルポートは今回は 8201 です。 ブラウザ ブラウザを使用して http://localhost:8201/ に接続し、コンシューマーボールトにアクセスし、説明に従って名前とコンテンツを含むシークレットを作成できます。 コンテンツを含むシークレットとファイルは次のとおりです。 • ファイル名`transfer-proxy-token-signer-private-key` のシークレット `consumer-private-key.pem.line` • ファイル名`transfer-proxy-token-signer-public-key` のシークレット `consumer-cert.pem.line`	

タスク	説明	必要なスキル

	• ファイル名transfer-proxy-token-encryption-aes-key のシークレット consumer-aes.key.line
	
	ボールト CLI
	
	ボールト CLI を使用すると、次のコマンドを実行してボールトにログインし、シークレットを作成できます。
	
	1. 内で設定したトークンを使用してボールトにログインしますconsumer_edc.yml 。
	
	`vault login -address=http://127.0.0.1:8201`
	
	正しいトークンで、メッセージが表示されます。"Success! You are now authenticated."
	2. 以前に作成した JSON 形式のファイルを使用してシークレットを作成するには、次のコードを実行します。
	
	`vault kv put -address=http://127.0.0.1:8201 secret/transfer-proxy-token-signer-p`	

タスク	説明	必要なスキル
	rivate-key @consumer -private-key.json vault kv put - address=http://12 7.0.0.1:8201 secret/ transfer-proxy-token -signer-public-key @consumer-cert.json vault kv put -address= http://127.0.0.1:8 201 secret/transfer- proxy-token-encrypti on-aes-key @consumer -aes.json vault kv put -address= http://127.0.0.1:8 201 secret/daps- private-key @daps-con sumer.key.json vault kv put - address=http://12 7.0.0.1:8201 secret/ daps-public-key @daps-consumer.cer t.json	

コネクタの管理 API とやり取りするように HTTP クライアントを設定する

タスク	説明	必要なスキル
ポート転送を設定します。	1. ポッドのステータスを確認するには、次のコマンドを実行します。 kubectl get pods -n provider kubectl get pods -n consumer	DevOps エンジニア

タスク	説明	必要なスキル

	2. Kubernetes デプロイが成功したことを確認するには、次のコマンドを実行して、プロバイダーとコンシューマーの Kubernetes ポッドのログを確認します。
	
	`kubectl logs -n provider <producer control plane pod name>`
	`kubectl logs -n consumer <consumer control plane pod name>`
	
	クラスターはプライベートであり、パブリックにアクセスすることはできません。コネクタを操作するには、Kubernetes ポート転送機能を使用して、マシンによって生成されたトラフィックをコネクタコントロールプレーンに転送します。
	
	1. 最初のターミナルで、コンシューマーのリクエストをポート 8300 経由で管理 API に転送します。
	
	`kubectl port-forward deployment/consumer-tractusx-connector-controlplane 8300:8081 -n consumer`	

タスク	説明	必要なスキル
	2. 2 番目のターミナルで、プロバイダーのリクエストをポート 8400 経由で管理 API に転送します。 `kubectl port-forward deployment/provider-tractusx-connector-controlplane 8400:8081 -n provider`	

タスク	説明	必要なスキル
プロバイダーとコンシューマーの S3 バケットを作成します。	現在、 の " コネクタは、ロールを引き受けることによって提供されるような一時的な AWS 認証情報を使用しません。「」 では、IAM アクセスキー ID とシークレットアクセスキーの組み合わせの使用のみがサポートされています。 後のステップでは 2 つの S3 バケットが必要です。1 つの S3 バケットは、プロバイダーが利用できるデータを保存するために使用します。もう 1 つの S3 バケットは、コンシューマーが受信したデータ用です。 IAM ユーザーには、2 つの名前付きバケット内のオブジェクトのみを読み書きするアクセス許可が必要です。 アクセスキー ID とシークレットアクセスキーペアを作成し、安全に保つ必要があります。この MVDS が廃止されたら、IAM ユーザーを削除する必要があります。 次のコードは、 ユーザーの IAM ポリシーの例です。 `{ "Version": "2012-10-17",`	DevOps エンジニア

タスク	説明	必要なスキル
	"Statement": [{ "Sid": "Stmt1708699805237", "Action": ["s3:GetObject", "s3:GetObjectVersion", "s3:ListAllMyBuckets", "s3:ListBucket", "s3:ListBucketMultipartUploads", "s3:ListBucketVersions", "s3:PutObject"], "Effect": "Allow", "Resource": ["arn:aws:s3:::<S3 Provider Bucket>", "arn:aws:s3:::<S3 Consumer Bucket>", "arn:aws:s3:::<S3 Provider Bucket>/*", "arn:aws:s3:::<S3 Consumer Bucket>/*"] }] }	

タスク	説明	必要なスキル
コネクタを操作するように Postman を設定します。	EC2 インスタンスを介してコネクタを操作できるようになりました。Postman を HTTP クライアントとして使用し、プロバイダーとコンシューマーコネクタの両方に Postman コレクションを提供します。 `aws-pattern-edc` リポジトリから Postman インスタンスにコレクションをインポートします。 このパターンでは、Postman コレクション変数を使用してリクエストに入力を提供します。	アプリ開発者、データエンジニア

コネクタを介して X 社カーボンエミッションフットプリントデータを提供する

タスク	説明	必要なスキル
共有する炭素放出強度データを準備します。	まず、共有するデータアセットを決定する必要があります。X 社のデータは、その車両フリートのカーボンエミッションフットプリントを表します。重量はトン単位の車両総重量 (GVW) で、排出量は Wheel-to-Well (WTW) 測定値に基づく 1 トン/キロメートル (g CO2 e/t-km) あたりの CO2 のグラム単位です。 CO2	データエンジニア、アプリ開発者

タスク	説明	必要なスキル
	• 車両タイプ: Van、重み: < 3.5、排出量: 800 • 車両タイプ: 都市トラック、重量: 3.5 ~ 7.5、排出量: 315 • 車両タイプ: 中型車両 (MGV)、重量: 7.5 ~ 20、排出量: 195 • 車両タイプ: 重量車両 (HGV)、重量: > 20、排出量: 115 サンプルデータは、 `aws-patterns-edc` リポジトリの `carbon_emissions_data.json` ファイルにあります。 X 社は Amazon S3 を使用してオブジェクトを保存します。 S3 バケットを作成し、サンプルデータオブジェクトをそこに保存します。次のコマンドは、デフォルトのセキュリティ設定で S3 バケットを作成します。Amazon S3 のセキュリティのベストプラクティスを参照することを強くお勧めします。 `aws s3api create-bucket <BUCKET_NAME> --region <AWS_REGION>`	

タスク	説明	必要なスキル
	``` # You need to add '--create-bucket-c onfiguration # LocationConstraint =<AWS_REGION>' if you want to create # the bucket outside of us- east-1 region aws s3api put-object --bucket <BUCKET_NAME> \ --key <S3 OBJECT NAME> \ --body <PATH OF THE FILE TO UPLOAD> ``` S3 バケット名はグローバルに一意である必要があります。命名規則の詳細については、AWS ドキュメントを参照してください。	

タスク	説明	必要なスキル
Postman を使用してデータアセットをプロバイダーのコネクタに登録します。	" コネクタのデータアセットは、データの名前とその場所を保持します。この場合、の " コネクタデータアセットは、S3 バケットに作成されたオブジェクトを指します。 • コネクタ： プロバイダー • リクエスト： アセットの作成 • コレクション変数： を更新しますASSET_NAME 。アセットを表すわかりやすい名前を選択します。 • リクエスト本文： プロバイダー用に作成した S3 バケットでリクエスト本文を更新します。 `"dataAddress": { "edc:type": "AmazonS3", "name": "Vehicle Carbon Footprint", "bucketName": "<REPLACE WITH THE SOURCE BUCKET NAME>", "keyName": "<REPLACE WITH YOUR OBJECT NAME>", "region": "<REPLACE WITH THE BUCKET REGION>", "accessKeyId": "<REPLACE WITH YOUR ACCESS KEY ID>",`	アプリ開発者、データエンジニア

タスク	説明	必要なスキル

	<pre>"secretAccessKey": "<REPLACE WITH SECRET ACCESS KEY>"
	}</pre>
	• レスポンス： リクエストが成功すると、作成された時刻と新しく作成されたアセットのアセット ID が返されます。
	<pre>{
	"@id": "c89aa31c-ec4c-44ed-9e8c-1647f19d7583"
	}</pre>
	• コレクション変数 ASSET_ID: Postman コレクション変数を、作成後に " コネクタによって自動的に生成された ID ASSET_IDで更新します。	

タスク	説明	必要なスキル

アセットの使用ポリシーを定義します。	" データアセットは、明確な使用ポリシーに関連付ける必要があります。まず、プロバイダーコネクタでポリシー定義を作成します。
	
	X 社のポリシーは、データスペースの参加者が炭素排出量フットプリントデータを使用できるようにすることです。
	
	• リクエスト本文：
	  • コネクタ： プロバイダー
	  • リクエスト： ポリシーの作成
	  • コレクション変数: ポリシーの名前でPolicy Name変数を更新します。
	• レスポンス： リクエストが成功すると、作成された時刻と新しく作成されたポリシーのポリシー ID が返されます。作成後に " コネクタによって生成されたポリシーの ID POLICY_ID でコレクション変数を更新します。
		アプリ開発者、データエンジニア

タスク	説明	必要なスキル
アセットとその使用ポリシーの " 契約オファーを定義します。	他の参加者がデータへのアクセスをリクエストできるようにするには、使用状況とアクセス許可を指定する契約でデータを提供します。 • コネクタ： プロバイダー • リクエスト： 契約定義の作成 • コレクション変数: `Contract Name`変数を契約オファーまたは定義の名前で更新します。	アプリ開発者、データエンジニア

アセットを検出し、定義された契約について合意に達する

タスク	説明	必要なスキル
X 社が共有しているデータカタログをリクエストします。	データスペースのデータコンシューマーとして、Y 社はまず他の参加者によって共有されているデータを検出する必要があります。 この基本的なセットアップでは、コンシューマーコネクタにプロバイダーコネクタから使用可能なアセットのカタログを直接リクエストするように依頼することで、これを行うことができます。 • コネクタ： コンシューマー	アプリ開発者、データエンジニア

タスク	説明	必要なスキル

	• リクエスト： リクエストカタログ
	• レスポンス： プロバイダーから利用可能なすべてのデータアセットと、アタッチされた使用ポリシー。データコンシューマーは、関心のある契約を探し、それに応じて以下のコレクション変数を更新します。
	  • CONTRACT_OFFER_ID – コンシューマーが交渉したい契約オファーの ID
	  • ASSET_ID – コンシューマーが交渉するアセットの ID
	  • PROVIDER_CLIENT_ID – ネゴシエートするプロバイダーコネクタの ID	

タスク	説明	必要なスキル
X 社からの炭素排出量強度データの契約交渉を開始します。	使用するアセットを特定したら、コンシューマーコネクタとプロバイダーコネクタの間で契約交渉プロセスを開始します。 • コネクタ： コンシューマー • リクエスト： 契約交渉 • コレクション変数： ネゴシエートするコンシューマーコネクタの ID でCONSUMER_CLIENT_ID 変数を更新します。 プロセスが VERIFIED 状態になるまでに時間がかかる場合があります。 Get Negotiation リクエストを使用して、契約交渉の状態と対応する契約 ID を確認できます。	アプリ開発者、データエンジニア

契約を使用してデータを使用する

タスク	説明	必要なスキル
HTTP エンドポイントからデータを使用します。	(オプション 1) HTTP データプレーンを使用してデータスペース内のデータを消費するには、webhook.site を使用して HTTP サーバーをエミュレートし、コンシューマーコネ	アプリ開発者、データエンジニア

タスク	説明	必要なスキル
	クタで転送プロセスを開始できます。 • コネクタ： コンシューマー • リクエスト： 契約交渉 • コレクション変数: `Contract Agreement ID`変数を " コネクタによって生成された契約契約の ID で更新します。 • リクエスト本文： リクエスト本文を更新して、ウェブフック URL `dataDestination` とともに HTTPとして指定します。 `{ "dataDestination": { "type": "HttpProxy" }, "privateProperties": { "receiverHttpEndpoint": "<WEBHOOK URL>" } }` コネクタは、ファイルのダウンロードに必要な情報をウェブフック URL に直接送信します。 受信したペイロードは次のようになります。	

タスク	説明	必要なスキル

	{
	"id": "dcc90391
	-3819-4b54-b401-1a
	005a029b78",
	"endpoint":
	"http://consumer-t
	ractusx-connector-
	dataplane.consumer
	:8081/api/public",
	"authKey":
	"Authorization",
	"authCode": "<AUTH
	CODE YOU RECEIVE IN
	THE ENDPOINT>",
	"properties": {
	"https://
	w3id.org/edc/v0.0.
	1/ns/cid": "vehicle-
	carbon-footprint-c
	ontract:4563abf7-5
	dc7-4c28-bc3d-97f4
	5e32edac:b073669b-
	db20-4c83-82df-46b
	583c4c062"
	}
	}
	受信した認証情報を使用して、プロバイダーによって共有された S3 アセットを取得します。
	
	この最後のステップでは、ペイロード () に記載されているように、コンシューマーデータプレーンにリクエストを送信する必要があります	

タスク	説明	必要なスキル
	(フォワードポートを適切に送信) endpoint。	

タスク	説明	必要なスキル
S3 バケットからのデータを直接消費します。	(オプション 2) Amazon S3 と " コネクタの統合を使用し、コンシューマーインフラストラクチャの S3 バケットを宛先として直接指します。 • リクエスト本文： リクエスト本文を更新して、dataDestination として S3 バケットを指定します。 これは、コンシューマーが受信したデータを保存するために以前に作成した S3 バケットである必要があります。 `{ "dataDestination": { "type": "AmazonS3", "bucketName": "{{ REPLACE WITH THE DESTINATION BUCKET NAME }}", "keyName": "{{ REPLACE WITH YOUR OBJECT NAME }}", "region": "{{ REPLACE WITH THE BUCKET REGION }}", "accessKeyId": "{{ REPLACE WITH YOUR ACCESS KEY ID }}", "secretAccessKey": "{{ REPLACE WITH SECRET ACCESS KEY }}"`	データエンジニア、アプリ開発者

タスク	説明	必要なスキル

	<pre>}
	}
	}</pre>	

トラブルシューティング

問題	ソリューション
コネクタは、証明書の PEM 形式に関する問題を引き起こす可能性があります。	を追加して、各ファイルの内容を 1 行に連結します\n。

関連リソース

- DSSC
- 持続可能性のユースケースのためのデータスペースの構築 (Think-it による AWS 規範ガイダンス戦略)
- データスペースの AWS
- Tractus-X ドキュメント
- DAPS
- データスペースと AWS によるデータ共有の有効化 (ブログ記事)

追加情報

データスペースの仕様

参加者

Participant	会社の説明	会社の焦点
X 社	欧州および南米で車両フリートを運用し、さまざまな商品を輸送します。	炭素排出量のフットプリントの強度を減らすために、デー

		タ駆動型の意思決定を行うことを目指します。
Y 社	環境規制当局	炭素排出量の強度など、企業や業界の環境への影響を監視および軽減するために設計された環境規制とポリシーを適用します。

ビジネスケース

X 社は、データスペーステクノロジーを使用してカーボンフットプリントデータをコンプライアンス監査者である Y 社と共有し、X 社の物流業務による環境への影響を評価して対処します。

データスペースの権限

データスペース機関は、データスペースを管理する組織のコンソーシアムです。このパターンでは、X 社と Y 社がガバナンスボディを形成し、フェデレーティッドデータスペースの権限を表します。

データスペースコンポーネント

コンポーネント	選択した実装	追加情報
データセット交換プロトコル	Dataspace Protocol バージョン 0.8	• JSON-LD • データカタログ語彙 (DCAT)
データスペースコネクタ	Tractus-X " コネクタバージョン 0.4.1	• " 拡張機能
データ交換ポリシー	デフォルトの USE ポリシー	• Open Digital Rights Language (ODRL)

データスペースサービス

サービス	実装	追加情報

ID サービス	動的属性プロビジョニングシステム (DAPS)	「動的属性プロビジョニングシステム (DAPS) には、組織やコネクタに対する特定の属性を確認する意図があります。したがって、DAPS アサーションを信頼しているサードパーティーは後者を信頼する必要はありません。— DAPS コネクタのロジックに焦点を当てるために、データスペースは Docker Compose を使用して Amazon EC2 マシンにデプロイされます。
検出サービス	Gaia-X フェデレーティッドカタログ	「フェデレーティッドカタログは、Gaia-X Self-Descriptions のインデックス付きリポジトリを構成し、プロバイダーとそのサービス提供の検出と選択を可能にします。自己説明は、参加者が自分自身とサービスについてプロパティとクレームの形式で提供する情報です。」 — Gaia-X エコシステム Kickstarter

交換するデータ

データアセット	説明	Format
炭素排出量データ	車両のフリート全体からの、指定されたリージョン (欧州および南米) 内のさまざまな車両タイプの強度値	JSON ファイル

データモデル

```
{
  "region": "string",
  "vehicles": [
// Each vehicle type has its Gross Vehicle Weight (GVW) category and its emission
 intensity in grams of CO2 per Tonne-Kilometer (g CO2 e/t-km) according to the "Well-
to-Wheel" (WTW) measurement.
    {
      "type": "string",
      "gross_vehicle_weight": "string",
      "emission_intensity": {
        "CO2": "number",
        "unit": "string"
      }
    }
  ]
}
```

Tractus-X " コネクタ

各 Tractus-X " パラメータのドキュメントについては、元の値ファイルを参照してください。

次の表に、すべての サービス、および対応する公開ポートと参照用エンドポイントを示します。

[サービス名]	ポートとパス
コントロールプレーン	「管理: – ポート: 8081 パス: `/management`
	「」コントロール – ポート: 8083 パス: `/control`
	「 protocolPort: 8084 パス: `/api/v1/dsp`
	「」メトリクス – ポート: 9090 パス: `/metrics`
	「オブザーバビリティ」 – ポート: 8085 パス: `/observability`
データプレーン	デフォルト – ポート: 8080 パス: `/api`

	public – ポート: 8081 パス: `/api/data plane/control` proxy – ポート: 8186 パス: `/proxy` メトリクス – ポート: 9090 パス: `/metrics` オブザーバビリティ – ポート: 8085 パス: `/observability`
ボールト	ポート: 8200
PostgreSQL	ポート: 5432

AWS Secrets Manager Manager の使用

HashiCorp Vault の代わりに Secrets Manager をシークレットマネージャーとして使用できます。これを行うには、 が " AWS Secrets Manager 拡張機能を使用または構築する必要があります。

Tractus-X は Secrets Manager をサポートしていないため、独自のイメージの作成と保守はお客様の責任となります。

これを行うには、コントロールプレーンとコネクタのデータプレーンの両方のビルド Gradle ファイルを変更する必要があります。そのためには、 AWS Secrets Manager 「 という拡張機能」を導入し (例についてはこの Maven アーティファクトを参照) 、Docker イメージを構築、保守、参照します。

Tractus-X コネクタの Docker イメージのリファクタリングの詳細については、「Refactor Tractus-X " Helm charts」を参照してください。

分かりやすいように、このパターンでコネクタイメージを再構築することは避け、HashiCorp Vault を使用します。

スカラー Python UDF を使用した Amazon Redshift クエリー結果の言語固有のソートの設定

作成者： Ethan Stark (AWS)

概要

このパターンは、スカラーの Python UDF (ユーザー定義関数) を使用した Amazon Redshift のクエリ結果について、大文字と小文字を区別しない言語ソートを設定する手順とサンプルコードを提供します。Amazon Redshift はバイナリ UTF-8 ソートに基づいて結果を返し、言語固有のソートをサポートしていないので、スカラー Python UDF を使用する必要があります。Python UDF は Python 2.7 プログラム基づいた SQL 以外の処理コードで、データウェアハウス内で実行されます。1 つのクエリで SQL ステートメントを使用して Python UDF コードを実行できます。詳細については、AWS ビッグデータブログ記事の「Amazon Redshift における Python UDF の紹介」を参照してください。

このパターンのサンプルデータは、デモ用にトルコ語のアルファベットに基づいています。このパターンのスカラー Python UDF は、Amazon Redshift のデフォルトのクエリ結果をトルコ語の文字の言語順序に合うように構築されています。詳細については、このパターンの「追加情報」セクションにあるトルコ語の例を参照してください。このパターンのスカラー Python UDF を他の言語に変更できます。

前提条件と制限

前提条件

- データベース、スキーマ、テーブルを含む Amazon Redshift クラスター
- テーブル作成権限と関数作成権限を持つ Amazon Redshift ユーザー
- Python 2.7 以降

機能制限

このパターンのクエリで使用される言語ソートでは、大文字と小文字は区別しません。

アーキテクチャ

テクノロジースタック

- Amazon Redshift
- Python UDF

ツール

AWS サービス

- Amazon Redshift は、AWS クラウド内でのマネージド型、ペタバイト規模のデータウェアハウスサービスです。Amazon Redshift はデータレイクと統合されているため、データを使用して、ビジネスと顧客に関する新しいインサイトを得られます。

その他のツール

- Python (UDF) ユーザー定義関数 は、Python で記述し、SQL ステートメントで呼び出すことができる関数です。

エピック

クエリ結果を言語順にソートするコードの開発

タスク	説明	必要なスキル

サンプルデータ用のテーブルを作成します。	Amazon Redshift でテーブルを作成し、次の SQL ステートメントを使用してサンプルデータを挿入します。
	`CREATE TABLE my_table (first_name varchar(30));`
	`INSERT INTO my_table (first_name) VALUES ('ali'), ('Ali'), ('ırmak'), ('IRMAK'),`
		データエンジニア

タスク	説明	必要なスキル
	('irem'), ('İREM'), ('oğuz'), ('OĞUZ'), ('ömer'), ('ÖMER'), ('sedat'), ('SEDAT'), ('şule'), **Note** サンプルデータの最初の名前には、トルコ語アルファベットの特殊文字が含まれています。詳細については、このパターンの追加情報セクションにあるトルコ語の例を参照してください。	

タスク	説明	必要なスキル

サンプルデータのデフォルトソートを確認します。	Amazon Redshift のサンプルデータのデフォルトソートを確認するには、次のクエリを実行します。
	
	`SELECT first_name FROM my_table ORDER BY first_name;`
	
	このクエリは、以前に作成したテーブルからファーストネームのリストを返します。
	
	`first_name`
	`---------------`
	`Ali`
	`IRMAK`
	`OĞUZ`
	`SEDAT`
	`ali`
	`irem`
	`oğuz`
	`sedat`
	`ÖMER`
	`ömer`
	`İREM`
	`ırmak`
	`ŞULE`
	`şule`
	
	デフォルトのバイナリ UTF-8 の順序がトルコ語の特殊文字の言語順序に対応していないため、クエリ結果の順序が正しくありません。
		データエンジニア

タスク	説明	必要なスキル
スカラー Python UDF の作成	スカラー Python UDF を作成するには、次の SQL コードを使用します。 `CREATE OR REPLACE FUNCTION collate_sort (value varchar) RETURNS varchar IMMUTABLE AS $$ def sort_str(val): import string dictionary = { 'I': 'ı', 'ı': 'h~', 'İ': 'i', 'Ş': 's~', 'ş': 's~', 'Ğ': 'g~', 'ğ': 'g~', 'Ü': 'u~', 'ü': 'u~', 'Ö': 'o~', 'ö': 'o~', 'Ç': 'c~', 'ç': 'c~' } for key, value in dictionary.items(): val = val.replace(key, value) return val.lower()`	データエンジニア

タスク	説明	必要なスキル

	`    return sort_str(value)`
	
	`$$ LANGUAGE plpythonu;`	

サンプルデータをクエリします。	Python UDF を使用してサンプルデータをクエリするには、次の SQL クエリーを実行します：
	
	`SELECT first_name FROM my_table ORDER BY collate_order(first_name);`
	
	これで、クエリはトルコ語の順序でサンプルデータを返すようになりました。
	
	`first_name`
	`---------------`
	`ali`
	`Ali`
	`ırmak`
	`IRMAK`
	`irem`
	`İREM`
	`oğuz`
	`OĞUZ`
	`ömer`
	`Ömer`
	`sedat`
	`SEDAT`
	`şule`
	`ŞULE`
		データエンジニア

関連リソース

- 「注文条件」(Amazon Redshift ドキュメント)
- 「スカラー Python UDF の作成」(Amazon Redshift ドキュメント)

追加情報

トルコ語の例

Amazon Redshift は、言語固有のソート順序ではなく、バイナリ UTF-8 の順序に基づいてクエリ結果を返します。つまり、トルコ語の文字を含む Amazon Redshift テーブルに対してクエリを実行した場合、クエリ結果はトルコ語の言語順でソートされません。トルコ語には、ラテン文字にはない 6 つの特殊文字 (、ı、、ö、ü) が含まれています。これらの特殊文字は、次のテーブルに示すように、UTF-8 バイナリの順序に基づいてソートされた結果セットの最後に配置されます。

UTF-8 バイナリの順序	トルコ語の順序付け
a	a
b	b
c	c
d	çe (*)
e	d
f	e
g	f
h	g
i	ğ (*)
j	h
k	ı (*)
l	i

m	j
n	k
o	l
p	m
r	n
s	o
t	ö (*)
u	p
v	r
y	s
z	ş (*)
çe (*)	t
ğ (*)	u
ı (*)	ü (*)
ö (*)	v
ş (*)	y
ü (*)	z

> **Note**
>
> アスタリスク (*) は、トルコ語の特殊文字を示します。

上の表が示すように、特殊文字 ç はトルコ語の順序では c と d の間ですが、UTF-8 バイナリの順序では z の後に表示されます。このモードのスカラー Python UDF は、トルコ語の特殊文字をラテン語の対応する文字に置き換えるために、次の文字置換辞書を使用します。

トルコ語の特殊文字	ラテン語に相当する文字
ç	c~
ı	h~
ğ	g~
ö	o~
ş	s~
ü	u~

> **Note**
>
> チルダ (~) 文字は、対応するトルコ語特殊文字を置き換えるラテン文字の末尾に追加されます。

スカラー Python UDF 関数を変更する

スカラー Python UDF 関数が locate パラメータを受け入れ、マルチトランザクション辞書をサポートするように、このスキーマに基づいて変更するには、次の SQL コードを使用します：

```
CREATE OR REPLACE FUNCTION collate_sort (value varchar, locale varchar)
RETURNS varchar
IMMUTABLE
AS
$$
    def sort_str(val):
       import string
       # Turkish Dictionary
       if locale == 'tr-TR':
            dictionary = {
               'I': 'ı',
```

```
                'ı': 'h~',
                'İ': 'i',
                'Ş': 's~',
                'ş': 's~',
                'Ğ': 'g~',
                'ğ': 'g~',
                'Ü': 'u~',
                'ü': 'u~',
                'Ö': 'o~',
                'ö': 'o~',
                'Ç': 'c~',
                'ç': 'c~'
        }
        # German Dictionary
        if locale == 'de-DE':
            dictionary = {
                ....
                ....
        }

        for key, value in dictionary.items():
            val = val.replace(key, value)

        return val.lower()

    return sort_str(value)

$$ LANGUAGE plpythonu;
```

次のコード例では、変更された Python UDF をクエリする方法を示しています。

```
SELECT first_name FROM my_table ORDER BY collate_order(first_name, 'tr-TR');
```

さまざまな AWS リージョンの S3 バケットからのイベント通知に Lambda 関数をサブスクライブする

作成者: Suresh Konathala、Andrew Preston、Arindom Sarkar

概要

Amazon Simple Storage Service (Amazon S3) イベント通知は、S3 バケット内の特定のイベント (オブジェクト作成イベント、オブジェクト削除イベント、オブジェクト復元イベントなど) の通知を発行します。AWS Lambda 関数を使用して、アプリケーションの要件に従ってこれらの通知を処理できます。ただし、Lambda 関数は、さまざまな AWS リージョンでホストされている S3 バケットからの通知を直接サブスクライブすることはできません。

このパターンのアプローチでは、各リージョンの Amazon Simple Notification Service (Amazon SNS) トピックを使用して、クロスリージョン S3 バケットからの Amazon S3 通知を処理するファンアウトシナリオをデプロイします。これらのリージョン SNS トピックは、Lambda 関数も含む中央リージョンの Amazon Simple Queue Service (Amazon SQS) キューに、Amazon S3 イベント通知を送信します。Lambda 関数はこの SQS キューにサブスクライブし、組織の要件に従ってイベント通知を処理します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Amazon SQS キューと Lambda 関数をホストする中央リージョンを含む、複数のリージョンにある既存の S3 バケット。
- AWS コマンドラインインターフェイス (AWS CLI) がインストール済みおよび設定済み。詳細については、AWS CLI ドキュメントの「CLI バージョン 2 のインストール、更新、およびアンインストール」を参照してください。
- Amazon SNS のファンアウトシナリオに精通していること。Amazon SNS のトピック作成の詳細については、「Amazon SNS のドキュメント」を参照してください。

アーキテクチャ

次の図は、このパターンのアプローチのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. Amazon S3 は、S3 バケットに関するイベント通知 (オブジェクトの作成、削除、復元など) を同じリージョンの SNS トピックに送信します。
2. SNS トピックは、中央リージョンの SQS キューにイベントを発行します。
3. SQS キューは Lambda 関数のイベントソースとして構成され、Lambda 関数のイベントメッセージをバッファします。
4. Lambda 関数は SQS キューにメッセージがないかポーリングし、アプリケーションの要件に従って Amazon S3 イベント通知を処理します。

テクノロジースタック

- Lambda
- Amazon SNS
- Amazon SQS
- Amazon S3

ツール

- CLI - AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールで、コマンドラインシェルのコマンドで AWS サービスとインタラクトします。最小限の構成で、コマンドプロンプトからブラウザベースの AWS マネジメントコンソールで提供される機能と同等の機能を実装する AWS CLI コマンドを実行できます。
- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理し、プロビジョニングすることが可能です。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。

- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージ配信や送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。
- Amazon SQS – Amazon Simple Queue Service (Amazon SQS) は、配信ソフトウェアシステムとコンポーネントを統合および分離できる、安全で耐久性があり、利用可能なホスト型キューを提供します。Amazon SQSは、スタンダードおよび FIFO キューの両方をサポートしています。

エピック

中央リージョンに SQS キューと Lambda 関数を作成する

タスク	説明	必要なスキル
Lambda トリガーを使用して SQS キューを作成する。	AWS マネジメントコンソールにサインインし、AWS Lambda ドキュメントの「Amazon SQS で Lambda を使用する」チュートリアルの手順に従って、中央リージョンに次のリソースを作成します。 • Lambda 実行ロール • Amazon S3 イベントを処理する Lambda 関数 • SQS キュー **Note** SQS キューを Lambda 関数のイベントソースとして設定していることを確認します。	AWS DevOps、クラウドアーキテクト

SNS トピックを作成し、必要な各リージョンの S3 バケットのイベント通知を設定する

タスク	説明	必要なスキル
Amazon S3 イベント通知を受信する SNS トピックを作成します。	Amazon S3 イベント通知を受信したいリージョンに SNS トピックを作成します。Amazon SNS のトピック作成の詳細については、Amazon SNS のドキュメントの「SNS トピックを作成する」を参照してください。 **Important** SNS トピックの Amazon リソースネーム (ARN) を必ず記録してください。	AWS DevOps、クラウドアーキテクト
中央 SQS キューにSNSトピックをサブスクライブします。	中央リージョンがホストする SQS キューに SNS トピックをサブスクライブします。これに関する詳細については、Amazon SNS ドキュメントの「SNS トピックにサブスクライブする」を参照してください。	AWS DevOps、クラウドアーキテクト
SNS トピックのアクセスポリシーを更新する。	1. Amazon SNS コンソールを開き、[トピック] を選択してから、先に作成した SNS トピックを選択します。 2. [編集] を選択し、[アクセスポリシー - オプション] セクションを展開します。	AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル

	3. 次のアクセスポリシーを SNS トピックにアタッチして Amazon S3 にsns:publish を許可し、[保存] を選択します。
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Sid": "0",
	"Effect": "Allow",
	"Principal": {
	"Service": "s3.amazonaws.com"
	},
	"Action": "sns:Publish",
	"Resource": "arn:aws:sns:us-west-2::s3Events-SNSTopic-us-west-2"
	}
	]
	}	

タスク	説明	必要なスキル
リージョンの各 S3 バケットに通知を設定します。	リージョンの各 S3 バケットにイベント通知を設定します。これに関する詳細については、Amazon S3 ドキュメントの「Amazon S3 コンソールを使用したイベント通知の有効化と構成」を参照してください。 **Note** 送信先セクションで、SNS トピックを選択し、前に作成した SNS トピックの ARN を指定します。	AWS DevOps、クラウドアーキテクト
必要なすべてのリージョンについてこのエピックを繰り返します。	**Important** 中央リージョンを含め、Amazon S3 イベント通知を受信するリージョンごとに、このエピックのタスクを繰り返します。	AWS DevOps、クラウドアーキテクト

関連リソース

- アクセスポリシーの構成 (Amazon SQS ドキュメント)
- SQS キューをイベントソースとして構成する (AWS Lambda ドキュメント)
- Lambda 関数を開始するための SQS キューの構成 (Amazon SQS ドキュメント)
- AWS::Lambda::Function リソース (AWS CloudFormation ドキュメント)

データを Apache Parquet に変換するための 3 つの AWS Glue ETL ジョブタイプ

作成者: Adnan Alvee (AWS)、Karthikeyan Ramachandran (AWS)、および Nith Govindasivan (AWS)

概要

Amazon Web Services (AWS) クラウド上で、AWS Glue は完全に管理された抽出、変換、ロード (ETL) サービスです。AWS Glue は、データを分類し、クリーニングし、リッチ化し、様々なデータストアやデータストリーム間で確実に移動させるための費用対効果を実現します。

このパターンでは、AWS Glue でさまざまなジョブタイプが提供され、3 つの異なるスクリプトを使用して ETL ジョブの作成を示しています。

AWS Glue を使用して Python シェル環境で ETL ジョブを記述できます。マネージド型の Apache Spark 環境で Python (PySpark) または Scala を使用して、バッチ ETL ジョブとストリーミング ETL ジョブの両方を作成することもできます。ETL ジョブの作成を始めるにあたって、このパターンは Python シェル、PySpark、および Scala を使用するバッチ ETL ジョブに焦点を当てています。Python シェルジョブは、より少ない計算能力を必要とするワークロードを対象としています。マネージド Apache Spark 環境は、高い計算能力を必要とするワークロードを対象としています。

Apache Parquet は、効率的な圧縮とエンコードスキームをサポートするように構築されています。データを列指向に保存するため、分析ワークロードを高速化できます。データを Parquet に変換すると、長期的にはストレージ容量、コスト、時間を節約できます。Parquet について詳しくは、ブログ記事「Apache Parquet:オープンソースの列指向データ形式でヒーローになる方法」をご覧ください。

前提条件と制限

前提条件

- AWS Identity and Access Management (IAM) ロール (ロールがない場合は、「追加情報」セクションを参照してください) 。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Glue

- Amazon Simple Storage Service (Amazon S3)
- Apache Parquet

自動化とスケール

- 「AWS Glue ワークフロー」は ETL パイプラインの完全自動化をサポートします。
- データ処理ユニット (DPU) の数またはワーカーのタイプを変更して、水平方向と垂直方向にスケールさせることができます。

ツール

AWS サービス

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- 「AWS Glue」は、さまざまなデータストアやデータストリーム間でデータを分類、クリーニング、強化、移動するための完全マネージド型の ETL サービスです。

その他のツール

- 「Apache Parquet」は、ストレージとデータの取得を目的として設計されたオープンソースの列指向データファイル形式です。

設定

AWS Glue ETL の計算能力を設定するには、以下の設定を使用してください。コストを削減するには、このパターンで提供されるワークロードを実行するときには最小限の設定を使用してください。

- Python シェル — 1 DPU を使用して 16 GB のメモリを利用するか、0.0625 DPU を使用して 1 GB のメモリを利用できます。このパターンでは、AWS Glue コンソールのデフォルトである 0.0625 DPU を使用します。
- Python または Scala for Spark — コンソールで Spark 関連のジョブタイプを選択した場合、AWS Glue はデフォルトで 10 個のワーカーと G.1X ワーカータイプを使用します。このパターンでは、2 つのワーカー (許容される最小数) と標準のワーカータイプを使用するため、十分かつ費用対効果が高くなります。

次の表は、Apache Spark 環境のさまざまな AWS Glue ワーカータイプを示しています。Python シェルジョブは Apache Spark 環境を使用して Python を実行しないため、このテーブルには含まれていません。

	標準	G.1X	G.2X
vCPU	4	4	8
「メモリ」	16 GB	16 GB	32 GB
ディスク容量	50 GB	64 GB	128 GB
ワーカーごとのエグゼキューター	2	1	1

コード

IAM ロールやパラメータ設定など、このパターンで使用されるコードについては、「追加情報」セクションを参照してください。

エピック

データをアップロードする

タスク	説明	必要なスキル
データを新規または既存の S3 バケットにアップロードします。	お使いののアカウント内で新しい S3 バケットを作成するか、既存の S3 バケットを使用します。「添付ファイル」セクションから sample_data.csv ファイルをアップロードし、S3 バケットとプレフィックスの場所を書き留めます。	AWS 全般

AWS Glue ジョブを作成して実行する

タスク	説明	必要なスキル
AWS Glue ジョブを作成します。	AWS Glue コンソールの ETL セクションで、AWS Glue ジョブを追加します。適切なジョブタイプ、AWS Glue バージョン、対応する DPU/ワーカータイプとワーカー数を選択します。詳細については、「設定」セクションを参照してください。	デベロッパー、クラウド、またはデータ
入力位置と出力位置を変更します。	AWS Glue ジョブに対応するコードをコピーし、「データのアップロード」エピックでメモした入力場所と出力場所を変更します。	デベロッパー、クラウド、またはデータ
パラメータを設定します。	「追加情報」セクションに記載されているスニペットを使用して、ETL ジョブのパラメータを設定できます。AWS Glue は内部で次の 4 つの引数名を使用します。 • `--conf` • `--debug` • `--mode` • `--JOB_NAME` `--JOB_NAME` パラメータは、AWS Glue コンソールで明示的に入力する必要があります。[ジョブ]、[ジョブの編集]、[セキュリティ設定]、	デベロッパー、クラウド、またはデータ

タスク	説明	必要なスキル
	[スクリプトライブラリ]、および [ジョブパラメータ] (オプション) を選択します。`--JOB_NAME` キーとして入力し、値を提供します。また、AWS コマンドラインインターフェイス (AWS CLI) または AWS Glue API でもこのパラメータを設定できます。`--JOB_NAME` パラメーターは Spark によって使用され、Python シェル環境のジョブでは必要ありません。 `--` をすべてのパラメーター名の前に追加する必要があります。そうしないと、コードは機能しません。たとえば、コードスニペットの場合、ロケーションパラメータは `--input_loc` と `--output_loc` によって呼び出される必要があります。	
ETL ジョブを実行する。	ジョブを実行し、出力を確認します。元のファイルからどれだけの容量が削減されたかに注意してください。	デベロッパー、クラウド、またはデータ

関連リソース

リファレンス

- Apache Spark
- 「AWS Glue:仕組み」

- 「AWS Glue の価格」

チュートリアルと動画

- 「AWS Glue とは何ですか?」

追加情報

IAM ロール

AWS Glue ジョブを作成するときは、次のコードスニペットに示されている権限を持つ既存の IAM ロールまたは新しいロールを使用できます。

新規ロールを作成するには、次の YAML コードを使用します。

```
# (c) 2022 Amazon Web Services, Inc. or its affiliates. All Rights Reserved. This AWS
 Content is provided subject to the terms of the AWS Customer
# Agreement available at https://aws.amazon.com/agreement/ or other written agreement
 between Customer and Amazon Web Services, Inc.

AWSTemplateFormatVersion: "2010-09-09"

Description: This template will setup IAM role for AWS Glue service.

Resources:
  rGlueRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
      Policies:
        - PolicyName: !Sub "${AWS::StackName}-s3-limited-read-write-inline-policy"
          PolicyDocument:
```

```
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "s3:PutObject"
                  - "s3:GetObject"
                Resource: "arn:aws:s3:::*/*"
      Tags:
        - Key  : "Name"
          Value : !Sub "${AWS::StackName}"

Outputs:
  oGlueRoleName:
    Description: AWS Glue IAM role
    Value:
      Ref: rGlueRole
    Export:
      Name: !Join [ ":", [ !Ref "AWS::StackName", rGlueRole ] ]
```

AWS Glue Python シェル

Python コードでは、Pandas ライブラリと PyArrow ライブラリを使用してデータを Parquet に変換します。Pandas ライブラリは既に使用可能です。PyArrow ライブラリは 1 回限りの実行なので、パターンを実行するとダウンロードされます。ホイールファイルを使用して PyArrow をライブラリに変換し、そのファイルをライブラリパッケージとして提供できます。wheel ファイルのパッケージングについての詳細は、「独自の Python ライブラリの提供」を参照してください。

AWS Glue Python シェルパラメータ

```
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["input_loc", "output_loc"])
```

AWS Glue Python シェルコード

```
from io import BytesIO
import pandas as pd
import boto3
import os
import io
import site
from importlib import reload
```

```
from setuptools.command import easy_install
install_path = os.environ['GLUE_INSTALLATION']
easy_install.main( ["--install-dir", install_path, "pyarrow"] )
reload(site)
import pyarrow


input_loc = "s3://bucket-name/prefix/sample_data.csv"
output_loc = "s3://bucket-name/prefix/"


input_bucket = input_loc.split('/', 1)[0]
object_key = input_loc.split('/', 1)[1]


output_loc_bucket = output_loc.split('/', 1)[0]
output_loc_prefix = output_loc.split('/', 1)[1]


s3 = boto3.client('s3')
obj = s3.get_object(Bucket=input_bucket, Key=object_key)
df = pd.read_csv(io.BytesIO(obj['Body'].read()))


parquet_buffer = BytesIO()
s3_resource = boto3.resource('s3')
df.to_parquet(parquet_buffer, index=False)
s3_resource.Object(output_loc_bucket, output_loc_prefix +  'data' +
 '.parquet').put(Body=parquet_buffer.getvalue())
```

Python を使った AWS Glue ースパークジョブ

Python で AWS Glue Spark ジョブタイプを使用するには、ジョブタイプとして　[Spark] を選択します。AWS Glue バージョンとして、ジョブの起動時間が改善された Spark 3.1、Python 3 (グルーバージョン 3.0) を選択してください。

AWS Glue Python パラメータ

```
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME", "input_loc", "output_loc"])
```

Python コードを使用した AWS Glue ースパークジョブ

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
from awsglue.dynamicframe import DynamicFrame
from awsglue.utils import getResolvedOptions
from awsglue.job import Job


sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

input_loc = "s3://bucket-name/prefix/sample_data.csv"
output_loc = "s3://bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(\
    connection_type = "s3", \
    connection_options = {
        "paths": [input_loc]}, \
    format = "csv",
    format_options={
        "withHeader": True,
        "separator": ","
    })


outputDF = glueContext.write_dynamic_frame.from_options(\
    frame = inputDyf, \
    connection_type = "s3", \
    connection_options = {"path": output_loc \
        }, format = "parquet")
```

圧縮されたサイズの大きいファイルが多数ある場合 (たとえば、1,000 個のファイルがそれぞれ約 3 MB)、次のコードに示すように、recurse パラメータと compressionType パラメータを組み合わせてプレフィックス内にあるすべてのファイルを読み取ります。

```
input_loc = "bucket-name/prefix/"
output_loc = "bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(
```

```
                    connection_type = "s3",
                    connection_options = {"paths": [input_loc],
                                          "compressionType":"gzip","recurse" :"True",
                                          },
                    format = "csv",
                    format_options={"withHeader": True,"separator": ","}
                    )
```

圧縮された小さなファイルが多数ある場合 (たとえば、それぞれが約 133 KB の 1,000 ファイル)、compressionType パラメーターと recurse パラメーターとともに、groupFiles パラメーターを使用してください。groupFiles パラメータは小さなファイルを複数の大きなファイルにグループ化し、groupSize パラメータは指定されたサイズ (たとえば 1 MB) にグループ化を制御します。次のコードス二ペットは、コード内でこれらのパラメーターを使用する例を示しています。

```
input_loc = "bucket-name/prefix/"
output_loc = "bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(
                    connection_type = "s3",
                    connection_options = {"paths": [input_loc],
                                          "compressionType":"gzip","recurse" :"True",
                                           "groupFiles" :"inPartition",
 "groupSize" :"1048576",
                                          },
                    format = "csv",
                    format_options={"withHeader": True,"separator": ","}
                    )
```

ワーカーノードを変更しなくても、これらの設定により、AWS Glue ジョブは複数のファイル (大小を問わず、圧縮の有無にかかわらず) を読み取り、Parquet 形式でターゲットに書き込むことができます。

Scala を使った AWS Glue ースパークジョブ

Scala で AWS Glue Spark ジョブタイプを使用するには、ジョブタイプとして [Spark] を選択し、言語として Scala を選択します。AWS Glue バージョンとして、ジョブの起動時間が改善された Spark 3.1、Scala 2 (Glue バージョン 3.0) を選択してください。ストレージ容量を節約するために、次の AWS Glue with Scala サンプルでも applyMapping 機能を使用してデータ型を変換しています。

AWS Glue Scala パラメータ

```
import com.amazonaws.services.glue.util.GlueArgParser val args =
 GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME", "inputLoc",
 "outputLoc").toArray)
```

Scala コードを使用した AWS Glue ースパークジョブ

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueScalaApp {
  def main(sysArgs: Array[String]) {

    @transient val spark: SparkContext = SparkContext.getOrCreate()
    val glueContext: GlueContext = new GlueContext(spark)

    val inputLoc = "s3://bucket-name/prefix/sample_data.csv"
    val outputLoc = "s3://bucket-name/prefix/"

    val readCSV = glueContext.getSource("csv", JsonOptions(Map("paths" ->
 Set(inputLoc)))).getDynamicFrame()

    val applyMapping = readCSV.applyMapping(mappings = Seq(("_c0", "string", "date",
 "string"), ("_c1", "string", "sales", "long"),
    ("_c2", "string", "profit", "double")), caseSensitive = false)

    val formatPartition = applyMapping.toDF().coalesce(1)

    val dynamicFrame = DynamicFrame(formatPartition, glueContext)

    val dataSink = glueContext.getSinkWithFormat(
        connectionType = "s3",
        options = JsonOptions(Map("path" -> outputLoc )),
        transformationContext = "dataSink", format =
 "parquet").writeDynamicFrame(dynamicFrame)
  }
```

```
}
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon Athena および Amazon QuickSight を使用して Amazon Redshift 監査ログを視覚化

サンケット・シルシカー（AWS）とゴパル・クリシュナ・バティア（AWS）によって作成された

概要

セキュリティは、Amazon Web Services (AWS) クラウドでのデータベース運用に欠かせないものです。組織は、潜在的なセキュリティインシデントやリスクを検出するために、データベースユーザーのアクティビティと接続を必ず監視する必要があります。このパターンは、セキュリティとトラブルシューティングの目的でデータベースを監視するのに役立ちます。このプロセスは、多くの場合データベース監査と知られます。

このパターンは、Amazon Redshift ログの監査に役立つ Amazon QuickSight のレポートダッシュボード用の Amazon Athena テーブルとビューの作成を自動化する SQL スクリプトを提供します。これにより、データベースアクティビティの監視を担当するユーザーは、データセキュリティ機能に簡単にアクセスできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 既存の Amazon Redshift クラスター。これに関する詳細については、Amazon Redshift ドキュメントの「Amazon Redshift クラスターの作成」を参照してください。
- 既存の Athena ワークグループへのアクセス。詳細については、Amazon Athena ドキュメントの「ワークグループの仕組み」を参照してください。
- 必要な AWS ID およびアクセス管理 (IAM) アクセス管理 (IAM) アクセス権限が付いた、既存の Amazon Simple Storage Service (Amazon S3) ソースバケット。詳細については、Amazon Redshift ドキュメントの「データベース監査ロギング」にある「Amazon Redshift 監査ログのバケット権限」を参照してください。

アーキテクチャ

テクノロジースタック

- Athena
- Amazon Redshift
- Amazon S3
- QuickSight

ツール

- Amazon Athena – Amazon Athena は、Amazon S3 内のデータを標準 SQL を使用して簡単に分析できるインタラクティブなクエリサービスです。
- Amazon QuickSight — QuickSight は、スケーラブルでサーバーレス、埋め込みが可能な、機械学習を活用したビジネスインテリジェンス (BI) サービスです。
- Amazon Redshift – Amazon Redshift は、エンタープライズレベル、ペタバイト規模、フルマネージド型のデータウェアハウスサービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。

エピック

Amazon Redshift クラスターを設定します。

タスク	説明	必要なスキル
Amazon Redshift クラスターの監査ログ記録を有効にします。	1. AWS マネジメントコンソールにサインインし、Amazon Redshift コンソールを開いて「CLUSTERS」を選択し、ロギングを有効にするクラスターを選択します。 2. 「プロパティ」タブを選択し、Amazon Redshift ドキュメントの「コンソールを使用して監査を設定す	DBA、データエンジニア

タスク	説明	必要なスキル
	る」の指示に従って監査を有効にします。	

タスク	説明	必要なスキル
Amazon Redshift クラスターのパラメータグループでロギングを有効にします。	接続ログ、ユーザーログ、およびユーザーアクティビティログは、AWS マネジメントコンソール、Amazon Redshift API リファレンス、または AWS コマンドラインインターフェイス (AWS CLI)を使用して、同時に有効になります。 ユーザーアクティビティログの監査については、`enable_user_activity_logging` データベースパラメータも有効にする必要があります。監査ログ作成機能のみを有効にし、関連するパラメータを有効にしない場合、データベース監査ログは接続ログとユーザーログの情報のみを記録し、ユーザーアクティビティログの情報は記録しません。この`enable_user_activity_logging` パラメータはデフォルトでは有効になっていませんが、`false`から`true`に変更することで有効にできます。 **Important** パラメータを有効にして新しいクラスター`user_activity_logging` パラメータグループ	DBA、データエンジニア

タスク	説明	必要なスキル
	を作成し、Amazon Redshift クラスターにアタッチする必要があります。詳細については、Amazon Redshift ドキュメントの「クラスターの変更」を参照してください。 このタスクの詳細については、Amazon Redshift ドキュメントの「Amazon Redshift パラメータグループ」と「コンソールを使用した監査の設定」を参照してください。	

タスク	説明	必要なスキル
Amazon Redshift クラスターロギングのための S3 バケットのアクセス権限を設定します。	ログ作成を有効にすると、Amazon Redshift はログ作成情報を収集し、 S3 バケットに格納されたログファイルにアップロードします。既存の S3 バケットを使用するか、新しいバケットを作成します。 **Important** Amazon Redshift に S3 バケットにアクセスするために必要な IAM アクセス許可があることを確認します。詳細については、Amazon Redshift ドキュメントの「データベース監査ロギング」の「Amazon Redshift 監査ログのバケット権限」を参照してください。	DBA、データエンジニア

Athena テーブルとビューを作成する

タスク	説明	必要なスキル
S3 バケットから Amazon Redshift 監査ログデータをクエリするための Athena テーブルとビューを作成します。	Amazon Athena コンソールを開き、`AuditLogging.sql` SQL スクリプト (添付) のデータ定義言語 (DDL) クエリを使用して、ユーザーアクティビ	データエンジニア

タスク	説明	必要なスキル
	ティログ、ユーザーログ、接続ログのテーブルとビューを作成します。 詳細と手順については、Amazon Athena ワークショップの「テーブルの作成とクエリの実行」チュートリアルを参照してください。	

QuickSight ダッシュボードでログモニタリングを設定する

タスク	説明	必要なスキル
Athena をデータソースとして使用して QuickSight ダッシュボードを作成します。	Amazon QuickSight コンソールを開き、Amazon Athena Athena ワークショップの「Athena を使用して QuickSight で視覚化」チュートリアルの指示に従って QuickSight ダッシュボードを作成します。	DBA、データエンジニア

関連リソース

- Athenaでテーブルの作成やクエリを実行する
- Athena を使用して QuickSight で視覚化する

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon QuickSight を使用してすべての AWS アカウントの IAM 認証情報レポートを視覚化します

作成者: Parag Nagwekar (AWS)、Arun Chandapillai (AWS)

概要

> **Warning**
>
> IAM ユーザーは長期認証情報を保有するため、セキュリティ上のリスクが生じます。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

AWS Identity and Access Management (IAM) 認証情報レポートを使用して、組織のセキュリティ、監査、コンプライアンス要件を満たすのに役立ちます。認証情報レポートには、AWS アカウントのすべてのユーザーのリストと、パスワード、アクセスキー、多要素認証 (MFA) デバイスなど、認証情報のステータスが表示されます。「AWS Organizations」が管理する複数のAWSアカウントに対して、認証情報レポートを使用できます。

このパターンには、Amazon QuickSight ダッシュボードを使用して組織内のすべての AWS アカウントの IAM 認証情報レポートを作成および共有するのに役立つステップとコードが含まれています。ダッシュボードは組織内の利害関係者と共有できます。このレポートは、組織が、次のような目標を絞ったビジネス成果を達成するのに役立ちます。

- IAM ユーザーに関連するセキュリティインシデントを特定します。
- IAM ユーザーのシングルサインオン (SSO) 認証への移行をリアルタイムで追跡する
- IAM ユーザーがアクセスした AWS リージョンを追跡する
- コンプライアンスを維持
- 他の利害関係者との情報共有

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- 組織内のメンバーアカウント。
- Organizations アカウントにアクセスする権限を持つ IAM ロール
- AWS コマンドラインインターフェース (AWS CLI) バージョン 2は、「インストール済み」および「設定済み」
- 「Amazon QuickSight エンタープライズエディション」の「サブスクリプション」

アーキテクチャ

テクノロジースタック

- Amazon Athena
- Amazon EventBridge
- Amazon QuickSight
- Amazon Simple Storage Service (Amazon S3)
- AWS Glue
- AWS Identity and Access Management (IAM)
- AWS Lambda
- AWS Organizations

ターゲットアーキテクチャ

次の図は、複数の AWS アカウントから IAM 認証情報レポートデータをキャプチャするワークフローを設定するためのアーキテクチャを示しています。

1. EventBridge は Lambda 関数を毎日呼び出します。
2. Lambda 関数は、組織全体のすべての AWS アカウントで IAM ロールを引き受けます。次に、この関数は IAM 認証情報レポートを作成し、レポートデータを一元管理された S3 バケットに保存します。S3 バケットで暗号化を有効にし、パブリックアクセスを無効にする必要があります。
3. AWS Glue クローラーは S3 バケットを毎日クロールし、それに応じて Athena テーブルを更新します。
4. QuickSight は認証情報レポートからデータをインポートして分析し、視覚化して関係者と共有できるダッシュボードを作成します。

ツール

AWS サービス

- Amazon Athena は、Amazon S3 内のデータを標準 SQL を使用して簡単に分析できるインタラクティブなクエリサービスです。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。たとえば、Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。
- Amazon QuickSight は、視覚化、分析、レポート生成に使用できるクラウドスケールのビジネスインテリジェンス (BI) サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

コード

このパターンのコードは GitHub 内の「getiamcredsreport-allaccounts-org」リポジトリで利用できます。このリポジトリのコードを使用して、Organizations の AWS アカウント全体の IAM 認証情報レポートを作成し、一元的に保存できます。

エピック

インフラストラクチャを設定します。

タスク	説明	必要なスキル
Amazon QuickSight Enterprise Editionをセットアップします。	1. AWS アカウントの Amazon QuickSight Enterprise エディションを有効化します。Amazon QuickSight - 詳細については、「Amazon QuickSight でのユーザーアカウントの管理」を参照してください。	AWS 管理者、AWS DevOps、クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	2. ダッシュボードにアクセス許可を付与するには、QuickSight ユーザーの Amazon リソースネーム (ARN) を取得します。	
Amazon QuickSight を Amazon S3 および Athena と統合します。	AWS CloudFormationスタックをデプロイする前に、QuickSightがAmazon S3とAthenaを使用するように「承認」する必要があります。	AWS 管理者、AWS DevOps、クラウド管理者、クラウドアーキテクト

インフラストラクチャを準備する

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	1. コマンド`git clone https://github.com/aws-samples/getiamcredsreport-allaccounts-org` を実行して、GitHub「getiamcredsreport-allaccounts-org」リポジトリをローカルマシンにクローンします。	AWS 管理者
インフラストラクチャを準備します。	1. AWS マネジメントコンソールにサインインし、CloudFront コンソールを開きます。 2. ナビゲーションペインで [スタックの作成] を選択し、[新しいリソース (標準) を使用] を選択します。	AWS 管理者

タスク	説明	必要なスキル
	3. 「リソースの識別」ページで「次へ」を選択します。 4. 「テンプレートの指定」ページの「テンプレートソース」で、「テンプレートファイルのアップロード」を選択します。 5. 「ファイルを選択」を選択し、クローンした GitHub リポジトリから`Cloudformation-createcredrepo.yaml` ファイルを選択して、「次へ」を選択します。 6. 「パラメータ」で IAM ロール で`IAMRoleName` を更新します。これは、Lambda に組織のすべてのアカウントで引き受けてもらいたい IAM ロールでなければなりません。このロールは認証情報レポートを作成します。注：スタック作成のこのステップでは、ロールはすべてのアカウントに存在していなくてもかまいません。 7. 「パラメータ」で、Lambda がすべてのアカウントの認証情報を保存できる S3 バケットの名前で`S3BucketName` を更新します。	

タスク	説明	必要なスキル
	8. スタック名に対して、スタック名を入力します。 9. [送信] を選択します。 10. Lambda 関数のロール名を書き留めておきます。	
IAM 権限ポリシーを作成します	以下の権限を使用して、組織全体のすべての AWS アカウントに「IAM ポリシーを作成」します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iam:GenerateCredentialReport", "iam:GetCredentialReport" ], "Resource": "*" } ] }`	AWS DevOps、クラウド管理者、クラウドアーキテクト、データエンジニア

タスク	説明	必要なスキル
IAM ロールを信頼ポリシーとともに作成します。	1. AWS アカウント用の IAM ロールを作成し、前のステップで作成した権限ポリシーをアタッチします。 2. 次の信頼ポリシーをIAM ロールにアタッチします。 `{ "Version":"2012-10-17", "Statement":[ { "Effect": "Allow", "Principal":{ "AWS":[ "arn:aws:iam::<MasterAccountID>:role/<LambdaRole>" ] }, "Action": "sts:AssumeRole" } ] }` **Important** を、前に書き留めた Lambda ロールの ARN `arn:aws:iam::<MasterAccountID>:role/`	クラウド管理者、クラウドアーキテクト、AWS 管理者

タスク	説明	必要なスキル
	`<LambdaRole>` に置き換えます。 **Note** 通常、組織はオートメーションを使用して AWS アカウントの IAM ロールを作成します。この自動化は、可能であればを使用することをお勧めします。または、コードリポジトリにある`CreateRoleforOrg.py` スクリプトを使用することもできます。このスクリプトには、既存の管理者ロール、またはすべての AWS アカウントで IAM ポリシーとロールを作成する権限を持つその他の IAM ロールが必要です。	

タスク	説明	必要なスキル
データを視覚化するように Amazon QuickSight を設定します。	1. 認証情報を使用して「QuickSight にサインイン」します。 2. Athena を使用して (`iamcredreportdb` データベースと“`cfn_iamcredreport`” テーブルを使用して)「データセットを作成」し、自動的に「データセットを更新」します。 3. QuickSight で分析を作成します。 4. QuickSight ダッシュボードを作成します。	AWS DevOps、クラウド管理者、クラウドアーキテクト、データエンジニア

追加情報

追加の考慮事項

以下の点を考慮してください。

- CloudFormation を使用してインフラストラクチャをデプロイしたら、Lambda と AWS Glue がそれぞれのスケジュールに従って実行されるまで、Amazon S3 で作成され、Athena によって分析されるレポートを受け取るのを待つことができます。または、Lambda を手動で実行して Amazon S3 のレポートを取得し、次に AWS Glue クローラーを実行してデータから作成された Athena テーブルを取得することもできます。
- QuickSight は、ビジネス要件に基づいてデータを分析および視覚化するための強力なツールです。QuickSight の「パラメータ」を使用して、選択したデータフィールドに基づいてウィジェットデータを制御できます。また、QuickSight分析を使用して、データセットからパラメータ(例えば、それぞれ`partition_0`、`partition_1`、`user`のようなアカウント、デート、ユーザーフィールド)を作成し、Account、Date、Userのパラメータのコントロールを追加することができます。

- 独自の QuickSight ダッシュボードを作成するには、AWS ワークショップスタジオのウェブサイトの「QuickSight ワークショップ」を参照してください。
- QuickSight ダッシュボードのサンプルを見るには、GitHub 「getiamcredsreport-allaccounts-org」コードリポジトリを参照してください。

ターゲットを絞ったビジネス成果

このパターンを使用すると、次のようなビジネス成果を達成できます。

- IAM ユーザーに関連するセキュリティインシデントを特定 — 組織内のすべての AWS アカウントのすべてのユーザーを 1 つの画面で調査できます。IAM ユーザーが最近アクセスした個々の AWS リージョンと、使用したサービスの傾向を追跡できます。
- IAM ユーザーの SSO 認証への移行をリアルタイムで追跡 — SSO を使用すると、ユーザーは 1 つの認証情報で 1 回サインインすると、複数の AWS アカウントとアプリケーションにアクセスできます。IAM ユーザーを SSO に移行することを計画している場合、このパターンは SSO に移行し、すべての AWS アカウントの IAM ユーザー認証情報の使用状況 (AWS マネジメントコンソールへのアクセスやアクセスキーの使用など) を追跡するのに役立ちます。
- IAM ユーザーがアクセスした AWS リージョンを追跡する — データ主権やコスト管理など、さまざまな目的でリージョンへの IAM ユーザーのアクセスを制御できます。すべての IAM ユーザーによるリージョンの使用状況を追跡することもできます。
- コンプライアンスを維持 — 最小権限の原則に従うことで、特定のタスクの実行に必要な特定の IAM アクセス権限のみを付与できます。また、AWS サービス、AWS マネジメントコンソールへのアクセス、および認証情報の長期使用状況を追跡できます。
- 他の利害関係者との情報の共有 — IAM 認証情報レポートや AWS アカウントへのアクセスを許可しなくても、厳選されたダッシュボードを他の利害関係者と共有できます。

その他のパターン

- Amazon Cognito と AWS Amplify UI を使用して既存の React アプリケーションユーザーを認証する
- Amazon Textract を使用して PDF ファイルからコンテンツを自動的に抽出する
- Amazon SageMaker AI Studio Lab の時系列に DeepAR を使用してコールドスタート予測モデルを構築する
- AWS Cost Explorer を使用して Amazon EMR クラスターの詳細なコストと使用状況レポートを作成する
- Amazon RDS と Amazon Aurora の詳細なコストと使用状況レポートを作成する
- AWS Cost Explorer を使用して、AWS Glue ジョブの詳細なコストと使用状況レポートを作成します。
- AWS CodePipeline CI/CD パイプラインで AWS Glue ジョブをデプロイする
- Amazon QuickSight ダッシュボードをローカルの Angular アプリケーションに埋め込みます。
- Amazon Redshift クラスターが作成時に暗号化されていることを確保
- Amazon EMR 保管中のデータの暗号化が起動時に有効になっていることを確保
- Amazon DynamoDB テーブルのストレージコストを推定
- Terraform を使用して Amazon Redshift SQL クエリを実行する
- データレイク内の AWS IoT SiteWise メタデータ属性を抽出してクエリする
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用して Db2 z/OS データインサイトを生成する
- SageMaker ノートブックインスタンスに、別の AWS アカウントの CodeCommit リポジトリへの一時的なアクセスを許可する
- Amazon Data Firehose リソースが AWS KMS キーで暗号化されていない場合の識別とアラート
- psycopg2 ライブラリを にインポート AWS Lambda して PostgreSQL データベースとやり取りする
- Microsoft Sentinel で AWS セキュリティログを取り込んで分析する
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する

- AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する
- Amazon S3 用 AWS PrivateLink と DistCP を使用して、オンプレミスの Hadoop 環境から Amazon S3 にデータを移行する
- カウチベースサーバーから AWS 上のカウチベースカペラへの移行
- オンプレミスの Cloudera ワークロードを AWS 上の Cloudera データプラットフォームに移行する
- Amazon EMR クラスターの起動時に転送中の暗号化をモニタリングする
- Application Recovery Controller を使用して EMR クラスターのマルチ AZ フェイルオーバーを管理する
- IaC 原則を使用して Amazon Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する
- GitHub Actions を使用して AWS CloudFormation テンプレートに基づいて AWS Service Catalog 製品をプロビジョニングする
- pytest フレームワーク AWS Glue を使用して で Python ETL ジョブのユニットテストを実行する
- AWS ParallelCluster 用の Grafana モニタリングダッシュボードを設定する
- アカウント間で Amazon Redshift クラスターから Amazon S3 にデータをアンロードする
- 新しい Amazon Redshift クラスターに必要な SSL エンドポイントがあることを確認する
- 新しい Amazon Redshift クラスターが VPC で起動することを検証
- フラスコと AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化

コンピューティング

トピック

- コンテナとマイクロサービス
- サーバーレス
- ネットワーク
- コンテンツ配信

コンテナとマイクロサービス

トピック

- Amazon EKS コンテナから Amazon Neptune データベースにアクセスする
- AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします
- AWS Fargate、AWS PrivateLink、Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします。
- AWS PrivateLink と Network Load Balancer を使用して、Amazon EKS 上のコンテナアプリケーションにプライベートアクセスします。
- Amazon EKS の AWS プライベート CA を使用して AWS App Mesh の mTLS をアクティベートします
- AWS Batch を使用して Amazon RDS for PostgreSQL DBインスタンスのバックアップを自動化します
- CI/CD パイプラインを使用して Amazon EKS へのノード終了ハンドラーのデプロイを自動化する
- CI/CD パイプラインを使用して Amazon EKS へ Java アプリケーションを自動的にビルドし、デプロイする
- Amazon ECS タスク定義を作成し、Amazon EFS を使用して EC2 インスタンスにファイルシステムをマウントする
- コンテナイメージを使用して Lambda 関数をデプロイする
- AWS Fargate を使用して Amazon ECS に Java マイクロサービスをデプロイする
- Amazon EKS と Amazon S3 の Helm チャートリポジトリを使用して Kubernetes のリソースとパッケージをデプロイする
- Amazon EKS にサンプル Java マイクロサービスをデプロイし、Application Load Balancer を使用してマイクロサービスを公開する
- AWS Copilot を使用してクラスター化されたアプリケーションを Amazon ECS にデプロイする
- gRPC ベースのアプリケーションを Amazon EKS クラスターにデプロイし、Application Load Balancer でアクセスする
- Amazon EKS クラスターをデプロイおよびデバッグ
- Elastic Beanstalk を使用してコンテナをデプロイする
- Lambda 関数、Amazon VPC、およびサーバーレスアーキテクチャを使用して静的アウトバウンド IP アドレスを生成する
- Amazon ECR リポジトリに移行するときに、重複するコンテナイメージを自動的に特定する

- Kubernetes DaemonSet を使用して Amazon EKS ワーカーノードに SSM エージェントをインストールします
- prebootstrapコマンドを使用して、SSM エージェントと CloudWatch エージェントを Amazon EKS ワーカーノードにインストールします。
- Amazon EKS Auto Mode を有効にするときに NGINX Ingress Controller を移行する
- コンテナワークロードを Azure Red Hat OpenShift (ARO) から Red Hat OpenShift Service on AWS (ROSA) に移行する
- App2Container が生成した Docker イメージを最適化する
- ノードアフィニティ、テイント、許容範囲を使用して Kubernetes ポッドを Amazon EKS に配置します。
- フィルタリングされた Amazon ECR コンテナイメージをアカウントまたはリージョン間で複製する
- コンテナを再起動せずにデータベースの認証情報をローテーションする
- Amazon ECS Anywhere を使用して Amazon WorkSpaces で Amazon ECS タスクを実行する
- Amazon EC2 Linux インスタンスで ASP.NET Core ウェブ API Docker コンテナを実行する
- AWS Fargate を使用してメッセージ駆動型の大規模なワークロード実行する
- AWS Fargate 搭載の Amazon EKS で Amazon EFS を使用して、永続的なデータストレージでステートフルワークロードを実行する
- Amazon EKS Pod Identity と KEDA を使用して Amazon EKS でイベント駆動型自動スケーリングを設定する
- PGO を使用して Amazon EKS への PostgreSQL デプロイを合理化する
- Application Load Balancer を使用して Amazon ECS の相互 TLS でアプリケーション認証を簡素化する
- その他のパターン

Amazon EKS コンテナから Amazon Neptune データベースにアクセスする

作成者: Ramakrishnan Palaninathan (AWS)

概要

このパターンは、フルマネージド型のグラフデータベースである Amazon Neptune と、コンテナオーケストレーションサービスである Amazon Elastic Kubernetes Service (Amazon EKS) との間の接続を確立し、Neptune データベースにアクセスします。Neptune DB クラスターは、 上の仮想プライベートクラウド (VPC) 内に制限されています AWS。このため、Neptune にアクセスするには、接続を有効にするために VPC を慎重に設定する必要があります。

Amazon Relational Database Service (Amazon RDS) for PostgreSQL とは異なり、Neptune は一般的なデータベースアクセス認証情報に依存しません。代わりに、認証に AWS Identity and Access Management (IAM) ロールを使用します。したがって、Amazon EKS から Neptune に接続するには、Neptune にアクセスするために必要なアクセス許可を持つ IAM ロールを設定する必要があります。

さらに、Neptune エンドポイントには、クラスターが存在する VPC 内でのみアクセスできます。つまり、Amazon EKS と Neptune 間の通信を容易にするためにネットワーク設定を構成する必要があります。特定の要件とネットワーク設定に応じて、Neptune と Amazon EKS 間のシームレスな接続を可能にするように VPC を設定する方法はさまざまです。各メソッドには、アプリケーションのニーズに合ったデータベースアーキテクチャを柔軟に設計できる、個別の利点と考慮事項があります。

前提条件と制限

前提条件

- 最新バージョンの kubectl をインストールします (手順を参照)。バージョンを確認するには、以下を実行します。

```
kubectl version --short
```

- 最新バージョンの eksctl をインストールします (手順を参照)。バージョンを確認するには、以下を実行します。

```
eksctl info
```

- AWS Command Line Interface （ AWS CLI) バージョン 2 の最新バージョンをインストールします (手順を参照 ）。バージョンを確認するには、以下を実行します。

```
aws --version
```

- Neptune DB クラスターを作成します (手順を参照 ）。クラスターの VPC と Amazon EKS 間の通信は、VPC ピアリング、、AWS Transit Gatewayまたは別の方法で確立してください。また、クラスターのステータスが「使用可能」であり、セキュリティグループのポート 8182 にインバウンドルールがあることを確認します。
- 既存の Amazon EKS クラスターで IAM OpenID Connect (OIDC) プロバイダーを設定します (手順を参照 ）。

製品バージョン

- Amazon EKS 1.27
- Amazon Neptune エンジンバージョン 1.3.0.0 (2023-11-15「」)

アーキテクチャ

次の図は、Amazon EKS クラスター内の Kubernetes ポッドと Neptune 間の接続を示し、Neptune データベースへのアクセスを提供します。

自動化とスケール

Amazon EKS Horizontal Pod Autoscaler を使用して、このソリューションをスケールできます。

ツール

サービス

- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon Neptune は、高度に接続されたデータセットで動作するアプリケーションの構築と実行に役立つグラフデータベースサービスです。

ベストプラクティス

ベストプラクティスについては、「Amazon EKS ベストプラクティスガイド」の「Identity and Access Management」を参照してください。

エピック

環境変数を設定する

タスク	説明	必要なスキル
クラスターコンテキストを確認します。	Helm またはその他のコマンドラインツールを使用して Amazon EKS クラスターを操作する前に、クラスターの詳細をカプセル化する環境変数を定義する必要があります。これらの変数は、正しいクラスターとリソースをターゲットにするために、後続のコマンドで使用されます。 まず、正しいクラスターコンテキスト内で動作していることを確認します。これにより、後続のコマンドが目的の Kubernetes クラスターに送信されます。現在のコンテキストを確認するには、次のコマンドを実行します。 `kubectl config current-context`	AWS 管理者、クラウド管理者
`CLUSTER_NAME` 変数を定義します。	Amazon EKS クラスターの`CLUSTER_NAME` 環境変数を定義します。次のコマンドで、サンプル値をクラス	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	ターus-west-2 AWS リージョン に適した値に置き換えます。サンプル値を既存のクラスター名eks-workshop に置き換えます。 `export CLUSTER_NAME=$(aws eks describe-cluster --region us-west-2 --name eks-workshop --query "cluster.name" --output text)`	
出力を検証します。	変数が正しく設定されていることを確認するには、次のコマンドを実行します。 `echo $CLUSTER_NAME` このコマンドの出力が、前のステップで指定した入力と一致することを確認します。	AWS 管理者、クラウド管理者

IAM ロールを作成して Kubernetes に関連付ける

タスク	説明	必要なスキル
サービスアカウントを作成します。	サービスアカウントの IAM ロールを使用して Kubernetes サービスアカウントを IAM ロールにマッピングし、Amazon EKS で実行されるアプリケーションのきめ細かなアクセス許可管理を有効にします。eksctl を	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	使用して IAM ロールを作成し、Amazon EKS クラスター内の特定の Kubernetes サービスアカウントに関連付けることができます。 AWS 管理ポリシー`NeptuneFullAccess` は、指定した Neptune クラスターへの書き込みおよび読み取りアクセスを許可します。 **Important** これらのコマンドを実行する前に、クラスターに関連付けられた OIDC エンドポイントが必要です。 という名前の管理 AWS ポリシーに関連付けるサービスアカウントを作成します`NeptuneFullAccess` 。 `eksctl create iamserviceaccount --name eks-neptune-sa --namespace default --cluster $CLUSTER_NAME --attach-policy-arn arn:aws:iam::aws:policy/NeptuneFullAccess --approve --override-existing-serviceaccounts`	

タスク	説明	必要なスキル
	ここで、 eks-neptune-sa は作成するサービスアカウントの名前です。 完了すると、このコマンドは次のレスポンスを表示します。 `2024-02-07 01:12:39 [#] created serviceaccount "default/eks-neptune-sa"`	

タスク	説明	必要なスキル

アカウントが正しく設定されていることを確認します。	eks-neptune-sa サービスアカウントがクラスターのデフォルトの名前空間で正しく設定されていることを確認します。
	`kubectl get sa eks-neptune-sa -o yaml`
	出力は次のようになります:
	`apiVersion: v1`
	`kind: ServiceAccount`
	`metadata:`
	`  annotations:`
	`  eks.amazonaws.com/role-arn: arn:aws:iam::123456789123:role/eksctl-eks-workshop-addon-iamserviceaccount-d-Role1-Q35yKgdQOlmM`
	`  creationTimestamp: "2024-02-07T01:12:39Z"`
	`  labels:`
	`  app.kubernetes.io/managed-by: eksctl`
	`  name: eks-neptune-sa`
	`  namespace: default`
	`  resourceVersion: "5174750"`
	`  uid: cd6ba2f7-a0f5-40e1-a6f4-4081e0042316`
		AWS 管理者、クラウド管理者

タスク	説明	必要なスキル

接続を確認します。	というサンプルポッドをデプロイpod-utilし、Neptune との接続を確認します。
	
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: pod-util`
	`  namespace: default`
	`spec:`
	`  serviceAccountName: eks-neptune-sa`
	`  containers:`
	`  - name: pod-util`
	`  image: public.ecr.aws/patrickc/troubleshoot-util`
	`  command:`
	`  - sleep`
	`  - "3600"`
	`  imagePullPolicy: IfNotPresent`
	
	`kubectl apply -f pod-util.yaml`
	
	`kubectl exec --stdin --tty pod-util -- /bin/bash`
	`bash-5.1# curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://db-neptune-1.cluster-xxxxxxxxxxxx.us-west-2.neptune.amazonaws.com:8182/gremlin`
	`{"requestId":"a4964f2d-12b1-4ed3-8a14-e`
		AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	ff511431a0e","stat us":{"message":"", "code":200,"attrib utes":{"@type":"g: Map","@value":[]}} ,"result":{"data": {"@type":"g:List", "@value":[]},"meta ":{"@type":"g:Map" ,"@value":[]}}} bash-5.1# exit exit	

接続アクティビティを検証する

タスク	説明	必要なスキル
IAM データベース認証を有効にします。	デフォルトでは、Neptune DB クラスターの作成時、IAM データベース認証は無効になっています。IAM データベース認証を有効または無効にするには、 を使用します AWS Management Console。 AWS ドキュメントの手順に従って、Neptune で IAM データベース認証を有効にします。	AWS 管理者、クラウド管理者
接続を確認します。	このステップでは、既に実行中のステータスにあるpod-utilコンテナを操作して、awscurl をインストールし、接続を検証します。	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル

	1. 次のコマンドを実行して、ポッドを検索します。
	`kubectl get pods`
	出力は次のようになります:
	`NAME READY STATUS RESTARTS AGE pod-util 1/1 Running 0 50m`
	2. awscurl をインストールするには、次のコマンドを実行します。
	`kubectl exec --stdin --tty pod-util -- /bin/bash bash-5.1# pip3 install awscurl Installing collected packages: idna, configparser, configargparse, charset-normalizer, certifi, requests, awscurl Successfully installed awscurl-0.32 certifi-2024.2.2 charset-normalizer-3.3.2 configargparse-1.7 configparser-6.0.0 idna-3.6 requests-2.31.0 bash-5.1# awscurl https://db-neptune-1.cluster-xxxxxxxxxxxx.us-west-2.neptune.amazonaws.com:8182/status --`	

タスク	説明	必要なスキル
	region us-west-2 --service neptune-db {"status":"healthy","startTime":"Thu Feb 08 01:22:14 UTC 2024","dbEngineVersion":"1.3.0.0.R1","role":"writer","dfeQueryEngine":"viaQueryHint","gremlin":{"version":"tinkerpop-3.6.4"},"sparql":{"version":"sparql-1.1"},"opencypher":{"version":"Neptune-9.0.20190305-1.0"},"labMode":{"ObjectIndex":"disabled","ReadWriteConflictDetection":"enabled"},"features":{"SlowQueryLogs":"disabled","ResultCache":{"status":"disabled"},"IAMAuthentication":"enabled","Streams":"disabled","AuditLog":"disabled"},"settings":{"clusterQueryTimeoutInMs":"120000","SlowQueryLogsThreshold":"5000"}}	

トラブルシューティング

問題	ソリューション
Neptune データベースにアクセスできません。	サービスアカウントにアタッチされている IAM ポリシーを確認します。実行するオペレーションに必要なアクション (など`neptune:Connec,neptune:DescribeDBInstances`) が許可されていることを確認します。

関連リソース

- Kubernetes サービスアカウント AWS を使用して へのアクセスを Kubernetes ワークロードに許可する (Amazon EKS ドキュメント)
- サービスアカウントの IAM ロール (Amazon EKS ドキュメント)
- 新しい Neptune DB クラスターの作成 (Amazon Neptune ドキュメント)

AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします

作成者: Kirankumar Chandrashekar (AWS)

概要

このパターンは、Network Load Balancer の背後にある Amazon Elastic Container Service (Amazon ECS) の Docker コンテナアプリケーションをプライベートでホストし、AWS PrivateLink を使用してアクセスする方法を示しています。その後、専用ネットワークで、Amazon Web Services（AWS）のクラウド上のサービスに安全にアクセスできるようになります。Amazon Relational Database Service (Amazon RDS) を使用して、Amazon ECS で実行される高可用性 (HA) アプリケーション用のリレーショナルデータベースをホストします。アプリケーションに永続的なストレージが必要な場合、Amazon Elastic File System (Amazon EFS) が使用されます。

Dockerアプリケーションを実行する Amazon ECS サービスは、フロントエンドに Network Load Balancer を備えており、AWS PrivateLink経由でアクセスできる仮想プライベートクラウド（VPC）エンドポイントと関連付けることができます。この VPC エンドポイントサービスは、VPC エンドポイントを使用して他の VPC と共有できます。

Amazon EC2 Auto Scaling の代わりに、AWS Fargate を使用することもできます。詳細については、「AWS Fargate、AWS PrivateLink、Network Load Balancerを使用して Amazon ECS でコンテナアプリケーションにプライベートにアクセス」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Linux、macOSまたは Windows にインストールし、設定されている「AWS コマンドラインインターフェイス (AWS CLI) バージョン 2」
- Linux、macOSまたは Windows にインストールし設定された「Docker」
- Docker 上で動作するアプリケーション

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon EC2 Auto Scaling
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Lambda
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

自動化とスケール

- 「AWS CloudFormation」を使用することで、「インフラストラクチャをコード」として使用してこのパターンを作成できます。

ツール

- 「Amazon EC2」— Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。
- 「Amazon EC2 Auto Scaling」— Amazon EC2 Auto Scaling は、アプリケーションの負荷を処理するために使用できる適切な数の Amazon EC2 インスタンスを確保するのに役立ちます。
- 「Amazon ECS」— Amazon Elastic Container Service (Amazon ECS) は、クラスター上のコンテナーの実行、停止、管理を簡単にする、拡張性の高い高速なコンテナー管理サービスです。
- 「Amazon ECR」— Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。
- 「Amazon EFS」— Amazon Elastic File System (Amazon EFS)は、AWS クラウドサービスやオンプレミスのリソースで使用できる、シンプルでスケーラブルな、フルマネージドされた伸縮自在な NFS ファイルシステムを提供します。

- 「AWS Lambda」– Lambda は、サーバーのプロビジョニングや管理を行わずにコードを実行するためのコンピューティング サービスです。
- 「Amazon RDS」— Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケールをより簡単にするウェブサービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。Web スケールのコンピューティングを開発者が容易にできるように設計されています。
- 「AWS Secrets Manager」— Secrets Manager は、Secrets Manager に API 呼び出しを提供してプログラムでシークレットを取得することで、コード内のハードコーディングされた認証情報 (パスワードを含む) を置き換えるのに役立ちます。
- 「Amazon VPC」— Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。
- 「Elastic Load Balancing」— Elastic Load Balancing は、受信するアプリケーションまたはネットワーク トラフィックを、複数のアベイラビリティーゾーン内の Amazon EC2 インスタンス、コンテナ、IP アドレスなどの複数のターゲットに分散します。
- 「Docker」— Docker を使用すると、開発者はあらゆるアプリケーションを軽量でポータブルな自給自足のコンテナとして簡単に梱包、出荷、および実行できます。

エピック

ネットワークコンポーネントの作成

タスク	説明	必要なスキル
VPC を作成します。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。[VPC の作成] を選択して、[VPC とその他] を選択します。 2. VPC の名前を入力し、適切な CIDR ブロック範囲を選択します。 3. 2 つのアベイラビリティーゾーン、2 つのパブリック	クラウド管理者

タスク	説明	必要なスキル
	サブネットと 4 つのプライベートサブネットを指定します。2 つのプライベートサブネットは Amazon ECS タスク用で、2 つのプライベートサブネットは Amazon RDS データベース用です。 4. アベイラビリティーゾーンごとに 1 つの NAT ゲートウェイを指定します。 5. [Create VPC (VPC の作成)] を選択します。	

ロードバランサーの作成

タスク	説明	必要なスキル
Network Load Balancer を作成します。	1. Amazon EC2 コンソールを開き、VPC を含む AWS リージョンを選択します。 2. [ロードバランシング] で [ロードバランサー] を選択し、[ロードバランサーの作成] を選択します。 3. [Network Load Balancer] を選択し、[作成] を選択します。 4. ロードバランサーの設定ページで、Network Load Balancer とリスナーを設定します。重要：Network Load Balancer のスキーム	クラウド管理者

タスク	説明	必要なスキル
	は必ず社内として選択してください。 5. 該当するセキュリティ設定を選択し、セキュリティグループとターゲットグループを設定します。インスタンスまたは IP をルーティングの設定セクションのターゲットタイプとしてを選択します。ターゲットを登録しないようにしてください。 6. すべての設定を行ったら、次へ:レビューを選択して、作成を選択します。	

タスク	説明	必要なスキル
Application Load Balancer を作成します。	1. Amazon EC2 コンソールで、VPC を含む同じリージョンを選択します。 2. [ロードバランシング] で [ロードバランサー] を選択し、[ロードバランサーの作成] を選択します。 3. [Application Load Balancer] を選択し、[作成] を選択します。 4. **Important** Application Load Balancer とそのリスナーを設定します。Application Load Balancer のスキームを内部として必ず選択してください。 5. 該当するセキュリティ設定を選択し、セキュリティグループとターゲットグループを設定します。インスタンスまたは IP をルーティングの設定セクションのターゲットタイプとしてを選択します。ターゲットを登録しないようにしてください。	クラウド管理者

タスク	説明	必要なスキル
	6. すべての設定を行ったら、次へ:レビューを選択して、作成を選択します。	

「Amazon EFS ファイルシステムの作成」

タスク	説明	必要なスキル
Amazon EFS ファイルシステムを作成します。	1. Amazon EFS コンソールを開き、ファイルシステムの作成を選択します。 2. [ファイルシステムの作成] ダイアログボックスで、ファイルシステム名を入力し、VPC を選択します。 3. [作成] を選択してファイルシステムを作成します。 4. Amazon EFS ファイルシステムを設定します。	クラウド管理者
サブネットのターゲットをマウントします。	1. Amazon EFS コンソールに戻り、[ファイルシステム] を選択します。[ファイルシステム] ページには、アカウント内の Amazon EFS ファイル システムが表示されます。 2. 作成したファイルシステムを選択し、[管理] を選択すると、[アベイラビリティーゾーン] が表示されます。マウントターゲットを追加するには、[マウントターゲットの追加] を選択し、	クラウド管理者

タスク	説明	必要なスキル
	作成した 4 つのプライベートサブネットを追加します。	
サブネットがターゲットとしてマウントされていることを確認します。	1. Amazon EFS コンソールで、[ファイルシステム] を選択します。 2. [Network(ネットワーク)]を選択して、既存のマウントターゲットのリストを表示します。作成した 4 つのサブネットがこれらに含まれていることを確認してください。	クラウド管理者

S3 バケットを作成する

タスク	説明	必要なスキル
S3 バケットを作成する。	Amazon S3 コンソールを開き、必要に応じてアプリケーションの静的アセットを保存する S3 バケットを作成します。	クラウド管理者

Secrets Manager シークレットを作成する

タスク	説明	必要なスキル
Secrets Manager シークレットを暗号化するために、AWS KMS キーを作成します。	AWS Key Management Service (AWS KMS) コンソールを開き、KMS キーを作成します。	クラウド管理者

タスク	説明	必要なスキル
Amazon RDS パスワードを保存する Secrets Manager シークレットを作成します。	1. AWS Secrets Manager コンソールを開き、[新しいシークレットを保存する] を選択して新しいシークレットを作成します。 2. 作成した KMS キーを選択し、新しいシークレットを保存します。	クラウド管理者

Amazon RDS DB インスタンスの作成

タスク	説明	必要なスキル
DB サブネットグループを作成します。	1. Amazon RDS コンソールを開き、[サブネットグループ] を選択します。 2. [DB サブネットグループの作成] を選択し、DB サブネットグループ用の名前と説明を入力します。 3. 以前に作成した VPC を選択して、アベイラビリティーゾーンとサブネットを選択します。次に [作成] を選択します。	クラウド管理者
Amazon RDS DB インスタンスを作成します。	プライベートサブネット内に Amazon RDS インスタンスを作成して設定します。HA のため、[マルチ AZ] がオンになっていることを確認してください。	クラウド管理者

タスク	説明	必要なスキル
Amazon RDS インスタンスにデータをロードします。	アプリケーションに必要なリレーショナルデータを Amazon RDS インスタンスにロードします。このプロセスは、アプリケーションのニーズや、データベーススキーマの定義方法や設計方法により、異なっています。	クラウド管理者、DBA

Amazon ECS コンポーネントの作成

タスク	説明	必要なスキル
EKS クラスターを作成します。	1. Amazon MSK コンソールを開き、[クラスター] を選択します。 2. [クラスターの作成] を選択し、必要な仕様に応じて ECS クラスターを設定します。	クラウド管理者
Docker イメージを作成します。	[関連リソース] セクションの指示に従い、Docker イメージを作成します。	クラウド管理者
Amazon ECR リポジトリを作成します。	1. Amazon ECR コンソールで [リポジトリ] を選択します。 2. [リポジトリの作成] を選択し、リポジトリの一意の名前を入力します。 3. 必要に応じて AWS KMS 暗号化を含め、仕様に応じてリポジトリを設定します。	クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
Amazon ECR レジストリに対し、Docker CLI を認証します。	Amazon ECR リポジトリ用の Docker クライアントを認証するには、AWS CLI で `aws ecr get-login-password` コマンドを実行します。	クラウド管理者
Docker イメージを Amazon ECR リポジトリにプッシュします。	1. プッシュする Docker イメージを特定し、AWS CLI で `docker images` コマンドを実行します。 2. Amazon ECR レジストリ、リポジトリ、およびオプションのイメージタグ名の組み合わせを使用してイメージにタグを付けます。 3. `docker push` コマンドを実行することで、Docker イメージをプッシュします。 4. 必要なすべてのイメージについても同様のステップを繰り返します。	クラウド管理者

タスク	説明	必要なスキル
Amazon ECS タスク定義を作成します。	Amazon ECSで Docker コンテナを実行するには、タスク定義が必要です。 1. Amazon ECS コンソールに戻り、[タスク定義] を選択して、[新しいタスク定義の作成] を選択します。 2. [互換性の選択] ページで、タスクで使用する起動タイプを選択し、[次のステップ] を選択します。 ⚠ Important タスク定義の設定方法については、関連リソースセクションのタスク定義の作成を参照してください。Amazon ECR にプッシュした Docker イメージを必ず指定してください。	クラウド管理者
Amazon ECS サービスを作成する	前に作成した ECS クラスターを使用して Amazon ECS サービスを作成します。 、起動タイプとして必ず Amazon EC2 を選択し、前の手順で作成したタスク定義と Application Load Balancer のターゲットグループを選択してください。	クラウド管理者

Amazon EC2 Auto Scaling グループの作成

タスク	説明	必要なスキル
起動設定を作成します。	Amazon EC2 コンソールを開き、起動設定を作成します。ユーザーデータに、EC2 インスタンスが必要な ECS クラスターに参加できるようにするコードが含まれていることを確認します。必要なコードの例として、関連リソースセクションを参照してください。	クラウド管理者
Amazon EC2 Auto Scaling グループを作成します。	Amazon EC2 コンソールに戻り、Auto Scaling で Auto Scaling グループを選択します。Amazon EC2 Auto Scaling をセットアップします。プライベートサブネットが選択されていることを確認し、前に作成した設定を開始します。	クラウド管理者

AWS PrivateLinkの設定

タスク	説明	必要なスキル
AWS PrivateLink エンドポイントを設定します。	1. Amazon VPC コンソールで AWS PrivateLink エンドポイントを作成します。 2. このエンドポイントを Network Load Balancer に関連付けます。これにより、Amazon ECS でホストされているアプリケーショ	クラウド管理者

タスク	説明	必要なスキル
	ンを顧客にプライベートに利用させるようになります。 詳細については、[関連リソース] セクションを参照してください。	

VPC エンドポイントの作成

タスク	説明	必要なスキル
VPC エンドポイントを作成します。	前に作成した AWS PrivateLink エンドポイント用の VPC エンドポイントを作成します。VPC エンドポイントの完全修飾ドメイン名 (FQDN) は、AWS PrivateLink エンドポイント FQDN を指します これにより、DNS エンドポイントがアクセスできる VPC エンドポイントサービスへの耐障害性のあるネットワークインタフェースが作成されます。	クラウド管理者

Lambda 関数を作成する

タスク	説明	必要なスキル
Lambda 関数を作成します。	AWS Lambda コンソールで、Lambda関数を作成して、Network Load Balancer のターゲットとして Applicati	アプリ開発者

タスク	説明	必要なスキル
	on Load Balancer IPアドレスを更新します。詳細については、AWS Lambda を使用して Application Load Balancer の静的 IP アドレスを有効にするブログ記事を参照してください。	

関連リソース

ロードバランサーの作成

- Amazon ECS の Network Load Balancer を使用する
- 「Network Load Balancer の作成」
- Amazon ECS 用の Application Load Balancer を使用する
- 「Application Load Balancer の作成」

Amazon EFS ファイルシステムの作成

- 「Amazon EFS ファイルシステムの作成」
- 「Amazon EFS でマウントターゲットを作成」

S3 バケットの作成

- S3 バケットを作成する

Secrets Manager シークレットの作成

- 「AWS KMS でキーを作成」
- 「AWS Secrets Manager でシークレットを作成する」

Amazon RDS インスタンスの作成

- 「Amazon RDS DB インスタンスの作成」

Amazon ECS コンポーネントの作成

- 「Amazon ECS クラスターの作成」
- 「Docker イメージの作成」
- 「Amazon ECR リポジトリの作成」
- 「Amazon ECR リポジトリで Docker を認証」
- 「イメージを Amazon ECR リポジトリにプッシュ」
- 「Amazon ECS タスク定義の作成」
- 「Amazon ECS サービスの作成」

Amazon EC2 Auto Scaling グループを作成します。

- 「起動設定を作成する」
- 起動設定を使用して Auto Scaling グループを作成する
- 「Amazon EC2 ユーザーデータを使用してコンテナインスタンスをブートストラップする」

AWS PrivateLinkの設定：

- 「VPC エンドポイントサービス (AWS PrivateLink)」

VPC エンドポイントの作成

- 「インターフェイス VPC エンドポイント (AWS PrivateLink)」

Lambda 関数の作成

- 「Lambda 関数の作成」

その他のリソース

- 「Application Load Balancer に静的 IP アドレスを使用」
- 「AWS PrivateLink を介したサービスへの安全なアクセス」

AWS Fargate、AWS PrivateLink、Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします。

作成者: Kirankumar Chandrashekar (AWS)

概要

このパターンは、Network Load Balancer の後ろで AWS Fargate 起動タイプの Amazon Elastic Container Service (Amazon ECS) を使用することで、Amazon Web Services (AWS) クラウドで Docker コンテナアプリケーションをプライベートでホストし、AWS PrivateLink を使用してアプリケーションにアクセスする方法を示しています。Amazon Relational Database Service (Amazon RDS) を使用して、Amazon ECS で実行される高可用性 (HA) アプリケーション用のリレーショナルデータベースをホストします。アプリケーションに永続的なストレージが必要な場合、Amazon Elastic File System (Amazon EFS) を使用できます。

このパターンでは、Docker アプリケーションを実行する Amazon ECS サービスに「Fargate 起動タイプ」を使用し、フロントエンドにはNetwork Load Balancerを使用します。また、AWS PrivateLink 経由でアクセスするために、仮想プライベートクラウド (VPC)　エンドポイントと関連付けられることが可能です。この VPC エンドポイントサービスは、VPC エンドポイントを使用して他の VPC と共有できます。

Fargate と Amazon ECS を使用すると、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのサーバーやクラスターを管理することなくコンテナを実行できます。Fargate の代わりに、Amazon EC2 Auto Scaling グループを使用することもできます 詳細については、「AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスする」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Linux、macOSまたは Windows にインストールし、設定されている「AWS コマンドラインインターフェイス (AWS CLI) バージョン 2」
- Linux、macOSまたは Windows にインストールし設定された「Docker」
- Docker 上で動作するアプリケーション

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon EFS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Fargate
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

自動化とスケール

- 「AWS CloudFormation」を使用することで、「インフラストラクチャをコード」として使用してこのパターンを作成できます。

ツール

AWS のサービス

- Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブル、信頼性の高いマネージド AWS コンテナイメージレジストリサービスです。
- Amazon Elastic Container Service (Amazon ECS) は、クラスター上のコンテナの実行、停止、管理を容易にする、高度にスケーラブルで高速なコンテナ管理サービスです。
- Amazon Elastic File System (Amazon EFS) は、AWS クラウドサービスとオンプレミスリソースで使用できる、シンプルでスケーラブルなフルマネージド型の Elastic NFS ファイルシステムを提供します。

- AWS Fargate は、Amazon EC2 インスタンスのAmazon EC2 で使用できるテクノロジーです。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースを簡単にセットアップ、運用、スケーリングできるウェブサービスです AWS クラウド。
- Amazon Simple Storage Service (Amazon S3) は、インターネット用のストレージです。Web スケールのコンピューティングを開発者が容易にできるように設計されています。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。
- Elastic Load Balancing (ELB) は、受信アプリケーションまたはネットワークトラフィックをEC2 インスタンス、コンテナ、IP アドレスなど、複数のアベイラビリティーゾーン内の複数のターゲットに分散します。

その他のツール

- Docker を使用すると、開発者はあらゆるアプリケーションを軽量でポータブルで自己完結型のコンテナとして簡単に梱包、出荷、実行できます。

エピック

ネットワークコンポーネントの作成

タスク	説明	必要なスキル
VPC を作成します。	1. AWS マネジメントコンソールにサインインし、Amazon VPC コンソールを開きます。[VPC の作成] を選択して、[VPC とその他] を選択します。 2. VPC の名前を入力し、適切な CIDR ブロック範囲を選択します。	クラウド管理者

タスク	説明	必要なスキル
	3. 2 つのアベイラビリティーゾーン、2 つのパブリックサブネットと 4 つのプライベートサブネットを指定します。2 つのプライベートサブネットは Amazon ECS タスク用で、2 つのプライベートサブネットは Amazon RDS データベース用です。 4. アベイラビリティーゾーンごとに 1 つの NAT ゲートウェイを指定します。 5. [Create VPC (VPC の作成)] を選択します。	

ロードバランサーの作成

タスク	説明	必要なスキル
Network Load Balancer を作成します。	1. Amazon EC2 コンソールを開き、VPC を含むAWS リージョンを選択します。 2. [ロードバランシング] で [ロードバランサー] を選択し、[ロードバランサーの作成] を選択します。 3. [Network Load Balancer] を選択し、[作成] を選択します。 4. ロードバランサーの設定ページで、Network Load Balancer とリスナーを設	クラウド管理者

タスク	説明	必要なスキル
	定します。重要：Network Load Balancer のスキームは必ず [社内] として選択してください。 5. 該当するセキュリティ設定を選択し、セキュリティグループとターゲットグループを設定します。ルーティングの設定セクションでターゲットタイプとして IP を選択します。ターゲットを登録しないようにしてください。 6. すべての設定を行ったら、次へ:レビューを選択して、作成を選択します。 このストーリーやその他のストーリーに関するヘルプは、「関連リソース」セクションを参照してください。	

タスク	説明	必要なスキル
Application Load Balancer を作成します。	1. Amazon EC2 コンソールで、VPC を含む同じリージョンを選択します。 2. [ロードバランシング] で [ロードバランサー] を選択し、[ロードバランサーの作成] を選択します。 3. [Application Load Balancer] を選択し、[作成] を選択します。 4. **Important** Application Load Balancer とそのリスナーを設定します。Application Load Balancer のスキームを内部として必ず選択してください。 5. 該当するセキュリティ設定を選択し、セキュリティグループとターゲットグループを設定します。ルーティングの設定セクションでターゲットタイプとして IP を選択します。ターゲットを登録しないようにしてください。 6. すべての設定を行ったら、次へ:レビューを選択して、作成を選択します。	クラウド管理者

「Amazon EFS ファイルシステムの作成」

タスク	説明	必要なスキル
Amazon EFS ファイルシステムを作成します。	1. Amazon EFS コンソールを開き、ファイルシステムの作成を選択します。 2. [ファイルシステムの作成] ダイアログボックスで、ファイルシステム名を入力し、VPC を選択します。 3. [作成] を選択してファイルシステムを作成します。 4. Amazon EFS ファイルシステムを設定します。	クラウド管理者
サブネットのターゲットをマウントします。	1. Amazon EFS コンソールに戻り、[ファイルシステム] を選択します。[ファイルシステム] ページには、アカウント内の Amazon EFS ファイル システムが表示されます。 2. 作成したファイルシステムを選択して、管理を選択すると、アベイラビリティーゾーンが表示されます。 3. マウントターゲットを追加するには、[マウントターゲットの追加] を選択し、作成した 4 つのプライベートサブネットを追加します。	クラウド管理者

タスク	説明	必要なスキル
サブネットがターゲットとしてマウントされていることを確認します。	1. Amazon EFS コンソールで、[ファイルシステム] を選択します。 2. [Network(ネットワーク)]を選択して、既存のマウントターゲットのリストを表示します。作成した 4 つのサブネットがこれらに含まれていることを確認してください。	クラウド管理者

S3 バケットを作成する

タスク	説明	必要なスキル
S3 バケットを作成する。	Amazon S3 コンソールを開き、必要に応じてアプリケーションの静的アセットを保存する S3 バケットを作成します。	クラウド管理者

Secrets Manager シークレットを作成する

タスク	説明	必要なスキル
Secrets Manager シークレットを暗号化するために、AWS KMS キーを作成します。	AWS Key Management Service (AWS KMS) コンソールを開き、KMS キーを作成します。	クラウド管理者
Amazon RDS パスワードを保存する Secrets Manager シークレットを作成します。	1. AWS Secrets Manager コンソールを開き、[新しいシークレットを保存する] を選	クラウド管理者

タスク	説明	必要なスキル
	択して新しいシークレットを作成します。 2. 作成した KMS キーを選択し、新しいシークレットを保存します。	

Amazon RDS DB インスタンスの作成

タスク	説明	必要なスキル
DB サブネットグループを作成します。	1. Amazon RDS コンソールを開き、サブネットグループを選択します。 2. [DB サブネットグループの作成] を選択し、DB サブネットグループ用の名前と説明を入力します。 3. 以前に作成した VPC を選択して、アベイラビリティーゾーンとサブネットを選択します。次に [作成] を選択します。	クラウド管理者
Amazon RDS DB インスタンスを作成します。	プライベートサブネット内に Amazon RDS インスタンスを作成して設定します。高可用性 (HA) のため、マルチ AZ がオンになっていることを確認してください。	クラウド管理者
Amazon RDS インスタンスにデータをロードします。	アプリケーションに必要なリレーショナルデータを Amazon RDS インスタンスにロードします。このプロセス	DBA

タスク	説明	必要なスキル
	は、アプリケーションのニーズや、データベーススキーマの定義方法や設計方法により、異なっています。	

Amazon ECS コンポーネントの作成

タスク	説明	必要なスキル
EKS クラスターを作成します。	1. Amazon MSK コンソールを開き、[クラスター] を選択します。 2. [クラスターの作成] を選択し、必要な仕様に応じて ECS クラスターを設定します。	クラウド管理者
Docker イメージを作成します。	AWS ドキュメントの指示に従って Docker イメージを作成します。	クラウド管理者
Amazon ECR リポジトリを作成します。	1. Amazon ECR コンソールを開き、リポジトリを選択します。 2. [リポジトリの作成] を選択し、リポジトリの一意の名前を入力します。 3. 必要に応じて AWS KMS 暗号化を含め、仕様に応じてリポジトリを設定します。	クラウド管理者、DevOps エンジニア
Docker イメージを Amazon ECR リポジトリにプッシュします。	1. プッシュする Docker イメージを特定し、AWS CLI で `docker images` コマンドを実行します。	クラウド管理者

タスク	説明	必要なスキル
	2. Amazon ECR レジストリ、リポジトリ、およびオプションのイメージタグ名の組み合わせを使用してイメージにタグを付けます。 3. `docker push` コマンドを実行することで、Docker イメージをプッシュします。 4. 必要なすべてのイメージについても同様のステップを繰り返します。	

タスク	説明	必要なスキル
Amazon ECS タスク定義を作成します。	Amazon ECSで Docker コンテナを実行するには、タスク定義が必要です。 1. Amazon ECS コンソールに戻り、[タスク定義] を選択して、[新しいタスク定義の作成] を選択します。 2. [互換性の選択] ページで、タスクで使用する起動タイプを選択し、[次のステップ] を選択します。 **⚠ Important** タスク定義の設定方法については、関連リソースセクションのタスク定義の作成を参照してください。Amazon ECR にプッシュした Docker イメージを必ず指定してください。	クラウド管理者

タスク	説明	必要なスキル
ECS サービスを作成し、起動タイプとして Fargate を選択します。	1. 前に作成した ECS クラスターを使用して Amazon ECS サービスを作成します。起動タイプとして必ずFargateを選択してください。 2. 前の手順で作成したタスク定義を選択し、Application Load Balancerのターゲットグループを選択します。	クラウド管理者

AWS PrivateLinkの設定

タスク	説明	必要なスキル
AWS PrivateLink エンドポイントを設定します。	1. Amazon VPC コンソールを開き、AWS PrivateLink エンドポイントを作成します。 2. このエンドポイントを Network Load Balancer に関連付けます。これにより、Amazon ECS でホストされているアプリケーションを顧客にプライベートに利用させるようになります。	クラウド管理者

VPC エンドポイントの作成

タスク	説明	必要なスキル
VPC エンドポイントを作成します。	前に作成した AWS PrivateLink エンドポイントの VPC エンドポイントを作成します。VPC エンドポイントの完全修飾ドメイン名 (FQDN) は、AWS PrivateLink エンドポイント FQDN を指しますこれにより、ドメインネームサービスのエンドポイントがアクセスできる VPC エンドポイントサービスへのElastic Network Interface が作成されます。	クラウド管理者

ターゲットを設定する

タスク	説明	必要なスキル
Application Load Balancer をターゲットとして追加します。	Application Load Balancer を Network Load Balancer のターゲットとして追加するには、AWS ドキュメントの指示に従います。	アプリ開発者

関連リソース

ロードバランサーの作成

- Amazon ECS の Network Load Balancer を使用する
- 「Network Load Balancer の作成」
- Amazon ECS 用の Application Load Balancer を使用する

- 「Application Load Balancer の作成」

Amazon EFS ファイルシステムの作成

- 「Amazon EFS ファイルシステムの作成」
- 「Amazon EFS でマウントターゲットを作成」

Secrets Manager シークレットの作成

- 「AWS KMS でキーを作成」
- 「AWS Secrets Manager でシークレットを作成する」

Amazon RDS インスタンスの作成

- 「Amazon RDS DB インスタンスの作成」

Amazon ECS コンポーネントの作成

- 「Amazon ECR リポジトリの作成」
- 「Amazon ECR リポジトリで Docker を認証」
- 「イメージを Amazon ECR リポジトリにプッシュ」
- 「Amazon ECS タスク定義の作成」
- 「Amazon ECS サービスの作成」

その他のリソース

- 「AWS PrivateLink を介したサービスへの安全なアクセス」

AWS PrivateLink と Network Load Balancer を使用して、Amazon EKS 上のコンテナアプリケーションにプライベートアクセスします。

作成者: Kirankumar Chandrashekar (AWS)

概要

このパターンは、Network Load Balancer の背後にAmazon Elastic Kubernetes Service (Amazon EKS) で Docker コンテナアプリケーションをプライベートにホストし、AWS PrivateLink を使用してアプリケーションにアクセスする方法を示しています。その後、専用ネットワークで、Amazon Web Services (AWS) のクラウド上のサービスに安全にアクセスできるようになります。

Docker アプリケーションを実行している Amazon EKS クラスターは、フロントエンドにNetwork Load Balancer があり、仮想プライベートクラウド (VPC) エンドポイントに関連付けて AWS PrivateLink 経由でアクセスできます。この VPC エンドポイントサービスは、VPC エンドポイントを使用して他の VPC と共有できます。

このパターンで説明されている設定は、VPC と AWS アカウント間でアプリケーションアクセスを共有するための安全な方法です。コンシューマーアカウントとプロバイダーアカウント間の接続はグローバル AWS バックボーン上にあり、パブリックインターネットを経由しないため、特別な接続やルーティング設定は必要ありません。

前提条件と制限

前提条件

- 「Docker」、 Linux、macOS または Windows にインストールし、設定されています。
- Docker 上で実行するアプリケーション。
- アクティブなAWS アカウント
- Linux、macOS または Windows にインストールして設定されている「AWS Command Line Interface (AWS CLI) バージョン 2」。
- タグ付けされたプライベートサブネットとホストアプリケーションに設定された既存の Amazon EKS クラスター。詳細については、「Amazon EKS ドキュメント」の「サブネットタギング」を参照してください。
- Kubectl は、Amazon EKS クラスター上のリソースにアクセスするようにインストールし、設定されています。詳細については、Amazon EKS ドキュメントの「kubectl のインストール」を参照してください。

アーキテクチャ

テクノロジースタック

- Amazon EKS
- AWS PrivateLink
- Network Load Balancer

自動化とスケール

- Kubernetes マニフェストは、Git ベースのリポジトリで追跡および管理でき、AWS CodePipeline で継続的インテグレーションと継続的デリバリー (CI/CD) を使用してデプロイできます。
- AWS CloudFormation を使用して、コードとしての Infrastructure as Code (IaC) を使用してこのパターンを作成できます。

ツール

- 「AWS CLI」— AWS コマンドラインインターフェイス (AWS CLI) はオープンソースツールで、コマンドラインシェルのコマンドを使用して AWS サービスと対話できます。
- 「Elastic Load Balancing」— Elastic Load Balancing は、Amazon Elastic Compute Cloud (Amazon EC2)インスタンス、コンテナ、IPアドレスなど、単一または複数のアベイラビリティゾーンにある複数のターゲットに、受信するアプリケーションまたはネットワークトラフィックを分散します。

- 「Amazon EKS」— Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストール、運用、保守することなく、AWS 上で Kubernetes を実行するために使用できるマネージドサービスです。
- 「Amazon VPC」— Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。
- 「Kubectl」 — Kubectl は、Kubernetes クラスターに対してコマンドを実行するためのコマンドラインユーティリティです。

エピック

Kubernetes デプロイとサービスマニフェストファイルをデプロイします。

タスク	説明	必要なスキル

Kubernetes デプロイマニフェストファイルを作成します。	ご要求に応じて以下のサンプルファイルを変更して、デプロイマニフェストファイルを作成します。
	
	`apiVersion: apps/v1`
	`kind: Deployment`
	`metadata:`
	`  name: sample-app`
	`spec:`
	`  replicas: 3`
	`  selector:`
	`  matchLabels:`
	`  app: nginx`
	`  template:`
	`  metadata:`
	`  labels:`
	`  app: nginx`
	`  spec:`
	`  containers:`
	`  - name: nginx`
	`  image:`
	` public.ecr.aws/z9d2n7e1/nginx:1.19.5`
	`  ports:`
	`  - name: http`
	`  containerPort: 80`
	
	Note
	これは、NGINX Docker イメージを使用してデプロイされる NGINX サンプル設定
		DevOps エンジニア

タスク	説明	必要なスキル
	ファイルです。詳細については、「Docker ドキュメント」の「公式 NGINX Docker イメージの使用方法」を参照してください。	
Kubernetes デプロイマニフェストファイルをデプロイします。	次のコマンドを実行し、Amazon EKS クラスターにデプロイマニフェストファイルを適用します。 `kubectl apply -f <your_deployment_file_name>`	DevOps エンジニア

タスク	説明	必要なスキル

Kubernetes サービスマニフェストファイルを作成します。	以下のサンプルファイルをご要求に応じて変更して、サービスマニフェストファイルを作成します。
	
	`apiVersion: v1`
	`kind: Service`
	`metadata:`
	`  name: sample-service`
	`  annotations:`
	`  service.beta.kuber`
	`netes.io/aws-load-`
	`balancer-type: nlb`
	`  service.beta.kuber`
	`netes.io/aws-load-`
	`balancer-internal:`
	` "true"`
	`spec:`
	`  ports:`
	`  - port: 80`
	`  targetPort: 80`
	`  protocol: TCP`
	`  type: LoadBalancer`
	`  selector:`
	`  app: nginx`
	
	⚠ Important
	内部 Network Load Balancer を定義するannotations には、以下を必ず含めてください。
	
	`service.beta.kuber`
	`netes.io/aws-load-`
	`balancer-type: nlb`
		DevOps エンジニア

タスク	説明	必要なスキル
	`service.beta.kubernetes.io/aws-load-balancer-internal: "true"`	
Kubernetes サービスのマニフェストファイルをデプロイします。	次のコマンドを実行し、サービスマニフェストファイルを Amazon EKS クラスターに適用します。 `kubectl apply -f <your_service_file_name>`	DevOps エンジニア

エンドポイントを作成する

タスク	説明	必要なスキル
Network Load Balancer 名を記録します。	次のコマンドを実行して、Network Load Balancer 名を取得します。 `kubectl get svc sample-service -o wide` AWS PrivateLink エンドポイントの作成に必要な Network Load Balancer 名を記録します。	DevOps エンジニア
AWS PrivateLink エンドポイントを作成します。	AWS マネジメントコンソールにサインインし、Amazon VPC コンソールを開き、AWS PrivateLink エンドポイントを作成します。このエンドポイ	クラウド管理者

タスク	説明	必要なスキル
	ントをNetwork Load Balancerに関連付けることにより、アプリケーションは顧客が非公開に利用できるようになります。詳細については、Amazon VPC ドキュメントの「VPC エンドポイントサービス (AWS PrivateLink)」を参照してください。 **Important** コンシューマーアカウントがアプリケーションにアクセスする必要がある場合は、コンシューマーアカウントの AWS アカウント ID を AWS PrivateLink エンドポイント設定の許可されたプリンシパルリストに追加する必要があります。詳細については、「Amazon VPC ドキュメント」の「エンドポイントサービスのアクセス権限の追加と削除」を参照してください。	

タスク	説明	必要なスキル
VPC エンドポイントを作成します。	Amazon VPC コンソールで、[エンドポイントサービス] を選択し、[エンドポイントサービスの作成] を選択します。AWS PrivateLink エンドポイント用の VPC エンドポイントを作成します。 VPC エンドポイントの完全修飾ドメイン名 (FQDN) は AWS PrivateLink エンドポイントの FQDN を指します。これにより、DNS エンドポイントがアクセスできる VPC エンドポイントサービスへの耐障害性のあるネットワークインタフェースが作成されます。	クラウド管理者

関連リソース

- 公式 NGINX Docker イメージの使用
- 「Amazon EKS でのネットワークロードバランシング」
- 「VPC エンドポイントサービス (AWS PrivateLink)」
- 「エンドポイントサービスのアクセス権限の追加または削除」

Amazon EKS の AWS プライベート CA を使用して AWS App Mesh の mTLS をアクティベートします

作成者: Omar Kahil (AWS)、Emmanuel Saliu (AWS)、Muhammad Shahzad (AWS)、Andy Wong (AWS)

概要

このパターンは、AWS App Mesh の AWS プライベート認証局 (AWS プライベート CA) からの証明書を使用して、Amazon Web Services (AWS) に相互トランスポート層セキュリティ (mTLS) を実装する方法を示しています。全員のためのセキュアプロダクションアイデンティティフレームワーク (SPIFFE) を通じて Envoy シークレットディスカバリーサービス (SDS) API を使用しています。SPIFFE はクラウド・ネイティブ・コンピューティング財団 (CNCF) のオープンソース・プロジェクトで、幅広いコミュニティからの支持を得ており、きめ細かく動的なワークロードID管理を実現しています。SPIFFE 標準を実装するには、SPIRE SPIFE ランタイム環境を使用してください。

App Mesh で mTLSを使用すると、TLS上にセキュリティレイヤーが追加され、メッシュ内のサービスが接続を行うクライアントを確認できるため、双方向のピア認証が可能になります。クライアントとサーバーの関係にあるクライアントは、セッションネゴシエーションプロセス中に X.509 証明書も提供します。サーバーは、この証明書を使用してクライアントを識別し、認証します。これは、証明書が信頼できる認証局 (CA) によって発行されたかどうか、また、証明書が有効であるかどうかを確認するのに役立ちます。

前提条件と制限

前提条件

- セルフマネージドノードグループまたはマネージドノードグループを含む Amazon Elastic Kubernetes Service (Amazon EKS) クラスター
- SDS がアクティブ化されたクラスターにデプロイされた App Mesh コントローラー
- AWS プライベート CA によって発行される AWS Certificate Manager (ACM) からのプライベート証明書

機能制限

- SPIRE エージェントは Kubernetes DaemonSet として実行する必要があるため、SPIRE を AWS Fargate にインストールすることはできません。

製品バージョン

- AWS App Mesh Controller Gurt 1.3.0 以降

アーキテクチャ

次の図は、VPC 内の App Mesh を使用した EKS クラスターを示しています。あるワーカーノードの SPIRE サーバーは、他のワーカーノードの SPIRE エージェントと AWS プライベート CA と通信します。Envoy は SPIRE エージェントのワーカーノード間の mTLS 通信に使用されます。

この図表は以下のステップを示しています。

1. 証明書が発行されます。
2. 証明書の署名と証明書をリクエストします。

ツール

AWS サービス

- AWS Private CA – AWS Private Certificate Authority (AWS Private CA) では、オンプレミス CA の運用にかかる投資コストや保守コストなしに、ルート CA や下位 CA を含むプライベート認証機関 (CA) 階層を作成できます。
- AWS App Mesh – AWS App Mesh (App Mesh) は、サービスのモニタリングとコントロールを容易にするサービスメッシュです。App Mesh は、サービスの通信方法を標準化し、アプリケーション内のすべてのサービスについて一貫した可視性とネットワークトラフィックコントロールを実現できます。
- 「Amazon EKS」— Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストール、運用、保守することなく、AWS 上で Kubernetes を実行するために使用できるマネージドサービスです。

その他のツール

- Helm – Helm は、Kubernetes クラスターでのアプリケーションのインストールと管理を支援する Kubernetes のパッケージマネージャーです。このパターンでは、Helm を使用して AWS App Mesh コントローラーをデプロイします。

- AWS App Mesh コントローラチャート — このパターンでは AWS App Mesh コントローラチャートを使用して Amazon EKS で AWS App Mesh を有効にします。

エピック

環境をセットアップする

タスク	説明	必要なスキル
Amazon EKS でApp Mesh をセットアップします。	「リポジトリ」に用意されている基本的なデプロイ手順に従います。	DevOps エンジニア
SPIREをインストールします。	「spire_setup.yaml」を使用して EKS クラスターに SPIRE をインストールします。	DevOps エンジニア
AWS Private CA 証明書をインストールします。	「AWS ドキュメント」の指示に従って、プライベートルート CA の証明書を作成してをインストールします。	DevOps エンジニア
クラスターノードインスタンスロールに権限を付与します。	クラスターノードインスタンスロールにポリシーをアタッチするには、「追加情報」セクションにあるコードを使用します。	DevOps エンジニア
AWS プライベート CA 用の SPIRE プラグインを追加します。	SPIRE サーバー設定にプラグインを追加するには、「追加情報」セクションにあるコードを使用してください。`certificate_authority_arn` Amazon リソースネーム (ARN) を、プライベート CA ARN に置き換えます。使用する署名アルゴリズムは、プライベート CA の署名	DevOps エンジニア

タスク	説明	必要なスキル
	アルゴリズムと同じである必要があります。`your_region` をお使いの AWS リージョンに置き換えます。 プラグインの詳細については、「サーバープラグイン：UpStreamAuthority「aws_pca」」を参照してください。	
バンドル.cert を更新してください。	SPIRE サーバーを作成すると、`spire-bundle.yaml` ファイルが作成されます。`spire-bundle.yaml` ファイル内の`bundle.crt` 値をプライベート CA からパブリック証明書に変更します。	DevOps エンジニア

ワークロードをデプロイして登録します。

タスク	説明	必要なスキル
ノードとワークロードのエントリを SPIRE に登録する。	ノードとワークロード (サービス) を SPIRE Server に登録するには、「リポジトリ」内のコードを使用します。	DevOps エンジニア
mTLS をアクティブにした状態で App Mesh でメッシュを作成します。	マイクロサービスアプリケーションのすべてのコンポーネント (仮想サービス、仮想ルーター、仮想ノードなど) を含む新しいメッシュを App Mesh に作成します。	DevOps エンジニア

タスク	説明	必要なスキル
登録されたエントリを調べる。	以下のコマンドを実行して、ノードとワークロードの登録済みエントリを調べることができます。 `kubectl exec -n spire spire-server-0 -- /opt/spire/bin/spire-server entry show` これにより、SPIRE Agents のエントリが表示されます。	DevOps エンジニア

mTLS トラフィックを検証してください。

タスク	説明	必要なスキル
mTLS トラフィックを検証します。	1. フロントエンドサービスからバックエンドサービスに HTTP ヘッダーを送信し、SPIRE に登録されているサービスで成功応答を確認します。 2. 相互TLS認証の場合、以下のコマンドを実行することで、`ssl.handshake` 統計を調べることができます。 `kubectl exec -it $POD -n $NAMESPACE -c envoy -- curl http://localhost:9901/stats | grep ssl.handshake`	DevOps エンジニア

タスク	説明	必要なスキル
	先ほどのコマンドを実行すると、リスナーのssl.handshake カウントが表示され、以下の例のようになります。 `listener.0.0.0.0_15000.ssl.handshake: 2`	
証明書が AWS プライベート CA から発行されていることを確認します。	SPIRE サーバーのログを見ることで、プラグインが正しく設定され、証明書がアップストリームのプライベート CA から発行されていることを確認できます。以下のコマンドを実行してください。 `kubectl logs spire-server-0 -n spire` 次に、生成されたログを確認します。このコードでは、サーバーの名前がspire-server-0 であり、spire ネームスペースでホストされていると仮定しています。プラグインが正常に読み込まれ、上流のプライベート CA への接続が確立されているはずです。	DevOps エンジニア

関連リソース

- Amazon EKS の AWS App Mesh で SPIFFE/SPIRE で mTLS を使用する

- マルチアカウントの Amazon EKS 環境で SPIFFE/SPIRE を使用して AWS App Mesh で mTLS を有効にする
- このパターンで使用されているウォークスルー
- サーバープラグイン：アップストリーム権限「aws_pca」
- Kubernetes 用クイックスタート

追加情報

クラスターノードインスタンスロールに権限をアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ACMPCASigning",
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:IssueCertificate",
                "acm-pca:GetCertificate",
                "acm:ExportCertificate"
            ],
            "Resource": "*"
        }
    ]
}
AWS Managed Policy: "AWSAppMeshEnvoyAccess"
```

ACM 用の SPIRE プラグインを追加します。

```
Add the SPIRE plugin for ACM
Change certificate_authority_arn to your PCA ARN. The signing algorithm used must be
 the same as the signing algorithm on the PCA. Change your_region to the appropriate
 AWS Region.
UpstreamAuthority "aws_pca" {
    plugin_data {
        region = "your_region"
        certificate_authority_arn = "arn:aws:acm-pca:...."
        signing_algorithm = "your_signing_algorithm"
    }
}
```

AWS Batch を使用して Amazon RDS for PostgreSQL DBインスタンスのバックアップを自動化します

作成者: Kirankumar Chandrashekar (AWS)

概要

PostgreSQL データベースのバックアップは重要なタスクで、通常 pg_dump ユーティリティ を使用して完了します。このユーティリティでは、デフォルトで COPY コマンドを使用して、PostgreSQL データベースのスキーマとデータダンプを作成します。ただし、複数の PostgreSQL データベースを定期的にバックアップする必要がある場合、このプロセスは繰り返しになる可能性があります。PostgreSQL データベースがクラウドでホストされている場合は、Amazon Relational Database Service (Amazon RDS) の PostgreSQL 用の Amazon Relational Database Service (Amazon RDS) により提供される 自動バックアップ 特徴量を活用することもできます。このパターンでは、pg_dump ユーティリティを使用して Amazon RDS for PostgreSQL インスタンスのの定期バックアップを自動化する方法を説明します。

注：手順は Amazon RDS を使用していることを前提としています。ただし、この方法は Amazon RDS の外部でホストされている PostgreSQL データベースにも使用できます。バックアップを取るには、AWS Lambda 関数がデータベースにアクセスできる必要があります。

時間ベースの Amazon CloudWatch Events イベントは、Amazon RDS 上の PostgreSQL DB インスタンスのメタデータに適用された特定のバックアップタグ を検索する Lambda 関数を開始します。PostgreSQL DB インスタンスに bkp:AutomatedDBDump = Active タグとその他の必要なバックアップタグがある場合、Lambda 関数はデータベースバックアップごとに個別のジョブを AWS Batch に送信します。

AWS Batch はこれらのジョブを処理し、Amazon Simple Storage Service (Amazon S3) バケットにバックアップデータをアップロードします。このパターンでは、Dockerfile と entrypoint.sh ファイルを使用して、AWS Batch ジョブでバックアップを作成するために使用される Docker コンテナイメージを構築します。バックアッププロセスが完了すると、AWS Batch はバックアップの詳細を Amazon DynamoDB のインベントリテーブルに記録します。追加の安全対策として、AWS Batch でジョブが失敗した場合に CloudWatch Events イベントイベントによって Amazon Simple Notification Service (Amazon SNS) 通知が開始されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- 既存のマネージド型または非マネージド型のコンピューティング環境。詳細については、AWS Batch ドキュメントの マネージドコンピューティング環境とアンマネージドコンピューティング環境 を参照してください。
- AWS コマンドラインインターフェイス (AWS CLI) バージョン 2 (インストールと設定)。
- Amazon RDS for PostgreSQL DB インスタンス用 Amazon RDS for PostgreSQL DB インスタンス。
- 既存の S3 バケットを使用する
- Docker、Linux、macOS、または Windows にインストールして設定します。
- Lambda でのコーディングに精通していること。

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch Events
- Amazon DynamoDB
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon RDS
- Amazon SNS
- Amazon S3
- AWS Batch
- AWS Key Management Service (AWS KMS)
- AWS Lambda
- AWS Secrets Manager
- Docker

ツール

- Athena で Amazon CloudWatch Events を使用する – CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。

- 「Amazon ECR」— Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。
- 「Amazon RDS」— Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケールをより簡単にするウェブサービスです。
- 「Amazon SNS」— Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバーへのメッセージ配信を提供するマネージドサービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。
- AWS Batch — AWS Batch では、AWS クラウドでバッチコンピューティングワークロードを実行できます。
- AWS KMS – AWS Key Management Service (AWS KMS)は、データの暗号化に使用される暗号化キーの作成と管理を容易にするマネージド型サービスです。
- AWS Lambda はサーバーをプロビジョニングしたり管理しなくてもコードを実行できるコンピューティングサービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- Docker — Dockerを使用すると、開発者はあらゆるアプリケーションを軽量でポータブルな自給自足のコンテナとして簡単に梱包、出荷および実行できます。

Amazon RDS の PostgreSQL DB インスタンスには、メタデータにタグが適用されている 必要があります。Lambda 関数はタグを検索してバックアップすべき DB インスタンスを識別します。通常は次のタグが使用されます。

タグ	説明
bkp:AutomatedDBDump = アクティブ	Amazon RDS DB インスタンスをバックアップの候補として識別します。
bkp:AutomatedBackupSecret = <secret_name>	Amazon RDS ログイン認証情報を含む Secrets Manager シークレットを識別します。
bkp:AutomatedDBDumpS3Bucket = <s3_bucket_name>	バックアップの送信先となる S3 バケットを識別します。

bkp:AutomatedDBDumpFrequency bkp:AutomatedDBDumpTime	データベースをバックアップする頻度と時間を特定してください。
bkp:pgdumpcommand = <pgdump_command>	バックアップが必要なデータベースを識別します。

エピック

DynamoDB でインベントリテーブルを作成する

タスク	説明	必要なスキル
DynamoDB でテーブルを作成します。	AWS マネジメントコンソールにサインインし、 で Amazon DynamoDB コンソールを開きます。このストーリーやその他のストーリーに関するヘルプは、「関連リソース」セクションを参照してください。	クラウド管理者、DBA
テーブルが作成されたことを確認します。	`aws dynamodb describe-table --table-name <table-name> | grep TableStatus` コマンドを実行します。テーブルが存在する場合、`"TableStatus": "ACTIVE",` コマンドは結果を返します。	クラウド管理者、DBA

AWS Batch で失敗したジョブイベントの SNS トピックを作成する

タスク	説明	必要なスキル
SNS トピックを作成します。	Amazon SNS コンソールを開き、トピックを選択	クラウド管理者

タスク	説明	必要なスキル
	し、JobFailedAlert 名前で SNS トピックを作成します。トピックにアクティブな E メールアドレスを登録し、E メールの受信トレイをチェックして、AWS Notifications からの SNS サブスクリプションメールを確認します。	
AWS Batch の失敗したジョブイベントルールを作成します。	Amazon CloudWatchコンソールに移動し、イベントを選択して ルールの作成 を選択します。詳細オプションを表示、編集 の順に選択します。ターゲットで処理するイベントを選択するパターンを構築, では、既存のテキストを追加情報セクションの「Failed job event」コードに置き換えます。このコードでは、AWS Batch に Failed イベントが発生したときに開始される CloudWatch Events イベントルールを定義します。	クラウド管理者
イベントルールターゲットを追加します。	tターゲット]で、ターゲットの追加 を選択し、「SNS topic」 を選択します。CloudWatch Events ルールの作成と設定	クラウド管理者

Docker イメージを Amazon ECR リポジトリにプッシュするには

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	Amazon ECR コンソールを開き、リポジトリを作成する AWS リージョンを選択します。リポジトリの追加 を選択し、リポジトリの作成 を選択します。要件に従ってリポジトリを構成します。	クラウド管理者
Dockerfile を作成します。	Docker にサインインし、追加情報 セクションの「サンプル Dockerfile」と「サンプル entrypoint.sh ファイル」を使用して Dockerfile を作成します。	DevOps エンジニア
次に、Docker イメージが Amazon ECR イメージリポジトリにプッシュされます。	Docker イメージに Docker ファイルをビルドし、Amazon ECR リポジトリにプッシュします。このストーリーやその他のストーリーに関するヘルプは、関連リソースセクションを参照してください。	DevOps エンジニア

AWS Batch コンポーネントを作成する

タスク	説明	必要なスキル
AWS Batch ジョブ定義を作成する	AWS Batch コンソールを開き、Amazon ECR リポジトリのユニフォームリソース識別子 (URI) `Image` をプロパティ	クラウド管理者

タスク	説明	必要なスキル
	として含むジョブ定義を作成します。	
AWS Batch ジョブキューを設定します。	AWS Batch コンソールで Job キューを選択し、キューの作成を選択します。AWS Batch がコンピューティング環境内のリソースで実行されるまでジョブを保存するジョブキューを作成します。重要：バックアップの詳細を DynamoDB インベントリテーブルに記録する AWS Batch のロジックを必ず記述してください。	クラウド管理者

Lambda 関数を作成して発行する

タスク	説明	必要なスキル
Lambda 関数を作成して、タグを検索します。	PostgreSQL DB インスタンス上のタグを検索し、バックアップ候補を識別する Lambda 関数を作成します。Lambda `bkp:AutomatedDBDump = Active` 関数がタグとその他の必要なタグをすべて識別できることを確認してください。重要：Lambda 関数は、AWS Batch ジョブキューにジョブを追加することも可能でなければなりません。	DevOps エンジニア

タスク	説明	必要なスキル
時間ベースの CloudWatch イベントイベントを作成します。	Amazon CloudWatch コンソールを開き、cron 式を使用して Lambda 関数を定期的に実行する CloudWatch イベントイベントを作成します。スケジュールされたイベントはすべて UTC+0 のタイムゾーンを使用しています。	クラウド管理者

バックアップ自動化のテスト

タスク	説明	必要なスキル
Amazon KMS キーを作成します。	Amazon KMS コンソールを開き、AWS Secrets Manager に保存されている Amazon RDS 認証情報を暗号化するために使用できる KMS キーを作成します。	クラウド管理者
AWS Secrets Manager シークレットを作成する	AWS Secrets Manager コンソールを開き、Amazon RDS for PostgreSQL データベースの認証情報をシークレットとして保存します。	クラウド管理者
必要なタグを PostgreSQL DB インスタンスに追加します。	**⚠ Important** Amazon RDS コンソールを開き、自動的にバックアップしたい PostgreSQL DB インスタンスにタグを追加します。ツールセクションの表に	クラウド管理者

タスク	説明	必要なスキル
	あるタグを使用できます。同じ Amazon RDS インスタンス内の複数の PostgreSQL データベースからのバックアップが必要な場合は、 `bkp:pgdumpcommand` タグの値`-d test:-d test1`として を使用します。 `test`および`test1`はデータベース名です。コロン (:) の後にスペースがないことを確認します。	
バックアップ自動化を検証してください。	バックアップの自動化を確認するには、Lambda 関数を呼び出すか、バックアップスケジュールの開始を待つことができます。バックアッププロセスが完了したら、DynamoDB インベントリテーブルに PostgreSQL DB インスタンスの有効なバックアップエントリがあることを確認します。一致すれば、バックアップ自動化プロセスは成功です。	クラウド管理者

関連リソース

DynamoDB でインベントリテーブルを作成する

- Amazon DynamoDB テーブルの作成

AWS Batch で失敗したジョブイベントの SNS トピックを作成する

- Amazon SNS トピックを作成します。
- AWS Batch で失敗したジョブイベントの SNS アラートを送信する

Docker イメージをを構築して、Amazon ECR リポジトリにプッシュする

- Amazon ECR リポジトリの作成
- ドッカーファイルを作成し、Docker イメージを作成して Amazon ECR にプッシュする

AWS Batch コンポーネントの作成

- AWS Batchジョブ定義の作成
- コンピューティング環境と AWS Batch ジョブキューの設定
- AWS Batch でジョブキューを作成する

Lambda 関数を作成する

- Lambda 関数を作成してコードを記述する
- DynamoDB でLambda を使用する

CloudWatch イベントを作成する

- 時間ベースの CloudWatch イベントイベントを作成する
- クラウドウォッチイベントで cron エクスプレッションを使用する

バックアップ自動化のテスト

- Amazon KMS キーを作成する
- Secrets Manager シークレットを作成する
- Amazon RDS インスタンスにタグを追加する

追加情報

失敗したジョブイベント：

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
  ],
  "detail": {
    "status": [
      "FAILED"
    ]
  }
}
```

サンプル Dockerfile：

```
FROM alpine:latest
RUN apk --update add py-pip postgresql-client jq bash && \
pip install awscli && \
rm -rf /var/cache/apk/*
ADD entrypoint.sh /usr/bin/
RUN chmod +x /usr/bin/entrypoint.sh
ENTRYPOINT ["entrypoint.sh"]
```

サンプル entrypoint.sh ファイル：

```
 #!/bin/bash
set -e
DATETIME=`date +"%Y-%m-%d_%H_%M"`
```

```
FILENAME=RDS_PostGres_dump_${RDS_INSTANCE_NAME}
FILE=${FILENAME}_${DATETIME}

aws configure --profile new-profile set role_arn arn:aws:iam::${TargetAccountId}:role/
${TargetAccountRoleName}
aws configure --profile new-profile set credential_source EcsContainer

echo "Central Account access provider IAM role is: "
aws sts get-caller-identity

echo "Target Customer Account access provider IAM role is: "
aws sts get-caller-identity --profile new-profile

securestring=$(aws secretsmanager get-secret-value --secret-id $SECRETID --output json
 --query 'SecretString' --region=$REGION --profile new-profile)

if [[ ${securestring} ]]; then
    echo "successfully accessed secrets manager and got the credentials"
    export PGPASSWORD=$(echo $securestring | jq --raw-output | jq -r '.DB_PASSWORD')
    PGSQL_USER=$(echo $securestring | jq --raw-output | jq -r '.DB_USERNAME')
    echo "Executing pg_dump for the PostGres endpoint ${PGSQL_HOST}"
    # pg_dump -h $PGSQL_HOST -U $PGSQL_USER -n dms_sample | gzip -9 -c  | aws s3 cp -
 --region=$REGION  --profile new-profile s3://$BUCKET/$FILE
    # in="-n public:-n private"
    IFS=':' list=($EXECUTE_COMMAND);
    for command in "${list[@]}";
      do
        echo $command;
        pg_dump -h $PGSQL_HOST -U $PGSQL_USER ${command} | gzip -9 -c  | aws s3 cp - --
region=$REGION --profile new-profile s3://${BUCKET}/${FILE}-${command}".sql.gz"
        echo $?;
        if  [[ $? -ne 0 ]]; then
            echo "Error occurred in database backup process. Exiting now....."
            exit 1
        else
            echo "Postgresql dump was successfully taken for the RDS endpoint
 ${PGSQL_HOST} and is uploaded to the following S3 location s3://${BUCKET}/${FILE}-
${command}.sql.gz"
            #write the details into the inventory table in central account
            echo "Writing to DynamoDB inventory table"
            aws dynamodb put-item --table-name ${RDS_POSTGRES_DUMP_INVENTORY_TABLE} --
region=$REGION --item '{ "accountId": { "S": "'"${TargetAccountId}"'" }, "dumpFileUrl":
 {"S": "'"s3://${BUCKET}/${FILE}-${command}.sql.gz"'" }, "DumpAvailableTime": {"S":
 "'"`date +"%Y-%m-%d::%H::%M::%S"` UTC"'"}}'
```

```
            echo $?
            if  [[ $? -ne 0 ]]; then
                echo "Error occurred while putting item to DynamoDb Inventory Table.
 Exiting now....."
                exit 1
            else
                echo "Successfully written to DynamoDb Inventory Table
 ${RDS_POSTGRES_DUMP_INVENTORY_TABLE}"
            fi
        fi
      done;
else
    echo "Something went wrong {$?}"
    exit 1
fi

exec "$@"
```

CI/CD パイプラインを使用して Amazon EKS へのノード終了ハンドラーのデプロイを自動化する

Sandip Gangapadhyay (AWS)、John Vargas (AWS)、Pragtideep Singh (AWS)、Sandeep Gawande (AWS)、Viyoma Sachdeva (AWS) によって作成されました

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

Amazon Web Services (AWS) クラウドでは、オープンソースプロジェクトの「AWS Node Termination Handler」を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのシャットダウンを正常に処理できます。AWS Node Termination Handler は、EC2 インスタンスが使用できなくなる原因となるイベントに Kubernetes コントロールプレーンが適切に対応できるようにするのに役立ちます。下記のイベントには、次が含まれます。

- 「EC2 インスタンスの定期メンテナンス」
- 「Amazon EC2 スポットインスタンスの中断」
- 「Auto Scaling グループのスケールイン」
- アベイラビリティーゾーン間の 「Auto Scaling グループのリバランシング」
- API または AWS マネジメントコンソールによる EC2 インスタンスの終了

イベントが処理されないと、アプリケーションコードが正常に停止しない可能性があります。また、完全な可用性を回復するまでに時間がかかったり、ダウンしているノードに誤って作業をスケジューリングしてしまうこともあります。`aws-node-termination-handler` (NTH) は、インスタンスメタデータサービス (IMDS) とキュープロセッサの 2 つの異なるモードで動作できます。二つのノードについての詳細情報については、「Readme ファイル」を参照してください。

このパターンでは、 を使用し AWS CodeCommit、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを通じてキュープロセッサを使用して NTH のデプロイを自動化します。

> **Note**
>
> EKS マネージド型ノードグループを使用している場合は、 は必要ありません`aws-node-termination-handler`。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS マネジメントコンソールでの使用がサポートされているウェブブラウザ。「サポートされるブラウザのリスト」を参照してください。
- 「インストール済み」の AWS Cloud Development Kit (AWS CDK)
- `kubectl`、Kubernetes コマンドラインツールが 「インストール」されています。
- `eksctl`、Amazon Elastic Kubernetes Service (Amazon EKS) の AWS コマンドラインインターフェイス (AWS CLI) が「インストール」されています。
- バージョン 1.20 以降の稼動中の EKS クラスター
- EKS クラスターに接続されているセルフマネージド型ノードグループ。自己管理型ノードグループで Amazon EKS クラスターを作成するには、以下のコマンドを実行します。

```
eksctl create cluster --managed=false --region <region> --name <cluster_name>
```

  `eksctl` の詳細については、「eksctl のドキュメント」を参照してください。
- クラスターAWS Identity and Access Management (IAM) OpenID Connect (OIDC) プロバイダー 詳細について、「クラスターの IAM プロバイダーを作成する」を参照してください。

機能制限

- Amazon EKS サービスをサポートする AWS リージョンを使用する必要があります。

製品バージョン

- Kubernetes バージョン 1.20 以降
- `eksctl` バージョン 0.107.0 以降
- AWS CDK バージョン 2.27.0 またはそれ以降

アーキテクチャ

ターゲットテクノロジースタック

- 仮想プライベートクラウド (VPC)

- EKS クラスター
- Amazon Simple Queue Service (Amazon SQS)
- IAM
- Kubernetes

ターゲットアーキテクチャ

次のダイアグラム表は、ノード終了が開始されたときのエンドツーエンドステップのハイレベルビューです。

このダイアグラムに示されているワークフローは、以下の大まかなステップで構成されています。

1. 自動スケーリングの EC2 インスタンス終了イベントは SQS キューに送信されます。
2. NTH ポッドは SQS キュー内の新しいメッセージを監視します。
3. NTH ポッドは新しいメッセージを受信し、以下を実行します。
   - 新しいポッドがノード上で実行されないように、ノードをコード化します。
   - ノードをドレインし、既存のポッドを退避させます。
   - ライフサイクルフックシグナルを Auto Scaling グループに送信して、ノードを終了できるようにします。

自動化とスケール

- コードは AWS CloudFormation のネストスタックによって支えられた AWS CDK によって管理およびデプロイされます。
- 「Amazon EKS コントロールプレーン」は、複数のアベイラビリティーゾーンで実行され、高可用性を保証します。
- 「自動スケーリング」については、Amazon EKS は Kubernetes「クラスター」オートスケーラーと「カーペンター」をサポートしています。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CodeBuild はフルマネージドの構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- 「Amazon Elastic Kubernetes Service (Amazon EKS)」は、AWS で Kubernetes を実行する際に役立ち、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持する必要はありません。
- 「Amazon EC2 Auto Scaling」は、定義した条件に従って、Amazon EC2 インスタンスを自動的に追加または削除できるように、アプリケーションの可用性を維持するのに役立ちます。
- 「Amazon Simple Queue Service (Amazon SQS)」は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。

その他のツール

- 「kubectl」は、 Kubernetes クラスターに対してコマンドを実行するための Kubernetes コマンドラインツールです。kubectl を使用して、アプリケーションのデプロイ、クラスターリソースの検査と管理、ログの表示を行うことができます。

コード

このパターンのコードは、GitHub 内の「deploy-nth-to-eks」リポジトリで利用できます。コードリポジトリには、以下のファイルとフォルダが含まれます。

- `nth folder` — ノード終了ハンドラー用の AWS CloudFormation テンプレートをスキャンしてデプロイするためのHelm チャート、値ファイル、スクリプト。
- `config/config.json` — アプリケーションの設定パラメータファイル。このファイルには、CDK のデプロイに必要なすべてのパラメーターが含まれています。
- `cdk` — AWS CDK のソースコード。
- `setup.sh` — AWS CDK アプリケーションをデプロイして必要な CI/CD パイプラインとその他の必要なリソースを作成するために使用されるスクリプト。

- `uninstall.sh` — リソースをクリーンアップするために使用されるスクリプト。

例のコードを使用するには、エピックセクションの指示に従います。

ベストプラクティス

AWS ノード終了ハンドラーを自動化する際のベストプラクティスについては、以下を参照してください。

- EKS ベストプラクティスガイド
- 「ノード終了ハンドラー-設定」

エピック

環境をセットアップします。

タスク	説明	必要なスキル

リポジトリを変更します。	SSH (セキュアシェル) を使用してリポジトリをクローンするには、以下のコマンドを実行します。
	`git clone git@github.com:aws-samples/deploy-nth-to-eks.git`
	HTTPSを使用して、リポジトリを複製して、以下のコマンドを実行します。
	`git clone https://github.com/aws-samples/deploy-nth-to-eks.git`
	リポジトリを複製すると、`deploy-nth-to-eks`
		アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	という名前のフォルダーが作成されます。 そのディレクトリに変更します。 `cd deploy-nth-to-eks`	
kubeconfig ファイルを設定します。	ターミナルで AWS 認証情報を設定し、クラスターロールを引き受ける権限があることを確認します。次の例のコードを使用できます。 `aws eks update-kubeconfig --name <Cluster_Name> --region <region>--role-arn <Role_ARN>`	AWS DevOps、DevOps エンジニア、App 開発者

CI/CD パイプラインをデプロイする

タスク	説明	必要なスキル
パラメータをセットアップします。	`config/config.json` ファイルで、以下の必須パラメータを設定します。 • `pipelineName` : AWS CDK によって作成される CI/CD パイプラインの名前 (例:`deploy-nth-to-eks-pipeline`)。AWS CodePipeline は、この名前	アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	のパイプラインを作成します。 • `repositoryName` ：作成する AWS CodeCommit リポジトリ (例:`deploy-nth-to-eks-repo`)。AWS CDK はこのリポジトリを作成し、CI/CD パイプラインのソースとして設定します。 **Note** このソリューションでは、この CodeCommit リポジトリとブランチ (次のブランチパラメータで提供) が作成されます。 • `branch`: リポジトリ内のブランチ名 (例:`main`)。このブランチにコミットすると、CI/CD パイプラインが開始されます。 • `cfn_scan_script` ：NTH (`scan.sh`) の AWS CloudFormation テンプレートをスキャンするために使用されるスクリプトのパス。このスクリプトは、AWS CodeCommit `nth` リポジトリの一部となるフォルダにあります。	

タスク	説明	必要なスキル
	• cfn_deploy_scrip ： NTH (installApp.sh) の AWS CloudFormation テンプレートをデプロイするために使用されるスクリプトのパス。 • stackName ：デプロイする CloudFormation スタックの名前。 • eksClusterName - 既存の EKS クラスターの名前。 • eksClusterRole ：すべての Kubernetes API 呼び出しで EKS クラスターにアクセスするために使用される IAM ロール (例:clusteradmin)。通常、このロールは aws-auth ConfigMap 追加されます。 • create_cluster_role ：eksClusterRole IAM ロールを作成するには、yes と入力します。eksClusterRole パラメータに既存のクラスターロールを指定する場合は、no と入力します。 • create_iam_oidc_provider ：クラスターの IAM OIDC プロバイダーを作成するには、はいを入力します。IAM OIDC プロバイダーが既に存在する場合は、no と入力します。詳細	

タスク	説明	必要なスキル
	について、「クラスターの IAM プロバイダーを作成する」 を参照してください。 • `AsgGroupName` : EKS クラスターの一部である Auto Scaling グループ名をカンマで区切ったリスト (例:`ASG_Group_1,ASG_Group_2`) 。 • `region`: クラスターが配置されている AWS リージョンの名前 (例: `us-east-2`)。 • `install_cdk` : AWS CDK が現在マシンにインストールされていない場合は、yes と入力します。`cdk --version` コマンドを実行して、インストールされている AWS CDK のバージョンが 2.27.0 以降かどうかを確認します。その場合は no と入力します。 yes と入力すると、setup.sh `sudo npm install -g cdk@2.27.0` スクリプトはコマンドを実行して AWS CDK をマシンにインストールします。このスクリプトには sudo 権限が必要なので、プロンプトが表示されたらアカウントパスワードを入力します。	

タスク	説明	必要なスキル
NTH をデプロイする CI/CD パイプラインを作成します。	setup.sh スクリプトを実行します。 `./setup.sh` このスクリプトは、`config/config.json` ファイル内のユーザー入力パラメータに基づいてサンプルコード、パイプライン、および CodeBuild プロジェクトを含む CodeCommit リポジトリを作成する AWS CDK アプリケーションをデプロイします。 このスクリプトは sudo コマンドで npm パッケージをインストールする際にパスワードを要求します。	アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
CI/CD パイプラインを確認します。	AWS マネジメントコンソールを開き、スタックに作成された以下のリソースを確認します。 • nth フォルダの内容を含む CodeCommit リポジトリ • AWS CodeBuild プロジェクト cfn-scan。CloudFormation テンプレートの脆弱性をスキャンします。 • CodeBuild プロジェクト Nth-Deploy 。AWS CloudFormation テンプレートと対応する NTH Helm チャートを AWS CodePipeline パイプラインを通じてデプロイします。 • NTHをデプロイする CodePipeline パイプライン。 パイプラインが正常に実行された後、Helm release aws-node-termination-handler が EKS クラスターにインストールされます。また、aws-node-termination-handler という名前のポッドがクラスターの kube-system 名前空間で実行されています。	アプリ開発者、AWS DevOps、DevOps エンジニア

NTH デプロイをテスト

タスク	説明	必要なスキル
Auto Scaling グループのスケールインイベントをシミュレートします。	自動スケーリングスケールインイベントをシミュレートするには、以下を実行します。 1. AWS コンソールで EC2 コンソールを開き、[Auto Scaling グループ] を選択します。 2. `config/config.json` で指定されているものと同じ名前の Auto Scaling グループを選択し、編集を選択します。 3. 必要容量と最小容量を 1 ずつ減らします。 4. [Update] (更新) を選択します。	
ログを見直します。	スケールインイベント中、NTH ポッドは対応するワーカーノード (スケールインイベントの一部として終了する EC2 インスタンス) をコードオンしてドレインします。ログを確認するには、追加情報セクションのコードを使用してください。	アプリ開発者、AWS DevOps、DevOps エンジニア

クリーンアップ

タスク	説明	必要なスキル
すべての AWS リソースをクリーンアップします。	このパターンで作成されたリソースをクリーンアップするには、以下のコマンドを実行します。 `./uninstall.sh` これにより、CloudFormation スタックが削除され、このパターンで作成されたすべてのリソースがクリーンアップされます。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
npm レジストリが正しく設定されていません。	このソリューションのインストール中、スクリプトは npm install をインストールして、必要なパッケージをすべてダウンロードします。インストール中に「モジュールが見つかりません」というメッセージが表示された場合は、npm レジストリが正しく設定されていない可能性があります。以下のコマンドを実行して、現在のレジストリ設定を確認します。 `npm config get registry` 以下のコマンドを実行して、 `https://registry.npmjs.org/` でレジストリを設定します。

問題	ソリューション
	`npm config set registry https://registry.npmjs.org`
SQS メッセージの配信を遅らせる。	トラブルシューティングの一環として、NTH ポッドへの SQS メッセージ配信を遅延させたい場合は、SQS 配信遅延パラメータを調整できます。詳細については、「Amazon SQS ディレイキュー」を参照してください。

関連リソース

- 「AWS ノード終了ハンドラーのソースコード」
- 「EC2 ワークショップ」
- AWS CodePipeline
- Amazon エラスティックKubernetesサービス (Amazon EKS)
- 「AWS クラウド開発キット」
- AWS CloudFormation

追加情報

1. N 番目のポッド名を検出します。

```
kubectl get pods -n kube-system |grep aws-node-termination-handler
aws-node-termination-handler-65445555-kbqc7   1/1     Running   0          26m
kubectl get pods -n kube-system |grep aws-node-termination-handler
aws-node-termination-handler-65445555-kbqc7   1/1     Running   0          26m
```

2. ログを確認します。次のようなログの例があります。Auto Scaling グループのライフサイクルフック完了シグナルを送信する前に、ノードが封鎖され、排水されたことを示しています。

```
kubectl -n kube-system logs aws-node-termination-handler-65445555-kbqc7
022/07/17 20:20:43 INF Adding new event to the event store
 event={"AutoScalingGroupName":"eksctl-my-cluster-target-nodegroup-
ng-10d99c89-NodeGroup-ZME36IGAP7O1","Description":"ASG Lifecycle Termination
```

```
 event received. Instance will be interrupted at 2022-07-17 20:20:42.702
 +0000 UTC \n","EndTime":"0001-01-01T00:00:00Z","EventID":"asg-lifecycle-
term-33383831316538382d353564362d343332362d613931352d383430666165636334333564","InProgress":fal
east-2.compute.internal","NodeProcessed":false,"Pods":null,"ProviderID":"aws:///us-
east-2c/i-0409f2a9d3085b80e","StartTime":"2022-07-17T20:20:42.702Z","State":""}
2022/07/17 20:20:44 INF Requesting instance drain event-id=asg-lifecycle-
term-33383831316538382d353564362d343332362d613931352d383430666165636334333564
 instance-id=i-0409f2a9d3085b80e kind=SQS_TERMINATE node-name=ip-192-168-75-60.us-
east-2.compute.internal provider-id=aws:///us-east-2c/i-0409f2a9d3085b80e
2022/07/17 20:20:44 INF Pods on node node_name=ip-192-168-75-60.us-
east-2.compute.internal pod_names=["aws-node-qchsw","aws-node-termination-
handler-65445555-kbqc7","kube-proxy-mz5x5"]
2022/07/17 20:20:44 INF Draining the node
2022/07/17 20:20:44 ??? WARNING: ignoring DaemonSet-managed Pods: kube-system/aws-node-
qchsw, kube-system/kube-proxy-mz5x5
2022/07/17 20:20:44 INF Node successfully cordoned and drained
 node_name=ip-192-168-75-60.us-east-2.compute.internal reason="ASG Lifecycle
 Termination event received. Instance will be interrupted at 2022-07-17 20:20:42.702
 +0000 UTC \n"
2022/07/17 20:20:44 INF Completed ASG Lifecycle Hook (NTH-K8S-TERM-HOOK) for instance
 i-0409f2a9d3085b80e
```

CI/CD パイプラインを使用して Amazon EKS へ Java アプリケーションを自動的にビルドし、デプロイする

作成者: MAHESH RAGHUNANDANAN (AWS)、James Radtke (AWS)、Jomcy Pappachen (AWS)

概要

このパターンでは、推奨される DevSecOps プラクティスを使用して Java アプリケーションを自動的に構築し、 上の Amazon Elastic Kubernetes Service (Amazon EKS) クラスターにデプロイする継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを作成する方法について説明します AWS クラウド。このパターンでは、Spring Boot Java フレームワークで開発され、Apache Maven を使用するグリーティングアプリケーションを使用しています。

このパターンのアプローチを使用して Java アプリケーションのコードをビルドし、アプリケーションのアーティファクトを Docker イメージとしてパッケージ化し、イメージをセキュリティスキャンし、そのイメージをワークロードコンテナとして Amazon EKS にアップロードできます。このパターンのアプローチは、緊密に結合されたモノリシックアーキテクチャからマイクロサービスアーキテクチャに移行する場合に便利です。このアプローチは、Java アプリケーションのライフサイクル全体を監視および管理する上でも役立ち、より高いレベルの自動化が可能になり、エラーまたはバグを回避できます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface （ AWS CLI) バージョン 2、インストールおよび設定済み。詳細については、 AWS CLI ドキュメントの「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。

  AWS CLI バージョン 2 は、Amazon EKS クラスターを作成するのと同じ AWS Identity and Access Management (IAM) `aws-auth` ロールで設定する必要があります。これは、そのロールのみが に他の IAM ロールを追加する権限を持っているためです`ConfigMap`。設定の詳細と手順については AWS CLI、 AWS CLI ドキュメントの「設定の構成」を参照してください。
- へのフルアクセスを持つ IAM ロールとアクセス許可 AWS CloudFormation。詳細については、AWS CloudFormation ドキュメントの「IAM によるアクセスの制御」を参照してください。
- EKS クラスター内のワーカーノードの IAM ロール名と IAM ロールの Amazon リソースネーム (ARN) の詳細を含む既存の Amazon EKS クラスター。

- Amazon EKS クラスターにインストールおよび設定済みの Kubernetes クラスターオートスケーラー。詳細については、Amazon EKS ドキュメントの「Karpenter と Cluster Autoscaler を使用してクラスターコンピューティングをスケールする」を参照してください。
- GitHub リポジトリのコードへのアクセス。

> **Important**
>
> AWS Security Hub は、このパターンのコードに含まれる AWS CloudFormation テンプレートの一部として有効になります。デフォルトでは、Security Hub を有効にすると、30 日間の無料トライアルが付属しています。トライアル後、これに関連するコストが発生します AWS のサービス。料金の詳細については、「AWS Security Hub の料金」を参照してください。

製品バージョン

- Helm バージョン 3.4.2 以降
- Apache Maven バージョン 3.6.3 以降
- BridgeCrew Checkov バージョン 2.2 以降
- Aqua Security Trivy バージョン 0.37 以降

アーキテクチャ

テクノロジースタック

- AWS CodeBuild
- AWS CodeCommit

  注意: AWS CodeCommit は新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら。ただし、このソリューションは、最小限の変更で GitHub や GitLab GitLabプロバイダーと連携します。
- Amazon CodeGuru
- AWS CodePipeline
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon EKS

- Amazon EventBridge
- AWS Security Hub
- Amazon Simple Notification Service (Amazon SNS)

ターゲット アーキテクチャ

この図表は、次のワークフローを示しています：

1. 開発者は CodeCommit リポジトリのベースブランチにある Java アプリケーションコードを更新し、プルリクエスト (PR) を作成します。
2. PR が送信されるとすぐに、Amazon CodeGuru Reviewer はコードを自動的にレビューし、Java のベストプラクティスに基づいて分析し、開発者に推奨事項を提示します。
3. PR がベースブランチにマージされると、Amazon EventBridge イベントが作成されます。
4. EventBridge イベントは CodePipeline パイプラインを起動し、開始されます。
5. CodePipeline は CodeSecurity スキャンステージ (継続的セキュリティ) を実行します。
6. AWS CodeBuild は、Dockerfile および Kubernetes デプロイ Helm ファイルが Checkov を使用してスキャンされ、アプリケーションのソースコードが増分コード変更に基づいてスキャンされるセキュリティスキャンプロセスを開始します。アプリケーションのソースコードスキャンは、CodeGuru Reviewer コマンドラインインターフェイス (CLI) ラッパーによって実行されます。
7. セキュリティスキャンステージが成功すると、ビルドステージ (継続的インテグレーション) が開始されます。
8. ビルド段階では、CodeBuild はアーティファクトをビルドし、アーティファクトを Docker イメージにパッケージ化し、Aqua Security Trivy を使用してセキュリティの脆弱性に対するイメージをスキャンし、イメージを Amazon ECR に保存します。
9. ステップ 8 で検出された脆弱性は Security Hub にアップロードされ、開発者またはエンジニアによってさらに分析されます。Security Hub は、脆弱性を修復するための概要と推奨事項を提供します。
10. CodePipeline パイプライン内のシーケンシャルフェーズの E メール通知は、Amazon SNS を介して送信されます。
11. 継続的インテグレーションフェーズが完了すると、CodePipeline は Deploy ステージ (継続的デリバリー) に入ります。

12Docker イメージは、Helm チャートを使用してコンテナワークロード (ポッド) として Amazon EKS にデプロイされます。

13.アプリケーションポッドは、アプリケーションのプロファイリングデータ (CPU、ヒープ使用量、レイテンシー) を CodeGuru Profiler に送信する Amazon CodeGuru Profiler エージェントを使用して設定され、デベロッパーがアプリケーションの動作を理解するのに役立ちます。 CodeGuru

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。

  注意: AWS CodeCommit は新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら
- Amazon CodeGuru Profiler は、ライブアプリケーションからランタイムパフォーマンスデータを収集し、アプリケーションのパフォーマンスを微調整する上で役立つ推奨事項を提供します。
- Amazon CodeGuru Reviewer は、プログラム解析と機械学習により、開発者が検索するのが難しい潜在的な不具合を検出し、Java と Python のコードの改善案を提案します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS) を使用すると、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行できます。
- Amazon EventBridge は、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなど、さまざまなソースからのリアルタイムデータにアプリケーションを接続するのに役立つサーバーレスイベントバスサービスです AWS アカウント。

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。また、セキュリティ業界標準とベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意のデータ量を保存、保護、取得する際に役立つクラウドベースのオブジェクトストレージサービスです。

その他のサービス

- Helm は Kubernetes 用のオープンソースのパッケージマネージャーです。
- Apache Maven は、ソフトウェアプロジェクトを管理する包括ツールです。
- BridgeCrew Checkov は、Infrastructure as Code (IaC)ファイルをスキャンして、セキュリティまたはコンプライアンスの問題につながる可能性のある設定ミスを検出する静的コード分析ツールです。
- Aqua Security Trivy は、設定の問題に加えて、コンテナイメージ、ファイルシステム、Git リポジトリの脆弱性のための包括的スキャナーです。

コード

このパターンのコードは、GitHub 内の「aws-codepipeline-devsecops-amazoneks」リポジトリで利用できます。

ベストプラクティス

- このパターンは、IAM セキュリティのベストプラクティスに従って、ソリューションのすべてのフェーズに IAM エンティティの最小特権の原則を適用します。追加ツール AWS のサービス またはサードパーティーツールを使用してソリューションを拡張する場合は、IAM ドキュメントの最小特権のアクセス許可の適用に関するセクションを確認することをお勧めします。
- 複数の Java アプリケーションがある場合は、アプリケーションごとに個別の CI/CD パイプラインを作成することをお勧めします。
- モノリスアプリケーションがある場合は、可能な限りアプリケーションをマイクロサービスに分割することをお勧めします。マイクロサービスは柔軟性が高く、アプリケーションをコンテナとして簡単にデプロイでき、アプリケーションのビルドとデプロイ全体をよりよく把握できます。

エピック

環境をセットアップする

タスク	説明	必要なスキル
GitHub リポジトリのクローン作成	リポジトリのクローンを作成するには、次の コマンドを実行します。 `git clone https://github.com/aws-samples/aws-codepipeline-devsecops-amazoneks`	アプリ開発者、DevOps エンジニア
S3 バケットを作成し、コードをアップロードします。	1. にサインインし AWS Management Console、Amazon S3 コンソールを開き、このソリューションをデプロイする予定の AWS リージョン に S3 バケットを作成します。詳細については、Amazon EFS ユーザーガイドの「バケットの作成」を参照してください。 2. S3 バケットに codeという名前のフォルダを作成します。 3. リポジトリのクローンを作成した場所にナビゲートします。.zip 拡張子 (`cicdstack.zip`) を使用してコード全体の圧縮バージョンを作成し、.zip ファイルを検証するには、次の	AWS DevOps、クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル

	コマンドを順番に実行します。
	
	`cd aws-codepipeline-devsecops-amazoneks`
	`python -m zipfile -c cicdstack.zip *`
	`python -m zipfile -t cicdstack.zip`
	
	Note
	python コマンドが失敗し、Python が見つからなかったことを示す場合は、python3代わりにを使用します。
	
	4. S3 バケットで以前に作成したコードフォルダに cicdstack.zip ファイルをアップロードします。	

タスク	説明	必要なスキル
AWS CloudFormation スタックを作成します。	1. AWS CloudFormation コンソールを開き、[スタックの作成] を選択します。 2. [Specify Template (テンプレートを指定)] で、[Upload a template file (テンプレートファイルをアップロード)] を選択し、`cf_templates/codecommit_ecr.yaml` ファイルをアップロードしてから、[Next (次へ)] を選択します。 3. Specify stack details (スタック詳細を指定) に、スタック名を入力してから、次の入力パラメータ値を指定します。 • `CodeCommitRepositoryBranchName` : コードが存在するブランチの名前 (デフォルトは `main`) • `CodeCommitRepositoryName` : 作成する CodeCommitrepository の名前 • `CodeCommitRepositoryS3Bucket` : コードフォルダを作成した S3 バケットの名前 • `CodeCommitRepositoryS3BucketObjKey` :	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	code/cicdstack.zip • ECRRepositoryName : 作成する Amazon ECR リポジトリの名前 4. [Next (次へ)] を選択し、スタックの設定オプションのデフォルト設定を使用してから、[Next (次へ)] を選択します。 5. 確認セクションで、テンプレートとスタック詳細を確認してから、[Create stack (スタックを作成)] を選択します。その後、CodeCommit リポジトリと Amazon ECR リポジトリなどのスタックが作成されます。 6. Java CI/CD パイプラインの設定に必要な CodeCommit リポジトリと Amazon ECR リポジトリの名前を書き留めます。	

タスク	説明	必要なスキル
CloudFormation スタックデプロイを検証します。	1. CloudFormation コンソールの スタックで、デプロイした CloudFormation スタックのステータスを確認します。スタックのステータスは作成完了になっているはずです。 2. コンソールから、CloudFormation リポジトリと Amazon ECR リポジトリがプロビジョニングされ、準備が整っていることを確認します。	AWS DevOps、DevOps エンジニア
S3 バケットを削除します。	以前に作成した S3 バケットを空にして削除します。詳細については、Amazon EFS ユーザーガイドの「バケットの削除」を参照してください。	AWS DevOps、DevOps エンジニア

Helm チャートを設定する

タスク	説明	必要なスキル
Java アプリケーションの Helm チャートを設定します。	1. GitHub リポジトリのクローンを作成した場所で、フォルダ `helm_charts/aws-proserve-java-greeting` にナビゲートします。このフォルダの `values.dev.yaml` ファイルには、Amazon EKS へのコンテナデプロイ用	DevOps エンジニア

タスク	説明	必要なスキル

	に変更できる Kubernetes リソース設定に関する情報が含まれています。AWS アカウント ID、および Amazon ECR リポジトリ名を指定して AWS リージョン、Docker リポジトリパラメータを更新します。
	
	`image:`
	`  repository:`
	`  <account-id>.dkr.ecr.<region>.amazonaws.com/<app-ecr-repo-name>`
	
	2. Java ポッドのサービスタイプは `LoadBalancer` に設定されます。
	
	`service:`
	`  type: LoadBalancer`
	`  port: 80`
	`  targetPort: 8080`
	`  path: /hello`
	`  initialDelaySeconds: 60`
	`  periodSeconds: 30`
	
	別のサービス (なNodePort) を使用するには、このパラメータを変更できます。詳細については、Kubernetes ドキュメントを参照してください。
	3. `autoscaling` パラメータを `enabled: true` に変更	

タスク	説明	必要なスキル

	することで、「Kubernetes 水平ポッドオートスケーラー」を有効化できます。
	
	`autoscaling:`
	`  enabled: true`
	`  minReplicas: 1`
	`  maxReplicas: 100`
	`  targetCPUUtilizationPercentage: 80`
	`  # targetMemoryUtilizationPercentage: 80`
	
	4. Kubernetes ワークロードのさまざまな機能を有効にするには、 values.<ENV>.yaml ファイルの値を変更します。ここで、<ENV>は開発、本番稼働、UAT、または QA 環境です。	

タスク	説明	必要なスキル

Helm チャートの構文エラーを検証します。	1. ターミナルから、次のコマンドを実行して、ローカルワークステーションに Helm v3 がインストールされていることを確認します。
	`helm --version`
	Helm v3 がインストールされていない場合は、インストールします。
	2. ターミナルで、Helm チャートディレクトリ (`helm_charts/aws-proserve-java-greeting`) にナビゲートし、次のコマンドを実行します。
	`helm lint . -f values.dev.yaml`
	これで、Helm チャートの構文エラーがチェックされます。
		DevOps エンジニア

Java CI/CD パイプラインを設定する

タスク	説明	必要なスキル
CI/CD パイプラインを作成します。	1. AWS CloudFormation コンソールを開き、[スタックの作成] を選択します。	AWS DevOps

タスク	説明	必要なスキル
	2. テンプレートを指定で、[Upload a template file (テンプレートファイルをアップロード)] を選択し、`cf_templates/build_deployment.yaml` テンプレートをアップロードしてから、[Next (次へ)] を選択します。 3. スタック詳細を指定で、スタック名を指定し、入力パラメータに次の値を指定します。 • `CodeBranchName` : コードが存在する CodeCommit リポジトリのブランチ名 • `EKSClusterName` : EKS クラスターの名前 (`EKSCluster` ID ではない) • `EKSCodeBuildAppName` : アプリ Helm チャートの名前 (`aws-proserve-java-greeting`) • `EKSWorkerNodeRoleARN` : Amazon EKS ワーカーノードに割り当てられた IAM ロールの ARN • `EKSWorkerNodeRoleName` : Amazon EKS ワー	

タスク	説明	必要なスキル
	カーノードに割り当てられた IAM ロールの名前 • `EcrDockerRepository` ：コードの Docker イメージが保存される Amazon ECR リポジトリの名前 • `EmailRecipient` ：ビルド通知を送信する E メールアドレス • `EnvType`: 環境 (開発、テスト、製品など) • `SourceRepoName` ：コードが存在する CodeCommit リポジトリの名前 4. [次へ] を選択します。スタックオプションの設定のデフォルト設定を使用してから、[Next(次へ)] を選択します。 5. レビューセクションで、CloudFormation テンプレートとスタックの詳細を確認し、次へを選択します。 6. [Create stack (スタックの作成)] を選択します。 7. CloudFormation スタックのデプロイ中に、パラメータで指定したメールアドレスの所有者は SNS トピックへのサブスクライブを求めるメッセージを受け取りま	

タスク	説明	必要なスキル
	す。Amazon SNS をサブスクライブするには、所有者はメッセージ内のリンクを選択する必要があります。 8. スタックが作成されたら、スタックの[Outputs (出力)] タブを開いてから、`EksCodeBuildkubeRoleARN` 出力キーの ARN 値を記録します。この IAM ARN 値は、後で CodeBuild IAM ロールに Amazon EKS クラスターにワークロードをデプロイするためのアクセス許可を提供するときに必要です。	

Security Hub と Aqua Security 間の統合をアクティブ化する

タスク	説明	必要なスキル
Aqua Security の統合をオンにします。	このステップは、Trivy が報告した Docker イメージの脆弱性検出結果を Security Hub にアップロードするために必要です。 AWS CloudFormation は Security Hub 統合をサポートしていないため、このプロセスは手動で行う必要があります。 1. AWS Security Hub コンソールを開き、統合に移動します。	AWS 管理者、DevOps エンジニア

タスク	説明	必要なスキル
	2. Aqua Security を検索し、[Aqua Security: Aqua Security] を選択します。 3. [Accept findings (検出結果を受け入れる)] を選択します。	

Helm コマンドまたは kubectl コマンドを実行する CodeBuild を設定する

タスク	説明	必要なスキル
CodeBuild が Amazon EKS クラスターで Helm または kubectl コマンドを実行できるようにします。	CodeBuild を認証して Amazon EKS クラスターで Helm または `kubectl` コマンドを使用するには、IAM ロールを `aws-auth` に追加する必要がありますConfigMap 。この場合、IAM ロールの ARN を追加します。これはEksCodeBuildkubeRoleARN 、CodeBuild サービスが Amazon EKS クラスターにアクセスしてワークロードをデプロイするために作成された IAM ロールです。このアクティビティは 1 回限りです。 **⚠ Important** CodePipeline のデプロイ承認ステージの前に、次の手順を完了する必要があります。	DevOps

タスク	説明	必要なスキル

	1. Amazon Linux または macOS 環境で cf_templates/kube_aws_auth_configmap_patch.sh シェルスクリプトを開きます。
	2. 次のコマンドを実行して Amazon EKS クラスターを認証します。
	`aws eks --region <aws-region> update-kubeconfig --name <eks-cluster-name>`
	3. 次のコマンドを使用してシェルスクリプトを実行し、<rolearn-eks-codebuild-kubectl> を以前記録した EksCodeBuildkubeRoleARN のARN 値に置き換えます。
	`bash cf_templates/kube_aws_auth_configmap_patch.sh <rolearn-eks-codebuild-kubectl>`
	
	aws_auth ConfigMap が設定され、アクセスが付与されます。	

CI/CD パイプラインを検証する

タスク	説明	必要なスキル

CI/CD パイプラインが自動的に開始されることを確認します。	1. Checkov が Dockerfile または Helm チャートで脆弱性を検出した場合、パイプラインの CodeSecurity Scan ステージは通常、失敗します。ただし、この例の目的は、CI/CD パイプライン (通常は DevSecOps プロセス) で修正するのではなく、潜在的なセキュリティの脆弱性を特定するプロセスを確立することです。buildspec/buildspec_secscan.yaml ファイルでは、checkov コマンドは --soft-fail フラグを使用してパイプラインの障害を回避します。
	
	`- echo -e "\n Running Dockerfile Scan"`
	`- checkov -f code/app/Dockerfile --framework dockerfile --soft-fail --summary-position bottom`
	`- echo -e "\n Running Scan of Helm Chart files"`
	`- cp -pv helm_charts/$EKS_CODEBUILD_APP_NAME/values.dev.yaml helm_charts/$EKS_CODEBUILD_`
		DevOps

タスク	説明	必要なスキル

	APP_NAME/values.ya
	ml
	- checkov -d helm_char
	ts/$EKS_CODEBUILD_
	APP_NAME --framewo
	rk helm --soft-fail
	  --summary-position
	  bottom
	- rm -rfv helm_char
	ts/$EKS_CODEBUILD_
	APP_NAME/values.ya
	ml
	
	Dockerfile と Helm チャートの脆弱性が報告されたときにパイプラインが機能しなくなるようにするには、checkov コマンドから `--soft-fail` オプションを削除する必要があります。その後、開発者またはエンジニアは脆弱性を修正し、その変更を CodeCommit ソースコードリポジトリにコミットできます。
	2. CodeSecurity スキャンと同様に、ビルドステージは Aqua Security Trivy を使用して、アプリケーションを Amazon ECR にプッシュする前に HIGHおよびCRITICAL Docker イメージの脆弱性を識別します。
	
	- AWS_REGION=$AWS_DE
	FAULT_REGION	

タスク	説明	必要なスキル
	AWS_ACCOUNT_ID=$AWS_ACCOUNT_ID trivy -d image --no-progress --ignore-unfixed --exit-code 0 --severity HIGH,CRITICAL --format template --template "@securityhub/asff.tpl" -o securityhub/report.asff $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION この例では、Docker イメージの脆弱性が報告されてもパイプラインは失敗しません。これは、 buildspec/buildspec.yml ファイルの trivy コマンドに値を持つ フラグ --exit-code が含まれているためです0。HIGH CRTICAL と脆弱性が報告されたときにパイプラインが失敗するには、 の値を --exit-code に変更します1。その後、開発者またはエンジニアは脆弱性を修正し、その変更を CodeCommit ソースコードリポジトリにコミットできます。	

タスク	説明	必要なスキル
	3. Aqua Security Trivy によって報告された Docker イメージの脆弱性は、Security Hub にアップロードされます。Security Hub コンソールで、結果に移動します。レコードの状態 = アクティブ、製品 = Aqua Security で結果をフィルタリングします。これにより、Security Hub の Docker イメージの脆弱性が一覧表示されます。Security Hub に脆弱性が表示されるまでに 15 分から 1 時間かかる場合があります。 CodePipeline を使用してパイプラインを開始する方法の詳細については、CodePipeline ドキュメントのCodePipeline でパイプラインを開始する」、「手動でパイプラインを開始する」、および「スケジュールに従ってパイプラインを開始するCodePipeline 」を参照してください。	

タスク	説明	必要なスキル
デプロイを承認します。	1. ビルドフェーズが完了すると、デプロイ承認ゲートが表示されます。レビュー担当者またはリリースマネージャーはビルドを検査し、すべての要件を満たしている場合は承認する必要があります。これは、アプリケーションのデプロイに継続的デリバリーを使用するチームに推奨されるアプローチです。 2. 承認後、パイプラインは Deploy ステージを開始します。 3. Deploy ステージが成功すると、このステージの CodeBuild ログにアプリケーションの URL が表示されます。URL を使用してアプリケーションの準備が整っていることを確認します。	DevOps

タスク	説明	必要なスキル
アプリケーションプロファイリングを検証します。	デプロイが完了し、アプリケーションポッドが Amazon EKS にデプロイされると、アプリケーションで設定された Amazon CodeGuru Profiler エージェントは、アプリケーションのプロファイリングデータ (CPU、ヒープサマリー、レイテンシー、ボトルネック) を CodeGuru Profiler に送信しようとします。 アプリケーションの初期デプロイの場合、CodeGuru Profiler はプロファイリングデータを視覚化するのに約 15 分かかります。	AWS DevOps

関連リソース

- AWS CodePipeline ドキュメント
- の Trivy を使用したイメージのスキャン AWS CodePipeline (AWS ブログ記事)
- Amazon CodeGuru Profiler を使用した Java アプリケーションの改善 (AWS ブログ記事)
- AWS Security Finding Format (ASFF) 構文
- Amazon EventBridge のイベントパターン
- 「Helm アップグレード」

追加情報

- AWS CodeCommit は新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら。このソリューションは、GitHub や GitLab などのバージョン管理システム (VCS) Git プロバイダーでも最小限の変更で動作します。

- CodeGuru Profiler は、機能の観点から AWS X-Ray サービスと混同しないでください。CodeGuru Profiler を使用して、ボトルネックやセキュリティ問題を引き起こす可能性のある最も高価なコード行を特定し、潜在的なリスクになる前に修正することをお勧めします。X-Ray サービスは、アプリケーションパフォーマンスのモニタリング用です。
- このパターンでは、イベントルールはデフォルトのイベントバスに関連付けられます。必要に応じて、カスタムイベントバスを使用するようにパターンを拡張できます。
- このパターンでは、CodeGuru Reviewer をアプリケーションコードの静的アプリケーションセキュリティテスト (SAST) ツールとして使用します。このパイプラインは、SonarQube または Checkmarx などの他のツールにも使用できます。これらのツールのいずれかのスキャンセットアップ手順を に追加`buildspec/buildspec_secscan.yaml`して、CodeGuru スキャン手順を置き換えることができます。

Amazon ECS タスク定義を作成し、Amazon EFS を使用して EC2 インスタンスにファイルシステムをマウントする

作成者: Durga Prasad Cheepuri (AWS)

概要

このパターンは、Amazon Elastic Container Service (Amazon ECS) タスク定義を作成しますが、このタスク定義により、Amazon Web Services (AWS) クラウド内のAmazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行され、Amazon Elastic File System (Amazon EFS) を使用してファイルシステムをマウントします。Amazon EFS を使用する Amazon ECS タスクは、タスク定義で指定したファイルシステムを自動的にマウントし、これらのファイルシステムを AWS リージョンのすべてのアベイラビリティーゾーンのタスクのコンテナで使用できるようにします。

永続ストレージと共有ストレージの要件を満たすには、Amazon ECS と Amazon EFS を同時に使用できます。たとえば、Amazon EFS を使用してアプリケーションの永続的なユーザーデータやアプリケーションデータを保存し、可用性を高めるために異なるアベイラビリティーゾーンで実行されるアクティブ/スタンバイ ECS コンテナペアを使用できます。Amazon EFS により、ECS コンテナと分散ジョブワークロードからparallel アクセスできる共有データを保存することもできます。

Amazon ECS で Amazon EFS を使用するには、タスク定義に 1 つ以上のボリューム定義を追加できます。ボリューム定義には、Amazon EFS ファイルシステム ID、アクセスポイント ID、AWS Identity and Access Management (IAM) 認可または転送中の Transport Layer Security (TLS) 暗号化の設定が含まれています。タスク定義内のコンテナ定義により、コンテナの実行時にマウントされるタスク定義ボリュームを指定できます。Amazon EFS ファイルシステムを使用するタスクを実行すると、Amazon ECS はファイルシステムがマウントされ、アクセスが必要なコンテナで使用できるようにします。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 仮想プライベートネットワーク (VPN) エンドポイントまたはルーターを使用する仮想プライベートクラウド (VPC)
- （推奨) Amazon ECS コンテナエージェント 1.38.0 以降は、Amazon EFS アクセスポイントおよび IAM 認可機能と互換性があります (詳細については、AWS ブログ記事「New for Amazon EFS – IAM Authorization and Access Points」を参照してください）。

制約事項

- 1.35.0 より前のバージョンの Amazon ECS コンテナエージェントは、EC2 起動タイプを使用するタスク用の Amazon EFS ファイルシステムをサポートしません。

アーキテクチャ

次の図は、Amazon ECS により、タスク定義を作成し、ECS コンテナ内の EC2 インスタンスに Amazon EFS ファイルシステムをマウントするアプリケーションの例を示しています。

この図表は、次のワークフローを示しています：

1. Amazon EFS ファイルシステムを作成します。
2. コンテナにより、タスク定義を作成します。
3. Amazon EFS ファイルシステムをマウントするようにコンテナインスタンスを設定します。タスク定義はボリューム マウントを参照するため、コンテナ インスタンスは Amazon EFS ファイルシステムを使用できます。ECS タスクは、タスクが作成されたコンテナインスタンスと関係なく、同じ Amazon EFS ファイルシステムにアクセスできます。
4. タスク定義の 3 つのインスタンスにより、Amazon ECS サービスを作成します。

テクノロジースタック

- Amazon EC2
- Amazon ECS
- Amazon EFS

ツール

- 「Amazon EC2」— Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動し、スケールアウトまたはスケールインできます。
- 「Amazon ECS」— Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理に使用される、高度にスケーラブルで高速のコンテナ管理サービスです。AWS Fargate が管理するサーバーレスインフラ上でタスクやサービスを実行できます。また

は、インフラストラクチャをより詳細に制御するために、管理する EC2 インスタンスのクラスターでタスクとサービスを実行できます。

- 「Amazon EFS」— Amazon Elastic File System (Amazon EFS)は、AWS クラウドサービスやオンプレミスのリソースで使用できる、シンプルでスケーラブルな、フルマネージドされた伸縮自在な NFS ファイルシステムを提供します。
- 「AWS CLI」— AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールで、コマンドラインシェルのコマンドで AWS サービスと対話します。最小限の構成で、コマンドプロンプトからブラウザベースの AWS マネジメントコンソールで提供される機能と同等の機能を実装する AWS CLI コマンドを実行できます。

エピック

「Amazon EFS ファイルシステムの作成」

タスク	説明	必要なスキル
AWS マネジメントコンソールを使用して Amazon EFS ファイルシステムを作成します。	1. **Note** 「Amazon EFS ファイルシステムを作成」し、コンテナを含む VPC を選択します。: 別の VPC を使用する場合は、VPC ピアリング接続を設定します。 2. ファイルシステム ID をメモします。	AWS DevOps

Amazon EFS ファイルシステムまたは AWS CLI のいずれかで Amazon ECS タスク定義を作成します。

タスク	説明	必要なスキル
Amazon EFS ファイルシステムでタスク定義を作成します。	「新しい Amazon ECS コンソール」または「従来の Amazon ECS コンソール」を以下の設定で使用して、タスク定義を作成します。 • 新しいコンソールを使用する場合は、アプリケーション環境に Amazon EC2 インスタンスを選択します。クラシックコンソールを使用する場合は、起動タイプとして EC2 を選択します。 • ボリュームを追加します。ボリューム名を入力し、ボリュームタイプに EFS を選択し、先にメモしたファイルシステム ID を選択します。ルートディレクトリには、Amazon ECS コンテナホストでホストする Amazon EFS ファイルシステムパスを選択します。	AWS DevOps
AWS CLI を使用してタスク定義を作成します。	1. タスク定義用の入力パラメータプレースホルダーを含む JSON テンプレートを作成するには、以下のコマンドを実行してください。 `aws ecs register-task-definition`	AWS DevOps

タスク	説明	必要なスキル

	`--generate-cli-skeleton`
	2. JSON テンプレートでタスク定義を作成します。
	`aws ecs register-task-definition --cli-input-json file://<path_to_your_json_file>`
	3. **Note**
	`task_definition_parameters.json` ファイル (添付済み) に基づき、JSON テンプレートに入力パラメータを入力します。: 入力パラメータの詳細については、「タスク定義パラメータ (Amazon ECS ドキュメント) 」および register-task-definition」(AWS CLI コマンドリファレンス) を参照してください。	

関連リソース

- 「Amazon ECSの タスク定義」
- 「Amazon EFS ボリューム」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

コンテナイメージを使用して Lambda 関数をデプロイする

作成者: Ram Kandaswamy (AWS)

概要

AWS Lambda は、コンテナイメージをデプロイモデルとしてサポートします。このパターンは、コンテナイメージを使用して Lambda 関数をデプロイする方法を示しています。

Lambda はサーバーレスでイベント駆動型のコンピューティングサービスであり、サーバーをプロビジョニングしたり管理したりしなくても、実質どのようなタイプのアプリケーションやバックエンドサービスでも実行できます。Lambda 関数のコンテナイメージサポートにより、アプリケーションアーティファクト用に最大 10 GB のストレージを確保できるというメリットと、使い慣れたコンテナイメージ開発ツールを使用できるというメリットがあります。

このパターンの例では、基礎となるプログラミング言語として Python を使用していますが、Java、Node.js、Go などの他の言語も使用できます。ソースには、GitHubGitLab ベースのシステムを検討するか、Amazon Simple Storage Service (Amazon S3) を使用します。

前提条件と制限

前提条件

- Amazon Elastic Container Registry (Amazon ECR) がアクティブ化される
- アプリケーションコード
- ランタイムインターフェイスクライアントと最新バージョンの Python を含む Docker イメージ
- Git の実用的な知識

機能制限

- サポートされる最大モデルサイズは 10 GB です。
- Lambda ベースのコンテナデプロイの最大実行時間は 15 分です。

アーキテクチャ

ターゲットアーキテクチャ

1. Git リポジトリを作成し、アプリケーションコードをリポジトリにコミットします。
2. AWS CodeBuild プロジェクトはコミットの変更によってトリガーされます。
3. CodeBuild プロジェクトは Docker イメージを作成し、ビルドされたイメージを Amazon ECR に発行します。
4. Lambda 関数は、Amazon ECR のイメージを使用して作成します。

自動化とスケール

このパターンは AWS CloudFormation、、 AWS Cloud Development Kit (AWS CDK)または SDK の API オペレーションを使用して自動化できます。Lambda はリクエスト数に基づいて自動的にスケーリングでき、同時実行パラメータを使用して調整できます。詳細については、Lambda のドキュメントを参照してください。

ツール

AWS サービス

- AWS CloudFormation AWS CloudFormationhelpsは、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理します AWS リージョン。このパターンでは AWS CloudFormation Application Composer を使用します。これにより、 AWS CloudFormation テンプレートを視覚的に表示および編集できます。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- GitHub、GitLab、Bitbucket は、ソースコードの変更を追跡するために一般的に使用される Git ベースのソース管理システムの一部です。

ベストプラクティス

- 不要なファイルが読み込まれないように、関数はできるだけ効率的かつ小さくしてください。
- 静的レイヤーは Docker ファイルリストの上位に配置し、頻繁に変更されるレイヤーは下位に配置するようにしてください。これによりキャッシュが向上し、パフォーマンスが向上します。
- イメージ所有者は、イメージの更新とパッチの適用を担当します。その更新頻度を運用プロセスに追加してください。詳細については、AWS Lambda のドキュメントを参照してください。

エピック

CodeBuild プロジェクトを作成する

タスク	説明	必要なスキル
Git リポジトリを作成します。	アプリケーションのソースコード、Dockerfile、`buildspec.yaml` およびファイルを含む Git リポジトリを作成します。	開発者
CodeBuild プロジェクトを作成する。	CodeBuild プロジェクトを使用してカスタム Lambda イメージを作成するには、次の手順を実行します。 1. にサインインし AWS Management Console、https://console.aws.amazon.com/codesuite/codebuild/://www.com」で CodeBuild コンソールを開きます。 2. 新しいプロジェクトを作成します。ソース で、作成した Git リポジトリを選択します。さまざまな種類の Git リポジトリ統合の詳	開発者

タスク	説明	必要なスキル
	細については、「接続の使用」のドキュメントを参照してください。 3. 特権モードが有効になっていることを確認します。Docker イメージをビルドするには、これが必要です。そうしないと、イメージは正常にビルドされません。 4. プロジェクト名と説明の値を指定します。	
Dockerfile を編集する。	Dockerfile は、アプリケーションを開発している最上位のディレクトリに配置する必要があります。Python コードは `src` フォルダにあるはずです。 イメージを作成するときは、Lambda がサポートする公式イメージを使用してください。そうしないと、起動エラーが発生し、パッキング処理がより困難になります。 詳細については、「追加情報」セクションを参照してください。	開発者

タスク	説明	必要なスキル
Amazon ECR でリポジトリを作成します。	Amazon ECR にコンテナリポジトリを作成します。次のコマンド例では、作成されたリポジトリの名前は ですcf-demo。 `aws ecr create-repository --cf-demo` リポジトリは buildspec.yaml ファイルで参照されます。	AWS 管理者、デベロッパー
Amazon ECR にイメージをプッシュします。	CodeBuild を使用してイメージビルドプロセスを実行できます。CodeBuild には Amazon ECR とやりとりしたり S3 を操作したりするためのアクセス権限が必要です。プロセスの一環として、Docker イメージがビルドされ、Amazon ECR レジストリにプッシュされます。テンプレートとコードの詳細については、「追加情報」セクションを参照してください。	開発者

タスク	説明	必要なスキル
イメージがリポジトリにあることを確認する。	イメージがリポジトリにあることを確認するには、Amazon ECR コンソールで [リポジトリ] を選択します。Amazon ECR 設定で脆弱性スキャン機能が有効になっている場合は、イメージがタグ付きで一覧表示され、脆弱性スキャンレポートの結果も表示されます。 詳細については、AWS ドキュメントを参照してください。	開発者

イメージを実行する Lambda 関数を作成する

タスク	説明	必要なスキル
Lambda 関数を作成します。	Lambda コンソールで [関数の作成] を選択し、[コンテナイメージ] を選択します。Amazon ECR リポジトリにあるイメージの関数名と URI を入力し、[関数の作成] を選択します。詳細については、AWS Lambda のドキュメントを参照してください。	アプリ開発者
Lambda 関数をテストします。	関数を呼び出してテストするには、[テスト] を選択します。詳細については、AWS Lambda のドキュメントを参照してください。	アプリ開発者

トラブルシューティング

問題	ソリューション
ビルドが成功しない。	1. CodeBuild プロジェクトで特権モードが有効になっているかどうかを確認します。 2. Docker 関連のコマンドに必要なアクセス許可を備えていることを確認します。コマンドに sudo を追加してみます。 3. CodeBuild に関連付けられている IAM ロールに、Amazon ECR、Amazon S3、および CloudWatch のログを操作するための適切なアクションを含むポリシーがあることを確認します。

関連リソース

- Lambda のベースイメージ
- CodeBuild の Docker サンプル
- 一時的な認証情報を渡す

追加情報

Docker ファイルの編集

次のコードは、Dockerfile で編集するコマンドを示しています。

```
FROM public.ecr.aws/lambda/python:3.xx

# Copy function code
COPY app.py ${LAMBDA_TASK_ROOT}
COPY requirements.txt  ${LAMBDA_TASK_ROOT}

# install dependencies
RUN pip3 install --user -r requirements.txt

# Set the CMD to your handler (could also be done as a parameter override outside of
 the Dockerfile)
```

```
CMD [ "app.lambda_handler" ]
```

FROM コマンドで、Lambda でサポートされている Python バージョンに適切な値を使用します (例: 3.12) 。これは、パブリック Amazon ECR イメージリポジトリで使用できるベースイメージになります。

COPY app.py ${LAMBDA_TASK_ROOT} コマンドは、Lambda 関数が使用するタスクルートディレクトリにコードをコピーします。このコマンドは環境変数を使用するので、実際のパスを心配しなくて済みます。実行する関数は引数として CMD ["app.lambda_handler"] コマンドに渡されます。

COPY requirements.txt コマンドはコードに必要な依存関係をキャプチャします。

RUN pip install --user -r requirements.txt コマンドは、依存関係をローカルユーザーディレクトリにインストールします。

イメージを構築するには、次のコマンドを実行します。

```
docker build -t <image name> .
```

Amazon ECR にイメージを追加

次のコードでは、アカウント番号に aws_account_id を置き換え、別のリージョンを使用している場合は us-east-1 を置き換えてください。buildspec ファイルは CodeBuild ビルド番号を使用して、イメージバージョンをタグ値として一意に識別します。要件に合わせて変更できます。

buildspec のカスタムコード

```
phases:
  install:
    runtime-versions:
       python: 3.xx
  pre_build:
    commands:
      - python3 --version
      - pip3 install --upgrade pip
      - pip3 install --upgrade awscli
      - sudo docker info
  build:
    commands:
      - echo Build started on `date`
```

```
      - echo Building the Docker image...
      - ls
      - cd app
      - docker build -t cf-demo:$CODEBUILD_BUILD_NUMBER .
      - docker container ls
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.us-east-1.amazonaws.com
      - docker tag cf-demo:$CODEBUILD_BUILD_NUMBER aws_account_id.dkr.ecr.us-
east-1.amazonaws.com/cf-demo:$CODEBUILD_BUILD_NUMBER
      - docker push aws_account_id.dkr.ecr.us-east-1.amazonaws.com/cf-demo:
$CODEBUILD_BUILD_NUMBER
```

AWS Fargate を使用して Amazon ECS に Java マイクロサービスをデプロイする

作成者: Vijay Thompson (AWS)、Sandeep Bondugula (AWS)

概要

このパターンは、AWS Fargate を使用して Amazon Elastic Container Service (Amazon ECS) 上にコンテナ化された Java マイクロサービスをデプロイするためのガイダンスを提供します。このパターンでは、コンテナ管理に Amazon Elastic Container Registry (Amazon ECR) は使用せず、代わりに、Docker イメージは Docker ハブからプルされます。

前提条件と制限

前提条件

- Docker ハブにある既存の Java マイクロサービスアプリケーション
- パブリック Docker リポジトリ
- アクティブなAWS アカウント
- Amazon ECS や Fargate などの AWS サービスに精通していること
- Docker、Java、Spring Boot フレームワーク
- Amazon Relational Database Service (Amazon RDS) が実行中 (オプション)
- アプリケーションで Amazon RDS が必要な場合の仮想プライベートクラウド (VPC) (オプション)

アーキテクチャ

ソーステクノロジースタック

- Java マイクロサービス (たとえば、Spring Boot で実装されたもの) と Docker にデプロイされたもの

ソースアーキテクチャ

ターゲットテクノロジースタック

- Fargate を使用して各マイクロサービスをホストする Amazon ECS クラスター
- Amazon ECS クラスターと関連するセキュリティグループをホストする VPC ネットワーク
- Fargate を使用してコンテナを起動する各マイクロサービスのクラスター/タスク定義

ターゲットアーキテクチャ

ツール

ツール

- Amazon ECSでは、独自のコンテナオーケストレーションソフトウェアのインストールと運用、仮想マシンのクラスターの管理とスケーリング、またはそれらの仮想マシン上でコンテナをスケジュールする必要がなくなります。
- AWS Fargate を使用すると、サーバーまたは Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを管理しないでコンテナを実行する上で役立ちます。Amazon Elastic Container Service (Amazon ECS) と合わせて使用されます。
- Docker は、アプリケーションを素早くビルド、テスト、およびデプロイできるソフトウェアプラットフォームです。Docker は、ライブラリ、システムツール、コード、ランタイムなど、ソフトウェアの実行に必要なものがすべて揃ったコンテナと呼ばれる標準化されたユニットにソフトウェアをパッケージ化します。

Docker コード

次の Dockerfile では、使用する Java 開発キット (JDK) のバージョン、Java アーカイブ (JAR) ファイルが存在する場所、公開されるポート番号、およびアプリケーションのエントリポイントを指定します。

```
FROM openjdk:11
ADD target/Spring-docker.jar Spring-docker.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","Spring-docker.jar"]
```

エピック

新しいタスク定義を作成する

タスク	説明	必要なスキル
タスク定義を作成します。	Amazon ECSで Docker コンテナを実行するには、タスク定義が必要です。Amazon ECS コンソール（https://console.aws.amazon.com/ecs/）を開き、[Task definitions（タスク定義）]を選択してから、新しいタスク定義を作成します。詳細については、Amazon S3 のドキュメント参照してください。	AWS システム管理者、アプリ開発者
[Launch type（起動タイプ）]を選択します。	起動タイプとして、[Fargate]を選択します。	AWS システム管理者、アプリ開発者
タスクを設定します。	タスク名を定義し、適切な量のタスクメモリと CPU でアプリケーションを設定します。	AWS システム管理者、アプリ開発者
コンテナを定義します。	コンテナ名を指定します。イメージには、Docker サイト名、リポジトリ名、および Docker イメージのタグ名 (`docker.io/sample-repo/sample-application:sample-tag-name`) を入力します。アプリケーションのメモリ制限を設定し、許可するポートのポートマッピング (`8080, 80`) を設定します。	AWS システム管理者、アプリ開発者

タスク	説明	必要なスキル
タスクを作成します。	タスクとコンテナの設定が完了したら、タスクを作成します。詳細な手順については、関連リソースセクションのリンクを参照してください。	AWS システム管理者、アプリ開発者

クラスターを設定する

タスク	説明	必要なスキル
クラスターを作成および設定します。	クラスタータイプとして [Networking only (ネットワーキングのみ)] を選択し、名前を設定してからクラスターを作成、または可能な場合は既存のクラスターを使用します。詳細については、Amazon ECS ドキュメントを参照してください。	AWS システム管理者、アプリ開発者

タスクの設定

タスク	説明	必要なスキル
タスクを作成します。	クラスター内で、[Run new task (新しいタスクを実行)] を選択します。	AWS システム管理者、アプリ開発者
[Launch type (起動タイプ)] を選択します。	起動タイプとして、[Fargate] を選択します。	AWS システム管理者、アプリ開発者
タスク定義、リビジョン、プラットフォームバージョンを選択します。	実行するタスク、タスク定義のリビジョン、プラット	AWS システム管理者、アプリ開発者

タスク	説明	必要なスキル
	フォームバージョンを選択します。	
クラスターを選択します。	タスクを実行するクラスターを選択します。	AWS システム管理者、アプリ開発者
タスクの数を指定します。	実行するタスクの数を設定します。2 つ以上のタスクで起動する場合、タスク間でトラフィックを分散するロードバランサーが必要です。	AWS システム管理者、アプリ開発者
タスクグループを指定します。	(オプション) 一連の関連するタスクをタスクグループとして特定するタスクグループ名を指定します。	AWS システム管理者、アプリ開発者
クラスター VPC、サブネット、セキュリティグループを設定します。	アプリケーションをデプロイする、クラスター VPC とサブネットを設定します。セキュリティグループ (HTTP、HTTPS、ポート 8080) を作成または更新して、インバウンド接続とアウトバウンド接続にアクセスします。	AWS システム管理者、アプリ開発者
パブリック IP 設定を設定します。	Fargate タスクにパブリック IP アドレスを使用するかどうかに応じて、パブリック IP を有効または無効にします。デフォルトの推奨オプションは、有効です。	AWS システム管理者、アプリ開発者
設定を確認してタスクを作成する	設定を確認してから、[Run Task (タスク実行)] を選択します。	AWS システム管理者、アプリ開発者

カットオーバー

タスク	説明	必要なスキル
アプリケーション URL をコピーします。	タスクステータスが実行中に更新されたら、タスクを選択します。ネットワーキングセクションで、パブリック IP をコピーします。	AWS システム管理者、アプリ開発者
アプリケーションをテストします。	ブラウザにパブリック IP を入力してアプリケーションをテストします。	AWS システム管理者、アプリ開発者

関連リソース

- Amazon ECS 用 Docker の基本(Amazon ECS ドキュメント)
- AWS Fargate 上の Amazon ECS(Amazon ECS ドキュメント)
- タスク定義の作成(Amazon ECS ドキュメント)
- クラスターの作成(Amazon ECS ドキュメント)
- 基本的なサービスパラメータの設定(Amazon ECS ドキュメント)
- ネットワークの設定(Amazon ECS ドキュメント)
- Amazon ECS への Java マイクロサービスのデプロイ(ブログ投稿)

Amazon EKS と Amazon S3 の Helm チャートリポジトリを使用して Kubernetes のリソースとパッケージをデプロイする

作成者: Sagar Panigrahi (AWS)

概要

このパターンは、Kubernetes アプリケーションをその複雑さに関係なく効率的に管理するのに役立ちます。このパターンでは、Helm を既存の継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインに統合して、アプリケーションを Kubernetes クラスターにデプロイします。Helm は Kubernetes アプリケーションの管理に役立つ Kubernetes パッケージマネージャです。Helm チャートは、複雑な Kubernetes アプリケーションの定義、インストール、アップグレードに役立ちます。チャートをバージョン管理して Helm リポジトリに保存できるため、システム停止時の平均復元時間 (MTTR) が短縮されます。

このパターンでは、Kubernetes クラスターに対して Amazon Elastic Kubernetes Service (Amazon EKS) を使用します。Amazon Simple Storage Service (Amazon S3) を Helm チャートリポジトリとして使用しているため、チャートを一元的に管理し、組織全体の開発者がアクセスできます。

前提条件と制限

前提条件

- 仮想プライベートクラウド (VPC) を使用するアクティブな Amazon Web Services (AWS) アカウント
- Amazon EKS クラスター
- Amazon EKS クラスター内にセットアップされ、すぐにワークロードを処理できるワーカーノード
- クライアントマシンのターゲットクラスターの Amazon EKS kubeconfig ファイルを設定するための Kubectl
- バケットにアクセスするための AWS Identity and Access Management (IAM) ロール。
- クライアントマシンから Amazon S3 への IAM (プログラムまたはロール) アクセス
- ソースコードの編成、および CI/CD パイプライン

制限

- 現時点では、カスタムリソース定義 (CRD) のアップグレード、削除、管理はサポートされていません。
- CRD を参照するリソースを使用している場合は、CRD を別に (図の外に) インストールする必要があります。

製品バージョン

- Helm v3.6.3

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EKS
- Amazon VPC
- Amazon S3
- ソースコードの編成
- Helm
- Kubectl

ターゲット アーキテクチャ

自動化とスケール

- AWS CloudFormation を使用すると、インフラストラクチャの作成を自動化できます。詳細については、Amazon EKS ドキュメントのAWS CloudFormation を使用した Amazon EKS リソースの作成」を参照してください。
- Helm を既存の CI/CD 自動化ツールに組み込んで、Helm チャートのパッケージ化とバージョニングを自動化します (このパターンの対象外です)。
- GitVersion または Jenkins のビルド番号を使用すると、チャートのバージョニングを自動化できます。

ツール

ツール

- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンを立ち上げたり保守したりする必要がなく、AWS で Kubernetes を実行するためのマネージドサービスです。Kubernetes は、コンテナ化されたアプリケーションのデプロイ、スケーリング、および管理を自動化するためのオープンソースシステムです。
- Helm – Helm は、Kubernetes クラスター上でアプリケーションをインストールおよび管理するのに役立つ Kubernetes のパッケージマネージャです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- Kubectl – Kubectl は Kubernetes クラスターに対してコマンドを実行するためのコマンドラインユーティリティです。

コード

サンプルコードは添付されています。

エピック

Helm の設定と初期化

タスク	説明	必要なスキル
Helm クライアントをインストールする。	Helm クライアントをローカルシステムにダウンロードしてインストールするには、次のコマンドを実行します。 `sudo curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash`	DevOps エンジニア

タスク	説明	必要なスキル
Helm のインストールを検証する。	Helm が Amazon EKS クラスター内の Kubernetes API サーバーと通信できることを検証するには、`helm version` を実行してください。	DevOps エンジニア

Amazon EKS クラスターに Helm チャートを作成してインストールする

タスク	説明	必要なスキル
NGINX 用の Helm チャートを作成する。	クライアントマシンで `my-nginx` と名前を付けた Helm チャートを作成するには、`helm create my-nginx` を実行します。	DevOps エンジニア
チャートの構造を確認する。	グラフの構造を確認するには、ツリーコマンド `tree my-nginx/` を実行します。	DevOps エンジニア
チャート内のサービスアカウント作成を無効にする。	`values.yaml` の `serviceAccount` セクションの下で、`create` キーを `false` に設定します。このパターンではサービスアカウントを作成する必要がないため、これはオフになっています。	DevOps エンジニア
変更したチャートに構文エラーがないか検証 (lint) する。	ターゲットクラスターにインストールする前に、チャートに構文エラーがないか検証するには、`helm lint my-nginx/` を実行します。	DevOps エンジニア

タスク	説明	必要なスキル
チャートをインストールして Kubernetes リソースをデプロイする。	Helm チャートのインストールを実行するには、次のコマンドを使用します。 `helm install --name my-nginx-release --debug my-nginx/ --namespace helm-space` オプションの debug フラグは、インストール中のすべてのデバッグメッセージを出力します。namespace フラグは、このチャートのリソース部分が作成される名前空間を指定します。	DevOps エンジニア
Amazon EKS クラスターのリソースを確認する。	`helm-space` ネームスペースの Helm チャートの一部として作成されたリソースを確認するには、以下のコマンドを使用します。 `kubectl get all -n helm-space`	DevOps エンジニア

Kubernetes アプリケーションの以前のバージョンにロールバックします。

タスク	説明	必要なスキル
リリースを変更してアップグレードする。	チャートを変更するには、`values.yaml` で `replicaCount` 値を 2 に変更します。次に、次のコマンドを実行して、既にインス	DevOps エンジニア

タスク	説明	必要なスキル
	トールされているリリースをアップグレードします。 `helm upgrade my-nginx-release my-nginx/ --namespace helm-space`	
Helm リリースの履歴を確認する。	Helm を使用してインストールされた特定のリリースのすべてのリビジョンを一覧表示するには、以下のコマンドを実行します。 `helm history my-nginx-release`	DevOps エンジニア
特定のリビジョンの詳細を確認する。	動作中のバージョンに切り替えたり、ロールバックしたりする前や、リビジョンをインストールする前に追加の検証を行う場合は、次のコマンドを使用して各リビジョンに渡された値を確認してください。 `helm get --revision=2 my-nginx-release`	DevOps エンジニア

タスク	説明	必要なスキル
以前のバージョンにロールバックする。	以前のバージョンにロールバックするには、次のコマンドを使用します。 `helm rollback my-nginx-release 1` この例では、リビジョン番号 1 にロールバックしています。	DevOps エンジニア

S3 バケットを Helm リポジトリとして初期化する

タスク	説明	必要なスキル
Helm チャート用の S3 バケットを作成します。	独自の S3 バケットを作成します。バケットに charts という名前のフォルダを作成します。このパターンの例では、ターゲットチャートリポジトリとして `s3://my-helm-charts/charts` を使用しています。	クラウド管理者
Amazon S3 Helm プラグインをインストールします。	helm-s3 プラグインをクライアントマシンにインストールするには、以下のコマンドを使用します。 `helm plugin install https://github.com/hypnoglow/helm-s3.git --version 0.10.0`	DevOps エンジニア

タスク	説明	必要なスキル
	注: Helm V3 のサポートは、プラグインバージョン 0.9.0 以降で利用できます。	
Amazon S3 Helm リポジトリを初期化する。	ターゲットフォルダを Helm リポジトリとして初期化するには、次のコマンドを使用します。 `helm S3 init s3://my-helm-charts/charts` このコマンドは、ターゲット内に `index.yaml` ファイルを作成して、その場所に保存されているすべてのチャート情報を追跡します。	DevOps エンジニア
Amazon S3 リポジトリを Helm に追加する。	クライアントマシンにリポジトリを追加するには、次のコマンドを使用します。 `helm repo add my-helm-charts s3://my-helm-charts/charts` このコマンドは、Helm クライアントマシンのターゲットリポジトリにエイリアスを追加します。	DevOps エンジニア
リポジトリリストを確認する。	Helm クライアントマシン内のリポジトリのリストを表示するには、`helm repo list` を実行します。	DevOps エンジニア

チャートをパッケージ化して Amazon S3 Helm リポジトリに保存する

タスク	説明	必要なスキル
チャートをパッケージ化します。	作成した my-nginx チャートをパッケージ化するには、`helm package ./my-nginx/` を実行します。このコマンドは、my-nginx チャートフォルダのすべての内容をアーカイブファイルにパッケージ化します。アーカイブファイルには、`Chart.yaml` ファイルに記載されているバージョン番号を使用して名前が付けられます。	DevOps エンジニア
パッケージを Amazon S3 Helm リポジトリに保存する。	Amazon S3 Helm リポジトリにパッケージをアップロードするには、正しい .tgz ファイル名を使用して次のコマンドを実行します。 `helm s3 push ./my-nginx-0.1.0.tgz my-helm-charts`	DevOps エンジニア
Helm チャートを検索する。	グラフがローカルと Amazon S3 Helm リポジトリの両方に表示されることを確認するには、次のコマンドを実行します。 `helm search repo my-nginx`	DevOps エンジニア

チャートの修正、バージョン管理、パッケージ化

タスク	説明	必要なスキル
グラフを変更してパッケージ化する。	`values.yaml` では、`replicaCount` の値を 1 に設定します。次に、`helm package ./my-nginx/` を実行してチャートをパッケージ化します。今度はバージョンを `Chart.yaml` から`0.1.1` に変更します。 バージョニングは、CI/CD パイプライン内の GitVersion や Jenkins のビルド番号などのツールを使用して自動化することで更新するのが理想的です。バージョン番号の自動化は、このパターンの範囲外です。	DevOps エンジニア
新しいバージョンを Amazon S3 Helm リポジトリにプッシュする。	バージョン 0.1.1 の新しいパッケージを Amazon S3 の `my-helm-charts` Helm リポジトリにプッシュするには、次のコマンドを実行します。 `helm s3 push ./my-nginx-0.1.1.tgz my-helm-charts`	DevOps エンジニア

Amazon S3 Helm リポジトリからチャートを検索してインストールします。

タスク	説明	必要なスキル
my-nginx チャートのすべてのバージョンを検索する。	使用可能なすべてのバージョンのチャートを表示するには、`--versions` フラグを付けて以下のコマンドを実行します。 `helm search repo my-nginx --versions` フラグがない場合、Helm はデフォルトでアップロードされた最新バージョンのチャートを表示します。	DevOps エンジニア
Amazon S3 Helm リポジトリからチャートをインストールする。	前のタスクの検索結果には、`my-nginx` チャートの複数のバージョンが表示されます。Amazon S3 Helm リポジトリから新しいバージョン (0.1.1) をインストールするには、次のコマンドを使用します。 `helm upgrade my-nginx-release my-helm-charts/my-nginx --version 0.1.1 --namespace helm-space`	DevOps エンジニア

関連リソース

- 「psql ドキュメント」
- helm-s3 plugin (MIT ライセンス)

- HELM クライアントバイナリ
- Amazon EKS ドキュメント

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon EKS にサンプル Java マイクロサービスをデプロイし、Application Load Balancer を使用してマイクロサービスを公開する

作成者: Vijay Thompson (AWS) および Akkamahadevi Hiremath (AWS)

概要

このパターンでは、`eksctl` コマンドラインユーティリティと Amazon Elastic Container Registry (Amazon ECR) を使用して、サンプル Java マイクロサービスをコンテナ化されたアプリケーションとして Amazon Elastic Kubernetes Service (Amazon EKS) にデプロイする方法を説明します。Application Load Balancer を使用して、アプリケーショントラフィックをロードバランスします。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- macOS、Linux、または Windows にインストールおよび設定されている AWS コマンドラインインターフェイス (AWS CLI) バージョン 1.7 以降
- 実行中の Docker デーモン
- macOS、Linux、または Windows にインストールおよび設定されている `eksctl` コマンドラインユーティリティ (詳細については、Amazon EKS ドキュメントの Amazon EKS の使用開始 — eksctl を参照してください。)
- macOS、Linux、または Windows にインストールおよび設定されている `kubectl` コマンドラインユーティリティ (詳細については、Amazon EKS ドキュメントの「kubectl のインストールまたは更新を参照してください。)

制限

- このパターンは、Application Load Balancer の SSL 証明書のインストールには適用されません。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon ECR

- Amazon EKS
- エラスティックロードバランシング

ターゲットアーキテクチャ

次の図は、Amazon EKS で Java マイクロサービスをコンテナ化するアーキテクチャを示しています。

ツール

- Amazon Elastic Container Registry (Amazon ECR) は、安全、スケーラブル、信頼できるマネージド型のコンテナイメージのレジストリサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、AWS で Kubernetes を実行する際に役立ち、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持する必要はありません。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Elastic Load Balancing は、受信トラフィックを複数のアベイラビリティーゾーン(Amazon Elastic Compute Cloud (Amazon EC2)のインスタンス、コンテナ、IP アドレスなど) の複数のターゲットに自動的に配信します。
- eksctl は Amazon EKS にクラスターを作成する上で役立ちます。
- kubectl を使用すると、Kubernetes クラスターにコマンドを実行できるようになります。
- Docker は、コンテナと呼ばれるパッケージにアプリケーションをビルド、テスト、配信する上で役立ちます。

エピック

eksctl を使用して Amazon EKS クラスターを作成する

タスク	説明	必要なスキル
Amazon EKS クラスターを作成します。	2 つの t2.small Amazon EC2 インスタンスをノードとして使用する Amazon EKS クラス	開発者、システム管理者

タスク	説明	必要なスキル
	ターを作成するには、以下のコマンドを実行します。 `eksctl create cluster --name <your-cluster-name> --version <version-number> --nodes=1 --node-type=t2.small` **Note** このプロセスには 15 ～20 分かかる場合があります。クラスターが作成されると、適切な Kubernetes 設定が kubeconfig ファイルに追加されます。kubeconfig ファイルを kubectl と併用して、後の手順でアプリケーションをデプロイできます。	
Amazon EKS クラスターを検証します。	クラスターが作成され、接続できることを確認するには、`kubectl get nodes` コマンドを実行します。	開発者、システム管理者

Amazon ECR リポジトリを作成して Docker イメージをプッシュします。

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	Amazon ECR ドキュメントのプライベートリポジトリの作成の指示に従います。	開発者、システム管理者
POM XML ファイルを作成します。	このパターンの追加情報セクションの POM ファイル例のコードに基づき、pom.xml ファイルを作成します。	開発者、システム管理者

ソースファイルを作成します。	次の例に基づいて、src/main/java/eksExample パスの HelloWorld.java というソースファイルを作成します。
	`package eksExample;`
	`import static spark.Spark.get;`
	`public class HelloWorld {`
	`  public static void main(String[] args) {`
	`  get("/", (req, res) -> {`
	`  return "Hello World!";`
	`  });`
	`  }`
	`}`
	以下のディレクトリ構造を使用してください。
	`### Dockerfile`
	`### deployment.yaml`	

タスク	説明	必要なスキル

	``
	### ingress.yaml
	### pom.xml
	### service.yaml
	### src
	    ### main
	        ### java
	            ###
	 eksExample
	                ###
	 HelloWorld.java
	``	

Dockerfile を作成します。	このパターンの追加情報セクションにある Dockerfile 例のコードに基づき、Dockerfile を作成します。	開発者、システム管理者

タスク	説明	必要なスキル

Docker イメージをビルドおよびプッシュします。	お使いの Dockerfile がイメージをビルド、タグ付け、Amazon ECR にプッシュするディレクトリで、以下のコマンドを実行します。
	
	`aws ecr get-login -password --region <region>| docker login --username <username> --password-stdin <account_number>.dkr.ecr.<region>.amazonaws.com`
	`docker buildx build --platform linux/amd64 -t hello-world-java:v1 .`
	`docker tag hello-world-java:v1 <account_number>.dkr.ecr.<region>.amazonaws.com/<repository_name>:v1`
	`docker push <account_number>.dkr.ecr.<region>.amazonaws.com/<repository_name>:v1`
	
	Note
	上記のコマンドで AWS リージョン、アカウント番号、リポジトリの詳細を変更します。後で使用できるように、画像の URL をメモしておきます。	

タスク	説明	必要なスキル
	Important M1 チップを搭載した macOS システムでは、AMD64 プラットフォームで実行されている Amazon EKS と互換性のあるイメージの構築に問題があります。この問題を解決するには、docker buildx を使用して Amazon EKS で動作する Docker イメージをビルドします。	

Java マイクロサービスをデプロイする

タスク	説明	必要なスキル
デプロイファイルを作成します。	このパターンの追加情報セクションにあるサンプルデプロイファイルのコードに基づき、 `deployment.yaml` という YAML ファイルを作成します。 **Note** 前にコピーしたイメージ URL を Amazon ECR リポジトリのイ	開発者、システム管理者

タスク	説明	必要なスキル
	メージファイルのパスとして使用します。	
Java マイクロサービスを Amazon EKS クラスターにデプロイします。	Amazon EKS クラスターにデプロイを作成するには、`kubectl apply -f deployment.yaml` コマンドを実行します。	開発者、システム管理者
ポッドのステータスを確認します。	1. ポッドのステータスを確認するには、`kubectl get pods` コマンドを実行します。 2. ステータスが準備完了 に変わるまで待ちます。	開発者、システム管理者
サービスを作成します。	1. このパターンの追加情報セクションにあるサンプルサービスファイルのコードに基き、 `service.yaml` というファイルを作成します。 2. `kubectl apply -f service.yaml` コマンドを実行します。	開発者、システム管理者

タスク	説明	必要なスキル
AWS Load Balancer Controller アドオンをインストールします。	Amazon EKS ドキュメントの AWS Load Balancer Controller のアドオンのインストールの指示に従います。 **Note** Kubernetes サービス用の Application Load Balancer または Network Load Balancer を作成するには、 アドオンがインストールされている必要があります。	開発者、システム管理者
Ingress リソースを作成します。	このパターンの追加情報セクションにある サンプル Ingress リソースファイルのコードに基づき、 `ingress.yaml` という YAML ファイルを作成します。	開発者、システム管理者
Application Load Balancer を作成します。	Ingress リソースをデプロイしてApplication Load Balancer を作成するには、`kubectl apply -f ingress.yaml` コマンドを実行します。	開発者、システム管理者

アプリケーションをテストする

タスク	説明	必要なスキル
アプリケーションをテストおよび検証します。	1. ADDRESS フィールドからロードバランサーの DNS 名を取得するには、`kubectl get ingress.networking.k8s.io/java-microservice-ingress` コマンドを実行します。 2. Amazon EKS ノードと同じ VPC 内の EC2 インスタンスで、`curl -v <DNS address from previous command>` コマンドを実行します。	開発者、システム管理者

関連リソース

- プライベートリポジトリの作成(Amazon ECR ドキュメント)
- Docker イメージのプッシュ(Amazon ECR ドキュメント)
- Ingress Controllers(Amazon EKS ワークショップ)
- Docker Buildx(Docker ドキュメント)

追加情報

サンプル POM ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>helloWorld</groupId>
  <artifactId>helloWorld</artifactId>
  <version>1.0-SNAPSHOT</version>


  <dependencies>
    <dependency>
      <groupId>com.sparkjava</groupId><artifactId>spark-core</
artifactId><version>2.0.0</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-jar-plugin</
artifactId><version>2.4</version>
        <configuration><finalName>eksExample</finalName><archive><manifest>
              <addClasspath>true</addClasspath><mainClass>eksExample.HelloWorld</
mainClass><classpathPrefix>dependency-jars/</classpathPrefix>
            </manifest></archive>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-compiler-plugin</
artifactId><version>3.1</version>
        <configuration><source>1.8</source><target>1.8</target></configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-assembly-plugin</
artifactId>
        <executions>
          <execution>
            <goals><goal>attached</goal></goals><phase>package</phase>
            <configuration>
              <finalName>eksExample</finalName>
              <descriptorRefs><descriptorRef>jar-with-dependencies</descriptorRef></
descriptorRefs>
              <archive><manifest><mainClass>eksExample.HelloWorld</mainClass></
manifest></archive>
            </configuration>
          </execution>
        </executions>
```

```
      </plugin>
    </plugins>
  </build>
</project>
```

サンプル Dockerfile

```
FROM bellsoft/liberica-openjdk-alpine-musl:17

RUN apk add maven
WORKDIR /code

# Prepare by downloading dependencies
ADD pom.xml /code/pom.xml
RUN ["mvn", "dependency:resolve"]
RUN ["mvn", "verify"]

# Adding source, compile and package into a fat jar
ADD src /code/src
RUN ["mvn", "package"]

EXPOSE 4567
CMD ["java", "-jar", "target/eksExample-jar-with-dependencies.jar"]
```

サンプルデプロイファイル

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: microservice-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app.kubernetes.io/name: java-microservice
  template:
    metadata:
      labels:
        app.kubernetes.io/name: java-microservice
    spec:
      containers:
      - name: java-microservice-container
        image: .dkr.ecr.amazonaws.com/:
```

```
        ports:
        - containerPort: 4567
```

サンプルサービスファイル

```
apiVersion: v1
kind: Service
metadata:
  name: "service-java-microservice"
spec:
  ports:
    - port: 80
      targetPort: 4567
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: java-microservice
```

サンプル Ingress リソースファイル

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: "java-microservice-ingress"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/load-balancer-name: apg2
    alb.ingress.kubernetes.io/target-type: ip
  labels:
    app: java-microservice
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: "service-java-microservice"
                port:
                  number: 80
```

AWS Copilot を使用してクラスター化されたアプリケーションを Amazon ECS にデプロイする

ジャン・バティスト・ギロワ（AWS）、マシュー・ジョージ（AWS）、トーマス・スコット（AWS）によって作成されました

概要

このパターンは、Amazon Elastic Container Service (Amazon ECS) クラスターにコンテナをデプロイする方法を示しています。これは、Amazon Web Services (AWS) マネジメントコンソールを使用する方法と AWS Copilot を使用する方法で、AWS Copilot がデプロイタスクを簡素化する方法を示しています。

Amazon ECS は、クラスターでコンテナを簡単に実行、停止、管理できる非常にスケーラブルで高速なコンテナ管理サービスです。コンテナは、個々のタスクやサービス内のタスクを実行するために使用するタスク定義で定義されます。タスクとサービスは、AWS Fargate で管理されているサーバーレスインフラストラクチャで実行できます。または、インフラストラクチャをより詳細に制御するために、管理する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターでタスクとサービスを実行できます。

AWS Copilot コマンドラインインターフェイスの継続的インテグレーションと継続的デリバリー (CLI) コマンドは、ローカル開発環境から、Amazon ECS での本番稼働対応のコンテナ化されたアプリケーションの構築、リリース、および運用を簡素化します。AWS Copilot CLI は、Infrastructure as Code 使用することから、ユーザーの代わりにプロビジョニングされた継続的インテグレーションと継続的デリバリー（CI/CD）パイプラインを作成することまで、最新のアプリケーションのベストプラクティスをサポートするデベロッパーワークフローと連携します。AWS Copilot CLI を毎日の開発の一部として使用し、 の代替としてテストのサイクルをAWS マネジメントコンソールの代行として使用します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS アカウントを使用するようにローカルにインストールされ、設定されている AWS Command Line Interface (AWS CLI) (AWS CLI) (AWS CLI ドキュメントのインストール手順 と設定手順 を参照)
- AWS Copilot がローカルにインストールされている (Amazon ECS ドキュメントのインストール手順を参照)

- Docker がローカルマシンにインストールされている (Docker のドキュメント を参照)

制約事項

- Docker は無料プランでは、IP アドレスごとに 6 時間あたり 100 個のコンテナイメージのプル制限を設けています。

アーキテクチャ

ターゲットテクノロジースタック

- 仮想プライベートクラウド (VPC)、パブリックおよびプライベートサブネット、およびセキュリティグループを使用して設定される AWS 環境
- Amazon ECS クラスター
- Amazon ECS サービスとタスク定義
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon DynamoDB
- Application Load Balancer
- AWS Fargate
- Amazon Identity and Access Management (IAM)
- Amazon CloudWatch
- AWS CloudTrail

ターゲット アーキテクチャ

このパターンのサンプルアプリケーションをデプロイすると、複数のタスクが作成され、別々のアベイラビリティーゾーンにデプロイされます。各タスクは Amazon DynamoDB にデータを保存します。あるタスクのウェブページにアクセスすると、他のすべてのタスクのデータを表示できます。

ツール

AWS サービス

- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた マネージドコンテナイメージレジストリサービスです。Amazon ECR は、 IAM を使用するリソースベースの許可を持つプライベートリポジトリをサポートします。
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理を簡単に行うことのできる、高度にスケーラブルで高速なコンテナ管理サービスです。タスクとサービスは、AWS Fargate で管理されているサーバーレスインフラストラクチャで実行できます。または、インフラストラクチャをより詳細に制御するために、管理する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターでタスクとサービスを実行できます。
- AWS Copilot — AWS Copilot には、レジストリへのプッシュ、タスク定義の作成、クラスターの作成など、コンテナ化されたアプリケーションを AWS で起動および管理するのに役立つコマンドラインインターフェイスが用意されています。
- AWS Fargate — AWS Fargate はサーバーレスの従量制料金計算エンジンであり、サーバーを管理しなくてもアプリケーションの構築に集中できます。AWS Fargate は Amazon ECS とAmazon Elastic Kubernetes Service (Amazon EKS) の両方に対応しています。Fargate 起動タイプまたは Fargate 容量プロバイダーを使用して Amazon ECS タスクやサービスを実行する場合、アプリケーションをコンテナにパッケージ化し、CPU とメモリ要件を指定し、ネットワークとIAM ポリシーを定義して、アプリケーションを起動します。各Fargate タスクは、独自の分離境界を持ち、基盤となるカーネル、CPU リソース、メモリリソース、Elastic Network Interface を別のタスクと共有しません。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。
- Elastic Load Balancing (ELB) は、受信したトラフィックを複数のアベイラビリティーゾーンの複数のターゲット (EC2 インスタンス、コンテナ、IP アドレスなど) に自動的に分散させます。登録されているターゲットの状態をモニタリングし、正常なターゲットにのみトラフィックをルーティングします。Elastic Load Balancing は、受信トラフィックの時間的な変化に応じて、ロードバランサーをスケーリングします。また、大半のワークロードに合わせて自動的にスケーリングできます。

ツール

- Docker Command Line Interface
- AWS コマンドラインインターフェイス (AWS CLI)
- AWS Copilotコマンドラインインターフェイス

コード

このパターンで使用されているサンプルアプリケーションのコードは、GitHub のクラスターサンプルアプリケーション リポジトリにあります。次のセクションの指示に従って、サンプルファイルを使用します。

エピック

アプリケーションスタックのデプロイ-オプション 1 (AWS マネジメントコンソール)

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	以下のコマンドを使用してサンプルコードリポジトリを複製します。 `git clone https://github.com/aws-samples/cluster-sample-app cluster-sample-app && cd cluster-sample-app`	アプリ開発者、AWS DevOps
Amazon ECR リポジトリを作成します。	1. AWS マネジメントコンソールにサインインして Amazon ECRコンソール (https://console.aws.amazon.com/redshift/) を開きます。 2. リポジトリの作成を選択します。 3. リポジトリ名には、cluster-sample-app と入力します。 4. 他の設定はすべてデフォルト値のままにしておきます。 5. リポジトリの作成を選択します。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	詳細については、Amazon ECR ユーザーガイドのリポジトリの作成 を参照してください。	

タスク	説明	必要なスキル

Docker イメージをビルドし、タグ付けして Amazon ECR リポジトリにプッシュします。	1. 作成したリポジトリを選択し、プッシュコマンドの表示を選択します。
	2. 表示されるコマンドをコピーしてローカルで実行し、docker イメージのビルド、タグ付け、プッシュを行います。 コマンドの出力は、次のようになります。
	
	レジストリに Docker クライアントを認証します。
	
	`aws ecr get-login -password --region <YOUR_AWS_REGION> | docker login --username AWS --password-stdin <YOUR_AWS_ACCOUNT>.dkr.ecr.<YOUR_AWS_REGION>.amazonaws.com`
	
	Docker イメージの構築：
	
	`docker build -t cluster-sample-app .`
	
	Docker イメージをタグ付けするには：
	
	`docker tag cluster-sample-app:latest <YOUR_AWS_ACCOUNT>.dkr.ecr.<YOUR_AWS_REGION>.amazonaws`
		アプリ開発者、AWS DevOps

タスク	説明	必要なスキル

	`.com/cluster-sample-app:latest`
	
	リポジトリにプッシュするイメージにタグを付けます。
	
	`docker push <YOUR_AWS_ACCOUNT>.dkr.ecr.<YOUR_AWS_REGION>.amazonaws.com/cluster-sample-app:latest`	

タスク	説明	必要なスキル
アプリケーションをデプロイします。	1. AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation/) を開きます。 2. スタックの作成 を選択します。 3. テンプレートの準備 で、テンプレートの準備完了を選択します。 4. テンプレートの指定セクションで、テンプレートファイルのアップロード を選択します。 5. GitHub `cluster-sample-app-stack.yml` リポジトリからクローンしたローカルファイルを CloudFormation テンプレートとして選択し、次へ を選択します。 6. スタックの名前を入力し、次へ を選択します。 7. デフォルトのオプションを保持するには、次へを選択します。 8. すべてのオプションを確認し、IAM リソースが作成されたことを確認して、スタックの作成を選択します。 9. アプリケーションスタックがデプロイされたら、出力タブを選択し、URL をコ	AWS DevOps、アプリケーション開発者

タスク	説明	必要なスキル
	ピーして、ブラウザで開いてアプリケーションにアクセスします。 CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントのAWS CloudFormation ドキュメントでのスタックの作成 を参照してください。	

アプリケーションスタックのデプロイ-オプション 2 (AWS Copilot CLI)

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	以下のコマンドを使用してサンプルコードリポジトリを複製します。 `git clone https://github.com/aws-samples/cluster-sample-app cluster-sample-app && cd cluster-sample-app`	アプリ開発者、AWS DevOps
AWS Copilot CLI を使用してコンテナイメージを AWS にデプロイします。	プロジェクトのルートディレクトリで以下のコマンドを使用して、アプリケーションを 1 ステップでデプロイします。 `copilot init --app cluster-sample-app --`	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	`name demo --type "Load Balanced Web Service" --dockerfile ./Dockerfile --port 8080 --deploy` これで、出力として提供された DNS 名を使用してアプリケーションにアクセスできるはずです。	

作成したリソースを削除します。

タスク	説明	必要なスキル
AWS マネジメントコンソールを使用して作成したリソースを削除します。	オプション 1 (AWS マネジメントコンソール) を使用してアプリケーションスタックをデプロイした場合、作成したリソースを削除する準備ができたら次の手順に従います。 1. クラウドフォーメーション コンソール のhttps://console.aws.amazon.com/cloudformation/ 開きます： 2. 作成したロールを選択し、削除を選択します。 3. Amazon ECR コンソールを https://console.aws.amazon.com/ecr/repositories:// https://https://https://https://https://https://https://https	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	4. 作成したリポジトリを選択し、削除を選択します。	
AWS Copilot によって作成されたリソースを削除します。	オプション 2 (AWS Copilot CLI) を使用してアプリケーションスタックをデプロイした場合、作成したリソースを削除する準備ができたら次の手順に従います。 `copilot app delete`	アプリ開発者、AWS DevOps

関連リソース

- AWS CLI の最新バージョンをインストールまたは更新します (AWS CLI ドキュメント)
- AWS Copilot コマンドラインインターフェイスの使用 (Amazon ECS ドキュメント)
- AWS Fargate 上の Amazon ECS(Amazon ECS ドキュメント)
- Amazon ECS ドキュメント
- Amazon ECR ドキュメント
- Amazon CloudFormation のドキュメント
- Docker デスクトップ (Docker ドキュメント)

gRPC ベースのアプリケーションを Amazon EKS クラスターにデプロイし、Application Load Balancer でアクセスする

作成者: Kirankumar Chandrashekar (AWS) と Huy Nguyen (AWS)

概要

このパターンでは、gRPC ベースのアプリケーションを Amazon Elastic Kubernetes Service (Amazon EKS) クラスターでホストし、Application Load Balancer を介して安全にアクセスする方法を説明します。

「gRPC」は、任意の環境で実行できるオープンソースのリモートプロシージャコール (RPC) フレームワークです。マイクロサービスの統合やクライアントとサーバーの通信に使用できます。gRPC の詳細については、AWS ブログ記事Application Load Balancer support for end-to-end HTTP/2 and gRPC」を参照してください。

このパターンは、Amazon EKS の Kubernetes ポッドで実行される gRPC ベースのアプリケーションをホストする方法を示しています。gRPC クライアントは、SSL/TLS 暗号化接続を使用して HTTP/2 プロトコルを介して Application Load Balancer に接続します。Application Load Balancer は、Amazon EKS ポッドで実行される gRPC アプリケーションにトラフィックを転送します。gRPC ポッドの数は、「Kubernetes 水平ポッドオートスケーラー」 を使用してトラフィックに基づいて自動的にスケーリングできます。Application Load Balancer のターゲットグループは、Amazon EKS ノードでヘルスチェックを実行し、ターゲットが正常かどうかを評価し、トラフィックを正常なノードにのみ転送します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Docker、Linux、macOS、または Windows にインストールして設定します。
- Linux、macOS または Windows にインストールして設定されている「AWS Command Line Interface (AWS CLI) バージョン 2」。
- eksctl、Linux、macOS、または Windows にインストールおよび設定済み。
- `kubectl`、Amazon EKS クラスターのリソースにアクセスするようにインストールおよび設定されています。詳細については、Amazon EKS ドキュメントの「kubectl のインストールまたは更新」を参照してください。

- 「GrpCurl」、インストールおよび設定。
- 新規または既存の Amazon EKS クラスター。詳細については、「Amazon EKS の開始方法」を参照してください。
- Amazon EKS クラスターにアクセスするように設定されたコンピュータターミナル。詳細については、Amazon EKS ドキュメントの「クラスターと通信するようにコンピュータを設定する」を参照してください。
- Amazon EKS クラスターにプロビジョニングされた「AWS Load Balancer Controller」
- 有効な SSL または SSL/TLS 証明書を持つ既存の DNS ホスト名。AWS Certificate Manager (ACM) を使用するか、既存の証明書を ACM にアップロードすることで、ドメインの証明書を取得できます。この 2 つのオプションの詳細については、ACM ドキュメントの「パブリック証明書のリクエスト」と「AWS 認定 Manager への証明書のインポート」を参照してください。

アーキテクチャ

次の図は、このパターンによって実装されるアーキテクチャを示しています。

次の図は、SSL/TLS トラフィックを gRPC クライアントから受信し、Application Load Balancer にオフロードするワークフローを示しています。トラフィックは仮想プライベートクラウド (VPC) から送信されるため、gRPC サーバーにはプレーンテキストで転送されます。

ツール

AWS サービス

- AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 受信したアプリケーションまたはネットワークトラフィックを複数のターゲットに分散するには、Elastic Load Balancing を使用します。例えば、1 つまたは複数のアベイラビリティーゾーンの Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレスにトラフィックを分散できます。
- 「Amazon Elastic Container Registry (Amazon ECR)」は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。

- Amazon Elastic Kubernetes Service (Amazon EKS) は、 で Kubernetes を実行する際に役立ちます。独自の Kubernetes コントロールプレーンまたはノードをインストールおよび維持する必要はありません。

ツール

- eksctl は、Amazon EKS でクラスターを作成するためのシンプルな CLI ツールです。
- 「kubectl」は、 Kubernetes クラスターに対してコマンドを実行するためのコマンドラインユーティリティです。
- AWS Load Balancer Controllerは、Kubernetes クラスターの AWS Elastic Load Balancers の管理を支援します。
- 「GrpCurl」 は gRPC サービスとのやり取りを支援するコマンドラインツールです。

コードリポジトリ

このパターンのコードは、GitHub grpc-traffic-on-alb-to-eks リポジトリで入手できます。

エピック

gRPC サーバーの Docker イメージをビルドして Amazon ECR にプッシュします

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	AWS マネジメントコンソールにサインインし、Amazon ECR コンソールを開き、Amazon ECR リポジトリを作成します。詳細については、Amazon ECR ドキュメントの「リポジトリの作成」を参照してください。Amazon ECR リポジトリの URL を必ず記録してください。 次のコマンドを実行して、AWS CLI で Amazon	クラウド管理者

タスク	説明	必要なスキル
	ECR リポジトリを作成することもできます。 `aws ecr create-repository --repository-name helloworld-grpc`	

タスク	説明	必要なスキル

Docker イメージを作成します。	1. GitHub grpc-traffic-on-alb-to-eks リポジトリのクローンを作成します。
	`git clone https://github.com/aws-samples/grpc-traffic-on-alb-to-eks.git`
	2. リポジトリのルートディレクトリから、Dockerfile が存在することを確認し、次のコマンドを実行して Docker イメージを構築します。
	`docker build -t <amazon_ecr_repository_url>:<Tag> .`
	Important
	を、前に作成した Amazon ECR リポジトリの URL `<amazon_ecr_repository_url>` に置き換えてください。
		DevOps エンジニア

タスク	説明	必要なスキル

Amazon ECR にDocker イメージをプッシュします。	1. 次のコマンドを実行して、Amazon ECR リポジトリにログインします。
	`aws ecr get-login -password --region us-east-1 --no-cli-auto-prompt | docker login --username AWS --password-stdin <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com`
	2. 次のコマンドを実行して、Docker イメージを Amazon ECR リポジトリにプッシュします。
	`docker push <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-grpc:1.0`
	Important
	を AWS アカウント ID <your_aws_account_id> に置き換えてください。
		DevOps エンジニア

Kubernetes マニフェストを Amazon EKS クラスターにデプロイします。

タスク	説明	必要なスキル
Kubernetes マニフェストファイルの値を変更します。	1. 要件に従って`grpc-sample.yaml` 、リポジトリの Kubernetes フォルダの Kubernetes マニフェストファイルを変更します。Ingress リソースのアノテーションとホスト名を変更する必要があります。Ingress リソースのサンプルについては、追加情報セクションを参照してください。イングレスアノテーションの詳細について、Kubernetes ドキュメントの「イングレスアノテーション」を参照してください。 2. Kubernetes デプロイリソースで、`image` デプロイリソースを Docker イメージをプッシュした Amazon ECR リポジトリのユニフォームリソース識別子 (URI) に変更します。デプロイリソースについては、追加情報セクションを参照してください。	DevOps エンジニア
Kubernetes マニフェストファイルをデプロイします。	次の`kubectl`コマンドを実行して、`grpc-sample.yaml` ファイルを Amazon EKS クラスターにデプロイします。	DevOps エンジニア

タスク	説明	必要なスキル
	`kubectl apply -f ./kubernetes/grpc-sample.yaml`	

Application Load Balancerの FQDN の DNS レコードを作成します。

タスク	説明	必要なスキル
Application Load Balancerの FQDN を記録します。	1. 次の `kubectl` コマンドを実行して、Application Load Balancer を管理する Kubernetes 入力リソースを記述します。 `kubectl get ingress -n grpcserver` サンプル出力については、「追加情報」セクションを参照してください。`HOSTS` 出力のフィールドには、SSL 証明書が作成された DNS ホスト名が表示されます。 2. 出力の `Address`フィールドから Application Load Balancer の完全修飾ドメイン名 (FQDN) を記録します。 3. Application Load Balancer の FQDN を指す DNS レコードを作成します。DNS プロバイダーが Amazon Route 53 の場合は、アプ	DevOps エンジニア

タスク	説明	必要なスキル
	リケーションロードバランサーの FQDN をポイントするエイリアスレコードを作成できます。このオプションの詳細については、Route 53 ドキュメントの「エイリアスレコードと非エイリアスレコードの選択」を参照してください。	

ソリューションをテストする

タスク	説明	必要なスキル
gRPC サーバーをテストします。	GrpCurl を使用して次のコマンドを実行して、エンドポイントをテストします。 `grpcurl grpc.example.com:443 list grpc.reflection.v1alpha.ServerReflection helloworld.helloworld` **Note** を DNS 名`grpc.example.com` に置き換えます。	DevOps エンジニア
gRPC クライアントを使用して gRPC サーバーをテストします。	サンプル gRPC `helloworld_client_ssl.py` クライアントで、 のホスト名を `grpc.example.com` gRPC	DevOps エンジニア

タスク	説明	必要なスキル

	サーバーに使用されるホスト名に置き換えます。
	
	次のコードサンプルは、クライアントのリクエストに対する gRPC サーバーからの応答を示しています。
	
	`python ./app/helloworld_client_ssl.py`
	`message: "Hello to gRPC server from Client"`
	
	`message: "Thanks for talking to gRPC server!! Welcome to hello world. Received message is \"Hello to gRPC server from Client\""`
	`received: true`
	
	これは、クライアントがサーバーと通信でき、接続が成功したことを示しています。	

クリーンアップ

タスク	説明	必要なスキル
DNS レコードを削除します。	前に作成した Application Load Balancer の FQDN を指す DNS レコードを削除します。	クラウド管理者
ロードバランサーを削除します。	Amazon EC2 コンソールで、ロードバランサーを選択し、Kubernetes コントロー	クラウド管理者

タスク	説明	必要なスキル
	ラーがイングレスリソース用に作成したロードバランサーを削除します。	
Amazon EKS クラスターを削除します。	を使用して Amazon EKS クラスターを削除しますeksctl。 `eksctl delete cluster -f ./eks.yaml`	AWS DevOps

関連リソース

- 「Amazon EKS でのネットワーク負荷分散」
- 「Application Load Balancer のターゲットグループ」

追加情報

サンプル入力リソース:

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP
    alb.ingress.kubernetes.io/ssl-redirect: "443"
    alb.ingress.kubernetes.io/backend-protocol-version: "GRPC"
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]'
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:<AWS-
Region>:<AccountId>:certificate/<certificate_ID>
  labels:
    app: grpcserver
    environment: dev
  name: grpcserver
  namespace: grpcserver
```

```
spec:
  ingressClassName: alb
  rules:
  - host: grpc.example.com # <----- replace this as per your host name for which the
 SSL certtficate is available in ACM
    http:
      paths:
      - backend:
          service:
            name: grpcserver
            port:
              number: 9000
        path: /
        pathType: Prefix
```

サンプルデプロイリソース:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grpcserver
  namespace: grpcserver
spec:
  selector:
    matchLabels:
      app: grpcserver
  replicas: 1
  template:
    metadata:
      labels:
        app: grpcserver
    spec:
      containers:
      - name: grpc-demo
        image: <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-
grpc:1.0   #<------- Change to the URI that the Docker image is pushed to
        imagePullPolicy: Always
        ports:
        - name: grpc-api
          containerPort: 9000
        env:
        - name: POD_IP
          valueFrom:
```

```
            fieldRef:
              fieldPath: status.podIP
    restartPolicy: Always
```

サンプル出力:

```
NAME              CLASS           HOSTS                           Address
 PORTS          AGE
 grpcserver       <none>      <DNS-HostName>                  <ELB-address>
 80             27d
```

Amazon EKS クラスターをデプロイおよびデバッグ

作成者：Svenja Raether (AWS) と Mathew George (AWS)

概要

コンテナはクラウドネイティブアプリケーション開発に欠かせないものになりつつあります。Kubernetes は、コンテナを効率的に管理およびオーケストレーションする方法を提供します。Amazon Elastic Kubernetes Service (Amazon EKS) は、Amazon Web Services (AWS) で、Kubernetes クラスターを構築、保護、運用、保守するための、完全マネージドの、認定の Kubernetes 準拠サービスです。AWS Fargate でポッドを実行できるため、適切なサイズのコンピューティングキャパシティをオンデマンドで提供できます。

開発者や管理者にとって、コンテナ化されたワークロードを実行する際のデバッグオプションを知ることは重要です。このパターンでは、AWS Fargate を使用して Amazon EKS にコンテナをデプロイし、デバッグする手順を順を追って説明します。これには、Amazon EKS ワークロードの作成、デプロイ、アクセス、デバッグ、クリーンアップが含まれます。

前提条件と制限

前提条件

- アクティブなAWSアカウント
- AWS Identity and Access Management (IAM) ロールを作成および操作するのに十分なアクセス許可、設定された IAM ロール、サービスにリンクされたロール
- ローカルマシンにインストールされたAWS コマンドラインインターフェイス (AWS CLI)
- eksctl
- kubectl
- Helm

機能制限

- このパターンでは、開発環境に役立つデバッグ方法を開発者に提供します。実稼働環境のベストプラクティスが示されていません。
- Windows を実行している場合は、オペレーティングシステム固有のコマンドを使用して環境変数を設定します。

使用している製品バージョン

- AWS CLI バージョン 2
- 使用している Amazon EKS コントロールプレーンのマイナーバージョンの 1 つの違い以内のkubectl バージョン
- eksctl 最新バージョン
- ヘルム v3

アーキテクチャ

テクノロジースタック

- Application Load Balancer
- Amazon EKS
- AWS Fargate

ターゲット アーキテクチャ

図に示されているすべてのリソースは、ローカルマシンから発行された `eksctl` と `kubectl` コマンドを使用してプロビジョニングされます。プライベートクラスターは、プライベート VPC 内のインスタンスから実行する必要があります。

ターゲットアーキテクチャでは、Fargate 起動タイプを使用する EKS クラスターで構成されています。これにより、サーバータイプを指定しなくても、適切なサイズのコンピューティング能力をオンデマンドで提供できます。EKS クラスターには、クラスターノードとワークロードを管理するためのコントロールプレーンがあります。ポッドは、複数のアベイラビリティーゾーンにまたがるプライベート VPC サブネットにプロビジョニングされます。Amazon ECR パブリックギャラリーを参照して NGINX ウェブサーバーのイメージを取得し、クラスターのポッドにデプロイします。

この図表では、 `kubectl` コマンドを使用して Amazon EKS コントロールプレーンにアクセスする方法と、Application Load Balancer を使用してアプリケーションにアクセスする方法を示しています。

.

1. AWS クラウド外部のローカルマシンは、Amazon EKS が管理する VPC 内の Kubernetes コントロールプレーンにコマンドを送信します。
2. Amazon EKS は Fargate プロファイルのセレクターに基づいてポッドをスケジュールします。
3. ローカルマシンは、ブラウザでApplication Load Balancer の URL を開きます。
4. Application Load Balancer は、複数のアベイラビリティーゾーンにまたがるプライベートサブネットにデプロイされた Fargate クラスターノードの Kubernetes ポッド間のトラフィックを分割します。

ツール

AWS サービス

- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS)は、 で Kubernetes を実行する際に役立ちます。独自の Kubernetes コントロールプレーンまたはノードをインストールおよび維持する必要はありません。このパターンでは、eksctl コマンドラインツールを使用して Amazon EKS の Kubernetes クラスターと連携することもできます。
- AWS Fargate を使用すると、サーバーや Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを管理する必要がないコンテナを実行できます。Amazon Elastic Container Service (Amazon ECS) と組み合わせて使用されます。
- Elastic Load Balancing (ELB) は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。例えば、1 つ以上のアベイラビリティゾーンにある Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレス間でトラフィックを分散できます。このパターンでは、Kubernetes ingress 入力がプロビジョニングされるときに、AWS Load Balancer Controller 制御コンポーネントを使用してApplication Load Balancer を作成します。Application Load Balancer は、受信トラフィックを複数のターゲットに分散します。

その他のツール

- Helm は Kubernetes のオープンソースのパッケージマネージャーです。このパターンでは、Helm を使用して AWS Load Balancer コントローラーをインストールします。
- Kubernetes は、コンテナ化されたアプリケーションのデプロイ、スケーリング、および管理を自動化するためのオープンソースシステムです。

- NGINX は高性能なウェブおよびリバースプロキシサーバーです。

エピック

EKS クラスターを作成します。

タスク	説明	必要なスキル
ファイルを作成します。	追加情報 セクションのコードを使用して、次のファイルを作成します。 • `clusterconfig-fargate.yaml` • `nginx-deployment.yaml` • `nginx-service.yaml` • `nginx-ingress.yaml` • `index.html`	アプリ開発者、AWS 管理者、AWS DevOps
環境変数を設定する。	**Note** 以前の未完了のタスクが原因でコマンドが失敗した場合、数秒待ってからコマンドを再度実行します。 このパターンでは、`clusterconfig-fargate.yaml` ファイルに定義されている AWS リージョンとクラスター名を使用します。環境変数と同じ値を設定し	アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	て、以降のコマンドで参照できるようにします。
	
	`export AWS_REGION="us-east-1"`
	`export CLUSTER_NAME="my-fargate"`	

EKS クラスターを作成します。	`clusterconfig-fargate.yaml` ファイルの仕様を使用する EKS クラスターを作成するには、次のコマンドを実行します。 `eksctl create cluster -f clusterconfig-fargate.yaml` このファイルには、`us-east-1` リージョンで `my-fargate-cluster` という名前の新しい EKS クラスターと、1 つのデフォルトの Fargate プロファイル (`fp-default`) をプロビジョニングする `ClusterConfig` が含まれています。 デフォルトの Fargate プロファイルは 2 つのセレクター(`default` と `kube-system`) で設定されます。	アプリ開発者、AWS DevOps、AWS 管理者

タスク	説明	必要なスキル

作成したクラスターを確認します。	続いて、次のコマンドを使用してクラスターを再起動します。
	`eksctl get cluster --output yaml`
	出力は、以下を返します。
	`- Name: my-fargate`
	`  Owned: "True"`
	`  Region: us-east-1`
	CLUSTER_NAME を使用して、作成した Fargate プロファイルを確認します。
	`eksctl get fargatepr ofile --cluster $CLUSTER_NAME --output yaml`
	このコマンドは、リソースに関する情報を表示します。この情報を使用して、作成したクラスターを検証します。出力は、以下を返します。
	`- name: fp-default`
	`  podExecutionRoleAR N: arn:aws:iam::<YOUR -ACCOUNT-ID>:role/ eksctl-my-fargate- cluster-FargatePod ExecutionRole-xxx`
	`  selectors:`
	`  - namespace: default`
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	<pre>- namespace: kube-system
	status: ACTIVE
	subnets:
	- subnet-aaa
	- subnet-bbb
	- subnet-ccc</pre>	

コンテナをデプロイ

タスク	説明	必要なスキル
NGINX ウェブサーバーを起動します。	NGINX ウェブサーバーのデプロイをクラスターに適用するには、次のコマンドを実行します。 <pre>kubectl apply -f ./nginx-deployment.yaml</pre>出力は、以下を返します。 <pre>deployment.apps/nginx-deployment created</pre>デプロイには、Amazon ECR 公開ギャラリーから取得した NGINX イメージの 3 つのレプリカが含まれています。イメージはデフォルトの名前空間にデプロイされ、実行中のポッドのポート 80 に公開されます。	アプリ開発者、AWS DevOps、AWS システム管理者
デプロイとポッドを確認します。	(オプション) デプロイを確認します。クラスターのステー	アプリ開発者、AWS DevOps、AWS 管理者

タスク	説明	必要なスキル

	タスのクエリを実行するには、次のコマンドを使用します。
	`kubectl get deployment`
	出力は、以下を返します。
	`NAME READY UP-TO-DATE AVAILABLE AGE nginx-deployment 3/3 3 3 7m14s`
	ポッドは Kubernetes 内のデプロイ可能なオブジェクトで、1 つ以上のコンテナが含まれています。すべてのポッドを一覧表示するには、以下のコマンドを使用します。
	`kubectl get pods`
	出力は、以下を返します。
	`NAME READY STATUS RESTARTS AGE nginx-deployment-xxxx-aaa 1/1 Running 0 94s nginx-deployment-xxxx-bbb 1/1 Running 0 94s`	

タスク	説明	必要なスキル
	`nginx-deployment-xxxx-ccc 1/1 Running 0 94s`	

ローカルデプロイ。	`deployment.yaml` で指定された 3 つのレプリカから 4 つのレプリカにデプロイをスケールするには、以下のコマンドを使用します。
	`kubectl scale deployment nginx-deployment --replicas 4`
	出力は、以下を返します。
	`deployment.apps/nginx-deployment scaled`
		アプリ開発者、AWS DevOps、AWS システム管理者

AWS Load Balancer Controller

タスク	説明	必要なスキル
環境変数を設定する。	クラスターの CloudFormation スタックを記述して、その VPC に関する情報を取得します。 `aws cloudformation describe-stacks --stack-name eksctl-$CLUSTER_NAME-cluster --query "Stacks[0].Outputs[?OutputKey==\`VPC\`].OutputValue"`	アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	出力は、以下を返します。
	
	`[ "vpc-<YOUR-VPC-ID>" ]`
	
	VPC ID をコピーし、環境変数としてエクスポートします。
	
	`export VPC_ID="vpc-<YOUR-VPC-ID>"`	

クラスターサービスアカウントの IAMを設定する	前のエピックから AWS_REGION と CLUSTER_NAME を使用して、クラスター用の IAM Open ID Connect プロバイダーを作成します。 `eksctl utils associate-iam-oidc-provider \ --region $AWS_REGION \ --cluster $CLUSTER_NAME \ --approve`	アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

IAM ポリシーをダウンロードして作成します。	ユーザーに代わって AWS API を呼び出すことを許可する、AWS Load Balancer Controller 用の IAM ポリシーをダウンロードします。
	`curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam_policy.json`
	AWS CLI を使用して、AWS アカウントのポリシーを作成します。
	`aws iam create-policy \ --policy-name AWSLoadBalancerControllerIAMPolicy \ --policy-document file://iam-policy.json`
	次のような出力が表示されます。
	`{ "Policy": { "PolicyName": "AWSLoadBalancerControllerIAMPolicy", "PolicyId": "<YOUR_POLICY_ID>",`
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
	"Arn": "arn:aws:iam::<YOUR-ACCOUNT-ID>:policy/AWSLoadBalancerControllerIAMPolicy", "Path": "/", "DefaultVersionId": "v1", "AttachmentCount": 0, "PermissionsBoundaryUsageCount": 0, "IsAttachable": true, "CreateDate": "<YOUR-DATE>", "UpdateDate": "<YOUR-DATE>" } } ポリシーの Amazon リソースネーム (ARN)。 export POLICY_ARN=”arn:aws:iam::<YOUR-ACCOUNT-ID>:policy/AWSLoadBalancerControllerIAMPolicy”	

タスク	説明	必要なスキル

IAM サービスアカウントを作成します	aws-load-balancer-controller 名前空間の kube-system という名前のIAM サービスアカウントを作成する。以前に設定した CLUSTER_NAME 、 AWS_REGION 、及び POLICY_ARN を使用します。
	`eksctl create iamserviceaccount \ --cluster=$CLUSTER_NAME \ --region=$AWS_REGION \ --attach-policy-arn=$POLICY_ARN \ --namespace=kube-system \ --name=aws-load-balancer-controller \ --override-existing-serviceaccounts \ --approve`
	作成を検証します。
	`eksctl get iamserviceaccount \ --cluster $CLUSTER_NAME \ --name aws-load-balancer-controller \ --namespace kube-system \ --output yaml`
	出力は、以下を返します。
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	<pre>- metadata:
	    name: aws-load-
	balancer-controller
	    namespace: kube-
	system
	  status:
	    roleARN: arn:aws:i
	am::<YOUR-ACCOUNT-
	ID>:role/eksctl-my
	-fargate-addon-iam
	serviceaccount-ku-
	Role1-<YOUR-ROLE-ID>
	  wellKnownPolicies:
	    autoScaler: false
	    awsLoadBalancerCon
	troller: false
	    certManager: false
	    ebsCSIController:
	 false
	    efsCSIController:
	 false
	    externalDNS: false
	    imageBuilder: false</pre>	

タスク	説明	必要なスキル

AWS Load Balancer コントローラをインストールします。	Helm リポジトリを追加します。
	`helm repo update`
	Amazon EKS チャートリポジトリを Helm リポジトリに追加します。
	`helm repo add eks https://aws.github.io/eks-charts`
	AWS Load Balancer コントローラー eks-chart で使用されている Kubernetes カスタムリソース定義 (CRD) をバックグラウンドで適用します。
	`kubectl apply -k "github.com/aws/eks-charts/stable/aws-load-balancer-controller//crds?ref=master"`
	出力は、以下を返します。
	`customresourcedefinition.apiextensions.k8s.io/ingressclassparams.elbv2.k8s.aws created customresourcedefinition.apiextensions.k8s.io/targetgro`
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	upbindings.elbv2.k8s.aws created
	
	以前に設定した環境変数を使用して Helm チャートをインストールします。
	
	`helm install aws-load-balancer-controller eks/aws-load-balancer-controller \ --set clusterName=$CLUSTER_NAME \ --set serviceAccount.create=false \ --set region=$AWS_REGION \ --set vpcId=$VPC_ID \ --set serviceAccount.name=aws-load-balancer-controller \ -n kube-system`
	
	出力は、以下を返します。
	
	`NAME: aws-load-balancer-controller LAST DEPLOYED: <YOUR-DATE> NAMESPACE: kube-system STATUS: deployed REVISION: 1 TEST SUITE: None NOTES: AWS Load Balancer controller installed!`	

タスク	説明	必要なスキル

サービスを作成します。	`nginx-service.yaml` ファイルを使用して NGINX ポッドを公開するサービスを作成します。
	`kubectl apply -f nginx-service.yaml`
	出力は、以下を返します。
	`service/nginx-service created`
		アプリ開発者、AWS DevOps、AWS システム管理者

Kubernetes イングレスリソースを作成します。	`nginx-ingress.yaml` ファイルを使用して Kubernetes NGINX イングレスを公開するサービスを作成します。
	`kubectl apply -f nginx-ingress.yaml`
	出力は、以下を返します。
	`ingress.networking .k8s.io/nginx-ingress created`
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

ロードバランサー URL を取得します。	入力情報を取得するには、次のコマンドを使用します。
	`kubectl get ingress nginx-ingress`
	出力は、以下を返します。
	`NAME CLASS HOSTS ADDRESS PORTS AGE`
	`nginx-ingress <none> * k8s-default-nginxing-xxx.us-east-1.elb.amazonaws.com 80 80s`
	出力から ADDRESS (たとえば `k8s-default-nginxing-xxx.us-east-1.elb.amazonaws.com`) をコピーし、ブラウザに貼り付けて、`index.html` ファイルにアクセスします。
		アプリ開発者、AWS DevOps、AWS システム管理者

実行中のコンテナをデバッグ

タスク	説明	必要なスキル
ポッドを選択します。	すべてのポッドを一覧表示し、希望のポッドの名前をコピーします。	アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	`kubectl get pods`
	
	出力は、以下を返します。
	
	`NAME READY STATUS RESTARTS AGE nginx-deployment-xxxx-aaa 1/1 Running 0 55m nginx-deployment-xxxx-bbb 1/1 Running 0 55m nginx-deployment-xxxx-ccc 1/1 Running 0 55m nginx-deployment-xxxx-ddd 1/1 Running 0 42m`
	
	このコマンドは、既存のポッドと追加情報を一覧表示します。
	
	特定のポッドに関心がある場合、関心のあるポッドの名前を POD_NAME 変数に入力するか、環境変数として設定します。それ以外の場合、このパラメーターを省略して、すべてのリソースを検索します。	

タスク	説明	必要なスキル
	`export POD_NAME="nginx-deployment-<YOUR-POD-NAME>"`	
アクセスログの使用	デバッグするポッドからログを取得します。 `kubectl logs $POD_NAME`	アプリケーション開発者、AWS システム管理者、AWS DevOps

NGINX ポートを転送します。	ポート転送を使用して NGINX Web サーバにアクセスするためのポッドのポートをローカルマシンのポートにマップします。
	`kubectl port-forward deployment/nginx-deployment 8080:80`
	ブラウザで、次の URL を開きます。
	`http://localhost:8080`
	`port-forward` コマンドは、ロードバランサー経由で公開せずに `index.html` ファイルへのアクセスを提供します。これは、実行中のアプリケーションをデバッグ中にアクセスするのに役に立ちます。ポート転送を停止するには、キーボードコマンド Ctrl+C を押します。
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
ポッド内でコマンドを実行します。	現在の index.html ファイルを確認するには、次のコマンドを使用します。 `kubectl exec $POD_NAME -- cat /usr/share/nginx/html/index.html` exec コマンドを使用して、ポッド内で任意のコマンドを直接発行できます。これは実行中のアプリケーションをデバッグするのに役に立ちます。	アプリ開発者、AWS DevOps、AWS システム管理者

ファイルをポッドにコピーします。	このポッドのデフォルト index.html ファイルを削除します。
	
	`kubectl exec $POD_NAME -- rm /usr/share/nginx/html/index.html`
	
	カスタマイズされたローカルファイル index.html をポッドにアップロードします。
	
	`kubectl cp index.html $POD_NAME:/usr/share/nginx/html/`
	
	cp コマンドを使用して、任意のポッドにファイルを直接変更または追加できます。
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

ポート転送を使用して変更を表示します。	ポートフォワーディングを使用して、このポッドに加えた変更を確認します。
	`kubectl port-forward pod/$POD_NAME 8080:80`
	次の URL をブラウザで開きます。
	`http://localhost:8080`
	`index.html` ファイルに適用された変更がブラウザに表示されるはずです。
		アプリ開発者、AWS DevOps、AWS システム管理者

リソースの削除

タスク	説明	必要なスキル

ロードバランサーを削除するには、 を参照してください。	入力を削除します。
	`kubectl delete ingress/nginx-ingress`
	出力は、以下を返します。
	`ingress.networking.k8s.io "nginx-ingress" deleted`
	サービスを削除します。
	`kubectl delete service/nginx-service`
		アプリ開発者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル

	出力は、以下を返します。
	`service "nginx-service" deleted`
	ロードバランサーコントローラーを削除します。
	`helm delete aws-load-balancer-controller -n kube-system`
	出力は、以下を返します。
	`release "aws-load-balancer-controller" uninstalled`
	サービスアカウントを削除します。
	`eksctl delete iamserviceaccount --cluster $CLUSTER_NAME --namespace kube-system --name aws-load-balancer-controller`	

タスク	説明	必要なスキル

デプロイを削除します。	次のコマンドを使用して リソースを削除します。
	`kubectl delete deploy/nginx-deployment`
	出力は、以下を返します。
	`deployment.apps "nginx-deployment" deleted`
		アプリ開発者、AWS DevOps、AWS システム管理者

クラスターを削除します。	次のコマンドを使用して、`my-fargate` が EKS クラスターのクラスター名を削除します。 `eksctl delete cluster --name $CLUSTER_NAME` このコマンドは、関連するすべてのリソースを含むクラスター全体を削除します。	アプリ開発者、AWS DevOps、AWS システム管理者
IAM ポリシーを削除するには	AWS CLI を使用して、以前に作成したポリシーを削除します。 `aws iam delete-policy --policy-arn $POLICY_ARN`	アプリ開発者、AWS 管理者、AWS DevOps

トラブルシューティング

問題	ソリューション
ターゲットのアベイラビリティーゾーンにはクラスターをサポートするのに十分な容量がないというクラスターの作成時のエラーメッセージが表示されます。次の例に示すようなメッセージが表示されます。 `Cannot create cluster 'my-fargate' because us-east-1e, the targeted availability zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these availability zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1f`	エラーメッセージの推奨アベイラビリティーゾーンを使用してクラスターを再作成します。`clusterconfig-fargate.yaml` ファイルの最後の行にアベイラビリティーゾーンのリストを指定します (例： `availabilityZones: ["us-east-1a", "us-east-1b", "us-east-1c"]`)。

関連リソース

- Amazon EKS ドキュメント
- Amazon EKS でのアプリケーション負荷分散
- EKS ベストプラクティスガイド
- AWS Load Balancer Controller ドキュメント
- eksctl ドキュメント
- Amazon ECR 公開ギャラリー NGINX イメージ
- ヘルム-ドキュメント
- 実行中のポッドのデバッグ (Kubernetes ドキュメント)
- Amazon EKS の仕組み
- EKS クラスター作成エラー

追加情報

clusterconfig-fargate.yaml

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-fargate
  region: us-east-1

fargateProfiles:
  - name: fp-default
    selectors:
      - namespace: default
      - namespace: kube-system
```

nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "nginx-deployment"
  namespace: "default"
spec:
  replicas: 3
  selector:
    matchLabels:
      app: "nginx"
  template:
    metadata:
      labels:
        app: "nginx"
    spec:
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:latest
        ports:
        - containerPort: 80
```

nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
```

```
    alb.ingress.kubernetes.io/target-type: ip
  name: "nginx-service"
  namespace: "default"
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  type: NodePort
  selector:
    app: "nginx"
```

nginx-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: "default"
  name:  "nginx-ingress"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  rules:
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: "nginx-service"
              port:
                number: 80
```

index.html

```
<!DOCTYPE html>
<html>

<body>
    <h1>Welcome to your customized nginx!</h1>
    <p>You modified the file on this running pod</p>
</body>
```

```
</html>
```

Elastic Beanstalk を使用してコンテナをデプロイする

作成者: Thomas Scott (AWS)、Jean-Baptiste Guillois (AWS)

概要

Amazon Web Services (AWS) クラウドでは、AWS Elastic Beanstalk が Docker を利用可能なプラットフォームとしてサポートしているため、作成した環境でコンテナを実行できます。このパターンは、Elastic Beanstalk サービスを使用してコンテナをデプロイする方法を示しています。このパターンのデプロイでは、Docker プラットフォームベースのウェブサーバー環境が使用されます。

ウェブアプリケーションやサービスをデプロイ、スケーリングするために Elastic Beanstalk を使用する場合、コードをアップロードすると、デプロイが自動的に処理されます。キャパシティのプロビジョニング、ロードバランシング、自動スケーリング、アプリケーションのヘルスモニタリングも含まれます。Elastic Beanstalk を使用すると、ユーザーに代わって作成される AWS リソースを完全に制御できます。Elastic Beanstalk に対する追加料金はありません。アプリケーションを保存し、実行するための AWS リソースに料金を支払うだけです。

このパターンには、AWS Elastic Beanstalk コマンドラインインターフェイス (EB CLI) と AWS マネジメントコンソールを使用したデプロイ手順が含まれています。

ユースケース

以下に、Elastic Beanstalk ラベルの一般的なユースケースを示します。

- プロトタイプ環境をデプロイして、フロントエンドアプリケーションのデモを行います。(このパターンでは Dockerfile を例として使用しています)。
- API をデプロイして、特定のドメインの API リクエストを処理します。
- Docker-Compose を使用して、オーケストレーションソリューションをデプロイします(このモードでは、`docker-compose.yml` は実際の例として使用されません)。

前提条件と制限

前提条件

- AWS アカウント
- AWS EB CLI がローカルにインストールされています
- Docker がローカルマシンにインストールされています

機能制限

- 無料プランでは、Docker のプル制限は IP アドレスごとに 6 時間あたり 100 回までです。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Elastic Compute Cloud (Amazon EC2) インスタンス
- セキュリティグループ
- Application Load Balancer
- Auto Scaling グループ

ターゲット アーキテクチャ

自動化とスケール

AWS Elastic Beanstalk は、リクエストの数に基づいて自動的にスケールします。環境枠用に作成された AWS リソースには、Application Load Balancer、Auto Scaling グループ、1 つ以上の Amazon EC2 インスタンスが含まれます。

ロードバランサーは、Auto Scaling グループに属する Amazon EC2 インスタンスの前に配置されています Amazon EC2 Auto Scaling は、アプリケーションへの負荷の増大に対応するために追加の Amazon EC2 インスタンスを自動的に開始します。アプリケーションへの負荷が軽減されると、Amazon EC2 Auto Scaling はインスタンスを停止しますが、少なくとも 1 つのインスタンスは引き続き実行されます。

自動スケーリングトリガー

Elastic Beanstalk 環境の Auto Scaling グループは、2 つの Amazon CloudWatch アラームを使用してスケーリングオペレーションを開始します。各インスタンスの 5 分間の平均アウトバウンドネットワークトラフィックが 6 MB 以上または 2 MB 以下の場合は、デフォルトのトリガーがスケーリングされます。Amazon EC2 Auto Scaling を効率的に使用するには、アプリケーション、インスタンスタイプ、サービス要件に合ったトリガーを設定します。レイテンシー、ディスク I/O、CPU 使用率、リクエスト数などの複数の統計に基づいて、スケールすることができます。詳細については、「自動スケーリングトリガー」を参照してください。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS EB コマンドラインインターフェイス (EB CLI)」は、Elastic Beanstalk 環境の作成、設定、管理に使用できるコマンドラインクライアントです。
- 受信したアプリケーションまたはネットワークトラフィックを複数のターゲットに分散するには、Elastic Load Balancing を使用します。例えば、1 つまたは複数のアベイラビリティーゾーンの Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレスにトラフィックを分散できます。

その他のサービス

- Docker は、ライブラリ、システムツール、コード、ランタイムを含むコンテナと呼ばれる、標準化されたユニットにソフトウェアをパッケージ化します。

コード

このパターンのコードは、GitHub 内の「クラスターサンプルアプリケーション」リポジトリで利用できます。

エピック

Dockerfile で構築する

タスク	説明	必要なスキル
リモートリポジトリをクローンを作成します。	• リポジトリのクローンを作成するには、`git clone https://github.com/aws-samples/cluster-sample-app.git` コマンドを実行します。< /p>	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
Elastic Beanstalk Docker プロジェクトを初期化します。	1. ルートディレクトリに aws.json という名前のファイルを作成します。 2. aws.json のファイルに次のコードを追加します。 3. プロジェクトのルートディレクトリで eb init -p docker コマンドを実行します。	アプリ開発者、AWS 管理者、AWS DevOps
プロジェクトをローカルでテストします。	1. プロジェクトのルートディレクトリで eb local run コマンドを実行します。 2. http://localhost に移動して、アプリケーションをテストします。	アプリ開発者、AWS 管理者、AWS DevOps

```
{
         "AWSEBDoc
kerrunVersion":"1",
         "Image":{
            "Name":"c
luster-sample-app"
         },
         "Ports":[
            {

  "ContainerPort":80
,

  "HostPort":8080
            }
         ]
      }
```

EB CLI を使用してデプロイする

タスク	説明	必要なスキル
デプロイコマンドの実行	1. プロジェクトのルートディレクトリで `eb create docker-sample-cluster-app` コマンドを実行します。	アプリ開発者、AWS 管理者、AWS DevOps
デプロイされたバージョンにアクセスします。	デプロイコマンドの終了後、`eb open` コマンドを使用してプロジェクトにアクセスします。	アプリ開発者、AWS 管理者、AWS DevOps

コンソールを使用してデプロイする

タスク	説明	必要なスキル
ブラウザを使用してアプリケーションをデプロイします。	1. コンソールを開きます。 2. Elastic Beanstalk コンソールに移動します。 3. [アプリケーションの作成] を選択します。 4. [アプリケーション名] には、Cluster-Sample-App と入力します。 5. Docker をプラットフォームとして選択します。 6. [コードのアップロード] を選択します。 7. ローカルの.zip ファイル (クローンプロジェクトのルートディレクトリにあります)、またはパブリッ	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	ク Amazon Simple Storage Service (Amazon S3) URL を選択します。	
デプロイされたバージョンにアクセスします。	デプロイ後、デプロイされたアプリケーションにアクセスし、表示された URL を選択します。	アプリ開発者、AWS 管理者、AWS DevOps

関連リソース

- 「ウェブサーバー環境」
- 「macOS で EB CLI をインストール」
- 「EB CLI の手動インストール」

追加情報

Elastic Beanstalk を使用するメリット

- インフラストラクチャの自動プロビジョニング
- 基盤となるプラットフォームの自動管理
- アプリケーションをサポートするための自動パッチ適用とアップデート
- アプリケーションの自動スケーリング
- ノード数のカスタマイズが可能
- 必要に応じて、インフラストラクチャーコンポーネントにアクセス可能
- 他のコンテナデプロイのソリューションよりもデプロイが簡単

Lambda 関数、Amazon VPC、およびサーバーレスアーキテクチャを使用して静的アウトバウンド IP アドレスを生成する

作成者:トーマス・スコット (AWS)

概要

このパターンでは、サーバーレスアーキテクチャを使用して、Amazon Web Services (AWS) クラウドで静的アウトバウンド IP アドレスを生成する方法を説明します。組織がセキュアファイル転送プロトコル (SFTP) を使用して別の事業体にファイルを送信したい場合、この方法を利用することができる。つまり、事業体は、ファイアウォールを通過するファイルを許可する IP アドレスにアクセスできる必要があります。

このパターンのアプローチは、Elastic IP アドレスをアウトバウンド IP アドレスとして使用する AWS Lambda 関数を作成するのに役立ちます。このパターンの手順に従うことで、Lambda 関数と、静的 IP アドレスを持つインターネットゲートウェイ経由でアウトバウンドトラフィックをルーティングする仮想プライベートクラウド (VPC) を作成できます。静的 IP アドレスを使用するには、Lambda 関数を VPC とそのサブネットにアタッチします。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS Identity and Access Management (IAM) アクセス権限で、Lambda 関数の作成とデプロイ、VPC とそのサブネットの作成が可能です。詳細については、AWS Lambda ドキュメントの「実行ロールとユーザー権限」を参照してください。
- Infrastructure as Code (IaC) を使用してこのパターンのアプローチを実装する場合は、AWS Cloud9 などの統合開発環境 (IDE) が必要です。詳細については、AWS Cloud9 ドキュメントにある「AWS Cloud9 とは?」を参照してください。

アーキテクチャ

このパターンのサーバーレスアーキテクチャを次の図に示します。

この図表は、次のワークフローを示しています：

1. アウトバウンドトラフィックは NAT gateway 1 の Public subnet 1 から離れます。
2. アウトバウンドトラフィックは NAT gateway 2 の Public subnet 2 から離れます。
3. Lambda 関数は Private subnet 1 または Private subnet 2 で実行できます。
4. Private subnet 1 と Private subnet 2 はパブリックサブネットの NAT ゲートウェイにトラフィックをルーティングします。
5. NAT ゲートウェイは、パブリックサブネットからインターネットゲートウェイにアウトバウンドトラフィックを送信します。
6. アウトバウンドデータは、インターネットゲートウェイから外部サーバーに転送されます。

テクノロジースタック

- Lambda
- Amazon Virtual Private Cloud (Amazon VPC)

自動化とスケール

異なるアベイラビリティーゾーンにある 2 つのパブリックサブネットと 2 つのプライベートサブネットを使用することで、高可用性 (HA) を確保できます。1 つのアベイラビリティーゾーンが使用できなくなっても、パターンのソリューションは引き続き機能します。

ツール

- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「Amazon VPC」— Amazon Virtual Private Cloud (Amazon VPC) では、AWS クラウドの論理的に隔離されたセクションをプロビジョニングすることで、ユーザーが定義した仮想ネットワーク内で AWS リソースを起動できます。仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

VPC を新規作成する

タスク	説明	必要なスキル
新しい VPC を作成します。	AWS マネジメントコンソールにサインインし、Amazon VPC コンソールを開いて、`10.0.0.0/25` を IPv4 CIDR範囲とする `Lambda VPC` という名前のVPCを作成します。 VPC の作成に関する詳細は、Amazon VPC ドキュメントの「Amazon VPC の使用開始」を参照してください。	AWS 管理者

2 つのパブリックサブネットを作成します。

タスク	説明	必要なスキル
最初のパブリックサブネットを作成します。	1. Amazon VPC コンソールで [サブネット] を選択し、その後 [サブネットの作成] を選択します。 2. 名前タグ に「`public-one`」を入力します。 3. VPCで、`Lambda VPC` を選択します。 4. アベイラビリティーゾーンを選択して記録します。 5. [IPv4 CIDR ブロック] の場合は、`10.0.0.0/28` を	AWS 管理者

タスク	説明	必要なスキル
	入力して、[サブネットの作成] を選択します。	
2 番目のパブリックサブネットを作成します。	1. Amazon VPC コンソールで [サブネット] を選択し、その後 [サブネットの作成] を選択します。 2. 名前タグ に「`public-two` 」を入力します。 3. VPCで、`Lambda VPC` を選択します。 4. **Important** アベイラビリティーゾーンを選択して記録します。: `public-one` サブネットを含むアベイラビリティーゾーンを使用することはできません。 5. [IPv4 CIDR ブロック] の場合は、`10.0.0.16/28` を入力して、[サブネットの作成] を選択します。	AWS 管理者

プライベートサブネットを 2 つ作成する

タスク	説明	必要なスキル
最初のプライベートサブネットを作成します。	1. Amazon VPC コンソールで [サブネット] を選択し、そ	AWS 管理者

タスク	説明	必要なスキル
	の後 [サブネットの作成] を選択します。 2. 名前タグ に「private-one 」を入力します。 3. VPCで、Lambda VPC を選択します。 4. 前に作成した public-one サブネットを含むアベイラビリティーゾーンを選択します。 5. [IPv4 CIDR ブロック] の場合は、10.0.0.32/28 を入力して、[サブネットの作成] を選択します。	
2 番目のプライベートサブネットを作成する	1. Amazon VPC コンソールで [サブネット] を選択し、その後 [サブネットの作成] を選択します。 2. 名前タグ に「private-two 」を入力します。 3. VPCで、Lambda VPC を選択します。 4. 前に作成した public-two サブネットを含む同じアベイラビリティーゾーンを選択します。 5. [IPv4 CIDR ブロック] の場合は、10.0.0.64/28 を入力して、[サブネットの作成] を選択します。	AWS 管理者

NAT ゲートウェイ用に 2 つの Elastic IP アドレス作成する

タスク	説明	必要なスキル
最初の Elastic IP アドレスを作成します。	1. Amazon VPC コンソールで [Elastic IP] を選択し、[新しいアドレスを割り当てる] を選択します。 2. [割り当て] を選択し、新しく作成した Elastic IP アドレスの [割り当て ID] を記録します。 **Note** この Elastic IP アドレスは、最初の NAT ゲートウェイに使用されます。	AWS 管理者
2 番目の Elastic IP アドレスを作成します。	1. Amazon VPC コンソールで [Elastic IP] を選択し、[新しいアドレスを割り当てる] を選択します。 2. [割り当て] を選択し、この 2 番目の Elastic IP アドレスの [割り当て ID] を記録します。 **Note** この Elastic IP アドレスは、2 番目の NAT	AWS 管理者

タスク	説明	必要なスキル
	ゲートウェイに使用されます。	

インターネットゲートウェイを作成する

タスク	説明	必要なスキル
インターネットゲートウェイを作成します。	1. Amazon VPC コンソールで [インターネットゲートウェイ] を選択し、[インターネットゲートウェイの作成] を選択します。 2. 名前として `Lambda internet gateway` を入力し、[インターネットゲートウェイの作成] を選択します。インターネットゲートウェイ ID を記録しておくことを確認してください。	AWS 管理者
インターネットゲートウェイを VPC にアタッチします。	作成したインターネットゲートウェイを選択して、[アクション]、[VPC にアタッチ] を選択します。	AWS 管理者

NAT ゲートウェイを 2 つ作成します。

タスク	説明	必要なスキル
最初の NAT ゲートウェイを作成します。	1. Amazon VPC コンソールで [NAT ゲートウェイ] を選択	AWS 管理者

タスク	説明	必要なスキル
	し、[NAT ゲートウェイの作成] を選択します。 2. NAT ゲートウェイ名として `nat-one` を入力します。 3. NAT ゲートウェイを作成するサブネットとして `public-one` を選択します。 4. [接続タイプ] には [パブリック] を選択します。 5. [Elastic IP 割り当て ID] には、前に作成した最初の Elastic IP アドレスを選択し、NAT ゲートウェイに関連付けます。 6. [NAT ゲートウェイを作成] を選択します。	

タスク	説明	必要なスキル
2 番目の NAT ゲートウェイを作成します。	1. Amazon VPC コンソールで [NAT ゲートウェイ] を選択し、[NAT ゲートウェイの作成] を選択します。 2. NAT ゲートウェイ名として `nat-two` を入力します。 3. NAT ゲートウェイを作成するサブネットとして `public-two` を選択します。 4. [接続タイプ] には [パブリック] を選択します。 5. [Elastic IP 割り当て ID] には、前に作成した 2 つ目の Elastic IP アドレスを選択し、NAT ゲートウェイに関連付けます。 6. [NAT ゲートウェイを作成] を選択します。	AWS 管理者

パブリックサブネットとプライベートサブネットのルートテーブルを作成します。

タスク	説明	必要なスキル
パブリックワンサブネット用のルートテーブルを作成します。	1. Amazon VPC コンソールで [ルートテーブル] を選択し、[ルートテーブルの作成] を選択します。 2. ルートテーブル名として「`public-one-subnet`」を入力し、[ルートテーブルの作成] を選択します。	AWS 管理者

タスク	説明	必要なスキル
	3. [public-one-subnet ルートテーブル]、[ルートの編集]、[ルートの追加] の順に選択します。 4. [送信先] ボックスで 0.0.0.0 を指定し、[ターゲット] リストでインターネットゲートウェイ ID を選択します。 5. [サブネットの関連付け] タブで [サブネットの関連付けを編集] を選択し、10.0.0.0/28 CIDR 範囲の public-one サブネットを選択して、[関連付けを保存] を選択します。 6. [Save Changes] を選択します。	

タスク	説明	必要なスキル
パブリック2 サブネットのルートテーブルを作成します。	1. Amazon VPC コンソールで [ルートテーブル] を選択し、[ルートテーブルの作成] を選択します。 2. ルートテーブル名として「`public-two-subnet`」を入力し、[ルートテーブルの作成] を選択します。 3. [`public-two-subnet` ルートテーブル]、[ルートの編集]、[ルートの追加] の順に選択します。 4. [送信先] ボックスで `0.0.0.0` を指定し、[ターゲット] リストでインターネットゲートウェイ ID を選択します。 5. [サブネットの関連付け] タブで [サブネットの関連付けを編集] を選択し、`10.0.0.16/28` CIDR 範囲の `public-two` サブネットを選択して、[関連付けを保存] を選択します。 6. [Save Changes] を選択します。	AWS 管理者

タスク	説明	必要なスキル
プライベートワンサブネットのルートテーブルを作成します。	1. Amazon VPC コンソールで [ルートテーブル] を選択し、[ルートテーブルの作成] を選択します。 2. ルートテーブル名として「`private-one-subnet`」を入力し、[ルートテーブルの作成] を選択します。 3. [`private-one-subnet` ルートテーブル]、[ルートの編集]、[ルートの追加] の順に選択します。 4. [送信先] ボックスで `0.0.0.0` を指定し、[ターゲット] リストの `public-one` サブネット内の NAT ゲートウェイを選択します。 5. [サブネットの関連付け] タブで [サブネットの関連付けを編集] を選択し、`10.0.0.32/28` CIDR 範囲の `private-one` サブネットを選択して、[関連付けを保存] を選択します。 6. [Save Changes] を選択します。	AWS 管理者

タスク	説明	必要なスキル
プライベートツーサブネットのルートテーブルを作成します。	1. Amazon VPC コンソールで [ルートテーブル] を選択し、[ルートテーブルの作成] を選択します。 2. ルートテーブル名として「`private-two-subnet`」を入力し、[ルートテーブルの作成] を選択します。 3. [`private-two-subnet` ルートテーブル]、[ルートの編集]、[ルートの追加] の順に選択します。 4. [送信先] ボックスで `0.0.0.0` を指定し、[ターゲット] リストの `public-two` サブネット内の NAT ゲートウェイを選択します。 5. [サブネットの関連付け] タブで [サブネットの関連付けを編集] を選択し、`10.0.0.64/28` CIDR 範囲の `private-two` サブネットを選択して、[関連付けを保存] を選択します。 6. [Save Changes] を選択します。	AWS 管理者

Lambda 関数を作成し、VPC に追加して、ソリューションをテストする

タスク	説明	必要なスキル
新しい Lambda 関数の作成	1. AWS Lambda コンソールを開き、[関数の作成] を選択します。 2. [基本情報]の [関数名] に `Lambda test` を入力し、[ランタイム] で目的の言語を選択します。 3. [Create function (関数の作成)] を選択します。	AWS 管理者
Lambda 関数 を VPC に追加します。	1. AWS Lambda コンソールで、[関数] を選択し、前に作成した関数を選択します。 2. [Configuration] (設定)、VPC の順に選択します。 3. [編集] を選択し、Lambda VPC と両方のプライベートサブネットを選択します。 4. テスト用に [既定のセキュリティグループ] を選択し、[保存] を選択します。	AWS 管理者
外部サービスを呼び出すコードを記述します。	1. 選択したプログラミング言語で、IP アドレスを返す外部サービスを呼び出すコードを記述します。 2. 返された IP アドレスが Elastic IP アドレスのいずれかと一致することを確認します。	AWS 管理者

関連リソース

- VPC 内のリソースにアクセスするように Lambda 関数を設定する

Amazon ECR リポジトリに移行するときに、重複するコンテナイメージを自動的に特定する

作成者: Rishabh Yadav (AWS) と Rishi Singla (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、異なるコンテナリポジトリに保存されているイメージが重複しているかどうかを識別する自動化されたソリューションを提供します。このチェックは、他のコンテナリポジトリから Amazon Elastic Container Registry (Amazon ECR) にイメージを移行する場合に便利です。

基本的な情報として、このパターンでは、イメージダイジェスト、マニフェスト、タグなど、コンテナイメージのコンポーネントについても説明します。Amazon ECR への移行を計画するときは、イメージのダイジェストを比較して、コンテナレジストリ間でコンテナイメージを同期することを決定できます。コンテナイメージを移行する前に、重複を防ぐために、これらのイメージが Amazon ECR リポジトリに既に存在するかどうかを確認する必要があります。ただし、イメージダイジェストを比較して重複を検出するのは難しく、初期移行フェーズで問題が発生する可能性があります。 このパターンは、異なるコンテナレジストリに保存されている 2 つの類似イメージのダイジェストを比較し、ダイジェストが異なる理由を説明し、イメージを正確に比較するのに役立ちます。

前提条件と制限

- アクティブな AWS アカウント
- Amazon ECR パブリックレジストリへのアクセス
- 以下に精通していること AWS のサービス。
  - AWS CodeCommit
  - AWS CodePipeline
  - AWS CodeBuild
  - AWS Identity and Access Management (IAM)
  - Amazon Simple Storage Service (Amazon S3)
- CodeCommit 認証情報の設定 (手順を参照)

アーキテクチャ

コンテナイメージコンポーネント

次の図は、コンテナイメージのコンポーネントの一部を示しています。これらのコンポーネントは、図の後に説明されています。

用語と定義

以下の用語は、Open Container Initiative (OCI) イメージ仕様で定義されています。

- レジストリ： イメージのストレージと管理のためのサービス。
- クライアント： レジストリと通信し、ローカルイメージと連携するツール。
- プッシュ： レジストリにイメージをアップロードするプロセス。
- プル： レジストリからイメージをダウンロードするプロセス。
- Blob: レジストリによって保存され、ダイジェストによって対処できるコンテンツのバイナリ形式。
- インデックス： さまざまなコンピュータプラットフォーム (x86-64 や ARM 64 ビットなど) またはメディアタイプに対して複数のイメージマニフェストを識別するコンストラクト。詳細については、「OCI Image Index Specification」を参照してください。
- マニフェスト： マニフェストのエンドポイントを介してアップロードされるイメージまたはアーティファクトを定義する JSON ドキュメント。マニフェストは、記述子を使用してリポジトリ内の他の BLOB を参照できます。詳細については、「OCI Image Manifest Specification」を参照してください。
- ファイルシステムレイヤー： イメージのシステムライブラリおよびその他の依存関係。
- 設定： アーティファクトメタデータを含み、マニフェストで参照される BLOB。詳細については、「OCI イメージ設定仕様」を参照してください。
- オブジェクトまたはアーティファクト： BLOB として保存され、設定とともに付随するマニフェストに関連付けられている概念的なコンテンツ項目。
- ダイジェスト： マニフェストのコンテンツの暗号化ハッシュから作成される一意の識別子。イメージダイジェストは、イミュータブルなコンテナイメージを一意に識別するのに役立ちます。ダイジェストを使用してイメージをプルすると、オペレーティングシステムまたはアーキテクチャで毎回同じイメージがダウンロードされます。詳細については、「OCI イメージ仕様」を参照してください。

- タグ： 人間が読み取れるマニフェスト識別子。イミュータブルなイメージダイジェストと比較して、タグは動的です。イメージを指すタグは、基になるイメージダイジェストは同じままですが、イメージ間で変更および移動できます。

ターゲット アーキテクチャ

次の図は、Amazon ECR とプライベートリポジトリに保存されているイメージを比較して、重複するコンテナイメージを識別するために、このパターンによって提供されるソリューションの高レベルのアーキテクチャを示しています。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- AWS CodeBuildは、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。

コード

このパターンのコードはGitHub リポジトリ の自動ソリューションでリポジトリ間で重複するコンテナイメージを識別するために使用できます。

ベストプラクティス

- AWS CloudFormation ベストプラクティス

- AWS CodePipeline ベストプラクティス

エピック

Amazon ECR パブリックリポジトリとプライベートリポジトリからコンテナイメージをプルする

タスク	説明	必要なスキル

Amazon ECR パブリックリポジトリからイメージをプルします。	ターミナルから次のコマンドを実行して、Amazon ECR パブリックリポジトリamazonlinux からイメージをプルします。
	`$~ % docker pull public.ecr.aws/amazonlinux/amazonlinux:2018.03`
	イメージがローカルマシンにプルされると、イメージインデックスを表す次のプルダイジェストが表示されます。
	`2018.03: Pulling from amazonlinux/amazonlinux`
	`4ddc0f8d367f: Pull complete`
	`Digest: sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5`
	`Status: Downloaded newer image for public.ecr.aws/ama`
		アプリ開発者、AWS DevOps、AWS 管理者

タスク	説明	必要なスキル
	zonlinux/amazonlin ux:2018.03 public.ecr.aws/a mazonlinux/amazonl inux:2018.03	

タスク	説明	必要なスキル

イメージを Amazon ECR プライベートリポジトリにプッシュします。	1. 米国東部 (バージニア北部) リージョン () `test_ecr_repository` に という名前のプライベート Amazon ECR リポジトリを作成しますus-east-1 。
	`$~ % aws ecr get-login -password --region us-east-1 | docker login --username AWS --password-stdin <account-id>.dkr.ecr.us-east-1.amazonaws.com`
	`Login Succeeded`
	ここで、 <account-id> は を参照します AWS アカウント。
	2. 前にプルしたローカルイメージにタグを付けます。値を使用して public.ecr.aws/amazonlinux/amazonlinux:2018.03 Amazon ECR プライベートリポジトリにプッシュします。
	`$~ % docker tag public.ecr.aws/amazonlinux/amazonlinux:2018.03 <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
		AWS 管理者、AWS DevOps、アプリ開発者

タスク	説明	必要なスキル

	`$~ % docker push <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	
	イメージを Amazon ECR リポジトリにプッシュすると、Docker はイメージインデックスではなく、基盤となるイメージをプッシュします。
	
	`The push refers to repository [<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository] d5655967c2c4: Pushed latest: digest: sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02 size: 529`	

タスク	説明	必要なスキル

Amazon ECR プライベートリポジトリから同じイメージをプルします。	1. ターミナルから次のコマンドを実行して、以前に Amazon ECR プライベートリポジトリにプッシュしたイメージをプルします。
	`$~ % docker pull <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	`latest: Pulling from test_ecr_repository`
	`Digest: sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02`
	`Status: Image is up to date for <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	`<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	このイメージのダイジェストは、Amazon ECR プライベートリポジトリにプッシュしたイメージのダイジェストと一致し、基盤となるイメージを表します。この値は、パブリックリポジトリからプルしたイメー
		アプリ開発者、AWS DevOps、AWS 管理者

タスク	説明	必要なスキル

	ジインデックスと一致しません。
	2. 確認するには、ダイジェストでイメージインデックスを取得します。
	
	curl -k -H “Authorization: Bearer $TOKEN” https://public.ecr.aws/v2/amazonlinux/amazonlinux/manifests/sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b55
	{
	“schemaVersion”: 2,
	“mediaType”: “application/vnd.docker.distribution.manifest.list.v2+json”,
	“manifests”: [
	{
	“mediaType”: “application/vnd.docker.distribution.manifest.v2+json”,
	“size”: 529,
	“digest”: “sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02",
	“platform”: {	

タスク	説明	必要なスキル

	<pre>“architec
	ture”: “amd64”,
	“os”:
	“linux”
	}
	}
	]
	}</pre>	

イメージマニフェストを比較する

タスク	説明	必要なスキル

Amazon ECR パブリックリポジトリに保存されているイメージのマニフェストを見つけます。	ターミナルから次のコマンドを実行して、Amazon ECR パブリックリポジトリpublic.ecr.aws/amazonlinux/amazonlinux:2018.03 からイメージのマニフェストをプルします。
	<pre>$~ % docker manifest
	inspect public.ec
	r.aws/amazonlinux/
	amazonlinux:2018.03
	{
	"schemaVersion": 2,
	"mediaType":
	"application/vnd.d
	ocker.distribution
	.manifest.list.v2+
	json",
	"manifests": [
	{
	"mediaType":
	"application/vnd.d</pre>
		AWS 管理者、AWS DevOps、アプリ開発者

タスク	説明	必要なスキル

	ocker.distribution
	.manifest.v2+json",
	"size": 529,
	"digest":
	"sha256:52db900007
	3d93b9bdee6a7246a6
	8c35a741aaade05a8f
	4febba0bf795cdac02",
	"platform": {
	"architec
	ture": "amd64",
	"os":
	"linux"
	}
	}
	]
	}	

タスク	説明	必要なスキル

Amazon ECR プライベートリポジトリに保存されているイメージのマニフェストを見つけます。	ターミナルから次のコマンドを実行して、Amazon ECR プライベートリポジトリ<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest からイメージのマニフェストをプルします。
	
	`$~ % docker manifest inspect <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	
	`{ "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "config": { "mediaType": "application/vnd.docker.container.image.v1+json", "size": 1477, "digest": "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68" }, "layers": [ {`
		AWS DevOps、AWS システム管理者、アプリケーション開発者

タスク	説明	必要なスキル
	"mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 62267075, "digest": "sha256:4ddc0f8d367f424871a060e2067749f32bd36a91085e7142dcb159952f2d71453" }] }	

タスク	説明	必要なスキル
Docker によってプルされたダイジェストと Amazon ECR プライベートリポジトリ内のイメージのマニフェストダイジェストを比較します。	もう 1 つの問題は、docker pull コマンドによって提供されるダイジェストが、イメージ のマニフェストのダイジェストと異なる理由です`<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest` 。 docker pull に使用されるダイジェストは、レジストリに保存されているイメージマニフェストのダイジェストを表します。このダイジェストはハッシュチェーンのルートと見なされます。マニフェストには、ダウンロードされて Docker にインポートされるコンテンツのハッシュが含まれているためです。 Docker で使用されるイメージ ID は、このマニフェストで として確認できます`config.digest` 。これは、Docker が使用するイメージ設定を表します。したがって、マニフェストがエンベロープであり、イメージがエンベロープの内容であると言うことができます。マニフェストダイジェストは常にイメージ ID とは異なります。ただし、特定のマニフェストは常に同じイメー	AWS DevOps、AWS システム管理者、アプリケーション開発者

タスク	説明	必要なスキル
	ジ ID を生成する必要があります。マニフェストダイジェストはハッシュチェーンであるため、特定のイメージ ID で常に同じであることを保証することはできません。ほとんどの場合、同じダイジェストを生成しますが、Docker はそれを保証できません。マニフェストダイジェストの考えられる違いは、Docker が gzip でローカルに圧縮された BLOB を保存しないことです。したがって、レイヤーをエクスポートすると、圧縮されていないコンテンツは同じままですが、別のダイジェストが発生する可能性があります。イメージ ID は、圧縮されていないコンテンツが同じであることを確認します。つまり、イメージ ID がコンテンツアドレス指定可能な識別子 () になりましたchainID。 この情報を確認するには、Amazon ECR パブリックリポジトリとプライベートリポジトリで docker inspect コマンドの出力を比較します。 1. Amazon ECR パブリックリポジトリに保存されているイメージのターミナルか	

タスク	説明	必要なスキル

	ら次のコマンドを実行します。
	`$~ % docker inspect public.ecr.aws/amazonlinux/amazonlinux:2018.03`
	コマンドからの出力については、「追加情報」セクションを参照してください。
	2. Amazon ECR プライベートリポジトリに保存されているイメージのターミナルから次のコマンドを実行します。
	`$~ % docker inspect <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	コマンドからの出力については、「追加情報」セクションを参照してください。
	結果は、両方のイメージに同じイメージ ID ダイジェストとレイヤーダイジェストがあることを確認します。
	ID: f7cee5e1af28ad4e147589c474d399b12d9b	

タスク	説明	必要なスキル
	551ef4c3e11e02d982fce5eebc68 レイヤー: d5655967c2c4e8d68f8ec7cf753218938669e6c16ac1324303c073c736a2e2a2 さらに、ダイジェストは、ローカルで管理されるオブジェクトのバイト数 (ローカルファイルはコンテナイメージレイヤーの tar です) またはレジストリサーバーにプッシュされる BLOB に基づいています。ただし、BLOB をレジストリにプッシュすると、tar は圧縮され、ダイジェストは圧縮された tar ファイルで計算されます。したがって、Docker プルダイジェスト値の違いは、レジストリ (Amazon ECR プライベートまたはパブリック) レベルで適用される圧縮によって発生します。 **Note** この説明は、Docker クライアントの使用に固有です。この動作は、nerdctl や Finch などの他のクライアントでは表示されません。これは、プッシュおよびプルオペレー	

タスク	説明	必要なスキル
	ション中にイメージが自動的に圧縮されないためです。	

Amazon ECR パブリックリポジトリとプライベートリポジトリ間で重複するイメージを自動的に識別する

タスク	説明	必要なスキル
リポジトリをクローン作成します。	このパターンの Github リポジトリをローカルフォルダにクローンします。 `$git clone https://github.com/aws-samples/automated-solution-to-identify-duplicate-container-images-between-repositories`	AWS 管理者、AWS DevOps
CI/CD パイプラインをセットアップします。	GitHub リポジトリには、パイプラインを設定する AWS CloudFormation スタックを作成する `.yaml` ファイルが含まれています AWS CodePipeline。 1. にサインイン AWS Management Console し、AWS CloudFormation コンソールを開きます。 2. クローンされたリポジトリの `code`フォルダにあるテンプレート`pipeline.`	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	`yaml` ファイルを使用してスタックを作成します。 3. パラメータのデフォルト値を受け入れるか変更します。以下の値を指定します。 • スタックの名前 • `ArtifactStoreBucketName` – AWS CodePipeline アーティファクトの保存に使用される既存の S3 バケット • `OutputBucket` – 重複イメージの URIs の保存に使用される既存の S3 バケット • `SourceImageFile` – パブリックリポジトリからのイメージ URIs `input.txt` を含むという名前の既存のテキストファイル。Amazon ECR プライベートリポジトリと照合して重複を検出します。 4. スタックオプションを確認して調整し、送信を選択してテンプレートを実行します。 パイプラインは、パブリックリポジトリにも存在するプライベートリポジトリ内のイ	

タスク	説明	必要なスキル
	メージを識別するために、2 つのステージ (アーキテクチャ図に示すようにCodeCommit と CodeBuild) でセットアップされます。パイプラインは、次のリソースで設定されます。 • デプロイパイプラインのオーケストレーション用の CodePipeline。 • bash スクリプトと入力ファイルを保存する CodeCommit リポジトリ。bash スクリプトは、パブリックリポジトリとプライベートリポジトリのコンテナイメージ IDs を比較して重複を見つけるために使用されます。このチェックは、単一の で指定された 内のすべてのリポジトリ AWS アカウント で実行されます AWS リージョン。 • Amazon ECR リポジトリに既に存在するイメージを識別するために bash スクリプトを呼び出す CodeBuild プロジェクト。 • アクセスを許可するために必要な IAM ロール。 • イメージ URIs を含む出力ファイルを保存する S3 バケット。	

タスク	説明	必要なスキル
	• CodePipeline アーティファクトを保存する別の S3 バケット。	

タスク	説明	必要なスキル
CodeCommit リポジトリを入力します。	CodeCommit リポジトリに入力するには、次の手順を実行します。 1. CodeCommit コンソールを開き、CloudFormation スタックを作成した AWS リージョン に移動します。 2. リストから CloudFormation スクリプトを使用してプロビジョニングしたリポジトリを見つけ、URL のクローンを選択し、HTTPS URL プロトコルをコピーしてリポジトリに接続します。 3. コマンドプロンプトを開き、前のステップでコピーした HTTPS URL を使用して git クローンコマンドを実行します。 4. ルートディレクトリに移動します。という名前のファイルを作成し`input.txt` 、プライベート Amazon ECR リポジトリで検索する Amazon ECR パブリックイメージレジストリ URIs をこのファイルに入力します。 5. GitHub リポジトリの自動ソリューションのローカルコピー`input.txt` からファイル `script.sh` 、`buildspec.yml` 、お	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

	よび をコピーして、リポジトリ間の重複コンテナイメージをクローンされた CodeCommit リポジトリに識別します。 https://github.com/aws-samples/automated-solution-to-identify-duplicate-container-images-between-repositories/
	6. 次のコマンドを使用して、ファイルを CodeCommit にアップロードします。
	`git add .`
	`git commit -m “added input files”`
	`git push`	

タスク	説明	必要なスキル
クリーンアップを行います。	今後の料金が発生しないようにするには、以下の手順に従ってリソースを削除します。 1. CodePipeline アーティファクトを保存する S3 バケットに移動し、バケットを空にします。 2. 重複するイメージ URI を保存する S3 バケットに移動し、バケットを空にします。 URIs 3. CloudFormation コンソールに移動し、パイプラインをセットアップするために作成したスタックを削除します。	AWS 管理者

トラブルシューティング

問題	ソリューション
ターミナルまたはコマンドラインから CodeCommit リポジトリをプッシュ、プル、またはその他の方法で操作しようとすると、ユーザー名とパスワードを指定するように求められ、IAM ユーザーの Git 認証情報を指定する必要があります。	このエラーの最も一般的な原因は次のとおりです。 • ローカルコンピュータが認証情報管理をサポートしていないオペレーティングシステムを実行しているか、認証情報管理ユーティリティがインストールされていません。 • IAM ユーザーの Git 認証情報は、これらの認証情報管理システムのいずれかに保存されていません。

問題	ソリューション
	オペレーティングシステムおよびローカル環境によっては、認証情報マネージャーのインストール、オペレーティングシステム内の認証情報マネージャーの設定、またはローカル環境のカスタマイズを行い、認証情報ストレージを使用する必要がある場合があります。例えば、コンピュータで macOS が実行されている場合は、Keychain Access ユーティリティを使用して認証情報を保存できます。コンピュータで Windows が実行されている場合は、Git for Windows と一緒にインストールされている Git 認証情報マネージャーを使用できます。詳細については、CodeCommit ドキュメントの「Git 認証情報を使用した HTTPS ユーザーのセットアップ」および Git ドキュメントの「認証情報ストレージ」を参照してください。 CodeCommit

問題	ソリューション
Amazon ECR リポジトリにイメージをプッシュすると、HTTP 403 または「基本的な認証認証情報なし」というエラーが発生します。	aws ecr get-login-password コマンドを使用して Docker に対して正常に認証された場合でも、docker push コマンドまたは docker pull コマンドからこれらのエラーメッセージが表示されることがあります。既知の原因は次のとおりです。 • 別の リージョンに対して認証されています。詳細については、Amazon ECR ドキュメントの「プライベートレジストリ認証」を参照してください。 • アクセス許可がないリポジトリにプッシュするように認証されました。詳細については、Amazon ECR ドキュメントの「プライベートリポジトリポリシー」を参照してください。 • トークンの有効期限が切れています。`GetAuthorizationToken` オペレーションを使用して取得したトークンのデフォルトの有効期限は 12 時間です。

関連リソース

- リポジトリ間で重複するコンテナイメージを識別する自動ソリューション (GitHub リポジトリ)
- Amazon ECR パブリックギャラリー
- Amazon ECR のプライベートイメージ (Amazon ECR ドキュメント)
- AWS::CodePipeline::Pipeline リソース (AWS CloudFormation ドキュメント)
- OCI イメージ形式の仕様

追加情報

Amazon ECR パブリックリポジトリ内のイメージの Docker 検査の出力

```
[
    {
        "Id":
 "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68",
        "RepoTags": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest",
            "public.ecr.aws/amazonlinux/amazonlinux:2018.03"
        ],
        "RepoDigests": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/
test_ecr_repository@sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02",
            "public.ecr.aws/amazonlinux/
amazonlinux@sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2023-02-23T06:20:11.575053226Z",
        "Container":
 "ec7f2fc7d2b6a382384061247ef603e7d647d65f5cd4fa397a3ccbba9278367c",
        "ContainerConfig": {
            "Hostname": "ec7f2fc7d2b6",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
                "CMD [\"/bin/bash\"]"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
```

```
            "OnBuild": null,
            "Labels": {}
        },
        "DockerVersion": "20.10.17",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/bash"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 167436755,
        "VirtualSize": 167436755,
        "GraphDriver": {
            "Data": {
                "MergedDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/merged",
                "UpperDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/diff",
                "WorkDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/work"
            },
            "Name": "overlay2"
        },
```

```
        "RootFS": {
            "Type": "layers",
            "Layers": [

 "sha256:d5655967c2c4e8d68f8ec7cf753218938669e6c16ac1324303c073c736a2e2a2"
            ]
        },
        "Metadata": {
            "LastTagTime": "2023-03-02T10:28:47.142155987Z"
        }
    }
]
```

Amazon ECR プライベートリポジトリ内のイメージの Docker 検査の出力

```
[
    {
        "Id":
 "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68",
        "RepoTags": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest",
            "public.ecr.aws/amazonlinux/amazonlinux:2018.03"
        ],
        "RepoDigests": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/
test_ecr_repository@sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02",
            "public.ecr.aws/amazonlinux/
amazonlinux@sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2023-02-23T06:20:11.575053226Z",
        "Container":
 "ec7f2fc7d2b6a382384061247ef603e7d647d65f5cd4fa397a3ccbba9278367c",
        "ContainerConfig": {
            "Hostname": "ec7f2fc7d2b6",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
```

```
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
                "CMD [\"/bin/bash\"]"
            ],
            "Image":
"sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": {}
        },
        "DockerVersion": "20.10.17",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/bash"
            ],
            "Image":
"sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
```

```
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 167436755,
        "VirtualSize": 167436755,
        "GraphDriver": {
            "Data": {
                "MergedDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/merged",
                "UpperDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/diff",
                "WorkDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [

 "sha256:d5655967c2c4e8d68f8ec7cf753218938669e6c16ac1324303c073c736a2e2a2"
            ]
        },
        "Metadata": {
            "LastTagTime": "2023-03-02T10:28:47.142155987Z"
        }
    }
]
```

Kubernetes DaemonSet を使用して Amazon EKS ワーカーノードに SSM エージェントをインストールします

作成者: Mahendra Revanasiddappa (AWS)

概要

注、2021 年 9 月：Amazon EKS に最適化された最新の AMI では SSM エージェントが自動的にインストールされます。詳細については、リリースノートの「June 2021 AMI」セクションを参照してください。

Amazon Elastic Kubernetes Service (Amazon EKS) では、セキュリティガイドラインにより、ワーカーノードにはSecure Shell (SSH) キーペアがアタッチされていません。このパターンは、手動でインストールしたり、ノードの Amazon マシンイメージ (AMI) を置き換えたりする代わりに、Kubernetes DaemonSet リソースタイプを使用して AWS Systems Manager Agent (SSM Agent) をすべてのワーカーノードにインストールする方法を示しています。DaemonSet はワーカーノードの cron ジョブを使用して SSM エージェントのインストールをスケジュールします。このパターンを使用して他のパッケージをワーカーノードにインストールすることもできます。

クラスター内の問題をトラブルシューティングする場合、SSM Agent をオンデマンドでインストールすると、SSH キーペアがなくても、ワーカーノードとの SSH セッションの確立、ログの収集、またはインスタンス設定の確認が可能になります。

前提条件と制限

前提条件

- Amazon Elastic Compute Cloud (Amazon EC2) ワーカーノードを備えた既存の Amazon EKS クラスター。
- コンテナインスタンスには、SSM サービスと通信するために必要な許可がなければなりません。AWS Identity と Access Management (IAM) マネージドロール「AmazonSSMManagedInstanceCoreは、EC2 インスタンスで SSM エージェントを実行するために必要な許可を提供します。詳細については、 AWS Systems Manager のドキュメントを参照してください。

制約事項

- DaemonSets は Fargate プラットフォームではサポートされていないため、このパターンは AWS Fargate には適用されません。

- このパターンは Linux ベースのワーカーノードにのみ適用されます。
- DaemonSet ポッドは特権モードで実行されます。Amazon EKS クラスターに特権モードでポッドをブロックするウェブフックがある場合、SSM エージェントはインストールされません。

アーキテクチャ

このパターンのアーキテクチャを以下に図で示します。

ツール

ツール

- 「kubectl」は、Amazon EKS クラスターを操作するために使用されるコマンドラインユーティリティです。このパターンはkubectlを使用して Amazon EKS クラスターに DaemonSet をデプロイします。これにより、SSM エージェントがすべてのワーカーノードにインストールされます。
- Amazon EKS は、独自の Kubernetes コントロールプレーンまたはノードをインストール、操作、維持することなく、 で Kubernetes を簡単に実行できるようにするマネージドサービスです。Kubernetes は、コンテナ化されたアプリケーションのデプロイ、スケーリング、および管理を自動化するためのオープンソースシステムです。
- AWS Systems Manager Session Manager は、EC2 インスタンス、オンプレミスインスタンス、仮想マシン (VM) を、インタラクティブなワンクリックブラウザベースのシェルまたは AWS コマンドラインインターフェイス (AWS CLI) を介して管理できます。

コード

次のコードを使用して、Amazon EKS クラスターに SSM エージェントをインストールする DaemonSet 設定ファイルを作成します。「エピック 」セクションの指示に従います。

```
cat << EOF > ssm_daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    k8s-app: ssm-installer
  name: ssm-installer
  namespace: kube-system
spec:
```

```
  selector:
    matchLabels:
      k8s-app: ssm-installer
  template:
    metadata:
      labels:
        k8s-app: ssm-installer
    spec:
      containers:
      - name: sleeper
        image: busybox
        command: ['sh', '-c', 'echo I keep things running! && sleep 3600']
      initContainers:
      - image: amazonlinux
        imagePullPolicy: Always
        name: ssm
        command: ["/bin/bash"]
        args: ["-c","echo '* * * * * root yum install -y https://s3.amazonaws.com/
ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm & rm -rf /etc/
cron.d/ssmstart' > /etc/cron.d/ssmstart"]
        securityContext:
          allowPrivilegeEscalation: true
        volumeMounts:
        - mountPath: /etc/cron.d
          name: cronfile
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      volumes:
      - name: cronfile
        hostPath:
          path: /etc/cron.d
          type: Directory
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      terminationGracePeriodSeconds: 30
EOF
```

エピック

kubectl のセットアップ

タスク	説明	必要なスキル
EKS クラスターにアクセスするように kubectl をインストールして設定します。	kubectlがまだインストールされておらず、Amazon EKS クラスタにアクセスするように構成されていない場合、Amazon EKS ドキュメントの「kubectl のインストール」を参照してください。	DevOps

DaemonSet をデプロイ

タスク	説明	必要なスキル
DeamonSet設定ファイルを作成します。	このパターンの前述の「コード」セクションのコードを使用して、`ssm_daemonset.yaml` という名前の DaemonSet 設定ファイルを作成します。このファイルは Amazon EKS クラスターにデプロイされます。 DaemonSet によって起動されるポッドには、メインコンテナと`init`コンテナがあります。メインコンテナには`sleep`コマンドがあります。`init`コンテナには、パス `/etc/cron.d/` に SSM Agent をインストールするための cron ジョブファイルを作成する`command`セクションが含ま	DevOps

タスク	説明	必要なスキル
	れています。cron ジョブは 1 回だけ実行され、作成されるファイルはジョブの完了後に自動的に削除されます。 init コンテナが終了すると、メインコンテナは 60 分間待機してから終了します。60 分後、新しいポッドが起動します。このポッドは、SSM エージェントがない場合はインストールするか、SSM エージェントを最新バージョンに更新します。 必要であれば、`sleep`コマンドを変更して、ポッドを1日1回再起動したり、もっと頻繁に実行したりすることができる。	
DaemonSet を Amazon EKS クラスターにデプロイします。	前のステップで作成した DaemonSet 設定ファイルを Amazon EKS クラスターにデプロイするには、次のコマンドを使用します。 `kubectl apply -f ssm_daemonset.yaml` このコマンドは、ワーカーノードでポッドを実行して SSM Agent をインストールする DaemonSet を作成します。	DevOps

関連リソース

- kubectl のインストール (Amazon EKS ドキュメント)
- セッションマネージャーのセットアップ (AWS Systems Manager ドキュメント)

prebootstrapコマンドを使用して、SSM エージェントと CloudWatch エージェントを Amazon EKS ワーカーノードにインストールします。

アッカマハデヴィ・ ハイアマス (AWS) によって作成された

概要

このパターンは、Amazon EKS クラスターの作成中に AWS Systems Manager Agent (SSM Agent) と Amazon CloudWatch エージェントをAmazon Web Services (AWS) クラウド内の Amazon Elastic Kubernetes Service (Amazon EKS) ワーカーノードにインストールするためのコードサンプルと手順を提供します。SSM エージェントと CloudWatch エージェントは、`eksctl`「設定ファイルスキーマ」 (Weaveworks ドキュメント) `preBootstrapCommands`のプロパティを使用してインストールできます。そうすれば、Amazon Elastic Compute Cloud (Amazon EC2) key pair を使用せずに SSM Agent を使用してワーカーノードに接続できます。さらに、CloudWatch エージェントを使用して Amazon EKS ワーカーノードのメモリとディスクの使用状況をモニタリングできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- macOS、Linux、または Windows で、インストールおよび設定されている「eksctl コマンドラインユーティリティ」
- macOS、Linux、または Windows で、インストールおよび設定されている「kubectl コマンドラインユーティリティ」

制約事項

- 実行時間の長いスクリプトは`preBootstrapCommands` プロパティに追加しないことをお勧めします。追加すると、スケーリングアクティビティ中にノードが Amazon EKS クラスターに参加するのが遅れるためです。代わりに「カスタム Amazon Machine Image (AMI)」を作成することをお勧めします。
- このパターンは、Amazon EC2 Linux インスタンスにのみ適用されます。

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch
- Amazon エラスティックKubernetesサービス (Amazon EKS)
- Systems Manager パラメータストア

ターゲットアーキテクチャ

次の図は、preBootstrapCommandsを使用してインストールされた SSM エージェントを使用して Amazon EKS ワーカーノードに接続するユーザーの例を示しています。

この図表は、次のワークフローを示しています：

1. ユーザは、preBootstrapCommandsプロパティを持つeksctl設定ファイルを使用してAmazon EKSクラスタを作成し、SSMエージェントとCloudWatchエージェントをインストールします。
2. スケーリングアクティビティによって後でクラスターに追加される新しいインスタンスは、プレインストールされた SSM エージェントと CloudWatch エージェントを使用して作成されます。
3. ユーザーは SSM エージェントを使用して Amazon EC2 に接続し、CloudWatch エージェントを使用してメモリとディスクの使用状況をモニタリングします。

ツール

- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムに監視します。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、AWS で Kubernetes を実行する際に役立ち、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持する必要はありません。
- AWS Systems Manager Parameter Store は、設定データ管理とシークレット管理用の安全な階層型ストレージを提供します。
- 「AWS Systems Manager Session Manager」は、AWS Systems Manager で、インタラクティブなワンクリックブラウザベースのシェルまたは AWS CLI Command Line Interface (AWS CLI) を介して管理できます。
- eksctl – これは Amazon EKS で Kubernetes クラスターを作成および管理するコマンドラインユーティリティです。
- kubectl – これはクラスター API サーバーとの通信用コマンドラインユーティリティです。

エピック

Amazon EKS クラスターを作成します。

タスク	説明	必要なスキル
CloudWatch エージェント設定ファイルを保存します。	CloudWatch エージェント設定ファイルを、Amazon EKS クラスターを作成する AWS リージョンの「AWS Systems Manager Parameter Store」に保存します。これを行うには、AWS Systems Manager Parameter Storeで「パラメータを作成」し、パラメータの名前 (例：`AmazonCloudwatch-linux`) を書き留めます。 詳細については、このパターンの「追加情報」セクションにある「CloudWatch エージェント設定ファイルの例」を参照してください。	DevOps エンジニア
eksctl 設定ファイルとクラスターを作成します。	1. CloudWatch エージェントと SSM エージェントのインストール手順を含む`eksctl`設定ファイルを作成します。詳細については、このパターンの「追加情報」セクションにある「eksctl 設定ファイルのコード例」を参照してください。 2. `eksctl create cluster -f cluster.y`	AWS DevOps

タスク	説明	必要なスキル
	`aml` コマンドを実行してクラスターを作成します。	

SSM エージェントと CloudWatch エージェントが動作することを確認する

タスク	説明	必要なスキル
SSM Agent をテストします。	SSH を使用して、AWS Systems Manager ドキュメントの「セッションの開始」で説明されている方法のいずれかを使用して Amazon EKS クラスターノードに接続します。	AWS DevOps
CloudWatch エージェントをテストします。	CloudWatch コンソールを使用して CloudWatch エージェントを検証します。 1. AWS マネジメントコンソールにサインインし、CloudFront コンソール を開きます。 2. ナビゲーションペインで、[メトリクス] を展開し、[すべてのメトリクス] を選択します。 3. 「ブラウズ」 タブの検索ボックスに「CWAgent メトリクス」を入力して選択すると、 メモリとディスクのメトリクスが表示されます。	AWS DevOps

関連リソース

- サーバーにエージェントをインストールして実行する (Amazon CloudWatch ドキュメント)
- Systems Manager パラメータを作成する (コンソール) (AWS Systems Manager ドキュメント)
- CloudWatch エージェント設定ファイルを作成する (Amazon CloudWatch ドキュメント)
- セッションを開始します (AWS Systems Manager のドキュメント)
- セッションを開始する (Amazon EC2 コンソール) (AWS Systems Manager ドキュメント)

追加情報

CloudWatch エージェント設定ファイルの例

次の例では、CloudWatch エージェントは Amazon Linux インスタンスのディスクとメモリの使用状況を監視するように設定されています。

```
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "cwagent"
    },
    "metrics": {
        "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
            "disk": {
                "measurement": [
                    "used_percent"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            },
            "mem": {
                "measurement": [
                    "mem_used_percent"
```

```
                ],
                "metrics_collection_interval": 60
            }
        }
    }
}
```

eksctl 設定ファイルの例

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: test
  region: us-east-2
  version: "1.24"
managedNodeGroups:
  - name: test
    minSize: 2
    maxSize: 4
    desiredCapacity: 2
    volumeSize: 20
    instanceType: t3.medium
    preBootstrapCommands:
    - sudo yum install amazon-ssm-agent -y
    - sudo systemctl enable amazon-ssm-agent
    - sudo systemctl start amazon-ssm-agent
    - sudo yum install amazon-cloudwatch-agent -y
    - sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c ssm:AmazonCloudwatch-linux
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
        - arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

その他のコードの詳細

- preBootstrapCommandsプロパティの最後の行には、AmazonCloudwatch-linuxはAWS System Manager Parameter Storeで作成されたパラメータ名です。Amazon EKS クラスターを作成したときと同じ AWS リージョンのパラメータストアにAmazonCloudwatch-linuxを含める

必要があります。ファイルパスを指定することもできますが、自動化と再利用を容易にするために Systems Manager を使用することをお勧めします。

- eksctl設定ファイルでpreBootstrapCommandsを使用すると、AWS マネジメントコンソールに 2 つの起動テンプレートが表示されます。最初の起動テンプレートには、preBootstrapCommandsで指定されているコマンドが含まれています。2 番目のテンプレートには、preBootstrapCommandsで指定されているコマンドとデフォルトの Amazon EKS ユーザーデータが含まれます。このデータは、ノードをクラスターに参加させるために必要です。ノードグループの Auto Scaling グループは、このユーザーデータを使用して新しいインスタンスを起動します。
- eksctl設定ファイルのiam属性を使用する場合は、デフォルトの Amazon EKS ポリシーと、添付AWS Identity and Access Management (IAM) ポリシーに必要な追加ポリシーを一覧表示する必要があります。「eksctl 設定ファイルとクラスターの作成」ステップのコードスニペットには、CloudWatchAgentServerPolicyとAmazonSSMMangedInstanceCoreは、CloudWatch エージェントとSSMエージェントが期待通りに動作することを確認するために追加されたポリシーで
す。AmazonEKSWorkerNodePolicy、AmazonEKS_CNI_Policy、AmazonEC2ContainerRegistryR
リシーは Amazon EKS クラスターが正しく機能するために必要な必須ポリシーです。

Amazon EKS Auto Mode を有効にするときに NGINX Ingress Controller を移行する

作成者: Olawale Olaleye (AWS) と Shamanth Devagari (AWS)

概要

Amazon Elastic Kubernetes Service (Amazon EKS) の EKS Auto Mode を使用すると、Kubernetes クラスターでワークロードを実行する際の運用オーバーヘッドを削減できます。このモードでは AWS 、 がユーザーに代わってインフラストラクチャを設定および管理することもできます。既存のクラスターで EKS Auto Mode を有効にする場合は、NGINX Ingress Controller 設定の移行を慎重に計画する必要があります。これは、Network Load Balancer を直接転送できないためです。

既存の Amazon EKS クラスターで EKS Auto Mode を有効にすると、Blue/Green デプロイ戦略を使用して NGINX Ingress Controller インスタンスを移行できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Kubernetes バージョン 1.29 以降を実行している Amazon EKS クラスター
- 最小バージョンを実行する Amazon EKS アドオン
- kubectl の最新バージョン
- 既存の NGINX Ingress Controller インスタンス
- （オプション) DNS ベースのトラフィックシフト用の Amazon Route 53 のホストゾーン

アーキテクチャ

ブルー/グリーンデプロイは、2 つの異なる同一の環境を作成するデプロイ戦略です。ブルー/グリーンデプロイでは、ダウンタイムがほぼゼロのリリースとロールバック機能が提供されます。基本的な考え方は、アプリケーションの異なるバージョンを実行している 2 つの同一の環境間でトラフィックを移行することです。

次の図は、EKS Auto Mode を有効にする際の 2 つの異なる NGINX Ingress Controller インスタンスからの Network Load Balancer の移行を示しています。Blue/Green デプロイを使用して、2 つの Network Load Balancer 間でトラフィックをシフトします。

元の名前空間は青の名前空間です。これは、EKS Auto Mode を有効にする前に、元の NGINX Ingress Controller サービスとインスタンスが実行される場所です。元のサービスとインスタンスは、Route 53 で設定された DNS 名を持つ Network Load Balancer に接続します。AWS Load Balancer Controller は、この Network Load Balancer をターゲット仮想プライベートクラウド (VPC) にデプロイしました。

この図は、ブルー/グリーンデプロイ用の環境を設定するための次のワークフローを示しています。

1. 別の NGINX Ingress Controller インスタンスを緑色の名前空間にインストールして設定します。
2. Route 53 で、新しい Network Load Balancer の DNS 名を設定します。

ツール

AWS のサービス

- Amazon Elastic Kubernetes Service (Amazon EKS) を使用すると、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行できます。
- 受信したアプリケーションまたはネットワークトラフィックを複数のターゲットに分散するには、Elastic Load Balancing を使用します。例えば、1 つまたは複数のアベイラビリティーゾーンの Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレスにトラフィックを分散できます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

その他のツール

- Helm は、Kubernetes クラスターへのアプリケーションのインストールと管理に役立つ Kubernetes のオープンソースパッケージマネージャーです。
- kubectlは、Kubernetes クラスターに対してコマンドを実行するためのコマンドラインインターフェイスです。
- NGINX Ingress Controller は、Kubernetes アプリケーションとサービスをリクエスト処理、認証、セルフサービスのカスタムリソース、デバッグに接続します。

エピック

既存の環境を確認する

タスク	説明	必要なスキル

元の NGINX Ingress Controller インスタンスが動作していることを確認します。	次のコマンドを入力して、ingress-nginx 名前空間内のリソースが動作していることを確認します。NGINX Ingress Controller を別の名前空間にデプロイした場合は、このコマンドで名前空間名を更新します。
	`kubectl get all -n ingress-nginx`
	出力で、NGINX Ingress Controller ポッドが実行中の状態であることを確認します。以下は、その出力例です。
	`NAME READY STATUS RESTARTS AGE pod/ingress-nginx-admission-create-xqn9d 0/1 Completed 0 88m pod/ingress-nginx-admission-patch-lhk4j 0/1 Completed 1 88m pod/ingress-nginx-controller-68f68f859-xrz74 1/1 Running 2 (10m ago) 72m`
		DevOps エンジニア

タスク	説明	必要なスキル
	NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE service/ingress-nginx-controller LoadBalancer 10.100.67.255 k8s-ingressn-ingressn-abcdefg-12345.elb.eu-west-1.amazonaws.com 80:30330/TCP,443:31462/TCP 88m service/ingress-nginx-controller-admission ClusterIP 10.100.201.176 <none> 443/TCP 88m NAME READY UP-TO-DATE AVAILABLE AGE deployment.apps/ingress-nginx-controller 1/1 1 1 88m NAME DESIRED CURRENT READY AGE	

タスク	説明	必要なスキル
	replicaset.apps/ ingress-nginx-co ntroller-68f68f859 1 1 1 72m replicaset.apps/ ingress-nginx-co ntroller-d8c96cf68 0 0 0 88m NAME STATUS COMPLETIONS DURATION AGE job.batch/ingress- nginx-admission-create Complete 1/1 4s 88m job.batch/ingress- nginx-admission-patch Complete 1/1 5s 88m	

サンプルの HTTPd ワークロードをデプロイして NGINX Ingress Controller を使用する

タスク	説明	必要なスキル
Kubernetes リソースを作成します。	次のコマンドを入力して、サンプルの Kubernetes デプロイ、サービス、イングレスを作成します。 `kubectl create deployment demo --image=httpd --port=80`	DevOps エンジニア

タスク	説明	必要なスキル

	`kubectl expose deployment demo`
	
	`kubectl create ingress demo --class=nginx \ --rule nginxautomode.local.dev/=demo:80`	

タスク	説明	必要なスキル

デプロイされたリソースを確認します。	次のコマンドを入力して、デプロイされたリソースのリストを表示します。
	`kubectl get all,ingress`
	出力で、サンプル HTTPd ポッドが実行中の状態であることを確認します。以下は、その出力例です。
	`NAME READY STATUS RESTARTS AGE pod/demo-7d94f8cb4f-q68wc 1/1 Running 0 59m`
	`NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE service/demo ClusterIP 10.100.78.155 <none> 80/TCP 59m service/kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 117m`
	`NAME READY UP-TO-DATE AVAILABLE AGE deployment.apps/demo 1/1 1 1 59m`
		DevOps エンジニア

タスク	説明	必要なスキル
	NAME DESIRED CURRENT READY AGE replicaset.apps/ demo-7d94f8cb4f 1 1 1 59m NAME CLASS HOSTS ADDRESS PORTS AGE ingress.networ king.k8s.io/demo nginx nginxauto mode.local.dev k8s-ingre ssn-ingressn-abcde fg-12345.elb.eu-we st-1.amazonaws.com 80 56m	

タスク	説明	必要なスキル

サービスに到達できることを確認します。	次のコマンドを入力して、サービスが Network Load Balancer の DNS 名を介して到達可能であることを確認します。
	`curl -H "Host: nginxautomode.local.dev" http://k8s-ingressn-ingressn-abcdefg-12345.elb.eu-west-1.amazonaws.com`
	予想される出力は次のとおりです。
	`<html><body><h1>It works!</h1></body></html>`
		DevOps エンジニア

タスク	説明	必要なスキル

(オプション) DNS レコードを作成します。	1. Amazon Route 53 コンソールを使用してレコードを作成する (Route 53 ドキュメント) の手順に従って、設定されたドメインの DNS レコードを作成します。
	2. 次のコマンドを入力して、サービスが設定されたドメイン名を介して到達可能であることを確認します。
	`curl "http://nginxautomode.local.dev/?[1-5]"`
	予想される出力は次のとおりです。
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
		DevOps エンジニア、AWS DevOps

既存のクラスターで EKS 自動モードを有効にする

タスク	説明	必要なスキル
EKS 自動モードを有効にします。	既存のクラスターで EKS 自動モードを有効にする (Amazon EKS ドキュメント) の手順に従います。	AWS DevOps

新しい NGINX Ingress Controller をインストールする

タスク	説明	必要なスキル

新しい NGINX Ingress Controller インスタンスを設定します。	1. deploy.yaml テンプレートをダウンロードします。
	2. 任意のエディタで deploy.yaml テンプレートを開きます。
	3. `kind: Namespace` セクションに、 などの名前空間の一意の名前を入力しますingress-nginx-v2 。
	`apiVersion: v1`
	`kind: Namespace`
	`metadata:`
	`labels:`
	`app.kuber`
	`netes.io/instance:`
	`ingress-nginx`
	`app.kuber`
	`netes.io/name:`
	`ingress-nginx`
	`name: ingress-n`
	`ginx-v2`
		DevOps エンジニア

タスク	説明	必要なスキル

	4. 各セクションで、namespace 値を新しい名前に更新します。
	5. [kind: Deployment] セクションで、次の操作を行います。
	a. など--controller-class 、 の一意の値を入力しますk8s.io/ingress-nginx-v2 。
	b. など--ingress-class、 の一意の値を入力しますnginx-v2。
	`apiVersion: apps/v1`
	`kind: Deployment`
	`  name: ingress-nginx-controller`
	`  namespace: ingress-nginx-v2`
	`...`
	`    spec:`
	`      containers:`
	`      - args:`
	`        - /nginx-ingress-controller`
	`        - --publish-service=$(POD_NAMESPACE)/ingress-nginx-controller`
	`        - --election-id=ingress-nginx-leader`
	`        - --controller-class=k8s.io/ingress-nginx-v2`
	`        - --ingress-class=nginx-v2`	

タスク	説明	必要なスキル

	6. kind: IngressClass セクションで、前のセクションで使用--ingress-class したものと同じ値を --controller-class と に入力します。
	
	`apiVersion: networking.k8s.io/v1`
	`kind: IngressClass`
	`metadata:`
	`  labels:`
	`  app.kubernetes.io/component: controller`
	`  app.kubernetes.io/instance: ingress-nginx`
	`  app.kubernetes.io/name: ingress-nginx`
	`  app.kubernetes.io/part-of: ingress-nginx`
	`  app.kubernetes.io/version: 1.12.0`
	`  name: nginx-v2`
	`spec:`
	`  controller: k8s.io/ingress-nginx-v2`
	
	7. 次のセクションで、loadBalancerClass: eks.amazonaws.com/nlb を追加して NGINX Ingress Controller インスタンスの Network Load	

タスク	説明	必要なスキル

	Balancer をプロビジョニングします。
	
	`apiVersion: v1`
	`kind: Service`
	`metadata:`
	`  name: ingress-nginx-controller`
	`  namespace: ingress-nginx-v2`
	`spec:`
	`...`
	`  selector:`
	`  app.kubernetes.io/component: controller`
	`  app.kubernetes.io/instance: ingress-nginx`
	`  app.kubernetes.io/name: ingress-nginx`
	`  type: LoadBalancer`
	`  loadBalancerClass: eks.amazonaws.com/nlb`
	
	8. deploy.yaml テンプレートを保存して閉じます。	

新しい NGINX Instance Controller インスタンスをデプロイします。	次のコマンドを入力して、変更されたマニフェストファイルを適用します。 `kubectl apply -f deploy.yaml`	DevOps エンジニア

タスク	説明	必要なスキル

デプロイが成功したことを確認します。	次のコマンドを入力して、ingress-nginx-v2 名前空間内のリソースが動作していることを確認します。
	`kubectl get all -n ingress-nginx-v2`
	出力で、NGINX Ingress Controller ポッドが実行中の状態であることを確認します。以下は、その出力例です。
	`NAME READY STATUS RESTARTS AGE pod/ingress-nginx-admission-create-7shrj 0/1 Completed 0 24s pod/ingress-nginx-admission-patch-vkxr5 0/1 Completed 1 24s pod/ingress-nginx-controller-757bfcbc6d-4fw52 1/1 Running 0 24s NAME TYPE CLUSTER-IP EXTERNAL-IP`
		DevOps エンジニア

タスク	説明	必要なスキル
	PORT(S) AGE service/ingress- nginx-controller LoadBalancer 10.100.208.114 k8s-ingressn-ingre ssn-2e5e37fab6-848 337cd9c9d520f.elb.eu- west-1.amazonaws.com 80:31469/TCP,443:3 0658/TCP 24s service/ingress- nginx-controller- admission ClusterIP 10.100.150.114 <none> 443/TCP 24s NAME READY UP-TO-DATE AVAILABLE AGE deployment.apps/ ingress-nginx-co ntroller 1/1 1 1 24s NAME DESIRED CURRENT READY AGE replicaset.apps/ ingress-nginx-co ntroller-757bfcbc6d 1 1 1 24s	

タスク	説明	必要なスキル
	NAME STATUS COMPLETIONS DURATION AGE job.batch/ingress-nginx-admission-create Complete 1/1 4s 24s job.batch/ingress-nginx-admission-patch Complete 1/1 5s 24s	
サンプル HTTPd ワークロードの新しい進入を作成します。	既存のサンプル HTTPd ワークロードの新しい進入を作成するには、次のコマンドを入力します。 `kubectl create ingress demo-new --class=nginx-v2 \ --rule nginxautomode.local.dev/=demo:80`	DevOps エンジニア

タスク	説明	必要なスキル

新しい進入が機能することを確認します。	次のコマンドを入力して、新しい進入が機能することを確認します。
	`curl -H "Host: nginxautomode.local.dev" k8s-ingressn-ingressn-2e5e37fab6-848337cd9c9d520f.elb.eu-west-1.amazonaws.com`
	予想される出力は次のとおりです。
	`<html><body><h1>It works!</h1></body></html>`
		DevOps エンジニア

カットオーバー

タスク	説明	必要なスキル
新しい名前空間にカットオーバーします。	1. （オプション）「レコードの編集」(Route 53 ドキュメント) の指示に従って DNS レコードを更新します。 2. 新しい NGINX Ingress Controller インスタンスが正常に動作していることを確認したら、元のインスタンスを削除します。 3. セルフマネージド AWS Load Balancerコントロー	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	ラーを削除します。手順については、「廃止された ALB Ingress Controller からアプリケーションを移行する」(Amazon EKS ドキュメント) を参照してください。 4. マネージド型ノードグループをドレインします。手順については、「ノードグループの削除とドレイン」(eksctl ドキュメント) を参照してください。	

タスク	説明	必要なスキル

2 つの進入を確認します。	次のコマンドを入力して、サンプルの HTTPd ワークロード用に作成された 2 つの進入を確認します。
	`kubectl get ingress`
	以下は、その出力例です。
	`NAME  CLASS  HOSTS  ADDRESS  PORTS  AGE`
	`demo  nginx  nginxautomode.local.dev  k8s-ingressn-ingressn-abcdefg-12345.elb.eu-west-1.amazonaws.com  80  95m`
	`demo-new  nginx-v2  nginxautomode.local.dev  k8s-ingressn-ingressn-2e5e37fab6-848337cd9c9d520f.elb.eu-west-1.amazonaws.com  80  33s`
		DevOps エンジニア

関連リソース

- 既存のクラスターで EKS Auto Mode を有効にする (Amazon EKS ドキュメント)

- Amazon EKS の Kubernetes サービスコントローラーによって作成されたロードバランサーのトラブルシューティング (AWS re:Post ナレッジセンター)
- NGINX Ingress Controller (NGINX ドキュメント)

コンテナワークロードを Azure Red Hat OpenShift (ARO) から Red Hat OpenShift Service on AWS (ROSA) に移行する

作成者: Naveen Ramasamy (AWS)、Gileesh Sreekantan (AWS)、Srikanth Rangavajhala (AWS)

概要

このパターンでは、コンテナワークロードを Azure Red Hat OpenShift (ARO) から Red Hat OpenShift Service on AWS (ROSA) に移行するためのstep-by-stepの手順を示します。ROSA は、Red Hat が と共同で提供するマネージド Kubernetes サービスです AWS。これは、Kubernetes プラットフォームを使用してコンテナ化されたアプリケーションをデプロイ、管理、スケーリングするのに役立ちます。また、Red Hat の Kubernetes に関する専門知識と AWS クラウド インフラストラクチャの両方を活用できます。

コンテナワークロードを ARO、他のクラウド、またはオンプレミスから ROSA に移行するには、アプリケーション、設定、データをあるプラットフォームから別のプラットフォームに転送する必要があります。このパターンは、 AWS クラウド サービス、セキュリティ、コスト効率を最適化しながら、スムーズな移行を確保するのに役立ちます。ワークロードを ROSA クラスターに移行するための 2 つの方法、CI/CD と Migration Toolkit for Containers (MTC) について説明します。

このパターンは、両方の方法を対象としています。選択する方法は、移行プロセスの複雑さと確実性によって異なります。アプリケーションの状態を完全に制御でき、パイプラインを通じて一貫したセットアップを保証できる場合は、CI/CD メソッドを使用することをお勧めします。ただし、アプリケーションの状態が不確実性、予期しない変更、または複雑なエコシステムを伴う場合は、信頼性が高く制御されたパスとして MTC を使用して、アプリケーションとそのデータを新しいクラスターに移行することをお勧めします。2 つの方法の詳細な比較については、「追加情報」セクションを参照してください。

ROSA に移行するメリット：

- ROSA はネイティブサービス AWS として とシームレスに統合されます。経由で簡単にアクセス AWS Management Console でき、1 つの で請求されます AWS アカウント。他の との完全な互換性を提供し AWS のサービス 、 AWS と Red Hat の両方からの共同サポートを提供します。
- ROSA は、ハイブリッドおよびマルチクラウドのデプロイをサポートしています。これにより、アプリケーションはオンプレミスのデータセンターや複数のクラウド環境で一貫して実行できます。

- ROSA は、Red Hat のセキュリティ重視のメリットを享受し、ロールベースのアクセスコントロール (RBAC)、イメージスキャン、脆弱性評価などの機能を提供して、安全なコンテナ環境を確保します。
- ROSA は、アプリケーションを簡単にスケールするように設計されており、高可用性オプションを提供します。これにより、信頼性を維持しながら、必要に応じてアプリケーションを拡張できます。
- ROSA は、手動セットアップおよび管理方法と比較して、Kubernetes クラスターのデプロイを自動化および簡素化します。これにより、開発とデプロイのプロセスが加速されます。
- ROSA は AWS クラウド サービスのメリットを享受し、データベースサービス、ストレージソリューション、セキュリティサービスなどの AWS サービスとシームレスに統合できます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS のサービス その ROSA 用に設定されたアクセス許可は、 の機能を提供するために に依存します。詳細については、ROSA ドキュメントの「前提条件」を参照してください。
- ROSA コンソールで ROSA が有効になっています。手順については、ROSA ドキュメントを参照してください。
- ROSA クラスターがインストールされ、設定されています。詳細については、ROSA ドキュメントの「ROSA の使用を開始する」を参照してください。ROSA クラスターをセットアップするためのさまざまな方法を理解するには、 AWS 「 規範ガイダンスガイド」の「ROSA 実装戦略」を参照してください。
- オンプレミスネットワークから (推奨) または AWS Direct Connect (AWS Virtual Private NetworkAWS VPN) AWS を介して に確立されたネットワーク接続。
- 、OpenShift CLI () クライアント`aws client`、ROSA クライアント、Git バイナリなどのツールをインストールするための Amazon Elastic Compute Cloud (Amazon EC2oc) インスタンスまたは別の仮想サーバー。

CI/CD メソッドのその他の前提条件：

- 新しいパイプラインの作成、ステージの追加、OpenShift クラスターの追加、ビルドの実行を行うアクセス許可を持つオンプレミス Jenkins サーバーへのアクセス。

- アプリケーションソースコードが維持されている Git リポジトリへのアクセス。新しい Git ブランチを作成し、新しいブランチへのコミットを実行するアクセス許可が付与されます。

MTC メソッドのその他の前提条件：

- Amazon Simple Storage Service (Amazon S3) バケット。レプリケーションリポジトリとして使用されます。
- ソース ARO クラスターへの管理アクセス。これは、MTC 接続をセットアップするために必要です。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ROSA には、パブリック、プライベート、および .PrivateLink の 3 つのネットワークデプロイパターンがありますAWS PrivateLink。Red Hat サイト信頼性エンジニアリング (SRE) チームは、既存の VPC 内のクラスターの PrivateLink エンドポイントに接続されたプライベートサブネットを使用してクラスターを管理できます。

PrivateLinkoption を選択すると、最も安全な設定が提供されます。そのため、機密性の高いワークロードや厳格なコンプライアンス要件には、このポリシーを使用することをお勧めします。パブリックおよびプライベートネットワークデプロイオプションの詳細については、Red Hat OpenShift ドキュメントを参照してください。

> **Important**
>
> PrivateLink クラスターは、インストール時にのみ作成できます。インストール後に PrivateLink を使用するようにクラスターを変更することはできません。

次の図は、 がオンプレミス環境と ARO 環境への接続 AWS Direct Connect に使用する ROSA クラスターの PrivateLink アーキテクチャを示しています。

AWS ROSA への アクセス許可

ROSA へのアクセス AWS 許可については、存続期間の短い動的トークンで AWS Security Token Service （ AWS STS) を使用することをお勧めします。この方法では、最小特権の事前定義されたロールとポリシーを使用して、 で運用するための最小限のアクセス許可を ROSA に付与し AWS アカウント、ROSA のインストール、コントロールプレーン、コンピューティング機能をサポートします。

CI/CD パイプラインの再デプロイ

CI/CD パイプラインの再デプロイは、成熟した CI/CD パイプラインを持つユーザーに推奨される方法です。このオプションを選択すると、任意の DevOps デプロイ戦略を使用して、アプリケーションの負荷を ROSA のデプロイに徐々に移行できます。

> **Note**
>
> このパターンは、オンプレミスの Git、JFrog Artifactory、および Jenkins パイプラインがある一般的なユースケースを想定しています。このアプローチでは、オンプレミスネットワークから AWS までのネットワーク接続を確立し AWS Direct Connect、エピックセクションの指示に従う前に ROSA クラスターを設定する必要があります。詳細については、「前提条件」セクションを参照してください。

次の図は、この方法のワークフローを示しています。

MTC メソッド

Migration Toolkit for Containers (MTC) を使用して、コンテナ化されたワークロードを ARO から ROSA など、さまざまな Kubernetes 環境間で移行できます。MTC は、いくつかの主要なタスクを自動化し、移行ライフサイクルを管理するための包括的なフレームワークを提供することで、移行プロセスを簡素化します。

次の図は、この方法のワークフローを示しています。

ツール

AWS のサービス

- AWS DataSync は、 AWS ストレージサービスとの間でファイルまたはオブジェクトデータを移動するのに役立つオンラインデータ転送および検出サービスです。
- AWS Direct Connect は、標準のイーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、ネットワークパスでインターネットサービスプロバイダーをバイパス AWS のサービス しながら、仮想インターフェイスをパブリックに直接作成できます。
- AWS PrivateLink は、仮想プライベートクラウド (VPCs) から VPC 外のサービスへの一方向のプライベート接続を作成するのに役立ちます。
- Red Hat OpenShift Service on AWS (ROSA) は、Red Hat OpenShift ユーザーがコンテナ化されたアプリケーションを構築、スケーリング、管理するのに役立つマネージドサービスです AWS。
- AWS Security Token Service （ AWS STS ） は、ユーザーの権限が制限された一時的な認証情報をリクエストするのに役立ちます。

その他のツール

- Migration Toolkit for Containers (MTC) には、コンテナ化されたアプリケーションを ARO から ROSA に移行するためのコンソールと API が用意されています。

ベストプラクティス

- 回復力を高め、セキュリティコンプライアンスワークロードがある場合は、PrivateLink を使用するマルチ AZ ROSA クラスターを設定します。詳細については、ROSA ドキュメントを参照してください。

> **Note**
>
> PrivateLink は、インストール後に設定することはできません。

- レプリケーションリポジトリに使用する S3 バケットはパブリックにしないでください。適切な S3 バケットポリシーを使用してアクセスを制限します。
- MTC メソッドを選択した場合は、ステージ移行オプションを使用して、カットオーバー中のダウンタイムウィンドウを短縮します。

- ROSA クラスターをプロビジョニングする前と後に、サービスクォータを確認します。必要に応じて、要件に従ってクォータの引き上げをリクエストします。詳細については、Service Quotas ドキュメントを参照してください。
- ROSA セキュリティガイドラインを確認し、セキュリティのベストプラクティスを実装します。
- インストール後にデフォルトのクラスター管理者を削除することをお勧めします。詳細については、Red Hat OpenShift のドキュメントを参照してください。
- マシンプールの自動スケーリングを使用して、ROSA クラスター内の未使用のワーカーノードをスケールダウンし、コストを最適化します。詳細については、「ROSA ワークショップ」を参照してください。
- OpenShift Container Platform 用の Red Hat Cost Management サービスを使用して、クラウドとコンテナのコストをよりよく理解し、追跡します。詳細については、「ROSA ワークショップ」を参照してください。
- を使用して、ROSA クラスターインフラストラクチャサービスとアプリケーションをモニタリングおよび監査します AWS のサービス。詳細については、「ROSA ワークショップ」を参照してください。

エピック

オプション 1: CI/CD パイプラインを使用する

タスク	説明	必要なスキル
新しい ROSA クラスターを Jenkins に追加します。	1. Jenkins コンソールで、Jenkins の管理、システムの設定を選択します。 2. Jenkins の管理ページの OpenShift プラグインセクションで、新しいクラスターの追加を選択します。 3. ROSA で認証するために、クラスター名、API サーバー URL、トークン情報などの必要な情報を入力します。	AWS 管理者、AWS システム管理者、AWS DevOps

タスク	説明	必要なスキル
oc クライアントを Jenkins ノードに追加します。	1. Jenkins コンソールで、Jenkins の管理、グローバルツール設定を選択します。 2. OpenShift クライアントツールセクションで、ROSA クラスターバージョンと同じ OpenShift CLI (oc) クライアントバージョンをインストールします。	AWS 管理者、AWS システム管理者、AWS DevOps
新しい Git ブランチを作成します。	の Git リポジトリに新しいブランチを作成しますrosa-dev。このブランチは、ROSA のコードまたは設定パラメータの変更を既存のパイプラインから分離します。	AWS DevOps
ROSA のイメージにタグを付けます。	ビルドステージでは、別のタグを使用して、ROSA パイプラインからビルドされたイメージを識別します。	AWS 管理者、AWS システム管理者、AWS DevOps
パイプラインを作成する	既存のパイプラインに似た新しい Jenkins パイプラインを作成します。このパイプラインでは、前に作成した rosa-dev Git ブランチを使用し、既存のパイプラインと同じ Git チェックアウト、コードスキャン、ビルドステージを必ず含めてください。	AWS 管理者、AWS システム管理者、AWS DevOps

タスク	説明	必要なスキル
ROSA デプロイステージを追加します。	新しいパイプラインで、ステージを追加して ROSA クラスターにデプロイし、Jenkins グローバル設定に追加した ROSA クラスターを参照します。	AWS 管理者、AWS DevOps、AWS システム管理者
新しいビルドを開始します。	Jenkins で、パイプラインを選択して今すぐビルドを選択するか、Git の`rosa-dev`ブランチに変更をコミットして新しいビルドを開始します。	AWS 管理者、AWS DevOps、AWS システム管理者
デプロイメントを確認する	oc コマンドまたは ROSA コンソールを使用して、アプリケーションがターゲット ROSA クラスターにデプロイされていることを確認します。	AWS 管理者、AWS DevOps、AWS システム管理者
ターゲットクラスターにデータをコピーします。	ステートフルワークロードの場合は、 AWS DataSync または rsync などのオープンソースユーティリティを使用して、ソースクラスターからターゲットクラスターにデータをコピーするか、MTC メソッドを使用できます。詳細については、AWS DataSync のドキュメントを参照してください。	AWS 管理者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
アプリケーションをテストする	1. oc route コマンドまたは ROSA コンソールを使用して、アプリケーションのルート URL を取得します。 https://console.aws.amazon.com/rosa 2. ルート URL を使用してアプリケーションをテストします。	AWS 管理者、AWS DevOps、AWS システム管理者
カットオーバー。	テストが成功した場合は、適切な Amazon Route 53 ポリシーを使用して、トラフィックを ARO ホストアプリケーションから ROSA ホストアプリケーションに移動します。このステップを完了すると、アプリケーションのワークロードは ROSA クラスターに完全に移行します。	AWS 管理者、AWS システム管理者

オプション 2: MTC を使用する

タスク	説明	必要なスキル
MTC 演算子をインストールします。	ARO クラスターと ROSA クラスターの両方に MTC 演算子をインストールします。 1. ARO または ROSA コンソールで、オペレータ、OperatorHubページに移動します。	AWS 管理者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
	2. キーワードでフィルタリング ボックスに、MTC を検索または入力します。 3. MTC 演算子を選択して追加情報を表示します。 4. 演算子に関する情報を読み終わったら、インストールを選択します。	
レプリケーションリポジトリへのネットワークトラフィックを設定します。	プロキシサーバーを使用している場合は、レプリケーションリポジトリとクラスター間のネットワークトラフィックを許可するように設定します。レプリケーションリポジトリは、MTC がデータの移行に使用する中間ストレージオブジェクトです。ソースクラスターとターゲットクラスターは、移行中にレプリケーションリポジトリへのネットワークアクセスを持っている必要があります。	AWS 管理者、AWS DevOps、AWS システム管理者
ソースクラスターを MTC に追加します。	MTC ウェブコンソールで、ARO ソースクラスターを追加します。	AWS 管理者、AWS DevOps、AWS システム管理者
Amazon S3 をレプリケーションリポジトリとして追加します。	MTC ウェブコンソールで、Amazon S3 バケット (「前提条件」を参照) をレプリケーションリポジトリとして追加します。	AWS 管理者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
移行計画を作成します。	MTC ウェブコンソールで、移行計画を作成し、データ転送タイプをコピーとして指定します。これにより、MTC はソース (ARO) クラスターから S3 バケットにデータをコピーし、バケットからターゲット (ROSA) クラスターにコピーするように指示されます。	AWS 管理者、AWS DevOps、AWS システム管理者

タスク	説明	必要なスキル
移行計画を実行します。	ステージまたはカットオーバーオプションを使用して移行計画を実行します。 • Stage を選択すると、アプリケーションを停止せずにターゲットクラスターにデータをコピーできます。複数のステージ移行を実行して、カットオーバー移行の前にほとんどのデータをコピーできます。これにより、カットオーバー移行の期間が短縮されます。 • カットオーバー を選択して、リソースをターゲットクラスターに移動しながら、ソースクラスターでアプリケーションを停止させます。 アプリケーションの停止を防ぐには、移行時のチェックボックスでソースクラスターの Halt トランザクションをクリアします。	AWS 管理者、AWS DevOps、AWS システム管理者

トラブルシューティング

問題	ソリューション
接続の問題	コンテナワークロードを ARO から ROSA に移行すると、移行を成功させるために解決する必要がある接続の問題が発生する可能性がありま

問題	ソリューション
	す。移行中のこれらの接続の問題 (この表を参照) に対処するには、綿密な計画、ネットワークチームとセキュリティチームとの調整、徹底的なテストが不可欠です。段階的な移行戦略を実装し、各ステップで接続を検証すると、潜在的な中断を最小限に抑え、ARO から ROSA へのスムーズな移行を確保できます。
ネットワーク設定の違い	ARO と ROSA では、仮想ネットワーク (VNet) 設定、サブネット、ネットワークポリシーなど、ネットワーク設定にバリエーションがある場合があります。サービス間で適切に通信するには、ネットワーク設定が 2 つのプラットフォーム間で一致していることを確認してください。
セキュリティグループとファイアウォールのルール	ROSA と ARO では、デフォルトのセキュリティグループとファイアウォールの設定が異なる場合があります。これらのルールを調整して更新し、移行中にコンテナとサービス間の接続を維持するために必要なトラフィックを許可してください。
IP アドレスと DNS の変更	ワークロードを移行すると、IP アドレスと DNS 名が変更される可能性があります。静的 IPsまたは特定の DNS 名に依存するアプリケーションを再設定します。
外部サービスアクセス	アプリケーションがデータベースや APIs などの外部サービスに依存している場合は、接続設定を更新して、ROSA からの新しいサービスと通信できるようにする必要がある場合があります。

問題	ソリューション
Azure プライベートリンクの設定	ARO で Azure プライベートリンクまたはプライベートエンドポイントサービスを使用する場合は、リソース間のプライベート接続を確保するために、ROSA で同等の機能を設定する必要があります。
AWS VPN または AWS Direct Connect のセットアップ	オンプレミスネットワークと ARO の間に既存の AWS VPN または AWS Direct Connect 接続がある場合は、ローカルリソースとの中断のない通信のために、ROSA と同様の接続を確立する必要があります。
Ingress and Load Balancer の設定	Ingress Controller とロードバランサーの設定は、ARO と ROSA で異なる場合があります。これらの設定を再構成して、 サービスへの外部アクセスを維持します。
証明書と TLS の処理	アプリケーションが SSL 証明書または TLS を使用している場合は、証明書が有効で、ROSA で正しく設定されていることを確認します。
コンテナレジストリへのアクセス	コンテナが外部コンテナレジストリでホストされている場合は、ROSA の適切な認証とアクセス許可を設定します。
モニタリングとログ記録	モニタリングとログ記録の設定を更新して ROSA の新しいインフラストラクチャを反映し、コンテナのヘルスとパフォーマンスを効果的にモニタリングし続けることができます。

関連リソース

AWS リファレンス

- とは Red Hat OpenShift Service on AWS (ROSA ドキュメント)
- ROSA の使用を開始する (ROSA ドキュメント)

- Red Hat OpenShift Service on AWS 実装戦略 (AWS 規範ガイダンス)
- Red Hat OpenShift Service on AWS Now GA (AWS ブログ記事)
- ROSA ワークショップ
- ROSA に関するよくある質問
- ROSA ワークショップに関するよくある質問
- ROSA の料金

Red Hat OpenShift ドキュメント

- にクラスターをすばやくインストールする AWS
- 制限されたネットワーク AWS での へのクラスターのインストール
- 既存の VPC AWS への へのクラスターのインストール
- CloudFormation テンプレート AWS を使用して のユーザープロビジョニングインフラストラクチャにクラスターをインストールする
- ユーザーがプロビジョニングしたインフラストラクチャを使用して制限されたネットワーク AWS にクラスターをインストールする
- カスタマイズ AWS を使用した へのクラスターのインストール
- OpenShift CLI の開始方法

追加情報

MTC と CI/CD パイプラインの再デプロイオプションの選択

ある OpenShift クラスターから別のクラスターにアプリケーションを移行するには、慎重に検討する必要があります。理想的には、CI/CD パイプラインを使用してアプリケーションを再デプロイし、永続的なボリュームデータの移行を処理することで、スムーズな移行が望まれます。ただし、実際には、クラスターで実行中のアプリケーションは、時間の経過とともに予期しない変更を受けやすくなります。これらの変更により、アプリケーションが元のデプロイ状態から徐々に逸脱する可能性があります。MTC は、名前空間の正確な内容が不確実で、すべてのアプリケーションコンポーネントを新しいクラスターにシームレスに移行することが重要なシナリオ向けのソリューションを提供します。

適切な選択を行うには、特定のシナリオを評価し、各アプローチの利点を比較検討する必要があります。これにより、ニーズと優先順位に沿った、成功したシームレスな移行を実現できます。2 つのオプションから選択するための追加のガイドラインを次に示します。

CI/CD パイプラインの再デプロイ

パイプラインを使用してアプリケーションを確実に再デプロイできる場合は、CI/CD パイプライン方式が推奨されます。これにより、移行が制御され、予測可能で、既存のデプロイプラクティスと整合されます。この方法を選択すると、ブルー/グリーンデプロイ戦略または Canary デプロイ戦略を使用して、ロードを ROSA のデプロイに徐々にシフトできます。このシナリオでは、このパターンは、Jenkins がオンプレミス環境からのアプリケーションデプロイをオーケストレーションしていることを前提としています。

利点:

- ソース ARO クラスターへの管理アクセスや、ソースクラスターまたは宛先クラスターに演算子をデプロイする必要はありません。
- このアプローチは、DevOps 戦略を使用してトラフィックを徐々に切り替えるのに役立ちます。

欠点:

- アプリケーションの機能をテストするには、より多くの労力が必要です。
- アプリケーションに永続データが含まれている場合は、 AWS DataSync または他のツールを使用してデータをコピーするための追加のステップが必要です。

MTC 移行

実際には、実行中のアプリケーションが予期しない変更を受け、最初のデプロイから逸脱する可能性があります。ソースクラスター上のアプリケーションの現在の状態が不明な場合は、MTC オプションを選択します。たとえば、アプリケーションエコシステムがさまざまなコンポーネント、設定、データストレージボリュームにまたがる場合は、アプリケーションとその環境全体をカバーする完全な移行を確実に行うために、MTC を選択することをお勧めします。

利点:

- MTC は、ワークロードの完全なバックアップと復元を提供します。
- ワークロードの移行中に、永続データをソースからターゲットにコピーします。
- ソースコードリポジトリへのアクセスは必要ありません。

欠点:

- 送信元クラスターと送信先クラスターに MTC 演算子をインストールするには、管理者権限が必要です。
- DevOps チームは、MTC ツールを使用し、移行を実行するためのトレーニングが必要です。

App2Container が生成した Docker イメージを最適化する

作成者: Varun Sharma (AWS)

概要

AWS App2Container は、コードを変更することなく、オンプレミスまたは仮想マシンで実行中の既存のアプリケーションをコンテナに変換する上で役立つコマンドラインツールです。

アプリケーションタイプに基づき、App2Container は保守的なアプローチで依存関係を特定します。プロセスモードの場合は、アプリケーションサーバー上のすべての非システムファイルがコンテナイメージに含まれます。このような場合、かなり大きなイメージが生成される可能性があります。

このパターンは、App2Container によって生成されるコンテナイメージを最適化するアプローチを提供します。App2Container がプロセスモードで検出したすべての Java アプリケーションに適用できます。このパターンで定義されているワークフローは、アプリケーションサーバー上で実行されるように設計されています。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Linux サーバー上のアプリケーションサーバー上で実行中の Java アプリケーション
- Linuxサーバー上で、すべての前提条件を満たした状態で App2Container をインストールして設定

アーキテクチャ

ソーステクノロジースタック

- Linux サーバー上で実行中の Java アプリケーション

ターゲットテクノロジースタック

- App2Container が生成した Docker イメージ

ターゲットアーキテクチャのフロー

1. アプリケーションサーバー上で実行中の Java アプリケーションを検出し、アプリケーション分析します。
2. アプリケーションをコンテナ化します。
3. Docker イメージのサイズを評価します。イメージが大きすぎる場合は、手順 4 に進みます。
4. シェルスクリプト (添付) を使用して、サイズの大きいファイルを特定します。
5. `analysis.json` ファイル内の `appExcludedFiles` と `appSpecificFiles` リストを更新します。

ツール

ツール

- App2Container - AWS App2Container (A2C) はオンプレミスデータセンターまたは仮想マシンで実行中のアプリケーションをリフトアンドシフトし、Amazon Elastic Container Service (Amazon ECS) または Amazon Elastic Kubernetes Service (Amazon EKS) で管理されるコンテナで実行できるようにするコマンドラインツールです。

コード

`optimizeImage.sh` シェルスクリプトと `analysis.json` ファイルの例が添付されています。

`optimizeImage.sh` ファイルは App2Container で生成されたファイル `ContainerFiles.tar` のコンテンツを確認するユーティリティスクリプトです。このレビューでは、サイズが大きくて除外できるファイルまたはサブディレクトリが特定されます。このスクリプトは次の tar コマンドのラッパーです。

```
tar -Ptvf <path>|tr -s ' '|cut -d ' ' -f3,6| awk '$2 ~/<filetype>$/'| awk '$2 ~/
^<toplevel>/'| cut -f1-<depth> -d'/'|awk '{ if ($1>= <size>) arr[$2]+=$1 } END { for
 (key in arr) { if(<verbose>) printf("%-50s\t%-50s\n", key, arr[key]) else printf("%s,
\n", key) } } '|sort -k2 -nr
```

tar コマンドでは、スクリプトは次の値を使用します。

`path`	`ContainerFiles.tar` へのパス
`filetype`	一致するファイルタイプ

`toplevel`	一致する最上位レベルのディレクトリ
`depth`	絶対パスの深さ
`size`	各ファイルのサイズ

スクリプトは、次を実行します。

1. `tar -Ptvf` を使用して、ファイルを抽出せずに一覧表示します。
2. 最上位レベルのディレクトリから始めて、ファイルタイプ別にファイルをフィルタリングします。
3. 深さに基づき、絶対パスをインデックスとして生成します。
4. インデックスとストアに基づき、サブディレクトリの合計サイズを提供します。
5. サブディレクトリのサイズを出力します。

tar コマンドで値を手動で置き換えることもできます。

エピック

アプリケーションを検出、分析、コンテナ化する

タスク	説明	必要なスキル
オンプレミス Java アプリケーションを検出します。	アプリケーションサーバーで実行中のすべてのアプリケーションを検出するには、以下のコマンドを実行します。 `sudo app2container inventory`	AWS DevOps
検出したアプリケーションを分析します。	インベントリ段階で取得した `application-id` を使用して各アプリケーションを分析するには、以下のコマンドを実行します。	AWS DevOps

タスク	説明	必要なスキル
	`sudo app2container analyze --application-id <java-app-id>`	
分析したアプリケーションをコンテナ化します。	アプリケーションをコンテナ化するには、次のコマンドを実行します。 `sudo app2container containerize --application-id <application-id>` このコマンドは、ワークスペースの場所に Docker イメージと tar バンドルを生成します。 Docker イメージが大きすぎる場合は、次の手順に進みます。	AWS DevOps

App2 Container で抽出された tar ファイルから、appExcludedFiles と appSpecificFiles を特定する

タスク	説明	必要なスキル
アーティファクトの tar ファイルサイズを特定します。	`{workspace}/{java-app-id}/Artifacts` にある `ContainerFiles.tar` ファイルを特定します。ここで、`workspace` は App2Container ワークスペース、`java-app-id` はアプリケーション ID です。	AWS DevOps

タスク	説明	必要なスキル
	`./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 0 -t / -v` これは最適化後の tar ファイルの合計サイズです。	

/ディレクトリの下にあるサブディレクトリとそのサイズを一覧表示します。	/ 最上位レベルディレクトリにある主要なサブディレクトリのサイズを特定するには、以下のコマンドを実行します。
	
	`./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 1 -t / -s 1000000 -v`
	`/var`
	`554144711`
	`/usr`
	`2097300819`
	`/tmp`
	`18579660`
	`/root`
	`43645397`
	`/opt`
	`222320534`
	`/home`
	`65212518`
	`/etc`
	`11357677`
		AWS DevOps

タスク	説明	必要なスキル
/ディレクトリの下にある大きなサブディレクトリを特定します。	前のコマンドで一覧表示されている各サブディレクトリの場合は、そのサブディレクトリのサイズを特定します。-t で深さを増分し、-d で最上位レベルのディレクトリを示します。 たとえば、/var を最上位レベルのディレクトリとして使用します。/var で、大きなサブディレクトリとそのサイズをすべて特定します。 `./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 2 -t /var -s 1000000 -v` 前のステップで一覧表示された各サブディレクトリ (/usr、/tmp、/opt、/homeなど) に、このプロセスを繰り返します。	AWS DevOps

タスク	説明	必要なスキル

/ディレクトリの下の各サブディレクトリにある大きなフォルダを分析します。	前の手順で一覧表示された各サブディレクトリについて、アプリケーションの実行に必要なフォルダを特定します。
	
	たとえば、前の手順のサブディレクトリを使用して、/var ディレクトリ内のすべてのサブディレクトリとそのサイズを一覧表示します。アプリケーションで必要なサブディレクトリをすべて特定します。
	
	`/var/tmp 237285851`
	`/var/lib 24489984`
	`/var/cache 237285851`
	
	アプリケーションで必要のないサブディレクトリを除外するには、analysis.json ファイルで、これらのサブディレクトリを containerParameters の appExcludedFiles セクションに追加します。
	
	analysis.json ファイルの例が添付されています。
		AWS DevOps

タスク	説明	必要なスキル
AppExcludes リストから必要なファイルを特定します。	AppExcludes リストに追加される各サブディレクトリについて、アプリケーションで必要なそのサブディレクトリにあるファイルをすべて特定します。analysis.json ファイルで、containerParameters の appSpecificFiles セクションに特定のファイルまたはサブディレクトリを追加します。 たとえば、/usr/lib ディレクトリが除外リストに追加されるが、アプリケーションによって /usr/lib/jvm が必要な場合は、その appSpecificFiles セクションに /usr/lib/jvm を追加します。	AWS DevOps

アプリケーションを再度抽出し、コンテナ化する

タスク	説明	必要なスキル
分析したアプリケーションをコンテナ化します。	アプリケーションをコンテナ化するには、次のコマンドを実行します。 `sudo app2container containerize --application-id <application-id>`	AWS DevOps

タスク	説明	必要なスキル
	このコマンドは、ワークスペースの場所に Docker イメージと tar バンドルを生成します。	
アーティファクトの tar ファイルサイズを特定します。	{workspace}/{java-app-id}/Artifacts にある ContainerFiles.tar ファイルを特定します。ここで、 workspace は App2Container ワークスペース、 java-app-id はアプリケーション ID です。 `./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 0 -t / -v` これは最適化後の tar ファイルの合計サイズです。	AWS DevOps

タスク	説明	必要なスキル
Docker イメージを実行します。	イメージがエラーなしで起動することを検証するには、以下のコマンドを使用して Docker イメージをローカルで実行します。 コンテナーの `imageId` を特定するには、`docker images |grep java-app-id` を使用します。 コンテナを実行するには、`docker run -d <image id>` を使用します。	AWS DevOps

関連リソース

- App2Container とは
- AWS App2Container — Java と.NET アプリケーション用の新しいコンテナ化ツール (ブログ投稿)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

ノードアフィニティ、テイント、許容範囲を使用して Kubernetes ポッドを Amazon EKS に配置します。

Hitesh Parikh (AWS) とRaghu Bhamidimarri (AWS) によって作成されました

概要

このパターンは、Kubernetes ノードアフィニティ、ノードテイント、ポッドトレレーションを使用して、Amazon Web Services (AWS) クラウド上の Amazon Elastic Kubernetes Service (Amazon EKS) クラスター内の特定のワーカーノードにアプリケーションポッドを意図的にスケジュールする方法を示しています。

テイントは、ノードが一連のポッドを拒否できるようにするノードプロパティです。トレレーションは、Kubernetes スケジューラーがテイントが一致するノードにポッドをスケジュールできるようにするポッドプロパティです。

ただし、許容値だけでは、スケジューラーがテイントのないワーカーノードにポッドを配置するのを防ぐことはできません。たとえば、許容範囲のある計算負荷の高いポッドが、テイントされていない汎用のノードに意図せずスケジュールされる可能性があります。このシナリオでは、ポッドのノードアフィニティプロパティは、ノードアフィニティで指定されたノード選択基準を満たすノードにポッドを配置するようにスケジューラーに指示します。

テイント、トレレーション、ノードアフィニティが一体となって、ポッドで指定されたノードアフィニティノード選択基準に一致するテイントとノードラベルを持つノード上で一貫してポッドをスケジュールするようにスケジューラーに指示します。

このパターンは、Kubernetes デプロイマニフェストファイルの例と、EKS クラスターの作成、アプリケーションのデプロイ、ポッド配置の検証の手順を示しています。

前提条件と制限

前提条件

- AWS アカウントでリソースを作成するように設定された認証情報を 持つ AWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI)
- eksctl
- kubectl
- Docker がインストールされ (使用しているオペレーティングシステム用)、エンジンが起動した (Docker のライセンス要件については、Docker サイト を参照してください)。

- Fedora バージョン 34 以降
- お気に入りの統合開発環境 (IDE) で実行されている Java マイクロサービス。例えば、IntelliJ IDEA Community Edition や Eclipse など (Java マイクロサービスがない場合は、「Amazon EKS パターンにサンプルの Java マイクロサービスをデプロイする」と「Spring でマイクロサービスを作成する」を参照してマイクロサービスを作成する)

機能制限

- このパターンは Java コードを提供するものではなく、既に Java に精通していることを前提としています。基本的な Java マイクロサービスを作成するには、Amazon EKS へのサンプル Java マイクロサービスのデプロイ を参照してください。
- この記事のステップでは、コストが発生する可能性のある AWS リソースを作成します。パターンを実装して検証する手順を完了したら、必ず AWS リソースをクリーンアップしてください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EKS
- Java
- Docker
- Amazon Elastic Container Registry (Amazon ECR)

ターゲット アーキテクチャ

ソリューションアーキテクチャ図は、2 つのポッド (デプロイ 1 とデプロイ 2) と、それぞれ 2 つのノードを持つ 2 つのノードグループ (ng1 と ng2) で構成される Amazon EKS を示しています。トリガー と には、以下のプロパティがあります。

	デプロイ 1 ポッド	デプロイ 2 ポッド	ノードグループ 1 (ng1)	ノードグループ 2 (ng2)
容認	キー：classified_workload、値：true、効果：NoSchedule	なし		

	キー：machine_learning_workload、値：true、エフェクト：NoSchedule			
ノードアフィニティ	キー：alpha.eksctl.io/nodegroup-name = ng1;	なし	nodeGroups.name = ng1	
テイント			キー：classified_workload、値：true、効果：NoSchedule キー：machine_learning_workload、値：true、エフェクト：NoSchedule	なし

1. Deployment 1 Pod には許容範囲とノードアフィニティが定義されています。これにより、Kubernetes スケジューラーにデプロイメントポッドをノードグループ 1 (ng1) ノードに配置するよう指示されます。
2. ノードグループ 2 (ng2) にはデプロイ 1 のノードアフィニティノードセレクター式と一致するノードラベルがないため、ポッドは ng2 ノードでスケジュールされません。
3. Deployment 2 ポッドには、デプロイマニフェストで許容範囲やノードアフィニティが定義されていません。ノードにテイントがあるため、スケジューラーはノードグループ 1 でのデプロイ 2 ポッドのスケジュールを拒否します。
4. ノードにはテイントがないため、デプロイ 2 ポッドは代わりにノードグループ 2 に配置されます。

このパターンは、テイントとトレレーションをノードアフィニティと組み合わせて使用することで、特定のワーカーノードセットでのポッドの配置を制御できることを示しています。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、 で Kubernetes を実行する際に役立ちます。独自の Kubernetes コントロールプレーンまたはノードをインストールおよび維持する必要はありません。
- eksctl はAWS kubectlに相当し、EKSの作成に役立ちます。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- kubectlは、Kubernetes クラスターに対してコマンドを実行するためのコマンドラインインターフェイスです。

エピック

EKS クラスターを作成します。

タスク	説明	必要なスキル

クラスター.yaml ファイルを作成します。	`cluster.yaml` というファイルを次の内容で作成します。
	`apiVersion: eksctl.io/v1alpha5`
	`kind: ClusterConfig`
		アプリ所有者、AWS DevOps、クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル

	<pre>metadata:
	  name: eks-taint-demo
	  region: us-west-1
	
	# Unmanaged nodegroup
	s with and without
	 taints.
	nodeGroups:
	  - name: ng1
	  instanceType:
	 m5.xlarge
	  minSize: 2
	  maxSize: 3
	  taints:
	  - key: classifie
	d_workload
	  value: "true"
	  effect:
	 NoSchedule
	  - key: machine_l
	earning_workload
	  value: "true"
	  effect:
	 NoSchedule
	
	  - name: ng2
	  instanceType:
	 m5.xlarge
	  minSize: 2
	  maxSize: 3</pre>	

eksctl を使用してクラスターを作成します。	cluster.yaml ファイルを実行して EKS クラスターを作成します。クラスターの作成には数分かかることがあります。
	
	<pre>eksctl create cluster -f
	 cluster.yaml</pre>
		AWS DevOps、AWS システム管理者、アプリケーション開発者

イメージを作成して Amazon ECR にアップロードします。

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	Amazon ECR コンソールを使ってリポジトリを作成するには、リポジトリを作成するを参照してください。リポジトリの URI をメモしておきます。	AWS DevOps、DevOps エンジニア

Dockerfile を作成します。	パターンのテストに使用する Docker コンテナイメージがすでにある場合は、このステップを省略できます。
	
	Dockerfile を作成するには、以下のスニペットを参考にしてください。エラーが発生した場合は、トラブルシューティング セクションを参照してください。
	
	`FROM adoptopenjdk/openjdk11:jdk-11.0.14.1_1-alpine`
	
	`RUN apk add maven`
	`WORKDIR /code`
	
	`# Prepare by downloading dependencies`
	`ADD pom.xml /code/pom.xml`
	`RUN ["mvn", "dependency:resolve"]`
	`RUN ["mvn", "verify"]`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	`# Adding source, compile and package into a fat jar`
	`ADD src /code/src`
	`RUN ["mvn", "package"]`
	
	`EXPOSE 4567`
	`CMD ["java", "-jar", "target/eksExample-jar-with-dependencies.jar"]`	

pom.xml とソースファイルを作成し、Docker イメージをビルドしてプッシュします。	pom.xml ファイルと Java ソースファイルを作成するには、Amazon EKS パターンにサンプル Java マイクロサービスをデプロイする を参照してください。 そのパターンの指示に従って Docker イメージをビルドしてプッシュします。	AWS DevOps、DevOps エンジニア

Amazon ECS へのデプロイ

タスク	説明	必要なスキル
デプロイメント.yaml ファイルを作成します。	deployment.yaml ファイルを作成するには、追加情報セクションのコードを使用してください。 コードでは、ノードアフィニティの鍵はノードグループの作成時に作成するラベルです。このパターンは eksctl が作成したデフォルトのラ	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	ベルを使用します。ラベルのカスタマイズについては、Kubernetes ドキュメントの[ノードへのポッドの割り当て]を参照してください。 ノードアフィニティキーの値は、`cluster.yaml` によって作成されたノードグループの名前です。 次のコマンドを実行して、テイントのキーと値を取得します。 `kubectl get nodes -o json | jq '.items[].spec.taints'` イメージは、前のステップで作成した Amazon ECR リポジトリの URI です。	
ファイルをデプロイします。	Amazon EKS にデプロイするには、次のコマンドを実行します。 `kubectl apply -f deployment.yaml`	アプリ開発者、DevOps、AWS DevOps エンジニア

タスク	説明	必要なスキル

デプロイを確認してください。	1. 次のコマンドを実行して、ポッドの準備が整っているかどうかを確認します。
	`kubectl get pods -o wide`
	ポッドの準備が整うと、出力は Running STATUS と表示されます。
	`NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES`
	`<pod_name> 1/1 Running 0 12d 192.168.18.50 ip-192-168-20-110.us-west-1.compute.internal <none> <none>`
	ポッドの名前とノードの名前を書き留めます。次のステップをスキップできます。
	2. (オプション) ポッドの詳細情報を取得し、ポッドの許容範囲を確認するには、次のコマンドを実行します。
	`kubectl describe pod <pod_name>`
		アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	出力の例は追加情報のセクションにあります。
	3. 次のコマンドを実行して、ノードのポッドの配置が正しいことを確認します。
	`kubectl describe node <node name> | grep -A 1 "Taints"`
	ノードのテイントが許容範囲と一致し、ノードのラベルが deployment.yaml で定義されたノードアフィニティと一致することを確認します。
	許容範囲とノードアフィニティが設定されたポッドは、テイントとノードアフィニティラベルが一致するノードに配置する必要があります。前のコマンドでは、ノード上のテイントが表示されます。以下に出力例を示します。
	`kubectl describe node ip-192-168-29-181.us-west-1.compute.internal | grep -A 1 "Taints"`
	`Taints: classifled_workload=true:NoSchedule`	

タスク	説明	必要なスキル
	machine_learning_workload=true:NoSchedule さらに、以下のコマンドを実行して、ノードが配置されているノードのラベルがノードアフィニティノードのラベルと一致していることを確認します。 kubectl get node <node name> --show-labels 4. アプリケーションが意図したとおりに動作していることを確認するには、以下のコマンドを実行してノードのログを確認します。 kubectl logs -f <name-of-the-pod>	

タスク	説明	必要なスキル

許容範囲やノードアフィニティなしで 2 つ目のデプロイ.yaml ファイルを作成します。	この追加ステップは、デプロイメントマニフェストファイルにノードアフィニティや許容値が指定されていない場合に、生成されるポッドがテイントのあるノードにスケジュールされないことを検証することです。(テイントのないノードでスケジュールする必要があります)。次のコードを使用して、deploy_no_taint.yaml という新しいデプロイメントファイルを作成します。
	
	`apiVersion: apps/v1`
	`kind: Deployment`
	`metadata:`
	`  name: microservice-deployment-non-tainted`
	`spec:`
	`  replicas: 1`
	`  selector:`
	`  matchLabels:`
	`  app.kubernetes.io/name: java-microservice-no-taint`
	`  template:`
	`  metadata:`
	`  labels:`
	`  app.kubernetes.io/name: java-microservice-no-taint`
	`  spec:`
	`  containers:`
	`  - name: java-microservice-container-2`
		アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	<pre>image:
	 <account_number>.d
	kr.ecr<region>.ama
	zonaws.com/<reposi
	tory_name>:latest
	ports:
	- container
	Port: 4567</pre>	

2 番目のデプロイ.yaml ファイルをデプロイし、ポッドの配置を検証します。	1. 以下のコマンドを実行してください。
	<pre>kubectl apply -f
	 deploy_no_taint.ya
	ml</pre>
	2. デプロイが成功したら、前に実行したのと同じコマンドを実行して、ノードグループ内のポッドの配置をテイントなく確認します。
	<pre>kubectl describe node
	 <node_name> | grep
	 "Taints"</pre>
	出力は、以下を返します。
	<pre>Taints: <none></pre>
	これでテストは完了です。
		アプリ開発者、AWS DevOps、DevOps エンジニア

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースをクリーンアップします。	稼働したままのリソースに対して AWS 料金が発生しないようにするには、以下のコマンドを使用します。 `eksctl delete cluster --name <Name of the cluster> --region <region-code>`	AWS DevOps、アプリケーション開発者

トラブルシューティング

問題	ソリューション
システムが arm64 アーキテクチャ を使用している場合 (特に M1 Mac でこれを実行している場合)、これらのコマンドの一部は実行されない可能性があります。次の行はエラーになることがあります。 `FROM adoptopenjdk/openjdk11:jdk-11.0.14.1_1-alpine`	Dockerfile の実行中にエラーが発生した場合は、FROM その行を次の行に置き換えてください。 `FROM bellsoft/liberica-openjdk-alpine-musl:17`

関連リソース

- Amazon EKS にサンプル Java マイクロサービスをデプロイします
- Amazon ECR リポジトリを作成します。
- ノードへのポッドの割り当て (Kubernetes ドキュメント)
- テイントとトレレーション (Kubernetes ドキュメント)
- Amazon EKS

- Amazon ECR
- AWS CLI
- Docker
- IntelliJ IDEA
- Eclipse

追加情報

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: microservice-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: java-microservice
  template:
    metadata:
      labels:
        app.kubernetes.io/name: java-microservice
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: alpha.eksctl.io/nodegroup-name
                    operator: In
                    values:
                      - <node-group-name-from-cluster.yaml>
      tolerations: #only this pod has toleration and is viable to go to ng with taint
        - key: "<Taint key>" #classified_workload in our case
          operator: Equal
          value: "<Taint value>" #true
          effect: "NoSchedule"
        - key: "<Taint key>" #machine_learning_workload in our case
          operator: Equal
          value: "<Taint value>" #true
```

```
          effect: "NoSchedule"
      containers:
        - name: java-microservice-container
          image: <account_number>.dkr.ecr<region>.amazonaws.com/
<repository_name>:latest
          ports:
            - containerPort: 4567
```

ポッドのサンプル出力を説明してください

```
Name:         microservice-deployment-in-tainted-nodes-5684cc495b-vpcfx
Namespace:    default
Priority:     0
Node:         ip-192-168-29-181.us-west-1.compute.internal/192.168.29.181
Start Time:   Wed, 14 Sep 2022 11:06:47 -0400
Labels:       app.kubernetes.io/name=java-microservice-taint
              pod-template-hash=5684cc495b
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           192.168.13.44
IPs:
  IP:           192.168.13.44
Controlled By:  ReplicaSet/microservice-deployment-in-tainted-nodes-5684cc495b
Containers:
  java-microservice-container-1:
    Container ID:
 docker://5c158df8cc160de8f57f62f3ee16b12725a87510a809d90a1fb9e5d873c320a4
    Image:          934188034500.dkr.ecr.us-east-1.amazonaws.com/java-eks-apg
    Image ID:       docker-pullable://934188034500.dkr.ecr.us-east-1.amazonaws.com/
java-eks-apg@sha256:d223924aca8315aab20d54eddf3443929eba511b6433017474d01b63a4114835
    Port:           4567/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 14 Sep 2022 11:07:02 -0400
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-ddvvw (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
```

```
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-ddvvw:
    Type:                     Projected (a volume that contains injected data from
 multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 classifled_workload=true:NoSchedule
                             machine_learning_workload=true:NoSchedule
                             node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:                      <none>
```

フィルタリングされた Amazon ECR コンテナイメージをアカウントまたはリージョン間で複製する

作成者: Abdal Garuba (AWS)

概要

Amazon Elastic Container Registry (Amazon ECR) は、クロスリージョンおよびクロスアカウントのレプリケーション機能を使用して、イメージリポジトリ内のすべてのコンテナイメージをAmazon Web Services (AWS) リージョンと AWS アカウントにネイティブに複製できます。(詳細については、AWS ブログ投稿 Amazon ECR でのクロスリージョンレプリケーションの開始を参照してください。) ただし、AWS リージョンまたはアカウント間でコピーされたイメージを基準に基づいてフィルタリングする方法はありません。

このパターンでは、イメージタグパターンに基づいて、Amazon ECR に保存されているコンテナイメージを AWS アカウントとリージョン間で複製する方法を説明します。このパターンでは、Amazon CloudWatch Events を使用して、事前定義されたカスタムタグ付きイメージのプッシュイベントを監視します。プッシュイベントは AWS CodeBuild プロジェクトを開始し、イメージの詳細をそのプロジェクトに渡します。CodeBuild プロジェクトは、提供された詳細に基づいて、ソース Amazon ECR レジストリからコピー先のレジストリにイメージをコピーします。

このパターンでは、特定のタグが付いたイメージがアカウント間でコピーされます。たとえば、このパターンを使用して、本番環境に対応した安全なイメージのみを本番環境の AWS アカウントにコピーできます。開発アカウントでは、イメージを徹底的にテストした後、あらかじめ定義されたタグを安全なイメージに追加し、このパターンの手順に従ってマークしたイメージを本稼働アカウントにコピーできます。

前提条件と制限

前提条件

- ソースと送信先の Amazon ECR レジストリ用のアクティブなAWS アカウント
- このパターンで使用されるツールの管理者権限
- テスト用にローカルマシンにインストールされたDocker
- AWS コマンドラインインターフェイス (AWS CLI)(Amazon ECR への認証用)

制限

- このパターンは、1 つの AWS リージョンでのみソースレジストリのプッシュイベントを監視します。このパターンを他のリージョンにデプロイして、そのリージョンのレジストリを監視できます。
- このパターンでは、1 つの Amazon CloudWatch Events ルールが 1 つのイメージタグパターンを監視します。複数のパターンを確認する場合は、イベントを追加して追加したイメージタグパターンを監視できます。

アーキテクチャ

ターゲット アーキテクチャ

自動化とスケール

このパターンは、infrastructure as code (IaC) スクリプトで自動化し、大規模にデプロイできます。AWS CloudFormation テンプレートを使用してこのパターンをデプロイするには、添付ファイルをダウンロードし、追加情報セクションの指示に従います。

複数の Amazon CloudWatch Events イベント (異なるカスタムイベントパターン) を同じ AWS CodeBuild プロジェクトにポイントして複数のイメージタグパターンを複製できますが、複数のパターンをサポートするには、`buildspec.yaml` ファイル (添付ファイルとツールセクションに含まれる) の二次検証を次のように更新する必要があります。

```
...
if [[ ${IMAGE_TAG} != release-* ]]; then
...
```

ツール

Amazon サービス

- IAM— AWS Identity and Access Management (IAM) を使用すると、AWS サービスとリソースへのアクセスを安全に管理できます。このパターンでは、コンテナイメージを送信先レジストリにプッシュするときに AWS CodeBuild が引き受けるクロスアカウント IAM ロールを作成する必要があります。
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) は、フルマネージド型のレジストリで、 コンテナイメージとアーティファクトの保存、管理、デプロイをどこでも容易にできま

す。ソースレジストリへのイメージプッシュアクションは、Amazon CloudWatch Events によって取得されるイベントバスにシステムイベントの詳細を送信します。

- AWS CodeBuild — AWS CodeBuild は完全マネージド型の継続的インテグレーションサービスで、ソースコードのコンパイル、テストの実行、デプロイ準備が整ったアーティファクトの作成などのジョブを実行する処理能力を提供します。このパターンでは、AWS CodeBuild を使用して、ソース Amazon ECR レジストリから送信先レジストリへのコピーアクションを実行します。
- CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのストリームを提供します。このパターンでは、ルールを使用して Amazon ECR プッシュアクションを特定のイメージタグパターンと照合します。

ツール

- Docker CLI — Docker はコンテナの作成と管理を容易にするツールです。コンテナは、アプリケーションとそのすべての依存関係を 1 つのユニットまたはパッケージにまとめ、コンテナランタイムをサポートする任意のプラットフォームに簡単にデプロイできます。

コード

このパターンは、以下の 2 つの方法で実装できます。

- 自動セットアップ: 添付ファイルで提供されている 2 つの AWS CloudFormation テンプレートをデプロイします。手順については、追加情報セクションを参照してください。
- 手動設定: エピックセクションの手順に従います。

サンプル buildspec.yaml

このパターンで提供されている CloudFormation テンプレートを使用中の場合、`buildspec.yaml` ファイルは CodeBuild リソースに含まれています。

```
  version: 0.2
  env:
    shell: bash
  phases:
    install:
      commands:
        - export CURRENT_ACCOUNT=$(echo ${CODEBUILD_BUILD_ARN} | cut -d':' -f5)
        - export CURRENT_ECR_REGISTRY=${CURRENT_ACCOUNT}.dkr.ecr.
${AWS_REGION}.amazonaws.com
```

```
        - export DESTINATION_ECR_REGISTRY=${DESTINATION_ACCOUNT}.dkr.ecr.
${DESTINATION_REGION}.amazonaws.com
    pre_build:
      on-failure: ABORT
      commands:
        - echo "Validating Image Tag ${IMAGE_TAG}"
        - |
          if [[ ${IMAGE_TAG} != release-* ]]; then
            aws codebuild stop-build --id ${CODEBUILD_BUILD_ID}
            sleep 60
            exit 1
          fi
        - aws ecr get-login-password --region ${AWS_REGION} | docker login -u AWS --
password-stdin ${CURRENT_ECR_REGISTRY}
        - docker pull ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
    build:
      commands:
        - echo "Assume cross-account role"
        - CREDENTIALS=$(aws sts assume-role --role-arn ${CROSS_ACCOUNT_ROLE_ARN} --
role-session-name Rolesession)
        - export AWS_DEFAULT_REGION=${DESTINATON_REGION}
        - export AWS_ACCESS_KEY_ID=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.AccessKeyId')
        - export AWS_SECRET_ACCESS_KEY=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.SecretAccessKey')
        - export AWS_SESSION_TOKEN=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.SessionToken')
        - echo "Logging into cross-account registry"
        - aws ecr get-login-password --region ${DESTINATION_REGION} | docker login -u
 AWS --password-stdin ${DESTINATION_ECR_REGISTRY}
        - echo "Check if Destination Repository exists, else create"
        - |
          aws ecr describe-repositories --repository-names ${REPO_NAME} --region
 ${DESTINATION_REGION} \
          || aws ecr create-repository --repository-name ${REPO_NAME} --region
 ${DESTINATION_REGION}
        - echo "retag image and push to destination"
        - docker tag ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
 ${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
        - docker push ${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
```

エピック

IAM ロールを作成する

タスク	説明	必要なスキル

CloudWatch Events ロールを作成します。	ソース AWS アカウントで、Amazon CloudWatch Events が引き受ける IAM ロールを作成します。ロールには、AWS CodeBuild プロジェクトを開始する権限が必要です。
	
	AWS CLI を使用してロールを作成するには、IAM ドキュメントの指示に従います。
	
	信頼ポリシー(`trustpolicy.json`) の例:
	
	`{`
	`  "Version": "2012-10-17",`
	`  "Statement": {`
	`  "Effect": "Allow",`
	`  "Principal": {"Service": "events.amazonaws.com"},`
	`  "Action": "sts:AssumeRole"`
	`  }`
	`}`
	
	権限ポリシー (`permissionpolicy.json`)の例:
	
	`{`
	`  "Version": "2012-10-17",`
		AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル

	<pre> "Statement": {
	    "Effect": "Allow",
	    "Action": "codebuil
	d:StartBuild",
	    "Resource":
	 "<CodeBuild Project
	 ARN>"
	  }
	}</pre>	

タスク	説明	必要なスキル
CodeBuild ロールを作成します。	IAM ドキュメントの指示に従って、AWS CodeBuild が引き受ける IAM ロールを作成します。ロールには、次の権限が必要です。 • 送信先のクロスアカウントロールを引き受ける権限 • ロググループとログストリームを作成する権限、ログイベントを記録する権限 • AmazonEC2ContainerRegistryReadOnly 管理ポリシーをロールに追加することによる、すべての Amazon ECR リポジトリに対する読み取り専用権限 • CodeBuild を停止する権限 信頼ポリシー(`trustpolicy.json`)の例: `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "codebuild.amazonaws.com" }, "Action": "sts:AssumeRole" }`	AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル

	]
	}
	
	権限ポリシー (permissionpolicy.json)の例:
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Action": [
	"codebuild:StartBuild",
	"codebuild:StopBuild",
	"codebuild:Get*",
	"codebuild:List*",
	"codebuild:BatchGet*"
	],
	"Resource": "*",
	"Effect": "Allow"
	},
	{
	"Action": [
	"logs:CreateLogGroup",
	"logs:CreateLogStream",
	"logs:PutLogEvents"
	],	

タスク	説明	必要なスキル
	<pre>"Resource": "*", "Effect": "Allow" }, { "Action": "sts:AssumeRole", "Resource": "<ARN of destination role>", "Effect": "Allow", "Sid": "AssumeCrossAccountArn" }] }</pre> 以下のように CLI コマンドに管理ポリシー AmazonEC2ContainerRegistryReadOnly を添付します。 <pre>~$ aws iam attach-role-policy \ --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \ --role-name <name of CodeBuild Role></pre>	

タスク	説明	必要なスキル
クロスアカウントロールを作成します。	送信先 AWS アカウントで、ソースアカウントを引き受ける AWS CodeBuild ロール用の IAM ロールを作成します。クロスアカウントロールでは、コンテナイメージが新しいリポジトリを作成し、コンテナイメージを Amazon ECR にアップロードできるようにする必要があります。 AWS CLI を使用して IAM ロールを作成する場合は、IAM ドキュメントの指示に従います。 前の手順の AWS CodeBuild プロジェクトを許可するには、以下の信頼ポリシーを使用します。 `{ "Version": "2012-10-17", "Statement": { "Effect": "Allow", "Principal": { "AWS": "<ARN of source codebuild role>" }, "Action": "sts:AssumeRole" } }` 前の手順の AWS CodeBuild プロジェクトを許可して送信	AWS 管理者、AWS DevOps、クラウド管理者、クラウドアーキテクト、DevOps エンジニア、AWS システム管理者

タスク	説明	必要なスキル
	先レジストリにイメージを保存するには、以下の権限ポリシーを使用します。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["ecr:GetDownloadUrlForLayer", "ecr:BatchCheckLayerAvailability", "ecr:PutImage", "ecr:InitiateLayerUpload", "ecr:UploadLayerPart", "ecr:CompleteLayerUpload", "ecr:GetRepositoryPolicy", "ecr:DescribeRepositories", "ecr:GetAuthorizationToken", "ecr:CreateRepository"],`	

タスク	説明	必要なスキル
	<pre>"Resource": "*", "Effect": "Allow" }] }</pre>	

CodeBuild プロジェクトを作成する

タスク	説明	必要なスキル
CodeBuild プロジェクトを作成します。	AWS CodeBuild ドキュメントの指示に従って、ソースアカウントに AWS CodeBuild プロジェクトを作成します。プロジェクトはソースレジストリと同じリージョンにある必要があります。 以下のようにプロジェクトを設定します。 • 環境タイプ: `LINUX CONTAINER` • サービスロール : `CodeBuild Role` • 特権モード: `true` • 環境イメージ: `aws/codebuild/standard:x.x` (入手可能な最新イメージを使用) • 環境変数: • `CROSS_ACCOUNT_ROLE_ARN` : クロスアカウント	AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	ロールの Amazon リソースネーム (ARN) • DESTINATION_REGION：クロスアカウントリージョンの名前 • DESTINATION_ACCOUNT：送信先アカウントの番号 • ビルド仕様: ツールセクションにリストされている buildspec.yaml ファイルを使用します。	

イベントを作成する

タスク	説明	必要なスキル
イベントルールを作成します。	パターンはコンテンツフィルタリング機能を使用するため、Amazon EventBridge を使用してイベントを作成する必要があります。EventBridge ドキュメントの指示に従い、変更をしてイベントとターゲットを作成します。 • パターンの定義の場合は、[Event Pattern（イベントパターン）] を選択してから、[Custom pattern（カスタムパターン）] を選択します。 • 以下のカスタムイベントパターンのサンプルコー	AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル

	ドを、表示されたテキストボックスにコピーします。
	
	`{ "source": ["aws.ecr "], "detail-type": ["ECR Image Action"], "detail": { "action-type": ["PUSH"], "result": ["SUCCESS"], "image-ta g": [{ "prefix": "release-"}] } }`
	
	• ターゲットの選択の場合は、AWS CodeBuild プロジェクトを選択し、前のエピックで作成した AWS CodeBuild プロジェクトの ARN を貼り付けます。
	• 入力の設定の場合は、[Input Transformer (入力トランスフォーマー)] を選択します。
	  • Input Path (入力パス) テキストボックスに、以下を貼り付けます。
	
	`{"IMAGE_TAG":"$.de tail.image-tag","R EPO_NAME":"$.detai`	

タスク	説明	必要なスキル

	<pre>l.repository-name"
	}</pre>
	• Input Template (入力テンプレート) テキストボックスに、以下を貼り付けます。
	<pre>{"environmentVariablesOverride":
	[{"name":
	"IMAGE_TAG",
	"value":<IMAGE_TAG>},{"name":"REPO_NAME","value":<REPO_NAME>}]}</pre>
	• [Use existing role (既存のロールを使用)] を選択し、以前に [Create IAM roles (IAM ロールの作成)] エピックで作成した CloudWatch イベントロールの名前を選択します。	

検証

タスク	説明	必要なスキル
Amazon ECR で認証します。	Amazon ECR ドキュメントの手順に従って、ソースと送信先の両方のレジストリに対して認証します。	AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、DevOps エンジニア、クラウドアーキテクト
イメージのレプリケーションをテストします。	ソースアカウントで、`release-` のプレフィッ	AWS 管理者、AWS DevOps、AWS システム管理

タスク	説明	必要なスキル
	クスが付いたイメージタグを使用して、コンテナイメージを新規または既存の Amazon ECR ソースリポジトリにプッシュします。イメージをプッシュするには、Amazon ECR ドキュメントの手順に従います。 CodeBuild プロジェクトの進捗状況は、CodeBuild コンソールで監視できます。 CodeBuild プロジェクトが正常に完了したら、送信先の AWS アカウントにサインインし、Amazon ECR コンソールを開いて、イメージが送信先の Amazon ECR レジストリに存在することを確認します。	者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア
イメージの除外をテストします。	ソースアカウントで、カスタムプレフィックスのないイメージタグを使用して、コンテナイメージを新規または既存の Amazon ECR ソースリポジトリにプッシュします。 CodeBuild プロジェクトが開始されていないこと、送信先のレジストリにコンテナイメージが表示されていないことを確認します。	AWS 管理者、AWS DevOps、AWS システム管理者、クラウド管理者、クラウドアーキテクト、DevOps エンジニア

関連リソース

- CodeBuild の開始
- Amazon EventBridge の開始
- Amazon EventBridge イベントパターンでのコンテンツベースのフィルタリング
- IAM ロールを使用して AWS アカウント間でアクセスを委任する
- プライベートイメージのレプリケーション

追加情報

このパターンのリソースを自動的にデプロイするには、次の手順に従います。

1. 添付ファイルをダウンロードして、`part-1-copy-tagged-images.yaml` と `part-2-destination-account-role.yaml` の 2 つの CloudFormation テンプレートを抽出します。
2. AWS CloudFormation コンソールにログインし、ソース Amazon ECR レジストリと同じ AWS アカウントとリージョンに `part-1-copy-tagged-images.yaml` をデプロイします。必要に応じて、パラメータを更新します。テンプレートは以下のリソースをデプロイします。
   - Amazon CloudWatch Events IAM ロール
   - AWS CodeBuild プロジェクト IAM ロール
   - AWS CodeBuild プロジェクト
   - CloudWatch Events ルール
3. [Outputs (出力)] タブの `SourceRoleName` の値を書き留めます。この値は次のステップで必要になります。
4. Amazon ECR コンテナイメージのコピー先の AWS アカウントに、2 つ目の CloudFormation テンプレート `part-2-destination-account-role.yaml` をデプロイします。必要に応じて、パラメータを更新します。`SourceRoleName` パラメータの場合は、手順 3 の値を指定します。このテンプレートはクロスアカウント IAM ロールをデプロイします。
5. エピックセクションの最後の手順で説明したように、イメージの複製と除外を検証します。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

コンテナを再起動せずにデータベースの認証情報をローテーションする

作成者: Josh Joy (AWS)

概要

Amazon Web Services (AWS) クラウドでは、AWS Secrets Manager を使用して、ライフサイクル全体にわたってデータベース認証情報をローテーション、管理、取得できます。ユーザーとアプリケーションは Secrets Manager API を呼び出してシークレットを取得し、機密情報をプレーンテキストでコードに書き込む必要がなくなります。

マイクロサービスワークロードにコンテナを使用している場合は、認証情報を AWS Secrets Manager に安全に保存できます。設定とコードを区別するために、通常、これらの認証情報はコンテナに挿入されます。ただし、認証情報を定期的かつ自動的にローテーションすることは重要です。また、失効後に認証情報を更新する機能をサポートすることも重要です。同時に、アプリケーションには、ダウンストリームの可用性への潜在的な影響を抑えながら認証情報をローテーションする機能が必要です。

このパターンは、AWS Secrets Manager で保護されているシークレットを、コンテナを再起動せずにコンテナ内でローテーションする方法を示しています。さらに、このパターンでは Secrets Manager のクライアント側キャッシュコンポーネントを使用することにより、Secrets Manager への認証情報の検索回数が減ります。クライアント側のキャッシュコンポーネントを使用してアプリケーション内の認証情報を更新する場合、ローテーションされた認証情報を取得するためにコンテナを再起動する必要はありません。

このアプローチは、Amazon Elastic Kubernetes Service (Amazon EKS) および Amazon Elastic Container Service (Amazon ECS) で機能します。

2 つのシナリオについて説明します。シングルユーザーシナリオでは、シークレットローテーション時に期限切れの認証情報を検出してデータベース認証情報が更新されます。認証情報キャッシュにシークレットを更新するように指示され、アプリケーションがデータベース接続を再確立します。クライアント側のキャッシュコンポーネントは、認証情報をアプリケーション内にキャッシュするため、認証情報を検索するたびに Secrets Manager にアクセスする必要がなくなります。コンテナを再起動して認証情報を強制的に更新しなくても、認証情報はアプリケーション内でローテーションされます。

2 つ目のシナリオでは、2 人のユーザーが交代してシークレットをローテーションします。2 人のアクティブユーザーがいると、1 人のユーザーの認証情報が常にアクティブになるため、ダウンタイムの可能性が低くなります。2 人のユーザーによる認証情報のローテーションは、クラスターを含む大規模なデプロイで、認証情報の更新の伝達遅延がわずかに発生する可能性がある場合に役立ちます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon EKS または Amazon ECS のコンテナ内で実行されるアプリケーション。
- 認証情報はSecrets Manager に保存され、ローテーションが有効になっています。
- 2 ユーザーソリューションをデプロイする場合の Secrets Manager に保存される 2 番目の認証情報セット。コード例は GitHub リポジトリの aws-secrets-manager-rotation-lambdas にあります。
- Amazon Aurora データベース。

制限

- この例は Python アプリケーションを対象としています。Java アプリケーションでは、Java クライアント側キャッシュコンポーネントまたは Secrets Manager 用の JDBC クライアント側キャッシュライブラリを使用できます。

アーキテクチャ

ターゲットアーキテクチャ

シナリオ 1 – 単一ユーザーの認証情報のローテーション

最初のシナリオでは、1 つのデータベース認証情報が Secrets Manager によって定期的にローテーションされます。アプリケーションコンテナは Fargate で実行されます。最初のデータベース接続が確立されると、アプリケーションコンテナは Aurora のデータベース認証情報を取得します。その後、Secrets Manager キャッシュコンポーネントは、将来の接続確立に備えて認証情報をキャッシュします。ローテーション期間が経過すると、認証情報は失効し、データベースは認証エラーを返します。次に、アプリケーションはローテーションされた認証情報を取得し、キャッシュを無効にし、Secrets Manager のクライアント側キャッシュコンポーネントを介して認証情報キャッシュを更新します。

このシナリオでは、認証情報がローテーションされ、古い認証情報が古い接続に使用されている間は、最小限の中断しか発生しない可能性があります。この問題は 2 人のユーザーによるシナリオを使用することで解決できます。

シナリオ 2 – 2 人のユーザーの認証情報のローテーション

2 つ目のシナリオでは、2 つのデータベースユーザー認証情報 (Alice と Bob) が Secrets Manager によって定期的にローテーションされます。アプリケーションコンテナは Fargate クラスターで実行されます。最初のデータベース接続が確立されると、アプリケーションコンテナは最初のユーザー (Alice) の Aurora データベース認証情報を取得します。その後、Secrets Manager キャッシュコンポーネントは、将来の接続確立に備えて認証情報をキャッシュします。

ユーザーと認証情報は 2 つありますが、Secrets Manager によって管理されるアクティブな認証情報は 1 つだけです。この場合、キャッシュコンポーネントは定期的に有効期限が切れ、最新の認証情報を取得します。Secrets Manager のローテーション期間がキャッシュタイムアウトよりも長い場合、キャッシュコンポーネントは 2 人目のユーザー (Bob) のローテーションされた認証情報を取得します。例えば、キャッシュの有効期限が分単位、ローテーション期間が日単位で測定されている場合、キャッシュコンポーネントは定期的なキャッシュ更新の一部として新しい認証情報を取得します。この方法では、各ユーザーの認証情報が 1 回の Secrets Manager ローテーションでアクティブになるため、ダウンタイムが最小限に抑えられます。

自動化とスケール

AWS CloudFormation を使用してこのパターンをデプロイするには、インフラストラクチャをコードとして使用します。これにより、アプリケーションコンテナのビルドと作成、Fargate タスクの作成、Fargate へのコンテナのデプロイ、Aurora による Secrets Manager のセットアップと設定が行われます。デプロイの手順については、readme ファイルを参照してください。

ツール

ツール

- AWS Secrets Manager を使用すると、シークレットを取得するための Secrets Manager への API コールを使用して、パスワードを含むハードコードされた認証情報を置き換えることができます。Secrets Manager は、シークレットを短期のシークレットに置き換えることが可能となり、侵害されるリスクが大幅に減少します。
- Docker は、開発者があらゆるアプリケーションを軽量でポータブルかつ自給自足のコンテナとして梱包、出荷、実行できるよう支援します。

コード

Python コードサンプル

このパターンでは、Secrets Manager の Python クライアント側キャッシュコンポーネントを使用して、データベース接続を確立する際に認証情報を取得します。クライアント側のキャッシュコンポーネントにより、毎回 Secrets Manager にアクセスする必要がなくなります。

これで、ローテーション期間が経過すると、キャッシュされた認証情報は期限切れになり、データベースに接続すると認証エラーが発生します。MySQL の場合、認証エラーコードは 1045 です。この例では MySQL 用に Amazon Aurora を使用していますが、PostgreSQL などの別のエンジンを使用することもできます。認証エラーが発生すると、データベース接続例外処理コードがエラーをキャッチします。次に、Secrets Manager のクライアント側キャッシュコンポーネントに、シークレットを更新し、再認証してデータベース接続を再確立するように通知します。PostgreSQL または別のエンジンを使用している場合は、対応する認証エラーコードを調べる必要があります。

これで、コンテナアプリケーションは、コンテナを再起動しなくても、ローテーションされたパスワードでデータベースパスワードを更新できます。

データベース接続を処理するアプリケーションコードに次のコードを記述してください。この例では Django を使用しており、接続用のデータベースラッパーを使用してデータベースバックエンドをサブクラス化します。別のプログラミング言語またはデータベース接続ライブラリを使用している場合は、使用しているデータベース接続ライブラリを参照して、データベース接続取得をサブクラス化する方法を確認してください。

```
    def get_new_connection(self, conn_params):
        try:
            logger.info("get connection")
            databasecredentials.get_conn_params_from_secrets_manager(conn_params)
            conn =super(DatabaseWrapper,self).get_new_connection(conn_params)
            return conn
        except MySQLdb.OperationalError as e:
            error_code=e.args[0]
            if error_code!=1045:
                raise e

            logger.info("Authentication error. Going to refresh secret and try again.")
            databasecredentials.refresh_now()
            databasecredentials.get_conn_params_from_secrets_manager(conn_params)
            conn=super(DatabaseWrapper,self).get_new_connection(conn_params)
            logger.info("Successfully refreshed secret and established new database
connection.")
            return conn
```

AWS CloudFormation と Python コード

- https://github.com/aws-samples/aws-secrets-manager-credential-rotation-without-container-restart

エピック

認証情報のローテーション中もアプリケーションの可用性を維持

タスク	説明	必要なスキル
キャッシュコンポーネントをインストールします。	Python 用の Secrets Manager クライアント側キャッシュコンポーネントをダウンロードしてインストールします。ダウンロードリンクについては、「関連リソース」セクションを参照してください。	開発者
作業用認証情報をキャッシュしてください。	Secrets Manager のクライアント側キャッシュコンポーネントを使用して、作業用認証情報をローカルにキャッシュします。	開発者
データベース接続で不正エラーが発生した場合は、アプリケーションコードを更新して認証情報を更新してください。	Secrets Manager を使用してデータベース認証情報を取得および更新するようにアプリケーションコードを更新します。不正なエラーコードを処理するロジックを追加し、新しくローテーションされた認証情報を取得します。「Python コードサンプル」セクションを参照してください。	開発者

関連リソース

Secrets Manager シークレットの作成

- 「AWS KMS でキーを作成」
- AWS Secrets Manager でのシークレットの作成と管理

Amazon Aurora クラスターの作成

- Amazon RDS DB インスタンスの作成

Amazon ECS コンポーネントの作成

- 従来のコンソールを使用したクラスターの作成
- Docker イメージの作成
- プライベートリポジトリを作成する
- Amazon ECR プライベートレジストリ
- Docker イメージをプッシュする
- Amazon ECS のタスク定義
- クラシックコンソール内の Amazon ECS サービスの作成

Secrets Manager のクライアント側キャッシュコンポーネントのダウンロードとインストール

- Python キャッシュクライアント

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon ECS Anywhere を使用して Amazon WorkSpaces で Amazon ECS タスクを実行する

作成者：Akash Kumar (AWS)

概要

Amazon Elastic Container Service (Amazon ECS) Anywhere は、Amazon Web Services (AWS) マネージドインフラストラクチャやカスタマーマネージドインフラストラクチャを含むあらゆる環境の Amazon ECS タスクのデプロイをサポートします。これは、クラウド上で稼働し、常に最新の状態に保たれる、完全に AWS マネージド型のコントロールプレーンを使用しながら行うことができます。

多くの場合、企業はコンテナベースのアプリケーションの開発に Amazon WorkSpaces を使用します。このため、ECS タスクをテストして実行するには、Amazon Elastic Compute Cloud (Amazon EC2) または Amazon ECS クラスターの AWS Fargate が必要でした。Amazon ECS Anywhere を使用することで、Amazon WorkSpaces を外部インスタンスとして ECS クラスターに直接追加できるようになり、タスクを直接実行できるようになりました。これにより、Amazon WorkSpaces のローカルの ECS クラスターを使用してコンテナをテストできるため、開発時間を短縮できます。コンテナアプリケーションのテストに EC2 または Fargate インスタンスを使用するコストを節約することもできます。

このパターンは、Amazon ECS Anywhere を使用して ECS タスクを Amazon WorkSpaces にデプロイする方法を示しています。ECS クラスターをセットアップし、AWS Directory Service Simple AD を使用して WorkSpaces を起動します。次に、サンプルの ECS タスクは WorkSpaces で NGINX を起動します。

前提条件と制限

- アクティブな AWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI)
- マシンに設定されている AWS 認証情報

アーキテクチャ

ターゲットテクノロジースタック

- 仮想プライベートクラウド (VPC)

- Amazon ECS クラスター
- Amazon WorkSpaces
- Simple AD を使用した AWS Directory Service

ターゲット アーキテクチャ

このアーキテクチャには、以下のサービスとリソースが含まれます。

- カスタム VPC にパブリックサブネットとプライベートサブネットを持つ ECS クラスター
- Amazon WorkSpaces へのユーザーアクセスを提供するための VPC 内の Simple AD
- Simple AD を使用して VPC にプロビジョニングされた Amazon WorkSpaces
- Amazon WorkSpaces をマネージドインスタンスとして追加するために AWS Systems Manager がアクティブ化されました
- Amazon ECS と AWS Systems Manager Agent (SSM Agent) を使用して、Amazon WorkSpaces がSystems Manager と ECS クラスターに追加されました
- ECS クラスターの WorkSpaces で実行する ECS タスクの例

ツール

- AWS Directory Service Simple Active Directory (Simple AD) は、Samba 4 Active Directory 互換サーバーを搭載したスタンドアロンの管理対象ディレクトリです。Simple AD では、AWS Managed Microsoft AD が提供する機能のサブセットを使用できます。例えば、ユーザーを管理する機能や Amazon EC2 インスタンスに安全に接続する機能などがあります。
- Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理をサポートする、高速でスケーラブルなコンテナ管理サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。
- Amazon WorkSpaces は、WorkSpaces として知られる、クラウドベースの仮想 Microsoft Windows または Amazon Linux デスクトップをユーザーにプロビジョニングするのに役立ちま

す。WorkSpaces は、ハードウェアの調達とデプロイ、または複雑なソフトウェアのインストールの必要性を排除します。

エピック

ECS クラスターをセットアップ

タスク	説明	必要なスキル
ECS クラスターを作成して設定する。	ECS クラスターを作成するには、次の手順を含む AWS ドキュメントの指示に従います。 • [クラスターの互換性の選択] で、[ネットワーキングのみ] を選択します。これにより、Amazon WorkSpace が ECS クラスターの外部インスタンスとしてサポートされます。 • 新しい VPC を作成することを選択します。	クラウドアーキテクト

Amazon WorkSpaces を起動する

タスク	説明	必要なスキル
Simple AD をセットアップし、Amazon WorkSpaces を起動します。	新しく作成した VPC に Simple AD ディレクトリをプロビジョニングして Amazon WorkSpaces を起動するには、AWS ドキュメントの指示に従ってください。	クラウドアーキテクト

ハイブリッド環境用に AWS Systems Manager をセットアップする

タスク	説明	必要なスキル
添付のスクリプトをダウンロードしてください。	ローカルマシンで、「添付ファイル」セクションにある `ssm-trust-policy.json` および `ssm-activation.json` ファイルをダウンロードします。	クラウドアーキテクト

IAM ロールを追加します。	ビジネス要件に基づいて環境変数を追加します。
	`export AWS_DEFAULT_REGION=${AWS_REGION_ID}`
	`export ROLE_NAME=${ECS_TASK_ROLE}`
	`export CLUSTER_NAME=${ECS_CLUSTER_NAME}`
	`export SERVICE_NAME=${ECS_CLUSTER_SERVICE_NAME}`
	以下のコマンドを実行してください。
	`aws iam create-role --role-name $ROLE_NAME --assume-role-policy-document file://ssm-trust-policy.json`
		クラウドアーキテクト

AmazonSSMManagedInstanceCore ポリシーを IAM ロールに追加します。	以下のコマンドを実行してください。 `aws iam attach-role-policy --role-name $ROLE_NAME --policy-`	クラウドアーキテクト

タスク	説明	必要なスキル
	`arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore`	
[AmazonEC2ContainerServiceforEC2Role] ポリシーを IAM ロールに追加します。	以下のコマンドを実行してください。 `aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role`	クラウドアーキテクト
IAM ロールを検証します。	次のコマンドを実行して、IAM ロールを検証します。 `aws iam list-attached-role-policies --role-name $ROLE_NAME`	クラウドアーキテクト
Systems Manager を起動します。	以下のコマンドを実行してください。 `aws ssm create-activation --iam-role $ROLE_NAME | tee ssm-activation.json`	クラウドアーキテクト

ECS クラスターにWorkSpaces を追加

タスク	説明	必要なスキル
WorkSpaces に接続します。	ワークスペースに接続して設定するには、AWS ドキュメントの指示に従ってください。	アプリ開発者
ecs-anywhere インストールスクリプトをダウンロードします。	コマンドプロンプトで、次のコマンドを入力します。 `curl -o "ecs-anywhere-install.sh" "https://amazon-ecs-agent-packages-preview.s3.us-east-1.amazonaws.com/ecs-anywhere-install.sh" && sudo chmod +x ecs-anywhere-install.sh`	アプリ開発者
シェルスクリプトの整合性をチェックしてください。	(オプション) 次のコマンドを実行します。 `curl -o "ecs-anywhere-install.sh.sha256" "https://amazon-ecs-agent-packages-preview.s3.us-east-1.amazonaws.com/ecs-anywhere-install.sh.sha256" && sha256sum -c ecs-anywhere-install.sh.sha256`	アプリ開発者
Amazon Linux に EPEL リポジトリを追加します。	Enterprise Linux (EPEL) リポジトリに追加パッケージを追加するには、コマンド sudo	アプリ開発者

タスク	説明	必要なスキル
	`amazon-linux-extras install epel -y` を実行します。	
Amazon ECS Anywhere にインストールできます。	インストールスクリプトを実行するには、次のコマンドを使用します。 `sudo ./ecs-anywhere-install.sh --cluster $CLUSTER_NAME --activation-id $ACTIVATION_ID --activation-code $ACTIVATION_CODE --region $AWS_REGION`	
ECS クラスターのインスタンス情報を確認します。	Systems Manager と ECS クラスターインスタンスの情報を確認し、WorkSpaces がクラスターに追加されたことを確認するには、ローカルマシンで次のコマンドを実行します。 `aws ssm describe-instance-information" && "aws ecs list-container-instances --cluster $CLUSTER_NAME`	アプリ開発者

WorkSpaces に ECS タスクを追加

タスク	説明	必要なスキル
タスク実行 IAM ロールを作成するには	「添付ファイル」セクションから task-execution-assume-role.json および external-task-definition.json をダウンロードします。 ローカルマシンで次のコマンドを実行します。 `aws iam --region $AWS_DEFAULT_REGION create-role --role-name $ECS_TASK_EXECUTION_ROLE --assume-role-policy-document file://task-execution-assume-role.json`	クラウドアーキテクト
ポリシーを実行ロールに追加します。	以下のコマンドを実行してください。 `aws iam --region $AWS_DEFAULT_REGION attach-role-policy --role-name $ECS_TASK_EXECUTION_ROLE --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy`	クラウドアーキテクト
タスクロールを作成します。	以下のコマンドを実行してください。	クラウドアーキテクト

タスク	説明	必要なスキル
	`aws iam --region $AWS_DEFAULT_REGION create-role --role-name $ECS_TASK_EXECUTION_ROLE --assume-role-policy-document file://task-execution-assume-role.json`	
タスク定義をクラスターに登録します。	ローカルマシンで次のコマンドを実行します。 `aws ecs register-task-definition --cli-input-json file://external-task-definition.json`	クラウドアーキテクト
タスクを実行します。	ローカルマシンで次のコマンドを実行します。 `aws ecs run-task --cluster $CLUSTER_NAME --launch-type EXTERNAL --task-definition nginx`	クラウドアーキテクト

タスク	説明	必要なスキル

タスクの実行状態を検証します。	タスク ID を取得するには、次のコマンドを実行します。
	`export TEST_TASKID=$(aws ecs list-tasks --cluster $CLUSTER_NAME | jq -r '.taskArns[0]')`
	タスク ID を使用して、次のコマンドを実行します。
	`aws ecs describe-tasks --cluster $CLUSTER_NAME --tasks ${TEST_TASKID}`
		クラウドアーキテクト

WorkSpace でタスクを確認します。	NGINX が WorkSpace で実行されていることを確認するには、コマンド `curl http://localhost:8080` を実行します。	アプリ開発者

関連リソース

- ECS クラスター
- ハイブリッド環境の設定
- Amazon WorkSpaces
- Simple AD

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon EC2 Linux インスタンスで ASP.NET Core ウェブ API Docker コンテナを実行する

作成者：Vijai Anand Ramalingam (AWS) と Sreelaxmi Pai (AWS)

概要

このパターンは、Amazon Web Services (AWS) クラウドでアプリケーションのコンテナ化を始めるユーザーを対象としています。クラウド上でアプリケーションのコンテナ化を始める場合、通常、コンテナオーケストレーションプラットフォームは設定されていません。このパターンは、複雑なコンテナオーケストレーションインフラストラクチャを必要とせずに、コンテナ化されたアプリケーションをテストするためのインフラストラクチャを AWS にすばやくセットアップするのに役立ちます。

モダナイゼーションの最初のステップでは、アプリケーションを変換します。レガシー .NET Framework アプリケーションの場合は、まず、ランタイムを ASP.NET Core に変更する必要があります。次に、以下の操作を実行します。

- Docker コンテナイメージを作成する
- ビルドされたイメージを使用して Docker コンテナを実行する
- Amazon Elastic Container Service (Amazon ECS) またはAmazon Elastic Kubernetes Service (Amazon EKS) のような コンテナオーケストレーションプラットフォームにアプリケーションをデプロイする前に、アプリケーションを検証します。

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) Linux インスタンスでの最新のアプリケーション開発の構築、実行、検証の各側面を扱います。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント
- このパターンの AWS リソースを作成するのに十分なアクセス権を持つ AWS Identity and Access Management (IAM) ロール
- Visual Studio Community 2022 以降をダウンロードしてインストール済み
- ASP.NET Core にモダナイズされた .NET Framework プロジェクト
- GitHub リポジトリ

製品バージョン

- Visual Studio Community 2022 以降

アーキテクチャ

ターゲット アーキテクチャ

このパターンでは、AWS CloudFormation テンプレートを使用して、次の図に示す高可用性アーキテクチャを作成します。Amazon EC2 Linux インスタンスはプライベートサブネットで起動されます。AWS Systems Manager Session Manager は、プライベート Amazon EC2 Linux インスタンスにアクセスし、Docker コンテナで実行されている API をテストするために使用されます。

1. セッションマネージャーから Linux インスタンスにアクセスする

ツール

AWS サービス

- AWS コマンドラインインターフェイス — AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールで、コマンドラインシェルのコマンドで AWS サービスとやり取りします。最小限の設定で、ブラウザベースの AWS マネジメントコンソールで提供される機能と同等の機能を実装する AWS CLI コマンドを実行できます。
- AWS マネジメントコンソール – AWS マネジメントコンソールは、AWS リソースを管理するためのサービスコンソールの広範なコレクションで構成され、そのコレクションを参照するウェブアプリケーションです。最初にサインインすると、コンソールのホームページが表示されます。各サービスコンソールにアクセスできるホームページは、AWS に関連するタスクを実行するために必要な情報に 1 か所からアクセスできます。
- AWS Systems Manager Session Manager — Session Manager はフルマネージド型の AWS Systems Manager 機能です。Session Manager を使用すれば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス を管理できます。Session Manager は、受信ポートを開いたり、踏み台ホストを維持したり、SSH キーを管理したりすることなく、セキュアで監査可能なノード管理を提供します。

その他のツール

- Visual Studio 2022 — Visual Studio 2022 は統合開発環境 (IDE) です。
- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。

コード

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
 WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY ["DemoNetCoreWebAPI/DemoNetCoreWebAPI.csproj", "DemoNetCoreWebAPI/"]
RUN dotnet restore "DemoNetCoreWebAPI/DemoNetCoreWebAPI.csproj"
COPY . .
WORKDIR "/src/DemoNetCoreWebAPI"
RUN dotnet build "DemoNetCoreWebAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "DemoNetCoreWebAPI.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "DemoNetCoreWebAPI.dll"]
```

エピック

ASP.NET Core ウェブ API を開発する

タスク	説明	必要なスキル
Visual Studio を使用して ASP.NET Core ウェブ API のサンプルを作成します。	ASP.NET Core ウェブ API のサンプルを作成するには、次の手順に従います。 1. Visual Studio 2022 を開きます。	アプリ開発者

タスク	説明	必要なスキル

	2. [新しいプロジェクトを作成する] を選択します。
	3. ASP.NET Core Web API プロジェクトテンプレートを選択し、[次へ] を選択します。
	4. プロジェクト名として DemonetCoreWebAPI と入力し、[次へ] を選択します。
	5. [作成] を選択します。
	6. プロジェクトをローカルで実行するには、<F5> キーを押します。
	7. デフォルトの WeatherForecast API エンドポイントが Swagger を使用して結果を返していることを確認します。
	8. コマンドプロンプトを開き、.csproj プロジェクトフォルダに移動し、次のコマンドを実行して新しいウェブ API を GitHub リポジトリにプッシュします。
	`git add --all`
	`git commit -m “Initial Version”`
	`git push`	

タスク	説明	必要なスキル
Dockerfile を作成します。	Dockerfile を作成するには、以下のいずれかの操作を行います。 • 「コード」セクションにあるサンプル Docker ファイルを使用して、Docker ファイルを手動で作成します。要件に基づいて、適切な .NET ベースイメージを選択します。.NET および ASP.NET Core 関連のイメージについては、「Docker ハブ」を参照してください。 • Visual Studio と Docker Desktop を使用して Dockerfile を作成します。ソリューションエクスプローラーで、プロジェクトを右クリックし、[追加]-> [Docker サポート] を選択します。[ターゲット OS] には、Linux を選択します。新しい Dockerfile がソリューションファイル (.sln) と同じパスにあることを確認してください。 変更を GitHub リポジトリにプッシュするには、次のコマンドを実行します。 `git add --all`	アプリ開発者

タスク	説明	必要なスキル
	`git commit -m “Dockerfile added”` `git push`	

Amazon EC2 Linux インスタンスをセットアップする。

タスク	説明	必要なスキル
インフラストラクチャを設定します。	AWS CloudFormation テンプレートを起動して、以下を含むインフラストラクチャを作成します。 • AWS VPC クイックスタートを使用し、2 つのアベイラビリティーゾーンに 2 つのパブリックサブネットと 2 つのプライベートサブネットを持つ仮想プライベートクラウド (VPC)。 • AWS Systems Manager を有効にするために必要な IAM ロール。 • プライベートサブネットの 1 つに、最新の SSM エージェントがインストールされた Amazon Linux 2 デモインスタンスがあります。このインスタンスにはインターネットからの直接接続はありませんが、踏み台ホストを必要とせずに AWS Systems Manager Session	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	Manager を使用してセキュアにアクセスできます。 **Note** Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。 踏み台ホストを必要とせずに Session Manager を使用してプライベート Amazon EC2 インスタンスにアクセスする方法の詳細については、「踏み台のない世界へ」というブログ投稿を参照してください。	

タスク	説明	必要なスキル
Amazon EC2 Linux インスタンスにログインします。	プライベートサブネットの Amazon EC2 Linux インスタンスに接続するには、以下の手順に従います。 1. Amazon EC2 コンソールを開きます。 2. ナビゲーションペインで、[インスタンス] を選択します。 3. Amazon Linux 2 デモインスタンスを選択し、[Connect] を選択します。 **Note** Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。 4. [Session Manager] を選択します。 5. [接続] を選択して、新しいターミナルウィンドウを開きます。 6. 以下のコマンドを実行してください。 `sudo su`	アプリ開発者

タスク	説明	必要なスキル

Docker をインストールして実行します。	Amazon EC2 Linux インスタンスに Docker をインストールして起動するには、以下の手順に従います。
	
	1. Docker をインストールするには、以下のコマンドを実行します。
	
	`yum install -y docker`
	
	2. 次のコマンドを実行して、Docker サービスをスタートします。
	
	`service docker start`
	
	3. Docker のインストールを検証するには、以下のコマンドを実行します。
	
	`docker info`
		アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

GitHub をインストールして、リポジトリのクローンを作成します。	Amazon EC2 Linux インスタンスに Git をインストールし、GitHub からリポジトリをクローンするには、以下の手順に従います。
	
	1. Git をインストールするには、次のコマンドを実行します。
	
	`yum install git -y`
	
	2. リポジトリのクローンを作成するには、次のコマンドを実行します。
	
	`git clone https://github.com/<username>/<repo-name>.git`
	
	3. Docker ファイルに移動するには、次のコマンドを実行します。
	
	`cd <repo-name>/DemoNetCoreWebAPI/`
		アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

Docker コンテナをローカルで構築して実行します。	Docker イメージを構築し、Amazon EC2 Linux インスタンス内でコンテナを実行するには、次の手順に従います。
	
	1. Docker イメージを作成するには、次のコマンドを実行します。
	
	`docker build -t aspnetcorewebapiimage -f Dockerfile .`
	
	2. すべての Docker イメージを取得するには、次のコマンドを実行します。
	
	`docker images`
	
	3. コンテナを作成して実行するには、次のコマンドを実行します。
	
	`docker run -d -p 80:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage`
		アプリ開発者、AWS 管理者、AWS DevOps

ウェブ API をテストする

タスク	説明	必要なスキル
curl コマンドを使用して、ウェブ API をテストします。	ウェブ API をテストするには、次のコマンドを使用します。 `curl -X GET "http://localhost/WeatherForecast" -H  "accept: text/plain"` API レスポンスを確認してください。 **Note** ローカルで実行しているときに、Swagger から各エンドポイントの curl コマンドを取得できます。	アプリ開発者

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースを削除します。	スタックを削除してすべてのリソースを削除します。これにより、利用していないサービスに料金を支払うことはありません。	AWS 管理者、AWS DevOps

関連リソース

- 「PuTTY を使用した Windows から Linux インスタンスへの接続」
- 「ASP.NET コアを使用してウェブ API を作成する」
- 「踏み台のない世界へ」

AWS Fargate を使用してメッセージ駆動型の大規模なワークロード実行する

作成者: Stan Zubarev (AWS)

概要

このパターンでは、コンテナと AWS Fargate を使用して、メッセージ主導型の大規模なワークロードを AWS クラウドで実行する方法を説明しています。

コンテナを使用してプロセスデータを処理した場合、アプリケーションが処理するデータ量が関数ベースのサーバーレスコンピューティングサービスの制限を超える場合に役立ちます。例えば、アプリケーションが AWS Lambda から提供されるよりも多くの計算能力や処理時間を必要とする場合、Fargate を使用すればパフォーマンスを向上することができます。

次のセットアップ例では、TypeScript の AWS Cloud Development Kit (AWS CDK) を使用して、以下のリソースを AWS クラウドに設定してデプロイします。

- Fargate サービス
- Amazon Simple Queue Service (Amazon SQS) キュー
- Amazon DynamoDB テーブル
- Amazon CloudWatch ダッシュボード

Fargate サービスは Amazon SQS キューからメッセージを受信して処理し、Amazon DynamoDB テーブルに保存します。CloudWatch ダッシュボードを使用して、処理された Amazon SQS メッセージの数および Fargate によって作成された DynamoDB アイテムの数をモニタリングできます。

> **Note**
>
> このパターンのサンプルコードを使用して、イベント駆動型サーバーレスアーキテクチャでより複雑なデータ処理ワークロードを構築することもできます。詳細については、「AWS Fargate を使用して、イベント駆動型でスケジュール済みの大規模なワークロードを実行する」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI) の最新バージョンをローカルマシンにインストールおよび設定済み
- Gitをローカルマシンにインストールおよび設定済み
- AWS CDKをローカルマシンにインストールおよび設定済み
- Goをローカルマシンにインストールおよび設定済み
- Dockerをローカルマシンにインストールおよび設定済み

アーキテクチャ

ターゲットテクノロジースタック

- Amazon SQS
- AWS Fargate
- Amazon DynamoDB

ターゲット アーキテクチャ

次の図は、Fargate を使用して AWS クラウドでメッセージ駆動型の大規模なワークロードを実行するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. Fargate サービスは、Amazon SQS ロングポーリングを使用して、Amazon SQS キューからのメッセージを受信します。
2. 次に、Fargate サービスは Amazon SQS キューからメッセージを処理して、DynamoDB テーブルに保存します。

自動化とスケール

Fargate タスクカウントを自動スケーリングするには、Amazon Elastic Container Service (Amazon ECS) サービスの自動スケーリングを設定できます。アプリケーションの Amazon SQS キューに表示されるメッセージ数に基づいてスケーリングポリシーを設定するのがベストプラクティスです。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon SQSに基づくスケーリング」を参照してください。

ツール

AWS サービス

- AWS Fargate を使用すると、サーバーや Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの管理を行わずにコンテナを実行することができます。Amazon Elastic Container Service (Amazon ECS) と合わせて使用されます。
- Amazon Simple Queue Service (Amazon SQS) は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。

コード

このパターンのコードは GitHub 内の「sqs-fargate-ddb-cdk-go」リポジトリで利用できます。

エピック

AWS CDK を使用してリソースを作成し、デプロイします。

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	次のコマンドを実行して、GitHub sqs-fargate-ddb-cdk-go リポジトリをローカルマシンにクローンします。 `git clone https://github.com/aws-samples/sqs-fargate-ddb-cdk-go.git`	アプリ開発者

タスク	説明	必要なスキル
AWS CLI が正しい AWS アカウントに設定されていること、および AWS CDK が必要な権限を所有していることを確認します。	AWS CLI の設定が正しいかどうかを確認するには、次の Amazon Simple Storage Service (Amazon S3) ls コマンドを実行します。 `aws s3 ls` また、この手順では、AWS CDK が AWS アカウント内でインフラストラクチャをプロビジョニングする権限を所有している必要があります。必要なアクセス権限を付与するには、AWS CLI で名前付きの AWS プロファイルを作成し、AWS_PROFILE 環境変数としてエクスポートする必要があります。 **Note** AWS アカウントで AWS CDK を使用したことがない場合は、まず必要な AWS CDK リソースをプロビジョニングする必要があります。詳細については、AWS CDK v2 開発者ガイドの「ブートストラップ」を参照してください。	アプリ開発者

タスク	説明	必要なスキル

AWS CDK スタックを AWS アカウントにデプロイします。	1. 次の AWS CLI コマンドを実行して、コンテナイメージを作成します。
	`docker build -t go-fargate .`
	2. 次のコマンドを実行して、AWS CDK ディレクトリを開きます。
	`cd cdk`
	3. 必要な npm モジュールをインストールするには、次のコマンドを実行します。
	`npm i`
	4. 次のコマンドを実行して、AWS CDK パターンを AWS アカウントにデプロイします。
	`cdk deploy --profile ${AWS_PROFILE}`
		アプリ開発者

セットアップをテストする

タスク	説明	必要なスキル
Amazon SQS キューにテストメッセージを送信します。	手順については、「Amazon SQS デベロッパーガイド」の「キューへのメッセージの送信 (コンソール)」を参照してください	アプリ開発者

タスク	説明	必要なスキル
	Amazon SQS メッセージのサンプルをテストする `{ "message": "hello, Fargate" }`	
テストメッセージが Fargate サービスの CloudWatch ログに表示されていることを検証します。	Amazon ECS 開発者ガイドの「CloudWatch Logs を表示する」の手順に従ってください。go-service-cluster ECS クラスターの go-fargate-service ロググループのログを必ず確認してください。	アプリ開発者
テストメッセージが DynamoDB テーブルに表示されていることを検証します。	1. DynamoDB コンソールを開きます。 2. 左のナビゲーションペインで、[テーブル] を選択します。次に、リストから sqs-fargate-ddb-table というテーブルを選択します。 3. [テーブルアイテムの探索] を選択します。 4. テストメッセージが [返された項目] リストに表示されていることを確認します。	アプリ開発者

タスク	説明	必要なスキル
Fargate サービスがメッセージを CloudWatch Logs に送信していることを検証します。	1. CloudWatch コンソールを開きます。 2. 左のナビゲーションペインの [ダッシュボード] を選択します。 3. カスタムダッシュボードリストで、go-service-dashboard という名前のダッシュボードを選択します。 4. テストメッセージがログに表示されることを確認します。 **Note** AWS CDK は、AWS アカウントに CloudWatch ダッシュボードを自動的に作成します。	アプリ開発者

クリーンアップ

タスク	説明	必要なスキル
AWS CDK スタックを削除します。	1. 次のコマンドを実行して、AWS CLI で AWS CDK ディレクトリを開きます。 `cd cdk`	アプリ開発者

タスク	説明	必要なスキル
	2. 次のコマンドを実行して、AWS CDK スタックを削除します。 `cdk destroy --profile ${AWS_PROFILE}`	
AWS CDK スタックが削除されていることを検証します。	次のコマンドを実行して、スタックが削除されたことを確認します。 `aws cloudformation list-stacks --query \ "StackSummaries[?contains(StackName,'SqsFargate')].StackStatus" \ --profile ${AWS_PROFILE}` スタックが削除された場合、コマンド出力で返される `StackStatus` の値は `DELETE_COMPLETE` です。 詳細については、AWS CloudFormation ユーザーガイドの「 の CloudFormation スタックオペレーションコマンドの例 AWS CLI 」および PowerShell」を参照してください。 AWS CloudFormation	アプリ開発者

関連リソース

- 「AWS CLI の設定」(バージョン 2 用 AWS CLI ユーザーガイド)
- 「API リファレンス」(AWS CDK API リファレンス)
- AWS SDK for Go v2 (Go ドキュメント)

AWS Fargate 搭載の Amazon EKS で Amazon EFS を使用して、永続的なデータストレージでステートフルワークロードを実行する

作成者: Ricardo Morais (AWS)、Rodrigo Bersa (AWS)、Lucio Pereira (AWS)

概要

このパターンは、AWS Fargate を使用してコンピューティングリソースをプロビジョニングすることで、Amazon Elastic Kubernetes Service (Amazon EKS) で実行されているコンテナのストレージデバイスとして Amazon Elastic File System (Amazon EFS) を有効にするためのガイダンスを提供します。 EFS

このパターンで説明する設定は、セキュリティのベストプラクティスに従い、デフォルトで保管時と転送時のセキュリティを提供します。Amazon EFS ファイルシステムを暗号化するには、AWS Key Management Service (AWS KMS) キーを使用しますが、KMS キーの作成プロセスをディスパッチするキーエイリアスも指定できます。

このパターンの手順に従って、概念実証 (PoC) アプリケーションの名前空間と Fargate プロファイルを作成し、Kubernetes クラスターを Amazon EFS と統合するために使用される Amazon EFS コンテナストレージインターフェイス (CSI) ドライバーをインストールし、ストレージクラスを設定し、PoC アプリケーションをデプロイできます。これらの手順により、Fargate 上で実行中の Amazon EFS ファイルシステムが複数の Kubernetes ワークロード間で共有されます。このパターンには、これらの手順を自動化するスクリプトが付属しています。

このパターンは、コンテナ化されたアプリケーションにデータ永続性が必要で、スケーリングオペレーション中のデータ損失を回避したい場合に使用できます。以下に例を示します。

- DevOps ツール – 一般的なシナリオは、継続的インテグレーションと継続的デリバリー (CI/CD) 戦略を開発することです。この場合、Amazon EFS を共有ファイルシステムとして使用して、CI/CD ツールのさまざまなインスタンス間で設定を保存、または CI/CD ツールのさまざまなインスタンス間のパイプラインステージ用のキャッシュ (例、Apache Maven リポジトリ) を保存できます。
- ウェブサーバー – 一般的なシナリオは、HTTP ウェブサーバーとして Apache を使用することです。Amazon EFS を共有ファイルシステムとして使用して、Web サーバーのさまざまなインスタンス間で共有される静的ファイルを保存できます。このシナリオ例では、静的ファイルが Docker イメージにベイクされるのではなく、変更がファイルシステムに直接適用されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Kubernetes バージョン 1.17 以降の既存の Amazon EKS クラスター (バージョン 1.27 までテスト済み)
- Kubernetes StorageClass をバインドし、ファイルシステムを動的にプロビジョニングする既存の Amazon EFS ファイルシステム
- クラスター管理権限
- 目的の Amazon EKS クラスターを指すように設定されたコンテキスト

制限

- Fargate で Amazon EKS を使用する際には、考慮すべき制限がいくつかあります。例えば、DaemonSets や特権コンテナなどの一部の Kubernetes コンストラクトの使用はサポートされていません。Fargate の制限の詳細については、Amazon EKS ドキュメントの「AWS Fargate に関する考慮事項」を参照してください。
- このパターンで提供されるコードは、Linux または macOS を実行中のワークステーションをサポートします。

製品バージョン

- AWS コマンドラインインターフェイス (AWS CLI)バージョン 2 以降
- Amazon EFS CSI ドライバーバージョン 1.0 以降 (バージョン 2.4.8 までテスト済み)
- eksctl バージョン 0.24.0 以降 (バージョン 0.158.0 までテスト済み)
- jq バージョン 1.6 以降
- kubectl バージョン 1.17 以降 (バージョン 1.27 までテスト済み)
- Kubernetes バージョン 1.17 以降 (バージョン 1.27 までテスト済み)

アーキテクチャ

ターゲットアーキテクチャは、次のインフラストラクチャで構成されます。

- 仮想プライベートクラウド (VPC)
- 2 つのアベイラビリティーゾーン

- インターネットアクセスを提供する NAT ゲートウェイを持つパブリックサブネット
- Amazon EKS クラスターと Amazon EFS マウントターゲット (マウントポイントとも呼ばれます) を持つプライベートサブネット
- VPC レベルでの Amazon EFS

Amazon EKS クラスターの環境インフラストラクチャは次のとおりです。

- 名前空間レベルで Kubernetes コンストラクトに対応する AWS Fargate プロファイル
- 以下を含む Kubernetes 名前空間：
  - アベイラビリティーゾーンに分散された 2 つのアプリケーションポッド
  - クラスターレベルで永続ボリューム (PV) にバインドされた 1 つの永続ボリュームクレーム (PVC)
- 名前空間内の PVC にバインドされ、クラスター外のプライベートサブネット内の Amazon EFS マウントターゲットを指すクラスター全体の PV

ツール

AWS サービス

- AWS コマンドラインインターフェイス (AWS CLI) は、コマンドラインから AWS サービスとやり取りするために使用できるオープンソースツールです。
- Amazon Elastic File System (Amazon EFS) は、AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。このパターンでは、Amazon EKS で使用するシンプルで、スケーラブルな完全マネージド型の共有ファイルシステムを提供します。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自のクラスターをインストールまたは運用することなく、AWS で Kubernetes を実行するのに役立ちます。
- AWS Fargate は Amazon EKS 用のサーバーレスコンピューティングエンジンです。Kubernetes アプリケーションのコンピュートリソースを作成および管理します。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。

- eksctl – これは Amazon EKS で Kubernetes クラスターを作成および管理するコマンドラインユーティリティです。
- kubectlは、Kubernetes クラスターに対してコマンドを実行するためのコマンドラインインターフェイスです。
- jq は JSON を解析するためのコマンドラインツールです。

コード

このパターンのコードは、AWS Fargate リポジトリを使用した Amazon EKS での Amazon EFS を使用した GitHub Persistence Configuration で提供されています。 EFS スクリプトは、このパターンのエピックセクションの順序に対応する `epic06`から `epic01` までのフォルダ内のエピック別に整理されます。

ベストプラクティス

ターゲットアーキテクチャには以下のサービスとコンポーネントが含まれており、AWS Well-Architected Framework のベストプラクティスに従います。

- Amazon EFS は、シンプルで、スケーラブル、伸縮性のあるフルマネージド型の NFS ファイルシステムです。これは、選択した Amazon EKS クラスターのプライベートサブネットに分散される、ポッドで実行中の PoC アプリケーションのすべてのレプリケーションの共有ファイルシステムとして使用されます。
- 各プライベートサブネットの Amazon EFS マウントターゲット。これにより、クラスターの仮想プライベートクラウド (VPC) 内のアベイラビリティーゾーンごとの冗長性が確保されます。
- Amazon EKS は Kubernetes ワークロードを実行します。前提条件セクションで説明したように、このパターンを使用する前に Amazon EKS クラスターをプロビジョニングする必要があります。
- AWS KMS は、Amazon EFS ファイルシステムに保存されているコンテンツを保存時に暗号化します。
- Fargate は、コンテナのコンピュートリソースを管理するため、お客様はインフラストラクチャに負担をかけずにビジネス要件に集中できます。Fargate プロファイルはすべてのプライベートサブネットに作成されます。クラスターの仮想プライベートクラウド (VPC) 内のアベイラビリティーゾーンごとに冗長性を提供します。
- Kubernetes Pods は、アプリケーションの異なるインスタンスによってコンテンツを共有、消費、および書き込むことができることを検証します。

エピック

Amazon EKS クラスターのプロビジョニング (オプション)

タスク	説明	必要なスキル
Amazon EKS クラスターを作成します。	**Note** クラスターが既にデプロイされている場合は、次のエピックに進みます。既存の AWS アカウントに Amazon EKS クラスターを作成します。GitHub ディレクトリで、いずれかのパターンを使用して、Terraform または eksctl を使用して Amazon EKS クラスターをデプロイします。詳細については、Amazon EKS ドキュメントの「Amazon EKS クラスターの作成」を参照してください。Terraform パターンには、Fargate プロファイルを Amazon EKS クラスターにリンクし、Amazon EFS ファイルシステムを作成し、Amazon EKS クラスターに Amazon EFS CSI ドライバーを	AWS 管理者、Terraform または eksctl 管理者、Kubernetes 管理者

タスク	説明	必要なスキル
	デプロイする方法を示す例もあります。	

タスク	説明	必要なスキル

環境変数をエクスポートします。	env.sh 「https://https:」スクリプトを実行します。これにより、次のステップで必要な情報が提供されます。
	
	`source ./scripts/env.sh`
	`Inform the AWS Account ID:`
	`<13-digit-account-id>`
	`Inform your AWS Region:`
	`<aws-Region-code>`
	`Inform your Amazon EKS Cluster Name:`
	`<amazon-eks-cluster-name>`
	`Inform the Amazon EFS Creation Token:`
	`<self-genereated-uuid>`
	
	まだ記載されていない場合は、次の CLI コマンドを使用して、上記でリクエストされたすべての情報を取得できます。
	
	`# ACCOUNT ID`
	`aws sts get-caller-identity --query "Account" --output text`
	
	`# REGION CODE`
	`aws configure get region`
	
	`# CLUSTER EKS NAME`
		AWS システム管理者

タスク	説明	必要なスキル

	`aws eks list-clusters --query "clusters" --output text`
	
	`# GENERATE EFS TOKEN uuidgen`	

Kubernetes 名前空間とリンクされた Fargate プロファイルを作成する

タスク	説明	必要なスキル
アプリケーションワークロード用の Kubernetes 名前空間と Fargate プロファイルを作成します。	Amazon EFS とインタラクトするアプリケーションワークロードを受信する名前空間を作成します。`create-k8s-ns-and-linked-fargate-profile.sh` スクリプトを実行します。カスタム名前空間名またはデフォルトの指定の名前空間 を使用できます`poc-efs-eks-fargate` 。 カスタムアプリケーション名前空間名の場合： `export $APP_NAMESPACE=<CUSTOM_NAME> ./scripts/epic01/create-k8s-ns-and-linked-fargate-profile.sh \ -c "$CLUSTER_NAME" -n "$APP_NAMESPACE"`	権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

	カスタムアプリケーション名前空間名がない場合：
	
	`./scripts/epic01/create-k8s-ns-and-linked-fargate-profile.sh \`
	`  -c "$CLUSTER_NAME"`
	
	ここで、$CLUSTER_NAME は Amazon EKS クラスターの名前です。-n <NAMESPACE> パラメータはオプションです。通知されない場合は、デフォルトで生成された名前空間名が提供されます。	

「Amazon EFS ファイルシステムの作成」

タスク	説明	必要なスキル
一意のトークンを生成します。	Amazon EFS では冪等オペレーションを保証する作成トークンが必要です (同じ作成トークンでオペレーションを呼び出しても効果はありません)。この要件を満たすには、使用可能な手法を使用して一意のトークンを生成する必要があります。たとえば、作成トークンとして使用する汎用一意識別子 (UUID) を生成できます。	AWS システム管理者

タスク	説明	必要なスキル
Amazon EFS ファイルシステムを作成します。	アプリケーションワークロードによって読み書きされるデータファイルを受信するファイルシステムを作成します。暗号化されたファイルシステムまたは暗号化されていないファイルシステムを作成できます。(ベストプラクティスとして、このパターンのコードはデフォルトで保存時の暗号化を有効化する暗号化システムを作成します。) 一意の対称 AWS KMS キーを使用して、ファイルシステムを暗号化できます。カスタムキーが指定されていない場合は、AWS マネージドキーが使用されます。 Amazon EFS 用の一意のトークンを生成した後、create-efs.sh スクリプトを使用して暗号化された、または暗号化されていない Amazon EFS ファイルシステムを作成します。 KMS キーを使用しないで保存時に暗号化する場合: `./scripts/epic02/create-efs.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN"`	AWS システム管理者

タスク	説明	必要なスキル

	ここで、$CLUSTER_NAME はAmazon EKS クラスターの名前、$EFS_CREATION_TOKEN はファイルシステムの一意の作成トークンです。
	
	KMS キーを使用して保存時に暗号化する場合:
	
	`./scripts/epic02/create-efs.sh \`
	`    -c "$CLUSTER_NAME" \`
	`    -t "$EFS_CREATION_TOKEN" \`
	`    -k "$KMS_KEY_ALIAS"`
	
	ここで、$CLUSTER_NAME はAmazon EKS クラスターの名前、$EFS_CREATION_TOKEN はファイルシステムの一意の作成トークン、$KMS_KEY_ALIAS は KMS キーのエイリアスです。
	
	暗号化しない場合:
	
	`./scripts/epic02/create-efs.sh -d \`
	`    -c "$CLUSTER_NAME" \`
	`    -t "$EFS_CREATION_TOKEN"`
	
	ここで、$CLUSTER_NAME はAmazon EKS クラスターの名前、$EFS_CREATION_TOKE	

タスク	説明	必要なスキル
	N はファイルシステムの一意の作成トークン、–d は保存時の暗号化を無効にします。	
セキュリティグループを作成します。	Amazon EKS クラスターが Amazon EFS ファイルシステムにアクセスできるようにするセキュリティグループを作成します。	AWS システム管理者
セキュリティグループのインバウンドルールを更新します。	セキュリティグループのインバウンドルールを更新して、以下の設定で受信トラフィックを許可します。 • TCP プロトコル – ポート 2049 • ソース – Kubernetes クラスターを含む VPC 内のプライベートサブネットの CIDR ブロック範囲	AWS システム管理者
各プライベートサブネットのマウントターゲットを追加します。	Kubernetes クラスターの各プライベートサブネットに、ファイルシステムとセキュリティグループのマウントターゲットを作成します。	AWS システム管理者

Amazon EFS コンポーネントを Kubernetes クラスターにインストールします。

タスク	説明	必要なスキル
Amazon EFS CSI ドライバーをデプロイします。	Amazon EFS CSI ドライバーをクラスターにデプロイします。ドライバーは、アプリ	権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル
	ケーションによって作成された永続ボリュームクレームに従ってストレージをプロビジョニングします。`create-k8s-efs-csi-sc.sh` スクリプトを実行して、Amazon EFS CSI ドライバーとストレージクラスをクラスターにデプロイします。 `./scripts/epic03/create-k8s-efs-csi-sc.sh` このスクリプトは `kubectl` ユーティリティを使用するため、コンテキストが設定され、目的の Amazon EKS クラスターをポイントしていることを確認します。	
ストレージクラスをデプロイします。	Amazon EFS プロビジョナー (efs.csi.aws.com) のクラスターにストレージクラスをデプロイします。	権限が付与された Kubernetes ユーザー

PoC アプリケーションを Kubernetes クラスターにインストールします。

タスク	説明	必要なスキル
永続的ボリュームをデプロイします。	永続ボリュームをデプロイし、作成したストレージクラスと Amazon EFS ファイルシステムの ID にリンクします。アプリケーションは永	権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル
	続ボリュームを使用してコンテンツの読み取りと書き込みをします。ストレージフィールドでは任意のサイズの永続ボリュームを指定できます。Kubernetes ではこのフィールドが必要ですが、Amazon EFS は伸縮性のあるファイルシステムであるため、ファイルシステムの容量は強制しません。永続ボリュームは暗号化の有無にかかわらずデプロイできます。(Amazon EFS CSI ドライバーは、ベストプラクティスとしてデフォルトで暗号化が有効化されています。) deploy-poc-app.sh スクリプトを実行して、永続ボリューム、永続ボリュームクレーム、および 2 つのワークロードをデプロイします。 転送時の暗号化の場合: `./scripts/epic04/deploy-poc-app.sh \ -t "$EFS_CREATION_TOKEN"` ここで、$EFS_CREATION_TOKEN はファイルシステムの一意の作成トークンです。 転送時に暗号化しない場合:	

タスク	説明	必要なスキル
	`./scripts/epic04/deploy-poc-app.sh -d \ -t "$EFS_CREATION_TOKEN"` ここで、$EFS_CREATION_TOKEN はファイルシステムの一意の作成トークンで、-d は転送時の暗号化を無効化します。	
アプリケーションが要求した永続ボリューム要求をデプロイします。	アプリケーションが要求した永続ボリューム要求をデプロイし、ストレージクラスにリンクします。前に作成した永続ボリュームと同じアクセスモードを使用します。ストレージフィールドでは任意のサイズの永続ボリューム要求を指定できます。Kubernetes ではこのフィールドが必要ですが、Amazon EFS は伸縮性のあるファイルシステムであるため、ファイルシステムの容量は強制しません。	権限が付与された Kubernetes ユーザー
ワークロード 1 をデプロイします。	アプリケーションのワークロード 1 を表すポッドをデプロイします。このワークロードは、 ファイルにコンテンツを書き込みます`/data/out1.txt` 。	権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル
ワークロード 2 をデプロイします。	アプリケーションのワークロード 2 を表すポッドをデプロイします。このワークロードは、 ファイルにコンテンツを書き込みます`/data/out2.txt` 。	権限が付与された Kubernetes ユーザー

ファイルシステムの永続性、耐久性、共有性を検証する

タスク	説明	必要なスキル
のステータスを確認しますPersistentVolume 。	次のコマンドを入力して、のステータスを確認しますPersistentVolume 。 `kubectl get pv` 出力例については、「追加情報」セクションを参照してください。	権限が付与された Kubernetes ユーザー
のステータスを確認しますPersistentVolumeClaim 。	次のコマンドを入力して、のステータスを確認しますPersistentVolumeClaim 。 `kubectl -n poc-efs-eks-fargate get pvc` 出力例については、「追加情報」セクションを参照してください。	権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

ワークロード 1 がファイルシステムに書き込みできることを確認します。	次のコマンドを入力して、ワークロード 1 が に書き込んでいることを検証します/data/out1.txt 。
	`kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -f /data/out1.txt`
	結果は次のようになります。
	`...`
	`Thu Sep  3 15:25:07 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:25:12 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:25:17 UTC 2023 - PoC APP 1`
	`...`
		権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

ワークロード 2 がファイルシステムに書き込みできることを確認します。	次のコマンドを入力して、ワークロード 2 が に書き込んでいることを検証します/data/out2.txt 。
	`kubectl -n $APP_NAMESPACE exec -ti poc-app2 -- tail -f /data/out2.txt`
	結果は次のようになります。
	`... Thu Sep 3 15:26:48 UTC 2023 - PoC APP 2 Thu Sep 3 15:26:53 UTC 2023 - PoC APP 2 Thu Sep 3 15:26:58 UTC 2023 - PoC APP 2 ...`
		権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

ワークロード 1 がワークロード 2 によって書き込まれたファイルを読み取れることを検証します。	次のコマンドを入力して、ワークロード 1 がワークロード 2 によって書き込まれた/data/out2.txt ファイルを読み取れることを確認します。
	`kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -n 3 /data/out2.txt`
	結果は次のようになります。
	`...`
	`Thu Sep  3 15:26:48 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:53 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:58 UTC 2023 - PoC APP 2`
	`...`
		権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

ワークロード 2 がワークロード 1 によって書き込まれたファイルを読み取れることを検証します。	次のコマンドを入力して、ワークロード 2 がワークロード 1 によって書き込まれた/data/out1.txt ファイルを読み取れることを確認します。
	`kubectl -n $APP_NAMESPACE exec -ti poc-app2 -- tail -n 3 /data/out1.txt`
	結果は次のようになります。
	`...`
	`Thu Sep  3 15:29:22 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:29:27 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:29:32 UTC 2023 - PoC APP 1`
	`...`
		権限が付与された Kubernetes ユーザー

タスク	説明	必要なスキル

アプリケーションコンポーネントを削除する後もファイルが保持されることを確認します。	次に、スクリプトを使用してアプリケーションコンポーネント (永続ボリューム、永続ボリュームクレーム、ポッド) を削除し、ファイル/data/out1.txt と /data/out2.txt がファイルシステムに保持されていることを確認します。次のコマンドを使用して、validate-efs-content.sh スクリプトを実行します。
	
	`./scripts/epic05/validate-efs-content.sh \`
	`    -t "$EFS_CREATION_TOKEN"`
	
	ここで、$EFS_CREATION_TOKEN はファイルシステムの一意の作成トークンです。
	
	結果は次のようになります。
	
	`pod/poc-app-validation created`
	`Waiting for pod get Running state...`
	`Waiting for pod get Running state...`
	`Waiting for pod get Running state...`
	`Results from execution of 'find /data' on`
		権限が付与された Kubernetes ユーザー、システム管理者

タスク	説明	必要なスキル
	validation process pod: /data /data/out2.txt /data/out1.txt	

オペレーションを監視する

タスク	説明	必要なスキル
アプリケーションログを監視します。	2 日目のオペレーションの一環として、監視用に Amazon CloudWatch にアプリケーションログを送信します。	AWS システム管理者、アクセス許可を付与された Kubernetes ユーザー
Container Insights で、Amazon EKS と Kubernetes コンテナを監視します。	2 日目のオペレーションの一環として、Amazon CloudWatch Container Insights を使用して Amazon EKS システムと Kubernetes システムを監視します。このツールは、コンテナ化されたアプリケーションから、さまざまなレベルとディメンションでメトリクスを収集、集計、要約します。詳細については、[関連リソース] セクションを参照してください。	AWS システム管理者、アクセス許可を付与された Kubernetes ユーザー
CloudWatch で Amazon EFS を監視します。	2 日目のオペレーションの一環として、Amazon CloudWatch を使用してファイルシステムを監視し、Amazon EFS から生データを収集し、リアルタイムに	AWS システム管理者

タスク	説明	必要なスキル
	近い読み取り可能なメトリクスに処理します。詳細については、[関連リソース]セクションを参照してください。	

リソースをクリーンアップする

タスク	説明	必要なスキル
パターン用に作成されたすべてのリソースをクリーンアップします。	このパターンを完了したら、AWS 料金が発生しないようにすべてのリソースをクリーンアップします。PoC アプリケーションの使用が終了したら、`clean-up-resources.sh` スクリプトを実行してすべてのリソースを削除します。次のいずれかのオプションを実行します。 KMS キーを使用して保存時に暗号化する場合: `./scripts/epic06/clean-up-resources.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN" \ -k "$KMS_KEY_ALIAS"` ここで、`$CLUSTER_NAME` は Amazon EKS クラスターの名前、`$EFS_CREATION_TOKEN` はファイルシステムの	権限が付与された Kubernetes ユーザー、システム管理者

タスク	説明	必要なスキル
	作成トークン、$KMS_KEY_ALIAS は KMS キーのエイリアスです。 保存時に暗号化しない場合: `./scripts/epic06/clean-up-resources.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN"` ここで、$CLUSTER_NAME は Amazon EKS クラスターの名前、$EFS_CREATION_TOKEN はファイルシステムの作成トークンです。	

関連リソース

リファレンス

- AWS Fargate for Amazon EKS が Amazon EFS をサポートするようになりました (発表)
- AWS Fargate で Amazon EKS を使用するときにアプリケーションログをキャプチャする方法 (ブログ投稿)
- Container Insights の使用(Amazon CloudWatch ドキュメント)
- Amazon EKS と Kubernetes で Container Insights をセットアップする(Amazon CloudWatch ドキュメント)
- Amazon EKS と Kubernetes Container Insights のメトリクス(Amazon CloudWatch ドキュメント)
- Amazon CloudWatch での Amazon EFS の監視(Amazon EFS ドキュメント)

GitHub チュートリアルおよび例

- 静的プロビジョニング
- 転送時の暗号化
- 複数のポッドからファイルシステムへのアクセス
- StatefulSets での Amazon EFS の利用
- サブパスのマウント
- Amazon EFS アクセスポイントの使用
- Terraform の Amazon EKS 設計図

必要なツール

- AWS CLI バージョン 2 のインストール
- eksctl のインストール
- kubectl のインストール
- jq のインストール

追加情報

以下は、 `kubectl get pv` コマンドの出力例です。

```
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
            STORAGECLASS   REASON   AGE
poc-app-pv   1Mi        RWX            Retain           Bound    poc-efs-eks-fargate/
poc-app-pvc   efs-sc                  3m56s
```

以下は、 `kubectl -n poc-efs-eks-fargate get pvc` コマンドの出力例です。

```
NAME          STATUS   VOLUME       CAPACITY   ACCESS MODES   STORAGECLASS   AGE
poc-app-pvc   Bound    poc-app-pv   1Mi        RWX            efs-sc         4m34s
```

Amazon EKS Pod Identity と KEDA を使用して Amazon EKS でイベント駆動型自動スケーリングを設定する

作成者: Dipen Desai (AWS)、Abhay Diwan (AWS)、Kamal Joshi (AWS)、Mahendra Revanasiddappa (AWS)

概要

Amazon Elastic Kubernetes Service (Amazon EKS) などのオーケストレーションプラットフォームは、コンテナベースのアプリケーションのライフサイクル管理を合理化しました。これにより、組織はコンテナベースのアプリケーションの構築、保護、運用、保守に集中できます。イベント駆動型デプロイがより一般的になると、組織はさまざまなイベントソースに基づいて Kubernetes デプロイをスケーリングする頻度が高くなります。この方法と Auto Scaling を組み合わせると、アプリケーションロジックに合わせたオンデマンドコンピューティングリソースと効率的なスケーリングを提供することで、大幅なコスト削減につながります。

KEDA は Kubernetes ベースのイベント駆動型オートスケーラーです。KEDA は、処理する必要があるイベントの数に基づいて Kubernetes 内のコンテナをスケールするのに役立ちます。軽量で、任意の Kubernetes クラスターと統合できます。また、水平ポッド自動スケーリング (HPA) などの標準 Kubernetes コンポーネントでも動作します。KEDA には、認証を委任するのに役立つ機能である TriggerAuthentication も用意されています。これにより、ScaledObject およびデプロイコンテナとは別の認証パラメータを記述できます。

AWS は、Amazon EKS、Amazon EKS Anywhere AWS Identity and Access Management 、 Red Hat OpenShift Service on AWS (ROSA)、Amazon Elastic Compute Cloud (Amazon EC2) 上のセルフマネージド Kubernetes クラスターなど、さまざまな Kubernetes デプロイオプションをサポートする (IAM) ロールを提供します。 Amazon EKS Anywhere これらのロールは、OpenID Connect (OIDC) ID プロバイダーや IAM 信頼ポリシーなどの IAM コンストラクトを使用して、Amazon EKS サービスや APIs。詳細については、Amazon EKS ドキュメントのサービスアカウントの IAM ロールを参照してください。

Amazon EKS Pod Identity は、Kubernetes サービスアカウントが OIDC プロバイダーを必要とせずに IAM ロールを引き受けるプロセスを簡素化します。アプリケーションの認証情報を管理する機能を提供します。 AWS 認証情報を作成してコンテナに配布したり、Amazon EC2 インスタンスのロールを使用する代わりに、IAM ロールを Kubernetes サービスアカウントに関連付け、サービスアカウントを使用するように Pod を設定します。これにより、複数のクラスターで IAM ロールを使用し、IAM ロール間でアクセス許可ポリシーを再利用できるため、ポリシー管理を簡素化できます。

Amazon EKS Pod Identity で KEDA を実装することで、企業は効率的なイベント駆動型の自動スケーリングとシンプルな認証情報管理を実現できます。アプリケーションは需要に基づいてスケールするため、リソース使用率を最適化し、コストを削減できます。

このパターンは、Amazon EKS Pod Identity を KEDA と統合するのに役立ちます。`keda-operator` サービスアカウントを使用し、 で認証を委任する方法を説明しますTriggerAuthentication。また、KEDA オペレータの IAM ロールとアプリケーションの IAM ロール間の信頼関係を設定する方法についても説明します。この信頼関係により、KEDA はイベントキュー内のメッセージをモニタリングし、送信先 Kubernetes オブジェクトのスケーリングを調整できます。

前提条件と制限

前提条件

- AWS Command Line Interface （ AWS CLI) バージョン 2.13.17 以降、インストール済み
- Python バージョン 3.11.5 以降、インストール済み
- AWS SDK for Python (Boto3) バージョン 1.34.135 以降、インストール済み
- Helm バージョン 3.12.3 以降、インストール済み
- kubectl バージョン 1.25.1 以降、インストール済み
- Docker Engine バージョン 26.1.1 以降、インストール済み
- Amazon EKS クラスターバージョン 1.24 以降が作成されました
- Amazon EKS Pod Identity エージェントを作成するための前提条件が満たされました

制約事項

- `keda-operator` ロールとロールの間に信頼関係を確立する必要がありますkeda-identity。手順については、このパターンの「エピック」セクションを参照してください。

アーキテクチャ

このパターンでは、次の AWS リソースを作成します。

- Amazon Elastic Container Registry (Amazon ECR) リポジトリ – このパターンでは、このリポジトリの名前は ですkeda-pod-identity-registry。このプライベートリポジトリは、サンプルアプリケーションの Docker イメージを保存するために使用されます。
- Amazon Simple Queue Service (Amazon SQS) キュー – このパターンでは、このキューの名前はですevent-messages-queue。キューは、受信メッセージを収集して保存するメッセージバッ

ファとして機能します。KEDA は、メッセージ数やキューの長さなどのキューメトリクスをモニタリングし、これらのメトリクスに基づいてアプリケーションを自動的にスケーリングします。

- アプリケーションの IAM ロール – このパターンでは、このロールの名前は ですkeda-identity。keda-operator ロールはこのロールを引き受けます。このロールは、Amazon SQS キューへのアクセスを許可します。
- KEDA 演算子の IAM ロール – このパターンでは、このロールの名前は ですkeda-operator。KEDA 演算子は、このロールを使用して必要な AWS API コールを行います。このロールには、keda-identityロールを引き受けるアクセス許可があります。keda-operator とロール間の信頼関係のためkeda-identity、keda-operatorロールには Amazon SQS アクセス許可があります。

TriggerAuthentication および ScaledObjectKubernetes カスタムリソースを通じて、オペレーターは keda-identityロールを使用して Amazon SQS キューに接続します。キューのサイズに基づいて、KEDA はアプリケーションのデプロイを自動的にスケーリングします。キュー内の 5 つの未読メッセージごとに 1 つのポッドを追加します。デフォルト設定では、Amazon SQS キューに未読メッセージがない場合、アプリケーションは 0 ポッドにスケールダウンします。KEDA 演算子は、指定した間隔でキューをモニタリングします。

次の図は、Amazon EKS Pod Identity を使用して Amazon SQS キューへの安全なアクセスをkeda-operatorロールに提供する方法を示しています。

この図表は、次のワークフローを示しています：

1. Amazon EKS Pod Identity エージェントを Amazon EKS クラスターにインストールします。
2. Amazon EKS クラスターの KEDA 名前空間に KEDA 演算子をデプロイします。
3. ターゲットに keda-operatorおよび keda-identity IAM ロールを作成します AWS アカウント。
4. IAM ロール間に信頼関係を確立します。
5. アプリケーションを securityを名前空間にデプロイします。
6. KEDA 演算子は、Amazon SQS キュー内のメッセージをポーリングします。
7. KEDA は HPA を開始し、キューのサイズに基づいてアプリケーションを自動的にスケーリングします。

ツール

AWS のサービス

- 「Amazon Elastic Container Registry (Amazon ECR)」は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を認可するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- 「Amazon Simple Queue Service (Amazon SQS)」は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。

その他のツール

- KEDA は Kubernetes ベースのイベント駆動型オートスケーラーです。

コードリポジトリ

このパターンのコードは、EKS Pod Identity と KEDA リポジトリを使用した GitHub イベント駆動型自動スケーリングで使用できます。 https://github.com/aws-samples/event-driven-autoscaling-using-podidentity-and-keda/tree/main

ベストプラクティス

ACM のベストプラクティスに従うことをおすすめします。

- Amazon EKS のベストプラクティス
- IAM でのセキュリティのベストプラクティス
- Amazon SQS のベストプラクティス

エピック

AWS リソースの作成

タスク	説明	必要なスキル
KEDA 演算子の IAM ロールを作成します。	1. にサインインし AWS Management Console、IAM コンソールを開きます。 2. ナビゲーションペインで Roles (ロール) を選択してください。 3. [ロールの作成] を選択してください。 4. [Custom trust policy] (カスタム信頼ポリシー) ロールタイプを選択してください。 5. カスタム信頼ポリシー セクションで、このロールに次のカスタム信頼ポリシーを入力します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "pods.eks.amazonaws.com" }, "Action": [`	AWS 管理者

タスク	説明	必要なスキル

	<pre>"sts:AssumeRole",
	
	"sts:TagSession"
	]
	}
	]
	}</pre>
	6. [アクセス許可を追加] ページで [次へ] を選択してください。このロールにはポリシーを追加しません。
	7. [Role name] (ロール名) にkeda-operator と入力します。
	8. [ロールの作成] を選択してください。	

タスク	説明	必要なスキル
サンプルアプリケーションの IAM ロールを作成します。	1. IAM コンソールのナビゲーションペインで [ロール] を選択します。 2. [ロールの作成] を選択してください。 3. [Custom trust policy] (カスタム信頼ポリシー) ロールタイプを選択してください。 4. カスタム信頼ポリシー セクションで、このロールの次のカスタム信頼ポリシーを入力します。をターゲットアカウント番号<account number>に置き換えます。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "pods.eks.amazonaws.com", "AWS": "arn:aws:iam::<account number>:role/keda-operator" }, "Action": [ "sts:AssumeRole",`	AWS 管理者

タスク	説明	必要なスキル

	<pre>"sts:TagSession"
	]
	}
	]
	}</pre>
	5. アクセス許可の追加ページで、次の AWS マネージドポリシーをロールに追加します。
	• AmazonSQSReadOnlyAccess
	• AWSLambdaSQSQueueExecutionRole
	6. [次へ] を選択します。
	7. [Role name] (ロール名) にkeda-identity と入力します。
	8. [ロールの作成] を選択してください。	

Amazon SQS キューを作成します。	1. Amazon SQS コンソールを開きます。 2. [キューの作成] を選択します。 3. タイプ] で、[標準] を選択します。 4. キューの作成ページで、名前に と入力しますevent-messages-queue 。 5. [キューの作成]を選択します。このキューのデフォルト設定は変更しません。	AWS 全般

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	1. Amazon ECR コンソールを開きます。 2. [Create repository (リポジトリの作成)] を選択します。 3. リポジトリ名には、 と入力しますkeda-pod-identity-registry 。 4. [Create repository (リポジトリの作成)] を選択します。このリポジトリのデフォルト設定は変更しません。	AWS 全般

Amazon EKS クラスターのセットアップ

タスク	説明	必要なスキル
Amazon EKS Pod Identity エージェントをデプロイします。	ターゲット Amazon EKS クラスターで、Amazon EKS Pod Identity エージェントを設定します。Amazon EKS ドキュメントの「Amazon EKS Pod Identity Agent のセットアップ」の手順に従います。	AWS DevOps
KEDA をデプロイします。	1. 次のコマンドを入力して、ターゲット Amazon EKS クラスターに KEDA をデプロイします。 `# Add Helm Repo for Keda`	DevOps エンジニア

タスク	説明	必要なスキル

	<pre>helm repo add kedacore
	 https://kedacore.g
	ithub.io/charts
	# Update Helm repo
	helm repo update
	# Install Keda
	helm install keda
	 kedacore/keda --
	namespace keda --
	create-namespace</pre>
	詳細については、KEDA ドキュメントの「Deploying with Helm」を参照してください。
	2. デプロイが成功したら、出力で、KEDA 演算子用に 3 つのデプロイが作成されていることを確認します。以下は出力例です。
	<pre>NAME
	READY
	UP-TO-DATE
	AVAILABLE AGE
	keda-admission-
	webhooks 1/1
	1 1
	89s
	keda-operator
	1/1 1
	1
	89s
	keda-operator-
	metrics-apiserver
	1/1 1
	1 89s</pre>	

タスク	説明	必要なスキル
IAM ロールを Kubernetes サービスアカウントに割り当てます。	Amazon EKS ドキュメントの「Kubernetes サービスアカウントに IAM ロールを割り当てる」の手順に従います。以下の値を使用します。 • IAM ロールの場合は、 と入力しますkeda-operator 。 • Kubernetes 名前空間の場合は、 と入力しますkeda。 • Kubernetes サービスアカウントの場合は、「」と入力しますkeda-operator 。	AWS DevOps
名前空間を作成します。	次のコマンドを入力して、ターゲット Amazon EKS クラスターにsecurity名前空間を作成します。 `kubectl create ns security`	DevOps エンジニア

サンプルアプリケーションをデプロイする

タスク	説明	必要なスキル
アプリケーションファイルのクローンを作成します。	次のコマンドを入力して、GitHub から EKS Pod Identity と KEDA リポジトリを使用してイベント駆動型自動スケーリングのクローンを作成します。	DevOps エンジニア

タスク	説明	必要なスキル
	`git clone https://github.com/aws-samples/event-driven-autoscaling-using-podidentity-and-keda.git`	

Docker イメージを作成します。	1. 次のコマンドを入力して、クローンされたリポジトリに移動します。
	`cd event-driven-autoscaling-using-podidentity-and-keda`
	2. 次のコマンドを入力して、サンプルアプリケーションの Docker イメージを構築します。
	`docker build -t keda-pod-identity-registry .`
		DevOps エンジニア

タスク	説明	必要なスキル

Amazon ECR にDocker イメージをプッシュします。	1. Docker イメージをビルドしたターミナルで、次のコマンドを入力して Amazon ECR にログインします。<AWS_REGION> と を AWS 環境の値<AWS_ACCOUNT_ID> に置き換えます。
	`aws ecr get-login-password \ --region <AWS_REGION> | docker login \ --username AWS \ --password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com`
	2. 次のコマンドを入力して、イメージにタグを付けます。<AWS_REGION> と を AWS 環境の値<AWS_ACCOUNT_ID> に置き換えます。
	`docker tag keda-pod-identity-registry:latest <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/keda-pod-identity-registry:latest`
	3. 次のコマンドを入力して、イメージを Amazon
		DevOps エンジニア

タスク	説明	必要なスキル
	ECR にプッシュします。<AWS_REGION> と を AWS 環境の値<AWS_ACCOUNT_ID> に置き換えます。 `docker push <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/keda-pod-identity-registry:latest` **Note** プッシュコマンドを見つけるには、Amazon ECR リポジトリページに移動し、プッシュコマンドの表示を選択します。	

タスク	説明	必要なスキル
サンプルアプリケーションをデプロイします。	1. クローンされたリポジトリで、deploy.yaml ファイルを開きます。 2. <AWS_ACCOUNT_ID> と を環境の値<AWS_REGION> に置き換えます。 3. deploy.yaml ファイルを保存して閉じます。 4. 次のコマンドを入力して、ターゲット Amazon EKS クラスターにサンプルアプリケーションをデプロイします。 `kubectl apply -f deploy.yaml` このコマンドは、クラスターにデプロイとサービスアカウントを作成します。	DevOps エンジニア

タスク	説明	必要なスキル

アプリケーションサービスアカウントに IAM ロールを割り当てます。	サンプルアプリケーションのサービスアカウントに IAM `keda-identity` ロールを関連付けるには、次のいずれかを実行します。
	• Amazon EKS ドキュメントの「Kubernetes サービスアカウントに IAM ロールを割り当てる」の手順に従います。以下の値を使用します。
	  • IAM ロールの場合は、 と入力します`keda-identity` 。
	  • Kubernetes 名前空間の場合は、 と入力します`security`。
	  • Kubernetes サービスアカウントの場合は、「」と入力します`my-sqs-read-msgs` 。
	• 次のコマンドを入力します AWS CLI 。をターゲット Amazon EKS クラスターの名前`<cluster-name>` に置き換え、 を`keda-identity` ロールの Amazon リソースネーム (ARN) `<role-ARN>` に置き換えます。
		DevOps エンジニア

タスク	説明	必要なスキル
	`aws eks create-pod-identity-association \ --cluster-name <cluster-name> \ --role-arn <role-ARN> \ --namespace security \ --service-account my-sqs-read-msgs`	

タスク	説明	必要なスキル
ScaledObject と をデプロイしますTriggerAuthentication 。	1. クローンされたリポジトリで、keda.yaml ファイルを開きます。 2. をターゲットの ID {{AWS_ACCOUNT_ID}} に置き換えます AWS アカウント。 3. をターゲット{{AWS_REGION}} に置き換えます AWS リージョン。 4. (オプション) 21 ~ 24 行目で、ScaledObject スケーリングポリシーのパラメータを更新します。これらのパラメータの詳細については、以下を参照してください。 • pollingInterval • cooldownPeriod • idleReplicaCount • minReplicaCount 5. keda.yaml ファイルを保存して閉じます。 6. 次のコマンドを入力して、ScaledObject および TriggerAuthentication リソースをデプロイします。 `kubectl -n security apply -f keda.yaml`	DevOps エンジニア

自動スケーリングをテストする

タスク	説明	必要なスキル

Amazon SQS キューにメッセージを送信します。	1. 次のコマンドを入力して、クローンされたリポジトリに移動します。
	`cd event-driven-autoscaling-using-podidentity-and-keda`
	2. 次のコマンドを入力して、テストメッセージを Amazon SQS キューに送信します。
	`python sqs_send_msg.py`
	sqs_send_msg.py スクリプトは、自動スケーリングをテストするためのメッセージを生成するアプリケーションとして機能します。
	Note
	Python 3 を実行している場合は、「」と入力しますpython3 `sqs_send_msg.py` 。
		DevOps エンジニア

アプリケーションポッドをモニタリングします。	1. 別のターミナルで、次のコマンドを入力してポッドをモニタリングします。	DevOps エンジニア

タスク	説明	必要なスキル

	`kubectl -n security get po`
	2. Amazon SQS キュー内の未読メッセージ 5 件ごとに、KEDA は 1 つのポッドを追加します。前のコマンドの出力で、新しいポッドが追加されていることを確認します。以下は出力例です。
	`kubectl -n security get po`
	`NAME READY STATUS RESTARTS AGE`
	`q-read-797f4c7589-2bj76 1/1 Running 0 2s`
	`q-read-797f4c7589-4zxph 1/1 Running 0 49s`
	`q-read-797f4c7589-cg9dt 1/1 Running 0 18s`
	`q-read-797f4c7589-slc69 1/1 Running 0 33s`
	3. テストが完了したら、元のターミナルで、Ctrl + C (Windows) または CMD + C (macOS) と入力しま	

タスク	説明	必要なスキル
	す。これにより、Python sqs_send_msg.py スクリプトが停止します。	

トラブルシューティング

問題	ソリューション

KEDA 演算子はアプリケーションをスケールできません。	次のコマンドを入力して、IAM keda-operator ロールのログを確認します。
	`kubectl logs -n keda -l app=keda-operator -c keda-operator`
	
	HTTP 403 レスポンスコードがある場合、アプリケーションと KEDA スケーラーには Amazon SQS キューにアクセスするための十分なアクセス許可がありません。以下のステップを実行します。
	1. keda-identity ロールの IAM ポリシーとステートメントをチェックして、キューの読み取りアクセスが付与されていることを確認します。
	2. IAM ロール間の信頼関係を検証します。以下に例を示します。
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {`

問題	ソリューション

```
                "Service": "pods.eks
.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

Assume-Role エラーが発生した場合、Amazon EKS ノードの IAM ロールは、 に定義されている IAM ロールを引き受けることができませんTriggerAuthentication 。以下のステップを実行します。

1. 次のコマンドを入力してポッドを削除し、新しいkeda-operator ポッドを作成します。

```
kubectl delete pod keda-operator-
<alphenumeric-value> --namespace
 keda
```

2. 次のコマンドを入力して、ポッドが引き受ける ID を確認します。

```
kubectl describe pod <keda-operator-
pod-name> --namespace keda
```

3. ポッドが正常に再起動したら、次の 2 つの変数がポッドの説明に追加されていることを確認します。
   - AWS_CONTAINER_CREDENTIALS_FULL_URI

問題	ソリューション
	• AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE

関連リソース

- Amazon EKS Pod Identity Agent のセットアップ (Amazon EKS ドキュメント)
- KEDA のデプロイ (KEDA ドキュメント)
- ScaledObject 仕様 (KEDA ドキュメント)
- TriggerAuthentication による認証 (KEDA ドキュメント)

PGO を使用して Amazon EKS への PostgreSQL デプロイを合理化する

作成者: Shalaka Dengale (AWS)

概要

このパターンは、Crunchy Data (PGO) の Postgres Operator を Amazon Elastic Kubernetes Service (Amazon EKS) と統合して、クラウドネイティブ環境での PostgreSQL デプロイを合理化します。PGO は、Kubernetes で PostgreSQL データベースを管理するための自動化とスケーラビリティを提供します。PGO を Amazon EKS と組み合わせると、PostgreSQL データベースを効率的にデプロイ、管理、スケーリングするための堅牢なプラットフォームを形成します。

この統合には、次の主な利点があります。

- 自動デプロイ: PostgreSQL クラスターのデプロイと管理を簡素化します。
- カスタムリソース定義 (CRDs): PostgreSQL 管理に Kubernetes プリミティブ を使用します。
- 高可用性: 自動フェイルオーバーと同期レプリケーションをサポートします。
- 自動バックアップと復元: バックアップと復元プロセスを 合理化します。
- 水平スケーリング: PostgreSQL クラスターの動的スケーリング を有効にします。
- バージョンアップグレード: 最小限のダウンタイムでローリングアップグレードを容易にします。
- セキュリティ: 暗号化、アクセスコントロール、認証メカニズムを適用します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Linux、macOS または Windows にインストールして設定されている「AWS Command Line Interface (AWS CLI) バージョン 2」。
- コマンドラインから AWS リソースを接続する AWS CLI Config。
- Linux、macOS、または Windows にインストールおよび設定されている eksctl。
- `kubectl`、Amazon EKS クラスターのリソースにアクセスするようにインストールおよび設定されています。詳細については、Amazon EKS ドキュメントの「kubectl と eksctl のセットアップ」を参照してください。

- Amazon EKS クラスターにアクセスするように設定されたコンピュータターミナル。詳細については、Amazon EKS ドキュメントの「クラスターと通信するようにコンピュータを設定する」を参照してください。

製品バージョン

- Kubernetes バージョン 1.21 ~ 1.24 以降 (PGO ドキュメントを参照) 。
- PostgreSQL バージョン 10 以降。このパターンでは、PostgreSQL バージョン 16 を使用します。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EKS
- Amazon Virtual Private Cloud (Amazon VPC)
- Amazon Elastic Compute Cloud (Amazon EC2)

ターゲット アーキテクチャ

このパターンは、3 つのノードを持つ Amazon EKS クラスターを含むアーキテクチャを構築します。各ノードは、バックエンドの一連の EC2 インスタンスで実行されます。この PostgreSQL のセットアップは、読み取りが多いユースケースに特に効果的なプライマリレプリカアーキテクチャに従います。アーキテクチャには、以下のコンポーネントが含まれます。

- プライマリデータベースコンテナ (pg-primary) は、すべての書き込みオペレーションが指示されるメイン PostgreSQL インスタンスをホストします。
- セカンダリレプリカコンテナ (pg-replica) は、プライマリデータベースからデータをレプリケートし、読み取りオペレーションを処理する PostgreSQL インスタンスをホストします。

- PgBouncer は、PGO に含まれている PostgreSQL データベース用の軽量接続プーラーです。これはクライアントと PostgreSQL サーバーの間にあり、データベース接続の仲介として機能します。
- PGO は、この Kubernetes 環境における PostgreSQL クラスターのデプロイと管理を自動化します。
- Patroni は、PostgreSQL の高可用性設定を管理および自動化するオープンソースツールです。PGO に含まれています。Kubernetes で PGO で Patroni を使用すると、PostgreSQL クラスターの耐障害性と耐障害性を確保する上で重要な役割を果たします。詳細については、「Patroni ドキュメント」を参照してください。

ワークフローには、以下のステップが含まれます。

- PGO 演算子をデプロイします。Amazon EKS で実行される Kubernetes クラスターに PGO 演算子をデプロイします。これは、Kubernetes マニフェストまたは Helm チャートを使用して行うことができます。このパターンでは、Kubernetes マニフェストを使用します。
- PostgreSQL インスタンスを定義します。演算子の実行中に、カスタムリソース (CRs) を作成して、PostgreSQL インスタンスの望ましい状態を指定します。これには、ストレージ、レプリケーション、高可用性設定などの設定が含まれます。
- オペレーター管理。CRs、PostgreSQL インスタンスを作成、更新、または削除します。
- モニタリングとメンテナンス。Amazon EKS で実行されている PostgreSQL インスタンスのヘルスとパフォーマンスをモニタリングできます。オペレーターは、多くの場合、モニタリング目的でメトリクスとログ記録を提供します。必要に応じて、アップグレードやパッチ適用などの定期的なメンテナンスタスクを実行できます。詳細については、「Amazon EKS ドキュメント」の「クラスターのパフォーマンスをモニタリングし、ログを表示する」を参照してください。
- スケーリングとバックアップ: オペレータが提供する機能を使用して、PostgreSQL インスタンスをスケーリングし、バックアップを管理できます。

このパターンは、モニタリング、メンテナンス、バックアップオペレーションを対象としていません。

自動化とスケール

- を使用して AWS CloudFormation インフラストラクチャの作成を自動化できます。詳細については、Amazon EKS ドキュメントの「 を使用して Amazon EKS リソース AWS CloudFormationを作成する」を参照してください。
- GitVersion または Jenkins ビルド番号を使用して、データベースインスタンスのデプロイを自動化できます。

ツール

AWS のサービス

- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

その他のツール

- eksctl は、Amazon EKS でクラスターを作成するためのシンプルなコマンドラインツールです。
- 「kubectl」は、 Kubernetes クラスターに対してコマンドを実行するためのコマンドラインユーティリティです。
- PGO は、Kubernetes での PostgreSQL データベースの管理を自動化およびスケーリングします。

ベストプラクティス

スムーズで効率的なデプロイを確実に行うには、以下のベストプラクティスに従ってください。

- EKS クラスターを保護します。サービスアカウント (IRSA)、ネットワークポリシー、VPC セキュリティグループの AWS Identity and Access Management (IAM) ロールを使用するなど、EKS クラスターのセキュリティのベストプラクティスを実装します。EKS クラスター API サーバーへのアクセスを制限し、TLS を使用してノードと API サーバー間の通信を暗号化します。
- Amazon EKS で実行されている PGO と Kubernetes のバージョン互換性を確保します。一部の PGO 機能では、特定の Kubernetes バージョンが必要な場合や、互換性の制限が生じる場合があります。詳細については、PGO ドキュメントの「コンポーネントと互換性」を参照してください。
- CPU、メモリ、ストレージなど、PGO デプロイのリソース割り当てを計画します。PGO と管理する PostgreSQL インスタンスの両方のリソース要件を検討します。リソースの使用状況を監視し、必要に応じてリソースをスケーリングします。
- 高可用性を実現する設計。ダウンタイムを最小限に抑え、信頼性を確保するために、高可用性を実現する PGO デプロイを設計します。耐障害性のために、複数のアベイラビリティーゾーンに複数の PGO レプリカをデプロイします。

- PGO が管理する PostgreSQL データベースのバックアップおよび復元手順を実装します。Kubernetes および Amazon EKS と互換性のある PGO またはサードパーティーのバックアップソリューションが提供する機能を使用します。
- PGO デプロイのモニタリングとログ記録を設定して、パフォーマンス、ヘルス、イベントを追跡します。Prometheus などのツールを使用してメトリクスをモニタリングし、Grafana を使用して視覚化します。トラブルシューティングと監査のために PGO ログをキャプチャするようにログ記録を設定します。
- Kubernetes クラスター内の PGO、PostgreSQL インスタンス、およびその他の サービス間の通信を許可するようにネットワークを適切に設定します。ネットワークポリシーの適用とトラフィックの分離には、Amazon VPC ネットワーク機能、および Calico や Amazon VPC CNI などの Kubernetes ネットワークプラグインを使用します。
- パフォーマンス、耐久性、スケーラビリティなどの要素を考慮して、PostgreSQL データベースに適したストレージオプションを選択します。永続的ストレージには、Amazon Elastic Block Store (Amazon EBS) ボリュームまたは AWS マネージドストレージサービスを使用します。詳細については、Amazon EKS ドキュメントの「Amazon EBS で Kubernetes ボリュームを保存する」を参照してください。
- などの Infrastructure as Code (IaC) ツールを使用して AWS CloudFormation 、Amazon EKS での PGO のデプロイと設定を自動化します。EKS クラスター、ネットワーク、PGO リソースなどのインフラストラクチャコンポーネントを、一貫性、再現性、バージョン管理のためのコードとして定義します。

エピック

IAM ロールを作成する

タスク	説明	必要なスキル
IAM ロールを作成します。	1. で次のコマンドを使用して IAM ロールを作成します AWS CLI。 `aws iam create-role \ --role-name {YourRoleName} \ --assume-role-policy-document '{`	AWS 管理者

タスク	説明	必要なスキル
	`"Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Service": "eks.amazonaws.com" }, "Action": "sts:AssumeRole" }] }' && \ aws iam attach-role-policy \ --role-name {YourRoleName}\ --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy && \ aws iam attach-role-policy \ --role-name {YourRoleName}\ --policy-arn arn:aws:iam::aws:policy/AmazonEKSServicePolicy && \ aws iam attach-role-policy \ --role-name {YourRoleName}\ --policy-arn arn:aws:iam::aws:p`	

タスク	説明	必要なスキル

	`olicy/CloudWatchFu llAccess`
	2. でロールを確認します AWS Management Console。
	a. [IAM コンソール] を開きます。
	b. ロール を選択し、作成したロール名を検索します。
	c. 次のポリシーがアタッチされていることを確認します。
	`AmazonEKS ClusterPolicy`
	`AmazonEKS ServicePolicy`
	`CloudWatc hFullAccess`	

Amazon EKS クラスターを作成します。

タスク	説明	必要なスキル
Amazon EKS クラスターを作成します。	クラスターを既にデプロイしている場合は、このステップをスキップします。それ以外の場合は、`eksctl`、Terraform、または AWS アカウントを使用して、Amazon EKS クラスターを現在の にデプ	AWS 管理者、Terraform または eksctl 管理者、Kubernetes 管理者

タスク	説明	必要なスキル

	ロイします AWS CloudFormation。このパターンでは、クラスターのデプロイeksctlにを使用します。
	
	Note
	このパターンでは、Amazon EKS のノードグループとして Amazon EC2 を使用します。を使用する場合は AWS Fargate、eksctl ドキュメントmanagedNodeGroups の設定を参照してください。
	
	1. 次のeksctl入力ファイルを使用してクラスターを生成します。
	
	sample-cluster.yaml：
	
	`apiVersion: eksctl.io/v1alpha5`
	`kind: ClusterConfig`
	`metadata:`
	`  name: postgresql`
	`  region: us-east-1`
	`  version: "1.29"`
	`accessConfig:`	

タスク	説明	必要なスキル

	<pre> authenticationMode
	: API_AND_C
	ONFIG_MAP
	availabilityZones:
	- us-east-1a
	- us-east-1b
	- us-east-1c
	nodeGroups:
	  - name: ng-1
	    instanceType:
	 m5.16xlarge
	    desiredCapacity:
	 2
	  - name: ng-2
	    instanceType:
	 m5.16xlarge
	    desiredCapacity:
	 2
	  - name: ng-3
	    instanceType:
	 m5.16xlarge
	    desiredCapacity:
	 2
	vpc:
	  cidr: 192.168.0
	.0/16
	  clusterEndpoints:
	    publicAccess:
	 true
	  nat:
	    gateway: HighlyAva
	ilable
	iamIdentity
	Mappings:
	  - arn: arn:aws:i
	am::<account-id>:r
	ole/<role-name> #
	 update the IAM role
	 ARN created in step 1
	    username: <user-
	name> # Enter the</pre>	

タスク	説明	必要なスキル
	user name per your choice noDuplicateARNs: false 2. 次のコマンドを実行してクラスターを作成します (ファイルへのsample-cluster.yaml ファイルパスを指定します)。 eksctl create cluster -f sample-cluster.yaml	
クラスターのステータスを検証します。	次のコマンドを実行して、クラスター内のノードの現在のステータスを確認します。 kubectl get nodes エラーが発生した場合は、Amazon EKS ドキュメントのトラブルシューティングセクションを参照してください。	AWS 管理者、Terraform または eksctl 管理者、Kubernetes 管理者

OIDC ID プロバイダーを作成する

タスク	説明	必要なスキル
IAM OIDC プロバイダーを有効にします。	Amazon EBS Container Storage Interface (CSI) ドライバーの前提条件として、クラスターに既存の IAM OpenID	AWS 管理者

タスク	説明	必要なスキル
	Connect (OIDC) プロバイダーが必要です。 次のコマンドを使用して、IAM OIDC プロバイダーを有効にします。 `eksctl utils associate-iam-oidc-provider --region={region} --cluster={YourClusterNameHere} --approve` このステップの詳細については、Amazon EKS ドキュメントを参照してください。	

タスク	説明	必要なスキル

Amazon EBS CSI ドライバーの IAM ロールを作成します。	次のeksctlコマンドを使用して、CSI ドライバーの IAM ロールを作成します。
	
	`eksctl create iamserviceaccount \`
	`  --region {RegionName} \`
	`  --name ebs-csi-controller-sa \`
	`  --namespace kube-system \`
	`  --cluster {YourClusterNameHere} \`
	`  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \`
	`  --approve \`
	`  --role-only \`
	`  --role-name AmazonEKS_EBS_CSI_DriverRole`
	
	暗号化された Amazon EBS ドライブを使用する場合は、ポリシーをさらに設定する必要があります。手順については、Amazon EBS SCI ドライバーのドキュメントを参照してください。
		AWS 管理者

タスク	説明	必要なスキル
Amazon EBS CSI ドライバーを追加します。	次のeksctlコマンドを使用して、Amazon EBS CSI ドライバーを追加します。 `eksctl create addon \ --name aws-ebs-csi-driver \ --cluster <YourClusterName> service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity \ --query Account \ --output text):role/AmazonEKS_EBS_CSI_DriverRole \ --force`	AWS 管理者

PGO のインストール

タスク	説明	必要なスキル
PGO リポジトリのクローンを作成します。	PGO の GitHub リポジトリのクローンを作成します。 `git clone https://github.com/CrunchyData/postgres-operator-examples.git`	AWS DevOps
サービスアカウント作成のロールの詳細を指定します。	Amazon EKS クラスターに必要な AWS リソースへのアクセスを許可するには、`service_account.yaml` ファイルで前に作成した OIDC ロールの Amazon リ	AWS 管理者、Kubernetes 管理者

タスク	説明	必要なスキル

	ソースネーム (ARN) を指定します。このファイルは、リポジトリの名前空間フォルダにあります。
	`cd postgres-operator-examples`
	`--- metadata: annotations: eks.amazonaws.com/ role-arn: arn:aws:i am::<accountId>:role/ <role_name> # Update the OIDC role ARN created earlier`	

タスク	説明	必要なスキル

名前空間と PGO の前提条件を作成します。	1. 名前空間を作成するには、次のコマンドを実行します。
	
	`kubectl apply -k kustomize/install/namespace`
	
	これにより、PGO 専用の名前空間が確立されます。必要に応じて、 `namespace.yml` ファイルを変更し、名前空間に別の名前を割り当てることができます。
	2. 次のコマンドを実行して、デフォルト設定をクラスターに適用します。
	
	`kubectl apply --server-side -k kustomize/install/default`
	
	`kustomize/install/default` は、Kubernetes ロールベースのアクセスコントロール (RBAC)、カスタムリソース定義 (CRD)、および Kubernetes Manager ファイルのデフォルト設定を提供します。
		Kunernetes 管理者

タスク	説明	必要なスキル
ポッドの作成を確認します。	名前空間とデフォルト設定が作成されていることを確認します。 `kubectl get pods -n postgres-operator`	AWS 管理者、Kubernetes 管理者
PVCs を確認します。	次のコマンドを使用して、永続ボリュームクレーム (PVCs を確認します。 `kubectl describe pvc -n postgres-operator`	AWS 管理者、Kubernetes 管理者

演算子を作成してデプロイする

タスク	説明	必要なスキル

演算子を作成します。	にある ファイルの内容を修正/kustomize/postgres/postgres.yaml して、以下と一致させます。
	`spec:`
	`  instances:`
	`  - name: pg-1`
	`  replicas: 3`
	`  patroni:`
	`  dynamicConfiguration:`
	`  postgresql:`
	`  pg_hba:`
	`  - "host all all 0.0.0.0/0 trust"`
	`  # this line enabled logical replication`
		AWS 管理者、DBA、Kubernetes 管理者

タスク	説明	必要なスキル

	<pre>with programmatic access
	- "host all postgres 127.0.0.1/32 md5"
	synchronous_mode: true
	users:
	- name: replicator
	databases:
	- testdb
	options: "REPLICATION"</pre>
	これらの更新では、次のことが行われます。
	• PostgreSQL インスタンスへのアクセスを容易にするために、PostgreSQL の設定を調整します。
	• レプリケーションユーザー、データベースユーザー、スーパーユーザーの設定を含めて、ストリーミングレプリケーション、データベースアクセス、クラスター管理を有効にします。	

タスク	説明	必要なスキル
演算子をデプロイします。	PGO 演算子をデプロイして、Kubernetes 環境で PostgreSQL データベースの効率的な管理とオペレーションを有効にします。 `kubectl apply -k kustomize/postgres`	AWS 管理者、DBA、Kubernetes 管理者

デプロイメントを確認する	1. 演算子がデプロイされたことを確認します。
	`kubectl get pods -n postgres-operator --selector=postgres-operator.crunchydata.com/instance-set \ -L postgres-operator.crunchydata.com/role`
	2. オペレータポッドに関連付けられたサービスリソースが作成されていることを確認します。
	`kubectl get svc -n postgres-operator`
	コマンド出力から、プライマリレプリカ (`primary_pod_name`) とリードレプリカ () を書き留めます`read_pod_name`。これらは次のステップで使用します。
		AWS 管理者、DBA、Kubernetes 管理者

ストリーミングレプリケーションを検証する

タスク	説明	必要なスキル

プライマリレプリカにデータを書き込みます。	次のコマンドを使用して PostgreSQL プライマリレプリカに接続し、データベースにデータを書き込みます。
	`kubectl exec -it <primary_pod_name> bash -n postgres-operator`
	`psql`
	`CREATE TABLE customers (firstname text, customer_id serial, date_created timestamp );`
	`\dt`
		AWS 管理者、Kubernetes 管理者

リードレプリカに同じデータがあることを確認します。	PostgreSQL リードレプリカに接続し、ストリーミングレプリケーションが正しく機能しているかどうかを確認します。
	`kubectl exec -it {read_pod_name} bash -n postgres-operator`
	`psql`
	`\dt`
		AWS 管理者、Kubernetes 管理者

タスク	説明	必要なスキル
	リードレプリカには、前のステップでプライマリレプリカで作成したテーブルが必要です。	

トラブルシューティング

問題	ソリューション

ポッドは起動しません。	• 次のコマンドを使用して、ポッドのステータスを検査します。
	`kubectl get pods -n your-namespace`
	• ログにエラーがないか検査します。
	`kubectl logs your-pod-name -n your-namespace`
	• ポッドに関連する異常なイベントがないかポッドイベントを確認します。
	`kubectl describe pod your-pod-name -n your-namespace`

レプリカはプライマリデータベースのかなり後にあります。	• レプリケーションの遅延を確認します。
	`SELECT * FROM pg_stat_replication;`
	• レプリカに十分な CPU リソースとメモリリソースがあることを確認します。リソースの制限を確認します。
	`kubectl describe pod your-replica-pod -n your-namespace`

問題	ソリューション
	• ストレージバックエンドが最適に動作していることを確認します。ディスク I/O が遅いと、レプリケーションの遅延が発生する可能性があります。
PostgreSQL クラスターのパフォーマンスと状態を可視化することはできません。	• Amazon CloudWatch Logs を有効にし、分析のために口グが Amazon CloudWatch に送信されていることを確認します。詳細については、Amazon EKS のドキュメント参照してください。 • チェック `pg_stat_activity` ： `SELECT * FROM pg_stat_activity;`

問題	ソリューション

レプリケーションは機能しません。	• でレプリケーション設定を表示して、プライマリ設定を確認しますpostgresql.conf 。
	`wal_level = replica`
	`max_wal_senders = 10`
	`wal_keep_size = 64  # or wal_keep_segments in older versions`
	• にレプリケーション許可pg_hba.conf が含まれていることを確認します。
	`host replication replica_user all md5`
	• レプリカの設定を確認します。recovery.conf または同等の設定 (standby.signal と primary_conninfo) がレプリカで正しく設定されていることを確認します。

関連リソース

- Amazon Elastic Kubernetes Service (AWS でのデプロイオプションの概要ホワイトペーパー)
- AWS CloudFormation (AWS でのデプロイオプションの概要ホワイトペーパー)
- Amazon EKS の使用を開始する – eksctl (Amazon EKS ユーザーガイド)
- kubectl と eksctl のセットアップ (Amazon EKS ユーザーガイド)
- OpenID Connect フェデレーション用のロールを作成する (IAM ユーザーガイド)
- の設定 (AWS CLIAWS CLI ユーザーガイド)
- Crunchy Postgres for Kubernetes のドキュメント
- Crunch & Learn: Crunchy Postgres for Kubernetes 5.0 (ビデオ)

Application Load Balancer を使用して Amazon ECS の相互 TLS でアプリケーション認証を簡素化する

作成者: Olawale Olaleye (AWS) と Shamanth Devagari (AWS)

概要

このパターンは、Application Application Load Balancer (ALB) を使用して、Amazon Elastic Container Service (Amazon ECS) の相互 TLS でアプリケーション認証を簡素化し、セキュリティの負担を軽減するのに役立ちます。ALB では、X.509 クライアント証明書を認証できます AWS Private Certificate Authority。この強力な組み合わせにより、 サービス間の安全な通信を実現し、アプリケーション内の複雑な認証メカニズムの必要性を軽減できます。さらに、このパターンは Amazon Elastic Container Registry (Amazon ECR) を使用してコンテナイメージを保存します。

このパターンの例では、パブリックギャラリーの Docker イメージを使用して、最初にサンプルワークロードを作成します。その後、新しい Docker イメージは Amazon ECR に保存されるように構築されます。ソースとして、GitHubGitLab ベースのシステムを検討するか、Amazon Simple Storage Service Amazon S3 (Amazon S3) を使用します。Docker イメージを構築するには、後続のイメージ AWS CodeBuild に を使用することを検討してください。

前提条件と制限

前提条件

- AWS CloudFormation スタックをデプロイするためのアクセス権 AWS アカウント を持つアクティブな 。CloudFormation をデプロイするための AWS Identity and Access Management (IAM) ユーザーまたはロールのアクセス許可があることを確認します。
- AWS Command Line Interface （ AWS CLI) がインストールされています。ローカルマシンまたは環境で AWS 認証情報を設定するには、 を使用するか、 `~/.aws/credentials` ファイルで環境変数 AWS CLI を設定します。
- OpenSSL がインストールされました。
- Docker がインストールされました。
- 「 ツール」で AWS のサービス 説明されている に精通していること。
- Docker と NGINX に関する知識。

制約事項

- Application Load Balancer の相互 TLS は、X.509v3 クライアント証明書のみをサポートします。X.509v1 クライアント証明書はサポートされていません。
- このパターンのコードリポジトリで提供されている CloudFormation テンプレートには、スタックの一部として CodeBuild プロジェクトをプロビジョニングすることは含まれません。
- 一部の AWS のサービス は では使用できません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Docker バージョン 27.3.1 以降
- AWS CLI バージョン 2.14.5 以降

アーキテクチャ

次の図は、このパターンのアーキテクチャコンポーネントを示しています。

この図表は、次のワークフローを示しています：

1. Git リポジトリを作成し、アプリケーションコードをリポジトリにコミットします。
2. でプライベート認証機関 (CA) を作成します AWS Private CA。
3. CodeBuild プロジェクトを作成する。CodeBuildprojectは、コミットの変更によってトリガーされ、Docker イメージを作成し、ビルドされたイメージを Amazon ECR に発行します。
4. CA から証明書チェーンと証明書本文をコピーし、証明書バンドルを Amazon S3 にアップロードします。
5. Amazon S3 にアップロードした CA バンドルを使用してトラストストアを作成します。Application Load Balancer (ALB) の相互 TLS リスナーにトラストストアを関連付けます。
6. プライベート CA を使用して、コンテナワークロードのクライアント証明書を発行します。また、 を使用してプライベート TLS 証明書を作成します AWS Private CA。
7. プライベート TLS 証明書を AWS Certificate Manager (ACM) にインポートし、ALB で使用します。
8. のコンテナワークロードは、発行されたクライアント証明書`ServiceTwo`を使用して、 のコンテナワークロードと通信するときに ALB と認証します`ServiceOne`。

9. のコンテナワークロードは、発行されたクライアント証明書`ServiceOne`を使用して、 のコンテナワークロードと通信するときに ALB と認証します`ServiceTwo`。

自動化とスケール

このパターンは、CloudFormation、 AWS Cloud Development Kit (AWS CDK) または SDK の API オペレーションを使用して AWS リソースをプロビジョニングすることで完全に自動化できます。

AWS CodePipeline を使用して、CodeBuild を使用して継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを実装し、コンテナイメージのビルドプロセスを自動化し、Amazon ECS クラスターサービスに新しいリリースをデプロイできます。

ツール

AWS のサービス

- AWS Certificate Manager (ACM) は、 AWS ウェブサイトとアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書とキーの作成、保存、更新に役立ちます。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- Amazon Elastic Container Service (Amazon ECS) は、クラスター上のコンテナを実行、停止、管理するための非常にスケーラブルで高速なコンテナ管理サービスです。タスクとサービスは、が管理するサーバーレスインフラストラクチャで実行できます AWS Fargate。または、インフラストラクチャをより詳細に制御するために、管理する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターでタスクとサービスを実行できます。
- Amazon ECS Exec を使用すると、最初にホストコンテナオペレーティングシステムとやり取りしたり、インバウンドポートを開いたり、SSH キーを管理したりすることなく、コンテナと直接やり取りできます。ECS Exec を使用して、Amazon EC2 インスタンスまたは で実行されているコンテナでコマンドを実行したり、シェルを取得したりできます AWS Fargate。
- Elastic Load Balancing (ELB) は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。たとえば、1 つ以上のアベイラビリティーゾーンの Amazon EC2 インスタンス、コンテナ、IP アドレスにトラフィックを分散できます。ELB は、登録された

ターゲットの状態を監視し、正常なターゲットにのみトラフィックをルーティングします。ELB は、受信トラフィックが時間の経過とともに変化するにつれてロードバランサーをスケーリングします。ほとんどのワークロードに自動的にスケーリングできます。

- AWS Fargate を使用すると、サーバーや Amazon EC2 インスタンスを管理することなくコンテナを実行できます。Fargate は、Amazon ECS と Amazon Elastic Kubernetes Service (Amazon EKS) の両方と互換性があります。Amazon ECS タスクとサービスは、Fargate 起動タイプまたは Fargate キャパシティープロバイダーで実行できます。そのためには、アプリケーションをコンテナにパッケージ化し、CPU とメモリの要件を指定し、ネットワークと IAM ポリシーを定義して、アプリケーションを起動します。各Fargate タスクは、独自の分離境界を持ち、基盤となるカーネル、CPU リソース、メモリリソース、Elastic Network Interface を別のタスクと共有しません。
- AWS Private Certificate Authority では、オンプレミス CA の運用にかかる投資コストや保守コストなしに、ルート CA や下位 CA を含むプライベート認証機関 (CA) 階層を作成できます。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- GitHub、GitLab、Bitbucket は、ソースコードの変更を追跡するために一般的に使用される Git ベースのソース管理システムの一部です。
- NGINX オープンソースは、オープンソースのロードバランサー、コンテンツキャッシュ、ウェブサーバーです。このパターンでは、ウェブサーバーとして使用します。
- OpenSSL は、TLS と CMS の OpenSSL 実装で使用されるサービスを提供するオープンソースライブラリです。

コードリポジトリ

このパターンのコードは、GitHub mTLS-with-Application-Load-Balancer-in-Amazon-ECS リポジトリで入手できます。

ベストプラクティス

- Amazon ECS Exec を使用してコマンドを実行するか、Fargate で実行されているコンテナにシェルを取得します。ECS Exec を使用して、デバッグの診断情報を収集することもできます。
- セキュリティグループとネットワークアクセスコントロールリスト (ACLs) を使用して、サービス間のインバウンドトラフィックとアウトバウンドトラフィックを制御します。Fargate タスクは、Virtual Private Cloud (VPC) の設定済みサブネットから IP アドレスを受け取ります。

エピック

リポジトリを作成する

タスク	説明	必要なスキル
ソースコードをダウンロードします。	このパターンのソースコードをダウンロードするには、GitHub mTLS-with-Application-Load-Balancer-in-Amazon-ECS リポジトリをフォークまたはクローンします。	DevOps エンジニア
Git リポジトリを作成します。	Dockerfile と `buildspec.yaml` ファイルを含む Git リポジトリを作成するには、次の手順を実行します。 1. 仮想環境にフォルダを作成します。プロジェクト名で名前を付けます。 2. ローカルマシンでターミナルを開き、このフォルダに移動します。 3. mTLS-with-Application-Load-Balancer-in-Amazon-ECS リポジトリをプロジェクトディレクトリにクローンするには、次のコマンドを入力します。 `git clone https://github.com/aws-samples/mTLS-with-Application-Load-Balan`	DevOps エンジニア

タスク	説明	必要なスキル
	`cer-in-Amazon-ECS.git`	

CA を作成して証明書を生成する

タスク	説明	必要なスキル
でプライベート CA を作成します AWS Private CA。	プライベート認証機関 (CA) を作成するには、ターミナルで次のコマンドを実行します。サンプル変数の値を独自の値に置き換えます。 `export AWS_DEFAULT_REGION="us-west-2"` `export SERVICES_DOMAIN="www.example.com"` `export ROOT_CA_ARN=`aws acm-pca create-certificate-authority \` `--certificate-authority-type ROOT \` `--certificate-authority-configuration \` `"KeyAlgorithm=RSA_2048,` `SigningAlgorithm=SHA256WITHRSA,` `Subject={` `Country=US,` `State=WA,` `Locality=Seattle,` `Organization=Build on AWS,`	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル

	<pre>Organizat
	ionalUnit=mTLS Amazon
	ECS and ALB Example,
	CommonName=
	${SERVICES_DOMAIN}}" \
	--query
	CertificateAuthori
	tyArn --output text`</pre>
	詳細については、 AWS ドキュメントの「 でプライベート CA を作成する AWS Private CA」を参照してください。	

タスク	説明	必要なスキル
プライベート CA 証明書を作成してインストールします。	プライベートルート CA の証明書を作成してインストールするには、ターミナルで次のコマンドを実行します。 1. 証明書署名リクエスト (CSR) を生成します。 ``` ROOT_CA_CSR=`aws acm-pca get-certificate-authority-csr \     --certificate-authority-arn ${ROOT_CA_ARN} \     --query Csr --output text` ``` 2. ルート証明書を発行します。 ``` AWS_CLI_VERSION=$(aws --version 2>&1 | cut -d/ -f2 | cut -d. -f1) [[${AWS_CLI_VERSION} -gt 1]] && ROOT_CA_CSR="$(echo ${ROOT_CA_CSR} | base64)" ROOT_CA_CERT_ARN=`aws acm-pca issue-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --template-arn arn:aws:acm-pca:::template/RootCACertificate/V1 \ ```	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`--signing-algorithm SHA256WITHRSA \` `--validity Value=10,Type=YEARS \` `--csr "${ROOT_CA_CSR}" \` `--query CertificateArn --output text` 3. ルート証明書を取得します。 `ROOT_CA_CERT=`aws acm-pca get-certificate \` `--certificate-arn ${ROOT_CA_CERT_ARN} \` `--certificate-authority-arn ${ROOT_CA_ARN} \` `--query Certificate --output text`  `# store for later use` `aws acm-pca get-certificate \` `--certificate-arn ${ROOT_CA_CERT_ARN} \` `--certificate-authority-arn ${ROOT_CA_ARN} \` `--query Certificate --output text > ca-cert.pem`	

タスク	説明	必要なスキル

	4. ルート CA 証明書をインポートして CA にインストールします。
	
	`[[ ${AWS_CLI_VERSION} -gt 1 ]] && ROOT_CA_CERT="$(echo ${ROOT_CA_CERT} | base64)"`
	
	`aws acm-pca import-certificate-authority-certificate \ --certificate-authority-arn $ROOT_CA_ARN \ --certificate "${ROOT_CA_CERT}"`
	
	詳細については、 AWS ドキュメントの「CA 証明書のインストール」を参照してください。	

タスク	説明	必要なスキル
マネージド証明書をリクエストします。	でプライベート ALB AWS Certificate Manager で使用するプライベート証明書をリクエストするには、次のコマンドを使用します。 `export TLS_CERTIFICATE_ARN=`aws acm request-certificate \ --domain-name "*.${DOMAIN_DOMAIN}" \ --certificate-authority-arn ${ROOT_CA_ARN} \ --query CertificateArn --output text`	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル

プライベート CA を使用してクライアント証明書を発行します。	• 2 つのサービスの証明書署名リクエスト (CSR) を作成するには、次の AWS CLI コマンドを使用します。
	
	`openssl req -out client_csr1.pem -new -newkey rsa:2048 -nodes -keyout client_private-key1.pem`
	
	`openssl req -out client_csr2.pem -new -newkey rsa:2048 -nodes -keyout client_private-key2.pem`
	
	このコマンドは、2 つのサービスの CSR とプライベートキーを返します。
	
	• サービスの証明書を発行するには、次のコマンドを実行して、作成したプライベート CA を使用します。
	
	`SERVICE_ONE_CERT_ARN=`aws acm-pca issue-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --csr fileb://client_csr1.pem \`
		DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	--signing-algorithm "SHA256WITHRSA" \ --validity Value=5,Type="YEARS" --query CertificateArn --output text`  echo "SERVICE_ONE_CERT_ARN: ${SERVICE_ONE_CERT_ARN}"  aws acm-pca get-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --certificate-arn ${SERVICE_ONE_CERT_ARN} \ | jq -r '.Certificate' > client_cert1.cert  SERVICE_TWO_CERT_ARN=`aws acm-pca issue-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --csr fileb://client_csr2.pem \ --signing-algorithm "SHA256WITHRSA" \ --validity Value=5,Type="YEARS" --query CertificateArn --output text` echo "SERVICE_TWO_CERT_ARN: ${SERVICE_TWO_CERT_ARN}"	

タスク	説明	必要なスキル
	`aws acm-pca get-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --certificate-arn ${SERVICE_TWO_CERT_ARN} \	jq -r '.Certificate' > client_cert2.cert` 詳細については、 AWS ドキュメントの「プライベートエンドエンティティ証明書を発行する」を参照してください。	

AWS サービスをプロビジョニングする

タスク	説明	必要なスキル
CloudFormation テンプレート AWS のサービス を使用してプロビジョニングします。	Virtual Private Cloud (VPC)、Amazon ECS クラスター、Amazon ECS サービス、Application Load Balancer、Amazon Elastic Container Registry (Amazon ECR) をプロビジョニングするには、CloudFormation テンプレートを使用します。	DevOps エンジニア
変数を取得します。	2 つのサービスが実行されている Amazon ECS クラスターがあることを確認します。リソースの詳細を取得し、変数	DevOps エンジニア

タスク	説明	必要なスキル

	として保存するには、次のコマンドを使用します。
	
	`export LoadBalancerDNS=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerDNS`].OutputValue')`
	
	`export ECRRepositoryUri=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`ECRRepositoryUri`].OutputValue')`
	
	`export ECRRepositoryServiceOneUri=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`ECRRepositoryServiceOneUri`].OutputValue')`
	
	`export ECRRepositoryServiceTwoUri=$(aws cloudformation`	

タスク	説明	必要なスキル
	describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`ECRRepository ServiceTwoUri`].Ou tputValue') export ClusterName= $(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`ClusterName`] .OutputValue') export BucketName= $(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`BucketName`]. OutputValue') export Service1L istenerArn=$(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`Service1Liste nerArn`].OutputVal ue')	

タスク	説明	必要なスキル
	export Service2ListenerArn=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`Service2ListenerArn`].OutputValue')	

タスク	説明	必要なスキル
CodeBuild プロジェクトを作成する。	CodeBuild プロジェクトを使用して Amazon ECS サービスの Docker イメージを作成するには、次の手順を実行します。 1. にサインインし AWS Management Console、https://console.aws.amazon.com/codesuite/codebuild/ で CodeBuild コンソールを開きます。 2. 新しいプロジェクトを作成します。Source で、作成した Git リポジトリを選択します。さまざまな種類の Git リポジトリ統合の詳細については、 AWS ドキュメントの「接続の使用」を参照してください。 3. 特権モードが有効になっていることを確認します。Docker イメージを構築するには、このモードが必要です。そうしないと、イメージは正常にビルドされません。 4. 各サービスで共有されているカスタム`buildspec.yaml` ファイルを使用します。 5. プロジェクト名と説明の値を指定します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	詳細については、 AWS ドキュメントの「 でビルドプロジェクトを作成する AWS CodeBuild」を参照してください。	

タスク	説明	必要なスキル

Docker イメージを構築します。	CodeBuild を使用してイメージビルドプロセスを実行できます。CodeBuild には、Amazon ECR とやり取りし、Amazon S3 を操作するためのアクセス許可が必要です。
	
	プロセスの一環として、Docker イメージがビルドされ、Amazon ECR レジストリにプッシュされます。テンプレートとコードの詳細については、「追加情報」を参照してください。
	
	(オプション) テスト目的でローカルでビルドするには、次のコマンドを使用します。
	
	`# login to ECR`
	`aws ecr get-login`
	`-password | docker`
	`login --username`
	`AWS --password-stdin`
	`$ECRRepositoryUri`
	
	`# build image for`
	`service one`
	`cd /service1`
	`aws s3 cp s3://$Buc`
	`ketName/serviceone/`
	`service1/ --recursive`
	`docker build -t`
	`$ECRRepositoryServ`
	`iceOneUri .`
	`docker push $ECRRepos`
	`itoryServiceOneUri`
		DevOps エンジニア

タスク	説明	必要なスキル

	`# build image for service two`
	`cd ../service2`
	`aws s3 cp s3://$BucketName/servicetwo/service2/ --recursive`
	`docker build -t $ECRRepositoryServiceTwoUri .`
	`docker push $ECRRepositoryServiceTwoUri`	

相互 TLS を有効にする

タスク	説明	必要なスキル
CA 証明書を Amazon S3 にアップロードします。	CA 証明書を Amazon S3 バケットにアップロードするには、次のコマンド例を使用します。 `aws s3 cp ca-cert.pem s3://$BucketName/acm-trust-store/`	AWS DevOps、DevOps エンジニア
トラストストアを作成します。	トラストストアを作成するには、次のコマンド例を使用します。 `TrustStoreArn=`aws elbv2 create-trust-store --name acm-pca-trust-certs \` `--ca-certificates-bundle-s3-bucket $BucketName \`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`--ca-certificates-bundle-s3-key acm-trust-store/ca-cert.pem --query 'TrustStores[].TrustStoreArn' --output text``	

クライアント証明書をアップロードします。	Docker イメージのクライアント証明書を Amazon S3 にアップロードするには、次のコマンド例を使用します。
	`# for service one`
	`aws s3 cp client_cert1.cert s3://$BucketName/serviceone/`
	`aws s3 cp client_private-key1.pem s3://$BucketName/serviceone/`
	
	`# for service two`
	`aws s3 cp client_cert2.cert s3://$BucketName/servicetwo/`
	`aws s3 cp client_private-key2.pem s3://$BucketName/servicetwo/`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

リスナーを変更します。	ALB で相互 TLS を有効にするには、次のコマンドを使用して HTTPS リスナーを変更します。
	
	`aws elbv2 modify-listener \ --listener-arn $Service1ListenerArn \ --certificates CertificateArn=$TLS_CERTIFICATE_ARN_TWO \ --ssl-policy ELBSecurityPolicy-2016-08 \ --protocol HTTPS \ --port 8080 \ --mutual-authentication Mode=verify,TrustStoreArn=$TrustStoreArn,IgnoreClientCertificateExpiry=false`
	
	`aws elbv2 modify-listener \ --listener-arn $Service2ListenerArn \ --certificates CertificateArn=$TLS_CERTIFICATE_ARN_TWO \ --ssl-policy ELBSecurityPolicy-2016-08 \ --protocol HTTPS \ --port 8090 \`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`--mutual-authentication Mode=verify,TrustStoreArn=$TrustStoreArn,IgnoreClientCertificateExpiry=false` 詳細については、 AWS ドキュメントのApplication Load Balancer での相互 TLS の設定」を参照してください。	

サービスを更新する

タスク	説明	必要なスキル
Amazon ECS タスク定義を更新します。	Amazon ECS タスク定義を更新するには、新しいリビジョンで imageパラメータを変更します。 それぞれのサービスの値を取得するには、前のステップで構築した新しい Docker イメージ Uri でタスク定義を更新します。`echo $ECRRepositoryServiceOneUri` または `echo $ECRRepositoryServiceTwoUri` `"containerDefinitions": [ { "name": "nginx",`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	<pre>"image":
	"public.ecr.aws/nginx/nginx:latest",
	# <----- change to new Uri
	"cpu": 0,</pre>
	詳細については、 AWS ドキュメントの「コンソールを使用した Amazon ECS タスク定義の更新」を参照してください。	

タスク	説明	必要なスキル
Amazon ECS サービスを更新します。	最新のタスク定義でサービスを更新します。このタスク定義は、新しく構築された Docker イメージの設計図であり、相互 TLS 認証に必要なクライアント証明書が含まれています。 サービスを更新するには、次の手順を使用します。 1. https://console.aws.amazon.com/ecs/v2 で Amazon ECS コンソールを開きます。 2. [Clusters] (クラスター) ページで、クラスターを選択します。 3. クラスターの詳細ページの「サービス」セクションで、サービスの横にあるチェックボックスを選択し、「更新」を選択します。 4. サービスで新しいデプロイを開始するには、[Force new deployment] (新しいデプロイの強制) を選択します。 5. タスク定義で、タスク定義ファミリーと最新のリビジョンを選択します。 6. [更新] を選択します。	AWS 管理者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	他のサービスのステップを繰り返します。	

アプリケーションにアクセスする

タスク	説明	必要なスキル
アプリケーション URL をコピーします。	Amazon ECS コンソールを使用してタスクを表示します。タスクステータスが実行中に更新されたら、タスクを選択します。タスクセクションで、タスク ID をコピーします。	AWS 管理者、AWS DevOps

アプリケーションをテストする	アプリケーションをテストするには、ECS Exec を使用してタスクにアクセスします。
	1. サービス 1 の場合は、次のコマンドを使用します。
	`container="nginx"`
	
	`ECS_EXEC_TASK_ARN="<TASK ARN>"`
	`aws ecs execute-command --cluster $ClusterName \`
	`--task $ECS_EXEC_TASK_ARN \`
	`--container $container \`
	`--interactive \`
	`--command "/bin/bash"`
		AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

	2. サービス 1 タスクのコンテナで、次のコマンドを使用して、内部ロードバランサーurl と、サービス 2 を指すリスナーポートを入力します。次に、クライアント証明書へのパスを指定してアプリケーションをテストします。
	`curl -kvs https://<internal-alb-url>:8090 --key /usr/local/share/ca-certificates/client.key --cert /usr/local/share/ca-certificates/client.crt`
	3. サービス 2 タスクのコンテナで、次のコマンドを使用して、内部ロードバランサーurlと、サービス 1 を指すリスナーポートを入力します。次に、クライアント証明書へのパスを指定してアプリケーションをテストします。
	`curl -kvs https://<internal-alb-url>:8080 --key /usr/local/share/ca-certificates/client.key --cert /usr/local/share/ca-certificates/client.crt`	

タスク	説明	必要なスキル
	Note curl コマンドの -kフラグ (の一部として-kvs) は、SSL 証明書の検証を無効にします。ドメイン名に一致する SSL 証明書を使用する場合、このフラグを削除して、適切な証明書の検証を有効にできます。	

関連リソース

Amazon ECS ドキュメント

- コンソールを使用した Amazon ECS タスク定義の作成
- Amazon ECS で使用するコンテナイメージの作成
- Amazon ECS クラスター
- の Amazon ECS AWS Fargate
- Amazon ECS ネットワーキングのベストプラクティス
- Amazon ECS サービス定義パラメータ

その他の AWS リソース

- AWS プライベート CA を使用して Application Load Balancer で mTLS を設定する方法を教えてください。（AWS re:Post）

追加情報

Dockerfile の編集

次のコードは、サービス 1 の Dockerfile で編集したコマンドを示しています。

```
FROM public.ecr.aws/nginx/nginx:latest
WORKDIR /usr/share/nginx/html
RUN echo "Returning response from Service 1: Ok" > /usr/share/nginx/html/index.html
ADD client_cert1.cert client_private-key1.pem /usr/local/share/ca-certificates/
RUN chmod -R 400 /usr/local/share/ca-certificates/
```

次のコードは、サービス 2 の Dockerfile で編集したコマンドを示しています。

```
FROM public.ecr.aws/nginx/nginx:latest
WORKDIR /usr/share/nginx/html
RUN echo "Returning response from Service 2: Ok" > /usr/share/nginx/html/index.html
ADD client_cert2.cert client_private-key2.pem /usr/local/share/ca-certificates/
RUN chmod -R 400 /usr/local/share/ca-certificates/
```

CodeBuild を使用して Docker イメージを構築する場合、buildspecファイルは CodeBuild ビルド番号を使用してイメージバージョンをタグ値として一意に識別します。次のbuildspec カスタムコードに示すように、要件に合わせてbuildspecファイルを変更できます。

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $ECR_REPOSITORY_URI
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
        # change the S3 path depending on the service
      - aws s3 cp s3://$YOUR_S3_BUCKET_NAME/serviceone/ $CodeBuild_SRC_DIR/ --
recursive
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $ECR_REPOSITORY_URI:latest .
```

```
      - docker tag $ECR_REPOSITORY_URI:latest $ECR_REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $ECR_REPOSITORY_URI:latest
      - docker push $ECR_REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      # for ECS deployment reference
      - printf '[{"name":"%s","imageUri":"%s"}]' $CONTAINER_NAME $ECR_REPOSITORY_URI:
$IMAGE_TAG > imagedefinitions.json

artifacts:
  files:
    - imagedefinitions.json
```

その他のパターン

- CAST Highlight を使用して、AWS クラウドへの移行に向けたアプリケーションの準備状況を評価
- AWS CloudFormation スタックと関連リソースの削除を自動化する
- AWS Service Catalog と を使用して、Gitflow 環境にホットフィックスソリューションをデプロイするための動的パイプライン管理を自動化する AWS CodePipeline
- AWS CDK を使用してマイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する
- GitHub Actions と Terraform を使用して Docker イメージを構築して Amazon ECR にプッシュする
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化
- Firelens ログルーターを使用して Amazon ECS 用のカスタムログパーサーを作成する
- Amazon ECS に Java マイクロサービス用の CI/CD パイプラインをデプロイ
- EC2 インスタンスプロファイルを使用して AWS Cloud9 から Amazon EKS クラスターをデプロイする
- Terraform を使用して、コンテナ化された Blu Age アプリケーションの環境をデプロイする
- Amazon SageMaker の推論パイプラインを使用して、前処理ロジックを単一エンドポイントの ML モデルにデプロイする
- AWS コードサービスと AWS KMS マルチリージョンキーを使用して、複数のアカウントとリージョンへのマイクロサービスのブルー/グリーンデプロイを管理
- AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。
- Oracle GlassFish から AWS Elastic Beanstalk へ移行する
- Amazon ECS で Oracle WebLogic から Apache Tomcat (ToMee) に移行する
- 組織間でデータを共有するための最小限の実行可能なデータスペースを設定する
- ASP.NET ウェブフォームアプリケーションを AWS で最新化
- AWS CloudFormation と AWS Config を使用して Amazon ECR リポジトリのワイルドカード権限をモニタリングする
- AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する
- Amazon S3 に Helm v3 チャートリポジトリを設定する

- cert-managerと Let's Encrypt を使用して Amazon EKS のアプリケーションのエンドツーエンド暗号化を設定
- Flux を使用して Amazon EKS マルチテナントアプリケーションのデプロイを簡素化する
- SageMaker AI と Hybrid を使用して、ローカル開発からスケーラブルな実験まで機械学習ワークフローを合理化する
- AWS Lambda を使用して六角形アーキテクチャで Python プロジェクトを構築する
- LocalStack および Terraform Tests を使用して AWS インフラストラクチャをテストする
- GPU がサポートするカスタム ML モデルを Amazon SageMaker でトレーニングしてデプロイする
- AWS Fargate WaitCondition フックコンストラクトを使用してリソースの依存関係とタスク実行を調整する
- Amazon Bedrock エージェントを使用して、テキストベースのプロンプトによる Amazon EKS でのアクセスエントリコントロールの作成を自動化する

サーバーレス

トピック

- AWS Amplify を使用してサーバーレスの React Native モバイルアプリを構築する
- 複数の SaaS 製品間のテナントを単一のコントロールプレーンで管理する
- 組織内でクロスアカウント Amazon EventBridge 接続を作成する
- で Kinesis Data Streams と Firehose を使用して DynamoDB レコードを Amazon S3 に配信する AWS CDK
- Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する
- psycopg2 ライブラリを にインポート AWS Lambda して PostgreSQL データベースとやり取りする
- Amazon API Gateway を Amazon SQS と統合して非同期 REST APIs を処理する
- Amazon API Gateway と AWS Lambda を使用してイベントを非同期的に処理する
- Amazon API Gateway と Amazon DynamoDB Streams を使用してイベントを非同期的に処理する
- Amazon API Gateway、Amazon SQS、および AWS Fargate でイベントを非同期的に処理する
- AWS Step Functions から AWS Systems Manager Automation タスクを同期的に実行する AWS Step Functions
- AWS Lambda 関数で Python を使用して S3 オブジェクトの並列読み取りを実行する
- から OpenSearch AWS Lambda にテレメトリデータを送信して、リアルタイムの分析と視覚化を行う
- セルベースのアーキテクチャ用にサーバーレスセルルーターを設定する
- VPC エンドポイントを介して Amazon S3 バケットへのプライベートアクセスを設定する
- Amazon Bedrock AWS Step Functions を使用して の状態をトラブルシューティングする
- サーバーレスアプローチを使用して AWS サービスを連結する
- その他のパターン

AWS Amplify を使用してサーバーレスの React Native モバイルアプリを構築する

作成者: Deekshitulu Pentakota (AWS)

概要

このパターンでは、AWS Amplify と以下の AWS サービスを使用して React Native モバイルアプリのサーバーレスバックエンドを作成する方法を説明しています。

- AWS AppSync
- Amazon Cognito
- Amazon DynamoDB

Amplify を使用してアプリケーションのバックエンドを設定してデプロイすると、Amazon Cognito はアプリケーションユーザーを認証し、アプリケーションへのアクセスを許可します。 次に、AWS AppSync はフロントエンドアプリおよびバックエンド DynamoDB テーブルとやり取りして、データを作成およびフェッチします。

> **Note**
>
> このパターンでは、シンプルなToDoList」アプリを例として使用しますが、同様の手順を使用して React Native モバイルアプリを作成できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI) がインストール済みおよび設定済み
- XCode (任意のバージョン)
- Microsoft Visual Studio (任意のバージョン、任意のコードエディター、任意のテキストエディター)
- Amplify Gurify に精通している
- Amazon Cognito に精通している

- AWS AppSync に精通している
- DynamoDB に精通している
- Node.js に精通している
- npm に精通している
- React と React Native に精通している
- JavaScript と ECMAScript 6 (ES6) に精通している
- GraphQL に精通している

アーキテクチャ

次の図は、AWS クラウドで React Native モバイルアプリのバックエンドを実行するためのアーキテクチャの例を示しています。

以下は、アーキテクチャ図を示しています。

1. Amazon Cognito はアプリユーザーを認証し、アプリへのアクセスを許可します。
2. 次に、AWS AppSync はフロントエンドアプリおよびバックエンドの DynamoDB テーブルとやり取りし、データを作成して取得します。

ツール

AWS サービス

- AWS Amplify は、フロントエンドのウェブおよびモバイルデベロッパーが AWS で迅速かつ簡単にフルスタックアプリケーションを構築できるようにする専用のツールと機能のセットです。
- AWS AppSync は、Amazon DynamoDB、AWS Lambda、および HTTP API を含む、複数のソースからのデータを組み合わせることで、アプリケーション開発者にとって便利でスケーラブルな GraphQL インターフェイスを提供します。
- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。

コード

このパターンで使用されているサンプルアプリケーションのコードは、GitHub aws-amplify-react-native-ios-todo-app リポジトリから入手できます。このサンプルファイルを使用するには、このパターンの [エピック] セクションの指示に従ってください。

エピック

React Native アプリを作成して実行します

タスク	説明	必要なスキル
React Native 開発環境のセットアップ	手順については、React Native ドキュメントの「開発環境のセットアップ」を参照してください。	アプリ開発者

iOS シミュレーターで ToDoList React Native モバイルアプリを作成して実行します。	1. 新しいターミナルウィンドウで、次のコマンドを実行し、ローカル環境に新しい React Native モバイルアプリプロジェクトディレクトリを作成します。
	`npx react-native init ToDoListAmplify`
	2. 以下のコマンドを実行し、プロジェクトのルートディレクトリに移動します。
	`cd ToDoListAmplify`
	3. 次のコマンドを実行して、アプリを実行します。
	`npx react-native run-ios`
		アプリ開発者

アプリの新しいバックエンド環境を初期化します。

タスク	説明	必要なスキル

AAmplify でアプリをサポートするために必要なバックエンドサービスを作成します。	1. ローカル環境で、プロジェクトのルートディレクトリ (ToDoListAmplify) から次のコマンドを実行します。
	
	`amplify init`
	
	2. アプリに関する情報の入力を求めるプロンプトが表示されます。ユースケースに基づいて必要な情報を入力します。次に、<Enter> キーを押します。
	
	このパターンで使用する TodoList アプリの設定には、以下の設定例を適用してください。
	
	React Native Amplify アプリの構成設定の例
	
	`? Name: ToDoListAmplify`
	
	`? Environment: dev`
	
	`? Default editor: Visual Studio Code`
	
	`? App type: javascript`
	
	`? Javascript framework : react-native`
	
	`? Source Directory Path: src`
		アプリ開発者

タスク	説明	必要なスキル
	``` ? Distribution Directory Path: / ? Build Command: npm run-script build ? Start Command: npm run-script start ? Select the authentic ation method you want to use: AWS profile ? Please choose the profile you want to use: default ``` 詳細については、Amplify Dev Center ドキュメントの「新しい Amplify バックエンドの作成」を参照してください。 **Note** `amplify init` コマンドは、AWS CloudFormation を使用して次のリソースをプロビジョニングします。 • 認証されたユーザーと認証されていないユーザーの AWS Identity and Access Management (IAM) ロール	

タスク	説明	必要なスキル
	(認証ロールと認証解除ロール) • デプロイ用の Amazon Simple Storage Service (Amazon S3) バケット (このパターンのサンプルアプリケーション Amplify Meta.json) • AAmplify ホスティングのバックエンド環境	

Amplify React Native アプリに Amazon Cognito 認証を追加する

タスク	説明	必要なスキル
Amazon Cognito 認証サービスを作成します。	1. ローカル環境で、プロジェクトのルートディレクトリ (ToDoListAmplify) から次のコマンドを実行します。 `amplify add auth` 2. 認証サービスの構成設定に関する情報を求めるプロンプトが表示されます。ユースケースに基づいて必要な情報を入力します。次に、<Enter> キーを押します。 このパターンで使用する TodoList アプリの設定には、以下の設定例を適用してください。	アプリ開発者

タスク	説明	必要なスキル

	認証サービスの設定例
	`? Do you want to use the default authentication and security configuration? \`
	`Default configuration`
	
	`? How do you want users to be able to sign in? \`
	`Username`
	
	`? Do you want to configure advanced settings? \`
	`No, I am done`
	
	Note
	`amplify add auth` コマンドは、プロジェクトのルートディレクトリ内のローカルフォルダ (amplify) に必要なフォルダ、ファイル、依存関係ファイルを作成します。このパターンで使用する TodoList アプリの設定では、この目的で aws-exports.js が作成されます。	

タスク	説明	必要なスキル

Amazon Cognito サービスを AWS クラウドにデプロイします。	1. プロジェクトのルートディレクトリから、次の Amplify CLI コマンドを実行します。
	
	`amplify push`
	
	2. デプロイを確認するプロンプトが表示されます。Yes と入力します。次に、<Enter> キーを押します。
	
	Note
	プロジェクトにデプロイされたサービスを表示するには、次のコマンドを実行して Amplify コンソールに移動します。
	
	`amplify console`
		アプリ開発者

タスク	説明	必要なスキル

React Native に必要な Amplify ライブラリをインストールし、iOS には CocoaPods 依存関係をインストールします。	1. プロジェクトのルートディレクトリから、次のコマンドを実行して、必要な Amplify オープンソースのクライアントライブラリをインストールします。
	
	`npm install aws-amplify aws-amplify-react-native \ amazon-cognito-identity-js @react-native-community/netinfo \ @react-native-async-storage/async-storage`
	2. iOS に必要な CocoaPods 従属関係をインストールするには、次のコマンドを実行します。
	
	`npx pod-install`
		アプリ開発者

タスク	説明	必要なスキル

Amplify サービスをインポートして設定します。	アプリのエントリポイントファイル (App.js など) で、次のコード行を入力することで、Amplify サービスの設定ファイルをインポートして読み込みます。
	
	`import Amplify from 'aws-amplify'`
	`import config from './src/aws-exports'`
	`Amplify.configure(config)`
	
	Note
	アプリのエントリポイントファイルに Amplify サービスをインポートした後にエラーが表示された場合は、アプリを停止してください。次に、Xcode を開き、プロジェクトの iOS フォルダーから TodoListAmplify.xcWorkspace を選択して、アプリを実行します。
		アプリ開発者

タスク	説明	必要なスキル
WithAuthenticator 高次コンポーネント (HOC) を使用するようにアプリのエントリポイントファイルを更新します。	**Note** `withAuthenticator` HOC は、数行のコードのみを使用して、アプリでサインイン、サインアップ、パスワードを忘れた場合のワークフローを提供します。詳細については、Amplify Dev Center で「オプション 1: ビルド済み UI コンポーネントを使用する」を参照してください。また、React ドキュメントの 高次コンポーネント についても説明します。 1. アプリのエントリポイントファイル (例: App.js) で、次のコード行を入力して `withAuthenticator` HOC をインポートして読み込みます。 `import { withAuthenticator } from 'aws-amplify-react-native'`	アプリ開発者

タスク	説明	必要なスキル

	2. 次のコードを入力して、WithAuthenticator HOC をエクスポートします。
	`export default withAuthenticator(App)`
	
	WithAuthenticator HOC コード例
	`import Amplify from 'aws-amplify'`
	`import config from './src/aws-exports'`
	`Amplify.configure(config)`
	`import { withAuthenticator } from 'aws-amplify-react-native';`
	
	`const App = () => {`
	`  return null;`
	`};`
	
	`export default withAuthenticator(App);`
	
	Note
	iOS Simulator では、Amazon Cognito サービスが提供する口	

タスク	説明	必要なスキル
	グイン画面がアプリに表示されます。	
認証サービスの設定をテストします。	iOS シミュレータで、以下の操作を行います。 1. 実際のメールアドレスを使用して、アプリ内で新規アカウントを作成します。次に、登録したメールアドレスに確認コードが送信されます。 2. 確認メールで受信したコードを使用して、アカウントの設定を確認します。 3. 作成したユーザー名とパスワードを入力します。次に、[サインイン] を選択します。[ようこそ] 画面が表示されます。 **Note** Amazon Cognito コンソールを開き、ID プールで新しいユーザーが作成されたかどうかを確認することもできます。	アプリ開発者

AWS AppSync API と DynamoDB データベースをアプリにコネクトします

タスク	説明	必要なスキル

AWS AppSync API と DynamoDB データベースを作成します。	1. AWS AppSync API をアプリケーションに追加し、プロジェクトのルートディレクトリから次の Amplify CLI コマンドを実行して DynamoDB データベースを自動的にプロビジョニングします。
	
	`amplify add api`
	
	2. API およびデータベースの構成設定に関する情報を入力するよう求めるプロンプトが表示されます。ユースケースに基づいて必要な情報を入力します。次に、<Enter> キーを押します。Amplify CLI は、テキストエディタで GraphQL スキーマファイルを開きます。
	
	このパターンで使用する TodoList アプリの設定には、以下の設定例を適用してください。
	
	API とデータベースの設定例
	
	`? Please select from one of the below mentioned services: \ GraphQL`
		アプリ開発者

タスク	説明	必要なスキル
	? Provide API name: todolistamplify ? Choose the default authorization type for the API \ Amazon Cognito User Pool Do you want to use the default authentication and security configura tion ? Default configuration How do you want users to be able to sign in? \ Username Do you want to configure advanced settings? \ No, I am done. ? Do you want to configure advanced settings for the GraphQL API \ No, I am done. ? Do you have an annotated GraphQL schema? \ No ? Choose a schema template: \ Single object with fields (e.g., “Todo” with ID, name, description)	

タスク	説明	必要なスキル

	``
	? Do you want to edit
	 the schema now? \
	Yes
	``
	
	GraphQL スキーマの例
	
	``
	 type Todo @model {
	   id: ID!
	   name: String!
	   description: String
	}
	``	

タスク	説明	必要なスキル
AWS AppSync API をデプロイします。	1. プロジェクトのルートディレクトリで、次の Amplify CLI コマンドを実行します。 `amplify push` 2. API およびデータベースの構成設定に関する情報を入力するよう求めるプロンプトが表示されます。ユースケースに基づいて必要な情報を入力します。次に、<Enter> キーを押します。これで、アプリケーションが AWS AppSync API とやり取りできるようになりました。 このパターンで使用する TodoList アプリの設定には、以下の設定例を適用してください。 AWS AppSync API の設定例 **Note** 次の設定では、AWS AppSync に GraphQL API を作成し、Dynamo DB に Todo テーブルを作成します。	アプリ開発者

タスク	説明	必要なスキル
	``` ? Are you sure you want to continue? Yes ? Do you want to generate code for your newly created GraphQL API Yes ? Choose the code generation language target javascript ? Enter the file name pattern of graphql queries, mutations and subscriptions src/ graphql/**/*.js ? Do you want to generate/update all possible GraphQL operations - \ queries, mutations and subscriptions Yes ? Enter maximum statement depth \ [increase from default if your schema is deeply nested] 2 ```	

タスク	説明	必要なスキル
アプリのフロントエンドを AWS AppSync API に接続します。	このパターンで提供されているサンプル TodoList アプリケーションを使用するには、aws-amplify-react-native-ios-todo-app GitHub リポジトリの App.js ファイルからコードをコピーします。次に、サンプルコードをローカル環境に統合します。 リポジトリの App.js ファイルにあるサンプルコードは以下の処理を行います。 • [タイトル] フィールドと [説明] フィールドを含む [ToDo 項目] を作成するためのフォームを表示します。 • To Do 項目 ([タイトル] と [説明]) のリストを表示します。 • `aws-amplify` メソッドを使用してデータを投稿して取得します。	アプリ開発者

関連リソース

- 「AWS Amplify」
- Amazon Cognito
- AWS AppSync
- 「Amazon DynamoDB」
- 「React」 (React ドキュメント)

複数の SaaS 製品間のテナントを単一のコントロールプレーンで管理する

作成者：Ramanna Avancha (AWS)、Jenifer Pascal (AWS)、Kishan Kavala (AWS) と Anusha Mandava (AWS)

概要

このパターンは、AWS クラウドの単一のコントロールプレーンで、複数のサービスを提供する SaaS 製品間のテナントのライフサイクルを管理する方法を示しています。提供されるリファレンスアーキテクチャは、組織が個々の SaaS 製品間で冗長な共有機能の実装を減らし、ガバナンスの効率性を大幅に向上させるのに役立ちます。

大手企業は、さまざまなビジネスユニット間で複数のSaaS製品を保有できます。多くの場合、これらの製品は、さまざまなサブスクリプションレベルの外部テナントが使用できるようにプロビジョニングする必要があります。共通のテナントソリューションがない場合、IT 管理者はコア製品の機能開発に集中するのではなく、複数のSaaS API 間で差別化されていない機能の管理に時間を費やす必要があります。

このパターンで提供される共通テナントソリューションは、次のような組織の共有 SaaS 製品機能の多くを一元管理するのに役立ちます。

- セキュリティ
- テナントプロビジョニング
- テナントデータストレージ
- テナントコミュニケーション
- 製品管理
- マトリックス記録とモニタリング

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon Cognito または第三者の ID プロバイダー (IdP) に関する知識
- Amazon API Gateway に関する知識
- AWS Lambda に関する知識
- Amazon DynamoDB に関する知識

- AWS Identity and Access Management (IAM)
- AWS Step Functions に関する知識
- AWS CloudTrail と Amazon CloudWatch に関する知識
- Python ライブラリとコードに関する知識
- さまざまなタイプのユーザー (組織、テナント、管理者とアプリケーションユーザー)、サブスクリプションモデルとテナント分離モデルを含む SaaS API に関する知識
- 組織のマルチプロダクト SaaS 要件とマルチテナントサブスクリプションに関する知識

制約事項

- 一般的なテナントソリューションと個々の SaaS 製品との統合は、このパターンには含まれていません。
- このパターンでは、Amazon Cognito サービスは単一の AWS リージョンにのみデプロイされます。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon API Gateway
- Amazon Cognito
- AWS CloudTrail
- Amazon CloudWatch
- Amazon DynamoDB
- IAM
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Notification Service (Amazon SNS)
- AWS Step Functions

ターゲットアーキテクチャ

次の図は、AWS クラウドの 1 つのコントロールプレーンで複数の SaaS 製品間におけるテナントライフサイクルを管理するためのワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. AWS ユーザーは、API ゲートウェイエンドポイントを呼び出して、テナントのプロビジョニング、製品のプロビジョニングまたは管理関連のアクションを開始します。
2. ユーザーは、Amazon Cognito ユーザープールまたは別の IdP から取得したアクセストークンにより認証されます。
3. 個々のプロビジョニングまたは管理タスクは、API Gateway API エンドポイントと統合された Lambda 関数により、実行されます。
4. 共通テナントソリューション (テナント、製品、ユーザー用) の管理 API は、必要な入力パラメータ、ヘッダー、トークンをすべて収集します。その後、管理 API は関連する Lambda 関数を呼び出します。
5. 管理 API と Lambda 関数の両方の IAM 権限は、IAM サービスにより、検証されます。
6. Lambda 関数は、DynamoDB と Amazon S3 のカタログ (テナント、製品、ユーザー用) のデータを保存し、取得します。
7. アクセス権限が検証されると、特定のタスクを実行するために、AWS Step Functions ワークフローが呼び出されます。この図の例は、テナントのプロビジョニングワークフローを示しています。
8. 個々の AWS Step Functions ワークフロータスクは、あらかじめ決められたワークフロー (ステートマシン) で実行されます。
9. 各ワークフロータスクに関連付けられた Lambda 関数の実行に必要かつ重要なデータは、DynamoDB または Amazon S3 から取得されます。他の AWS リソースは、AWS CloudFormation テンプレートでプロビジョニングする必要がある場合があります。
10. 必要に応じて、ワークフローは特定の SaaS 製品用に追加の AWS リソースをプロビジョニングするリクエストをその製品の AWS アカウントに送信します。
11. リクエストが成功または失敗する場合、ワークフローはステータス更新をメッセージとして Amazon SNS トピックに発行します。
12. Amazon SNS は Step Functions ワークフローの Amazon SNS トピックにサブスクライブされています。
13. 次に、Amazon SNS はその後、ワークフローステータスの更新を AWS ユーザーに送り返します。
14. API 呼び出しの監査証跡を含む各 AWS サービスのアクションのログは CloudWatch に送信されます。CloudWatch では、ユースケースごとに特定のルールとアラームを設定できます。

15.ログは監査の目的で Amazon S3 バケットにアーカイブされます。

自動化とスケール

このパターンは、CloudFormation テンプレートで、共通テナントソリューションのデプロイの自動化に役立ちます。テンプレートは、関連するリソースをすばやくスケールアップまたはスケールダウンするのに役立ちます。

詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormation テンプレートの操作」を参照してください。

ツール

AWS サービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。
- 「AWS CloudTrail」は、AWS アカウントのガバナンス、コンプライアンス、運用のリスクの監査をサポートします。
- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- AWS Step Functionsは、AWS Lambda関数と他のAWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。

ベストプラクティス

このパターンのソリューションでは、単一のコントロールプレーンで複数のテナントのオンボーディングを管理し、複数のSaaS製品へのアクセスをプロビジョニングします。コントロールプレーンは、管理ユーザーがほかの 4 つの機能に固有のプレーンを管理することに役立ちます。

- セキュリティプレーン
- ワークフロープレーン
- コミュニケーションプレーン
- ログ記録とモニタリング

エピック

セキュリティプレーンの設定

タスク	説明	必要なスキル
マルチテナント SaaS プラットフォームの要件を確立します。	以下の要件の詳細を確立します。 • テナント • [ユーザー] • ロール • SaaS 製品 • サブスクリプション • プロファイル交換	クラウドアーキテクト、AWS システム管理者
Amazon Cognito サービスを設定します。	「Amazon Cognito 開発者ガイド」の「Amazon Cognito の使用開始方法」に記載されている指示に従ってください。	クラウドアーキテクト
必要な IAM ポリシーを設定します。	ユースケースに必要なIAMポリシーを作成します。その後、ポリシーを Amazon Cognito の IAM ロールにマッピングします。	クラウド管理者、クラウドアーキテクト、AWS IAM sセキュリティ

タスク	説明	必要なスキル
	詳細については、「Amazon Cognito 開発者ガイド」の「ポリシーを使用したアクセスの管理」と「ロールベースアクセスコントロール」を参照してください。	
必要な API アクセス権限を設定します。	IAM ロールとポリシーおよび Lambda 認証で API Gateway のアクセス権限を設定します。 手順については、「Amazon API Gateway 開発者ガイド」の以下のセクションを参照してください。 • 「IAM アクセス許可により API へのアクセスを制御する」 • 「API Gateway Lambda オーソライザーの使用」	クラウド管理者、クラウドアーキテクト

データプレーンの設定

タスク	説明	必要なスキル
必要なデータカタログを作成します。	1. ユーザーカタログのデータを保存する DynamoDB テーブルを作成します。ユーザー属性とロールを含めることを確保してください。また、カタログテーブルでデータモデリングを行い、各ユーザーとロールの	DBA

タスク	説明	必要なスキル
	必須属性とオプション属性を維持するようにしてください。 2. 製品カタログのデータを保存する DynamoDB テーブルを作成します。SaaS 製品の特定のユースケースをモデル化することを確保してください。 3. テナントカタログのデータを保存する DynamoDB テーブルを作成します。テナント、マルチ SaaS サブスクリプションの製品とライセンスおよびタグのサブスクリプションモデルを必ず設定してください。 詳細については、Amazon DynamoDB 開発者ガイドの「DynamoDB のセットアップ」を参照してください。	

コントロールプレーンの設定

タスク	説明	必要なスキル
Lambda 関数と API Gateway API を作成して、必要なコントロールプレーンタスクを実行します。	Lambda 関数と API Gateeway API を個別に作成して、次の項目の追加、削除と管理を行います。 • [ユーザー] • テナント	アプリ開発者

タスク	説明	必要なスキル
	• 製品 詳細については、AWS Lambda 開発者ガイドの「Amazon API Gateway での AWS Lambda の使用」参照してください。	

ワークフロープレーンを設定

タスク	説明	必要なスキル
AWS Step Functions ワークフローが実行する必要のあるタスクを特定します。	次の AWS Step Functions ワークフロー要件の詳細を特定して文書化します。 • [ユーザー] • テナント • 製品 ⚠ Important 主要な利害関係者が要件を承認していることを確認します。	アプリ所有者
必要な AWS Step Functions ワークフローを作成します。	1. ユーザー、テナントと製品に必要なワークフローを AWS Step Functions で作成します。詳細については、AWS Step Functions デベロッパーガイドを参照してください。	アプリ開発者、ビルドリード

タスク	説明	必要なスキル
	2. 再試行とエラー処理のメカニズムを特定します。詳細については、AWS ブログの「エラーの処理、再試行、Step Function ステートマシンへのアラートの追加」を参照してください。 3. Lambda 関数でワークフロー天順を実装します。手順については、AWS Step Functions 開発者ガイドの「Lambda を使用する Step Functions ステートマシンの作成」を参照してください。 4. 必要に応じて、外部サービスを AWS Step Functions と統合します。 5. 各ワークフローのステータスを DynamoDB テーブルで管理し、Amazon SNS を使用して各ワークフローのステータスを連絡します。	

コミュニケーションプレーンの設定

タスク	説明	必要なスキル
Amazon SNS トピックを作成します。	Amazon SNS トピックを作成して、以下に関する通知を受信します。 • ワークフローステータス • エラー	アプリ所有者、Cloud アーキテクト

タスク	説明	必要なスキル
	• 再試行 詳細については、Amazon SNS 開発者ガイドの「SNS トピックの作成」を参照してください。	
エンドポイントを各 Amazon SNS トピックにサブスクライブします。	Amazon SNS トピックにパブリッシュされたメッセージを受信するには、各トピックにエンドポイントをサブスクライブする必要があります。 詳細については、Amazon SNS 開発者ガイドの「Amazon SNS トピックのサブスクライブ」を参照してください。	アプリ開発者、Cloud アーキテクト

記録プレーンとモニタリングプレーンの設定

タスク	説明	必要なスキル
共通テナントソリューションの各コンポーネントの記録を有効にします。	作成した共通テナントソリューション内の各リソースの記録をコンポーネントレベルで有効にします。 手順については、以下を参照してください。 • 「API Gateway REST API または WebSocket API のトラブルシューティングのために CloudWatch Logs を	アプリ開発者、AWS システム管理者、クラウド管理者

タスク	説明	必要なスキル
	有効にする方法」(AWS ナレッジセンター) • 「CloudWatch Logs を使用した記録」(AWS Step Functions 開発者ガイド) • 「Python での AWS Lambda 関数の記録」(AWS Lambda 開発者ガイド) • 「Amazon Cognito での口グ記録とモニタリング」(Amazon Cognito 開発者ガイド) • 「Amazon CloudWatch によるモニタリング」(Amazon DynamoDB 開発者ガイド) **Note** IAM ポリシーを使用して、各リソースのログを一元化されたログ記録アカウントに統合できます。詳細については、「集中記録と複数アカウントのセキュリティガードレール」を参照してください。	

共通テナントソリューションのプロビジョニングとデプロイ

タスク	説明	必要なスキル
CloudFormation テンプレートを作成します。	CloudFormation テンプレートを使用して、共通テナントソリューション全体とそのすべてのコンポーネントのデプロイとメンテナンスを自動化します。 詳細については、「AWS CloudFormation ユーザーガイド」を参照してください。	アプリ開発者、DevOpsエンジニア、CloudFormation 開発者

関連リソース

- 詳細については、Amazon API Gateway デベロッパーガイドの「Amazon Cognito ユーザープールをオーソライザーとして使用して REST API へのアクセスを制御する」を参照してください
- 「API Gateway Lambda オーソライザーを使用する」(Amazon API Gateway 開発者ガイド)
- 「Amazon Cognito ユーザープール」(Amazon Cognito 開発者ガイド)
- 「クロスアカウントクロスリージョン CloudWatch コンソール」(Amazon CloudWatch ユーザーガイド)

組織内でクロスアカウント Amazon EventBridge 接続を作成する

作成者: Sam Wilson (AWS) と Robert Stone (AWS)

概要

大規模な分散システムでは、Amazon EventBridge を使用して、 AWS Organizations 組織内のさまざまな Amazon Web Services (AWS) アカウント間で状態の変化を通信します。ただし、EventBridge は通常、同じ 内のエンドポイントまたはコンシューマーのみをターゲットにできます AWS アカウント。例外は、別のアカウントのイベントバスです。そのイベントバスは有効なターゲットです。別のアカウントのイベントバスからイベントを消費するには、イベントをソースアカウントのイベントバスから送信先アカウントのイベントバスにプッシュする必要があります。異なる 内のアプリケーション間で重要なイベントを管理する際の課題を回避するには AWS アカウント、このパターンで示されている推奨アプローチを使用します。

このパターンは、 AWS Organizations 組織 AWS アカウント 内の複数の が関与する EventBridge でイベント駆動型アーキテクチャを実装する方法を示しています。このパターンでは AWS Cloud Development Kit (AWS CDK) Toolkit と を使用します AWS CloudFormation。

EventBridge には、イベントを受信、フィルタリング、変換、ルーティング、配信するのに役立つサーバーレスイベントバスが用意されています。EventBridge は、イベント駆動型アーキテクチャの重要なコンポーネントであり、メッセージのプロデューサーとそれらのメッセージのコンシューマーの分離をサポートしています。単一のアカウントでは、これは簡単です。マルチアカウント構造では、1 つのアカウントのイベントバス上のイベントを同じ組織内の他のアカウントで消費するために、追加の考慮事項が必要です。

プロデューサーとコンシューマーのアカウント固有の考慮事項については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- 少なくとも 2 つの が関連付けられている AWS Organizations 組織 AWS アカウント
- 両方の の AWS Identity and Access Management (IAM) ロール。 AWS アカウント を使用して、両方の でインフラストラクチャをプロビジョニングできます AWS アカウント 。 AWS CloudFormation
- Git がローカルにインストールされている
- AWS Command Line Interface （ AWS CLI) ローカルにインストールされている

- AWS CDK ローカルにインストールされ、両方の でブートストラップされている AWS アカウント

製品バージョン

このパターンは、次のツールとバージョンを使用して構築およびテストされています。

- AWS CDK ツールキット 2.126.0
- Node.js 18.19.0
- npm 10.2.3「」
- Python 3.12

このパターンは、任意のバージョンの AWS CDK v2 または npm で機能します。Node.js バージョン 13.0.0 から 13.6.0 は と互換性がありません AWS CDK。

アーキテクチャ

ターゲット アーキテクチャ

次の図は、あるアカウントからイベントをプッシュし、別のアカウントで消費するためのアーキテクチャワークフローを示しています。

ワークフローには以下のステップが含まれます。

1. ソースアカウントのプロデューサー AWS Lambda 関数は、アカウントの EventBridge イベントバスにイベントを配置します。
2. クロスアカウント EventBridge ルールは、イベントを送信先アカウントの EventBridge イベントバスにルーティングします。
3. 送信先アカウントの EventBridge イベントバスには、コンシューマー Lambda 関数を呼び出すターゲット Lambda ルールがあります。

ベストプラクティスは、コンシューマー Lambda 関数の失敗した呼び出しを処理するためにデッドレターキュー (DLQ) を使用することです。ただし、DLQ はわかりやすくするためにこのソリューションから省略されています。ワークフローに DLQ を実装し、ワークフローが障害から回復する機能を向上させる方法の詳細については、AWS Lambda 「エラー処理パターンの実装」ブログ記事を参照してください。

自動化とスケール

AWS CDK は、必要なアーキテクチャを自動的にプロビジョニングします。EventBridge は、 に応じて 1 秒あたり数千レコードにスケールできます AWS リージョン。詳細については、Amazon EventBridge のクォータドキュメントを参照してください。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。このパターンでは、 AWS CDK アプリケーションを操作するのに役立つコマンドラインクラウド開発キットである AWS CDK Toolkit を使用します。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。

その他のツール

- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub cross-account-eventbridge-in-organization リポジトリで入手できます。

ベストプラクティス

EventBridge を使用する際のベストプラクティスについては、以下のリソースを参照してください。

- Amazon EventBridge イベントパターンのベストプラクティス
- Amazon EventBridge でルールを定義する際のベストプラクティス

エピック

ローカル AWS CDK デプロイ環境を準備する

タスク	説明	必要なスキル
ソースアカウントと送信先アカウントのローカル認証情報を設定します。	「新しい設定と認証情報の設定」を確認し、環境に最も適した認証と認証情報の方法を使用します。 **Important** ソースアカウント認証と送信先アカウント認証の両方 AWS CLI に を設定してください。 これらの手順は、 `sourceAccount` と の 2 つの AWS プロファイルをローカルで設定していることを前提としていますdestinationAccount 。	アプリ開発者
両方をブートストラップします AWS アカウント。	アカウントをブートストラップするには、次のコマンドを実行します。	アプリ開発者

タスク	説明	必要なスキル
	`cdk bootstrap --profile sourceAccount` `cdk bootstrap --profile destinationAccount`	

パターンコードをクローンします。	リポジトリのクローンを作成するには、次のコマンドを実行します。
	`git clone git@github.com:aws-samples/aws-cdk-examples.git`
	次に、 ディレクトリを新しくクローンされたプロジェクトフォルダに変更します。
	`cd aws-cdk-examples/python/cross-account-eventbridge-in-organization`
		アプリ開発者

ソースアカウントに ProducerStack をデプロイする

タスク	説明	必要なスキル
AWS Organizations とアカウントの詳細`cdk.json`を使用して を変更します。	プロジェクトのルートフォルダで、 に次の変更を加えます`cdk.json`。 • `organization_id` – デプロイに関係するアカウントの Organizations ID	アプリ開発者

タスク	説明	必要なスキル
	• `event_bus_name` – `CrossAccount` 、または名前を変更できます • `rules[].targets[].arn` – 消費アカウント (送信先アカウント) の AWS アカウント ID	
ProducerStack リソースをデプロイします。	プロジェクトのルートディレクトリから次のコマンドを実行します。 `cdk deploy ProducerStack --profile sourceAccount` プロンプトが表示されたら、 を介して作成された新しい IAM ロールとその他のセキュリティ関連のアクセス許可を受け入れます AWS CloudFormation。	アプリ開発者

タスク	説明	必要なスキル
ProducerStack リソースがデプロイされていることを確認します。	リソースを確認するには、以下を実行します。 1. ソースアカウントの AWS Management Console で、CloudFormation を選択します。 2. スタックのリストから、ProducerStack を選択します。 3. スタック情報タブで、スタックのステータスが であることを確認しますCREATE_COMPLETE 。必要に応じて、リソースタブで、設定されたリソースを確認します。	アプリ開発者

ConsumerStack を送信先アカウントにデプロイする

タスク	説明	必要なスキル
ConsumerStack リソースをデプロイします。	プロジェクトのルートディレクトリから次のコマンドを実行します。 `cdk deploy ConsumerStack --profile destinationAccount` プロンプトが表示されたら、 を介して作成された新しい IAM ロールとその他のセキュリティ関連のアクセ	アプリ開発者

タスク	説明	必要なスキル
	ス許可を受け入れます AWS CloudFormation。	
ConsumerStack リソースがデプロイされていることを確認します。	1. 送信先アカウントのコンソールで、CloudFormation を選択します。 2. スタックのリストから、ConsumerStack を選択します。 3. スタック情報タブで、スタックのステータスが であることを確認しますCREATE_COMPLETE 。必要に応じて、リソースタブで、設定されたリソースを確認します。	アプリ開発者

イベントの生成と消費

タスク	説明	必要なスキル
プロデューサー Lambda 関数を呼び出します。	1. ソースアカウントのコンソールで、Lambda を選択します。 2. 関数のリストから、ProducerStack-ProducerLambdaXXXX を選択します (XXXX は AWS CDK によって自動的に生成される一連の文字を表します)。 3. [テスト] タブを選択します。	アプリ開発者

タスク	説明	必要なスキル
	4. テストイベントセクションで、テストを選択します。 イベント JSON テキストエリアの内容は、ペイロードとして Lambda 関数に提供される任意の有効な JSON にすることができます。この場合、デフォルトで指定された JSON で十分です。 5. 実行関数: 成功したメッセージが、テストイベントセクションの上にある緑色のバナーに表示されることを確認します。	

タスク	説明	必要なスキル
イベントが受信されたことを確認します。	1. 送信先アカウントのコンソールで、Lambda を選択します。 2. 関数のリストから、ConsumerStack-ConsumerLambdaXXXX を選択します (XXXX は AWS CDK によって自動的に生成される一連の文字を表します)。 3. [Monitor] (モニタリング) タブを選択します。 4. Monitor セクションで、CloudWatch ログの表示」を選択します。 5. 新しく開いたタブで、最新のログストリームのログストリーム名を選択します。 6. 次のようなログステートメントが表示されることを確認します。 `[DEBUG] 2024-04-08T19:08:10.091Z 9c16844a-f9de-444d-b621-86afe64d4cc8 Event: {'version':'0', 'id':'0b9faa96-973f-8be2-ecf8-75e4f328b980', 'detail-type':'TestType', 'source':'Producer', 'account'`	アプリ開発者

タスク	説明	必要なスキル
	:'XXXXXXXXXXXX', 'time':'2024-04-08 T19:08:09Z', 'region':'us-east-1', 'resources': [], 'detail': {'key1':'value1', 'key2':'value2', 'key3':'value3'}}	

クリーンアップ

タスク	説明	必要なスキル
ConsumerStack リソースを破棄します。	このパターンをテストとして使用している場合は、追加コストが発生しないように、デプロイされたリソースをクリーンアップします。 プロジェクトのルートディレクトリから次のコマンドを実行します。 `cdk destroy ConsumerStack --profile destinationAccount` スタックの削除を確認するメッセージが表示されます。	アプリ開発者
ProducerStack リソースを破棄します。	プロジェクトのルートディレクトリから次のコマンドを実行します。	アプリ開発者

タスク	説明	必要なスキル
	`cdk destroy ProducerStack --profile sourceAccount` スタックの削除を確認するメッセージが表示されます。	

トラブルシューティング

問題	ソリューション
送信先アカウントでイベントは受信されませんでした。	1. 指定された Organizations ID が正しいことを確認します。 2. ソースアカウントが提供された組織の一部であることを確認します。 3. ソースアカウントのイベントバスルールが、送信先アカウントの正しい情報にマッピングされていることを確認します。
コンソールから Lambda 関数を呼び出すと、次のエラーが返されます。 `User: arn:aws:iam::123456789012:user/XXXXX is not authorized to perform: lambda:Invoke`	AWS アカウント 管理者に連絡して、`ProducerStack-ProducerLambdaXXXX` Lambda 関数に対する適切な `lambda:Invoke` アクション権限を受け取ります。

関連リソース

リファレンス

- AWS Organizations ユーザーガイド
- Amazon EventBridge のイベントパターン
- Amazon EventBridge のルール

チュートリアルと動画

- チュートリアル: 組織の作成と設定
- AWS re:Invent 2023 - Amazon EventBridge を使用した高度なイベント駆動型パターン (COM301-R)

追加情報

プロデューサールール

ソースアカウントでは、EventBridge イベントバスが作成され、プロデューサーからのメッセージを受け入れます (アーキテクチャセクションを参照) 。このイベントバスには、IAM アクセス許可が付随するルールが作成されます。ルールは、次のcdk.json構造に基づいて、送信先アカウントの EventBridge イベントバスをターゲットにします。

```
"rules": [
  {
    "id": "CrossAccount",
    "sources": ["Producer"],
    "detail_types": ["TestType"],
    "targets": [
      {
        "id": "ConsumerEventBus",
        "arn": "arn:aws:events:us-east-2:012345678901:event-bus/CrossAccount"
      }
    ]
  }
]
```

消費するイベントバスごとに、イベントパターンとターゲットイベントバスを含める必要があります。

イベントパターン

イベントパターンは、このルールが適用されるイベントをフィルタリングします。この例では、イベントソースとレコードは、ソースアカウントのイベントバスから送信先アカウントのイベントバスに送信するイベントdetail_typesを識別します。

ターゲットイベントバス

このルールは、別のアカウントに存在するイベントバスを対象としています。ターゲットイベントバスを一意に識別するには、完全な arn (Amazon リソースネーム) が必要です。 idは、 で使用される論理 ID です AWS CloudFormation。ターゲットイベントバスは、ターゲットルールの作成時に実際に存在する必要はありません。

送信先アカウント固有の考慮事項

送信先アカウントでは、ソースアカウントのイベントバスからメッセージを受信する EventBridge イベントバスが作成されます。ソースアカウントからイベントを発行できるようにするには、リソースベースのポリシーを作成する必要があります。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowOrgToPutEvents",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "events:PutEvents",
    "Resource": "arn:aws:events:us-east-2:012345678901:event-bus/CrossAccount",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalOrgID": "o-XXXXXXXXX"
      }
    }
  }]
}
```

権限を付与することは特に重要です。これによりevents:PutEvents、同じ組織内の他のアカウントがこのイベントバスにイベントを発行できるようになります。を組織 ID aws:PrincipalOrgIdとして設定すると、必要なアクセス許可が付与されます。

イベントパターン

含まれているイベントパターンは、ユースケースに合わせて変更できます。

```
rule = events.Rule(
    self,
    self.id + 'Rule' + rule_definition['id'],
    event_bus=event_bus,
    event_pattern=events.EventPattern(
        source=rule_definition['sources'],
        detail_type=rule_definition['detail_types'],
```

```
    )
)
```

不要な処理を減らすには、イベントパターンで、送信先アカウントによって処理されるイベントのみが送信先アカウントのイベントバスに送信されるように指定する必要があります。

リソースベースのポリシー

この例では、組織 ID を使用して、送信先アカウントのイベントバスにイベントを配置できるアカウントを制御します。ソースアカウントの指定など、より制限の厳しいポリシーの使用を検討してください。

EventBridge クォータ

次のクォータに注意してください。

- イベントバスあたり 300 ルールがデフォルトのクォータです。これは必要に応じて拡張できますが、ほとんどのユースケースに合わせる必要があります。
- ルールごとに 5 つのターゲットが最大許容数です。アプリケーションアーキテクトは、イベントパターンをきめ細かく制御できるように、送信先アカウントごとに個別のルールを使用することをお勧めします。

で Kinesis Data Streams と Firehose を使用して DynamoDB レコードを Amazon S3 に配信する AWS CDK

シャシャンク・シュリバスタヴァ(AWS)とダニエル・マトゥキ・ダ・クーニャ(AWS)によって作成されました

概要

このパターンは、Amazon Kinesis Data Streams と Amazon Data Firehose を使用して Amazon DynamoDB から Amazon Simple Storage Service (Amazon S3) にレコードを配信するためのサンプルコードとアプリケーションを提供します。このパターンのアプローチでは AWS Cloud Development Kit (AWS CDK) L3 コンストラクトを使用し、Amazon Web Services (AWS) クラウド上のターゲット S3 バケットにデータが配信される AWS Lambda 前に でデータ変換を実行する方法の例を示します。

Kinesis Data Streams は、DynamoDB テーブルの項目レベルの変更を記録し、それらを必須の Kinesis Data Streams にレプリケートします。アプリケーションは Kinesis Data Streams にアクセスし、項目レベルの変更をほぼリアルタイムで表示できます。Kinesis Data Streams は、Firehose や Amazon Managed Service for Apache Flink などの他の Amazon Kinesis サービスへのアクセスも提供します。つまり、リアルタイムのダッシュボードの提供、アラートの生成、動的な料金設定、広告の実装、高度なデータ分析の実行を行うアプリケーションを構築できます。

このパターンは、データ統合のユースケースに使用できます。たとえば、輸送車両や産業機器は、DynamoDB テーブルに大量のデータを送信できます。その後、このデータを変換して Amazon S3 でホストされているデータレイクに保存できます。その後、Amazon Athena、Amazon Redshift Spectrum、Amazon Rekognition、 などのサーバーレスサービスを使用して、データのクエリと処理を行い、潜在的な欠陥を予測できます AWS Glue。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み。詳細については、AWS CLI ドキュメントの「 の開始方法 AWS CLI」を参照してください。
- Node.js (18.x+) と npm、インストールおよび設定済み。詳細については、npmドキュメントの Node.js と npm のダウンロードとインストールを参照してください。

- aws-cdk (2.x+)、インストールおよび設定済み。詳細については、 AWS CDK ドキュメントの「 の開始方法 AWS CDK」を参照してください。
- GitHub aws-dynamodb-kinesisfirehose-s3-インジェスト リポジトリは、ローカルマシンにクローン化されて設定されています。
- DynamoDB テーブルの既存のサンプルデータ。データは以下のフォーマットを使用する必要があります：`{"SourceDataId": {"S": "123"},"MessageData":{"S": "Hello World"}}`

アーキテクチャ

次の図は、Kinesis Data Streams と Firehose を使用して DynamoDB から Amazon S3 にレコードを配信するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. データは Amazon API Gateway を DynamoDB のプロキシとして使用して取り込まれます。他のソースを使用して DynamoDB にデータを取り込むこともできます。
2. アイテムレベルの変更は、Kinesis Data Streams でほぼリアルタイムで生成され、Amazon S3 に配信されます。
3. Kinesis Data Streams は、変換と配信のためにレコードを Firehose に送信します。
4. Lambda 関数は、レコードを DynamoDB レコード形式から、レコード項目の属性名と値のみを含む JSON 形式に変換します。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウドインフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK Toolkit は、 AWS CDK アプリの操作に役立つコマンドラインクラウド開発キットです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体の管理に役立ちます AWS リージョン。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-dynamodb-kinesisfirehose-s3-Ingestion」リポジトリで利用できます。

エピック

サンプルコードのセットアップおよび設定

タスク	説明	必要なスキル

SDK の依存関係をインストールします。	ローカルマシンで、次のコマンドを実行して、package.json 、pattern/aws-dynamodb-kinesisstreams-s3 、および sample-application ディレクトリのファイルから依存関係をインストールします。
	`cd <project_root>/pattern/aws-dynamodb-kinesisstreams-s3`
	`npm install && npm run build`
	`cd <project_root>/sample-application/`
	`npm install && npm run build`
		アプリ開発者、AWS 全般

CloudFormation テンプレートを生成します。	1. cd <project_root>/sample-appli	アプリケーション開発者、一般的な AWS、AWS DevOps

タスク	説明	必要なスキル
	cation/ コマンドを実行します。 2. cdk synth コマンドを実行して CloudFormation テンプレートを生成します。 3. AwsDynamodbKinesisfirehoseS3IngestionStack.template.json 出力はcdk.out ディレクトリに保存されています。 4. AWS CDK または を使用して AWS Management Console CloudFormation でテンプレートを処理します。	

リソースのデプロイ

タスク	説明	必要なスキル
リソースを確認してデプロイしてください。	1. cdk diff コマンドを実行して、 AWS CDK コンストラクトによって作成されるリソースタイプを識別します。 2. リソースをデプロイするには、cdk deploy コマンドを実行します。	アプリケーション開発者、一般的な AWS、AWS DevOps

DynamoDB テーブルにデータを取り込み、ソリューションをテストします。

タスク	説明	必要なスキル

DynamoDB テーブルにサンプルデータを取り込みます。	次のコマンドを実行して、DynamoDB テーブルにリクエストを送信します AWS CLI。
	
	`aws dynamodb put-item --table-name <your_table_name> --item '{"<table_partition_key>": {"S": "<partition_key_ID>"},"MessageData":{"S": "<data>"}}'`
	
	例：
	
	`aws dynamodb put-item --table-name SourceData_table --item '{"SourceDataId": {"S": "123"},"MessageData":{"S": "Hello World"}}'`
	
	デフォルトでは、put-item オペレーションが成功しても、は出力として値を返しません。操作が失敗した場合はエラーを返します。データは DynamoDB に保存され、Kinesis Data Streams と Firehose に送信されます。
		アプリ開発者

タスク	説明	必要なスキル
	Note DynamoDB テーブルにデータを追加するには、さまざまなアプローチを使用します。詳細については、DynamoDB ドキュメントの「テーブルにデータをロードする」を参照してください。	
S3 バケットに新しいオブジェクトが作成されたことを確認します。	にサインイン AWS Management Console し、S3 バケットをモニタリングして、送信したデータで新しいオブジェクトが作成されていることを確認します。 詳細については、Amazon S3 ドキュメントのGetObject」を参照してください。	アプリ開発者、AWS 全般

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースをクリーンアップします。	`cdk destroy` コマンドを実行して、このパターンで使用されているリソースをすべて削除します。	アプリ開発者、AWS 全般

関連リソース

- s3-static-site-stack.ts (GitHub リポジトリ)
- aws-apigateway-dynamodb モジュール (GitHub リポジトリ)
- aws-kinesisstreams-kinesisfirehose-s3 モジュール (GitHub リポジトリ)
- DynamoDB Streams の変更データキャプチャ (DynamoDB ドキュメント)
- Kinesis Data Streams を使用して DynamoDB への変更をキャプチャする (DynamoDB ドキュメント)

Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する

作成者: Corey Schnedl (AWS)、Anbazhagan Ponnuswamy (AWS)、Marcelo Barbosa (AWS)、Gaurav Samudra (AWS)、Mario Lopez Martinez (AWS)、Abhilash Vinod (AWS)

概要

このパターンは、カスタムドメインの API マッピング機能を使用して、Amazon API Gateway のパスベースの API バージョニングソリューションを実装する方法を示しています。

Amazon API Gateway は、あらゆる規模で APIs を作成、公開、保守、モニタリング、保護するために使用できるフルマネージドサービスです。サービスのカスタムドメイン機能を使用すると、API ユーザーに提供できるより直感的な URLs を使用して、よりシンプルなカスタムドメイン名を作成できます。API マッピングを使用して、API ステージをカスタムドメイン名に接続できます。ドメイン名を作成し、DNS レコードを設定したら、API マッピングを使用して、カスタムドメイン名を使用して API にトラフィックを送信します。

API が公開されると、コンシューマーはその API を使用します。パブリック API が進化するにつれて、そのサービス契約も新機能を反映するように進化します。ただし、既存の機能を変更または削除することは賢明ではありません。重大な変更は、コンシューマーのアプリケーションに影響を与え、実行時に破損する可能性があります。API バージョニングは、下位互換性を破ったり、契約を破ったりしないようにするために重要です。

コンシューマーが API バージョニングを採用できるように、明確な戦略が必要です。パスベースの URLs を使用した APIs のバージョニングは、最も簡単で一般的に使用されるアプローチです。このタイプのバージョニングでは、バージョンは API URIs の一部として明示的に定義されます。次の URLs の例は、コンシューマーが URI を使用してリクエストの API バージョンを指定する方法を示しています。

```
https://api.example.com/api/v1/orders
```

```
https://api.example.com/api/v2/orders
```

```
https://api.example.com/api/vX/orders
```

このパターンでは、 を使用して API 用のスケーラブルなパスベースのバージョニングソリューションのサンプル実装 AWS Cloud Development Kit (AWS CDK) を構築、デプロイ、テストします。AWS CDK は、使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースをモデル化およびプロビジョニングするオープンソースのソフトウェア開発フレームワークです。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- このパターンのサンプルリポジトリを使用し、Amazon API Gateway カスタムドメイン機能を使用するには、ドメインの所有権が必要です。Amazon Route 53 を使用して、組織のドメインを作成および管理できます。Route 53 でドメインを登録または移管する方法については、Route 53 ドキュメントの「新しいドメインの登録」を参照してください。
- API のカスタムドメイン名を設定する前に、SSL/TLS 証明書を準備する必要があります AWS Certificate Manager。
- API エンドポイントにマッピングするために DNS プロバイダーのリソースレコードを作成または更新する必要があります。このようなマッピングがない場合、カスタムドメイン名にバインドされた API リクエストは API Gateway に到達できません。

機能制限

- カスタムドメイン名はプライベート API ではサポートされていません。
- カスタムドメイン名は、すべての AWS リージョン で 内で一意である必要があります AWS アカウント。
- 複数のレベルで API マッピングを設定するには、リージョン別カスタムドメイン名と TLS 1.2 セキュリティポリシーを使用する必要があります。
- API マッピングでは、カスタムドメイン名とマッピングされた APIsが同じである必要があります AWS アカウント。
- API マッピングには、文字、数字、および次の文字のみを含める必要があります。 `$-_.+!*'()/`
- API マッピングのパスの最大文字数は 300 文字です。
- ドメイン名ごとに、複数のレベルで 200 個の API マッピングを設定できます。
- TLS 1.2 セキュリティポリシーでは、HTTP APIs をリージョン別カスタムドメイン名にのみマッピングできます。
- WebSocket API を HTTP API または REST API と同じカスタムドメイン名にマッピングすることはできません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- このサンプル実装では、AWS CDK TypeScript バージョン 2.149.0 で を使用します。

アーキテクチャ

次の図は、アーキテクチャワークフローを示しています。

この図表は、以下を示すものです：

1. API ユーザーは、カスタムドメイン名を使用して Amazon API Gateway にリクエストを送信します。
2. API Gateway は、リクエストの URL に示されているパスに基づいて、ユーザーのリクエストを API Gateway の適切なインスタンスとステージに動的にルーティングします。次の表は、異なる URL ベースのパスを API Gateway の異なるインスタンスの特定のステージにルーティングする方法の例を示しています。

API	ステージ	パス	デフォルトエンドポイント
CalculationAPIv1	api	apiv1	有効
CalculationAPIv2	api	apiv2	有効
CalculationAPIvX	api	apivX	有効

3. 送信先 API Gateway インスタンスはリクエストを処理し、結果をユーザーに返します。

自動化とスケール

API のバージョンごとに個別の AWS CloudFormation スタックを使用することをお勧めします。この方法では、カスタムドメイン APIs マッピング機能によって にルーティングできるバックエンド API を完全に分離できます。このアプローチの利点は、別の API を変更するリスクを発生させることなく、API の異なるバージョンを個別にデプロイまたは削除できることです。このアプローチは、CloudFormation スタックを分離することで耐障害性を向上させます。また、HTTP エンドポイント AWS Lambda、 アクションなど、API AWS Fargateのさまざまなバックエンドオプションも提供します AWS のサービス。

Gitflow などの Git 分岐戦略を分離された CloudFormation スタックと組み合わせて使用して、異なるバージョンの API にデプロイされるソースコードを管理できます。このオプションを使用すると、新しいバージョンのソースコードを複製することなく、API の異なるバージョンを維持できます。Gitflow を使用すると、リリースの実行時に git リポジトリ内のコミットにタグを追加できます。その結果、特定のリリースに関連するソースコードの完全なスナップショットが作成されます。更新を実行する必要がある場合は、特定のリリースのコードをチェックアウトし、更新を行い、対応するメジャーバージョンと一致する CloudFormation スタックに更新されたソースコードをデプロイできます。このアプローチにより、API の各バージョンには分離されたソースコードがあり、個別の CloudFormation スタックにデプロイされるため、別の API バージョンを破損するリスクが軽減されます。

ツール

AWS のサービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS Certificate Manager (ACM) は、 AWS ウェブサイトとアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書とキーの作成、保存、更新に役立ちます。
- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それを通じてプロビジョニングするためのオープンソースのソフトウェア開発フレームワークです AWS CloudFormation。このパターンのサンプル実装では、AWS CDK TypeScript で を使用します。TypeScript AWS CDK で を操作すると、Microsoft TypeScript コンパイラ (`tsc`) 、Node.js、ノードパッケージマネージャー () など、使い慣れたツールが使用されますnpm。必要に応じて Yarn を使用できますが、このパターンの例は を使用しますnpm。AWS コンストラクトライブラリを構成するモジュールは、npm リポジトリ npmjs.org://www.
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- AWS WAF は、保護されたウェブアプリケーションリソースに転送される HTTP および HTTPS リクエストをモニタリングするのに役立つウェブアプリケーションファイアウォールです。

その他のツール

- Bruno は、オープンソースの git フレンドリーな API テストクライアントです。
- cdk-nag は、ルールパックを使用して AWS CDK アプリケーションのベストプラクティスをチェックするオープンソースユーティリティです。

コードリポジトリ

このパターンのコードは、GitHub pathpath-based-versioning-with-api-gatewayリポジトリにあります。

ベストプラクティス

- で構築された CloudFormation スタックのテストとデプロイを自動化するには、堅牢な継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを使用します AWS CDK。この推奨事項の詳細については、AWS 「 Well-Architected DevOps Guidance」を参照してください。
- AWS WAF は、Amazon API Gateway などのサービスと簡単に統合できるマネージドファイアウォールです。 AWS WAF は、このバージョニングパターンが機能するために必要なコンポーネントではありませんが、 API Gateway AWS WAF に を含めることをセキュリティのベストプラクティスとしてお勧めします。
- API コンシューマーが定期的に最新バージョンの API にアップグレードして、古いバージョンの API を効率的に廃止および削除できるようにすることをお勧めします。
- このアプローチを本番環境で使用する前に、API のファイアウォールと認可戦略を実装してください。
- 最小特権アクセスモデル AWS アカウント を使用して、 の AWS リソース管理へのアクセスを実装します。
- で構築されたアプリケーションのベストプラクティスとセキュリティに関する推奨事項を適用するには AWS CDK、cdk-nag ユーティリティを使用することをお勧めします。

エピック

ローカル環境を準備する

タスク	説明	必要なスキル
リポジトリをクローン作成します。	サンプルアプリケーションリポジトリのクローンを作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/path-based-versioning-with-api-gateway`	アプリ開発者
クローンされたリポジトリに移動します。	クローンされたリポジトリフォルダの場所に移動するには、次のコマンドを実行します。 `cd api-gateway-custom-domain-versioning`	アプリ開発者
必要な依存ファイルをインストールします。	必要な依存関係をインストールするには、次のコマンドを実行します。 `npm install`	アプリ開発者

CloudFormation ルーティングスタックをデプロイする

タスク	説明	必要なスキル
ルーティングスタックのデプロイを開始します。	CloudFormation ルーティングスタック のデプロイを	アプリ開発者

タスク	説明	必要なスキル
	開始するにはCustomDomainRouterStack 、次のコマンドを実行し、 を所有しているドメインの名前example.com に置き換えます。 `npx cdk deploy CustomDomainRouterStack --parameters PrerequisiteDomainName=example.com` **Note** 次のドメイン DNS 検証タスクが正常に実行されるまで、スタックのデプロイは成功しません。	

ドメインの所有権を検証する

タスク	説明	必要なスキル
ドメインの所有権を確認します。	証明書は、関連付けられたドメインの所有権を証明するまで、検証保留中のステータスのままになります。 所有権を証明するには、ドメインに関連付けられているホストゾーンに CNAME レコードを追加します。詳細については、 AWS Certifica	アプリ開発者、AWS システム管理者、ネットワーク管理者

タスク	説明	必要なスキル
	te Manager ドキュメントの「DNS 検証」を参照してください。 適切なレコードを追加すると、`CustomDomainRouterStack` デプロイが成功します。	

タスク	説明	必要なスキル
API Gateway カスタムドメインを指すエイリアスレコードを作成します。	証明書が正常に発行および検証されたら、Amazon API Gateway カスタムドメイン URL を指す DNS レコードを作成します。 カスタムドメイン URL は、カスタムドメインのプロビジョニングによって一意に生成され、CloudFormation 出力パラメータとして指定されます。レコードの例を次に示します。 ルーティングポリシー: シンプルなルーティング レコード名: `demo.api-gateway-custom-domain-versioning.example.com` Alias (エイリアス): あり レコードタイプ: AWS リソースを指す「A」タイプの DNS レコード 値: `d-xxxxxxxxxx.execute-api.xx-xxxx-x.amazonaws.com` TTL (秒) : 300	アプリ開発者、AWS システム管理者、ネットワーク管理者

CloudFormation スタックをデプロイして API を呼び出す

タスク	説明	必要なスキル

`ApiStackV1` スタックをデプロイします。	`ApiStackV1` スタックをデプロイするには、次のコマンドを使用します。
	`npm run deploy-v1`
	次の CDK コードは API マッピングを追加します。
	`var apiMapping = new CfnApiMapping(this, "ApiMapping", { apiId: this.lambdaRestApi.restApiId, domainName: props.customDomainName.domainName, stage: "api", apiMappingKey: "api/v1", });`
		アプリ開発者

`ApiStackV2` スタックをデプロイします。	`ApiStackV2` スタックをデプロイするには、次のコマンドを使用します。 `npm run deploy-v2`	アプリ開発者
API を呼び出します。	Bruno を使用して API を呼び出し、API エンドポイントをテストするには、「追加情報」の手順を参照してください。	アプリ開発者

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースをクリーンアップします。	このサンプルアプリケーションに関連付けられているリソースを破棄するには、次のコマンドを使用します。 `npx cdk destroy --all` **Note** ドメイン所有権の検証プロセスのために手動で追加された Route 53 DNS レコードを必ずクリーンアップしてください。	アプリ開発者

トラブルシューティング

問題	ソリューション
証明書を検証できないため、 のデプロイはCustomDomainRouterStack　タイムアウトします。	前のタスクで説明したように、適切な DNS 検証 CNAME レコードを追加したことを確認してください。新しい証明書は、DNS 検証レコードを追加した後、最大 30 分間、検証保留中のステータスを引き続き表示する場合があります。詳細については、 AWS Certificate Manager ドキュメントの「DNS 検証」を参照してください。

関連リソース

- Amazon CloudFront を使用したヘッダーベースの API Gateway バージョニングの実装 – この AWS Compute Blog の投稿では、このパターンで説明されているパスベースのバージョニング戦略の代替として、ヘッダーベースのバージョニング戦略を提供しています。
- AWS CDK ワークショップ – この入門ワークショップでは、 を使用して でアプリケーションを構築およびデプロイ AWS することに重点を置いています AWS Cloud Development Kit (AWS CDK)。このワークショップでは、Go、Python、TypeScript をサポートしています。

追加情報

Bruno を使用した API のテスト

オープンソースの API テストツールである Bruno を使用して、パスベースのルーティングがサンプルアプリケーションに対して正しく動作していることを確認することをお勧めします。このパターンは、サンプル API のテストを容易にするサンプルコレクションを提供します。

API を呼び出してテストするには、次のステップを使用します。

1. Bruno をインストールします。
2. Bruno を開きます。
3. このパターンのコードリポジトリで、Bruno/Sample-API-Gateway-Custom-Domain-Versioning を選択し、コレクションを開きます。
4. ユーザーインターフェイス (UI) の右上にある環境ドロップダウンを表示するには、コレクション内の任意のリクエストを選択します。
5. Environments ドロップダウンで、Configure を選択します。
6. `REPLACE_ME_WITH_YOUR_DOMAIN` 値をカスタムドメインに置き換えます。
7. 保存を選択し、設定セクションを閉じます。
8. サンドボックス環境では、アクティブオプションが選択され ていることを確認します。
9. 選択したリクエストの -> ボタンを使用して API を呼び出します。
10. V2 と比較して V1 で検証 (数値以外の値を渡す) V2がどのように処理されるかに注意してください。

API 呼び出しの例と V1 と V2 の検証の比較のスクリーンショットを確認するには、このパターンのコードリポジトリの `README.md` ファイルでサンプル API をテストするを参照してください。

psycopg2 ライブラリを にインポート AWS Lambda して PostgreSQL データベースとやり取りする

作成者: Louis Hourcade (AWS)

概要

Psycopg は Python 用の PostgresSQL データベースアダプターです。開発者は、 psycopg2ライブラリを使用して、PostgreSQL データベースとやり取りする Python アプリケーションを記述します。

アマゾン ウェブ サービス (AWS) では、デベロッパーは AWS Lambda を使用してアプリケーションまたはバックエンドサービスのコードを実行します。Lambda は、サーバーレスのイベント駆動型コンピューティングサービスであり、サーバーのプロビジョニングや管理を必要とせずにコードを実行します。

デフォルトでは、Python ランタイム (バージョン 3.9、3.8、または 3.7) を使用する新しい関数を作成すると、 が提供する Lambda のベースイメージから Lambda ランタイム環境が作成されます AWS。pandas や などのライブラリはpsycopg2、ベースイメージに含まれません。ライブラリを使用するには、カスタムパッケージにバンドルして Lambda にアタッチする必要があります。

ライブラリをバンドルおよびアタッチするには、次のような複数の方法があります。

- .zip ファイルアーカイブから Lambda 関数をデプロイします。
- カスタムコンテナイメージから Lambda 関数をデプロイします。
- Lambda レイヤーを作成し、Lambda 関数にアタッチします。

このパターンは、最初の 2 つのオプションを示しています。

.zip デプロイパッケージでは、pandasライブラリを Lambda 関数に追加するのは比較的簡単です。Linux マシンにフォルダを作成し、Lambda スクリプトをpandasライブラリとライブラリの依存関係とともにフォルダに追加し、フォルダを圧縮して、Lambda 関数のソースとして提供します。

.zip デプロイパッケージを使用することは一般的な方法ですが、そのアプローチはpsycopg2ライブラリでは機能しません。このパターンは、最初に .zip デプロイパッケージを使用してpsycopg2ライブラリを Lambda 関数に追加した場合に発生するエラーを示します。このパターンは、Dockerfile から Lambda をデプロイし、Lambda イメージを編集してpsycopg2ライブラリを機能させる方法を示しています。

パターンがデプロイする 3 つのリソースの詳細については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- このパターンで使用される AWS リソースをデプロイするための十分なアクセス許可 AWS アカウント を持つアクティブな
- AWS Cloud Development Kit (AWS CDK) を実行してグローバルにインストールされる `npm install -g aws-cdk`
- Git クライアント
- Python
- Docker

機能制限

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

製品バージョン

- AWS Lambda ランタイムバージョン: Python 3.8 (パターンは他の Python バージョンに合わせて調整できます) 。
- Psycopg2 バージョン 2.9.3
- Pandas バージョン 1.5.2

アーキテクチャ

ソリューションの概要

Lambda でpsycopg2ライブラリを使用する際に直面する可能性のある課題を説明するために、パターンは 2 つの Lambda 関数をデプロイします。

- .zip ファイルから作成された Python 3.8 ランタイムを持つ 1 つの Lambda 関数。psycopg2 および pandasライブラリは、pip を使用してこの .zip デプロイパッケージにインストールされます。
- Dockerfile から作成された Python 3.8 ランタイムを持つ 1 つの Lambda 関数。Dockerfile は、Lambda コンテナイメージに psycopg2および pandasライブラリをインストールします。

最初の Lambda 関数はpandas、ライブラリとその依存関係を .zip ファイルにインストールし、Lambda はそのライブラリを使用できます。

2 番目の Lambda 関数は、Lambda 関数のコンテナイメージを構築することで、Lambda で pandasおよび psycopg2ライブラリを実行できることを示しています。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウドインフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- pandas は、データ分析と操作のための Python ベースのオープンソースツールです。
- Psycopg は、マルチスレッドアプリケーション用に設計された Python 言語用の PostgreSQL データベースアダプターです。このパターンでは Psycopg 2 を使用します。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub の import-psycopg2-in-lambda-to-interact-with-postgres-database リポジトリで使用できます。

ベストプラクティス

このパターンでは、 AWS CDK を使用して Dockerfile から Lambda 関数を作成する作業例を示します。このコードをアプリケーションで再利用する場合は、デプロイされたリソースがすべてのセキュリティ要件を満たしていることを確認してください。インフラストラクチャがデプロイされる前に、クラウドインフラストラクチャ設定をスキャンして設定ミスを見つける Checkov などのツールを使用します。

エピック

リポジトリのクローンを作成し、デプロイを設定する

タスク	説明	必要なスキル

リポジトリをクローン作成します。	ローカルマシンで GitHub リポジトリのクローンを作成するには、次のコマンドを実行します。
	`git clone https://github.com/aws-samples/import-psycopg2-in-lambda-to-interact-with-postgres-database.git`
	`cd AWS-lambda-psycopg2`
		AWS 全般

デプロイを設定します。	次の情報を使用して app.py ファイルを編集します AWS アカウント。
	`aws_acccount = "AWS_ACCOUNT_ID"`
	`region = "AWS_REGION"`
	`# Select the CPU architecture you are using to build the image (ARM or X86)`
	`architecture = "ARM"`
		AWS 全般

AWS アカウントをブートストラップしてアプリケーションをデプロイする

タスク	説明	必要なスキル
をブートストラップします AWS アカウント。	AWS 環境をブートストラップしていない場合は、 AWS アカウントの AWS 認証情報を使用して次のコマンドを実行します。 `cdk bootstrap aws://<tooling-account-id>/<aws-region>`	AWS 全般
コードをデプロイします。	AWS CDK アプリケーションをデプロイするには、次のコマンドを実行します。 `cdk deploy AWSLambdaPyscopg2`	AWS 全般

AWS マネジメントコンソールから Lambda 関数をテストする

タスク	説明	必要なスキル
.zip ファイルから作成された Lambda 関数をテストします。	.zip ファイルから作成された Lambda 関数をテストするには、次の手順を実行します。 1. コンソールにサインインし、「https://https://console.aws.amazon.com/lambda/.com で Lambda コンソールを開きます。	AWS 全般

タスク	説明	必要なスキル
	2. `lambda-from-zip` Lambda 関数を選択します。 3. 関数を呼び出すテストイベントを作成します。 4. 呼び出されると、関数は次のメッセージを含むエラーを発生させます。 `"errorMessage": Unable to import module 'lambda_code': libpq.so.5: cannot open shared object, "stackTrace": [] "errorType": Runtime.ImportModuleError",` 5. https://console.aws.amazon.com/cloudwatch/ で Amazon CloudWatch コンソールを開きます。CloudWatch ログには、pandasライブラリが正常にインポートされたが、psycopg2ライブラリのインポートが失敗したことが示されます。 Lambda は、必要な PostgreSQL ライブラリをデフォルトイメージに見つからないため、psycopg2ライブラリを使用できません。	

タスク	説明	必要なスキル

Dockerfile から作成された Lambda 関数をテストします。	Lambda 関数内でpsycopg2ライブラリを使用するには、Lambda Amazon マシンイメージ (AMI) を編集する必要があります。
	
	Dockerfile から作成された Lambda 関数をテストするには、次の手順を実行します。
	
	1. コンソールにサインインし、Lambda コンソールを開きます。
	2. `lambda-from-docker` Lambda 関数を選択します。
	3. 関数を呼び出すテストイベントを作成します。
	4. 呼び出されると、関数は正常に実行されます。
	
	次のコードは、 AWS CDK テンプレートが作成する Dockerfile を示しています。
	
	`# Start from lambda Python3.8 image`
	`FROM public.ecr.aws/lambda/python:3.8`
	
	`# Copy the lambda code, together with its requirements`
	`COPY lambda/requirements.txt ${LAMBDA_TASK_ROOT}`
		AWS 全般

タスク	説明	必要なスキル

	<pre>COPY lambda/lambda_code
	.py ${LAMBDA_TASK_ROOT
	}
	
	# Install postgresql-
	devel in your image
	RUN yum install -y gcc
	 postgresql-devel
	
	# install the requireme
	nts for the Lambda code
	RUN pip3 install -r
	 requirements.txt
	 --target "${LAMBDA
	_TASK_ROOT}"
	
	# Command can be
	 overwritten by
	 providing a different
	 command in the template
	 directly.
	CMD ["lambda_code.hand
	ler"]</pre>
	Dockerfile は Python 3.8 ランタイム用に AWS 提供された Lambda イメージを取得し、postgresql-devel をインストールします。これには、PostgreSQL 管理サーバーと直接やり取りするアプリケーションをコンパイルするために必要なライブラリが含まれています。Dockerfile は、`requirements.txt` ファイルで示されている pandasおよび psycopg2ライブラリもインストールします。	

関連リソース

- AWS CDK ドキュメント
- AWS Lambda ドキュメント

追加情報

このパターンでは、 AWS CDK テンプレートは 3 つのリソースを持つ AWS スタックを提供します。

- Lambda 関数の AWS Identity and Access Management (IAM) ロール。
- Python 3.8 ランタイムを使用する Lambda 関数。関数はデプロイパッケージからConstructs/lambda/lambda_deploy.zipデプロイされます。
- Python 3.8 ランタイムを使用する Lambda 関数。関数は Constructsフォルダの Dockerfile からデプロイされます。

Lambda 関数の両方のスクリプトは、 ライブラリpandasと psycopg2ライブラリが正常にインポートされたかどうかを確認します。

```
import pandas
print("pandas successfully imported")

import psycopg2
print("psycopg2 successfully imported")

def handler(event, context):
    """Function that checks whether psycopg2  and pandas are successfully imported or
 not"""
    return {"Status": "psycopg2 and pandas successfully imported"}
```

lambda_deploy.zip デプロイパッケージは bash Constructs/lambda/build.sh スクリプトで構築されます。このスクリプトは、フォルダを作成し、Lambda スクリプトをコピーして、pandas および psycopg2ライブラリをインストールし、.zip ファイルを生成します。.zip ファイルを自分で生成するには、この bash スクリプトを実行して AWS CDK スタックを再デプロイします。

Dockerfile は、Python 3.8 ランタイムを使用する Lambda 用に AWS 提供されたベースイメージで始まります。Dockerfile は、デフォルトイメージの上に pandas および psycopg2ライブラリをインストールします。

このパターンは、Dockerfile から関数を作成し、必要な依存関係を Lambda イメージに追加することで、Lambda でpsycopg2ライブラリを使用する 1 つの方法を示しています。これを実現するその他の方法については、GitHub awslambda-psycopg2 リポジトリを参照してください。

Amazon API Gateway を Amazon SQS と統合して非同期 REST APIs を処理する

作成者: Natalia Colantonio Favero (AWS) と Gustavo Martim (AWS)

概要

REST APIsデプロイする場合、クライアントアプリケーションが発行できるメッセージキューを公開する必要がある場合があります。例えば、サードパーティー APIs のレイテンシーやレスポンスの遅延に問題がある場合や、データベースクエリの応答時間を回避したり、多数の同時 APIs がある場合にサーバーをスケーリングしたりすることを避けたい場合があります。これらのシナリオでは、キューに発行するクライアントアプリケーションは、API がデータを受信したことのみを知るだけで済みます。データの受信後に何が起こるかを知る必要はありません。

このパターンは、Amazon API Gateway を使用して Amazon Simple Queue Service (Amazon SQS) にメッセージを送信することで REST API エンドポイントを作成します。 Amazon SQS これにより、SQS キューへの直接アクセスを回避する 2 つのサービス間にeasy-to-implement統合が作成されます。

前提条件と制限

- アクティブな AWS アカウント

アーキテクチャ

この図は、これらのステップを示しています。

1. Postman、別の API、またはその他のテクノロジーなどのツールを使用して、POST REST API エンドポイントをリクエストします。
2. API Gateway は、リクエストの本文で受信されるメッセージをキューに投稿します。
3. Amazon SQS はメッセージを受信し、成功または失敗コードを含む回答を API Gateway に送信します。

ツール

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- 「Amazon Simple Queue Service (Amazon SQS)」は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。

エピック

SQS キューを作成する

タスク	説明	必要なスキル
キューを作成する。	REST API からメッセージを受信する SQS キューを作成するには： 1. AWS アカウントにサインインします。 2. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。 3. [キューの作成]を選択します。 4. キューの作成ページで、リージョンのドロップダウンリストから正しい AWS リージョン ものを選択します。 5. Type の場合、デフォルト設定 (標準) のままにします。 6. キュー名を入力します。	アプリ開発者

タスク	説明	必要なスキル
	7. 他のすべての設定のデフォルト値はそのままにしておきます。 8. [キューの作成]を選択します。	

Amazon SQS へのアクセスを提供する

タスク	説明	必要なスキル
IAM ロールを作成します。	この IAM ロールは、API Gateway リソースに Amazon SQS へのフルアクセスを許可します。 1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。 2. ナビゲーションペインで [ロール]、[ロールの作成] の順に選択します。 3. 信頼できるエンティティタイプ で、AWS のサービスを選択します。 4. ユースケースでは、ドロップダウンリストから API Gateway を選択し、次へ、次へを選択します。 5. ロール名に、AWSGatewayRoleForSQS とオプションの説明を入力し、ロールの作成を選択します。	アプリ開発者、AWS 管理者

タスク	説明	必要なスキル
	6. ロールペインで、AWSGatewayRoleForSQS を検索し、そのチェックボックスをオンにします。 7. [アクセス許可ポリシー] で、[アクセス許可を追加] と [ポリシーをアタッチ] を続けて選択します。 8. AmazonSQSFullAccess を検索して選択します。 9. [Add permissions (許可の追加)] を選択します。 10 AWSGatewayRoleForSQS の概要セクションで、Amazon リソース番号 (ARN) をコピーします。この ID は後のステップで使用します。	

REST API を作成する

タスク	説明	必要なスキル
REST API を作成します。	これは、HTTP リクエストが送信される REST API です。 1. API Gateway コンソール (「https://console.aws.amazon.com/apigateway」) を開きます。 2. REST API セクションで、ビルドを選択します。	アプリ開発者

タスク	説明	必要なスキル
	3. API 名には、API の名前とオプションの説明を入力し、他のすべてのデフォルト設定のままにして、API の作成を選択します。	

タスク	説明	必要なスキル
API Gateway を Amazon SQS に接続します。	このステップにより、HTTP リクエストの本文内から Amazon SQS にメッセージが流れるようになります。 1. API Gateway コンソールで、作成した API を選択します。 2. リソースページの「メソッド」セクションで、「メソッドの作成」を選択します。 3. [メソッドタイプ] で、[POST] を選択します。 4. 統合タイプで、 を選択しますAWS のサービス。 5. でAWS リージョン、SQS キューを作成したリージョンを選択します。 6. でAWS のサービス、Simple Queue Service (SQS) を選択します。 7. HTTP メソッドでは、POST を選択します。 8. アクションタイプ で、パスオーバーライドの使用 を選択します。 9. パスオーバーライドには、<AWS アカウント ID>/<SQS キューの名前> を入力します。	アプリ開発者

タスク	説明	必要なスキル
	10実行ロールには、前に作成したロールの ARN を貼り付けます。 11[メソッドの作成] を選択します。	

REST API をテストする

タスク	説明	必要なスキル
REST API をテストします。	テストを実行して、設定が欠落していないことを確認します。 1. API Gateway コンソールで、作成した REST API を選択します。 2. リソースペインで、POST メソッドを選択します。 3. [テスト] タブを選択します。(タブが表示されない場合は右矢印を使用します)。 4. リクエスト本文には、次の JSON コードを貼り付けます。 `{ "message": "lorem ipsum" }` 5. [テスト] を選択します。	アプリ開発者

タスク	説明	必要なスキル
	次のような エラーが表示されます。 `<UnknownOperationException/>`	

タスク	説明	必要なスキル
API 統合を変更して、リクエストを Amazon SQS に適切に転送します。	設定を完了して、統合エラーを修正します。 1. API Gateway コンソールで、作成した API を選択し、POST を選択します。 2. メソッド実行セクションには、API Gateway と Amazon SQS 間のビジュアルマッピングが表示されます。このセクションでは、統合リクエストを選択し、編集を選択します。 3. HTTP ヘッダーセクションを展開し、リクエストヘッダーの追加パラメータを選択します。 • Name に Content-Type を指定します。 • マッピング元の には、「application/x-www-form-urlencoded」と入力します。必ず一重引用符を含めてください。 • キャッシュチェックボックスをオンにします。 4. マッピングテンプレートセクションを展開します。 • [マッピングテンプレートの追加] を選択します。 • コンテンツタイプに application/json と入力します。	アプリ開発者

タスク	説明	必要なスキル
	• テンプレート本文には、次のコードを貼り付けます。 `Action=SendMessage&MessageBody=$input.body` • [Save] を選択します。	

タスク	説明	必要なスキル
Amazon SQS でメッセージをテストおよび検証します。	テストを実行して、テストが正常に完了したことを確認します。 1. API Gateway コンソールで、作成した REST API を選択します。 2. リソースペインで、POST メソッドを選択します。 3. [テスト] タブを選択します。(タブが表示されない場合は右矢印を使用します)。 4. リクエスト本文には、次の JSON コードを貼り付けます。 `{ "message": "lorem ipsum" }` 5. [テスト] を選択します。 6. Amazon SQS コンソール を開きます。 7. ナビゲーションペインで、キューを選択し、キューを選択します。 8. [メッセージの送信と受信] を選択します。 9. メッセージをポーリングを選択します。 10[メッセージ] を選択します。以下が表示されます。	アプリ開発者

タスク	説明	必要なスキル
	`Body { "message": "lorem ipsum" }`	

タスク	説明	必要なスキル

特殊文字を使用して API Gateway をテストします。	メッセージで許容されない特殊文字 (& など) を含むテストを実行します。
	
	1. API Gateway コンソールで、API を選択します。
	2. 次の JSON コードを使用して、前のステップのテストを繰り返します。
	
	`{ "message": "lorem ipsum &" }`
	
	3. [テスト] を選択します。
	
	次のようなエラーが表示されます。
	
	`{ "Error": { "Code": "AccessDenied", "Message": "Access to the resource https://sqs.us-east-2.amazonaws.com/976166761794/Apg2 is denied.", "Type": "Sender" }, "RequestId": "e83c9c67-bcf6-5e9a-91e9-c737094b17ab" }`
		アプリ開発者

タスク	説明	必要なスキル
	これは、メッセージ本文で特殊文字がデフォルトでサポートされていないためです。次のステップでは、特殊文字をサポートするように API Gateway を設定します。コンテンツタイプの変換の詳細については、API Gateway のドキュメントを参照してください。	

タスク	説明	必要なスキル

特殊文字をサポートするように API 設定を変更します。	メッセージで特殊文字を受け入れるように設定を調整します。
	
	1. API Gateway コンソールで、作成した API を選択し、POST を選択します。
	2. [統合リクエスト]、[編集] の順に選択します。
	3. コンテンツ処理を変更してテキストに変換します。
	4. 「マッピングテンプレート」セクションで、次の操作を行います。
	• コンテンツタイプに application/json と入力します。
	• テンプレート本文には、以下を指定します。
	`Action=SendMessage&MessageBody=$util.urlEncode($input.body)`
	• [Save] を選択します。
	5. [テスト] タブを選択します。
	6. リクエスト本文には、前のの JSON コードを入力します。
	`{`
		アプリ開発者

タスク	説明	必要なスキル
	<pre>" message": "lorem ipsum &" }</pre> 7. [テスト] を選択します。 8. Amazon SQS コンソール を開きます。 9. キューを選択し、メッセージの送受信、メッセージのポーリング、メッセージを選択します。 新しいメッセージには特殊文字が含まれている必要があります。	

REST API をデプロイする

タスク	説明	必要なスキル
API をデプロイします。	REST API をデプロイするには： 1. API Gateway コンソールを開きます。 2. API を選択します。 3. [API のデプロイ] を選択します。このステップの詳細については、API Gateway のドキュメントを参照してください。	アプリ開発者
外部ツールを使用してテストします。	外部ツールを使用してテストを実行して、メッセージが正	アプリ開発者

タスク	説明	必要なスキル
	常に受信されたことを確認します。 1. Postman、Insomnia、cURL などのツールを開きます。 2. API を実行します。 3. Amazon SQS コンソール を開きます。 4. キューを選択します。 5. メッセージをロードして新しいメッセージを表示します。	

クリーンアップ

タスク	説明	必要なスキル
API を削除します。	API Gateway コンソールで、作成した API を選択し、削除を選択します。	アプリ開発者
IAM ロールを削除します。	IAM コンソールのロールペインで、AWSGatewayRoleForSQS を選択し、削除を選択します。	アプリ開発者
SQS キューを削除します。	Amazon SQS コンソールのキューペインで、作成した SQS キューを選択し、削除を選択します。	アプリ開発者

関連リソース

- SQS-SendMessage (API Gateway ドキュメント)

- API Gateway でのコンテンツタイプの変換 (API Gateway ドキュメント)
- $util 変数 (API Gateway ドキュメント)
- API Gateway REST API を Amazon SQS と統合し、一般的なエラーを解決する方法 (AWS re:Post 記事)

Amazon API Gateway と AWS Lambda を使用してイベントを非同期的に処理する

作成者: Andrea Meroni (AWS)、Nadim Majed (AWS)、Marime Kthiri (AWS)、Michael Wallner (AWS)

概要

Amazon API Gateway は、開発者があらゆる規模で APIs を作成、公開、保守、モニタリング、保護するために使用できるフルマネージドサービスです。最大数十万の同時 API コールの受け入れと処理に関連するタスクを処理します。

API Gateway の重要なサービスクォータは、統合タイムアウトです。タイムアウトは、REST API がエラーを返す前にバックエンドサービスがレスポンスを返す必要がある最大時間です。29 秒のハード制限は、同期ワークロードで一般的に許容されます。ただし、この制限は、非同期ワークロードで API Gateway を使用するデベロッパーにとって課題となります。

このパターンは、API Gateway と を使用してイベントを非同期的に処理するアーキテクチャの例を示しています AWS Lambda。このアーキテクチャは、最大 15 分間の処理ジョブの実行をサポートし、インターフェイスとして基本的な REST API を使用します。

Projen は、ローカル開発環境をセットアップし、AWS Cloud Development Kit (AWS CDK) Toolkit AWS アカウント、Docker、Node.js と組み合わせてサンプルアーキテクチャをターゲットにデプロイするために使用されます。Projen は、事前コミットと、コードの品質保証、セキュリティスキャン、ユニットテストに使用されるツールを使用して、Python 仮想環境を自動的にセットアップします。詳細については、「ツール」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ワークステーションにインストールされている以下のツール：
  - AWS Cloud Development Kit (AWS CDK) ツールキットバージョン 2.85.0
  - Docker バージョン 20.10.21
  - Node.js バージョン 18.13.0
  - Projen バージョン 0.71.111
  - Python バージョン 3.9.16

機能制限

- ジョブの最大ランタイムは、Lambda 関数の最大ランタイム (15 分) によって制限されます。
- 同時ジョブリクエストの最大数は、Lambda 関数の予約済み同時実行数によって制限されます。

アーキテクチャ

次の図は、Amazon EventBridge イベントアーカイブに保存されたイベントと、イベント処理およびエラー処理 Lambda 関数とのジョブ API の相互作用を示しています。

一般的なワークフローには、次のステップが含まれます。

1. AWS Identity and Access Management (IAM) に対して認証し、セキュリティ認証情報を取得します。
2. HTTP `POST`リクエストを`/jobs`ジョブ API エンドポイントに送信し、リクエスト本文でジョブパラメータを指定します。
3. API Gateway REST API であるジョブ API は、ジョブ識別子を含む HTTP レスポンスを返します。
4. ジョブ API は、イベント処理 Lambda 関数を非同期的に呼び出します。
5. イベント処理関数はイベントを処理し、ジョブ結果を Amazon DynamoDB テーブルに配置
6. ステップ 3 の`/jobs/{jobId}`ジョブ識別子を として、HTTP `GET`リクエストをジョブ API エンドポイントに送信します`{jobId}`。
7. ジョブ API は DynamoDB `jobs` テーブルにクエリを実行してジョブ結果を取得します。
8. ジョブ API は、ジョブ結果を含む HTTP レスポンスを返します。
9. イベント処理が失敗した場合、イベント処理関数はイベントをエラー処理関数に送信します。
10. エラー処理関数は、ジョブパラメータを DynamoDB `jobs` テーブルに配置します。
11. ジョブ API エンドポイントに HTTP `GET`リクエストを送信することで、`/jobs/{jobId}`ジョブパラメータを取得できます。
12. エラー処理が失敗した場合、エラー処理関数はイベントを EventBridge イベントアーカイブに送信します。

    EventBridge を使用して、アーカイブされたイベントを再生できます。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドを通じて AWS のサービスを操作するのに役立つオープンソースツールです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを提供します。
- 「Amazon EventBridge」 は、アプリケーションをさまざまなソースのデータに接続するために支援するサーバーレスイベントバスサービスです。例えば、Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

その他のツール

- autopep8 は、Python Enhancement Proposal (PEP) 8 スタイルガイドに基づいて Python コードを自動的にフォーマットします。
- Bandit は Python コードをスキャンして、一般的なセキュリティ問題を見つけます。
- Commitizen は Git コミットチェッカーとCHANGELOGジェネレーターです。
- cfn-lint は linter AWS CloudFormation です
- Checkov は、Infrastructure as Code (IaC) のセキュリティとコンプライアンスの設定ミスをチェックする静的コード分析ツールです。
- jq は JSON を解析するためのコマンドラインツールです。
- Postman は API プラットフォームです。
- 事前コミットは Git フックマネージャーです。
- Projen はプロジェクトジェネレーターです。
- 「pytest」は、小さくて読みやすいテストを書くための Python フレームワークです。

コードリポジトリ

このアーキテクチャコードの例は、GitHub の API Gateway と Lambda による非同期イベント処理リポジトリにあります。

ベストプラクティス

- この例のアーキテクチャには、デプロイされたインフラストラクチャのモニタリングは含まれません。ユースケースでモニタリングが必要な場合は、CDK Monitoring Constructs または別のモニタリングソリューションの追加を評価します。
- このアーキテクチャ例では、IAM アクセス許可を使用してジョブ API へのアクセスを制御します。を引き受ける権限を持つユーザーはJobsAPIInvokeRole、ジョブ API を呼び出すことができます。そのため、アクセスコントロールメカニズムはバイナリです。ユースケースでより複雑な認可モデルが必要な場合は、別のアクセスコントロールメカニズムを使用して を評価します。
- ユーザーが/jobsジョブ API エンドポイントに HTTP POSTリクエストを送信すると、入力データは 2 つの異なるレベルで検証されます。
  - Amazon API Gateway は、最初のリクエスト検証を担当します。
  - イベント処理関数は 2 番目のリクエストを実行します。

    ユーザーが/jobs/{jobId}ジョブ API エンドポイントに HTTP GETリクエストを実行する場合、検証は実行されません。ユースケースで追加の入力検証とセキュリティレベルの向上が必要な場合は、AWS WAF を使用して API を保護します。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリのクローンをローカルに作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-lambda-cdk.git`	DevOps エンジニア

タスク	説明	必要なスキル

プロジェクトをセットアップします。	ディレクトリをリポジトリルートに変更し、Projen を使用して Python 仮想環境とすべてのツールを設定します。
	`cd asynchronous-event-processing-api-gateway-api-gateway-lambda-cdk`
	`npx projen`
		DevOps エンジニア

事前コミットフックをインストールします。	事前コミットフックをインストールするには、次の手順を実行します。
	1. Python 仮想環境をアクティブ化します。
	`source .env/bin/activate`
	2. 事前コミットフックをインストールします。
	`pre-commit install`
	`pre-commit install --hook-type commit-msg`
		DevOps エンジニア

サンプルアーキテクチャをデプロイする

タスク	説明	必要なスキル
ブートストラップ AWS CDK。	AWS CDK でブートストラップするには AWS アカウント、次のコマンドを実行します。	AWS DevOps

タスク	説明	必要なスキル
	`AWS_PROFILE=$YOUR_AWS_PROFILE npx projen bootstrap`	
サンプルアーキテクチャをデプロイします。	にサンプルアーキテクチャをデプロイするには AWS アカウント、次のコマンドを実行します。 `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy`	AWS DevOps

アーキテクチャをテストする

タスク	説明	必要なスキル
テストの前提条件をインストールします。	AWS Command Line Interface（AWS CLI）、Postman、jq をワークステーションにインストールします。 Postman を使用してこのサンプルアーキテクチャをテストすることをお勧めしますが、必須ではありません。代替 API テストツールを選択する場合は、AWS 署名バージョン 4 認証をサポートし、REST API をエクスポートして検査できる公開 API エンドポイントを参照するようにしてください。	DevOps エンジニア

タスク	説明	必要なスキル
を引き受けますJobsAPIInvokeRole 。	deploy コマンドから出力として出力JobsAPIInvokeRole された を仮定します。 CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \ --no-cli-pager \ --role-arn $<JOBS_API_INVOKE_ROLE_ARN> \ --role-session-name JobsAPIInvoke) export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS \| jq '.Credentials''.AccessKeyId') export AWS_SECRET_ACCESS_KEY=$(cat $CREDENTIALS \| jq '.Credentials''.SecretAccessKey') export AWS_SESSION_TOKEN==$(cat $CREDENTIALS | jq '.Credentials''.SessionToken')	AWS DevOps

タスク	説明	必要なスキル
Postman を設定します。	1. リポジトリに含まれる Postman コレクションをインポートするには、Postman ドキュメントの指示に従います。 2. JobsAPI 変数を次の値で設定します。 • accessKey assume-role – コマンドの Credentials.AccessKeyId 属性の値 • baseUrl – 末尾にスラッシュがない、デプロイコマンドからのJobsApiJobsAPIEndpoint 出力の値 • region – サンプルアーキテクチャをデプロイ AWS リージョン した の値 • seconds – サンプルジョブの入力パラメータの値。正の整数である必要があります • secretKey assume-role – コマンドの Credentials.SecretAccessKey 属性の値 • sessionToken assume-role – コマンドの Credentials.SessionToken 属性の値	AWS DevOps

タスク	説明	必要なスキル
サンプルアーキテクチャをテストします。	サンプルアーキテクチャをテストするには、ジョブ API にリクエストを送信します。詳細については、Postman ドキュメントを参照してください。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
Amazon CloudWatch Logs ロググループ `/aws/apigateway/JobsAPIAccessLogs` がすでに存在するため、サンプルアーキテクチャの破壊とその後の再デプロイは失敗します。	1. 必要に応じて、ログデータを Amazon S3 にエクスポートします。 2. CloudWatch Logs ロググループ を削除します `/aws/apigateway/JobsAPIAccessLogs` 。 3. サンプルアーキテクチャを再デプロイします。

関連リソース

- API Gateway マッピングテンプレートとアクセスロギング変数リファレンス
- バックエンド Lambda 関数の非同期呼び出しを設定する

Amazon API Gateway と Amazon DynamoDB Streams を使用してイベントを非同期的に処理する

作成者: Andrea Meroni (AWS)、Alessandro Trisolini (AWS)、Nadim Majed (AWS)、Marime Kthiri (AWS)、Michael Wallner (AWS)

概要

Amazon API Gateway は、開発者があらゆる規模で APIs を作成、公開、保守、モニタリング、保護するために使用できるフルマネージドサービスです。最大数十万の同時 API コールの受け入れと処理に関連するタスクを処理します。

API Gateway の重要なサービスクォータは、統合タイムアウトです。タイムアウトは、REST API がエラーを返す前にバックエンドサービスがレスポンスを返す必要がある最大時間です。29 秒のハード制限は、同期ワークロードで一般的に許容されます。ただし、この制限は、非同期ワークロードで API Gateway を使用するデベロッパーにとって課題となります。

このパターンは、API Gateway、Amazon DynamoDB Streams、および を使用してイベントを非同期的に処理するためのアーキテクチャの例を示しています AWS Lambda。このアーキテクチャは、同じ入力パラメータで並列処理ジョブの実行をサポートし、インターフェイスとして基本的な REST API を使用します。この例では、バックエンドとして Lambda を使用すると、ジョブの期間は 15 分に制限されます。この制限を回避するには、代替サービスを使用して受信イベント (など AWS Fargate) を処理します。

Projen は、ローカル開発環境をセットアップし AWS アカウント、AWS Cloud Development Kit (AWS CDK) Toolkit、Docker、Node.js と組み合わせてサンプルアーキテクチャをターゲットにデプロイするために使用されます。Projen は、事前コミットと、コードの品質保証、セキュリティスキャン、ユニットテストに使用されるツールを使用して、Python 仮想環境を自動的にセットアップします。詳細については、「 ツール」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ワークステーションにインストールされている以下のツール :
  - AWS Cloud Development Kit (AWS CDK) ツールキットバージョン 2.85.0 以降
  - Docker バージョン 20.10.21 以降

- Node.js バージョン 18 以降
- Projen バージョン 0.71.111 以降
- Python バージョン 3.9.16 以降

機能制限

- スロットリングを避けるため、DynamoDB Streams の推奨リーダーの最大数は 2 です。
- ジョブの最大ランタイムは、Lambda 関数の最大ランタイム (15 分) によって制限されます。
- 同時ジョブリクエストの最大数は、Lambda 関数の予約済み同時実行数によって制限されます。

アーキテクチャ

アーキテクチャ

次の図は、ジョブ API と DynamoDB Streams の相互作用と、Amazon EventBridge イベントアーカイブに保存されたイベントとのイベント処理およびエラー処理 Lambda 関数を示しています。

一般的なワークフローには、次のステップが含まれます。

1. AWS Identity and Access Management (IAM) に対して認証し、セキュリティ認証情報を取得します。
2. HTTP `POST`リクエストを`/jobs`ジョブ API エンドポイントに送信し、リクエスト本文でジョブパラメータを指定します。
3. ジョブ API は、ジョブ識別子を含む HTTP レスポンスを返します。
4. ジョブ API は、ジョブパラメータを `jobs_table` Amazon DynamoDB テーブルに配置します。
5. `jobs_table` DynamoDB テーブルの DynamoDB ストリームは、イベント処理 Lambda 関数を呼び出します。
6. イベント処理 Lambda 関数はイベントを処理し、ジョブ結果を DynamoDB `jobs_table` テーブルに配置します。一貫した結果を確保するために、イベント処理関数は楽観的なロックメカニズムを実装します。
7. ステップ 3 の`/jobs/{jobId}`ジョブ識別子を として、HTTP `GET`リクエストをジョブ API エンドポイントに送信します`{jobId}`。
8. ジョブ API は DynamoDB `jobs_table` テーブルにクエリを実行してジョブ結果を取得します。

9. ジョブ API は、ジョブ結果を含む HTTP レスポンスを返します。
10. イベント処理が失敗した場合、イベント処理関数のソースマッピングは、エラー処理を行う Amazon Simple Notification Service (Amazon SNS) トピックにイベントを送信します。
11. エラー処理 SNS トピックは、イベントをエラー処理関数に非同期的にプッシュします。
12. エラー処理関数は、DynamoDB `jobs_table` テーブルにジョブパラメータを配置します。

    ジョブ API エンドポイントに HTTP `GET`リクエストを送信することで、`/jobs/{jobId}`ジョブパラメータを取得できます。
13. エラー処理が失敗した場合、エラー処理関数はイベントを Amazon EventBridge アーカイブに送信します。

    EventBridge を使用して、アーカイブされたイベントを再生できます。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウドインフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを提供します。
- 「Amazon EventBridge」 は、アプリケーションをさまざまなソースのデータに接続するために支援するサーバーレスイベントバスサービスです。たとえば、AWS Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

その他のツール

- autopep8 は、Python Enhancement Proposal (PEP) 8 スタイルガイドに基づいて Python コードを自動的にフォーマットします。
- Bandit は Python コードをスキャンして、一般的なセキュリティ問題を見つけます。

- Commitizen は Git コミットチェッカーとCHANGELOGジェネレーターです。
- cfn-lint は linter AWS CloudFormation です
- Checkov は、Infrastructure as Code (IaC) のセキュリティとコンプライアンスの設定ミスをチェックする静的コード分析ツールです。
- jq は JSON を解析するためのコマンドラインツールです。
- Postman は API プラットフォームです。
- 事前コミットは Git フックマネージャーです。
- Projen はプロジェクトジェネレーターです。
- 「pytest」は、小さくて読みやすいテストを書くための Python フレームワークです。

コードリポジトリ

このアーキテクチャコードの例は、GitHub の API Gateway と DynamoDB Streams による非同期処理リポジトリにあります。

ベストプラクティス

- この例のアーキテクチャには、デプロイされたインフラストラクチャのモニタリングは含まれません。ユースケースでモニタリングが必要な場合は、CDK Monitoring Constructs または別のモニタリングソリューションの追加を評価します。
- このアーキテクチャ例では、IAM アクセス許可を使用してジョブ API へのアクセスを制御します。を引き受ける権限を持つユーザーはJobsAPIInvokeRole、ジョブ API を呼び出すことができます。そのため、アクセスコントロールメカニズムはバイナリです。ユースケースでより複雑な認可モデルが必要な場合は、別のアクセスコントロールメカニズムを使用して評価します。
- ユーザーが/jobsジョブ API エンドポイントに HTTP POSTリクエストを送信すると、入力データは 2 つの異なるレベルで検証されます。
  - API Gateway は、最初のリクエスト検証を担当します。
  - イベント処理関数は 2 番目のリクエストを実行します。

    ユーザーが/jobs/{jobId}ジョブ API エンドポイントに HTTP GETリクエストを実行する場合、検証は実行されません。ユースケースで追加の入力検証とセキュリティレベルの向上が必要な場合は、 AWS WAF を使用して API を保護します。

- スロットリングを避けるために、DynamoDB Streams ドキュメントでは、ユーザーが同じストリームのシャードから 3 人以上のコンシューマーと読むことを推奨していません。コンシュー

マーの数をスケールアウトするには、Amazon Kinesis Data Streams を使用することをお勧めします。

- この例では、DynamoDB `jobs_table` テーブル内の項目を一貫して更新するために、オプティミスティックロックが使用されています。ユースケースの要件によっては、悲観的ロックなど、より信頼性の高いロックメカニズムを実装する必要がある場合があります。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリのクローンをローカルに作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-dynamodb-streams-cdk.git`	DevOps エンジニア

プロジェクトをセットアップします。	ディレクトリをリポジトリルートに変更し、Projen を使用して Python 仮想環境とすべてのツールを設定します。
	`cd asynchronous-event-processing-api-gateway-api-gateway-dynamodb-streams-cdk`
	`npx projen`
		DevOps エンジニア

事前コミットフックをインストールします。	事前コミットフックをインストールするには、次の手順を実行します。	DevOps エンジニア

タスク	説明	必要なスキル

	1. Python 仮想環境をアクティブ化します。
	`source .env/bin/activate`
	2. 事前コミットフックをインストールします。
	`pre-commit install`
	`pre-commit install --hook-type commit-msg`	

サンプルアーキテクチャをデプロイする

タスク	説明	必要なスキル
ブートストラップ AWS CDK。	AWS CDK でブートストラップするには AWS アカウント、次のコマンドを実行します。 `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen bootstrap`	AWS DevOps
サンプルアーキテクチャをデプロイします。	にサンプルアーキテクチャをデプロイするには AWS アカウント、次のコマンドを実行します。 `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy`	AWS DevOps

アーキテクチャをテストする

タスク	説明	必要なスキル
テストの前提条件をインストールします。	AWS Command Line Interface（AWS CLI）、Postman、jq をワークステーションにインストールします。 Postman を使用してこのサンプルアーキテクチャをテストすることをお勧めしますが、必須ではありません。代替 API テストツールを選択する場合は、AWS 署名バージョン 4 認証をサポートし、REST API をエクスポートして検査できる公開 API エンドポイントを参照するようにしてください。	DevOps エンジニア
を引き受けますJobsAPIInvokeRole 。	deploy コマンドから出力として出力JobsAPIInvokeRole された を仮定します。 `CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \ --no-cli-pager \ --role-arn $<JOBS_API_INVOKE_ROLE_ARN> \ --role-session-name JobsAPIInvoke) export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS | jq '.Credentials''.AccessKeyId')`	AWS DevOps

タスク	説明	必要なスキル
	`export AWS_SECRE T_ACCESS_KEY=$(cat $CREDENTIALS \| jq '.Credentials''.Se cretAccessKey') export AWS_SESSI ON_TOKEN==$(cat $CREDENTIALS | jq '.Credentials''.Se ssionToken')`	

タスク	説明	必要なスキル

Postman を設定します。	• リポジトリに含まれる Postman コレクションをインポートするには、Postman ドキュメントの指示に従います。
	• JobsAPI 変数を次の値で設定します。
	  • accessKey assume-role – コマンドの Credentials.AccessKeyId 属性の値。
	  • baseUrl – 末尾にスラッシュを付けずに、deployコマンドからのJobsApiJobsAPIEndpoint 出力の値。
	  • region – サンプルアーキテクチャをデプロイ AWS リージョン した の値。
	  • seconds – サンプルジョブの入力パラメータの値。正の整数である必要があります。
	  • secretKey assume-role – コマンドの Credentials.SecretAccessKey 属性の値。
	  • sessionToken assume-role – コマンドの Credentia
		AWS DevOps

タスク	説明	必要なスキル
	`ls.SessionToken` 属性の値。	
サンプルアーキテクチャをテストします。	サンプルアーキテクチャをテストするには、ジョブ API にリクエストを送信します。詳細については、Postman ドキュメントを参照してください。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
Amazon CloudWatch Logs ロググループ `/aws/apigateway/JobsAPIAccessLogs` がすでに存在するため、サンプルアーキテクチャの破壊とその後の再デプロイは失敗します。	1. 必要に応じて、ログデータを Amazon Simple Storage Service (Amazon S3) にエクスポートします。 2. CloudWatch Logs ロググループ を削除します `/aws/apigateway/JobsAPIAccessLogs` 。 3. サンプルアーキテクチャを再デプロイします。

関連リソース

- API Gateway マッピングテンプレートとアクセスロギング変数リファレンス
- DynamoDB Streams のデータキャプチャを変更する
- バージョン番号によるオプティミスティックロック
- Kinesis Data Streams を使用して DynamoDB への変更をキャプチャする

Amazon API Gateway、Amazon SQS、および AWS Fargate でイベントを非同期的に処理する

作成者: Andrea Meroni (AWS)、Alessandro Trisolini (AWS)、Nadim Majed (AWS)、Marime Kthiri (AWS)、Michael Wallner (AWS)

概要

Amazon API Gateway は、開発者があらゆる規模で APIs を作成、公開、保守、モニタリング、保護するために使用できるフルマネージドサービスです。最大数十万の同時 API コールの受け入れと処理に関連するタスクを処理します。

API Gateway の重要なサービスクォータは、統合タイムアウトです。タイムアウトは、REST API がエラーを返す前にバックエンドサービスがレスポンスを返す必要がある最大時間です。29 秒のハード制限は、同期ワークロードで一般的に許容されます。ただし、この制限は、非同期ワークロードで API Gateway を使用するデベロッパーにとって課題となります。

このパターンは、API Gateway、Amazon Simple Queue Service (Amazon SQS)、および を使用してイベントを非同期的に処理するアーキテクチャの例を示しています AWS Fargate。このアーキテクチャは、期間制限なしでの処理ジョブの実行をサポートし、インターフェイスとして基本的な REST API を使用します。

Projen は、ローカル開発環境をセットアップし AWS アカウント、、DockerAWS Cloud Development Kit (AWS CDK)、Node.js と組み合わせてサンプルアーキテクチャをターゲットにデプロイするために使用されます。Projen は、事前コミットと、コードの品質保証、セキュリティスキャン、ユニットテストに使用されるツールを使用して、Python 仮想環境を自動的にセットアップします。詳細については、「 ツール」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ワークステーションにインストールされている以下のツール：
  - AWS Cloud Development Kit (AWS CDK) ツールキットバージョン 2.85.0 以降
  - Docker バージョン 20.10.21 以降
  - Node.js バージョン 18 以降
  - Projen バージョン 0.71.111 以降

- Python バージョン 3.9.16 以降

制約事項

- 同時ジョブは 1 分あたり 500 タスクに制限されます。これは、Fargate がプロビジョニングできるタスクの最大数です。

アーキテクチャ

次の図は、ジョブ API と `jobs` Amazon DynamoDB テーブル、イベント処理 Fargate サービス、およびエラー処理 AWS Lambda 関数の相互作用を示しています。イベントは Amazon EventBridge イベントアーカイブに保存されます。

一般的なワークフローには、次のステップが含まれます。

1. AWS Identity and Access Management (IAM) に対して認証し、セキュリティ認証情報を取得します。
2. HTTP `POST`リクエストを`/jobs`ジョブ API エンドポイントに送信し、リクエスト本文のジョブパラメータを指定します。
3. API Gateway REST API であるジョブ API は、ジョブ識別子を含む HTTP レスポンスを返します。
4. ジョブ API は、SQS キューにメッセージを送信します。
5. Fargate は SQS キューからメッセージをプルし、イベントを処理し、ジョブ結果を DynamoDB `jobs` テーブルに配置します。
6. ステップ 3 の`/jobs/{jobId}`ジョブ識別子を として、ジョブ API エンドポイントに HTTP `GET`リクエストを送信します`{jobId}`。
7. ジョブ API は、DynamoDB `jobs` テーブルにクエリを実行してジョブ結果を取得します。
8. ジョブ API は、ジョブ結果を含む HTTP レスポンスを返します。
9. イベント処理が失敗した場合、SQS キューはイベントをデッドレターキュー (DLQ) に送信します。
10. EventBridge イベントはエラー処理関数を開始します。
11. エラー処理関数は、DynamoDB `jobs` テーブルにジョブパラメータを配置します。
12. ジョブ API エンドポイントに HTTP `GET`リクエストを送信することで、`/jobs/{jobId}`ジョブパラメータを取得できます。

13エラー処理が失敗した場合、エラー処理関数はイベントを EventBridge アーカイブに送信します。

EventBridge を使用して、アーカイブされたイベントを再生できます。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Fargate は、サーバーや Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを管理することなくコンテナを実行するのに役立ちます。Amazon Elastic Container Service (Amazon ECS) と組み合わせて使用されます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Queue Service (Amazon SQS)」は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。

その他のツール

- autopep8 は、Python Enhancement Proposal (PEP) 8 スタイルガイドに基づいて Python コードを自動的にフォーマットします。
- Bandit は Python コードをスキャンして、一般的なセキュリティ問題を見つけます。
- Commitizen は Git コミットチェッカーとCHANGELOGジェネレーターです。
- cfn-lint は linter AWS CloudFormation です
- Checkov は、Infrastructure as Code (IaC) のセキュリティとコンプライアンスの設定ミスをチェックする静的コード分析ツールです。

- jq は JSON を解析するためのコマンドラインツールです。
- Postman は API プラットフォームです。
- 事前コミットは Git フックマネージャーです。
- Projen はプロジェクトジェネレーターです。
- 「pytest」は、小さくて読みやすいテストを書くための Python フレームワークです。

コードリポジトリ

このアーキテクチャコードの例は、GitHub の API Gateway と SQS による非同期処理リポジトリにあります。

ベストプラクティス

- この例のアーキテクチャには、デプロイされたインフラストラクチャのモニタリングは含まれません。ユースケースでモニタリングが必要な場合は、CDK モニタリングコンストラクトまたは別のモニタリングソリューションの追加を評価します。
- このアーキテクチャ例では、IAM アクセス許可を使用してジョブ API へのアクセスを制御します。を引き受ける権限を持つユーザーは`JobsAPIInvokeRole`、ジョブ API を呼び出すことができます。そのため、アクセスコントロールメカニズムはバイナリです。ユースケースでより複雑な認可モデルが必要な場合は、別のアクセスコントロールメカニズムを使用して評価します。
- ユーザーが`/jobs`ジョブ API エンドポイントに HTTP `POST`リクエストを送信すると、入力データは 2 つの異なるレベルで検証されます。
  - API Gateway は、最初のリクエスト検証を担当します。
  - イベント処理関数は 2 番目のリクエストを実行します。

    ユーザーが`/jobs/{jobId}`ジョブ API エンドポイントに HTTP `GET`リクエストを実行する場合、検証は実行されません。ユースケースで追加の入力検証とセキュリティレベルの向上が必要な場合は、 を使用して API AWS WAF を保護します。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリをローカルでクローンするには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-sqs-cdk.git`	DevOps エンジニア

プロジェクトをセットアップします。	ディレクトリをリポジトリルートに変更し、Projen を使用して Python 仮想環境とすべてのツールを設定します。
	
	`cd asynchronous-event-processing-api-gateway-api-gateway-sqs-cdk`
	`npx projen`
		DevOps エンジニア

事前コミットフックをインストールします。	事前コミットフックをインストールするには、次の手順を実行します。 1. Python 仮想環境をアクティブ化します。 `source .env/bin/activate` 2. コミット前フックをインストールします。	DevOps エンジニア

タスク	説明	必要なスキル
	`pre-commit install` `pre-commit install --hook-type commit-msg`	

サンプルアーキテクチャをデプロイする

タスク	説明	必要なスキル
ブートストラップ AWS CDK。	AWS CDK でブートストラップするには AWS アカウント、次のコマンドを実行します。 `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen bootstrap`	AWS DevOps
サンプルアーキテクチャをデプロイします。	にサンプルアーキテクチャをデプロイするには AWS アカウント、次のコマンドを実行します。 `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy`	AWS DevOps

アーキテクチャをテストする

タスク	説明	必要なスキル
テストの前提条件をインストールします。	AWS Command Line Interface（AWS CLI）、Postman、jq	DevOps エンジニア

タスク	説明	必要なスキル
	をワークステーションにインストールします。 Postman を使用してこのサンプルアーキテクチャをテストすることは推奨されますが、必須ではありません。代替 API テストツールを選択する場合は、AWS 署名バージョン 4 認証をサポートし、REST API をエクスポートして検査できる公開 API エンドポイントを参照するようにしてください。	

タスク	説明	必要なスキル
を引き受けますJobsAPIInvokeRole 。	deploy コマンドから出力として出力JobsAPIInvokeRole された を想定します。 CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \ --no-cli-pager \ --role-arn $<JOBS_API_INVOKE_ROLE_ARN> \ --role-session-name JobsAPIInvoke) export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS \| jq '.Credentials''.AccessKeyId') export AWS_SECRET_ACCESS_KEY=$(cat $CREDENTIALS \| jq '.Credentials''.SecretAccessKey') export AWS_SESSION_TOKEN==$(cat $CREDENTIALS | jq '.Credentials''.SessionToken')	AWS DevOps

タスク	説明	必要なスキル

Postman を設定します。	• リポジトリに含まれる Postman コレクションをインポートするには、Postman ドキュメントの指示に従います。
	• JobsAPI 変数を次の値で設定します。
	  • accessKey assume-role – コマンドの Credentials.AccessKeyId 属性の値。
	  • baseUrl – 末尾にスラッシュを付けずに、deployコマンドからのJobsApiJobsAPIEndpoint 出力の値。
	  • region – サンプルアーキテクチャをデプロイ AWS リージョン した の値。
	  • seconds – サンプルジョブの入力パラメータの値。正の整数である必要があります。
	  • secretKey assume-role – コマンドの Credentials.SecretAccessKey 属性の値。
	  • sessionToken assume-role – コマンドの Credentia
		AWS DevOps

タスク	説明	必要なスキル
	`ls.SessionToken` 属性の値。	
サンプルアーキテクチャをテストします。	サンプルアーキテクチャをテストするには、ジョブ API にリクエストを送信します。詳細については、Postman ドキュメントを参照してください。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
Amazon CloudWatch Logs ロググループ `/aws/apigateway/JobsAPIAccessLogs` がすでに存在するため、サンプルアーキテクチャの破壊とその後の再デプロイは失敗します。	1. 必要に応じて、ログデータを Amazon Simple Storage Service (Amazon S3) にエクスポートします。 2. CloudWatch Logs ロググループ を削除します `/aws/apigateway/JobsAPIAccessLogs` 。 3. サンプルアーキテクチャを再デプロイします。
CloudWatch Logs ロググループ `/aws/ecs/EventProcessingServiceLogs` がすでに存在するため、サンプルアーキテクチャの破壊とその後の再デプロイは失敗します。	1. 必要に応じて、ログデータを Amazon S3 にエクスポートします。 2. CloudWatch Logs ロググループを削除する `/aws/ecs/EventProcessingServiceLogs.` 3. サンプルアーキテクチャを再デプロイします。

関連リソース

- API Gateway マッピングテンプレートとアクセスロギング変数リファレンス
- API Gateway REST API を Amazon SQS と統合し、一般的なエラーを解決するにはどうすればよいですか？

AWS Step Functions から AWS Systems Manager Automation タスクを同期的に実行する AWS Step Functions

作成者: Elie El khoury (AWS)

概要

このパターンでは、 AWS Step Functions と を統合する方法について説明します AWS Systems Manager。 AWS SDK サービス統合を使用して、ステートマシンワークフローからタスクトークンで Systems Manager startAutomationExecution API を呼び出し、トークンが成功または失敗の呼び出しで戻るまで一時停止します。統合を実証するために、このパターンでは `AWS-RunShellScript`または ドキュメントの周囲にオートメーション`AWS-RunPowerShellScript`ドキュメント (ランブック) ラッパーを実装し、 を使用して `AWS-RunShellScript`または `.waitForTaskToken`を同期的に呼び出します`AWS-RunPowerShellScript`。Step Functions での AWS SDK サービス統合の詳細については、「 AWS Step Functions デベロッパーガイド」を参照してください。

Step Functions は、分散アプリケーションの構築、IT およびビジネスプロセスの自動化、サービスを使用したデータおよび機械学習パイプラインの構築に使用できる、ローコードのビジュアルワークフロー AWS サービスです。ワークフローは失敗、再試行、並列化、サービス統合、オブザーバビリティを管理するので、より価値の高いビジネスロジックに集中できます。

の一機能であるオートメーションは AWS Systems Manager、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Relational Database Service (Amazon RDS)、Amazon Redshift、Amazon Simple Storage Service (Amazon S3) AWS のサービス などの の一般的なメンテナンス、デプロイ、修復タスクを簡素化します。Amazon S3 オートメーションを使用すると、自動化の同時実行性をきめ細かく制御できます。例えば、同時実行のターゲットにするリソースの数や、オートメーションを停止する前に許容可能なエラーの発生数を指定することが可能です。

ランブックのステップ、パラメータ、例など、実装の詳細については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Identity and Access Management Step Functions と Systems Manager にアクセスするための (IAM) アクセス許可

- インスタンスに Systems Manager Agent (SSM Agent) がインストールされている EC2 インスタンス
- ランブックを実行する予定のインスタンスにアタッチされた Systems Manager の IAM インスタンスプロファイル
- 次の IAM アクセス許可を持つ Step Functions ロール (最小権限の原則に従います) 。

```
{
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": "*"
 }
```

製品バージョン

- SSM ドキュメントスキーマバージョン 0.3 以降
- SSM エージェントバージョン 2.3.672.0 以降。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Step Functions
- AWS Systems Manager Automation

ターゲットアーキテクチャ

自動化とスケール

- このパターンは、ランブックを複数のインスタンスにデプロイするために使用できる AWS CloudFormation テンプレートを提供します。(GitHub Step Functions and Systems Manager implementation を参照してください)。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Step Functions はサーバーレスオーケストレーションサービスであり、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立ちます。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理を簡素化し、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できるようにします。

コード

このパターンのコードは、GitHub 内の「Step Functions and Systems Manager implementation」リポジトリで利用できます。

エピック

ランブックの作成

タスク	説明	必要なスキル
CloudFormation テンプレートをダウンロードします。	GitHub リポジトリの `cloudformation` フォルダから `ssm-automation-documents.cfn.json` テンプレートをダウンロードします。	AWS DevOps
ランブックを作成します。	にサインインし AWS Management Console、AWS CloudFormation コンソールを開き、テンプレートをデプロイします。CloudFormation テンプレートのデプロイの詳細については、CloudFo	AWS DevOps

タスク	説明	必要なスキル
	rmation ドキュメントのAWS CloudFormation 「コンソールでのスタックの作成」を参照してください。 CloudFormation CloudFormation テンプレートは 3 つのリソースをデプロイします。 • `SfnRunCommandByInstanceIds` – インスタンス IDs `AWS-RunPowerShellScript` を使用して または を実行`AWS-RunShellScript` できるようにするランブック。 • `SfnRunCommandByTargets` – ターゲット`AWS-RunPowerShellScript` を使用して `AWS-RunShellScript` または を実行できるようにするランブック。 • `SSMSyncRole` – ランブックが引き受ける IAM ロール。	

サンプルステートマシンを作成する

タスク	説明	必要なスキル
テストステートマシンを作成します。	AWS Step Functions 「 デベロッパーガイド」の手順に	AWS DevOps

タスク	説明	必要なスキル
	従って、ステートマシンを作成して実行します。定義には、次のコードを使用します。必ず、アカウント内の有効な Systems Manager 対応インスタンスの ID で InstanceIds 値を更新してください。 `{ "Comment": "A description of my state machine", "StartAt": "StartAutomationWaitForCallBack", "States": { "StartAutomationWaitForCallBack": { "Type": "Task", "Resource": "arn:aws:states:::aws-sdk:ssm:startAutomationExecution.waitForTaskToken", "Parameters": { "DocumentName": "SfnRunCommandByInstanceIds", "Parameters": { "InstanceIds": ["i-1234567890abcdef0"], "taskToken.$": "States.Array($$.Task.Token)", "workingDirectory": [`	

タスク	説明	必要なスキル

	<pre>"/home/ssm-user/"
	],
	"Commands": [
	"echo \"This is a test running automation waitForTaskToken\" >> automation.log",
	"sleep 100"
	],
	"executionTimeout": [
	"10800"
	],
	"deliveryTimeout": [
	"30"
	],
	"shell": [
	"Shell"
	]
	}
	},
	"End": true
	}
	}
	}</pre>	

	このコードはランブックを呼び出して、Systems Manager Automation への `waitForTaskToken` 呼び出しを示す 2 つのコマンドを実行します。	
	`shell` パラメータ値 (`Shell` または `PowerShell`) は、自動化ドキュメントが `AWS-RunShellScript` または を実行するかどうかを決定し	

タスク	説明	必要なスキル
	ますAWS-RunPowerShellScript 。 タスクは「これはテスト実行中のオートメーション waitForTaskToken」を /home/ssm-user/automation.log ファイルに書き込み、タスクトークンで応答してワークフロー内の次のタスクを解放する前に 100 秒間スリープします。 代わりに SfnRunCommandByTargets ランブックを呼び出したい場合は、前のコードの Parameters セクションを以下のコードに置き換えてください。 `"Parameters": { "Targets": [ { "Key": "InstanceIds", "Values": [ "i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE" ] } ],`	

タスク	説明	必要なスキル
ステートマシンの IAM ロールを更新します。	前のステップでは、ステートマシンの専用の IAM ロールが自動的に作成されます。ただし、ランブックを呼び出すアクセス許可は付与されません。以下のアクセス許可を追加して、ロールを更新します。 `{ "Effect": "Allow", "Action": "ssm:StartAutomationExecution", "Resource": "*" }`	AWS DevOps
同期呼び出しを検証します。	ステートマシンを実行して、Step Functions と Systems Manager オートメーション間の同期呼び出しを検証します。 出力例については、「追加情報」セクションを参照してください。	AWS DevOps

関連リソース

- の開始方法 AWS Step Functions (AWS Step Functions 開発者ガイド)
- タスクトークンによるコールバックを待機する (AWS Step Functions 開発者ガイド、サービス統合パターン)
- send_task_success と send_task_failure の API コール (Boto3 ドキュメント)
- AWS Systems Manager 自動化 (AWS Systems Manager ユーザーガイド)

追加情報

実装の詳細

このパターンは、2 つの Systems Manager ランブックをデプロイする CloudFormation テンプレートを提供します。

- SfnRunCommandByInstanceIdsは、インスタンス IDs を使用して AWS-RunShellScriptまたは AWS-RunPowerShellScript コマンドを実行します。
- SfnRunCommandByTargetsは、ターゲットを使用して AWS-RunShellScriptまたは AWS-RunPowerShellScript コマンドを実行します。

各ランブックは、Step Functions で .waitForTaskTokenオプションを使用する場合に同期呼び出しを達成するための 4 つのステップを実装します。

[ステップ]	[アクション]	説明
1	Branch	shell パラメータ値 (Shell または PowerShell) をチェックして、Linux AWS-RunShellScript 用と Windows AWS-RunPowerShellScript 用のどちらを実行するかを決定します。
2	RunCommand_Shell 、または RunCommand_PowerShell	複数の入力を取得し、RunShellScript または RunPowerShellScript コマンドを実行します。詳細については、Systems Manager コンソールで RunCommand_Shell または RunCommand_PowerShell Automation ドキュメントの詳細タブを確認してください。

3	SendTaskFailure	ステップ 2 が中止またはキャンセルされたときに実行されます。Step Functions send_task_failure API を呼び出し、ステートマシンから渡されたトークン、障害エラー、障害の原因の説明の 3 つのパラメータを入力として受け入れます。
4	SendTaskSuccess	ステップ 2 が成功すると実行されます。ステートマシンから渡されたトークンを入力として受け入れる Step Functions send_task_success API を呼び出します。

ランブックパラメータ

SfnRunCommandByInstanceIds ランブック：

パラメータ名	Type	オプションまたは必須	説明
shell	文字列	必須	インスタンスシェルは、Linux または AWS-RunPowerShellScript Windows AWS-RunShellScript のどちらで を実行するかを決定します。
deliveryTimeout	整数	オプションです。	コマンドがインスタンスの SSM エージェントに配信されるまで待機する秒単位の時間。このパラ

			メータの最小値は 30 (0.5 分) で、最大値は 2592000「」(720 時間) です。
executionTimeout	String	オプションです。	コマンドが失敗したと見なされるまでに完了するまでの時間 (秒単位)。デフォルト値は 3600 (1 時間) です。最大値は 172800 (48 時間) です。
workingDirectory	String	オプションです。	インスタンスの作業ディレクトリへのパス。
Commands	StringList	必須	実行するシェルスクリプトまたはコマンド。
InstanceIds	StringList	必須	コマンドを実行するインスタンス ID です。
taskToken	String	必須	コールバックレスポンスに使用するタスクトークン。

SfnRunCommandByTargetsランブック：

名前	Type	オプションまたは必須	説明
shell	文字列	必須	インスタンスシェルは、Linux または AWS-RunPowerShellScript Windows AWS-RunSh

			ellScript のどちらで を実行するかを決定します。
deliveryTimeout	整数	オプションです。	コマンドがインスタンスの SSM エージェントに配信されるまで待機する秒単位の時間。このパラメータの最小値は 30 (0.5 分) で、最大値は 2592000「」(720 時間) です。
executionTimeout	整数	オプションです。	コマンドが失敗したと見なされるまでに完了するまでの時間 (秒単位)。デフォルト値は 3600 (1 時間) です。最大値は 172800 (48 時間) です。
workingDirectory	String	オプションです。	インスタンスの作業ディレクトリへのパス。
Commands	StringList	必須	実行するシェルスクリプトまたはコマンド。

Targets	MapList	必須	指定したキーと値のペアを使用してインスタンスを識別する検索条件の配列。例: [{"Key":"InstanceIds","Values":["i-02573cafcfEXAMPLE","i-0471e04240EXAMPLE"]}]
taskToken	String	必須	コールバックレスポンスに使用するタスクトークン。

出力例

次の表に、ステップ関数からの出力例を示します。ステップ 5 (TaskSubmitted) からステップ 6 (TaskSucceeded) までの合計実行時間が 100 秒を超えていることがわかります。これは、ステップ関数が、ワークフローの次のタスクに移動する前にsleep 100コマンドが終了するのを待ったことを示しています。

ID	Type	[ステップ]	[リソース]	経過時間 (ミリ秒)	タイムスタンプ
1	ExecutionStarted		-	0	2022 年 3 月 11 日午後 2 時 50 分 34.303 秒
2	TaskStateEntered	StartAutomationWaitForCallBack	-	40	2022 年 3 月 11 日午後 2 時 50 分 34.343 秒

3	TaskScheduled	StartAutomationWaitForCallBack	-	40	2022 年 3 月 11 日午後 2 時 50 分 34.343 秒
4	TaskStarted	StartAutomationWaitForCallBack	-	154	2022 年 3 月 11 日午後 2 時 50 分 34.457 秒
5	TaskSubmitted	StartAutomationWaitForCallBack	-	657	2022 年 3 月 11 日午後 2 時 50 分 34.960 秒
6	TaskSucceeded	StartAutomationWaitForCallBack	-	103835	2022 年 3 月 11 日午後 2 時 52 分 18.138 秒
7	TaskStateExited	StartAutomationWaitForCallBack	-	103860	2022 年 3 月 11 日午後 2 時 52 分 18.163 秒
8	ExecutionSucceeded		-	103897	2022 年 3 月 11 日午後 2 時 52 分 18.200 秒

AWS Lambda 関数で Python を使用して S3 オブジェクトの並列読み取りを実行する

作成者: Eduardo Bortoluzzi (AWS)

概要

このパターンを使用して、Amazon Simple Storage Service (Amazon S3) バケットからドキュメントのリストをリアルタイムで取得して要約できます。このパターンは、Amazon Web Services () の S3 バケットからオブジェクトを並列読み取りするコード例を示していますAWS。このパターンは、Python を使用して AWS Lambda 関数を使用して I/O バインドタスクを効率的に実行する方法を示しています。

ある金融会社は、このパターンをインタラクティブなソリューションで使用して、相関する金融取引をリアルタイムで手動で承認または拒否しました。財務トランザクションドキュメントは、市場に関連する S3 バケットに保存されました。オペレーターは S3 バケットからドキュメントのリストを選択し、ソリューションが計算したトランザクションの合計値を分析して、選択したバッチを承認または拒否することを決定しました。

I/O バインドタスクは複数のスレッドをサポートします。このサンプルコードでは、Lambda 関数が最大 1,024 個のスレッドをサポートしている場合でも、concurrent.futures.ThreadPoolExecutor は最大 30 個の同時スレッドで使用されます (これらのスレッドの 1 つがメインプロセスです)。この制限は、スレッドが多すぎると、コンテキストの切り替えとコンピューティングリソースの使用率が原因でレイテンシーの問題が発生するためです。また、すべてのスレッドが S3 オブジェクトのダウンロードを同時に実行botocoreできるように、 でプールの最大接続数を増やす必要があります。

サンプルコードでは、S3 バケット内の JSON データを含む 8.S3 KB オブジェクトを 1 つ使用します。オブジェクトは複数回読み込まれます。Lambda 関数がオブジェクトを読み取ると、JSON データは Python オブジェクトにデコードされます。2024 年 12 月、この例を実行した後の結果は、2,304 MB のメモリで設定された Lambda 関数を使用して 2.3 秒で 1,000 回の読み取り、27 秒で 10,000 回の読み取りでした。 は、128 MB から 10,240 MB (10 GB) のメモリ設定 AWS Lambda をサポートしますが、Lambdamemory を 2,304 MB を超えて増やすと、この特定の I/O バウンドタスクの実行時間を短縮するには役立ちませんでした。

AWS Lambda Power Tuning ツールを使用して、さまざまな Lambda メモリ設定をテストし、タスクの最適なperformance-to-costの比率を検証しました。テスト結果については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Python 開発の習熟度

機能制限

- Lambda 関数には、最大 1,024 の実行プロセスまたはスレッドを含めることができます。
- 新しい AWS アカウント の Lambda メモリ制限は 3,008 MB です。それに応じて AWS Lambda Power Tuning ツールを調整します。詳細については、「トラブルシューティング」セクションを参照してください。
- Amazon S3 には、パーティション化されたプレフィックスごとに 1 秒あたり 5,500 件の GET/HEAD リクエストの制限があります。

製品バージョン

- Python 3.9 以降
- AWS Cloud Development Kit (AWS CDK) v2
- AWS Command Line Interface （ AWS CLI) バージョン 2
- AWS Lambda 電源調整 4.3.6 (オプション ）

アーキテクチャ

ターゲットテクノロジースタック

- AWS Lambda
- Amazon S3
- AWS Step Functions （ AWS Lambda パワーチューニングがデプロイされている場合 ）

ターゲット アーキテクチャ

次の図は、S3 バケットからオブジェクトを並行して読み取る Lambda 関数を示しています。この図には、Lambda 関数メモリを微調整するための AWS Lambda Power Tuning ツールの Step Functions

ワークフローも含まれています。この微調整は、コストとパフォーマンスのバランスをとるのに役立ちます。

自動化とスケール

Lambda 関数は、必要に応じて高速にスケーリングされます。需要が高いときに Amazon S3 から 503 スローダウンエラーが発生しないように、スケーリングにいくつかの制限を設定することをお勧めします。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) v2 は、コードで AWS クラウド インフラストラクチャを定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。サンプルインフラストラクチャは、デプロイ用に作成されました AWS CDK。
- AWS Command Line InterfaceAWS CLI は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。このパターンでは、 AWS CLI バージョン 2 を使用してサンプル JSON ファイルをアップロードします。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service Amazon S3 は、任意の量のデータを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functions は、 AWS Lambda 関数と他の AWS のサービスを組み合わせてビジネスクリティカルなアプリケーションを構築するのに役立つサーバーレスオーケストレーションサービスです。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。アイドル状態のワーカースレッドの再利用は Python バージョン 3.8 で導入され、このパターンの Lambda 関数コードは Python バージョン 3.9 以降用に作成されました。

コードリポジトリ

このパターンのコードは、aws-lambda-parallel-download GitHub リポジトリで入手できます。

ベストプラクティス

- この AWS CDK コンストラクトは、インフラストラクチャをデプロイするための AWS アカウントユーザーのアクセス許可に依存します。 AWS CDK Pipelines またはクロスアカウントデプロイを使用する場合は、「スタックシンセサイザー」を参照してください。
- このサンプルアプリケーションでは、S3 バケットでアクセスログが有効になっていません。本番コードでアクセスログを有効にするのがベストプラクティスです。

エピック

開発環境を準備する

タスク	説明	必要なスキル
Python のインストール済みバージョンを確認します。	このコードは Python 3.9 と Python 3.13 で特別にテストされており、これらのリリース間のすべてのバージョンで動作するはずです。Python のバージョンを確認するには、ターミナル`python3 -V`で を実行し、必要に応じて新しいバージョンをインストールします。 必要なモジュールがインストールされていることを確認するには、 を実行します`python3 -c "import pip, venv"`。エラーメッセージが表示されない場合、モジュールが正しくインストールされ、この例を実行する準備ができていることを意味します。	クラウドアーキテクト

タスク	説明	必要なスキル
をインストールします AWS CDK。	まだインストール AWS CDK されていない場合は、「 の開始方法 AWS CDK」の手順に従ってください。インストールされている AWS CDK バージョンが 2.0 以降であることを確認するには、 を実行しますcdk -version。	クラウドアーキテクト
環境 をブートストラップします。	環境をブートストラップするには、まだ実行していない場合は、「環境をブートストラップして で使用する AWS CDK」の手順に従います。	クラウドアーキテクト

サンプルリポジトリのクローンを作成する

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリの最新バージョンのクローンを作成するには、次のコマンドを実行します。 `git clone --depth 1 --branch v1.2.0 \ git@github.com:aws-samples/aws-lambda-parallel-download.git`	クラウドアーキテクト
作業ディレクトリをクローンされたリポジトリに変更します。	次のコマンドを実行してください。 `cd aws-lambda-parallel-download`	クラウドアーキテクト

タスク	説明	必要なスキル
Python 仮想環境を作成します。	Python 仮想環境を作成するには、次のコマンドを実行します。 `python3 -m venv .venv`	クラウドアーキテクト
仮想環境をアクティブ化します。	仮想環境をアクティブ化するには、次のコマンドを実行します。 `source .venv/bin/activate`	クラウドアーキテクト
SDK の依存関係をインストールします。	Python 依存関係をインストールするには、 `pip` コマンドを実行します。 `pip install -r requirements.txt`	クラウドアーキテクト
コードを参照します。	(オプション) S3 バケットからオブジェクトをダウンロードするサンプルコードは、 にありますresources/parallel.py 。 インフラストラクチャコードは parallel_download フォルダにあります。	クラウドアーキテクト

アプリをデプロイしてテストする

タスク	説明	必要なスキル
アプリケーションをデプロイします。	`cdk deploy` を実行します。 AWS CDK 出力を書き留めます。 • `ParallelDownloadStack.LambdaFunctionARN` • `ParallelDownloadStack.SampleS3BucketName` • `ParallelDownloadStack.StateMachineARN`	クラウドアーキテクト
サンプル JSON ファイルをアップロードします。	リポジトリには、約 9 KB のサンプル JSON ファイルが含まれています。作成したスタックの S3 バケットにファイルをアップロードするには、次のコマンドを実行します。 `aws s3 cp sample.json s3://<ParallelDownloadStack.SampleS3BucketName>` を AWS CDK 出力の対応する値`<ParallelDownloadStack.SampleS3BucketName>` に置き換えます。	クラウドアーキテクト

タスク	説明	必要なスキル
アプリを実行します。	アプリを実行するには、次の操作を行います。 1. にサインインし AWS Management Console、Lambda コンソールに移動して、 AWS CDK 出力 から ARN を持つ Lambda 関数を見つけますParallelDownloadStack.LambdaFunction ARN 。 2. テストタブで、イベント JSON を次のように変更します。 `{"objectKey": "sample.json"}` 3. [テスト] を選択します。 4. 結果を表示するには、詳細を選択します。詳細には、並列ダウンロードの統計、実行の情報、および口グが表示されます。	クラウドアーキテクト
ダウンロード数を追加します。	(オプション) オブジェクト呼び出しを 1,500 回取得するには、 Testパラメータのイベント JSON で次の JSON を使用します。 `{"repeat": 1500, "objectKey": "sample.json"}`	クラウドアーキテクト

オプション: AWS Lambda 電源調整を実行する

タスク	説明	必要なスキル
AWS Lambda Power Tuning ツールを実行します。	1. コンソールにサインインし、Step Functions に移動します。 2. AWS CDK 出力 から ARN を持つステートマシンを見つけますParallelDownloadStack.StateMachineARN 。 3. Start execution を選択し、次の JSON を貼り付けます。 `{ "lambdaARN": "<ParallelDownloadStack.LambdaFunctionARN>", "num": 10, "strategy": "balanced", "payload": {"repeat": 2000, "objectKey": "sample.json"} }` を AWS CDK 出力の値<ParallelDownloadStack.LambdaFunctionARN> に置き換えることを忘れないでください。 実行の終了時に、結果は実行入出力タブに表示されます。	クラウドアーキテクト

タスク	説明	必要なスキル
グラフで AWS Lambda Power Tuning の結果を表示します。	実行入出力タブで、visualization プロパティリンクをコピーし、新しいブラウザタブに貼り付けます。	クラウドアーキテクト

クリーンアップ

タスク	説明	必要なスキル
S3 バケットからオブジェクトを削除します。	デプロイされたリソースを破棄する前に、S3 バケットからすべてのオブジェクトを削除します。 `aws s3 rm s3://<ParallelDownloadStack.SampleS3BucketName> \ --recursive` を AWS CDK 出力の値<ParallelDownloadStack.SampleS3BucketName> に置き換えることを忘れないでください。	クラウドアーキテクト
リソースを破棄します。	このパイロット用に作成されたすべてのリソースを破棄するには、次のコマンドを実行します。 `cdk destroy`	クラウドアーキテクト

トラブルシューティング

問題	ソリューション
`'MemorySize' value failed to satisfy constraint: Member must have value less than or equal to 3008`	新しいアカウントでは、Lambda 関数で 3,008 MB を超える を設定できない場合があります。AWS Lambda Power Tuning を使用してテストするには、Step Functions の実行を開始するときに、入力 JSON に次のプロパティを追加します。 `"powerValues": [ 512, 1024, 1536, 2048, 2560, 3008 ]`

関連リソース

- Python – concurrent.futures.ThreadPoolExecutor
- Lambda クォータ — 関数の設定、デプロイ、実行
- Python AWS CDK での の操作
- AWS Lambda パワーチューニングによる関数のプロファイリング

追加情報

コード

次のコードスニペットは、並列 I/O 処理を実行します。

```
with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
  for result in executor.map(a_function, (the_arguments)):
    ...
```

は、スレッドが利用可能になるとThreadPoolExecutor再利用します。

テストと結果

これらのテストは 2024 年 12 月に実施されました。

最初のテストでは 2,500 件のオブジェクト読み取りが処理され、次の結果が返されました。

3,009 MB から、処理時間レベルはメモリの増加に伴ってほぼ同じままでしたが、メモリサイズの増加に伴ってコストが増加しました。

別のテストでは、256 MB の倍数の値を使用し、10,000 件のオブジェクト読み取りを処理して、1,536 MB から 3,072 MB のメモリの範囲を調査しました。その結果は次のとおりです。

最高のperformance-to-costの比率は、2,304 MB のメモリ Lambda 設定でした。

比較のために、2,500 回のオブジェクト読み取りのシーケンシャルプロセスには 47 秒かかりました。2,304 MB の Lambda 設定を使用した並列処理には 7 秒かかり、85% 短縮されました。

から OpenSearch AWS Lambda にテレメトリデータを送信して、リアルタイムの分析と視覚化を行う

作成者: Tabby Ward (AWS)、Guy Bachar (AWS)、David Kilzer (AWS)

概要

最新のアプリケーションはますます分散され、イベント駆動型になり、リアルタイムのモニタリングとオブザーバビリティの必要性が強化されています。 AWS Lambda は、スケーラブルでイベント駆動型のアーキテクチャを構築する上で重要な役割を果たしているサーバーレスコンピューティングサービスです。ただし、Amazon CloudWatch Logs のみに依存すると、Lambda 関数のモニタリングとトラブルシューティングが難しくなり、レイテンシーと保持期間が制限される可能性があります。

この課題に対処するために、Lambda テレメトリ API AWS を導入しました。これにより、Lambda 関数はテレメトリデータをサードパーティーのモニタリングおよびオブザーバビリティツールに直接送信できます。この API は、ログ、メトリクス、トレースのリアルタイムストリーミングをサポートし、Lambda 関数のパフォーマンスと状態を包括的かつタイムリーに表示します。

このパターンでは、オープンソースの分散検索および分析エンジンである OpenSearch と Lambda Telemetry API を統合する方法について説明します。OpenSearch は、大量のデータの取り込み、保存、分析のための強力でスケーラブルなプラットフォームを提供するため、Lambda テレメトリデータに最適です。具体的には、このパターンは、Python で記述された Lambda 関数から、 が提供する Lambda 拡張機能を使用して OpenSearch クラスターに直接ログを送信する方法を示しています AWS。このソリューションは柔軟でカスタマイズ可能なため、独自の Lambda 拡張機能を作成したり、サンプルソースコードを変更して必要に応じて出力形式を変更したりできます。

このパターンでは、Lambda Telemetry API と OpenSearch の統合を設定および設定する方法を説明し、セキュリティ、コスト最適化、スケーラビリティに関するベストプラクティスが含まれています。目的は、Lambda 関数をより深く理解し、サーバーレスアプリケーションの全体的なオブザーバビリティを向上させることです。

> **Note**
>
> このパターンは、Lambda Telemetry API とマネージド OpenSearch の統合に焦点を当てています。ただし、説明されている原則と手法は、セルフマネージド OpenSearch と Elasticsearch にも適用できます。

前提条件と制限

統合プロセスを開始する前に、次の前提条件が設定されていることを確認してください。

AWS アカウント: 次の AWS リソースを作成および管理するための適切なアクセス許可 AWS アカウント を持つアクティブな 。

- AWS Lambda
- AWS Identity and Access Management (IAM)
- Amazon OpenSearch Service (マネージド OpenSearch クラスターを使用している場合)

OpenSearch クラスター :

- 既存のセルフマネージド OpenSearch クラスターまたは OpenSearch Service などのマネージドサービスを使用できます。
- OpenSearch Service を使用している場合は、OpenSearch Service ドキュメントの「Amazon OpenSearch Service の開始方法」の手順に従って OpenSearch クラスターを設定します。
- OpenSearch クラスターが Lambda 関数からアクセス可能であり、アクセスポリシー、暗号化、認証などの必要なセキュリティ設定で設定されていることを確認します。
- Lambda テレメトリデータを取り込むために必要なインデックスマッピングと設定で OpenSearch クラスターを設定します。詳細については、OpenSearch Service ドキュメントの「Amazon OpenSearch Service へのストリーミングデータのロード」を参照してください。 OpenSearch

ネットワーク接続 :

- Lambda 関数に OpenSearch クラスターにアクセスするために必要なネットワーク接続があることを確認します。Virtual Private Cloud (VPC) の設定方法のガイダンスについては、OpenSearch Service ドキュメントの「VPC 内での Amazon OpenSearch Service ドメインの起動」を参照してください。 OpenSearch

IAM ロールとポリシー :

- Lambda 関数が OpenSearch クラスターにアクセスし、 に保存されている認証情報にアクセスするために必要なアクセス許可を持つ IAM ロールを作成します AWS Secrets Manager。
- OpenSearch を操作するために必要なポリシーや追加のアクセス許可など、適切な IAM `AWSLambdaBasicExecutionRole`ポリシーをロールにアタッチします。

- Lambda 関数に付与された IAM アクセス許可で、OpenSearch クラスターへのデータの書き込みが許可されていることを確認します。IAM アクセス許可の管理の詳細については、Lambda ドキュメントの「実行ロールを使用した Lambda 関数のアクセス許可の定義」を参照してください。

プログラミング言語の知識：

- Lambda 関数と Lambda 拡張機能のサンプルコードを理解して変更するには、Python (または選択したプログラミング言語) に関する基本的な知識が必要です。

開発環境：

- Lambda 関数と拡張機能の構築とデプロイに必要なツールと依存関係を使用して、ローカル開発環境を設定します。

AWS CLI または AWS Management Console：

- AWS Command Line Interface （ AWS CLI ） をインストールして設定するか、適切な認証情報 AWS Management Console で を使用して必要な とやり取りします AWS のサービス。

モニタリングとログ記録：

- Amazon CloudWatch や などのサービスを含む AWS、 でのモニタリングとログ記録のベストプラクティスに精通し、モニタリングと監査 AWS CloudTrail の目的で使用します。
- Lambda 関数の CloudWatch Logs をチェックして、Lambda Telemetry API 統合に関連するエラーや例外を特定します。トラブルシューティングのガイダンスについては、Lambda Telemetry API ドキュメントを参照してください。

アーキテクチャ

このパターンでは、OpenSearch Service を使用して、Lambda 関数によって生成されたログとテレメトリデータを保存します。このアプローチにより、ログを OpenSearch クラスターに直接すばやくストリーミングできるため、CloudWatch Logs を仲介として使用する際のレイテンシーとコストを削減できます。

> **Note**
>
> Lambda 拡張コードは、OpenSearch API を直接使用するか、OpenSearch クライアントライブラリを使用して、テレメトリを OpenSearch Service にプッシュできます。OpenSearch Lambda 拡張機能は、OpenSearch API でサポートされている一括オペレーションを使用してテレメトリイベントをバッチ処理し、1 回のリクエストで OpenSearch Service に送信できます。

次のワークフロー図は、エンドポイントとして OpenSearch クラスターを使用する場合の Lambda 関数のログワークフローを示しています。

アーキテクチャには、以下のコンポーネントが含まれています。

- Lambda 関数: 実行中にログとテレメトリデータを生成するサーバーレス関数。
- Lambda 拡張機能: Lambda Telemetry API を使用して OpenSearch クラスターと直接統合する Python ベースの拡張機能。この拡張機能は、同じ実行環境で Lambda 関数と一緒に実行されます。
- Lambda Telemetry API: Lambda 拡張機能がログ、メトリクス、トレースなどのテレメトリデータをサードパーティーのモニタリングツールやオブザーバビリティツールに直接送信できるようにする API。
- Amazon OpenSearch Service クラスター: ホストされているマネージド OpenSearch クラスター AWS。このクラスターは、Lambda 拡張機能を介して Lambda 関数からストリーミングされたログデータの取り込み、保存、インデックス作成を行います。

ワークフローは、以下のステップで構成されます。

1. Lambda 関数が呼び出され、実行中にログとテレメトリデータが生成されます。
2. Lambda 拡張機能は 関数と一緒に実行され、Lambda Telemetry API を使用してログとテレメトリデータをキャプチャします。
3. Lambda 拡張機能は、OpenSearch Service クラスターとの安全な接続を確立し、ログデータをリアルタイムでストリーミングします。

4. OpenSearch Service クラスターは、Kibana やその他の互換性のあるアプリケーションなどのツールを使用して、ログデータを取り込み、インデックスを作成し、保存して、検索、分析、視覚化できるようにします。

CloudWatch Logs を回避し、ログデータを OpenSearch クラスターに直接送信することで、このソリューションにはいくつかの利点があります。

- リアルタイムログのストリーミングと分析により、トラブルシューティングが迅速になり、オブザーバビリティが向上します。
- CloudWatch Logs に関連するレイテンシーの短縮と潜在的な保持制限。
- Lambda 拡張機能をカスタマイズしたり、特定の出力形式や追加の処理用に独自の拡張機能を作成したりする柔軟性。
- OpenSearch Service の検索、分析、視覚化機能との統合により、ログの分析とモニタリングが可能になります。

エピックセクションでは、Lambda 拡張機能の設定、Lambda 関数の設定、OpenSearch Service クラスターとの統合に関するstep-by-step手順について説明します。セキュリティ上の考慮事項、コスト最適化戦略、ソリューションのモニタリングとトラブルシューティングのヒントについては、「ベストプラクティス」セクションを参照してください。

ツール

AWS サービス

- AWS Lambda はサーバーのプロビジョニングや管理をする必要がなく、コードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- Amazon OpenSearch Service は、 が提供するフルマネージドサービス AWS であり、OpenSearch クラスターをクラウドに簡単にデプロイ、運用、スケーリングできます。
- Lambda 拡張機能は、カスタムコードを一緒に実行することで、Lambda 関数の機能を拡張します。Lambda 拡張機能を使用して、Lambda をさまざまなモニタリング、オブザーバビリティ、セキュリティ、ガバナンスツールと統合できます。
- AWS Lambda Telemetry API を使用すると、拡張機能を使用して、Lambda から直接拡張モニタリングおよびオブザーバビリティデータをキャプチャし、任意の宛先に送信できます。
- AWS CloudFormation は、 AWS リソースのモデル化とセットアップに役立つため、それらのリソースの管理に費やす時間を減らし、アプリケーションに集中する時間を増やすことができます。

コードリポジトリ

- AWS Lambda 拡張機能には、独自の拡張機能の構築を開始するのに役立つ AWS および AWS パートナーからのデモとサンプルプロジェクトが含まれています。
- OpenSearch の Lambda テレメトリ統合の例には、Lambda 関数から OpenSearch クラスターにログを送信する方法を示すサンプル Lambda 拡張機能が用意されています。

その他のツール

- OpenSearch は、大量のデータを取り込む、保存、分析するための強力なプラットフォームを提供するオープンソースの分散検索および分析エンジンです。
- Kibana は、OpenSearch で使用できるオープンソースのデータ可視化および探索ツールです。視覚化と分析の実装はこのパターンの範囲外であることに注意してください。詳細については、Kibana ドキュメントおよびその他のリソースを参照してください。

ベストプラクティス

Lambda Telemetry API を OpenSearch と統合するときは、次のベストプラクティスを考慮してください。

セキュリティとアクセスコントロール

- 安全な通信: Lambda 関数と OpenSearch クラスター間のすべての通信を HTTPS を使用して暗号化します。Lambda 拡張機能と OpenSearch 設定で必要な SSL/TLS 設定を構成します。
- IAM アクセス許可 :
  - 拡張機能は Lambda 関数と同じ実行環境で実行されるため、ファイルシステム、ネットワーク、環境変数などのリソースへの同じレベルのアクセスを継承します。
  - Lambda Telemetry API にアクセスして OpenSearch クラスターにデータを書き込むために必要な最小限の IAM アクセス許可を Lambda 関数に付与します。最小権限の原則を使用して、アクセス許可の範囲を制限します。
- OpenSearch アクセスコントロール: OpenSearch クラスターにきめ細かなアクセスコントロールを実装して、機密データへのアクセスを制限します。OpenSearch では、ユーザー認証、ロールベースのアクセスコントロール、インデックスレベルのアクセス許可などの組み込みのセキュリティ機能を使用します。
- 信頼できる拡張機能: 常に信頼できるソースからのみ拡張機能をインストールします。などの Infrastructure as Code (IaC) ツールを使用して、IAM アクセス許可を含む同じ拡張機能設定を複数

の Lambda 関数にアタッチするプロセス AWS CloudFormation を簡素化します。IaC ツールは、以前に使用された拡張機能とバージョンの監査レコードも提供します。

- 機密データの処理: 拡張機能を構築するときは、機密データのログ記録を避けてください。監査目的でログ記録または保持する前に、ペイロードとメタデータをサニタイズします。

コスト最適化

- モニタリングとアラート: Lambda 関数から OpenSearch に送信されるデータの量を追跡するモニタリングとアラートのメカニズムを設定します。これにより、潜在的なコストオーバーランを特定して対処できます。
- データ保持: OpenSearch の Lambda テレメトリデータに適したデータ保持期間を慎重に検討してください。保持期間が長くなるとストレージコストが増加する可能性があるため、オブザーバビリティのニーズとコスト最適化のバランスを取ります。
- 圧縮とインデックス作成: データ圧縮を有効にし、OpenSearch インデックス作成戦略を最適化して、Lambda テレメトリデータのストレージフットプリントを削減します。
- CloudWatch への依存を減らす: Lambda Telemetry API を OpenSearch と直接統合することで、CloudWatch Logs への依存を減らすことができ、コスト削減につながる可能性があります。これは、Lambda Telemetry API によって OpenSearch に直接ログを送信できるため、CloudWatch にデータを保存して処理する必要がなくなります。

スケーラビリティと信頼性

- 非同期処理: Amazon Simple Queue Service (Amazon SQS) や Amazon Kinesis などの非同期処理パターンを使用して、Lambda 関数の実行を OpenSearch データ取り込みから切り離します。これにより、Lambda 関数の応答性を維持し、システムの全体的な信頼性を向上させることができます。
- OpenSearch クラスターのスケーリング: OpenSearch クラスターのパフォーマンスとリソース使用率をモニタリングし、必要に応じてスケールアップまたはスケールダウンして、Lambda テレメトリデータの増加を処理します。
- フェイルオーバーとディザスタリカバリ: 定期的なバックアップや障害発生時のデータを迅速に復元する機能など、OpenSearch クラスターの堅牢なディザスタリカバリ戦略を実装します。

オブザーバビリティとモニタリング

- ダッシュボードと視覚化: Kibana または他のダッシュボードツールを使用して、OpenSearch のテレメトリデータに基づいて Lambda 関数のパフォーマンスと状態に関するインサイトを提供するカスタムダッシュボードと視覚化を作成します。
- アラートと通知: Lambda 関数の異常、エラー、パフォーマンスの問題を積極的にモニタリングするようにアラートと通知を設定します。これらのアラートと通知を既存のインシデント管理プロセスと統合します。
- トレースと相関: Lambda テレメトリデータにリクエスト IDs や相関 IDs などの関連するトレース情報が含まれていることを確認し、分散サーバーレスアプリケーション全体でend-to-endのオブザーバビリティとトラブルシューティングを可能にします。

これらのベストプラクティスに従うことで、Lambda Telemetry API と OpenSearch の統合の安全性、コスト効率、スケーラビリティを確保し、サーバーレスアプリケーションに包括的なオブザーバビリティを提供できます。

エピック

Lambda 拡張機能レイヤーを構築およびデプロイする

タスク	説明	必要なスキル
ソースコードをダウンロードします。	拡張機能リポジトリからサンプルAWS Lambda 拡張機能をダウンロードします。	アプリ開発者、クラウドアーキテクト
`python-example-telemetry-opensearch-extension` フォルダに移動します。	ダウンロードした AWS Lambda Extensions リポジトリには、いくつかのユースケースと言語ランタイムに関する多数の例が含まれています。python-example-telemetry-opensearch-extension フォルダに移動し、OpenSearch にログを送信します。 OpenSearch	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル
拡張機能エンドポイントを実行するアクセス許可を追加します。	次のコマンドを実行して、拡張エンドポイントを実行可能にします。 `chmod +x python-example-telemetry-opensearch-extension/extension.py`	アプリ開発者、クラウドアーキテクト
拡張機能の依存関係をローカルにインストールします。	次のコマンドを実行して、Python コードのローカル依存関係をインストールします。 `pip3 install -r python-example-telemetry-opensearch-extension/requirements.txt -t ./python-example-telemetry-opensearch-extension/` これらの依存関係は、拡張コードとともにマウントされます。	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル

拡張機能の .zip パッケージを作成して、レイヤーとしてデプロイします。	拡張子 .zip ファイルには`extensions/` 、拡張子実行可能ファイルが配置されている というルートディレクトリと、拡張子のコアロジックとその依存関係が配置されている `python-example-telemetry-opensearch-extension/` という別のルートディレクトリが含まれている必要があります。
	
	拡張機能の .zip パッケージを作成します。
	
	`chmod +x extensions/python-example-telemetry-opensearch-extension`
	`zip -r extension.zip extensions python-example-telemetry-opensearch-extension`
		アプリ開発者、クラウドアーキテクト

拡張機能を Lambda レイヤーとしてデプロイします。	拡張子 .zip ファイルと次のコマンドを使用して Layer を公開します。
	
	`aws lambda publish-layer-version \`
	`--layer-name "python-example-telemetry-opensearch-extension" \`
	`--zip-file "fileb://extension.zip"`
		アプリ開発者、クラウドアーキテクト

拡張機能を関数に統合する

タスク	説明	必要なスキル
レイヤーを関数に追加します。	1. にサインイン AWS Management Console し、AWS Lambda コンソールの関数ページを開きます。 2. 関数を選択します。 3. [Layers] の [Add a layer] を選択します。 4. アンダーレイヤーを選択し、レイヤーソースとしてカスタムレイヤーを選択し、レイヤーを追加します。 Lambda 関数にレイヤーを追加する方法の詳細については、Lambda ドキュメントを参照してください。	アプリ開発者、クラウドアーキテクト
関数の環境変数を設定します。	関数ページで、設定タブを選択し、次の環境変数を関数に追加します。 • URL – ログが送信される OpenSearch エンドポイントの URI。 • AUTH_SECRET – に保存されている OpenSearch 認証情報の ARN AWS Secrets Manager。これはキーと値のペアとして保存し、`username` と の 2 つの	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル
	キーを持つ必要がありますpassword。 • PLATFORM_INDEX 、FUNCTION_INDEX 、 EXTENSION_INDEX - テレメトリデータ、関数ログ、拡張ログを保存するインデックスの名前。それらが適切な命名基準に従っていることを確認してください。そうしないと、インデックスは作成されません。 • DISPATCH_MIN_BATCH_SIZE – バッチ処理するログイベントの数。ただし、関数がシャットダウンすると、この設定に関係なくログがディスパッチされます。	

ログ記録ステートメントを追加し、関数をテストする

タスク	説明	必要なスキル
関数にログ記録ステートメントを追加します。	組み込みのログ記録メカニズムのいずれかまたは任意のログ記録モジュールを使用して、関数にログ記録ステートメントを追加します。 Python でのメッセージのログ記録の例を次に示します。	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル

	`print("Your Log Message Here")`
	`logger = logging.getLogger(__name__)`
	
	`logger.info("Test Info Log.")`
	`logger.error("Test Error Log.")`	

関数をテストします。	1. 関数ページで、[Test] タブを選択します。 2. 関数のテストイベントを作成し、テストを実行します。詳細については、Lambda ドキュメントの「コンソールでの Lambda 関数のテスト」を参照してください。 関数の実行: すべてが正常に機能すると成功するのが表示されます。	アプリ開発者、クラウドアーキテクト

OpenSearch でログを表示する

タスク	説明	必要なスキル
インデックスをクエリします。	OpenSearch で、次のコマンドを実行してインデックスをクエリします。 `SELECT * FROM index-name`	クラウドアーキテクト

タスク	説明	必要なスキル
	クエリ結果にログが表示されます。	

トラブルシューティング

問題	ソリューション
接続の問題	• Lambda 関数に OpenSearch クラスターにアクセスするために必要なネットワーク接続があることを確認します。VPC 設定の設定に関するガイダンスについては、OpenSearch Service のドキュメントを参照してください。 • Lambda 関数に付与された IAM アクセス許可で、OpenSearch クラスターへのデータの書き込みが許可されていることを確認します。IAM アクセス許可の管理については、Lambda ドキュメントを参照してください。
データ取り込みエラー	• Lambda 関数の CloudWatch Logs をチェックして、Lambda Telemetry API 統合に関連するエラーや例外を特定します。トラブルシューティングガイダンスについては、Lambda Telemetry API ドキュメントを参照してください。 • OpenSearch クラスターが正しく設定されており、Lambda テレメトリデータを取り込むために必要なインデックスマッピングと設定があることを確認します。詳細については、OpenSearch のドキュメントを参照してください。

関連リソース

- OpenSearch の Lambda テレメトリ統合の例 (GitHub リポジトリ)
- Lambda 拡張機能を使用した Lambda 関数の拡張 (Lambda ドキュメント)
- Lambda Telemetry API (Lambda ドキュメント)
- AWS Lambda Telemetry API の紹介 (AWS ブログ記事)
- AWS Lambda Telemetry API と Prometheus および OpenSearch の統合 (AWS ブログ記事)

追加情報

ログ構造の変更

拡張機能は、デフォルトでネストされたドキュメントとしてログを OpenSearch に送信します。これにより、ネストされたクエリを実行して個々の列値を取得できます。

デフォルトのログ出力が特定のニーズを満たさない場合は、 が提供する Lambda 拡張機能のソースコードを変更してカスタマイズできます AWS。 AWS は、出力をビジネス要件に合わせて調整するようお客様にお勧めします。ログ出力を変更するには、拡張機能のソースコード内の `telemetry_dispatcher.py` ファイルで `dispatch_to_opensearch`関数を見つけ、必要な変更を行います。

セルベースのアーキテクチャ用にサーバーレスセルルーターを設定する

作成者: Mian Tariq (AWS) と Ioannis Lioupras (AWS)

概要

グローバルセルベースのアプリケーションのシステムへのエントリポイントとして、セルルーターはユーザーを適切なセルに効率的に割り当て、ユーザーにエンドポイントを提供する責任があります。セルルーターは、user-to-cellマッピングの保存、セル容量のモニタリング、必要に応じて新しいセルのリクエストなどの機能を処理します。中断の可能性がある間は、セルルーター機能を維持することが重要です。

このパターンのセルルーター設計フレームワークは、回復力、スケーラビリティ、全体的なパフォーマンスの最適化に焦点を当てています。このパターンでは、クライアントが初回ログイン時にエンドポイントをキャッシュし、セルと直接通信する静的ルーティングを使用します。このデカップリングにより、セルルーターの障害発生時にセルベースのアプリケーションの中断のない機能を確保できるため、システムの耐障害性が向上します。

このパターンでは、 AWS CloudFormation テンプレートを使用してアーキテクチャをデプロイします。テンプレートのデプロイの詳細、または を使用して同じ設定をデプロイするには AWS Management Console、「追加情報」セクションを参照してください。

> **Important**
>
> このパターンで示されているデモンストレーション、コード、 AWS CloudFormation テンプレートは、説明のみを目的としています。提供されるマテリアルは、設計パターンを解説し、理解を支援する目的でのみ提供されています。デモとコードは本番稼働用ではないため、本番稼働用のアクティビティには使用しないでください。本番環境でコードまたはデモを使用しようとすることは強く推奨されず、お客様の責任となります。このパターンまたはそのコンポーネントを本番環境に実装する前に、適切な専門家に相談し、徹底的なテストを実行することをお勧めします。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント
- AWS Command Line Interface （ AWS CLI) の最新バージョン

- AWS CloudFormation スタック、 AWS Lambda 関数、および関連リソースの作成に必要なアクセス許可を持つ AWS 認証情報

製品バージョン

- Python 3.12

アーキテクチャ

次の図は、セルルーターの大まかな設計を示しています。

図は、以下のワークフローを順を追って示しています。

1. ユーザーは、セルルーター API エンドポイントのフロントとして機能する Amazon API Gateway に連絡します。
2. Amazon Cognito は認証と認可を処理します。
3. AWS Step Functions ワークフローは以下のコンポーネントで構成されます。
   - オーケストレーター – は AWS Step Functions `Orchestrator`を使用してワークフローまたはステートマシンを作成します。ワークフローはセルルーター API によってトリガーされます。は、リソースパスに基づいて Lambda 関数`Orchestrator`を実行します。
   - ディスパッチャー `Dispatcher`– Lambda 関数は、登録された新しいユーザーごとに 1 つの静的セルを識別して割り当てます。この関数は、ユーザー数が最も少ないセルを検索し、ユーザーに割り当てることで、エンドポイントを返します。
   - マッパー – `Mapper`オペレーションは、 AWS CloudFormation テンプレートによって作成された `RoutingDB` Amazon DynamoDB データベース内のuser-to-cell間のマッピングを処理します。トリガーされると、`Mapper`関数は、既に割り当てられたユーザーにエンドポイントを提供します。
   - Scaler – `Scaler`関数は、セルの占有率と使用可能な容量を追跡します。必要に応じて、`Scaler`関数は Amazon Simple Queue Service (Amazon SQS) を介して Provision and Deploy レイヤーにリクエストを送信して、新しいセルをリクエストできます。
   - Validator – `Validator`関数はセルエンドポイントを検証し、潜在的な問題を検出します。
4. は、セル情報と属性 (API エンドポイント、状態 AWS リージョン、メトリクス) `RoutingDB`を保存します。

5. セルの使用可能な容量がしきい値を超えると、セルルーターは Amazon SQS を介してプロビジョニングおよびデプロイサービスをリクエストして新しいセルを作成します。

新しいセルが作成されると、 RoutingDBはプロビジョニングレイヤーとデプロイレイヤーから更新されます。ただし、このプロセスはこのパターンの範囲外です。セルベースのアーキテクチャ設計の前提の概要と、このパターンで使用されるセルルーター設計の詳細については、「追加情報」セクションを参照してください。

ツール

AWS のサービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- Amazon Simple Queue Service (Amazon SQS) は、分散ソフトウェアシステムとコンポーネントの統合と分離に役立つ、安全で耐久性があり、利用可能なホスト型キューを提供します。
- AWS Step Functions は、Lambda 関数やその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub Serverless-Cell-Router リポジトリで入手できます。

ベストプラクティス

セルベースのアーキテクチャを構築する際のベストプラクティスについては、以下の AWS Well-Architected ガイダンスを参照してください。

- セルベースのアーキテクチャによる影響範囲の縮小
- AWS Well-Architected フレームワークの信頼性の柱: REL10-BP04 バルクヘッドアーキテクチャを使用して影響範囲を制限する

エピック

ソースファイルの準備

タスク	説明	必要なスキル
サンプルコードリポジトリのクローンを作成します。	Serverless-Cell-Router GitHub リポジトリをコンピュータにクローンするには、次のコマンドを使用します。 `git clone https://github.com/aws-samples/Serverless-Cell-Router/`	開発者
AWS CLI 一時的な認証情報を設定します。	の認証情報 AWS CLI を使用して を設定します AWS アカウント。このチュートリアルでは、 AWS IAM Identity Center コマンドラインまたはプログラムによるアクセスオプションによって提供される一時的な認証情報を使用します。これによりAWS_ACCES	開発者

タスク	説明	必要なスキル
	S_KEY_ID 、、AWS_SECRET_ACCESS_KEY 、および AWS_SESSION_TOKEN AWS 環境変数が、 で使用する適切な認証情報で設定されます AWS CLI。	
S3 バケットを作成する。	AWS CloudFormation テンプレートによるデプロイのために Serverless-Cell-Router Lambda 関数を保存してアクセスするために使用される S3 バケットを作成します。S3 バケットを作成するには、次のコマンドを使用します。 `aws s3api create-bucket --bucket <bucket name> --region eu-central-1 --create-bucket-configuration LocationConstraint=eu-central-1`	開発者

タスク	説明	必要なスキル

.zip ファイルを作成します。	Functions ディレクトリにある Lambda 関数ごとに 1 つの .zip ファイルを作成します。これらの .zip ファイルは、Lambda 関数のデプロイに使用されます。Mac では、次のzipコマンドを使用します。
	
	`zip -j mapper-scr.zip Functions/Mapper.py`
	`zip -j dispatcher-scr.zip Functions/Dispatcher.py`
	`zip -j scaler-scr.zip Functions/Scaler.py`
	`zip -j cp validator-scr.zip Functions/Validator.py`
	`zip -j dynamodbDummyData-scr.zip Functions/DynamodbDummyData.py`
		開発者

タスク	説明	必要なスキル

.zip ファイルを S3 バケットにコピーします。	すべての Lambda 関数の .zip ファイルを S3 バケットにコピーするには、次のコマンドを使用します。
	
	`aws s3 cp mapper-scr.zip s3://<bucket name>`
	`aws s3 cp dispatcher-scr.zip s3://<bucket name>`
	`aws s3 cp scaler-scr.zip s3://<bucket name>`
	`aws s3 cp validator-scr.zip s3://<bucket name>`
	`aws s3 cp dynamodbDummyData-scr.zip s3://<bucket name>`
		開発者

AWS CloudFormation スタックを作成する

タスク	説明	必要なスキル

AWS CloudFormation テンプレートをデプロイします。	AWS CloudFormation テンプレートをデプロイするには、次の AWS CLI コマンドを実行します。
	
	`aws cloudformation create-stack --stack-name serverless.cell-router \`
	`--template-body file://Serverless-Cell-Router-Stack-v10.yaml \`
		開発者

タスク	説明	必要なスキル
	--capabilities CAPABILITY_IAM \ --parameters ParameterKey=LambdaFunctionMapperS3KeyParameterSCR,ParameterValue=mapper-scr.zip \ ParameterKey=LambdaFunctionDispatcherS3KeyParameterSCR,ParameterValue=dispatcher-scr.zip \ ParameterKey=LambdaFunctionScalerS3KeyParameterSCR,ParameterValue=scaler-scr.zip \ ParameterKey=LambdaFunctionAddDynamoDBDummyItemsS3KeyParameterSCR,ParameterValue=dynamodbDummyData-scr.zip \ ParameterKey=LambdaFunctionsS3BucketParameterSCR,ParameterValue=<S3 bucket storing lambda zip files> \ ParameterKey=CognitoDomain,ParameterValue=<Cognito Domain Name> \ --region <enter your aws region id, e.g. "eu-central-1">	

タスク	説明	必要なスキル
進捗確認。	にサインインし AWS Management Console、 AWS CloudFormation コンソールをhttps://console.aws.amazon.com/cloudformation/://www.」で開き、スタック開発の進行状況を確認します。ステータスが CREATE_COMPLETE の場合、スタックは正常にデプロイされています。	開発者

評価と検証

タスク	説明	必要なスキル
ユーザーにセルを割り当てます。	を開始するにはOrchestrator 、次の curl コマンドを実行します。 `curl -X POST \ -H "Authorization: Bearer {User id_token}" \ https://xxxxxx.execute-api.eu-central-1.amazonaws.com/Cell_Router_Development/cells` は関数の実行をOrchestrator トリガーしますDispatcher 。次にDispatcher 、 はユーザーの存在を検証します。ユーザーが見つかった場	開発者

タスク	説明	必要なスキル
	合、 は関連付けられたセル ID とエンドポイント URLsDispatcher を返します。ユーザーが見つからない場合、 はユーザーにセルをDispatcher 割り当て、割り当てられたセルの残容量を評価するためにセル ID を Scaler関数に送信します。 Scaler 関数のレスポンスは次のとおりです。 `"cellID : cell-0002 , endPoint_1 : https://xxxxx.execute-api.eu-north-1 .amazonaws.com/ , endPoint_2 : https://xxxxxxx.execute-api .eu-central-1.amaz onaws.com/"`	

タスク	説明	必要なスキル

ユーザーセルを取得します。	を使用して Mapper関数Orchestrator を実行するには、次のコマンドを実行します。
	
	`curl -X POST \ -H "Authorization: Bearer {User id_token}" \ https://xxxxxxxxx.execute-api.eu-central-1.amazonaws.com/Cell_Router_Development/mapper`
	
	は、ユーザーに割り当てられたセルOrchestrator を検索し、次のレスポンスでセル ID と URLsを返します。
	
	`"cellID : cell-0002 , endPoint_1 : https://xxxxx.execute-api.eu-north-1.amazonaws.com/ , endPoint_2 : https://xxxxxxx.execute-api.eu-central-1.amazonaws.com/"`
		開発者

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップします。	アカウントに追加料金が発生しないようにするには、次の操作を行います。 1. Lambda 関数用に作成した S3 バケットを空にします。 2. バケットを削除します。 3. AWS CloudFormation スタックを削除します。	アプリ開発者

関連リソース

リファレンス

- アベイラビリティーゾーンを使用した静的安定性
- AWS 障害分離境界: 静的安定性

動画

Physalia: Amazon EBS での可用性を高めるセルベースのアーキテクチャ

https://www.youtube-nocookie.com/embed/6IknqRZMFic?controls=0

追加情報

セルベースのアーキテクチャ設計の施設

このパターンはセルルーターに焦点を当てていますが、環境全体を理解することが重要です。環境は 3 つの個別のレイヤーで構成されています。

- セルルーターを含むルーティングレイヤー、またはシンレイヤー
- さまざまなセルで構成されるセルレイヤー
- セルをプロビジョニングしてアプリケーションをデプロイする Provision and Deploy Layer

各レイヤーは、他のレイヤーに影響を与える障害が発生した場合でも機能を維持します。 AWS アカウント は障害分離の境界として機能します。

次の図は、レイヤーの概要を示しています。セルレイヤーとプロビジョニングおよびデプロイレイヤーは、このパターンの範囲外です。

セルベースのアーキテクチャの詳細については、「セルベースのアーキテクチャによる影響範囲の縮小: セルルーティング」を参照してください。

セルルーターの設計パターン

セルルーターはセル間で共有されるコンポーネントです。潜在的な影響を軽減するには、Routing Layer ができるだけ薄く、シンプルで水平方向にスケーラブルな設計を使用することが重要です。ルーティングレイヤーは、システムのエントリポイントとして機能し、ユーザーを適切なセルに効率的に割り当てするために必要なコンポーネントのみで構成されます。このレイヤー内のコンポーネントは、セルの管理や作成には関与しません。

このパターンでは静的ルーティングを使用します。つまり、クライアントは初回ログイン時にエンドポイントをキャッシュし、その後セルとの直接通信を確立します。クライアントとセルルーター間の定期的なやり取りが開始され、現在のステータスを確認したり、更新を取得したりできます。この意図的なデカップリングにより、セルルーターのダウンタイムが発生した場合に既存のユーザーの操作を中断することなく実行でき、システム内で継続的な機能と回復力を提供します。

このパターンでは、セルルーターは次の機能をサポートしています。

- プロビジョニングレイヤーとデプロイレイヤーのセルデータベースからセルデータを取得し、ローカルデータベースを保存または更新します。
- セル割り当てアルゴリズムを使用して、アプリケーションの新しい登録済みユーザーごとにセルを割り当てます。
- user-to-cells間のマッピングをローカルデータベースに保存します。
- ユーザーの割り当て中にセルの容量を確認し、自動販売機のイベントをプロビジョニングおよびデプロイレイヤーに引き上げてセルを作成します。
- セル作成基準アルゴリズムを使用してこの機能を提供します。
- 静的セルの URLs を指定して、新しく登録されたユーザーリクエストに応答します。これらの URLsは、有効期限 (TTL) を使用してクライアントにキャッシュされます。
- 新規または更新された URL を指定して、無効な URL の既存のユーザーリクエストに応答します。

AWS CloudFormation テンプレートによってセットアップされるデモセルルーターをさらに理解するには、以下のコンポーネントとステップを確認してください。

1. Amazon Cognito ユーザープールを設定および設定します。
2. セルルーターの API Gateway API をセットアップおよび設定します。
3. DynamoDB テーブルを作成します。
4. SQS キューを作成して設定します。
5. を実装しますOrchestrator。
6. Lambda 関数を実装します: Dispatcher、Scaler、Mapper、Validator。
7. 評価して検証します。

前提は、プロビジョニングレイヤーとデプロイレイヤーが既に確立されていることです。実装の詳細は、このアーティファクトの範囲外です。

これらのコンポーネントは AWS CloudFormation テンプレートによって設定および設定されるため、次の手順は説明的で高レベルで表示されます。セットアップと設定を完了するために必要な AWS スキルがあることを前提としています。

1. Amazon Cognito ユーザープールのセットアップと設定

にサインインし AWS Management Console、Amazon Cognito コンソールを https://console.aws.amazon.com/cognito/:///https://https://https://https://https://https://https://https://https://https://https://https://http アプリケーション統合、ホストされた UICellRouterPool、および必要なアクセス許可を使用して、 という名前の Amazon Cognito ユーザープールを設定および設定します。

2。セルルーターの API Gateway API をセットアップおよび設定する

API Gateway コンソール (「https://console.aws.amazon.com/apigateway」) を開きます。Amazon Cognito ユーザープール と統合された Amazon Cognito オーソライザーを使用してCellRouter、という名前の Amazon Cognitoを設定および設定しますCellRouterPool。次の要素を実装します。

- CellRouter POSTメソッドを含む API リソース
- ステップ 5 で実装された Step Functions ワークフローとの統合
- Amazon Cognito オーソライザーによる認可
- 統合リクエストとレスポンスのマッピング

- 必要なアクセス許可の割り当て

3。DynamoDB テーブルを作成する

https://console.aws.amazon.com/dynamodb/://」で DynamoDB コンソールを開き、次の設定tbl_routerで という名前の標準 DynamoDB テーブルを作成します。

- パーティションキー marketId–
- ソートキー cellId–
- キャパシティモード – プロビジョニング済み
- Point-in-timeリカバリ (PITR) – オフ

インデックスタブで、 というグローバルセカンダリインデックスを作成しますmarketId-currentCapacity-index。Scaler Lambda 関数は、インデックスを使用して、割り当てられたユーザーの数が最も少ないセルを効率的に検索します。

次の属性を使用してテーブル構造を作成します。

- marketId – 欧州
- cellId – cell-0002
- currentCapacity – 2
- endPoint_1 – <最初のリージョンのエンドポイント >
- endPoint_2 – < 2 番目のリージョンのエンドポイント >
- IsHealthy – True
- maxCapacity – 10
- regionCode_1 – eu-north-1
- regionCode_2 – eu-central-1
- userIds – < E メールアドレス >

4。SQS キューの作成と設定

https://console.aws.amazon.com/sqs/「https://」で Amazon SQS コンソールを開き、Amazon SQS キー暗号化でCellProvisioning設定された という名前の標準 SQS キューを作成します。Amazon SQS

5。オーケストレーターの実装

ルーターの として機能する Step Functions ワークフローを開発Orchestratorします。ワークフローは、セルルーター API を介して呼び出すことができます。ワークフローは、リソースパスに基づいて指定された Lambda 関数を実行します。ステップ関数をセルルーター の API Gateway API と統合しCellRouter、Lambda 関数を呼び出すために必要なアクセス許可を設定します。

以下の図に、ワークフローを示しています。選択状態は、Lambda 関数の 1 つを呼び出します。Lambda 関数が成功すると、ワークフローは終了します。Lambda 関数が失敗すると、フェイルステートが呼び出されます。

6。Lambda 関数の実装

Dispatcher、Mapper、Scaler、および Validator関数を実装します。デモンストレーションで各関数をセットアップして設定するときは、関数のロールを定義し、DynamoDB テーブル で必要なオペレーションを実行するために必要なアクセス許可を割り当てますtbl_router。さらに、各関数をワークフロー に統合しますOrchestrator。

ディスパッチャー関数

Dispatcher 関数は、新しい登録ユーザーごとに 1 つの静的セルを識別して割り当てる責任があります。新しいユーザーがグローバルアプリケーションに登録すると、リクエストは Dispatcher関数に送信されます。関数は、次のような事前定義された評価基準を使用してリクエストを処理します。

1. リージョン - ユーザーがいるマーケットのセルを選択します。たとえば、ユーザーが欧州からグローバルアプリケーションにアクセスする場合は、欧州 AWS リージョン で を使用するセルを選択します。
2. 近接性またはレイテンシー - ユーザーに最も近いセルを選択します。たとえば、ユーザーがオランダからアプリケーションにアクセスする場合、関数はフランクフルトとアイルランドを使用するセルを考慮します。どのセルが最も近いかの決定は、ユーザーの場所とセルリージョン間のレイテンシーなどのメトリクスに基づきます。このパターン例では、情報はプロビジョニングレイヤーとデプロイレイヤーから静的にフィードされます。
3. Health – Dispatcher関数は、指定されたセルの状態 (正常 = true または false) に基づいて、選択したセルが正常かどうかを確認します。
4. 容量 - ユーザーディストリビューションはセルロジック内の最小ユーザー数に基づいているため、ユーザーは最小ユーザー数を持つセルに割り当てられます。

> **Note**
>
> これらの基準は、このパターン例を説明するためにのみ提示されます。実際のセルルーター実装では、より洗練されたユースケースベースの基準を定義できます。

は Dispatcher 関数を`Orchestrator`呼び出して、ユーザーをセルに割り当てます。このデモ関数では、市場値は として定義された静的パラメータです`europe`。

この`Dispatcher`関数は、セルが既にユーザーに割り当てられているかどうかを評価します。セルがすでに割り当てられている場合、`Dispatcher`関数はセルのエンドポイントを返します。ユーザーにセルが割り当てられていない場合、関数はユーザー数が最も少ないセルを検索し、ユーザーに割り当てて、エンドポイントを返します。セル検索クエリの効率は、グローバルセカンダリインデックスを使用して最適化されます。

マッパー関数

この`Mapper`関数は、データベース内のuser-to-cell間のマッピングの保存とメンテナンスを監督します。登録された各ユーザーには、単一のセルが割り当てられます。各セルには 2 つの異なる URLs があり、AWS リージョンごとに 1 つずつあります。API Gateway でホストされている API エンドポイントとして機能するこれらの URLs、グローバルアプリケーションへのインバウンドポイントとして機能します。

`Mapper` 関数は、クライアントアプリケーションからリクエストを受信すると、DynamoDB テーブルでクエリを実行して、指定された E メール ID に関連付けられたuser-to-cellのマッピング`tbl_router`を取得します。割り当てられたセルが見つかった場合、`Mapper`関数はセルの 2 つの URLs を迅速に提供します。また、この`Mapper`関数はセル URLs の変更をアクティブにモニタリングし、ユーザー設定の通知または更新を開始します。

Scaler 関数

`Scaler` 関数はセルの残容量を管理します。関数は、新しいユーザー登録リクエストごとに、`Scaler`関数がユーザーに`Dispatcher`割り当てたセルの使用可能な容量を評価します。指定された評価基準に従ってセルが事前に定義された制限に達した場合、関数は Amazon SQS キューを介してプロビジョニングレイヤーとデプロイレイヤーへのリクエストを開始し、新しいセルのプロビジョニングとデプロイを要求します。セルのスケーリングは、次のような一連の評価基準に基づいて実行できます。

1. 最大ユーザー数 - 各セルには最大 500 人のユーザーを設定できます。

2. バッファ容量 - 各セルのバッファ容量は 20% です。つまり、各セルはいつでも 400 ユーザーに割り当てることができます。残りの 20% のバッファ容量は、将来のユースケースや予期しないシナリオ (セルの作成やプロビジョニングサービスが利用できない場合など) の処理用に予約されています。
3. セルの作成 - 既存のセルが容量の 70% に達するとすぐに、追加のセルを作成するリクエストがトリガーされます。

> **Note**
>
> これらの基準は、このパターン例を説明するためにのみ提示されます。実際のセルルーター実装では、より洗練されたユースケースベースの基準を定義できます。

デモScalerコードは、 が新しく登録されたユーザーにセルをDispatcher正常に割り当てOrchestratorた後、 によって実行されます。はScaler、 からセル ID を受け取るとDispatcher、事前定義された評価基準に基づいて、指定されたセルに追加のユーザーに対応するのに十分な容量があるかどうかを評価します。セルの容量が不十分な場合、Scaler関数は Amazon SQS サービスにメッセージをディスパッチします。このメッセージは、プロビジョニングレイヤーとデプロイレイヤー内のサービスによって取得され、新しいセルのプロビジョニングを開始します。

バリデータ関数

Validator 関数は、セルアクセスに関連する問題を特定して解決します。ユーザーがグローバルアプリケーションにサインインすると、アプリケーションはユーザープロファイル設定からセルの URLs を取得し、ユーザーリクエストをセル内の 2 つの割り当てられたリージョンのいずれかにルーティングします。URLs にアクセスできない場合、アプリケーションは検証 URL リクエストをセルルーターにディスパッチできます。セルルーターは をOrchestrator呼び出しますValidator。は検証プロセスValidatorを開始します。検証には、次のようなチェックが含まれる場合があります。

- リクエスト内のセル URLs をデータベースに保存されている URLs と相互参照して、潜在的な更新を特定して処理する
- ディープヘルスチェックの実行 (セルのエンドポイントへのHTTP GETリクエストなど)

結論として、 Validator関数はクライアントアプリケーションリクエストにレスポンスを配信し、必要な修復ステップとともに検証ステータスを提供します。

Validator は、ユーザーエクスペリエンスを向上させるように設計されています。インシデントによりセルが一時的に使用できなくなるため、特定のユーザーがグローバルアプリケーションにアクセスできないシナリオを考えてみましょう。Validator 関数は、一般的な工ラーを提示する代わりに、指示的な修復ステップを提供できます。これらのステップには、以下のアクションが含まれる場合があります。

- インシデントについてユーザーに通知します。
- サービスが使用可能になるまでのおおよその待機時間を指定します。
- 追加情報を取得するためのサポート連絡先番号を入力します。

Validator 関数のデモコードは、リクエスト内のユーザーが指定したセル URLs がtbl_routerテーブルに保存されたレコードと一致することを確認します。このValidator関数は、セルが正常かどうかも確認します。

VPC エンドポイントを介して Amazon S3 バケットへのプライベートアクセスを設定する

作成者: Martin Maritsch (AWS)、Gabriel Rodriguez Garcia (AWS)、Shukhrat Khodjaev (AWS)、Nicolas Jacob Baer (AWS)、Mohan Gowda Purushothama (AWS)、Joaquin Rinaudo (AWS)

概要

Amazon Simple Storage Service (Amazon S3) では、署名付き URLs を使用して、任意のサイズのファイルをターゲットユーザーと共有できます。デフォルトでは、Amazon S3 の署名付き URLs は有効期限内にインターネットからアクセスできるため、簡単に使用できます。ただし、企業環境では、多くの場合、Amazon S3 の署名付き URLs へのアクセスをプライベートネットワークのみに制限する必要があります。

このパターンは、インターネットトラバーサルなしでプライベートネットワークからの署名付き URLs を使用して S3 オブジェクトを安全に操作するためのサーバーレスソリューションを示しています。アーキテクチャでは、ユーザーは内部ドメイン名を介して Application Load Balancer にアクセスします。トラフィックは、Amazon API Gateway と S3 バケットの仮想プライベートクラウド (VPC) エンドポイントを介して内部的にルーティングされます。この AWS Lambda 関数は、プライベート VPC エンドポイントを介してファイルをダウンロードするための署名URLs を生成します。これにより、機密データのセキュリティとプライバシーが向上します。

前提条件と制限

前提条件

- 企業ネットワーク AWS アカウント に接続されている にデプロイされたサブネットを含む VPC (経由など AWS Direct Connect) 。

制約事項

- S3 バケットの名前はドメインと同じである必要があります。そのため、Amazon S3 バケットの命名規則を確認することをお勧めします。
- このサンプルアーキテクチャには、デプロイされたインフラストラクチャのモニタリング機能は含まれていません。ユースケースでモニタリングが必要な場合は、AWS モニタリングサービスの追加を検討してください。
- このサンプルアーキテクチャには、入力検証は含まれません。ユースケースで入力の検証とセキュリティの強化が必要な場合は、 AWS WAF を使用して API を保護することを検討してください。

- このサンプルアーキテクチャには、Application Load Balancer によるアクセスログ記録は含まれません。ユースケースでアクセスログ記録が必要な場合は、ロードバランサーのアクセスログを有効にすることを検討してください。

バージョン

- Python バージョン 3.11 以降
- Terraform バージョン 1.6 以降

アーキテクチャ

ターゲットテクノロジースタック

ターゲットテクノロジースタックでは、次の AWS サービスが使用されます。

- Amazon S3 は、ファイルを安全にアップロード、ダウンロード、保存するためのコアストレージサービスです。
- Amazon API Gateway は、S3 バケットを操作するためのリソースとエンドポイントを公開します。このサービスは、データをダウンロードまたはアップロードするための署名付き URLs を生成する役割を果たします。
- AWS Lambda URLs Amazon S3を生成します。Lambda 関数は API Gateway によって呼び出されます。
- Amazon VPC は VPC 内にリソースをデプロイして、ネットワークを分離します。VPC には、トラフィックフローを制御するためのサブネットとルーティングテーブルが含まれています。
- Application Load Balancer は、受信トラフィックを API Gateway または S3 バケットの VPC エンドポイントにルーティングします。これにより、企業ネットワークのユーザーは内部で リソースにアクセスできます。
- Amazon S3 の VPC エンドポイントを使用すると、パブリックインターネットを経由することなく、VPC 内のリソースと Amazon S3 間の直接のプライベート通信が可能になります。
- AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを制御します。アクセス許可は、API およびその他の サービスとの安全なやり取りを確保するために設定されます。

ターゲット アーキテクチャ

この図表は、以下を示すものです：

1. 社内ネットワークのユーザーは、内部ドメイン名を使用して Application Load Balancer にアクセスできます。企業ネットワークと のイントラネットサブネットの間に接続が存在することを前提としています AWS アカウント （ AWS Direct Connect 接続経由など ）。
2. Application Load Balancer は、受信トラフィックを API Gateway にルーティングして、Amazon S3 または S3 バケットの VPC エンドポイントにデータをダウンロードまたはアップロードするための署名付き URLs を生成します。どちらのシナリオでも、リクエストは内部でルーティングされるため、インターネットを経由する必要はありません。
3. API Gateway は、S3 バケットを操作するためのリソースとエンドポイントを公開します。この例では、S3 バケットからファイルをダウンロードするためのエンドポイントを提供しますが、アップロード機能を提供するために拡張することもできます。
4. Lambda 関数は、パブリック Amazon S3 ドメインの代わりに Application Load Balancer のドメイン名を使用してAmazon S3からファイルをダウンロードするための署名付き URL を生成します。
5. ユーザーは署名付き URL を受け取り、それを使用して Application Load Balancer を使用して Amazon S3 からファイルをダウンロードします。ロードバランサーには、API を意図したものではないトラフィックを S3 バケットの VPC エンドポイントに送信するデフォルトルートが含まれています。
6. VPC エンドポイントは、署名付き URL をカスタムドメイン名で S3 バケットにルーティングします。S3 バケットの名前はドメインと同じである必要があります。

自動化とスケール

このパターンでは、Terraform を使用してコードリポジトリから にインフラストラクチャをデプロイします AWS アカウント。

ツール

ツール

- 「Python」は汎用のコンピュータープログラミング言語です。
- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドを通じて AWS サービスとやり取りするのに役立つオープンソースツールです。

コードリポジトリ

このパターンのコードは、GitHub リポジトリの https://github.com/aws-samples/private-s3-vpce:// https://https://https://https://https://https://https://https://https://https://https://https://https

ベストプラクティス

このパターンのサンプルアーキテクチャでは、IAM アクセス許可を使用して API へのアクセスを制御します。有効な IAM 認証情報を持つユーザーは誰でも API を呼び出すことができます。ユースケースでより複雑な認可モデルが必要な場合は、別のアクセスコントロールメカニズムを使用することをお勧めします。

エピック

ソリューションを にデプロイする AWS アカウント

タスク	説明	必要なスキル
AWS 認証情報を取得します。	AWS 認証情報とアカウントへのアクセスを確認します。手順については、 AWS CLI ドキュメントの「設定と認証情報ファイルの設定」を参照してください。	AWS DevOps、AWS 全般
リポジトリをクローン作成します。	このパターンで提供されている GitHub リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/private-s3-vpce`	AWS DevOps、AWS 全般
変数を設定します。	1. コンピュータの GitHub リポジトリで、 `terraform` フォルダを開きます。 `cd terraform`	AWS DevOps、AWS 全般

タスク	説明	必要なスキル
	2. `example.tfvars` ファイルを開き、必要に応じてパラメータをカスタマイズします。	
ソリューションをデプロイします。	1. `terraform` フォルダで Terraform を実行し、カスタマイズした変数を渡します。 `terraform apply -var-file="example.tfvars"` 2. アーキテクチャ図に示されているリソースが正常にデプロイされたことを確認します。	AWS DevOps、AWS 全般

ソリューションをテストする

タスク	説明	必要なスキル
テストファイルを作成します。	Amazon S3 にファイルをアップロードして、ファイルのダウンロードのテストシナリオを作成します。Amazon S3 コンソールまたは次の AWS CLI コマンドを使用できます。 `aws s3 cp /path/to/testfile s3://your-bucket-name/testfile`	AWS DevOps、AWS 全般
署名付き URL 機能をテストします。	1. Application Load Balancer にリクエストを送信し	AWS DevOps、AWS 全般

タスク	説明	必要なスキル
	て、awscurl を使用してテストファイルの署名付き URL を作成します。 `awscurl https://your-domain-name/api/get_url?key=testfile` このステップでは、認証情報から有効な署名を作成し、API Gateway によって検証されます。 2. 前のステップで受け取ったレスポンスのリンクを解析し、署名付き URL を開いてファイルをダウンロードします。	
クリーンアップを行います。	不要になったリソースは必ず削除してください。 `terraform destroy`	AWS DevOps、AWS 全般

トラブルシューティング

問題	ソリューション
S3 オブジェクトキー名。数字記号 (#) ブレーク URL パラメータなどの特殊文字が含まれ、エラーが発生します。	URL パラメータを適切にエンコードし、S3 オブジェクトキー名が Amazon S3 ガイドラインに従っていることを確認します。

関連リソース

Amazon S3:

- 署名付き URLsを使用したオブジェクトの共有
- バケットポリシーを使用した VPC エンドポイントからのアクセスの制御

Amazon API Gateway:

- API Gateway のプライベート APIs に VPC エンドポイントポリシーを使用する

Application Load Balancer:

- ALB、S3、および PrivateLink を使用した内部 HTTPS 静的ウェブサイトのホスティング (AWS ブログ記事)

Amazon Bedrock AWS Step Functions を使用して の状態をトラブルシューティングする

作成者: Aniket Kurzadkar (AWS) と Sangam Kushwaha (AWS)

概要

AWS Step Functions エラー処理機能は、ワークフロー内の状態中に発生するエラーを確認するのに役立ちますが、エラーの根本原因を見つけてデバッグするのは難しい場合があります。このパターンは、この課題に対処し、Step Functions の状態中に発生するエラーを Amazon Bedrock がどのように解決できるかを示します。

Step Functions はワークフローオーケストレーションを提供するため、デベロッパーはプロセスを簡単に自動化できます。Step Functions には、次の利点を提供するエラー処理機能もあります。

- 開発者は、問題が発生したときに完全には失敗しない、より回復力の高いアプリケーションを作成できます。
- ワークフローには、さまざまなタイプのエラーを異なる方法で処理するための条件ロジックを含めることができます。
- システムは、エクスポネンシャルバックオフを使用して、失敗したオペレーションを自動的に再試行できます。
- エラーシナリオには代替の実行パスを定義できるため、ワークフローは処理を適応させて続行できます。

Step Functions ワークフローでエラーが発生すると、このパターンは、Step Functions でサポートされている Claude 3 などの基盤モデル (FM) にエラーメッセージとコンテキストを送信する方法を示しています。FM はエラーを分析し、分類し、潜在的な修復ステップを提案できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Step Functions とワークフローの基本的な理解
- Amazon Bedrock API 接続

制約事項

- このパターンのアプローチは、さまざまな に使用できます AWS のサービス。ただし、結果は、Amazon Bedrock によって評価 AWS Lambda される によって作成されたプロンプトによって異なる場合があります。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

この図は、Step Functions ステートマシンでのエラー処理と通知の自動ワークフローを示しています。

1. 開発者はステートマシンの実行を開始します。
2. Step Functions ステートマシンは状態の処理を開始します。考えられる結果は 2 つあります。
   - (a) すべての状態が正常に実行されると、ワークフローは Amazon SNS に直接進み、E メールの成功通知が送信されます。
   - (b) いずれかの状態が失敗すると、ワークフローは Lambda 関数の処理エラーに移動します。
3. エラーが発生した場合は、以下が発生します。
   - (a) Lambda 関数 (エラーハンドラー) がトリガーされます。Lambda 関数は、Step Functions ステートマシンが渡したイベントデータからエラーメッセージを抽出します。次に、Lambda 関数はこのエラーメッセージに基づいてプロンプトを準備し、プロンプトを Amazon Bedrock に送信します。プロンプトは、発生した特定のエラーに関連する解決策と提案をリクエストします。
   - (b) 生成 AI モデルをホストする Amazon Bedrock は、入力プロンプトを処理します。(このパターンでは、Amazon Bedrock がサポートする多くの FM の 1 つである Anthropic Claude 3 基盤モデル (FMs) を使用します) 。AI モデルはエラーコンテキストを分析します。次に、モデルがレスポンスを生成します。これには、エラーが発生した理由の説明、エラーを解決するための潜在的な解決策、および将来同じ間違いを起こさないようにするための提案が含まれます。

Amazon Bedrock は、AI によって生成されたレスポンスを Lambda 関数に返します。Lambda 関数はレスポンスを処理し、フォーマットしたり、キー情報を抽出したりする可能性があります。次に、Lambda 関数はステートマシン出力にレスポンスを送信します。

4. エラー処理または実行が成功すると、ワークフローは Amazon SNS をトリガーして E メール通知を送信することで終了します。

ツール

AWS のサービス

- Amazon Bedrock は、主要な AI スタートアップと Amazon の高性能基盤モデル (FMs) を統合 API を通じて使用できるようにするフルマネージドサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- AWS Step Functions はサーバーレスオーケストレーションサービスで、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立ちます。

ベストプラクティス

- Amazon Bedrock は、トレーニングされたデータから学習する生成 AI モデルであるため、そのデータを使用してコンテキストをトレーニングおよび生成します。ベストプラクティスとして、データ漏洩の問題につながる可能性のあるプライベート情報を隠します。
- 生成 AI は貴重なインサイトを提供できますが、重要なエラー処理の決定には、特に本番環境で人的監視が必要です。

エピック

ワークフロー用のステートマシンを作成する

タスク	説明	必要なスキル
ステートマシンの作成。	ワークフローに適したステートマシンを作成するには、次の手順を実行します。 1. にサインインし AWS Management Console、AWS Step Functions コンソールを開きます。 2. 左側のナビゲーションペインから、ステートマシンを選択します。 3. [ステートマシンの作成] を選択します。 4. ユースケースに応じてテンプレートを選択するか、空白を選択して要件に応じてテンプレートを作成します。	AWS DevOps

「Lambda 関数の作成」

タスク	説明	必要なスキル
Lambda 関数を作成する。	Lambda 関数を作成するには、次の手順を実行します。 1. で AWS Management Console、 AWS Lambda コンソールに移動します。	AWS DevOps

タスク	説明	必要なスキル
	2. 左のナビゲーションペインで、[Functions] (関数)、[Create function] (関数の作成) の順に選択します。 3. 関数の作成ページで、関数を作成するオプションから選択します。次に、関数名に名前を入力し、ランタイムのドロップダウンリストから適切な言語を選択します。 4. [Create function (関数の作成)] を選択します。	

タスク	説明	必要なスキル

Lambda コードで必要なロジックを設定します。	• を使用して Amazon Bedrock API に接続するにはAWS SDK for Python (Boto3)、次のコードを使用します。
	
	このコードは、Amazon Bedrock のクライアントを設定し、必要なパラメータを準備してから、指定されたプロンプトを使用して Claude 3 モデルにリクエストを送信します。
	
	このパターンは Claude 3 モデルを呼び出します。関連するモデル IDs「Amazon Bedrock でサポートされている基盤モデル」を参照してください。
	
	`client = boto3.client(`
	`service_n`
	`ame="bedrock-runti`
	`me", region_na`
	`me="selected-region"`
	`)`
	
	`# Invoke Claude 3`
	`with the text prompt`
	`model_id = "your-mod`
	`el-id" # Select your`
	`Model ID, Based on the`
	`Model Id, Change the`
	`body format`
	
	`try:`
		AWS DevOps

タスク	説明	必要なスキル
	<pre>response = client.invoke_model(modelId=model_id, body=json.dumps({ "anthropic_version": "bedrock-2023-05-31", "max_tokens": 1024, "messages": [{ "role": "user", "content": [{"type": "text", "text": prompt}], }], }),)</pre> • （オプション）AWS アカウント IDs をプレースホルダーアカウント IDs。セキュリティ上の理由から、このアプローチは、機密アカウント情報を含む可能性のあるログ、エラーメッセージ、またはその他の出力をサニタイズするのに役立ちます。	

タスク	説明	必要なスキル

	次のコードでは、コロンで囲まれた 12 桁の数字 (Amazon リソースネーム (ARNs) の AWS アカウント IDs の形式およびその他の AWS 識別子) が出現すると、それをプレースホルダーアカウント ID に置き換えます":123456789012:" 。
	
	`def replace_account_id(input_string):`
	
	`    # Use a regular expression to find the AWS account ID pattern`
	`    account_id_pattern = r'(:\d{12}:)'`
	
	`    # Replace the matched pattern with ":123456789012:"`
	`    modified_string = re.sub(account_id_pattern, ":123456789012:", input_string)`
	
	`    return modified_string`	

Step Functions を Lambda と統合する

タスク	説明	必要なスキル
Step Functions でエラーを処理するように Lambda を設定します。	ワークフローを中断せずにエラーを処理するように Step Functions を設定するには、次の手順を実行します。 1. Step Functions コンソールで、前に作成したステートマシンに移動します。 2. 編集 を選択し、エラー処理を設定するサービスを選択し、エラー処理 を選択します。 3. 新しいキャッチャーを追加を選択し、フォールバック状態で Lambda を選択し、前に作成した Lambda 関数を選択します。詳細については、Step Functions ドキュメントの「キャッチエラー」を参照してください。	AWS DevOps

トラブルシューティング

問題	ソリューション
Lambda は Amazon Bedrock API にアクセスできません (実行する権限がありません)	このエラーは、Lambda ロールに Amazon Bedrock API へのアクセス許可がない場合に発生します。この問題を解決するには、Lambda ロールの`AmazonBedrockFullAccess` ポリシーを追加します。詳細については、AWS 「 マネージドポリシーリファレンスガイド」

問題	ソリューション
	のAmazonBedrockFullAccess」を参照してください。
Lambda タイムアウトエラー	プロンプトによっては、レスポンスを生成して返すまでに 30 秒以上かかる場合があります。この問題を解決するには、設定時間を長くします。詳細については、「 AWS Lambda デベロッパーガイド」の「Lambda 関数のタイムアウトを設定する」を参照してください。

関連リソース

- Amazon Bedrock
- Amazon Bedrock API アクセス
- 最初の Lambda 関数を作成する
- Step Functions を使用したワークフローの開発
- AWS Step Functions

サーバーレスアプローチを使用して AWS サービスを連結する

作成者:Aniket Braganza (AWS)

概要

このパターンは、Amazon Simple Storage Service (Amazon S3)、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Queue Service (Amazon SQS)、AWS Lambda を一緒に連結させることで、アップロードされたファイルを処理するスケーラブルでサーバーレスのアプローチを実演します。アップロードされたサンプルファイルはデモ用です。ビジネス目標を満たすために必要な AWS サービスの組み合わせを連結することで、サーバーレスのアプローチを使用して、他のタスクを実行できます。サーバーレスアプローチでは、イベント駆動型の通知、耐障害性のあるストレージ、Function as a Service (FaaS) コンピューティングを利用してのリクエスト処理する非同期ワークフローを採用しています。サーバーレス方式を使用すると、コストを最小限に抑えながら、ニーズに合わせてスケールできます。

> **Note**
>
> サーバーレスアプローチを使用して AWS サービスを連鎖させるには、いくつかのオプションがあります。例えば、Amazon SNS や Amazon SQS ではなく、Lambda と Amazon S3 を組み合わせた方法を使用できます。ただし、このパターンでは Amazon SNS と Amazon SQS を使用しています。これは、イベント通知中に Lambda 呼び出しに複数の統合ポイントを追加し、処理オーバーヘッドを最小限に抑えながら、サーバーレスオーケストレーションに複数のリスナーを含めるように実装を拡張できるからです。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS アカウントへのプログラムされたアクセス。詳細については、以下を参照してください。
  - AWS Cloud Development Kit (AWS CDK) ドキュメントの前提条件
  - AWS コマンドラインインターフェイス (AWS CLI) 「ドキュメントの前提条件」
- AWS CDK がインストール済み
- AWS CLI がインストール済みおよび設定済み
- Python 3.9

製品バージョン

- AWS CDK 2.x
- Python 3.9

アーキテクチャ

次の図は、連鎖された AWS のサービスが、ユーザーが処理のために S3 バケットにファイルをアップロードできるようにする方法を示しています。

この図表は、次のワークフローを示しています：

1. ユーザーが S3 バケットにファイルをアップロードします。
2. アップロードにより、SNS トピックにメッセージを発行する S3 イベントが開始されます。メッセージには、S3 イベントの詳細が含まれます。
3. SNS の件名に投稿されたメッセージは SQS キューに挿入され、このキューはサブスクライブされ、その件名の通知を受信します。
4. Lambda 関数は SQS キューをポーリングして(イベントソースとして)、メッセージが処理されるのを待ちます。
5. Lambda 関数は SQS キューからメッセージを受信すると、それらを処理し、メッセージの受信を確認します。
6. メッセージが Lambda によって処理されない場合、メッセージは SQS キューに返され、最終的に「SQS デッドレターキュー」に転送されます。

テクノロジースタック

- Amazon S3
- Amazon SNS
- Amazon SQS
- AWS Lambda

ツール

AWS サービス

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Queue Service (Amazon SQS) は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

その他のツール

- AWS Cloud Development Kit (AWS CDK) は、AWS CDK アプリケーションを操作するための主要ツールです。アプリを実行し、定義したアプリケーションモデルを調べ、AWS CDK によって生成された AWS CloudFormation テンプレートを作成してデプロイします。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Python は、高水準のインタープリター型汎用プログラミング言語です。

コード

このパターンのコードは、GitHub 内の「S3 から SNS、SQS、Lambda への連結」リポジトリで利用できます。

エピック

サーバーレス環境の開発

タスク	説明	必要なスキル
リポジトリをクローン作成します。	リポジトリをクローンし、`python/s3-sns-sqs-lambda-chain` フォルダに移動します。	アプリ開発者

タスク	説明	必要なスキル
仮想環境をセットアップします。	1. AWS CDK で、`python3 -m venv .venv` コマンドを実行します。 2. MacOS/Linux または `.venv\Scripts\activate.bat` Windows で `source .venv/bin/activate` コマンドを実行します。	アプリ開発者
依存関係をインストールします。	`pip install -r requirements.txt` コマンドを実行します。	アプリ開発者

CloudFormation スタックをテストします。

タスク	説明	必要なスキル
ユニットテストを実行します。	1. `pip install -r requirements-dev.txt` コマンドを実行します。 2. **Important** (オプション) `cdk synth --no-staging > template.yml` コマンドを実行して CloudFormation スタックを生成します。 スタックは検査できますが、ステージングされた	アプリ開発者、テストエンジニア

タスク	説明	必要なスキル
	リソースとアーティファクトの生成は避けてください。 3. pytest コマンドを実行し、すべてのユニットテストを実行します。 4. (オプション) pytest tests/unit/<test_filename> コマンドを実行し、特定ファイルのテストを実行します。	

CloudFormation スタックをデプロイする

タスク	説明	必要なスキル
ブートストラップ環境をセットアップします。	AWS ドキュメントの「ブートストラップ」の説明に従って、CloudFormation スタックがデプロイされる各 AWS リージョンで AWS CDK デプロイ環境を起動します。 **Note** このステップでは、プログラムによるアクセスが可能な認証情報が必要です。	アプリ開発者、DevOps エンジニア、データエンジニア
CloudFormation スタックをデプロイします。	cdk deploy コマンドを実行してスタックをビルドし	アプリ開発者、AWS DevOps エンジニア、DevOps

タスク	説明	必要なスキル
	、AWS アカウントにデプロイします。	

環境のリソースをクリーンアップする

タスク	説明	必要なスキル
CloudFormation スタックを削除し、関連付けられたリソースを削除します。	作成した CloudFormation スタックを削除し、関連付けられたリソースをすべて削除するには、`run cdk destroy` コマンドを実行します。	アプリ開発者

その他のパターン

- Athena による Amazon DynamoDB テーブルへのアクセス、クエリ、結合
- Athena でのML 予測のため、Amazon DynamoDB 内のデータを集約
- GitHub Actions を使用して Python AWS CDK アプリケーションの Amazon CodeGuru レビューを自動化する
- AWS リソース評価を自動化する
- AWS SAM を使用してネストされたアプリケーションのデプロイを自動化
- 全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント
- DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする
- CodeCommit 内のモノリポジトリの変更を自動的に検出し、さまざまな CodePipeline パイプラインを開始する
- DevOps プラクティスと AWS Cloud9 を使用して、マイクロサービスで疎結合アーキテクチャを構築する
- Amazon OpenSearch Service におけるマルチテナントサーバーレスアーキテクチャの構築
- AWS クラウドで高度なメインフレームファイルビューアを構築
- AWS のサービスを使用してバリューアットリスク (VaR) を計算
- AWS Service Catalog 製品を異なる AWS アカウントと AWS リージョンにコピー
- Java および Python プロジェクト用の動的 CI パイプラインを自動的に作成
- CQRS とイベントソーシングを使用してモノリスをマイクロサービスに分解する
- React ベースのシングルページアプリケーションを Amazon S3 と CloudFront にデプロイする
- プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API を内部 Web サイトにデプロイする
- Amazon EKS クラスターをデプロイおよびデバッグ
- インフラストラクチャをコードとして使用して、AWS クラウドにサーバーレスデータレイクをデプロイして管理する
- Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする
- Amazon Bedrock エージェントとナレッジベースを使用して、完全に自動化されたチャットベースのアシスタントを開発する
- RAG および ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する
- Step Functions を使用して IAM アクセスアナライザーで IAM ポリシーを動的に生成

- Amazon S3 へのAmazon EMR ロギングが有効になっていることを確認する
- DynamoDB テーブルのコストをオンデマンドで見積る
- Amazon Personalize を使用して、パーソナライズされ再ランク付けされたレコメンデーションを生成します
- AWS Glue ジョブと Python を使用してテストデータを生成します
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- AWS Step Functions を使用して、サーバーレス Saga パターンを実装する
- AWS CDK を使用して複数の AWS リージョン、アカウント、OU にわたって Amazon DevOps Guru を有効にすることで、運用パフォーマンスを向上させます。
- Step Functions と Lambda プロキシ関数を使用して AWS アカウント間で CodeBuild プロジェクトを起動する
- AWS Glue を使用して Apache Cassandra ワークロードを Amazon Keyspaces に移行する
- 複数の で共有 Amazon マシンイメージの使用をモニタリングする AWS アカウント
- AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する
- AWS Step Functions を使用して ETL パイプラインを検証、変換、パーティショニングでオーケストレーションします
- Amazon Athena を使用して SQL で Amazon DynamoDB テーブルをクエリする Amazon Athena
- AWS Fargate を使用して、イベント駆動型でスケジュール済みの大規模ワークロードを実行する
- Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する
- 自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する
- AWS Lambda を使用して六角形アーキテクチャで Python プロジェクトを構築する
- 自然言語を OpenSearch クエリと Elasticsearch クエリのクエリ DSL に変換する
- マルチアカウント環境ですべてのセキュリティハブのメンバーアカウントにわたって、セキュリティ標準コントロールをオフにする
- アカウント間で Amazon Redshift クラスターから Amazon S3 にデータをアンロードする
- AWS Fargate WaitCondition フックコンストラクトを使用してリソースの依存関係とタスク実行を調整する
- Amazon Bedrock エージェントを使用して、テキストベースのプロンプトによる Amazon EKS でのアクセスエントリコントロールの作成を自動化する

ネットワーク

トピック

- AWS Transit Gateway によるリージョン間ピアリングの設定を自動化する
- AWS Transit Gateway を使用してネットワーク接続を一元化
- Application Load Balancer を使用して Oracle WebLogic 上の Oracle JD Edwards EnterpriseOne の HTTPS 暗号化を設定します
- プライベートネットワーク経由でアプリケーション移行サービスのデータプレーンをコントロールプレーンに接続
- AWS CloudFormation カスタムリソースと Amazon SNS を使用して Infoblox オブジェクトを作成します。
- の Amazon CloudWatch アラートをカスタマイズする AWS Network Firewall
- Terraform を使用して AWS Wavelength ゾーンにリソースをデプロイする
- DNS レコードを Amazon Route 53 プライベートホストゾーンに一括で移行する
- F5 から AWS のApplication Load Balancer に移行するときの HTTP ヘッダーを変更
- 複数の VPC から中央のAWS のサービスエンドポイントにプライベートにアクセスする
- 複数の でのインバウンドインターネットアクセスに関する Network Access Analyzer の検出結果のレポートを作成する AWS アカウント
- マルチアカウント AWS 環境でハイブリッドネットワークの DNS 解決を設定する
- ELB ロードバランサーに TLS 終了が必要であることを確認
- Splunk を使用して AWS Network Firewall のログとメトリクスを表示する
- その他のパターン

AWS Transit Gateway によるリージョン間ピアリングの設定を自動化する

作成者: Ram Kandaswamy (AWS)

概要

AWS Transit Gateway は、中央ハブを介して仮想プライベートクラウド (VPC) とオンプレミスネットワークを接続します。Transit Gateway のトラフィックは常にグローバル Amazon Web Services （AWS）バックボーンに留まり、パブリックインターネットを通過しないため、一般的なエクスプロイトや分散型サービス拒否（DDoS）攻撃などの脅威ベクトルが軽減されます。

2 つ以上の AWS リージョン間で通信する必要がある場合は、リージョン間 Transit Gateway のピアリングを使用して、異なるリージョンの Transit Gateway 間のピアリング接続を確立できます。ただし、Transit Gateway を使用してリージョン間ピアリングを手動で設定するのは、複数の手順を伴う時間のかかるプロセスです。このパターンでは、コードを使用してピアリングを実行することで、このような手動の手順を省くプロセスを自動化できます。このアプローチは、マルチリージョンの組織設定中に複数のリージョンと AWS アカウントを繰り返し設定する必要がある場合に使用できます。

このパターンでは、AWS Step Functions ワークフロー、AWS Lambda 関数、AWS Identity and Access Management (IAM) ロール、および Amazon CloudWatch Logs のロググループを含む AWS CloudFormation スタックを使用します。その後、Step Functions の実行を開始し、Transit Gateway のリージョン間ピアリング接続を作成できます。リージョン間ピアリングを手動で設定するには、VPCs AWS Transit Gateway」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 既存の Amazon Simple Storage Service (Amazon S3) バケット
- リクエスターのリージョンとアクセプターのリージョンで作成して設定されたトランジットゲートウェイ。　リクエスターのリージョンはピアリングリクエストを発信し、アクセプターのリージョンはピアリングリクエストを受け付けます。この詳細については、Amazon VPC ドキュメントの「VPC ピアリング接続の作成と承認」を参照してください。
- アクセプターのリージョンとリクエスターのリージョンにインストールされ、設定された VPC。VPC を作成する手順については、Amazon VPC ドキュメントの「Amazon VPC 入門ガイド」にある「VPC の作成」を参照してください。
- VPC は `addToTransitGateway` タグと `true` 値を使用する必要があります。

- 要件に応じて、VPC のセキュリティグループとネットワーク アクセスコントロールリスト (ネットワーク ACL) を設定します。この詳細については、Amazon VPC ドキュメントの「VPC のセキュリティグループ」と「ネットワーク ACL」を参照してください。

AWS リージョンと制限

- 特定の AWS リージョンのみが、リージョン間ピアリングをサポートしています。リージョン間ピアリングをサポートするリージョンの全リストについては、「AWS Transit Gateway」のよくある質問を参照してください。
- 添付のサンプルコードでは、リクエスターのリージョンは `us-east-2` で、アクセプターのリージョンは `us-west-2` であると仮定されています。異なるリージョンを設定する場合は、すべての Python ファイルでこれらの値を編集する必要があります。3 つ以上のリージョンを含むより複雑なセットアップを実装するには、Step Function を変更してリージョンをパラメータとして Lambda 関数に渡し、組み合わせごとに関数を実行できます。

アーキテクチャ

この図は、次のステップのワークフローを示しています。

1. ユーザーは AWS CloudFormation スタックを作成します。
2. AWS CloudFormation は、Lambda 関数を使用する Step Functions ステートマシンを作成します。詳細については、AWS Step Functions ドキュメントの「Lambda を使用する Step Functions ステートマシンの作成」を参照してください。
3. Step Functions はピアリング用の Lambda 関数を呼び出します。
4. Lambda 関数は、Transit Gateway 間でピアリング接続を作成します。
5. Step Function は、ルートテーブルを変更するための Lambda 関数を呼び出します。
6. Lambda 関数は、VPC の Classless Inter-Domain Routing (CIDR) ブロックを追加してルートテーブルを変更します。

Step Function ワークフロー

この図は、次の Step Function ワークフローを示しています。

1. Step Function ワークフローは、Transit Gateway ピアリング用の Lambda 関数を呼び出します。
2. 1 分間待機するタイマーコールがあります。
3. ピアリングステータスが取得され、条件ブロックに送信されます。ブロックがルーピングを実行します。
4. 成功条件が満たされない場合、ワークフローはタイマーステージに移行するようにコーディングされます。
5. 成功条件が満たされると、Lambda 関数が呼び出されてルートテーブルが変更されます。このコールの後、Step Function ワークフローは終了します。

ツール

- AWS CloudFormation — AWS CloudFormation は、AWS リソースをモデル化してセットアップするためのサービスです。
- Amazon CloudWatch Logs — CloudWatch Logs により、使用中のすべてのシステム、アプリケーション、AWS のサービスからのログを一元管理することができます。
- AWS Identity and Access Management (IAM) — IAM は、AWS サービスへのアクセスをセキュアに制御するためのウェブサービスです。
- AWS Lambda — Lambda は可用性の高いコンピューティングインフラストラクチャでコードを実行し、コンピューティングリソースの管理をすべて実行します。
- AWS Step Functions – Step Functions により、分散アプリケーションのコンポーネントをビジュアルワークフローの一連のステップとして簡単に編成できます。

エピック

ピアリングを自動化する

タスク	説明	必要なスキル
添付ファイルを S3 バケットにアップロードします。	AWS マネジメントコンソールにサインインし、Amazon S3 コンソールを開き、`modify-transit-gateway-routes.zip` 、`peer-transit-gateway.zip` 、`get-transit-gateway-peering-`	AWS 全般

タスク	説明	必要なスキル
	`status.zip` ファイル（添付ファイル）を S3 バケットにアップロードします。	

タスク	説明	必要なスキル
AWS CloudFormation スタックを作成します。	次のコマンドを実行して、`transit-gateway-peering.json` ファイル (添付ファイル) を使用して AWS CloudFormation スタックを作成します。 `aws cloudformation create-stack --stack-name myteststack --template-body file://sampletemplate.json` AWS CloudFormation スタックは、Step Functions ワークフロー、Lambda 関数、IAM ロール、および CloudWatch ロググループを作成します。 AWS CloudFormation テンプレートが、以前にアップロードしたファイルを含む S3 バケットを参照していることを確認してください。 **Note** AWS CloudFormation コンソールを使用してスタックを作成することもできます。この詳細については、AWS CloudFormation ドキュメントの「AWS CloudFormation コンソールでのスタックの	DevOps エンジニア

タスク	説明	必要なスキル
	作成」を参照してください。	
Step Functions 内で新しい実行を開始します。	Step Functions コンソールを開き、新しい実行を開始します。Step Functions は Lambda 関数を呼び出し、Transit Gateway のピアリング接続を作成します。JSON 入力ファイルは必要ありません。添付ファイルを使用できること、および接続タイプがピアリングであることを確認します。 この詳細については、AWS Steps Functions ドキュメントの「AWS Step Functions 入門ガイド」にある「新しい実行を開始する」を参照してください。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
ルートテーブル内のルートを検証します。	Transit Gateway 間でリージョン間ピアリングが確立されます。ルートテーブルは、ピアリージョン VPC の IPv4 CIDR ブロック範囲で更新されます。 Amazon VPC コンソールを開き、Transit Gateway アタッチメントに対応するルートテーブルの [関連付け] タブを選択します。ピアリングされたリージョンの VPC CIDR ブロック範囲を検証します。 詳細なステップと手順については、Amazon VPC ドキュメントの「Transit Gateway ルートテーブルを関連付ける」を参照してください。	ネットワーク管理者

関連リソース

- 「Step Functions で実行」
- 「Transit Gateway ピアリングアタッチメント」
- AWS Transit Gateway を使用して異なる AWS リージョンの VPCs をピア接続する
- 「AWS Transit Gateway を使用した AWS リージョン間の VPC の相互接続 - デモ」(ビデオ)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Transit Gateway を使用してネットワーク接続を一元化

作成者: Mydhili Palagummi (AWS) と Nikhil Marrapu (AWS)

概要

このパターンでは、AWS Transit Gateway を使用して、オンプレミスネットワークを AWS リージョン内の複数の AWS アカウントの仮想プライベートクラウド (VPC) に接続できる最も単純な構成を示しています。この設定を使用して、リージョン内の複数の VPC ネットワークとオンプレミスネットワークを接続するハイブリッドネットワークを確立できます。これは、トランジットゲートウェイと、オンプレミスネットワークへの仮想プライベートネットワーク (VPN) 接続を使用することで達成されます。

前提条件と制限

前提条件

- AWS Organizations の組織のメンバーアカウントとして管理される、ネットワークサービスをホストするためのアカウント
- Classless Inter-Domain Routing (CIDR) ブロックが重複しない、複数の AWS アカウントの VPC

機能制限

このパターンでは、特定の VPC 間またはオンプレミスネットワーク間のトラフィックの分離をサポートしません。トランジットゲートウェイに接続されているすべてのネットワークは、相互にアクセスできるようになります。トラフィックを分離するには、トランジットゲートウェイでカスタムルートテーブルを使用する必要があります。このパターンでは、最も単純な構成である単一のデフォルトトランジットゲートウェイルートテーブルを使用して、VPC とオンプレミスネットワークのみを接続します。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Transit Gateway
- AWS Site-to-Site VPN
- VPC
- AWS Resource Access Manager (AWS RAM)

ターゲットアーキテクチャ

ツール

AWS サービス

- 「AWS Resource Access Manager (AWS RAM)」 は、AWS アカウント、組織単位、または AWS Organizations の組織全体でリソースを安全に共有するのに役立ちます。
- AWS Transit Gatewayは、仮想プライベートクラウド (VPC) とオンプレミスネットワークを接続する中央ハブです。

エピック

ネットワークサービスアカウントでのトランジットゲートウェイの作成

タスク	説明	必要なスキル
transit gateway を作成します。	ネットワークサービスをホストする AWS アカウントで、ターゲット AWS リージョンにトランジットゲートウェイを作成します。手順については、「トランジットゲートウェイの作成」を参照してください。次の点に注意してください: • デフォルトルートテーブルの関連付けを選択します。 • デフォルトルートテーブル伝達を選択します。	ネットワーク管理者

オンプレミスネットワークをトランジットゲートウェイに接続する

タスク	説明	必要なスキル
VPN 接続のカスタマーゲートウェイデバイスを設定します。	カスタマーゲートウェイデバイスは、トランジットゲートウェイとオンプレミスネットワーク間の、Site-to-Site VPN 接続のオンプレミス側に接続されています。詳細については、AWS Site-to-Site VPN ユーザーガイドの「カスタマーゲートウェイデバイス」を参照してください。適用されるオンプレミスの顧客デバイスを特定または起動し、そのパブリック IP アドレスを書き留めます。VPN の設定は、このエピックの後半で完了します。	ネットワーク管理者
ネットワークサービスアカウントで、トランジットゲートウェイへの VPN アタッチメントを作成します。	接続をセットアップするには、トランジットゲートウェイの VPN アタッチメントを作成します。手順については、「トランジットゲートウェイ VPN アタッチメント」を参照してください。	ネットワーク管理者
オンプレミスネットワークのカスタマーゲートウェイデバイスに、VPN を設定します。	トランジットゲートウェイに関連付けられているSite-to-Site VPN 接続の設定ファイルをダウンロードして、カスタマーゲートウェイデバイスの VPN セッティングを設定します。手順については、「設定	ネットワーク管理者

タスク	説明	必要なスキル
	ファイルをダウンロード」を参照してください。	

ネットワークサービスアカウントのトランジットゲートウェイを、他の AWS アカウントまたは組織と共有する

タスク	説明	必要なスキル
AWS Organizations 管理アカウントで、共有をオンにします。	トランジットゲートウェイを組織する、または特定の組織単位と共有するには、AWS Organizations で共有をオンにします。そうしないと、アカウントごとにトランジットゲートウェイを個別に共有する必要性がでてきます。手順については、「AWS Organizations 内のリソース共有の有効化」 を参照してください。	AWS システム管理者
ネットワークサービスアカウントにトランジットゲートウェイリソースシェアを作成します。	組織の他の AWS アカウントの VPC がトランジットゲートウェイに接続できるようにするには、ネットワークサービスアカウントで AWS RAM コンソールを使用してトランジットゲートウェイリソースを共有します。手順については、「リソース共有の作成」を参照してください。	AWS システム管理者

Transit Gateway に VPC を接続する

タスク	説明	必要なスキル
個別のアカウントで VPC アタッチメントを作成します。	トランジットゲートウェイが共有されているアカウントで、トランジットゲートウェイ VPC アタッチメントを作成します。手順については、「VPC へのトランジットゲートウェイアタッチメントの作成」を参照してください。	ネットワーク管理者
VPC アタッチリクエストを受け入れます。	ネットワークサービスアカウントで、トランジットゲートウェイの VPC アタッチメントリクエストを受け入れます。手順については、「共有アタッチメントの承認」を参照してください。	ネットワーク管理者

ルーティングを設定する

タスク	説明	必要なスキル
個々のアカウント VPC にルートを設定します。	個々のアカウント VPC に、Transit Gateway をターゲットとして使用して、オンプレミスのネットワークと他の VPC ネットワークへのルートを追加します。手順については、「ルートテーブルからのルートの追加と削除」を参照してください。	ネットワーク管理者
トランジットゲートウェイのルートテーブル内のルート。	VPC と VPN 接続からのルートは伝達され、トランジッ	ネットワーク管理者

タスク	説明	必要なスキル
	トゲートウェイのデフォルトルートテーブルに表示されるはずです。必要に応じて、トランジットゲートウェイのデフォルトルートテーブルに静的ルート (静的 VPN 接続のための静的ルートが一例) を作成します。手順については、「静的ルートの作成」 を参照してください。	
セキュリティグループとネットワークをアクセスコントロールリスト (ACL)ルールに加えます。	EC2 インスタンスと VPC 内のその他のリソースについては、セキュリティグループルールルールとネットワーク ACL ルールが 、VPC とオンプレミスネットワーク間のトラフィックを許可していることを確認します。手順については、「セキュリティグループを使用してリソースへのトラフィックを制御」 と「ACL にルールを追加または削除」 を参照してください。	ネットワーク管理者

接続テスト

タスク	説明	必要なスキル
VPC 間の接続をテストします。	ネットワーク ACL とセキュリティグループが、インターネット制御メッセージプロトコル (ICMP) トラフィックを許可することを確認し、次に VPC 内のインスタンスから、	ネットワーク管理者

タスク	説明	必要なスキル
	同じくトランジットゲートウェイに接続されている別のVPCへのネットワーク接続を確認します。	
VPC とオンプレミスネットワーク間の接続をテストします。	ネットワーク ACL ルール、セキュリティグループルール、およびファイアウォールが ICMP トラフィックを許可することを確認し、オンプレミスネットワークと VPC の EC2 インスタンス間の接続を確認します。VPNを UP ステータスに接続させるには、まずオンプレミスネットワークからネットワーク通信を開始する必要があります。	ネットワーク管理者

関連リソース

- スケーラブルでセキュアなマルチ VPC の AWS ネットワークインフラストラクチャを構築する (AWS ホワイトペーパー)
- 「共有リソースの使用」 (AWS RAM ドキュメント)
- 「トランジットゲートウェイの操作」 (AWS Transit Gateway ドキュメント)

Application Load Balancer を使用して Oracle WebLogic 上の Oracle JD Edwards EnterpriseOne の HTTPS 暗号化を設定します

作成者: Thanigaivel Thirumalai (AWS)

概要

このパターンは、Oracle WebLogic ワークロード上の Oracle JD Edwards EnterpriseOne で SSL オフロード用に HTTPS 暗号化を設定する方法を説明しています。この方法では、ユーザーのブラウザとロードバランサー間のトラフィックを暗号化して、EnterpriseOne サーバーにかかる暗号化の負担を軽減します。

多くのユーザーは、「AWS Application Load Balancer」を使用して EnterpriseOne JAVA 仮想マシン (JVM) 階層を水平方向にスケーリングしています。ロードバランサーは、クライアントの単一窓口として機能し、複数の JVM に受信トラフィックを分散する。オプションで、ロードバランサーはトラフィックを複数のアベイラビリティーゾーンに分散し、EnterpriseOne の可用性を高めることができます。

このパターンで説明するプロセスでは、ロードバランサーと EnterpriseOne JVM 間のトラフィックを暗号化する代わりに、ブラウザとロードバランサー間の暗号化を設定します。このアプローチは SSL オフロードと呼ばれます。SSL 復号化プロセスを EnterpriseOne ウェブサーバーまたはアプリケーションサーバーからApplication Load Balancer にオフロードすることで、アプリケーション側の負担が軽減されます。ロードバランサーで SSL が終了すると、暗号化されていないトラフィックは AWS 上のアプリケーションにルーティングされます。

「Oracle JD Edwards EnterpriseOne」は、製品や物理的資産の製造、建設、流通、サービス、管理を行う組織向けのエンタープライズ・リソース・プランニング (ERP) ソリューションです。JD Edwards EnterpriseOne は、さまざまなハードウェア、オペレーティングシステム、データベースプラットフォームをサポートしています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS サービス呼び出しを行い、AWS リソースを管理するアクセス許可を持つ AWS Identity and Access Management (IAM) ロール
- SSL 証明書

製品バージョン

- このパターンは Oracle WebLogic 12c でテストされていますが、他のバージョンを使用することもできます。

アーキテクチャ

SSL オフロードを実行するには複数の方法があります。このパターンでは、次の図に示すように、Application Load Balancer と Oracle HTTP Server (OHS) を使用します。

次の図は、JD Edwards EnterpriseOne、Application Load Balancer、および Java アプリケーションサーバー (JAS) JVM レイアウトを示しています。

ツール

AWS サービス

- 「Application Load Balancer」は、複数のアベイラビリティゾーンにある Amazon Elastic Compute Cloud (Amazon EC2 インスタンス) などの複数のターゲットに、受信するアプリケーショントラフィックを分散する。
- 「AWS Certificate Manager (ACM)」は、AWS ウェブサイトとアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書とキーの作成、保存、更新に役立ちます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

ベストプラクティス

- ACM のベストプラクティスについては、「ACM のドキュメント」を参照してください。

エピック

WebLogic と OHS をセットアップ

タスク	説明	必要なスキル
Oracle コンポーネントのインストールと設定。	1. 標準のインストールプロセスに従って Fusion ミドルウェアインフラストラクチャをインストールします。このプログラムは WebLogic ドメインのインストールと設定に役立ちます。手順については、「Oracle ドキュメント」を参照してください。 2. 標準のインストール手順に従って OHS をインストールします。手順については、「Oracle ドキュメント」を参照してください。 3. インストールが完了したら、設定ウィザード (`config.sh` ファイル) を起動して OHS を設定します。 • 既存のドメインを更新することも、新しいドメインを作成することもできます。このパターンは、既存のドメインを更新することを前提としています。 • [使用可能なテンプレート] では、[Oracle Enterprise Manager-R	JDE CNC、WebLogic 管理者

タスク	説明	必要なスキル
	estricted JRF] と [Oracle HTTP Server (制限付き JRF)] を選択します。これらの Java Required Files (JRF) オプションを選択すると、外部データベースへの接続が不要になります。 • [マネージドサーバー]、[クラスター]、[サーバーテンプレート]、[Coherence クラスター]、[マシン]、[マシンへのサーバーの割り当て]、[仮想ターゲット]、[パーティション] の場合は、デフォルトの設定値をそのまま使用し、[次へ] を選択して次のカテゴリに移動します。 • OHS インスタンス (例: ohs1) の構成詳細 (管理者ホストとポート、リッスンアドレスとポート、サーバー名など) を入力します。	

タスク	説明	必要なスキル
WebLogic プラグインの名前を、ドメインレベルで有効にします。	負荷分散には WebLogic プラグインが必要です。プラグインを有効にするには: 1. 次のリンクを使用して WebLogic 管理コンソールにログインします。 `http://<WeblogicServer>:<Adminport>/console` 2. [ロックして編集] を選択し、[設定]、[ウェブアプリケーション] の順に選択します。 3. [WebLogic プラグインの有効化] (チェックボックスまたはドロップダウンオプション) を選択します。 4. [変更の保存と有効化] を選択します。	JDE CNC、WebLogic 管理者

タスク	説明	必要なスキル
設定ファイルを編集します。	mod_wl_ohs.conf ファイルでは OHS から WebLogic へのプロキシリクエストを設定します。 1. ファイルを編集します。場所は $ORACLE_HOME/user_projects/domains/ 以下に例を示します。 /home/oracle/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/config/fmwconfig/components/OHS/instances/ohs1 2. WebLogic のホスト (WebLogicHost) とポート (WebLogicPort) の値を追加します (このパターンはローカルホストとポート 8000 を前提としています)。 3. WLProxySSL と WLProxySSLPassThrough の値を次のように追加します。 <VirtualHost *:8000> <Location /jde> WLSRequest On	JDE CNC、WebLogic 管理者

タスク	説明	必要なスキル
	SetHandler weblogic-handler WebLogicHost localhost WebLogicPort 8000 WLProxySSL On WLProxySSLPassThrough On </Location> </VirtualHost>	

タスク	説明	必要なスキル
エンタープライズマネージャーを使用して OHS を起動します。	1. 次のリンクを使用して Enterprise Manager Fusion ミドルウェアにログインします。 `http://<WeblogicServer>:<Adminport>/em/` 2. [ターゲットナビゲーション] の [HTTP サーバー] で、OHS インスタンス (例: `ohs1`) を選択します。 3. [シャットダウン] と [起動] を選択して OHS インスタンスを再起動します。 4. OHS のセットアップが完了すると、EnterpriseOne サーバーのホスト名の代わりにポート 8000 の HTTP サーバーのホスト名を使用して EnterpriseOne HTML クライアントに接続できます。 • 古いリンク: `http://<Webserver>:80/jde/owhtml` • 新しいリンク: `http://<HTTP server or web server>:8000/jde/owhtml` デフォルトの Oracle HTTP ポート以外のポートを使用する場合は、`httpd.con`	JDE CNC、WebLogic 管理者

タスク	説明	必要なスキル

	f ファイルを編集してそのポートのリスナーを 2 か所に追加してください。
	
	`#[Listen] OHS_LISTEN_PORT`
	`Listen 8000`
	
	と
	
	`#`
	`ServerName <Weblogic Server1>:8000`	

Application Load Balancer の設定

タスク	説明	必要なスキル
ターゲットグループを設定します。	1. HTTP サーバーポート 8000 のターゲットグループを作成します。 2. ターゲットをターゲットグループの下に同じポートに登録します。 3. ターゲットの状態をチェックして、正常であることを確認します。 4. ヘルスチェックの設定は、必要に応じて行います。 詳細な手順については、「Elastic Load Balancing のド	AWS 管理者

タスク	説明	必要なスキル
	キュメント」を参照してください。	
ロードバランサーの設定	1. デフォルト属性と必要な仮想プライベートクラウド (VPC)、セキュリティグループ、サブネットを使用して Application Load Balancer を作成します。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。 2. HTTPS 443 のリスナーエントリを追加し、前のステップで作成したターゲットグループに転送します。(詳細については、「Elastic Load Balancing ドキュメント」を参照してください。)。HTTPS リスナーには SSL 証明書が必要です。ACM から証明書を選択するか、アップロードできます。 3. どちらのリスナーでも、「Elastic Load Balancing ドキュメント」の指示に従ってスティッキネスを有効にします。	AWS 管理者

タスク	説明	必要なスキル
Route 53 (DNS) レコードを追加します。	(オプション) サブドメインに Amazon Route 53 DNS レコードを追加できます。このレコードは、Application Load Balancer を指します。手順については、「Route 53 ドキュメント」を参照してください。	AWS 管理者

トラブルシューティング

問題	ソリューション
HTTP サーバーは表示されません。	Enterprise Manager コンソールの [ターゲットナビゲーション] リストに [HTTP サーバー] が表示されない場合は、以下の手順に従ってください。 1. [WebLogic ドメイン]、[管理] で [OHS インスタンス] を選択します。 2. [作成] を選択して、新しい OHS インスタンスを作成します。 3. インスタンス名を指定し、[OK] を選択してインスタンスを作成します。 インスタンスが作成され、変更が有効になると、[ターゲットナビゲーションパネル] に HTTP サーバーが表示されます。

関連リソース

AWS ドキュメント

- アプリケーション ロード バランサー
- 「パブリックホストゾーンの操作」
- 「プライベートホストゾーンの操作」

Oracle ドキュメンテーション:

- 「Oracle WebLogic サーバープロキシープラグインの概要」
- 「インフラストラクチャー・インストーラを使用した WebLogic Server のインストール」
- 「Oracle HTTP サーバーのインストールと設定」

プライベートネットワーク経由でアプリケーション移行サービスのデータプレーンをコントロールプレーンに接続

作成者: Dipin Jain (AWS) と Mike Kuznetsov (AWS)

概要

このパターンでは、インターフェイス VPC エンドポイントを使用して、プライベートで保護されたネットワーク上の AWS Application Migration Service データプレーンとコントロールプレーンに接続する方法について説明します。

Application Migration Service は、アプリケーションの移行を簡素化、迅速化、削減する高度に自動化されたlift-and-shift (リホスト) ソリューションです AWS。企業が、互換性の問題、パフォーマンスの中断、または長いカットオーバー期間に悩まされることなく、数多くの物理サーバー、仮想サーバー、またはクラウドサーバーをリホストすることを可能にします。Application Migration Service は、 から入手できます AWS Management Console。これにより AWS のサービス、Amazon CloudWatch AWS CloudTrailや AWS Identity and Access Management (IAM) などの他の とシームレスに統合できます。

AWS VPN サービス、または Application Migration Service の VPC ピアリングを使用して AWS Direct Connect、プライベート接続を介して、ソースデータセンターからデータプレーン、つまり宛先 VPC 内のデータレプリケーションのステージングエリアとして機能するサブネットに接続できます。また、 を搭載したインターフェイス VPC エンドポイントを使用して AWS PrivateLink 、プライベートネットワーク経由で Application Migration Service コントロールプレーンに接続することもできます。

前提条件と制限

前提条件

- ステージングエリアサブネット – Application Migration Service を設定する前に、ソースサーバーから AWS (つまり、データプレーン) にレプリケートされるデータのステージングエリアとして使用するサブネットを作成します。アプリケーション移行サービスコンソールに初めてアクセスする場合、「レプリケーション設定テンプレート」でこのサブネットを指定する必要があります。レプリケーション設定テンプレートで、特定のソースサーバーのこのサブネットをオーバーライドできます。で既存のサブネットを使用できますが AWS アカウント、この目的のために新しい専用サブネットを作成することをお勧めします。
- ネットワーク要件 – ステージングエリアサブネットで Application Migration Service によって起動されるレプリケーションサーバーは、 の Application Migration Service

API エンドポイントにデータを送信できる必要があります。 <region>はhttps://mgn.<region>.amazonaws.com/、 AWS リージョン レプリケート先の のコードです (例: https://mgn.us-east-1.amazonaws.com) 。Amazon Simple Storage Service (Amazon S3) のサービス URL は、アプリケーション移行サービスのサービス URL に必要です。

- AWS レプリケーションエージェントのインストーラは、Application Migration Service AWS リージョン で使用している の Amazon Simple Storage Service (Amazon S3) バケット URL にアクセスできる必要があります。
- ステージングエリアのサブネットは、Amazon S3 にアクセスできる必要があります。
- AWS レプリケーションエージェントがインストールされているソースサーバーは、ステージングエリアサブネットのレプリケーションサーバーと の Application Migration Service API エンドポイントにデータを送信できる必要がありますhttps://mgn.<region>.amazonaws.com/。

以下の表では、必要なポートを示しています。

ソース	送信先	[ポート]	詳細については、以下を参照
ソースデータセンター	Amazon S3 サービスの URL	443 (TCP)	TCP ポート 443 を介した通信
ソースデータセンター	AWS リージョン Application Migration Service の 固有のコンソールアドレス	443 (TCP)	「TCP ポート 443 を介したソースサーバーとアプリケーション移行サービス間の通信」
ソースデータセンター	ステージングエリアのサブネット	1500 (TCP)	「TCP ポート 1500 を介したソースサーバーとステージングエリアサブネット間の通信」
ステージングエリアのサブネット	AWS リージョン Application Migration Service の 固有のコンソールアドレス	443 (TCP)	「TCP ポート 443 を介したステージングエリアサブネットとアプリケーション

			移行サービス間の通信」
ステージングエリアのサブネット	Amazon S3 サービスの URL	443 (TCP)	「TCP ポート 443 を介した通信」
ステージングエリアのサブネット	サブネットの の Amazon Elastic Compute Cloud (Amazon EC2) エンドポイント AWS リージョン	443 (TCP)	「TCP ポート 443 を介した通信」

制限

Application Migration Service は現在、すべての AWS リージョン およびオペレーティングシステムで利用できるわけではありません。

- サポートされている AWS リージョン
- サポートされるオペレーティングシステム

アーキテクチャ

次の図表は、一般的な移行のネットワークアーキテクチャを示しています。このアーキテクチャの詳細については、「アプリケーション移行サービスのドキュメント」と「アプリケーション移行サービスのサービスのアーキテクチャおよびネットワークアーキテクチャのビデオ」を参照してください。

次の詳細ビューは、Amazon S3 とアプリケーション移行サービスを接続するためのステージングエリア VPC のインターフェイス VPC エンドポイントの設定を示しています。

ツール

- AWS Application Migration Service は、 でのアプリケーションのリホストを簡素化、迅速化し、コストを削減します AWS。

- インターフェイス VPC エンドポイントを使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続を必要と AWS PrivateLink せずに、 を搭載したのサービスに接続できます。VPC のインスタンスは、サービスのリソースと通信するためにパブリック IP アドレスを必要としません。VPC と他のサービス間のトラフィックは、Amazon ネットワークを離れません。

エピック

アプリケーション移行サービス、Amazon EC2、Amazon S3 のエンドポイントを作成

タスク	説明	必要なスキル
アプリケーション移行サービスのインターフェースエンドポイントを設定します。	ソースデータセンターとステージングエリア VPC は、ターゲットステージングエリア VPC で作成したインターフェイスエンドポイントを介して、アプリケーション移行サービスコントロールプレーンにプライベートに接続します。エンドポイントを作成するには： 1. Amazon Virtual Private Cloud (Amazon VPC) コンソールを開きます。 2. ナビゲーションペインで、[エンドポイント]、[エンドポイントを作成] の順に選択します。 3. [Service category] (サービスカテゴリ) で、AWS のサービス を選択します。 4. [サービス名] に `com.amazonaws.<region>.mgn` と入力します。[タイプ]	移行リード

タスク	説明	必要なスキル
	で、[インターフェイス] を選択します。 5. VPC は、ターゲットステージング領域VPCを選択して、エンドポイントを作成します。 6. サブネット で、エンドポイントネットワークインターフェイスを作成するサブネットを選択します。 7. インターフェイスエンドポイントのプライベート DNS をオンにするには、[追加設定]セクションで[DNS 名を有効にする]を選択します。 8. TCP 443 経由でステージングエリア VPC サブネットからの進入を許可するセキュリティグループを選択します。 9. [エンドポイントの作成] を選択します。 詳細については、Amazon VPC ドキュメントの「インターフェイス VPC エンドポイント AWS のサービス を使用して にアクセスする」を参照してください。	

タスク	説明	必要なスキル
Amazon EC2 のインターフェイスエンドポイントを設定します。	ステージングエリア VPC は、ターゲットのステージングエリア VPC で作成したインターフェイスエンドポイントを介して Amazon EC2 API にプライベートに接続します。エンドポイントを作成するには、前のストーリーで説明した手順に従います。 • [サービス名] に `com.amazonaws.<region>.ec2` と入力します。[タイプ] で、[Interface] (インターフェイス) を選択します。 • セキュリティグループは、ポート 443 経由でステージングエリア VPC サブネットからのインバウンド HTTPS トラフィックを許可する必要があります。 • [追加設定]セクションで [DNS 名を有効にする]を選択します。	移行リード

タスク	説明	必要なスキル
Amazon S3 のインターフェイスエンドポイントを設定します。	ソースデータセンターとステージングエリア VPC は、ターゲットステージングエリア VPC で作成したインターフェイスエンドポイントを介して Amazon S3 API にプライベートに接続します。エンドポイントを作成するには、最初のストーリーの手順に従います。 • [サービス名] に `com.amazonaws.<region>.s3` と入力します。[タイプ] で、[Interface] (インターフェイス) を選択します。 • VPC セキュリティグループは、ポート 443 経由でステージングエリア VPC サブネットからのインバウンド HTTPS トラフィックを許可する必要があります。 • [追加設定] セクションで、[DNS 名を有効にする] をオフにします。Amazon S3 インターフェイスエンドポイントは、プライベート DNS をサポートしていません。 **Note** ゲートウェイエンドポイント接続を VPC か	移行リード

タスク	説明	必要なスキル
	ら拡張できないため、インターフェイスエンドポイントを使用します。（詳細については、AWS PrivateLink ドキュメントを参照してください）。	
Amazon S3ゲートウェイエンドポイントを設定します。	設定フェーズでは、レプリケーションサーバーは S3 バケットに接続して、 AWS レプリケーションサーバーのソフトウェア更新をダウンロードする必要があります。ただし、Amazon S3 インターフェイスエンドポイントはプライベート DNS 名をサポートしていないため 、 Amazon S3 エンドポイント DNS 名をレプリケーションサーバーに提供する方法はありません。 この問題を軽減するには、ステージングエリアサブネットが属する VPC で Amazon S3 ゲートウェイエンドポイントを作成し、ステージングサブネットのルートテーブルを関連するルートで更新します。詳細については、AWS PrivateLink ドキュメントの「ゲートウェイエンドポイントの作成」を参照してください。	クラウド管理者

タスク	説明	必要なスキル
エンドポイントのプライベート DNS 名を解決するために、オンプレミス DNS を設定します。	アプリケーション移行サービスと Amazon EC2 のインターフェイスエンドポイントには、VPC で解決できるプライベート DNS 名があります。ただし、これらのインターフェイスエンドポイントのプライベート DNS 名を解決するように、オンプレミスサーバーを設定する必要もあります。 これらのサーバーを設定するには、複数の方法があります。このパターンでは、オンプレミスの DNS クエリをステージングエリア VPC の Amazon Route 53 Resolver インバウンドエンドポイントに転送することで、この機能をテストしました。詳細については、Route 53 ドキュメントの「VPC とネットワーク間における DNS クエリの解決」を参照してください。	移行エンジニア

プライベートリンクを介してアプリケーション移行サービスのコントロールプレーンに接続

タスク	説明	必要なスキル
を使用して AWS レプリケーションエージェントをインストールします AWS PrivateLink。	1. AWS レプリケーションエージェントを送信先リージョンのプライベート S3 バケットにダウンロードします。	移行エンジニア

タスク	説明	必要なスキル
	2. 移行するソースサーバーにログインします。 AWS レプリケーションエージェントのインストーラには、Application Migration Service と Amazon S3 エンドポイントへのネットワークアクセスが必要です。オンプレミスネットワークは Application Migration Service および Amazon S3 パブリックエンドポイントに対して開かれていないため、 を使用して前のステップで作成したインターフェイスエンドポイントを使用して エージェントをインストールする必要があります AWS PrivateLink。 以下は、Linux の例です： 1. コマンドを使用して、エージェントをダウンロードします： `wget -O ./aws-replication-installer-init.py \ https://aws-application-migration-service-<aws_region>.bucket.<s3-endpoint-DNS-name>/latest/linux/aws-repli`	

タスク	説明	必要なスキル

	`cation-installer-i nit.py`
	
	例えば、Amazon S3 インターフェイスエンドポイントの DNS 名が `vpce-009c 8b07adb052a11-qgf8 q50y.s3.us-west-1. vpce.amazonaws.com` で、 AWS リージョン がの場合`us-west-1` 、 コマンドを使用します。
	
	`wget -O ./aws-rep lication-installer- init.py \ https://aws-appl ication-migration- service-us-west-1. bucket.vpce-009c8b 07adb052a11-qgf8q5 0y.s3.us-west-1.vp ce.amazonaws.com/l atest/linux/aws-re plication-installe r-init.py`
	
	Note
	`bucket` は、Amazon S3 インターフェイスエンドポイント DNS 名の前に追加する必要がある静的キーワードです。詳細については、Amazon S3	

タスク	説明	必要なスキル
	のドキュメント を参照してください。 2. エージェントをインストールする： • アプリケーション移行サービスのインターフェイスエンドポイントを作成したときに DNS 名を有効にする を選択した場合、コマンドを実行します: `sudo python3 aws-replication-installer-init.py \ --region <aws_region> \ --aws-access-key-id <access-key> \ --aws-secret-access-key <secret-key> \ --no-prompt \ --s3-endpoint <s3-endpoint-DNS-name>` • アプリケーション移行サービスのインターフェイスエンドポイントを作成したときに DNS 名を有効にするを選択しなかった場合、コマンドを実行します	

タスク	説明	必要なスキル

	<pre>sudo python3 aws-replication-installer-init.py \
	--region <aws_region> \
	--aws-access-key-id <access-key> \
	--aws-secret-access-key <secret-key> \
	--no-prompt \
	--s3-endpoint <s3-endpoint-DNS-name> \
	--endpoint <mgn-endpoint-DNS-name></pre>
	詳細については、Application Migration Service ドキュメントのAWS 「レプリケーションエージェントのインストール手順」を参照してください。
	
	Application Migration Service との接続を確立し、 AWS レプリケーションエージェントをインストールしたら、Application Migration Service ドキュメントの指示に従って、ソースサーバーをターゲット VPC とサブネットに移行します。	

関連リソース

「アプリケーション移行サービスドキュメント」

- 「概念」
- 「移行ワークフロー」
- 「クイックスタートガイド」
- よくある質問
- トラブルシューティング

追加リソース

- VPC インターフェイスエンドポイント AWS を使用した のマルチアカウントアーキテクチャでのアプリケーションのリホスト (AWS 規範ガイダンスガイド)
- AWS Application Migration Service – 技術的な概要 (AWS トレーニングと認定のチュートリアル)
- AWS Application Migration Service アーキテクチャとネットワークアーキテクチャ (ビデオ)

追加情報

Linux サーバーでのレプリケーションエージェントのインストールのトラブルシューティング AWS

Amazon Linux サーバーで gcc エラーが発生した場合、パッケージリポジトリを設定し、以下のコマンドを使用します：

```
## sudo yum groupinstall "Development Tools"
```

AWS CloudFormation カスタムリソースと Amazon SNS を使用して Infoblox オブジェクトを作成します。

ティム・サットン (AWS) によって作成された

概要

注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら

Infoblox ドメインネームシステム (DNS)、Dynamic Host Configuration Protocol (DHCP)、IP アドレス管理 (Infoblox DDI) により、複雑なハイブリッド環境を一元化し、効率的に制御できます。Infoblox DDIを使用すると、すべてのネットワーク資産を検出して1つの信頼できるIPアドレス管理（IPAM）データベースに記録できます。また、同じアプライアンスを使用してオンプレミスとAmazon Web Services（AWS）クラウドでDNSを管理できます。

このパターンでは、AWS CloudFormation カスタムリソースを使用して Infoblox WAPI API を呼び出して Infoblox オブジェクト (DNS レコードや IP アドレス管理オブジェクトなど) を作成する方法を説明します。Infoblox WAPI の詳細については、Infoblox ドキュメントの「WAPI ドキュメント」を参照してください。

このパターンのアプローチを使用すると、レコードを作成してネットワークをプロビジョニングする手動プロセスを排除できるだけでなく、AWS 環境とオンプレミス環境のDNSレコードとIPAM構成を一元的に把握できます。このパターンのアプローチは、以下のユースケースに使用できます。

- Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成した後で A レコードを追加する
- Application Load Balancer の作成後の CNAME レコードの追加
- 仮想プライベートクラウド (VPC) 作成後のネットワークオブジェクトの追加
- 次のネットワーク範囲を指定し、その範囲を使用してサブネットを作成する

このパターンを拡張して、さまざまな DNS レコードタイプの追加や Infoblox vDiscovery の設定など、Infoblox デバイスの他の機能を使用することもできます。

このパターンでは、ハブが AWS クラウドまたはオンプレミスの Infoblox アプライアンスへの接続を必要とし、AWS Lambda を使用して Infoblox API を呼び出すハブアンドスポーク設計を採用しています。スポークは AWS Organizationsの同じまたは別のアカウントにあり、AWS CloudFormation カスタムリソースを使用して Lambda 関数を呼び出します。

前提条件と制限

前提条件

- AWS クラウド、オンプレミス、またはその両方にインストールされ、IPAM と DNS アクションを管理できる管理者ユーザーで構成された既存の Infoblox アプライアンスまたはグリッド。詳細については、Infoblox ドキュメントの「管理者アカウントについて」を参照してください。
- Infoblox アプライアンスにレコードを追加したい既存の DNS 権限ゾーン。詳細については、Infoblox ドキュメントの「権限のあるゾーンの設定」を参照してください。
- AWS Organizations 内の 2 つのアクティブな AWS アカウント。1 つのアカウントはハブアカウントで、もう 1 つのアカウントはスポークアカウントです。
- このハブとスポークアカウントは同じ AWS リージョンに存在する必要があります。
- ハブアカウントの VPC は、たとえば AWS Transit Gateway や VPC ピアリングを使用して Infoblox アプライアンスに接続する必要があります。
- 「AWS サーバーレスアプリケーションモデル (AWS SAM)」を、ローカルにインストールし、AWS Cloud9 または AWS CloudShell で設定します。
- `Infoblox-Hub.zip`および`ClientTest.yaml`ファイル (添付) は、AWS SAM を含むローカル環境にダウンロードされます。

機能制限

- AWS CloudFormation カスタムリソースのサービストークンは、作成しているスタックと同じリージョンのものである必要があります。あるリージョンで Amazon Simple Notification Service (Amazon SNS) トピックを作成し、別のリージョンで Lambda 関数を呼び出す代わりに、各リージョンでハブアカウントを使用することをお勧めします。

製品バージョン

- Infoblox API バージョン 2.7

アーキテクチャ

以下の図表に、このパターンのワークフローを示しています。

この図は、このパターンのソリューションを構成する以下のコンポーネントを示しています。

1. AWS CloudFormation カスタムリソースを使用すると、テンプレートにカスタムのプロビジョニングロジックを記述し、ユーザーがスタックを作成、更新、削除するたびに AWS CloudFormation がそれを実行します。スタックを作成するときに、AWS CloudFormation によって、EC2 インスタンスで実行しているアプリケーションが監視する SNS トピックに `create` リクエストを送信できます。
2. AWS CloudFormation カスタムリソースからの Amazon SNS 通知は、特定の AWS Key Management Service (AWS KMS) キーによって暗号化され、アクセスはOrganizations 内の組織内のアカウントに制限されます。SNS トピックは、Infoblox WAPI API を呼び出すLambda リソースを開始します。
3. Amazon SNS は、Infoblox WAPI URL、ユーザー名、およびパスワード AWS Secrets Manager Amazon リソースネーム (ARN) を環境変数として使用する次の Lambda 関数を呼び出します。
   - `dnsapi.lambda_handler`— AWS CloudFormation カスタムリソースから`DNSName`、`DNSType`、`DNSValue`の値を受け取り、それらを使用して DNS A レコードと CNAME を作成します。
   - `ipaddr.lambda_handler`— AWS CloudFormation カスタムリソースから`VPCCIDR`、`Type`、`SubnetPrefix`、`Network Name`の値を受け取り、それらを使用してネットワークデータを Infoblox IP アドレス管理データベースに追加したり、新しいサブネットの作成に使用できる次に使用可能なネットワークをカスタムリソースに提供したりします。
   - `describeprefixes.lambda_handler`— `"com.amazonaws."+Region+".s3"`フィルタを使用して`describe_managed_prefix_lists` AWS API を呼び出し、必要な`prefix ID`を取得します。

   > **Important**
   >
   > これらの Lambda 関数は Python で記述され、互いに似ていますが、異なる APIs。

4. Infoblox グリッドは、物理、仮想、またはクラウドベースのネットワークアプライアンスとしてデプロイできます。 オンプレミスで導入することも、VMware ESXi、Microsoft Hyper-V、Linux KVM、Xenなどのさまざまなハイパーバイザーを使用して仮想アプライアンスとして導入することもできます。Amazon マシンイメージ (AMI) を使用して Infoblox グリッドを AWS クラウドにデプロイすることもできます。
5. この図は、AWS クラウドとオンプレミスのリソースに DNS と IPAM を提供する Infoblox グリッドのハイブリッドソリューションを示しています。

テクノロジースタック

- AWS CloudFormation
- IAM
- AWS KMS
- AWS Lambda
- AWS SAM
- AWS Secrets Manager
- Amazon SNS
- Amazon VPC

ツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Organizations」は、複数の AWS アカウントを 1 つの組織に統合し、作成と一元管理するためのアカウント管理サービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- 「AWS サーバーレスアプリケーションモデル (AWS SAM)」 は、 AWS クラウドのサーバーレスアプリケーションを構築するために支援するオープンソースフレームワークです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されてい

た従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

コード

`ClientTest.yaml`サンプル AWS CloudFormation テンプレート (添付) を使用して Infoblox ハブをテストできます。AWS CloudFormation テンプレートをカスタマイズして、次の表のカスタムリソースを含めることができます。

Infoblox スポークカスタムリソースを使用して A レコードを作成します。	戻り値：
	`infobloxref` — Infoblox のリファレンス
	リソースの例：
	`ARECORDCustomResource:`
	`  Type: "Custom::InfobloxAPI"`
	`  Properties:`
	`  ServiceToken: !Sub arn:aws:sns:${AWS::Region}:${HubAccountID}:RunInfobloxDNSFunction`
	`  DNSName: 'arecordtest.company.com'`
	`  DNSType: 'ARecord'`
	`  DNSValue: '10.0.0.1'`

Infoblox スポークカスタムリソースを使用して CNAME レコードを作成します。	戻り値：
	`infobloxref` — Infoblox のリファレンス
	リソースの例：
	`CNAMECustomResource:`
	`  Type: "Custom::InfobloxAPI"`

```
  Properties:

    ServiceToken: !Sub arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfoblox

    DNSFunction

    DNSName: 'cnametest.company.com'

    DNSType: 'cname'

    DNSValue: 'aws.amazon.com'
```

Infoblox スポークカスタムリソースを使用してネットワークオブジェクトを作成します。

戻り値：

infobloxref — Infoblox のリファレンス

network— ネットワーク範囲 (と同じ) VPCCIDR

リソースの例：

```
VPCCustomResource:

  Type: 'Custom::InfobloxAPI'

  Properties:

    ServiceToken: !Sub  arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfobloxNextSubnetFunction

    VPCCIDR: !Ref VpcCIDR

    Type: VPC

    NetworkName: My-VPC
```

Infoblox スポークカスタムリソースを使用して、次に使用可能なサブネットを取得します。

戻り値：

`infobloxref` — Infoblox のリファレンス

`network` — サブネットのネットワーク範囲

リソースの例：

```
Subnet1CustomResource:

  Type: 'Custom::InfobloxAPI'

  DependsOn: VPCCustomResource

  Properties:

    ServiceToken: !Sub  arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfobloxNextSubnetFunction

    VPCCIDR: !Ref VpcCIDR

    Type: Subnet

    SubnetPrefix: !Ref SubnetPrefix

NetworkName: My-Subnet
```

エピック

ハブアカウントの VPC を作成して設定する

タスク	説明	必要なスキル
Infoblox アプライアンスに接続する VPC を作成します。	ハブアカウントの AWS マネジメントコンソールにサインインし、AWS クイックスタートの「AWS クラウドクイックスタートリファレンスデプロ	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	イの Amazon VPC」の手順に従って VPC を作成します。 **Important** VPC には Infoblox アプライアンスへの HTTPS 接続が必要です。この接続にはプライベートサブネットを使用することをお勧めします。	

タスク	説明	必要なスキル
(オプション) プライベートサブネット用の VPC エンドポイントを作成します。	VPC エンドポイントは、プライベートサブネットのパブリックサービスへの接続を提供します。次のエンドポイントが必要です。 • Lambda が AWS CloudFormation と通信できるようにする Amazon Simple Storage Service (Amazon S3) のゲートウェイエンドポイント • Secrets Manager との接続を有効にするSecrets Manager のインターフェースエンドポイント • SNS トピックと Secrets Manager シークレットの暗号化を可能にする AWS KMS のインターフェイスエンドポイント プライベートサブネット用のエンドポイント作成の詳細については、Amazon VPC ドキュメントの「VPC エンドポイント」を参照してください。	ネットワーク管理者、システム管理者

Infoblox ハブをデプロイします。

タスク	説明	必要なスキル
AWS SAM テンプレートをビルドします。	1. AWS SAM を含む環境でunzip Infoblox-Hub.zip コマンドを実行します。 2. cd Hub/コマンドを実行して、ディレクトリをHubディレクトリに変更します。 3. sam build コマンドは、AWS SAM テンプレートファイル、アプリケーションコード、および該当する言語固有のファイルと依存関係を処理します。このsam buildコマンドは、ワークフローの続きに期待される形式と場所へのビルドアーティファクトのコピーも実行します。	開発者、システム管理者
AWS SAM テンプレートをデプロイします。	sam deployコマンドは、必要なパラメータを取得してsamconfig.toml ファイルに保存し、AWS CloudFormation テンプレートと Lambda 関数を S3 バケットに保存し、次に AWS CloudFormation テンプレートをハブアカウントにデプロイします。 次のサンプルコードは、AWS SAM テンプレートをデプロイする方法を示しています。	開発者、システム管理者

タスク	説明	必要なスキル

	`$ sam deploy --guided`
	
	`Configuring SAM deploy`
	`==========`
	`==`
	`Looking for config file [samconfig.toml] : Found`
	`Reading default arguments : Success`
	`Setting default arguments for 'sam deploy'`
	`==========`
	`==========`
	`Stack Name [Infoblox-Hub]:`
	`AWS Region [eu-west-1]:`
	`Parameter InfobloxUsername:`
	`Parameter InfobloxPassword:`
	`Parameter InfobloxIPAddress [xxx.xxx.xx.xxx]:`
	`Parameter AWSOrganisationID [o-xxxxxxxxx]:`
	`Parameter VPCID [vpc-xxxxxxxxx]:`
	`Parameter VPCCIDR [xxx.xxx.xxx.xxx/16]:`
	`Parameter VPCSubnetID1 [subnet-xxx]:`
	`Parameter VPCSubnetID2 [subnet-xxx]:`	

タスク	説明	必要なスキル
	Parameter VPCSubnetID3 [subnet-xxx]: Parameter VPCSubnetID4 []: #Shows you resources changes to be deployed and require a 'Y' to initiate deploy Confirm changes before deploy [Y/n]: y #SAM needs permission to be able to create roles to connect to the resources in your template Allow SAM CLI IAM role creation [Y/n]: n Capabilities [['CAPABILITY_NAMED_IAM']]: Save arguments to configuration file [Y/n]: y SAM configuration file [samconfig.toml]: SAM configuration environment [default]: **Important** Infoblox サインイン認証情報は samconfig.toml ファイルに保存されないため、毎回 --guidedオプション	

タスク	説明	必要なスキル
	を使用する必要があります。	

関連リソース

- Postman を使用して API を使い始める (Infoblox ブログ)
- BYOL モデルを使用した AWS 向けの VNIO のプロビジョニング (Infoblox ドキュメント)
- クイックスタート-AWS-VPC (GitHub リポジトリ)
- describe_managed_prefix_lists (AWS SDK for Python)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

の Amazon CloudWatch アラートをカスタマイズする AWS Network Firewall

作成者：Jason Owens (AWS)

概要

このパターンは、 によって生成される Amazon CloudWatch アラートをカスタマイズするのに役立ちます AWS Network Firewall。事前定義されたルールを使用するか、アラートのメッセージ、メタデータ、重要度を決定するカスタムルールを作成できます。その後、これらのアラートに対応するか、Amazon EventBridge などの他のAmazonサービスによる対応を自動化することができます。

このパターンでは、Suricata 互換のファイアウォールルールを生成します。「Suricata」はオープンソースの脅威検出エンジンです。まず簡単なルールを作成し、次にそれらをテストして CloudWatch アラートが生成され、記録されていることを確認します。ルールを正常にテストしたら、ルールを変更してカスタムメッセージ、メタデータ、重要度を定義し、もう一度テストして更新を確認します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface （ AWS CLI) を Linux、macOS、または Windows ワークステーションにインストールして設定します。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。
- AWS Network Firewall CloudWatch Logs を使用するようにインストールおよび設定されています。詳細については、「 からのネットワークトラフィックの口グ記録 AWS Network Firewall」を参照してください。
- Network Firewall に保護された仮想プライベートクラウド (VPC) のプライベートサブネット内の Amazon Elastic Compute Cloud (Amazon EC2) インスタンス。

製品バージョン

- のバージョン 1 では AWS CLI、1.18.180 以降を使用します。のバージョン 2 では AWS CLI、2.1.2 以降を使用します。
- Suricata バージョン 5.0.2 の分類.config ファイル。この設定ファイルのコピーについては、「追加情報」セクションを参照してください。

アーキテクチャ

以下は、アーキテクチャ図を示しています。

1. プライベートサブネットの Amazon EC2 インスタンスは、curl または Wget を使用してリクエストを行います。
2. Network Firewall はトラフィックを処理し、アラートを生成します。
3. Network Firewall は、ログに記録されたアラートを CloudWatch Logs に送信します。

ツール

AWS のサービス

- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- Amazon CloudWatch Logs を使用すると、すべてのシステム、アプリケーション、および から のログを一元化 AWS のサービス できるため、ログをモニタリングして安全にアーカイブできます。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS Network Firewall は、 AWS クラウドの仮想プライベートクラウド (VPC) にステートフルなマネージドネットワークファイアウォールを形成する、侵入検知および防止サービスです。

その他のツール

- curl はオープンソースのコマンドラインツールおよびライブラリです。
- GNU Wget は無料のコマンドラインツールです。

エピック

ファイアウォールルールとルールグループを作成します。

タスク	説明	必要なスキル

ルールを作成します。	1. テキストエディタで、ファイアウォールに追加するルールのリストを作成します。各ルールは個別の行に記述する必要があります。`classtype` パラメーターの値はデフォルトの Suricata 分類設定ファイルからのものです。完全な設定ファイルコンテンツについては、「追加情報」セクションを参照してください。以下に 2 つの例を示します。
	`alert http any any -> any any (content:"badstuff"; classtype:misc-activity; sid:3; rev:1;)`
	`alert http any any -> any any (content:"morebadstuff"; classtype:bad-unknown; sid:4; rev:1;)`
	2. `custom.rules` というファイルにルールを保存します。
		AWS システム管理者、ネットワーク管理者

ルールグループを作成する	で AWS CLI、次のコマンドを入力します。これにより、	AWS システム管理者

タスク	説明	必要なスキル

	ルールグループが作成されます。
	
	`# aws network-firewall create-rule-group \ --rule-group-name custom --type STATEFUL \ --capacity 10 --rules file://custom.rules \ --tags Key=environment,Value=development`
	
	以下に出力例を示します。後のステップで必要になるので、 RuleGroupArn を書き留めておきます。
	
	`{ "UpdateToken": "4f998d72-973c-490a-bed2-fc3460547e23", "RuleGroupResponse": { "RuleGroupArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom", "RuleGroupName": "custom", "RuleGroupId": "238a8259-9eaf-48bb-90af-5e690cf8c48b", "Type": "STATEFUL", "Capacity": 10,`	

タスク	説明	必要なスキル

	<pre>"RuleGrou
	pStatus": "ACTIVE",
	"Tags": [
	{
	"Key":
	"environment",
	"Value":
	"development"
	}
	]
	}</pre>	

ファイアウォールポリシーを更新

タスク	説明	必要なスキル

ファイアウォールポリシーの ARN を取得します。	で AWS CLI、次のコマンドを入力します。ポリシーの Amazon リソースネーム (ARN) を返します。後のパターンで使用するために ARN を記録します。
	<pre># aws network-firewall
	describe-firewall \
	--firewall-name
	aws-network-firewall-
	anfw \
	--query 'Firewall
	.FirewallPolicyArn'</pre>
	以下はこのコマンドによって返される ARN の例を示しています。
	<pre>"arn:aws:network-f
	irewall:us-east-2:</pre>
		AWS システム管理者

タスク	説明	必要なスキル
	1234567890:firewal l-policy/firewall- policy-anfw"	

タスク	説明	必要なスキル
ファイアウォールポリシーを更新	テキストエディターで、次のコードを貼り付けます。前のエピックで記録した値に <RuleGroupArn> を置き換えてください。firewall-policy-anfw.json という名前でファイルを保存します。 `{ "StatelessDefaultActions": [ "aws:forward_to_sfe" ], "StatelessFragmentDefaultActions": [ "aws:forward_to_sfe" ], "StatefulRuleGroupReferences": [ { "ResourceArn": "<RuleGroupArn>" } ] }` で次のコマンドを入力します AWS CLI。このコマンドには、新しいルールを追加するための「更新トークン」が必要です。このトークンは、ポリシーを最後に取得してから変更されていないことを確認するために使用されます。	AWS システム管理者

タスク	説明	必要なスキル

	`UPDATETOKEN=(`aws network-firewall describe-firewall-policy \ --firewall-policy-name firewall-policy-anfw \ --output text --query UpdateToken`)`
	
	`aws network-firewall update-firewall-policy \ --update-token $UPDATETOKEN \ --firewall-policy-name firewall-policy-anfw \ --firewall-policy file://firewall-policy-anfw.json`	

タスク	説明	必要なスキル

ポリシーの更新を確認します。	(オプション) ルールが追加されたことを確認してポリシー形式を表示する場合は、 に次のコマンドを入力します AWS CLI。
	
	`# aws network-firewall describe-firewall-policy \ --firewall-policy-name firewall-policy-anfw \ --query FirewallPolicy`
	
	以下に出力例を示します。
	
	`{ "StatelessDefaultActions": [ "aws:forward_to_sfe" ], "StatelessFragmentDefaultActions": [ "aws:forward_to_sfe" ], "StatefulRuleGroupReferences": [ { "ResourceArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom" } ]`
		AWS システム管理者

タスク	説明	必要なスキル
	<pre>}</pre>	

テストアラート機能

タスク	説明	必要なスキル
テスト用のアラートを生成します。	1. ファイアウォール・サブネット内のテスト・ワークステーションにログインします。 2. アラートを生成するコマンドを入力します。たとえば、wget または curl を使用できます。 <pre>wget -U "badstuff" http://www.amazon.com -o /dev/null</pre> <pre>curl -A "morebadstuff" http://www.amazon.com -o /dev/null</pre>	AWS システム管理者
アラートが記録されていることを確認します。	1. CloudWatch コンソールを開きます。 2. 正しいロググループとストリームに移動します。詳細については、「CloudWatch Logs Insights に送信されたログデータの表示」(CloudWatch Logs ドキュメント) を参照してください。	AWS システム管理者

タスク	説明	必要なスキル

	3. 記録されたイベントが次に示す例のようになっていることを確認します。例には、アラートの関連部分のみ表示されています。
	
	例 1
	
	`"alert": { "action": "allowed", "signature_id": 3, "rev": 1, "signature": "", "category": "Misc activity", "severity": 3 }`
	
	例 2
	
	`"alert": { "action": "allowed", "signature_id": 4, "rev": 1, "signature": "", "category": "Potentially Bad Traffic", "severity": 2 }`	

ファイアウォールルールとルールグループを更新します。

タスク	説明	必要なスキル
ファイアウォールルールを更新します。	1. テキストエディタで、`custom.rules` ファイルを開きます。 2. 最初のルールを以下のように変更します。このルールはファイルの 1 行に入力する必要があります。 `alert http any any -> any any (msg:"Watch out - Bad Stuff!!"; content:"badstuff"; classtype:misc-activity; priority:2; sid:3; rev:2; metadata:custom-field-2 Danger!, custom-field More Info;)` これにより、ルールに以下の変更が生じます。 • 署名またはアラートに関するテキスト情報を提供する「msg」(Suricata ウェブサイト) 文字列を追加します。生成されたアラートでは、署名にマッピングされます。 • `misc-activity` のデフォルトの「優先度」(Suricata ウェブサイト) を 3 から 2 に調整します。各 `classtypes` の	AWS システム管理者

タスク	説明	必要なスキル
	デフォルト値については、「追加情報」セクションを参照してください。 • カスタム「メタデータ」(Suricata ウェブサイト) をアラートに追加します。これは署名に追加される追加情報です。キーバリューのペアを使用することをお勧めします。 • 「rev」(Suricata ウェブサイト) を 1 から 2 に変更します。これは署名のバージョンを表します。	

タスク	説明	必要なスキル

ルールグループを更新します。	で AWS CLI、次のコマンドを実行します。ファイアウォールポリシーの ARN のを使用します。これらのコマンドは更新トークンを取得し、ルールを変更してルールグループを更新します。
	`# UPDATETOKEN=(`aws network-firewall \ describe-rule-group \ --rule-group-arn arn:aws:network-firewall:us-east-2:123457890:stateful-rulegroup/custom \ --output text --query UpdateToken`)`
	`# aws network-firewall update-rule-group \ --rule-group-arn arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom \ --rules file://custom.rules \ --update-token $UPDATETOKEN`
	以下に出力例を示します。
	`{ "UpdateToken": "7536939f-6a1d-414c-96d1-bb28110996ed",`
		AWS システム管理者

タスク	説明	必要なスキル
	"RuleGroupResponse": { "RuleGroupArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom", "RuleGroupName": "custom", "RuleGroupId": "238a8259-9eaf-48bb-90af-5e690cf8c48b", "Type": "STATEFUL", "Capacity": 10, "RuleGroupStatus": "ACTIVE", "Tags": [{ "Key": "environment", "Value": "development" }] } }	

更新されたアラート機能のテスト

タスク	説明	必要なスキル
テスト用のアラートを生成します。	1. ファイアウォール・サブネット内のテスト・ワークステーションにログインします。 2. アラートを生成するコマンドを入力します。たと	AWS システム管理者

タスク	説明	必要なスキル
	えば、curl を使用できます。 `curl -A "badstuff" http://www.amazon.com -o /dev/null`	

タスク	説明	必要なスキル
アラートが変更されたことを確認します。	1. CloudWatch コンソールを開きます。 2. 正しいロググループとストリームに移動します。 3. 記録されたイベントが次に示す例のようなものであることを確認します。例には、アラートの関連部分のみ表示されています。 `"alert": { "action": "allowed", "signature_id": 3, "rev": 2, "signature": "Watch out - Bad Stuff!!", "category": "Misc activity", "severity": 2, "metadata": { "custom-field": [ "More Info" ], "custom-field-2": [ "Danger!" ] } }`	AWS システム管理者

関連リソース

リファレンス

- から Slack チャネル AWS Network Firewall にアラートを送信する (AWS 規範ガイダンス)
- Suricata AWS を使用した での脅威防止のスケーリング (AWS ブログ記事)
- のデプロイモデル AWS Network Firewall (AWS ブログ記事)
- 「スリカタメタキーワーク」 (Suricata のドキュメント)

チュートリアルと動画

- AWS Network Firewall ワークショップ

追加情報

以下は Suricata 5.0.2 の分類設定ファイルです。これらの分類はファイアウォールルールを作成するときに使用されます。

```
# config classification:shortname,short description,priority

config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1

# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
```

```
config classification: suspicious-login,An attempted login using a suspicious username
 was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual
 port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or
 event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web
 application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: inappropriate-content,Inappropriate Content was Detected,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and
 password,2

# Update
config classification: targeted-activity,Targeted Malicious Activity was Detected,1
config classification: exploit-kit,Exploit Kit Activity Detected,1
config classification: external-ip-check,Device Retrieving External IP Address
 Detected,2
config classification: domain-c2,Domain Observed Used for C2 Detected,1
config classification: pup-activity,Possibly Unwanted Program Detected,2
config classification: credential-theft,Successful Credential Theft Detected,1
config classification: social-engineering,Possible Social Engineering Attempted,2
config classification: coin-mining,Crypto Currency Mining Activity Detected,2
config classification: command-and-control,Malware Command and Control Activity
 Detected,1
```

Terraform を使用して AWS Wavelength ゾーンにリソースをデプロイする

作成者: Zahoor Chaudhrey (AWS) と Luca Iannario (AWS)

概要

AWS Wavelength は、マルチアクセスエッジコンピューティング (MEC) アプリケーションに最適化されたインフラストラクチャの構築に役立ちます。Wavelength Zones は、通信サービスプロバイダー (CSP) の 5G ネットワーク内に AWS コンピューティングおよびストレージサービスを埋め込む AWS インフラストラクチャデプロイです。5G デバイスからのアプリケーショントラフィックは、通信ネットワークを離れることなく、Wavelength Zones で実行されているアプリケーションサーバーに到達します。以下は、Wavelength を介したネットワーク接続を容易にします。

- Virtual Private Cloud (VPCs) – VPCs は、Wavelength Zone を含む複数のアベイラビリティーゾーンにまたがるように拡張 AWS アカウント できます。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスおよび関連サービスは、リージョン VPC の一部として表示されます。VPCsは Amazon Virtual Private Cloud (Amazon VPC) で作成および管理されます。
- キャリアゲートウェイ – キャリアゲートウェイは、Wavelength Zone のサブネットから CSP ネットワーク、インターネット、または CSP のネットワーク AWS リージョン を介した への接続を可能にします。キャリアゲートウェイには 2 つの目的があります。これにより、特定の場所の CSP ネットワークからのインバウンドトラフィックが許可され、通信ネットワークとインターネットへのアウトバウンドトラフィックが許可されます。

このパターンと関連する Terraform コードは、Amazon EC2 インスタンス、Amazon Elastic Block Store (Amazon EBS) ボリューム、VPCs、サブネット、キャリアゲートウェイなどのリソースを Wavelength Zone で起動するのに役立ちます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 統合開発環境 (IDE)
- ターゲット Wavelength Zone にオプトインする
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- Terraform バージョン 1.8.4 以降がインストールされている (Terraform ドキュメント)
- Terraform AWS Provider バージョン 5.32.1 以降、設定済み (Terraform ドキュメント)

- Git、インストール済み (GitHub)
- Amazon VPC、Wavelength、Amazon EC2 リソースを作成するアクセス許可

制約事項

すべての が Wavelength Zones AWS リージョン をサポートしているわけではありません。詳細については、「Wavelength ドキュメント」の「Available Wavelength Zones」を参照してください。

アーキテクチャ

次の図は、Wavelength Zone でサブネットと AWS リソースを作成する方法を示しています。Wavelength Zone にサブネットを含む VPCs は、キャリアゲートウェイに接続できます。キャリアゲートウェイを使用すると、次のリソースに接続できます。

- 通信事業者のネットワーク上の 4G/LTE および 5G デバイス。
- 一部の Wavelength Zone パートナーのワイヤレスアクセスを修正しました。詳細については、「マルチアクセス AWS Wavelength」を参照してください。
- パブリックインターネットリソースへのアウトバウンドトラフィック。

ツール

AWS のサービス

- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。
- AWS Wavelength は AWS クラウド 、インフラストラクチャを通信プロバイダーの 5G ネットワークに拡張します。これにより、モバイルデバイスやエンドユーザーに超低レイテンシーを提供するアプリケーションを構築できます。

その他のツール

- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンのコードは、GitHub の「Terraform AWS Wavelength を使用したインフラストラクチャの作成」リポジトリで入手できます。Terraform コードは、次のインフラストラクチャとリソースをデプロイします。

- VPC
- Wavelength Zone
- Wavelength Zone の恥骨サブネット
- Wavelength Zone のキャリアゲートウェイ
- Wavelength Zone の Amazon EC2 インスタンス

ベストプラクティス

- デプロイする前に、最新バージョンの Terraform と を使用していることを確認します AWS CLI。
- 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを使用して IaC をデプロイします。詳細については、 AWS ブログのAWS 「CI/CD Pipeline で Terraform State ファイルを管理するためのベストプラクティス」を参照してください。

エピック

インフラストラクチャをプロビジョニングする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	次のコマンドを入力して、Terraform AWS Wavelength を使用してインフラストラクチャを作成するリポジトリを環境にクローンします。 `git clone git@github.com:aws-samples/terraform-waveleng`	DevOps エンジニア

タスク	説明	必要なスキル
	`th-infrastructure.git`	
変数を更新してください。	1. クローンされたリポジトリに移動します。 `cd terraform-wavelength-infrastructure` 2. 任意のテキストエディタを使用して、ルートディレクトリに terraform.tfvars という名前のファイルを作成します。 3. 次の変数を作成し、値を入力します。 • `region = <enter Region name>` • `vpc_cidr = <enter CIDR block used by VPC>` • `wavelength_subnet_cidr = <enter CIDR block for the subnet in the Wavelength Zone>` • `availabilityzone_wavelength = <enter Wavelength Zone name>` 4. terraform.tfvars ファイルを保存します。	DevOps エンジニア、Terraform

タスク	説明	必要なスキル
設定を初期化します。	次のコマンドを入力して、作業ディレクトリを初期化します。 `terraform init`	DevOps エンジニア、Terraform
Terraform プランをプレビューしてください。	次のコマンドを入力して、ターゲットの状態を AWS 環境の現在の状態と比較します。このコマンドは、設定されるリソースのプレビューを生成します。 `terraform plan`	DevOps エンジニア、Terraform
検証してデプロイします。	1. Terraform プランの設定変更を確認し、変更を実装することを確認します。 2. 次のコマンドを入力してプランを適用し、インフラストラクチャを作成します。 `terraform apply` 3. 「」と入力yesして続行します。Terraform は、設定ファイルで宣言されたアーキテクチャを作成します。アーキテクチャの詳細については、このパターンの「ターゲットアーキテクチャ」セクションを参照してください。	DevOps エンジニア、Terraform

検証とクリーンアップ

タスク	説明	必要なスキル
インフラストラクチャのデプロイを確認します。	1. のパブリックサブネットに Amazon EC2 インスタンスがまだない場合は AWS リージョン、インスタンスを作成します。手順については、「Linux インスタンスを起動する」または「Windows インスタンスを起動する」を参照してください。このインスタンスを使用して、 から Wavelength Zone AWS リージョン への接続をテストします。 2. のインスタンスから Wavelength Zone のインスタンス AWS リージョンへの接続をテストします。手順については、Wavelength ドキュメントの「接続のテスト」を参照してください。	AWS DevOps、DevOps エンジニア
(オプション) インフラストラクチャをクリーンアップします。	Terraform によってプロビジョニングされたすべてのリソースを削除する必要がある場合は、次の手順を実行します。 1. 次のコマンドを入力します。 `terraform destroy`	DevOps エンジニア、Terraform

タスク	説明	必要なスキル
	2. yes を入力して確定します。	

トラブルシューティング

問題	ソリューション
の Amazon EC2 インスタンスへの接続 AWS リージョン。	「Linux インスタンスへの接続のトラブルシューティング」または「Windows インスタンスへの接続のトラブルシューティング」を参照してください。
Wavelength Zone の Amazon EC2 インスタンスへの接続。	「Wavelength Zone で起動された EC2 インスタンスへの SSH または RDP 接続のトラブルシューティング」を参照してください。
Wavelength Zone の容量。	「Wavelength Zones のクォータと考慮事項」を参照してください。
キャリアネットワークから へのモバイルまたはキャリア接続 AWS リージョン。	1. キャリアゲートウェイが動作していることを確認します。以下の操作を実行します。 a. Amazon VPC コンソールを開きます。 b. ナビゲーションペインで、[Your VPCs (お使いの VPC)] を選択します。 c. Wavelength Zone を含む VPC を選択します。 d. 詳細ペインのキャリアゲートウェイで、値がアタッチされていることを確認します。 2. Wavelength Zone のインスタンスにアタッチされた Elastic IP アドレスが動作していることを確認します。以下の操作を実行します。 a. Amazon EC2 コンソールを開きます。

問題	ソリューション
	b. ナビゲーションペインで、[インスタンス] を選択します。 c. Wavelength Zone でインスタンスを選択します。 d. [ネットワーク] タブを選択します。 e. Elastic Network Interface に Elastic IP アドレスがアタッチされていることを確認します。 3. キャリアネットワークサポートチームにお問い合わせください。

関連リソース

- とは AWS Wavelength
- の AWS Wavelength 仕組み
- の耐障害性 AWS Wavelength

DNS レコードを Amazon Route 53 プライベートホストゾーンに一括で移行する

作成者: Ram Kandaswamy (AWS)

概要

ネットワークエンジニアやクラウド管理者は、ドメインネームシステム (DNS) レコードを Amazon Route 53 のプライベートホストゾーンに追加する効率的で簡単な方法を必要としていました。Microsoft Excel ワークシートのエントリを Route 53 コンソールの適切な場所に手動でコピーする方法は、手間がかかり、エラーが発生しやすくなります。このパターンは、複数のレコードを追加するのに必要な時間と労力を削減する自動化されたアプローチについて説明します。また、複数のホストゾーンを作成する手順を繰り返すこともできます。

このパターンでは、Amazon Simple Storage Service (Amazon S3) を使用してレコードを保存します。データを効率的に操作するため、このパターンでは JSON 形式が使用されます。これは、シンプルさと Python ディクショナリ (`dict` データ型) のサポート機能が理由です。

> **Note**
>
> システムからゾーンファイルを生成できる場合は、代わりに Route 53 インポート機能の使用を検討してください。

前提条件と制限

前提条件

- プライベートホストゾーンのレコードを含む Excel ワークシート
- A レコード、 名前付け権限ポインタ (NAPTR) レコード、SRV レコードなど、さまざまな種類の DNS レコードを理解していること (「サポートされる DNS レコードタイプ」を参照)
- Python 言語とそのライブラリを理解していること

制約事項

- このパターンは、すべてのユースケースシナリオを網羅しているわけではありません。たとえば、change_resource_record_sets 呼び出しでは API の使用可能なプロパティがすべて使用されるわけではありません。

- Excel ワークシートでは、各行の値は固有であると想定されます。各完全修飾ドメイン名 (FQDN) の複数の値が同じ行に表示されることが予想されます。　そうでない場合は、このパターンで提供されるコードを修正して、必要な連結を実行する必要があります。
- このパターンでは、AWS SDK for Python (Boto3) を使用して、Route 53 サービスを直接呼び出します。コードを拡張して、`create_stack` および `update_stack` コマンドに AWS CloudFormation ラッパーを使用し、JSON 値を使用してテンプレートリソースを設定することができます。

アーキテクチャ

テクノロジースタック

- トラフィックをルーティングするための Route 53 プライベートホストゾーン
- 出力 JSON ファイルを保存するための Amazon S3

このワークフローは、前の図と「エピック」セクションで説明したように、以下のステップで構成されています。

1. レコードセット情報を含む Excel ワークシートを S3 バケットにアップロードします。
2. Excel データを JSON 形式に変換する Python スクリプトを作成して実行します。
3. S3 バケットのレコードを読み取り、データをクリーンアップします。
4. プライベートホストゾーンにレコードセットを作成します。

ツール

- Route 53 — Amazon Route 53 は、ドメイン登録、DNS ルーティング、ヘルスチェックを処理する、可用性が高くスケーラブルな DNS ウェブサービスです。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、オブジェクトストレージサービスです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。

エピック

データ自動化の準備

タスク	説明	必要なスキル
レコード用の Excel ファイルを作成します。	現在のシステムからエクスポートしたレコードを使用して、完全修飾ドメイン名 (FQDN)、レコードタイプ、有効期間 (TTL)、値など、レコードに必要な列を含む Excel ワークシートを作成します。 NAPTR レコードと SRV レコードの場合、値は複数のプロパティの組み合わせなので、Excel の `concat` メソッドを使用してこれらのプロパティを組み合わせてください。 FQDN Recor Value TTL e somet A 1.1.1.1 900 .exam org	データエンジニア、Excel のスキル
作業環境を確認します。	IDE で、Python ファイルを作成して Excel 入力ワークシートを JSON 形式に変換します。(IDE の代わりに、Amazon SageMaker ノートブックを使用して Python コードを使用することもできます)。	AWS 全般

タスク	説明	必要なスキル

	Python のバージョン 3.7 以降を使用していることを確認してください。
	`python3 --version`
	pandas パッケージをインストールします。
	`pip3 install pandas --user`	

Excel ワークシートのデータを JSON に変換します。	Excel から JSON に変換し、次のコードを含む Python ファイルを作成します。
	`import pandas as pd`
	`data=pd.read_excel('./Book1.xls')`
	`data.to_json(path_or_buf='my.json', orient='records')`
	ここで Book1 は Excel ワークシートの名前、`my.json` は出力 JSON ファイルの名前です。
		データエンジニア、Python スキル

S3 バケットに JSON ファイルをアップロードします。	S3 バケットに `my.json` ファイルをアップロードします。詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	アプリ開発者

レコードを挿入します

タスク	説明	必要なスキル

プライベートホストゾーンを作成します。	create_hosted_zone API と次の Python サンプルコードを使用して、プライベートホストゾーンを作成します。パラメータ hostedZoneName 、vpcRegion 、および vpcId をユーザー自身の値に置き換えてください。
	
	`import boto3`
	`import random`
	`hostedZoneName ="xxx"`
	`vpcRegion = "us-east-1"`
	`vpcId="vpc-xxxx"`
	`route53_client = boto3.client('route53')`
	`response = route53_client.create_hosted_zone(`
	`Name= hostedZoneName,`
	`VPC={`
	`'VPCRegion: vpcRegion,`
	`'VPCId': vpcId`
	`},`
	`CallerReference=str(random.random()*100000),`
	`HostedZoneConfig={`
	`'Comment': "private hosted zone`
		クラウドアーキテクト、ネットワーク管理者、Python のスキル

タスク	説明	必要なスキル

	<pre>created by automation",
	'PrivateZone': True
	}
	)
	print(response)</pre>
	AWS CloudFormation などの Infrastructure as Code (IaC) ツールを使用すれば、適切なリソースとプロパティを含むスタックを作成するテンプレートでこれらの手順を置き換えることもできます。	

タスク	説明	必要なスキル
Amazon S3 から詳細をディクショナリとして取得します。	以下のコードを使用して S3 バケットから読み取り、JSON 値を Python ディクショナリとして取得します。 `fileobj = s3_client .get_object( Bucket=bucket_name, Key='my.json' ) filedata = fileobj['Body'].read() contents = filedata.decode('utf-8') json_content=json.loads(contents) print(json_content )` `json_content` には、Pythonディクショナリが含まれます。	アプリ開発者、Python のスキル

タスク	説明	必要なスキル

スペースと Unicode 文字のデータ値を消去します。	データが正しいことを確認するための安全対策として、json_content コードを使用して値を削除してください。このコードでは、各文字列の先頭と末尾のスペース文字が削除されます。 また、replace メソッドを使用して (改行しない) ハードスペース (\xa0 文字) を削除します。
	
	`for item in json_content:`
	`  fqn_name = unicodedata.normalize("NFKD",item["FqdnName"].replace("u'", "'").replace('\xa0', '').strip())`
	`  rec_type = item["RecordType"].replace('\xa0', '').strip()`
	`  res_rec = {`
	`  'Value': item["Value"].replace('\xa0', '').strip()`
	`  }`
		アプリ開発者、Python のスキル

タスク	説明	必要なスキル
レコードを挿入します	for ループの一部として次のコードを使用します。 `change_response = route53_client.change_resource_record_sets( HostedZoneId="xxxxxxxx", ChangeBatch={ 'Comment': 'Created by automation', 'Changes': [ { 'Action': 'UPSERT', 'ResourceRecordSet': { 'Name': fqn_name, 'Type': rec_type, 'TTL': item["TTL"], 'ResourceRecords': res_rec } } ] } )`	アプリ開発者、Python のスキル

タスク	説明	必要なスキル
	ここで xxxxxxx は、このエピックの最初のステップであるホストゾーン ID です。	

関連リソース

リファレンス

- 「ゾーンファイルをインポートしてレコードを作成する」(Amazon Route 53 ドキュメント)
- 「create_hosted_zone メソッド」(Boto3 ドキュメント)
- 「change_resource_record_sets メソッド」(Boto3 ドキュメント)

チュートリアルと動画

- 「Python チュートリアル」(Python ドキュメント)
- 「Amazon Route 53 を使用した DNS 設計」(YouTube ビデオ、AWS オンライン Tech Talks)

F5 から AWS のApplication Load Balancer に移行するときの HTTP ヘッダーを変更

作成者：Sachin Trivedi (AWS)

概要

F5 ロードバランサーを使用するアプリケーションをAmazon Web Services (AWS) に移行し、AWSでApplication Load Balancer を使用したい場合、ヘッダー変更の F5 ルールを移行することがよくある問題です。Application Load Balancer はヘッダーの変更をサポートしていませんが、Amazon CloudFront をコンテンツ配信ネットワーク (CDN) として使用し、Lambda @Edge により、ヘッダーを変更できます。

このパターンは、必要なインテグレーションを説明し、AWS CloudFront と Lambda @Edge により、ヘッダーを変更するためのサンプルコードを提供します。

前提条件と制限

前提条件

- F5 ロードバランサーを使用するオンプレミスアプリケーションで、HTTP ヘッダー値を `if, else` により、置き換える設定になっています。この設定の詳細については、F5 製品ドキュメントの「HTTP:: header」を参照してください。

機能制限

- このパターンは F5 ロードバランサーのヘッダーのカスタマイズに適用されます。他の第三者ロードバランサーについては、ロードバランサーのドキュメントでサポート情報を確認してください。
- Lambda@Edge に使用する Lambda 関数は、米国東部 (バージニア北部) リージョンにある必要があります。

アーキテクチャ

次の図は、CDN と他の AWS コンポーネント間の統合フローを含む AWS のアーキテクチャを示しています。

ツール

AWS サービス

- 「Application Load Balancer」– Application Load Balancer は、開放型システム間相互接続 (OSI) モデルの第 7 層で機能する AWS フルマネージド型負荷分散サービスです。複数のターゲット間でトラフィックを分散し、HTTP ヘッダーとメソッド、クエリ文字列、ホストベースまたはパスベースのルーティングに基づく高度なルーティングリクエストをサポートします。
- 「Amazon CloudFront」— Amazon CloudFront は、ユーザーへの静的および動的なウェブコンテンツ (.html、.css、.js、イメージファイルなど) の配信を高速化するウェブサービスです。CloudFront では、エッジロケーションというデータセンターの世界的ネットワークを経由してコンテンツを配信し、レイテンシーを短縮し、パフォーマンスを向上させます。
- 「Lambda@Edge」— Lambda@Edge は、CloudFront が配信するコンテンツをカスタマイズする関数を実行できる AWS Lambda の拡張機能です。米国東部 (バージニア北部) リージョンで作成した関数を CloudFront ディストリビューションに関連付けると、サーバーのプロビジョニングや管理の必要もなく、コードを世界中に自動的に複製できます。これにより、待ち時間が短縮され、ユーザーエクスペリエンスが向上します。

コード

次のサンプルコードは、CloudFront レスポンスヘッダーを変更するためのブループリントを提供します。[エピック] セクションの指示に従い、コードをデプロイします。

```
exports.handler = async (event, context) => {
    const response = event.Records[0].cf.response;
    const headers = response.headers;


    const headerNameSrc = 'content-security-policy';
    const headerNameValue = '*.xyz.com';


    if (headers[headerNameSrc.toLowerCase()]) {
        headers[headerNameSrc.toLowerCase()] = [{
            key: headerNameSrc,
            value: headerNameValue,
        }];
        console.log(`Response header "${headerNameSrc}" was set to ` +
                    `"${headers[headerNameSrc.toLowerCase()][0].value}"`);
    }
```

```
    else {
            headers[headerNameSrc.toLowerCase()] = [{
            key: headerNameSrc,
            value: headerNameValue,
            }];
    }
    return response;
};
```

エピック

CDN ディストリビューションの作成

タスク	説明	必要なスキル
CloudFront ウェブディストリビューションを作成します。	このステップでは、CloudFront ディストリビューションを作成して、コンテンツの配信元とコンテンツ配信の追跡および管理方法の詳細を CloudFront に伝えます。 コンソールを使用してディストリビューションを作成するには、AWS マネジメントコンソールにサインインし、「CloudFront コンソール」を開き、「CloudFront のドキュメント」の手順に従います。	クラウド管理者

Lambda@Edge 関数の作成とデプロイ

タスク	説明	必要なスキル
Lambda@Edge 関数の作成とデプロイ	CloudFront レスポンスヘッダーを変更するためのブループリントを使用して Lambda@Edge 関数を作成できます	AWS 管理者

タスク	説明	必要なスキル
	。(他のブループリントもさまざまなユースケースで利用できます。詳細については、「CloudFront のドキュメント」の「Lambda @Edge サンプル関数」を参照してください。) Lambda@Edge 関数を作成するには 1. AWS マネジメントコンソールにサインインして AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。 2. 米国東部 (バージニア北部) リージョンにいることを確認してください。CloudFront ブループリントは、このリージョンでのみ利用可能です。 3. [Create function (関数の作成)] を選択します。 4. ブループリントを使用を選択し、ブループリント検索フィールドに cloudfront を入力します。 5. cloudfront-modify-response-header ブループリントを選択して、 [設定] を選択します。 6. [基本情報] ページで、次の情報を入力します。	

タスク	説明	必要なスキル
	a. 関数名を入力します。 b. [Execution role (実行ロール)] ドロップダウンリストで、[Create new role from AWS policy templates (AWS ポリシーテンプレートから新しいロールを作成)] を選択します。 c. AWS Identity and Access Management (IAM) ロールとオプションを関連付けます。 7. [Create function (関数の作成)] を選択します。 8. ページの [デザイナー] セクションで、関数名を選択します。 9. [関数コード] セクションで、テンプレートコードを、このパターンの [コード] セクションで以前に提供したサンプルコードに置き換えます。 10.サンプルコードでは、ドメイン名を `xyz.com` に置き換えます。 11.[Save] を選択します。	

タスク	説明	必要なスキル
Lambda@Edge 関数をデプロイします。	Amazon CloudFront のドキュメントのチュートリアル:簡単な Lambda @Edge 関数の作成の「ステップ 4」の手順に従い、CloudFront トリガーを設定し、関数をデプロイします。	AWS 管理者

関連リソース

CloudFront のドキュメント

- 「カスタムオリジンのリクエストとレスポンスの動作」
- 「ディストリビューションの使用」
- 「Lambda@Edge 関数の例」
- 「Lambda@Edge を使用したエッジでのカスタマイズ」
- 「チュートリアル:シンプルな Lambda@Edge 関数の作成」

複数の VPC から中央のAWS のサービスエンドポイントにプライベートにアクセスする

マーティン・グエントナー（AWS）とサミュエル・ゴードン（AWS）によって作成されました

概要

ユーザー環境のセキュリティとコンプライアンスの要件により、Amazon Web Services (AWS) のサービスまたはエンドポイントへのトラフィックがパブリックインターネットを経由してはならないことが指定されている場合があります。このパターンは、中央のハブ VPC が複数の分散スポーク VPCs に接続されるハブアンドスポークトポロジー向けにデザインされたソリューションです。このソリューションでは、AWS PrivateLink を使用して、ハブアカウントの AWS のサービスのインターフェイス VPC エンドポイントを作成します。次に、トランジットゲートウェイと分散型ドメインネームシステム (DNS) ルールを使用して、接続されている VPC 全体でエンドポイントのプライベート IP アドレスへのリクエストを解決します。

このパターンでは、AWS Transit Gateway、着信の Amazon Route 53 Resolver エンドポイント、および共有 Route 53 転送ルールを使用して、接続されている VPC 内のリソースからの DNS クエリを解決する方法を説明します。エンドポイント、トランジットゲートウェイ、リゾルバー、転送ルールはハブアカウントで作成します。次に、AWS Resource Access Manager (AWS RAM) を使用して、トランジットゲートウェイと転送ルールをスポーク VPC と共有します。提供されている AWS CloudFormation テンプレートは、ハブ VPC とスポーク VPC にリソースをデプロイして設定するのに役立ちます。

前提条件と制限

前提条件

- AWS Organizations の同じ組織で管理されている 1 つのハブアカウントと 1 つ以上のスポークアカウント。詳細については、「組織の作成と管理」を参照してください。
- AWS Resource Access Manager (AWS RAM) は、AWS Organizations の信頼されたサービスとして設定されています。詳細については、「Using AWS Organizations with other AWS services」(他の AWS のサービスでの AWS Organizations の使用) を参照してください。
- ハブアンドスポーク VPC で DNS 解決を有効にする必要があります。詳細については、「PVCの DNS 属性」(Amazon 仮想プライベートクラウドドキュメント）を参照してください。

制約事項

- このパターンは、同じ AWS リージョンのハブアカウントとスポークアカウントを接続します。マルチリージョンデプロイでは、リージョンごとにこのパターンを繰り返す必要があります。
- AWS のサービスは、インターフェイス VPC エンドポイントとして PrivateLink と統合する必要があります。詳細なリストについては、「AWS PrivateLink と統合する AWS のサービス」(PrivateLink ドキュメント) を参照してください。
- アベイラビリティーゾーンのアフィニティは保証されません。たとえば、アベイラビリティーゾーン A からのクエリは、アベイラビリティーゾーン B からの IP アドレスで応答する場合があります。
- VPC エンドポイントに関連付けられた Elastic Network Interface には、1 秒あたり 10,000 クエリの制限があります。

アーキテクチャ

ターゲットテクノロジースタック

- ハブ AWS アカウントのハブ VPC
- スポーク AWS アカウント内の 1 つ以上のスポーク VPC
- ハブアカウントの 1 つ以上のインターフェイス VPC エンドポイント
- ハブアカウントの着信と発信のルート 53 リゾルバー
- ハブアカウントにデプロイされ、スポークアカウントと共有されるルート 53 Resolver 転送ルール
- ハブアカウントにデプロイされ、スポークアカウントと共有されるトランジットゲートウェイ
- ハブとスポーク VPC を接続する AWS Transit Gateway

ターゲットアーキテクチャ

次のイメージは、このソリューションのサンプルアーキテクチャを示しています。このアーキテクチャでは、ハブアカウントの ルート 53 Resolver 転送ルールは、他のアーキテクチャコンポーネントと次のような関係にあります。

1. 転送ルールは AWS RAM を使用してスポーク VPC と共有されます。
2. 転送ルールは、ハブ VPC の発信リゾルバーに関連付けられています。
3. 転送ルールは、ハブ VPC の着信リゾルバーを対象としています。

次のイメージは、サンプルアーキテクチャを経由するトラフィックフローを示しています。

1. スポーク VPC の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスなどのリソースは、`<service>.<region>.amazonaws.com` に DNS リクエストを行います。リクエストはスポーク Amazon DNS リゾルバーによって受信されます。
2. ハブアカウントから共有され、スポーク VPC に関連付けられているルート 53 転送ルールがリクエストをインターセプトします。
3. ハブ VPC では、発信リゾルバーは転送ルールを使用してリクエストを着信リゾルバーに転送します。
4. 着信リゾルバーは、ハブ VPC Amazon DNS リゾルバーを使用して IP アドレスを VPC エンドポイントのプライベート IP `<service>.<region>.amazonaws.com` アドレスに解決します。VPC エンドポイントが存在しない場合は、パブリック IP アドレスに解決されます。

ツール

AWS のサービスとツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。仮想サーバーを必要な数だけ起動して、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Resource Access Manager (AWS RAM) は、 アカウント全体にわたり、リソースを安全に共有して運用上のオーバーヘッドを削減して、可視性と監査性を高めます。
- Amazon Route 53 は、可用性と拡張性に優れたドメインネームシステム (DNS) ウェブサービスです。
- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。
- AWS Transit Gateway は VPC とオンプレミスネットワークを接続する一元的ハブです。

- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

その他のツールとサービス

- 「nslookup」 は DNS レコードのクエリに使用されるコマンドラインツールです。このパターンでは、このツールを使用してソリューションをテストします。

コードリポジトリ

このパターンのコードは、GitHub 内の「vpc-endpoint-sharing」リポジトリで利用できます。このパターンには、次の 2 つの AWS CloudFormation テンプレートが用意されています。

- ハブアカウントに以下のリソースをデプロイするためのテンプレート。
  - `rSecurityGroupEndpoints` — VPC エンドポイントへのアクセスを制御するセキュリティグループ。
  - `rSecurityGroupResolvers` — ルート 53 リゾルバーへのアクセスを制御するセキュリティグループ。
  - `rKMSEndpoint`、`rSSMMessagesEndpoint`、`rSSMEndpoint`、`rEC2MessagesEndpoint` — ハブアカウントのインターフェイス VPC エンドポイントの例 ユースケースに合わせてこれらのエンドポイントをカスタマイズします。
  - `rInboundResolver` — ハブの Amazon DNS リゾルバーに対して DNS クエリを解決する ルート 53 リゾルバー。
  - `rOutboundResolver` — クエリを着信リゾルバーに転送する発信ルート 53 リゾルバー。
  - `rAWSApiResolverRule` — すべてのスポーク VPC と共有される ルート 53 リゾルバーの転送ルール。
  - `rRamShareAWSResolverRule` — スポーク VPC `rAWSApiResolverRule` が転送ルールを使用できるようにする AWS RAM 共有。
  - * `rVPC` — 共有サービスのモデル化に使用されるハブ VPC。
  - * `rSubnet1` — ハブリソースを格納するプライベートサブネット。
  - * `rRouteTable1` — ハブ VPC のルートテーブル。
  - * `rRouteTableAssociation1` — ハブ VPC `rRouteTable1` のルートテーブルの場合、プライベートサブネットのアソシエーション。

  - * rRouteSpoke — ハブ VPC からスポーク VPC へのルート。
  - * rTgw — すべてのスポーク VPC と共有されるトランジットゲートウェイ。
  - * rTgwAttach — ハブ VPC rTgw がトラフィックをトランジットゲートウェイにルーティングできるようにするアタッチメント。
  - * rTgwShare — rTgw スポークアカウントがトランジットゲートウェイを使用できるようにする AWS RAM 共有。
- スポークアカウントに以下のリソースをデプロイするためのテンプレート。
  - rAWSApiResolverRuleAssociation — スポーク VPC がハブアカウントの共有転送ルールを使用できるようにするアソシエーション。
  - * rVPC — スポーク VPC。
  - * rSubnet1, rSubnet2, rSubnet3 — スポークプライベートリソースを格納するために使用される各アベイラビリティーゾーンのサブネット。
  - * rTgwAttach — スポーク VPC がトラフィックを rTgw トランジットゲートウェイにルーティングできるようにするアタッチメント。
  - * rRouteTable1 — スポーク VPC のルートテーブル。
  - * rRouteEndpoints — スポーク VPC 内のリソースからトランジットゲートウェイへのルート。
  - * rRouteTableAssociation1/2/3 — スポーク VPC の rRouteTable1 に対して、プライベートサブネットのアソシエーション。
  - * rInstanceRole — ソリューションのテストに使用される IAM ロール。
  - * rInstancePolicy — ソリューションをテストするために使用される IAM ポリシー。
  - * rInstanceSg — ソリューションのテストに使用されたセキュリティグループ。
  - * rInstanceProfile — ソリューションのテストに使用された IAM インスタンスプロファイル。
  - * rInstance — AWS Systems Manager を介してアクセスできるように事前設定された EC2 インスタンス。このインスタンスを使用してソリューションをテストします。

* これらのリソースはサンプルアーキテクチャをサポートしているため、既存のランディングゾーンにこのパターンを実装する場合は必要ない場合があります。

エピック

CloudFormation テンプレートの準備

タスク	説明	必要なスキル
コードリポジトリを複製します。	1. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 2. 次のコマンドを入力します。 `git clone https://github.com/aws-samples/vpc-endpoint-sharing.git`	ネットワーク管理者、クラウドアーキテクト
テンプレートを変更します。	1. クローンしたリポジトリで hub.yml ファイルと spoke.yml ファイルを開きます。 2. これらのテンプレートにより、作成されたリソースを確認し、環境に合わせて必要に応じてテンプレートを調整してください。詳細なリストについては、「ツール」 のコードリポジトリセクションを参照してください。アカウントにすでにこれらのリソースがある場合は、CloudFormation テンプレートから削除します。詳細については、「 Working with templates」 (CloudFor	ネットワーク管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	mation ドキュメント) を参照してください。 3. hub.yml ファイルと spoke.yml ファイルを保存して閉じます。	

ターゲットアカウントにリソースをデプロイします。

タスク	説明	必要なスキル
ハブリソースをデプロイする。	hub.yml テンプレートを使用して、CloudFormation スタックを作成します。提示された場合、テンプレートのパラメータに値を提供します。詳細については、「スタックの作成」(CloudFormation ドキュメント) を参照してください。	クラウドアーキテクト、ネットワーク管理者
スポーク・リソースをデプロイします。	spoke.yml テンプレートを使用して、CloudFormation スタックを作成します。提示された場合、テンプレートのパラメータに値を提供します。詳細については、「スタックの作成」(CloudFormation ドキュメント) を参照してください。	クラウドアーキテクト、ネットワーク管理者

ソリューションをテストする

タスク	説明	必要なスキル
AWS のサービスへのプライベート DNS クエリをテストします。	1. AWS Systems Manager の一機能であるセッションマネージャーを使用して `rInstance` EC2 インスタンスConnect。詳細については、「セッションマネージャーを使用して Linux インスタンスへのConnect」(Amazon EC2 ドキュメント) を参照してください。 2. ハブアカウントに VPC エンドポイントがある AWS のサービスの場合は、`nslookup` を使用して、着信の ルート 53 Resolver のプライベート IP アドレスが返されることを確認します。 Amazon Systems Manager `nslookup` エンドポイントに到達するための使用例を次に示します。 `nslookup ssm.<region>.amazonaws.com` 3. AWS コマンドラインインターフェイス (AWS CLI) で、変更がサービスの機能に影響を及ぼさなかったことを確認するのに役立つコマンドを入力します。コ	ネットワーク管理者

タスク	説明	必要なスキル
	マンドの一覧については、「AWS CLI コマンドリファレンス」を参照してください。 たとえば、次のコマンドでは、Amazon Systems Manager のドキュメントのリストが返されます。 `aws ssm list-documents`	

タスク	説明	必要なスキル

AWS のサービスへのパブリック DNS クエリをテストします。	1. ハブアカウントに VPC エンドポイントがない AWS のサービスの場合は、nslookup を使用してパブリック IP アドレスが返されることを確認します。Amazon Simple Notification Service (Amazon SNS) nslookup エンドポイントに到達するための使用例を次に示します。
	`nslookup sns.<region>.amazonaws.com`
	2. AWS CLI では、変更がサービスの機能に影響を及ぼさなかったことを確認するのに役立つコマンドを入力します。コマンドの一覧については、「AWS CLI コマンドリファレンス」を参照してください。
	たとえば、ハブアカウントに Amazon SNS トピックが存在する場合、次のコマンドはトピックのリストを返します。
	`aws sns list-topics`
		ネットワーク管理者

関連リソース

- スケーラブルでセキュアなマルチ VPC の AWS ネットワークインフラストラクチャを構築する (AWS ホワイトペーパー)
- 「共有リソースの使用」 (AWS RAM ドキュメント)
- 「トランジットゲートウェイの操作」 (AWS Transit Gateway ドキュメント)

複数の でのインバウンドインターネットアクセスに関する Network Access Analyzer の検出結果のレポートを作成する AWS アカウント

作成者：Mike Virgilio (AWS)

概要

AWS リソースへの意図しないインバウンドインターネットアクセスは、組織のデータ境界にリスクをもたらす可能性があります。Network Access Analyzer は、Amazon Web Services () 上のリソースへの意図しないネットワークアクセスを特定するのに役立つ Amazon Virtual Private Cloud (Amazon VPC) 機能ですAWS。Network Access Analyzer を使用してネットワークアクセス要件を指定し、指定した要件を満たさない可能性のあるネットワークパスを特定できます。Network Access Analyzer を使用すると、次の操作を実行できます。

1. インターネットゲートウェイを介してインターネットにアクセスできる AWS リソースを特定します。
2. 運用環境と開発環境を分離し、またはトランザクションワークロードを分離するなど、仮想プライベートクラウド (VPC) が適切にセグメント化されていることを確認します。

Network Access Analyzer は、単一のコンポーネントだけでなく、エンドツーエンドのネットワーク到達可能性条件を分析します。リソースがインターネットにアクセスできるかどうかを判断するために、Network Access Analyzer はインターネットゲートウェイ、VPC ルートテーブル、ネットワークアクセスコントロールリスト (ACL)、Elastic Network Interface の公開 IP アドレスとセキュリティグループを評価します。これらのコンポーネントのいずれかがインターネットアクセスを妨げている場合、Network Access Analyzer は検出結果を生成しません。たとえば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに、`0/0` からのトラフィックを許可するオープンセキュリティグループがあっても、そのインスタンスがどのインターネットゲートウェイからもルーティングできないプライベートサブネットにある場合、Network Access Analyzer は検出結果を生成しません。これにより精度の高い結果が得られるため、インターネットから本当にアクセスできるリソースを特定できます。

Network Access Analyzerを実行するときには、「ネットワークアクセススコープ」を使用してネットワークアクセス要件を指定します。このソリューションは、インターネットゲートウェイとエラスティックネットワークインターフェース間のネットワークパスを識別します。このパターンでは、 が管理する組織 AWS アカウント 内の一元化された にソリューションをデプロイし AWS Organizations、組織 AWS リージョン内のすべてのアカウントを分析します。

このソリューションは、以下を念頭に置いて設計されました。

- AWS CloudFormation テンプレートは、このパターンで AWS リソースをデプロイするために必要な労力を削減します。
- デプロイ時に CloudFormation テンプレートと naa-script.sh スクリプトのパラメーターを調整して、環境に合わせてカスタマイズできます。
- Bash スクリプトは、複数のアカウントのネットワークアクセススコープをパラレルで自動的にプロビジョニングして分析します。
- Python スクリプトは検出結果を処理し、データを抽出し、結果を統合します。Network Access Analyzer の検出結果の統合レポートを CSV 形式または で確認できます AWS Security Hub。CSV レポートの例はこのパターンの「追加情報」セクションにあります。
- 検出結果を修正することも、naa-exclusions.csv ファイルに追加してfuture 分析から除外することもできます。

前提条件と制限

前提条件

- 組織のメンバーアカウントとして管理されるセキュリティサービスとツールをホスト AWS アカウント するための AWS Organizations。このパターンでは、このアカウントはセキュリティアカウントと呼ばれています。
- セキュリティアカウントには、アウトバウンドのインターネットにアクセスできるプライベートサブネットが必要です。手順については、「Amazon VPC のドキュメント」の「サブネットの作成」を参照してください。「NAT ゲートウェイ」または「インターフェイスVPC エンドポイント」を使用してインターネットアクセスを確立できます。
- CloudFormation の管理者権限を委任された AWS Organizations 管理アカウントまたはアカウントへのアクセス。手順については、「CloudFormation のドキュメント」の「委任管理者の登録」を参照してください。
- AWS Organizations と CloudFormation 間の信頼されたアクセスを有効にします。手順については、「CloudFormation ドキュメント」の「 信頼できるアクセスを有効にする AWS Organizations」を参照してください。
- Security Hub に結果をアップロードする場合は、アカウントと Amazon EC2 インスタンスがプロビジョニングされている AWS リージョン 場所で Security Hub を有効にする必要があります。詳細については、「セットアップ AWS Security Hub」を参照してください。

制約事項

- Network Access Analyzer 機能の制限により、クロスアカウントネットワークパスは現在分析されていません。
- ターゲットは の組織として管理 AWS アカウント する必要があります AWS Organizations。を使用していない場合は AWS Organizations、環境の naa-execrole.yaml CloudFormation テンプレートと naa-script.sh スクリプトを更新できます。代わりに、スクリプトを実行する AWS アカウント IDs とリージョンのリストを指定します。
- CloudFormation テンプレートは、アウトバウンドインターネットアクセスを持つプライベートサブネットに Amazon EC2 インスタンスをデプロイするように設計されています。 AWS Systems Manager エージェント (SSM エージェント) は、Systems Manager サービスエンドポイントに到達するためにアウトバウンドアクセスを必要とし、コードリポジトリのクローンを作成して依存関係をインストールするにはアウトバウンドアクセスが必要です。パブリックサブネットを使用する場合は、naa-resources.yaml テンプレートを変更して、Elastic IP アドレスを Amazon EC2 インスタンスに関連付ける必要があります。

アーキテクチャ

ターゲット アーキテクチャ

オプション 1: Amazon S3 バケットの検出結果にアクセス

図表に示す内容は以下のステップです。

1. ソリューションを手動で実行している場合、ユーザーは Session Manager を使用して Amazon EC2 インスタンスを認証し、naa-script.sh スクリプトを実行します。このシェルスクリプトはステップ 2 ~ 7 を実行します。

   ソリューションを自動的に実行する場合、 [naa-script.sh] スクリプトは cron 式で定義したスケジュールで自動的に起動します。このシェルスクリプトはステップ 2 ~ 7 を実行します。詳細については、このセクションの最後にある自動化とスケールを参照してください。
2. Amazon EC2 インスタンスは、Amazon S3 バケットから最新の naa-exception.csv ファイルをダウンロードします。このファイルは、プロセスの後半で Python スクリプトが除外を処理するときに使用されます。
3. Amazon EC2 インスタンスは `NAAEC2Role` AWS Identity and Access Management (IAM) ロールを引き受けます。これにより、Amazon S3 バケットにアクセスし、組織内の他のアカウントの `NAAExecRole` IAM ロールを引き受けるアクセス許可が付与されます。

4. Amazon EC2 インスタンスは、組織の管理アカウントの NAAExecRole IAM ロールを引き受け、組織内のアカウントのリストを生成します。
5. Amazon EC2 インスタンスは、組織のメンバーアカウント (アーキテクチャ図ではワークロードアカウントと呼ばれます) NAAExecRole の IAM ロールを引き受け、各アカウントでセキュリティ評価を実行します。結果は JSON ファイルとして Amazon EC2 インスタンスに保存されます。
6. Amazon EC2 インスタンスは、Python スクリプトを使用して JSON ファイルを処理し、データフィールドを抽出し、CSV レポートを作成します。
7. Amazon EC2 インスタンスは、CSV ファイルを Amazon S3 バケットにアップロードします。
8. Amazon EventBridge ルールはファイルのアップロードを検出し、Amazon SNS トピックを使用して、レポートが完了したことをユーザーに通知メールを送信します。
9. ユーザーは Amazon S3 バケットから CSV ファイルをダウンロードします。ユーザーはその結果を Excel テンプレートにインポートし、結果を確認します。

オプション 2: で検出結果にアクセスする AWS Security Hub

図に示す内容は以下のとおりです。

1. ソリューションを手動で実行している場合、ユーザーは Session Manager を使用して Amazon EC2 インスタンスを認証し、naa-script.sh スクリプトを実行します。このシェルスクリプトはステップ 2 ~ 7 を実行します。

   ソリューションを自動的に実行する場合、 [naa-script.sh] スクリプトは cron 式で定義したスケジュールで自動的に起動します。このシェルスクリプトはステップ 2 ~ 7 を実行します。詳細については、このセクションの最後にある自動化とスケールを参照してください。
2. Amazon EC2 インスタンスは、Amazon S3 バケットから最新の naa-exception.csv ファイルをダウンロードします。このファイルは、プロセスの後半で Python スクリプトが除外を処理するときに使用されます。
3. Amazon EC2 インスタンスは IAM NAAEC2Role ロールを引き受けます。これにより、Amazon S3 バケットにアクセスし、組織内の他のアカウントの NAAExecRole IAM ロールを引き受けるアクセス許可が付与されます。
4. Amazon EC2 インスタンスは、組織の管理アカウントの NAAExecRole IAM ロールを引き受け、組織内のアカウントのリストを生成します。

5. Amazon EC2 インスタンスは、組織のメンバーアカウント (アーキテクチャ図ではワークロードアカウントと呼ばれます) NAAExecRole の IAM ロールを引き受け、各アカウントでセキュリティ評価を実行します。結果は JSON ファイルとして Amazon EC2 インスタンスに保存されます。
6. Amazon EC2 インスタンスは Python スクリプトを使用して JSON ファイルを処理し、Security Hub にインポートするためのデータフィールドを抽出します。
7. Amazon EC2 インスタンスは、Network Access Analyzer の検出結果を Security Hub にインポートします。
8. Amazon EventBridge ルールはインポートを検出し、Amazon SNS トピックで処理が完了したことをユーザーに通知メールを送信します。
9. ユーザーは、Security Hub で検出結果を確認します。

自動化とスケール

naa-script.sh スクリプトをカスタムスケジュールで自動的に実行するようにこのソリューションをスケジュールできます。カスタムスケジュールを設定するには、naa-resources.yaml CloudFormation テンプレートで CronScheduleExpression パラメータを変更します。たとえば、0 0 * * 0 のデフォルト値は、ソリューションを毎週日曜日の深夜0時に実行します。 の値は、ソリューションを毎月第 1 日曜日の午前 0 時に実行します。0 0 * 1-12 0cron 式の使用に関する詳細は、「Systems Manager のドキュメント」の「cron 式と rate 式」を参照してください。

NAA-Resources スタックのデプロイ後にスケジュールを調整したい場合は、/etc/cron.d/naa-schedule で cron スケジュールを手動で編集できます。

ツール

AWS のサービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Identity and Access Management (IAM) は、誰が認証され、誰に使用を許可するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。

- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。また、セキュリティ業界標準やベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータの保存、保護、取得に役立つクラウドベースのオブジェクトストレージサービスです。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間が短縮され、 AWS リソースを大規模に安全に管理できます。このパターンは、システムマネージャーの機能である「Session Manager」 を使用します。

コードリポジトリ

このパターンのコードは、GitHub 内の「Network Access Analyzer マルチアカウント分析」リポジトリで利用できます。コードリポジトリには、以下のファイルが含まれます。

- naa-script.sh – この bash スクリプトは AWS アカウント、複数の Network Access Analyzer 分析を並行して開始するために使用されます。naa-resources.yaml CloudFormation テンプレートで定義されているように、このスクリプトは Amazon EC2 インスタンスの `/usr/local/naa`フォルダに自動的にデプロイされます。
- naa-resources.yaml — この CloudFormation テンプレートを使用して、組織のセキュリティアカウントにスタックを作成します。このテンプレートは、ソリューションをサポートするためにこのアカウントに必要なリソースをすべてデプロイします。このスタックは naa-execrole.yaml テンプレートの前にデプロイする必要があります。

> **Note**
>
> このスタックを削除して再デプロイする場合は、IAM ロール間のクロスアカウント依存関係を再構築するために、`NAAExecRole`スタックセットを再構築する必要があります。

- naa-execrole.yaml — この CloudFormation テンプレートを使用して、管理アカウントを含む組織内のすべてのアカウントに `NAAExecRole` IAM ロールをデプロイするスタックセットを作成します。
- naa-processfindings.py — naa-script.sh スクリプトは、この Python スクリプトを自動的に呼び出して Network Access Analyzer の JSON 出力を処理し、naa-exclusions.csv ファイル内の正常なリ

ソースをすべて除外してから、統合結果の CSV ファイルを生成するか、結果を Security Hub にインポートします。

エピック

デプロイの準備

タスク	説明	必要なスキル
コードリポジトリを複製します。	1. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 2. 次のコマンドを入力します。 `git clone https://github.com/aws-samples/network-access-analyzer-multi-account-analysis.git`	AWS DevOps
テンプレートを確認します。	1. クローンされたリポジトリで、naa-resources.yaml ファイルと naa-execrole.yaml ファイルを開きます。 2. これらのテンプレートにより、作成されたリソースを確認し、環境に合わせて必要に応じてテンプレートを調整してください。詳細については、CloudFormation ドキュメントの「テンプ	AWS DevOps

タスク	説明	必要なスキル
	レートの使用」を参照してください。 3. naa-resources.yaml ファイルと naa-execrole.yaml ファイルを保存して閉じます。	

CloudFormation スタックの作成

タスク	説明	必要なスキル
セキュリティアカウントにリソースをプロビジョニングします。	naa-resources.yaml テンプレートを使用して、セキュリティアカウントに必要なすべてのリソースをデプロイする CloudFormation スタックを作成します。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。 1. Specify template (テンプレートの指定) ページでテンプレートは準備完了を選択し、naa-resources.yaml ファイルをアップロードします。 2. [スタック詳細の指定] ページで、[スタック名] に NAA-Resources と入力します。	AWS DevOps

タスク	説明	必要なスキル
	3. [パラメータ] セクションで、次の値を入力します。 • VPCId — アカウント内の VPC を選択します。 • SubnetId — インターネットにアクセスできる非公開サブネットを選択します。 **Note** パブリックサブネットを選択した場合、CloudFormation テンプレートはデフォルトで Elastic IP アドレスをプロビジョニングおよびアタッチしないため、Amazon EC2 インスタンスにパブリック IP アドレスが割り当てられない場合があります。 • InstanceType — デフォルトのインスタンスタイプはそのままにしておきます。 • InstanceImageId — デフォルトのままにします。	

タスク	説明	必要なスキル
	• KeyPairName — SSH を使用してアクセスする場合は、既存の [key pair] を指定します。 • PermittedSSHInbound — SSH を使用してアクセスする場合は、許可された CIDR ブロックを指定します。SSH を使用していない場合は、127.0.0.1 のデフォルト値のままにします。 • BucketName — デフォルト値は naa-<accountID>-<region> です。これは必要に応じて変更できます。カスタム値を指定すると、アカウント ID とリージョンが指定した値に自動的に追加されます。 • EmailAddress — 分析が完了したときの Amazon SNS 通知用メールアドレスを指定します。 Note Amazon SNS サブスクリプション設定は、分析の完了前に確認する必	

タスク	説明	必要なスキル
	要があります。確認しないと、通知は送信されません。 • NAAEC2Role — 命名規則により、この IAM ロールに別の名前を付ける必要がある場合以外、デフォルトのままにします。 • NAAExecRole — [naa-execrole.yaml] のデプロイ時に別の名前が使用されない限り、デフォルトのままにします。 • Parallelism — 実行するパラレル評価の数を指定します。 • Regions – 分析 AWS リージョン する を指定します。 • ScopeNameValue — スコープに割り当てるタグを指定します。このタグは、ネットワークアクセスの範囲を決定することに使用されます。 • ExclusionFile — 除外ファイル名を指定します。このファイル内のエントリは検出結果から除外されます。	

タスク	説明	必要なスキル
	• FindingsToCSV — 検出結果を CSV に出力するかを指定します。有効な値は、true および false です。 • FindingsToSecurityHub — 検出結果を Security Hub にインポートするかを指定します。有効な値は、true および false です。 • EmailNotificationsForSecurityHub — 検出結果を Security Hub にインポートして電子メール通知を生成するかを指定します。有効な値は、true および false です。 • ScheduledAnalysis — ソリューションをスケジュールに従って自動的に実行させたい場合は、true を入力して、CronScheduleExpression パラメータにスケジュールをカスタマイズします。ソリューションを自動的に実行しない場合は、false を入力します。 • CronScheduleExpression — ソリューショ	

タスク	説明	必要なスキル
	ンを自動的に実行する場合は、cron 式を入力してスケジュールを定義します。詳細については、このパターンの「アーキテクチャ」セクションにおける自動化とスケールを参照してください。 1. レビューページで、次のリソースには機能が必要: [AWS:: IAM:: Role]を選択して、Create Stack を選択します。 2. スタックが正常に作成されたら、CloudFormation コンソールの [出力] タブで、NAAEC2Role Amazon リソースネーム (ARN)をコピーします。この ARN は、後で naa-execrole.yaml ファイルをデプロイするときに使用します。	

タスク	説明	必要なスキル
メンバーアカウントで IAM ロールをプロビジョニングします。	AWS Organizations 管理アカウントまたは CloudFormation の委任管理者権限を持つアカウントで、naa-execrole.yaml テンプレートを使用して CloudFormation スタックセットを作成します。スタックセットは、組織内のすべてのメンバーアカウントに `NAAExecRole` IAM ロールをデポロイします。手順については、「CloudFormation のドキュメント」の「サービスマネージド型のアクセス許可を持つスタックセットの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。 1. テンプレートの準備でテンプレートの準備完了を選択して、naa-execrole.yaml ファイルをアップロードします。 2. [StackSet の詳細を指定] ページで、スタックセットに `NAA-ExecRole` という名前を付けます。 3. [パラメータ] セクションで、次の値を入力します。 • `AuthorizedARN` — `NAA-Resources` スタックの作成時にコピー	AWS DevOps

タスク	説明	必要なスキル
	した NAAEC2Role ARN を入力します。 • NAARoleName — naa-resources.yaml ファイルのデプロイ時に別の名前を使用した場合を除き、NAAExecRole のデフォルト値のままにします。 4. [アクセス権限] で、[Service-managed permissions (サービスマネージド型のアクセス許可)] を選択します。 5. [デプロイオプションの設定] ページの [デプロイターゲット] で、[組織へのデプロイ] を選択し、すべてのデ フォルトを受け入れます。 **Note** スタックをすべてのメンバーアカウントに同時にデプロイする場合は、最大同時アカウント数と障害耐性を などの高い値に設定します100。 6. デプロイリージョンで、Network Access Analyzer の Amazon EC2	

タスク	説明	必要なスキル
	インスタンスがデプロイされるリージョンを選択します。IAM リソースはグローバルであり、リージョナルではないため、IAM ロールはすべてのアクティブなリージョンにデプロイされます。 7. レビューページで、 がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを確認してから、StackSet の作成を選択します。 8. スタックインスタンスタブ (個々のアカウントのステータス) と操作タブ (全体的なステータス) をモニタリングして、デプロイが完了したかを判断します。	

タスク	説明	必要なスキル
管理アカウントで IAM ロールをプロビジョニングします。	[naa-execrole.yaml] テンプレートを使用して、組織の管理アカウントに IAM ロールをデプロイする CloudFormation スタックを作成しました。以前に作成したスタックセットでは、管理アカウントに IAM ロールはデプロイされません。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。 1. [Specify template] (テンプレートの指定) ページで [テンプレートは準備完] を選択し、[naa-execrole.yaml] ファイルをアップロードします。 2. [スタック詳細の指定] ページで、[スタック名] に `NAA-ExecRole` と入力します。 3. [パラメータ] セクションで、次の値を入力します。 • `AuthorizedARN` — `NAA-Resources` スタックの作成時にコピーした `NAAEC2Role` ARN を入力します。 • `NAARoleName` — [naa-resources.yaml]	AWS DevOps

タスク	説明	必要なスキル
	ファイルのデプロイ時に別の名前を使用した場合を除き、NAAExecRole のデフォルト値のままにします。 4. レビューページで、次のリソースには機能が必要: [AWS:: IAM:: Role]を選択して、Create Stack を選択します。	

分析の実行

タスク	説明	必要なスキル
シェルスクリプトをカスタマイズします。	1. 組織のセキュリティアカウントにサインインします。 2. Session Manager を使用して、以前にプロビジョニングした Network Access Analyzer の Amazon EC2 インスタンスに接続します。手順については、「Session Manager を使用して Linux インスタンスに接続」を参照してください。接続できない場合は、このパターンの「トラブルシューティング」セクションを参照してください。 3. 次のコマンドを入力して、編集する naa-script.sh ファイルを開きます。	AWS DevOps

タスク	説明	必要なスキル
	``` sudo -i cd /usr/local/naa vi naa-script.sh ``` 4. このスクリプト内の調整可能なパラメーターと変数を確認して、環境に応じて変更します。カスタマイズオプションの詳細については、スクリプトの冒頭におけるコメントを参照してください。 たとえば、管理アカウントから組織内のすべてのメンバーアカウントのリストを取得する代わりに、スクリプトを変更してスキャン AWS リージョン する AWS アカウント IDs または を指定するか、これらのパラメータを含む外部ファイルを参照できます。 5. naa-script.sh ファイルを保存して閉じます。	

タスク	説明	必要なスキル

ターゲットアカウントを分析します。	1. 次のコマンドを入力します。これにより、naa-script.sh スクリプトが実行されます。
	
	`sudo -i`
	`cd /usr/local/naa`
	`screen`
	`./naa-script.sh`
	
	次の点に注意してください:
	• この `screen` コマンドは、接続がタイムアウトになりまたはコンソールにアクセスできなくなった場合でも、スクリプトの実行を継続できるようにします。
	• スキャンの開始後、Ctrl+A D を押すことで画面を強制的に切り離すことができますが、その後、分析の進行中はインスタンス接続を閉じることができます。
	• デタッチしたセッションを再開するには、インスタンスに接続し、`sudo -i` を押して、`screen -r` を押します。
	2. 出力をモニタリングして エラーがないかを確認し、スクリプトが正しく動作していることを確認します。
		AWS DevOps

タスク	説明	必要なスキル
	出力例については、このパターンの「追加情報」セクションを参照してください。 3. 分析が完了するまで待ちます。E メール通知を設定した場合、結果が Amazon S3 バケットにアップロードされるか、Security Hub にインポートされると、E メールが送信されます。	
オプション 1 – Amazon S3 バケットから結果を取得します。	1. `naa-<accountID>-<region>` バケットから CSV ファイルをダウンロードします。手順については、「Amazon S3 のドキュメント」の「オブジェクトのダウンロード」を参照してください。 2. Amazon S3 バケットから CSV ファイルを削除します。これはコスト最適化のベストプラクティスです。手順については、「Amazon S3 のドキュメント」の「オブジェクトの削除」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
オプション 2 — Security Hub で結果を確認します。	1. Security Hub コンソールを開きます。 2. ナビゲーションペインで [検出結果] を選択します。 3. Network Access Analyzer の検出結果を確認します。手順については、「Security Hub のドキュメント」の「検出結果の表示と詳細」を参照してください。 **Note** フィルターで始まるタイトルを追加し、と入力することで、結果を検索できます `Network Access Analyzer`。	AWS DevOps

検出結果の修正と除外

タスク	説明	必要なスキル
検出結果を修正します。	組織の管理アカウントにIAMロールをデプロイするCloudFormationスタックを作成しました。ID、リソース AWS 、ネットワークの周囲に境界を作成する方法の詳細とベストプラクティスについては、「 でのデータ境界の構築 AWS (AWS ホワイトペー	AWS DevOps

タスク	説明	必要なスキル
	パー）」を参照してください。	

タスク	説明	必要なスキル
ネットワークパスが正常であることがわかっているリソースを除きます。	Network Access Analyzer がインターネットからアクセスできるはずのリソースに関する検出結果を生成した場合、それらのリソースを除外リストに追加できます。今度、Network Access Analyzer を実行しても、そのリソースの検出結果は生成されません。 1. `/usr/local/naa` に移動して naa-script.sh スクリプトを開きます。`S3_EXCLUSION_FILE` 変数の値をメモしておきます。 2. `S3_EXCLUSION_FILE` 変数の値が `true` の場合、naa-exclusions.csv ファイルを `naa-<accountID>-<region>` バケットからダウンロードします。手順については、「Amazon S3 のドキュメント」の「オブジェクトのダウンロード」を参照してください。 `S3_EXCLUSION_FILE` 変数の値がの場合は `false` の場合、`/usr/local/naa` に移動して、naa-exclusions.csv ファイルを開きます。	AWS DevOps

タスク	説明	必要なスキル

	Note
	S3_EXCLUSION_FILE 変数の値が の場合false、スクリプトはローカルバージョンの除外ファイルを使用します。後で値を に変更するとtrue、スクリプトはローカルバージョンを Amazon S3 バケットの ファイルで上書きします。
	3. naa-exclusions.csv ファイルに、除外するリソースを入力します。1 行に 1 つのリソースを入力し、次の形式を使用します。
	`<resource_id>,<secgroup_id>,<sgrule_cidr>,<sgrule_portrange>,<sgrule_protocol>`
	以下に、リソースの例を示します。
	`eni-1111aaaaa2222bbbb,sg-3333ccccc44`	

タスク	説明	必要なスキル
	`44dddd,0.0.0.0/0,80 to 80,tcp` 4. naa-exclusions.csv ファイルを保存して閉じます。 5. Amazon S3 バケットから naa-exclusions.csv ファイルをダウンロードした場合は、新しいバージョンをアップロードします。手順については、Amazon S3 ドキュメントの「オブジェクトのアップロード」を参照してください。	

(オプション) [naa-script.sh] スクリプトを更新します。

タスク	説明	必要なスキル
[naa-script.sh] スクリプトを更新します。	naa-script.sh スクリプトをリポジトリ内の最新バージョンに更新するには、次の操作を行います。 1. Session Manager を使用して Amazon EC2 インスタンスに接続します。手順については、「Session Manager を使用して Linux インスタンスに接続」を参照してください。 2. 次のコマンドを入力します。 `sudo -i`	AWS DevOps

タスク	説明	必要なスキル

	3. naa-script.sh スクリプトディレクトリに移動します。
	`cd /usr/local/naa`
	4. 次のコマンドを入力してローカルスクリプトをスタッシュし、カスタム変更を最新バージョンにマージできるようにします。
	`git stash`
	5. 次のコマンドを入力して、スクリプトの最新バージョンを取得します。
	`git pull`
	6. 次のコマンドを入力して、カスタムスクリプトを最新バージョンのスクリプトとマージします。
	`git stash pop`	

(オプション) クリーンアップする

タスク	説明	必要なスキル
デプロイされたリソースをすべて削除します。	リソースはアカウントにデプロイしたままにしておくことができます。	AWS DevOps

タスク	説明	必要なスキル
	すべてのリソースのプロビジョニングを解除する場合は、以下のように実行します。 1. 管理アカウントにプロビジョニングされた NAA-ExecRole スタックを削除します。手順については、「CloudFormation のドキュメント」の「スタックの削除」を参照してください。 2. 組織の管理アカウントまたは委任された管理者アカウントにプロビジョニングされた スタックセットを削除します。手順については、「CloudFormation のドキュメント」の「スタックセットの削除」を参照してください。 3. naa-<accountID>-<region> Amazon S3 バケット内のすべてのオブジェクトを削除します。手順については、「Amazon S3 のドキュメント」の「オブジェクトの削除」を参照してください。 4. セキュリティアカウントにプロビジョニングされた NAA-Resources スタックを削除します。手順につ	

タスク	説明	必要なスキル
	いては、「CloudFormation のドキュメント」の「スタックの削除」を参照してください。	

トラブルシューティング

問題	ソリューション
Session Manager を使用して Amazon EC2 インスタンスに接続できません。	SSM エージェントは、Systems Manager エンドポイントと通信できる必要があります。以下の操作を実行します。 1. Amazon EC2 インスタンスがデプロイされているサブネットにインターネットアクセスがあることを検証します。 2. Amazon EC2 インスタンスを再起動します。
スタックセットをデプロイすると、CloudFormation コンソールに `Enable trusted access with AWS Organizations to use service-managed permissions` を求めるメッセージが表示されます。	これは、 AWS Organizations と CloudFormation の間で信頼されたアクセスが有効になっていないことを示します。サービスマネージド型のスタックセットをデプロイするには、信頼されたアクセスが必要です。このボタンを選択すると、信頼されたアクセスは有効になります。詳細については、「CloudFormation のドキュメント」の「信頼されたアクセスの有効化」を参照してください。

関連リソース

- 新規 – Amazon VPC Network Access Analyzer (AWS ブログ記事)
- AWS re:Inforce 2022 - での効果的なネットワークアクセスコントロールの検証 AWS (NIS202) (ビデオ)

- 「デモ- Network Access Analyzer を使用した組織全体のインターネット進入データパス分析」(ビデオ)

追加情報

コンソール出力の例

次のサンプルは、ターゲットアカウントのリストを生成し、ターゲットアカウントを分析した出力を示しています。

```
[root@ip-10-10-43-82 naa]# ./naa-script.sh
download: s3://naa-<account ID>-us-east-1/naa-exclusions.csv to ./naa-exclusions.csv

AWS Management Account: <Management account ID>

AWS Accounts being processed...
<Account ID 1> <Account ID 2> <Account ID 3>

Assessing AWS Account: <Account ID 1>, using Role: NAAExecRole
Assessing AWS Account: <Account ID 2>, using Role: NAAExecRole
Assessing AWS Account: <Account ID 3>, using Role: NAAExecRole
Processing account: <Account ID 1> / Region: us-east-1
Account: <Account ID 1> / Region: us-east-1 - Detecting Network Analyzer scope...
Processing account: <Account ID 2> / Region: us-east-1
Account: <Account ID 2> / Region: us-east-1 - Detecting Network Analyzer scope...
Processing account: <Account ID 3> / Region: us-east-1
Account: <Account ID 3> / Region: us-east-1 - Detecting Network Analyzer scope...
Account: <Account ID 1> / Region: us-east-1 - Network Access Analyzer scope detected.
Account: <Account ID 1> / Region: us-east-1 - Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
Account: <Account ID 2> / Region: us-east-1 - Network Access Analyzer scope detected.
Account: <Account ID 2> / Region: us-east-1 - Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
Account: <Account ID 3> / Region: us-east-1 - Network Access Analyzer scope detected.
Account: <Account ID 3> / Region: us-east-1 - Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
```

CSV レポートの例

以下の画像は CSV 出力の例です。

マルチアカウント AWS 環境でハイブリッドネットワークの DNS 解決を設定する

作成者: Anvesh Koganti (AWS)

概要

このパターンは、複数の Amazon Web Services (AWS) アカウントを含むハイブリッドネットワーク環境で DNS 解決を設定するための包括的なソリューションを提供します。これにより、エンドポイントを介してオンプレミスネットワークと AWS 環境間の双方向 DNS 解決が可能になります Amazon Route 53 Resolver 。このパターンは、マルチアカウントで一元化されたアーキテクチャで DNS 解決を可能にする 2 つのソリューションを示しています。

- 基本的なセットアップでは、Route 53 プロファイルを使用しません。複雑さの低い中小規模のデプロイのコストを最適化するのに役立ちます。
- 拡張セットアップでは、Route 53 Profiles を使用してオペレーションを簡素化します。大規模または複雑な DNS デプロイに最適です。

> **Note**
>
> 実装する前に、「制限」セクションでサービスの制限とクォータを確認してください。意思決定を行う際には、管理のオーバーヘッド、コスト、運用の複雑さ、チームの専門知識などの要因を考慮してください。

前提条件と制限

前提条件

- 共有サービスおよびワークロードアカウント全体にデプロイされた Amazon Virtual Private Cloud (Amazon VPC) を使用する AWS マルチアカウント環境 (アカウント構造のAWS ベストプラクティスに従って AWS Control Tower を介してセットアップすることが望ましい) 。
- オンプレミスネットワークと AWS 環境間の既存のハイブリッド接続 (AWS Direct Connect または AWS Site-to-Site VPN) 。
- Amazon VPC ピアリング AWS Transit Gateway、または VPCs AWS クラウド 間のレイヤー 3 ネットワーク接続用の WAN。 (この接続はアプリケーショントラフィックに必要です。 DNS 解決が機能する必要はありません。 DNS 解決はVPCs.)

- オンプレミス環境で実行されている DNS サーバー。

制約事項

- Route 53 Resolver のエンドポイント、ルール、プロファイルはリージョン構造であり、グローバル組織 AWS リージョン では複数の でレプリケーションが必要になる場合があります。
- Route 53 Resolver、プライベートホストゾーン、およびプロファイルのサービスクォータの包括的なリストについては、Route 53 ドキュメントの「クォータ」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Route 53 アウトバウンドエンドポイントとインバウンドエンドポイント
- 条件付き転送用の Route 53 Resolver ルール
- AWS Resource Access Manager (AWS RAM)
- Route 53 プライベートホストゾーン

ターゲット アーキテクチャ

アウトバウンドエンドポイントとインバウンドエンドポイント

次の図は、 からオンプレミス AWS への DNS 解決フローを示しています。これは、ドメインがオンプレミスでホストされているアウトバウンド解決の接続設定です。この設定に関連するプロセスの概要を次に示します。詳細については、エピックセクションを参照ください。

1. アウトバウンド Route 53 Resolver エンドポイントを共有サービス VPC にデプロイします。
2. オンプレミスでホストされているドメインの共有サービスアカウントに Route 53 Resolver ルール (転送ルール) を作成します。
3. オンプレミスのホストドメインを解決する必要があるリソースをホストする他のアカウントの VPCs とルールを共有して関連付けます。これは、このセクションで後述するように、ユースケースに応じてさまざまな方法で行うことができます。

接続を設定した後、アウトバウンド解決に関連するステップは次のとおりです。

1. Amazon Elastic Compute Cloud (Amazon EC2) インスタンスは、 の DNS 解決リクエストdb.onprem.example.comを VPC の Route 53 Resolver の VPC+2 アドレスに送信します。
2. Route 53 Resolver は、Resolver ルールをチェックし、アウトバウンドエンドポイントを使用してリクエストをオンプレミス DNS サーバー IPs に転送します。
3. アウトバウンドエンドポイントは、リクエストをオンプレミス DNS IPs。トラフィックは、共有サービス VPC とオンプレミスデータセンター間の確立されたハイブリッドネットワーク接続を経由します。
4. オンプレミス DNS サーバーはアウトバウンドエンドポイントに応答し、その後、VPC の Route 53 Resolver に応答を転送します。Resolver は EC2 インスタンスにレスポンスを返します。

次の図は、オンプレミス環境から への DNS 解決フローを示しています AWS。これは、ドメインがホストされているインバウンド解決の接続設定です AWS。この設定に関連するプロセスの概要を次に示します。詳細については、エピックセクションを参照ください。

1. 共有サービス VPC にインバウンドリゾルバーエンドポイントをデプロイします。
2. 共有サービスアカウントにプライベートホストゾーンを作成します (集中型アプローチ) 。
3. プライベートホストゾーンを共有サービス VPC に関連付けます。これらのゾーンを共有し、VPC-to-VPC VPCs。これは、このセクションで後述するように、ユースケースに応じてさまざまな方法で行うことができます。

接続を設定すると、インバウンド解決に関連するステップは次のとおりです。

1. オンプレミスリソースは、 の DNS 解決リクエストec2.prod.aws.example.comをオンプレミス DNS サーバーに送信します。
2. オンプレミス DNS サーバーは、ハイブリッドネットワーク接続を介して共有サービス VPC 内のインバウンドリゾルバーエンドポイントにリクエストを転送します。
3. インバウンドリゾルバーエンドポイントは、VPC Route 53 Resolver の助けを借りて、関連するプライベートホストゾーンでリクエストを検索し、適切な IP アドレスを取得します。
4. これらの IP アドレスはオンプレミス DNS サーバーに返送され、オンプレミスリソースにレスポンスが返されます。

この設定により、オンプレミスリソースは、インバウンドエンドポイントを介して適切な AWS プライベートホストゾーンにクエリをルーティングすることで、プライベートドメイン名を解決でき

ます。このアーキテクチャでは、プライベートホストゾーンは共有サービス VPC に一元化され、単一のチームによる一元的な DNS 管理が可能になります。これらのゾーンは、VPC から VPCs への DNS 解決のユースケースに対応するために、多くの VPC に関連付けることができます。 VPC-to-VPC または、DNS ドメインの所有権と管理をそれぞれに委任することもできます AWS アカウント。この場合、各アカウントは独自のプライベートホストゾーンを管理し、各ゾーンを中央共有サービス VPC に関連付けて、オンプレミス環境と統一された解決を行います。この分散型アプローチは、このパターンの範囲外です。詳細については、「Amazon VPCs」を参照してください。

Resolver エンドポイントを使用して基本的な DNS 解決フローを確立する場合は、 全体で Resolver ルールとプライベートホストゾーンの共有と関連付けを管理する方法を決定する必要があります AWS アカウント。これは、基本セットアップセクションで説明されているように、 AWS RAM を使用してリゾルバールールを共有し、プライベートホストゾーンの関連付けを直接行うか、拡張セットアップセクションで説明されているように Route 53 プロファイルを使用して行うという 2 つの方法でアプローチできます。選択は、組織の DNS 管理設定と運用要件によって異なります。次のアーキテクチャ図は、一般的なエンタープライズデプロイを表す、異なるアカウント間で複数の VPCs を含むスケーリングされた環境を示しています。

基本セットアップ

基本的なセットアップでは、マルチアカウント AWS 環境でのハイブリッド DNS 解決の実装では、を使用して Resolver 転送ルールとプライベートホストゾーンの関連付け AWS RAM を共有し、オンプレミスと AWS リソース間の DNS クエリを管理します。この方法では、オンプレミスネットワークに接続された共有サービス VPC 内の一元化された Route 53 Resolver エンドポイントを使用して、インバウンドとアウトバウンドの両方の DNS 解決を効率的に処理します。

- アウトバウンド解決の場合、リゾルバー転送ルールは共有サービスアカウントで作成され、 AWS アカウント を使用して他の と共有されます AWS RAM。この共有は、同じリージョン内のアカウントに限定されます。その後、ターゲットアカウントはこれらのルールを VPCs に関連付け、それらの VPCs内のリソースを有効にしてオンプレミスドメイン名を解決できます。
- インバウンド解決の場合、プライベートホストゾーンは共有サービスアカウントに作成され、共有サービス VPC に関連付けられます。その後、Route 53 API、 AWS SDKs、または AWS Command Line Interface () を使用して、これらのゾーンを他のアカウントの VPCs に関連付けることができますAWS CLI。その後VPCs のリソースは、プライベートホストゾーンで定義された DNS レコードを解決し、 AWS 環境全体で統合 DNS ビューを作成できます。

次の図は、この基本セットアップの DNS 解決フローを示しています。

この設定は、限られた規模で DNS インフラストラクチャを操作する場合に適しています。ただし、環境が大きくなるにつれて、 の管理が難しくなる可能性があります。プライベートホストゾーンとリゾルバールールを共有し、VPCs個別に関連付ける方法を管理する運用上のオーバーヘッドは、スケールとともに大幅に増加します。さらに、プライベートホストゾーンあたり 300 VPC 関連付け制限などのサービスクォータは、大規模なデプロイで制約要因になる可能性があります。拡張セットアップは、これらの課題に対処します。

拡張セットアップ

Route 53 Profiles は、複数の にわたるハイブリッドネットワークの DNS 解決を管理するための合理化されたソリューションを提供します AWS アカウント。プライベートホストゾーンとリゾルバールールを個別に管理する代わりに、DNS 設定を 1 つのコンテナにグループ化し、リージョン内の複数の VPCs とアカウント間で簡単に共有および適用できます。この設定では、DNS 設定の管理を大幅に簡素化しながら、一元化された Resolver エンドポイントアーキテクチャを共有サービス VPC に維持します。

次の図は、拡張セットアップの DNS 解決フローを示しています。

Route 53 プロファイルを使用すると、プライベートホストゾーンの関連付け、リゾルバー転送ルール、DNS ファイアウォールルールを 1 つの共有可能なユニットにパッケージ化できます。共有サービスアカウントでプロファイルを作成し、 を使用してメンバーアカウントと共有できます AWS RAM。プロファイルを共有してターゲット VPCs に適用すると、必要なすべての関連付けと設定がサービスによって自動的に処理されます。これにより、DNS 管理の運用オーバーヘッドが大幅に削減され、拡大する環境に優れたスケーラビリティが提供されます。

自動化とスケール

AWS CloudFormation や Terraform などの infrastructure as code (IaC) ツールを使用して、Route 53 Resolver エンドポイント、ルール、プライベートホストゾーン、プロファイルを自動的にプロビジョニングおよび管理します。DNS 設定を継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインと統合して、一貫性、再現性、迅速な更新を実現します。

ツール

AWS のサービス

- AWS Resource Access Manager（AWS RAM） は、 間でリソースを安全に共有 AWS アカウントし、運用上のオーバーヘッドを削減し、可視性と監査性を提供します。

- Amazon Route 53 Resolver は AWS リソースからの DNS クエリに再帰的に応答し、すべての VPCs でデフォルトで使用できます。Resolver エンドポイントと条件付き転送ルールを作成して、オンプレミスデータセンターと VPCs 間の DNS 名前空間を解決できます。
- Amazon Route 53 プライベートホストゾーンは、Route 53 がドメインとそのサブドメインの DNS クエリにどのように応答するかに関する情報を保持するコンテナです。
- Amazon Route 53 Profiles を使用すると、DNS 関連の Route 53 設定 AWS アカウント を多くの VPCsおよび管理できます。

ベストプラクティス

このセクションでは、Route 53 Resolver を最適化するためのベストプラクティスをいくつか紹介します。これらは Route 53 のベストプラクティスのサブセットを表します。包括的なリストについては、「Amazon Route 53 のベストプラクティス」を参照してください。

Resolver エンドポイントでのループ設定の回避

- VPC の関連付けを慎重に計画することで、再帰的なルーティングを防ぐように DNS アーキテクチャを設計します。VPC がインバウンドエンドポイントをホストする場合は、循環参照を作成する可能性のある Resolver ルールに VPC を関連付けないでください。
- アカウント間で DNS リソースを共有する場合は、クリーンルーティングパスを維持するために AWS RAM 戦略的に を使用します。

詳細については、Route 53 ドキュメントの「Resolver エンドポイントによるループ設定の回避」を参照してください。

Resolver エンドポイントのスケーリング

- 1 秒あたりのクエリ数 (QPS) が多い環境では、エンドポイントの ENI あたり 10,000 QPS の制限があることに注意してください。DNS QPS をスケールするために、エンドポイントにさらに ENIs を追加できます。
- Amazon CloudWatch は `InboundQueryVolume`および `OutboundQueryVolume`メトリクスを提供します (CloudWatch ドキュメントを参照) 。しきい値が特定の値 (10,000 QPS の 80%) を超えた場合に警告するモニタリングルールを設定することをお勧めします。
- Resolver エンドポイントのステートフルセキュリティグループルールを設定して、大量のトラフィック中に接続追跡制限が DNS クエリスロットリングを発生させないようにします。セキュリティグループでの接続追跡の仕組みの詳細については、Amazon EC2 ドキュメントの「Amazon EC2 セキュリティグループ接続追跡」を参照してください。 Amazon EC2

詳細については、Route 53 ドキュメントの「リゾルバーエンドポイントのスケーリング」を参照してください。

Resolver エンドポイントに高可用性を提供する

- 冗長性を確保するために、少なくとも 2 つのアベイラビリティーゾーンに IP アドレスを持つインバウンドエンドポイントを指定します。
- 追加のネットワークインターフェイスをプロビジョニングして、メンテナンス中やトラフィックの急増時に可用性を確保します。

詳細については、Route 53 ドキュメントの「Resolver エンドポイントの高可用性」を参照してください。

エピック

Route 53 Resolver エンドポイントをデプロイする

タスク	説明	必要なスキル
インバウンドエンドポイントをデプロイします。	Route 53 Resolverはインバウンド エンドポイントを使用して、DNS クエリをオンプレミスネットワークから Route 53 Resolver に転送します。手順については、Route 53 ドキュメントの「VPC へのインバウンド DNS クエリの転送」 を参照してください。受信エンドポイント IP アドレスをメモしておきます。	AWS 管理者、クラウド管理者
アウトバウンドエンドポイントをデプロイします。	Route 53 Resolver は、アウトバウンドエンドポイントを使用して DNS クエリをオンプレミスの DNS レゾルバ に送信します。手順については、Route 53 ドキュメントの「アウトバウンド DNS クエリ	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	のネットワークへの転送」 を参照してください。出力エンドポイント ID を書きとめておきます。	

Route 53 プライベートホストゾーンを設定および共有する

タスク	説明	必要なスキル
ホストされているドメインのプライベートホストゾーンを作成します AWS。	このゾーンは、オンプレミス環境から解決する必要がある AWSホストドメイン (などprod.aws.example.com) 内のリソースの DNS レコードを保持します。手順については、Route 53 ドキュメントの「プライベートホストゾーンの作成」を参照してください。 プライベートホストゾーンを作成するときは、VPC を同じアカウントが所有するホストゾーンに関連付ける必要があります。この目的のために共有サービス VPC を選択します。	AWS 管理者、クラウド管理者
基本的なセットアップ: プライベートホストゾーンを他のアカウントの VPCs に関連付けます。	基本セットアップを使用している場合 (アーキテクチャセクションを参照)。 メンバーアカウント VPCs のリソースがこのプライベートホストゾーンの DNS レコー	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	ドを解決できるようにするには、VPCs をホストゾーンに関連付ける必要があります。関連付けを承認してから、関連付けをプログラムで作成する必要があります。手順については、Route 53 ドキュメントの「Associating an Amazon VPC and a private host zone that you created with different AWS アカウント」を参照してください。	

タスク	説明	必要なスキル
拡張セットアップ: Route 53 プロファイルを設定して共有します。	拡張セットアップを使用している場合 (アーキテクチャセクションを参照) 。 1. Route 53 プロファイルを作成し、関連するプライベートホストゾーンを関連付けます。手順については、Route 53 ドキュメントの「Route 53 プロファイルの作成」を参照してください。 2. AWS RAM を使用してプロファイルをメンバーアカウントと共有し、共有プロファイルをターゲット VPCs。手順については、Route 53 ドキュメントの「Route 53 プロファイルの共有」および「Route 53 プロファイルを VPCs」を参照してください。 **Note** 組織の構造と DNS 要件によっては、異なるアカウントまたはワークロードに対して複数のプロファイルを作成および管理する必要がある場合があります。	AWS 管理者、クラウド管理者

Route 53 Resolver 転送ルールを設定および共有する

タスク	説明	必要なスキル
オンプレミスでホストされているドメインの転送ルールを作成します。	このルールは、オンプレミスドメイン (などonprem.example.com) の DNS クエリをオンプレミス DNS リゾルバーに転送するように Route 53 Resolver に指示します。このルールを作成するには、オンプレミス DNS リゾルバーの IP アドレスとアウトバウンドエンドポイント ID が必要です。手順については、Route 53 ドキュメントの「転送ルールの作成」を参照してください。	AWS 管理者、クラウド管理者
基本設定: 転送ルールを共有し、他のアカウントの VPCsに関連付けます。	基本セットアップを使用している場合： 転送ルールを有効にするには、ルールを共有し、他のアカウントの VPCs に関連付ける必要があります。その後、Route 53 Resolver はドメインを解決するときにルールを考慮します。手順については、Route 53 ドキュメントの「Resolver ルール AWS アカウント を他の と共有ルールを使用する」および「転送ルールを VPC に関連付ける」を参照してください。	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
拡張セットアップ: Route 53 プロファイルを設定して共有します。	拡張セットアップを使用している場合： 1. 前のステップで Route 53 プロファイルを既に作成している場合は、同じプロファイルを使用できます。そうでない場合は、Route 53 プロファイルを作成し、関連するリゾルバー転送ルールを関連付けます。手順については、Route 53 ドキュメントの「Route 53 プロファイルの作成」を参照してください。 2. AWS RAM を使用してプロファイルをメンバーアカウントと共有し、共有プロファイルをターゲット VPCs。手順については、Route 53 ドキュメントの「Route 53 プロファイルの共有」および「Route 53 プロファイルを VPCs」を参照してください。 **Note** 組織の構造と DNS 要件によっては、異なるアカウントまたはワークロードに対して複数のプロファイルを作成	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	および管理する必要がある場合があります。	

AWS 統合用にオンプレミス DNS リゾルバーを設定する

タスク	説明	必要なスキル
オンプレミス DNS リゾルバーで条件付き転送を設定します。	解決のためにオンプレミス環境 AWS から に DNS クエリを送信するには、インバウンドエンドポイント IP アドレスを指すようにオンプレミス DNS リゾルバーで条件付き転送を設定する必要があります。これにより、DNS リゾルバーは、Route AWS 53 Resolver による解決のために、ホストされたドメイン (の場合など`prod.aws.example.com`) のすべての DNS クエリをインバウンドエンドポイント IP アドレスに転送するように指示されます。	ネットワーク管理者

ハイブリッド環境でend-to-endの DNS 解決を検証する

タスク	説明	必要なスキル
からオンプレミス環境 AWS への DNS 解決をテストします。	転送ルールが関連付けられている VPC 内のインスタンスから、オンプレミスホストドメイン (の場合など`db.onprem`	ネットワーク管理者

タスク	説明	必要なスキル
	`.example.com`) の DNS クエリを実行します。	
オンプレミス環境から への DNS 解決をテストします AWS。	オンプレミスサーバーから、AWSがホストするドメインの DNS 解決を実行します (の場合など`ec2.prod.aws.example.com`)。	ネットワーク管理者

関連リソース

- Amazon VPC のハイブリッドクラウド DNS オプション (AWS ホワイトペーパー)
- プライベートホストゾーンの使用 (Route 53 ドキュメント)
- Route 53 Resolver の開始方法 (Route 53 ドキュメント)
- Route 53 Resolver を使用してマルチアカウント環境で DNS 管理を簡素化する (AWS ブログ記事)
- Amazon Route 53 Profiles と複数の VPCs を使用して DNS 管理を統一する AWS アカウント (AWS ブログ記事)
- マルチアカウント DNS 環境を Amazon Route 53 Profiles に移行する (AWS ブログ記事)
- スケーラブルなマルチアカウント AWS 環境での Amazon Route 53 プロファイルの使用 (AWS ブログ記事)

ELB ロードバランサーに TLS 終了が必要であることを確認

作成者: Priyanka Chaudhary (AWS)

概要

Amazon Web Services (AWS) クラウドでは、Elastic Load Balancing (ELB) が、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、IP アドレス、AWS Lambda 関数などの複数のターゲットにわたって受信したアプリケーショントラフィックを自動的に分散します。ロードバランサーはリスナーを使用して、ユーザーからトラフィックを受け入れるために、ロードバランサーが使用するポートとプロトコルを定義します。Application Load Balancer はアプリケーションレイヤーでルーティングを決定し、HTTP/HTTPS プロトコルを使用します。Classic Load Balancer は、TCP または Secure Sockets Layer (SSL) プロトコルを使用するトランスポート層、または HTTP/HTTPS を使用してアプリケーション層でルーティングを決定します。

パターンは、Application Load Balance と Classic Load Balancerの複数のイベントタイプを検査する、セキュリティ制御を提供します。関数が呼び出される場合、AWS Lambda はイベントを検査し、ロードバランサーが準拠していることを確認します。

この関数は、次のAPI呼び出しでAmazon CloudWatch Eventsを初期化します: 「CreateLoadBalancer」 、 「CreateLoadBalancerListeners」 、 「DeleteLoadBalancerListeners」 、 「CreateLoadBalancerPolicy」 、 「SetLoadBalancerPoliciesOfListener」 、 「CreateListener」 、 「DeleteListener」 、及び 「ModifyListener」 。イベントが3つの API のいずれかを検出する場合、Python スクリプトを実行する AWS Lambda を呼び出します。Python スクリプトは、リスナーに SSL 証明書が含まれているか、および適用されるポリシーがTransport Layer Security (TLS) を使用しているかを評価します。SSL ポリシーが TLS 以外であると判断された場合、関数から Amazon Simple Notification Service (Amazon SNS) 通知が関連の情報を持つユーザーに送信されます。

前提条件と制限

前提条件

- アクティブなAWS アカウント

機能制限

- このセキュリティコントロールでは、ロードバランサーリスナーが更新されない限り、既存のロードバランサーをチェックしません。

- このセキュリティ制御はリージョンごとに行われます。監視する AWS リージョンごとにデプロイする必要があります。

アーキテクチャ

ターゲット アーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、モニタリングする複数のアカウントにこのテンプレートをデプロイできます。

ツール

AWS サービス

- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。
- 「Amazon S3」 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージ配信や送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、以下の添付ファイルが含まれます。

- `ELBRequirestlstermination.zip` — セキュリティコントロール用の Lambda コード。
- `ELBRequirestlstermination.yml` — イベントとLambda 関数をセットアップする CloudFormation テンプレート。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを定義します。	「Amazon S3 コンソール」で、Lambda コードの .zip ファイルをホストする S3 バケットを選択、または作成します。この S3 バケットは、評価したいロードバランサーと同じ AWS リージョンに存在する必要があります。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されます。S3 バケット名には、先頭にスラッシュを含めることはできません。	クラウドアーキテクト
Lambda コードをアップロードします。	添付ファイルセクションで提供されている Lambda コード (`ELBRequirestlstermination.zip` ファイル) を S3 バケットにアップロードします。	クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートを起動します。	S3 バケットと同じ「AWS リージョンで AWS CloudFormation コンソール」を開き、添付のテンプレート `ELBRequirestlstermination.yml` をデプロイします。AWS CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントの「AWS CloudFormation コンソールでのスタックの作成」を参照してください。	クラウドアーキテクト
テンプレートのパラメータを入力します。	テンプレートを起動すると、次の情報の入力を求められます。 • S3 バケット: 最初のエピックで作成、または選択したバケットを指定します。これは、添付された Lambda コード (`ELBRequirestlstermination.zip` ファイル) をアップロードした場所です。 • S3 キー: S3 バケットの Lambda .zip ファイルの位置を指定します(例えば, `ELBRequirestlstermination.zip` または `controls/ELBRequir`	クラウドアーキテクト

タスク	説明	必要なスキル
	estlstermination.zip)。先頭にスラッシュを含めないでください。 • 通知メールアドレス: Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。 • Lambda ロギングレベル: Lambda 関数のロギングレベルと頻度を指定します。[情報] を使用して、進行状況に関する詳細な情報メッセージをログに記録し、引き続きデプロイを続行できるエラーイベントには「エラー」、潜在的に有害な状況の場合は「警告」を使用します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定したメールアドレスにサブスクリプションメールが送信されます。違反通知の受信を開始するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「AWS CloudFormation コンソールでのスタックの作成」(AWS CloudFormation ドキュメント)
- 「AWS Lambda とは?」(AWS Lambda ドキュメント)
- 「Classic Load Balancer とは ?」(ELB ドキュメント)
- 「Application Load Balancer とは?」(ELB ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Splunk を使用して AWS Network Firewall のログとメトリクスを表示する

作成者: Ivo Pinto

概要

多くの組織は、さまざまなソースからのログとメトリクスを一元的に集約および可視化するツールとして Splunk Enterprise を使用しています。このパターンは、Splunk アドオン for AWS を使用して Amazon CloudWatch Logs から AWS Network Firewall ログとメトリクスを取得するように Splunk を設定するのに役立ちます。 Amazon CloudWatch

これを実現するには、読み取り専用の AWS Identity and Access Management (IAM) ロールを作成します。Splunk Add-On for AWS は、このロールを使用して CloudWatch にアクセスします。CloudWatch からメトリクスとログを取得するように Splunk Add-On for AWS を設定します。最後に、取得したログデータとメトリクスから Splunk で視覚化を作成します。

前提条件と制限

前提条件

- Splunk アカウント
- Splunk Enterprise インスタンス、バージョン 8.2.2 以降
- アクティブな AWS アカウント
- Network Firewall、CloudWatch Logs にログを送信するように設定および設定

制約事項

- Splunk Enterprise は、AWS クラウド内の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターとしてデプロイする必要があります。
- Amazon EC2 で自動的に検出された IAM ロールを使用してデータを収集することは、AWS 中国リージョンではサポートされていません。

アーキテクチャ

この図表は、以下を示すものです：

1. Network Firewall は CloudWatch Logs にログを発行します。

2. Splunk Enterprise は CloudWatch からメトリクスとログを取得します。

このアーキテクチャのメトリクスとログの例を入力するために、ワークロードは Network Firewall エンドポイントを通過してインターネットに移動するトラフィックを生成します。これは、ルートテーブルの使用によって実現されます。このパターンではワークロードとして単一の Amazon EC2 インスタンスが使用されますが、Network Firewall が CloudWatch Logs にログを送信するように設定されている場合、このパターンは任意のアーキテクチャに適用できます。

このアーキテクチャでは、別の Virtual Private Cloud (VPC) の Splunk Enterprise インスタンスも使用します。ただし、Splunk インスタンスは、CloudWatch APIs に到達できる限り、ワークロードと同じ VPC など、別の場所に配置できます。

ツール

AWS サービス

- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、 AWS からのログを一元化することを支援して、ログを監視して安全にアーカイブできるようにします。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Network Firewall は、ステートフルでマネージド型のネットワークファイアウォールならびに侵入検知および防止サービスです。

その他のツール

- 「Splunk」 はログデータのモニタリング、視覚化、分析に役立ちます。

エピック

IAM ロールを作成する

タスク	説明	必要なスキル
IAM ポリシーを作成します。	「JSON エディタを使用してポリシーを作成する」の手順に従って、CloudWatch Logs	AWS 管理者

タスク	説明	必要なスキル
	データおよび CloudWatch メトリクスへの読み取り専用アクセスを許可する IAM ポリシーを作成します。以下の ポリシーを JSON エディタに貼り付けます。 `{ "Statement": [ { "Action": [ "cloudwatch:List*", "cloudwatch:Get*", "network-firewall:List*", "logs:Describe*", "logs:Get*", "logs:List*", "logs:StartQuery", "logs:StopQuery", "logs:TestMetricFilter", "logs:FilterLogEvents", "network-firewall:Describe*" ], "Effect": "Allow",`	

タスク	説明	必要なスキル
	"Resource": "*" }], "Version": "2012-10-17" }	
新しい IAM ロールを作成します。	「AWS のサービスにアクセス許可を委任するロールの作成」の手順に従って、AWS 用 Splunk アドオンが CloudWatch へのアクセスに使用する IAM ロールを作成します。アクセス許可ポリシーで、前に作成したポリシーを選択します。	AWS 管理者
Splunk クラスターの EC2 インスタンスに IAM ロールを割り当てます。	1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。 2. ナビゲーションペインで、[インスタンス] を選択します。 3. Splunk クラスター内の EC2 インスタンスを選択します。 4. アクション、セキュリティを選択し、IAM ロールを変更します。 5. 前に作成した IAM ロールを選択し、保存を選択します。	AWS 管理者

AWS 用の Splunk アドオンをインストールする

タスク	説明	必要なスキル
アドオンをインストールします。	1. Splunk ダッシュボードで、Splunk Apps に移動します。 2. Amazon Web Services の Splunk アドオンを検索します。 3. [インストール] を選択します。 4. Splunk 認証情報を入力します。	Splunk 管理者
AWS 認証情報を設定します。	1. Splunk ダッシュボードで、Splunk Add-on for AWS に移動します。 2. [設定] を選択します。 3. Autodiscovered IAM Role 列で、前に作成した IAM ロールを選択します。 詳細については、Splunk ドキュメントの「Splunk プラットフォームインスタンス内で IAM ロールを検索する」を参照してください。	Splunk 管理者

CloudWatch への Splunk アクセスを設定する

タスク	説明	必要なスキル
CloudWatch Logs からの Network Firewall ログの取得を設定します。	1. Splunk ダッシュボードで、Splunk Add-on for AWS に移動します。 2. 入力 を選択します。 3. Create New Input を選択します。 4. リストで、カスタムデータ型を選択し、CloudWatch Logs を選択します。 5. Network Firewall ログの名前、AWS アカウント、AWS リージョン、およびロググループを指定します。 6. [Save] を選択します。 デフォルトでは、Splunk は 10 分ごとにログデータを取得します。これは、詳細設定で設定可能なパラメータです。詳細については、Splunk ドキュメントの「Splunk Web を使用して CloudWatch Logs 入力を設定する」を参照してください。	Splunk 管理者
CloudWatch からの Network Firewall メトリクスの取得を設定します。	1. Splunk ダッシュボードで、Splunk Add-on for AWS に移動します。 2. 入力 を選択します。	Splunk 管理者

タスク	説明	必要なスキル
	3. 新しい入力の作成 を選択します。 4. リストで、CloudWatch を選択します。 5. Network Firewall メトリクスの名前、AWS アカウント、および AWS リージョンを指定します。 6. メトリクス設定の横にある詳細モードで編集を選択します。 7. (オプション) 事前設定された名前空間をすべて削除します。 8. 名前空間の追加を選択し、AWS/NetworkFirewall という名前を付けます。 9. ディメンション値に以下を追加します。 `[{"AvailabilityZone":[".*"],"Engine":[".*"],"FirewallName":[".*"]}]` 10.メトリクス で、すべて を選択します。 11.メトリクス統計で、合計 を選択します。 12[OK] を選択します。 13[Save] を選択します。 デフォルトでは、Splunk は 5 分ごとにメトリクスデータを	

タスク	説明	必要なスキル
	取得します。これは、詳細設定で設定可能なパラメータです。詳細については、Splunk ドキュメントの「Splunk Web を使用して CloudWatch 入力を設定する」を参照してください。	

クエリを使用して Splunk ビジュアライゼーションを作成する

タスク	説明	必要なスキル
上位のソース IP アドレスを表示します。	1. Splunk ダッシュボードで、検索とレポートに移動します。 2. ここに検索を入力するボックスに、次のように入力します。 `sourcetype="aws:cloudwatchlogs" | top event.src_ip` このクエリは、トラフィックが最も多い送信元 IP アドレスのテーブルを降順に表示します。 3. グラフィカル表現の場合は、視覚化を選択します。	Splunk 管理者
パケット統計を表示します。	1. Splunk ダッシュボードで、検索とレポートに移動します。	Splunk 管理者

タスク	説明	必要なスキル
	2. ここに検索を入力するボックスに、次のように入力します。 `sourcetype="aws:cloudwatch"| timechart sum(Sum) by metric_name` このクエリは、1 分ReceivedPackets あたりのメトリクス DroppedPackets 、PassedPackets 、および のテーブルを表示します。 3. グラフィカル表現の場合は、視覚化を選択します。	

タスク	説明	必要なスキル
最も使用されているソースポートを表示します。	1. Splunk ダッシュボードで、検索とレポートに移動します。 2. ここに検索を入力するボックスに、次のように入力します。 `sourcetype="aws:cloudwatchlogs" | top event.dest_port` このクエリは、トラフィックが最も多いソースポートのテーブルを降順で表示します。 3. グラフィカル表現の場合は、視覚化を選択します。	Splunk 管理者

関連リソース

AWS ドキュメント

- AWS サービスにアクセス許可を委任するロールの作成 (IAM ドキュメント)
- 「IAM ポリシーの作成」(IAM ドキュメント)
- AWS Network Firewall でのログ記録とモニタリング (Network Firewall ドキュメント)
- AWS Network Firewall のルートテーブル設定 (Network Firewall ドキュメント)

ブログの投稿

- AWS Network Firewall デプロイモデル

「AWS Marketplace」

- Splunk Enterprise Amazon マシンイメージ (AMI)

その他のパターン

- Session Manager と Amazon EC2 Instance Connect により踏み台ホストにアクセス
- AWS Fargate、AWS PrivateLink、Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします。
- AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします
- AWS Managed Microsoft AD およびオンプレミスの Microsoft Active Directory を使用して DNS 解決を一元化する
- IPv4 および IPv6 対応セキュリティグループのイングレスルール内の単一ホストネットワークエントリを確認する
- 、Angular AWS Amplify、および Module Federation を使用してマイクロフロントエンド用のポータルを作成する
- AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイする
- プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API を内部 Web サイトにデプロイする
- を使用してパブリックサブネットの検出属性ベースのアクセスコントロールをデプロイする AWS Config
- パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする
- Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする
- AWS Transit Gateway Connect を使用して VRF を AWS に拡張する
- F5 BIG-IPワークロードをAWS クラウド上のF5 BIG-IP VEに移行する
- Amazon EKS Auto Mode を有効にするときに NGINX Ingress Controller を移行する
- 非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約
- サービスコントロールポリシーを使用して、アカウントレベルでのインターネットアクセスを防止する
- AWS Network Firewall から Slack チャネルにアラートを送信
- Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する
- AWS Elastic Disaster Recoveryで Oracle JD Edwards EnterpriseOne のディザスタリカバリをセットアップする
- BMC ディスカバリークエリを使用して移行計画のために移行データを抽出

- Network Firewall を使用して、アウトバウンドトラフィックのサーバー名表示から DNS ドメイン名をキャプチャする

コンテンツ配信

トピック

- AWS Firewall Manager と Amazon Data Firehose を使用して Splunk に AWS WAF ログを送信する
- Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する
- その他のパターン

AWS Firewall Manager と Amazon Data Firehose を使用して Splunk に AWS WAF ログを送信する

マイケル・フリーデンタール（AWS）、アマン・カウル・ガンジー（AWS）、J・J・ジョンソン（AWS）によって作成されました

概要

これまで、データを Splunk に移動する方法には、プッシュアーキテクチャとプルアーキテクチャの 2 つがありました。プルアーキテクチャでは再試行によるデータ配信が保証されますが、データをポーリングする専用のリソースが Splunk に必要になります。プルアーキテクチャはポーリングを行うため、通常はリアルタイムではありません。のプッシュアーキテクチャは、通常、レイテンシが低く、拡張性が高く、運用の複雑さとコストが削減されます。ただし、配信を保証することはできません。通常、エージェントが必要です。

Splunk と Amazon Data Firehose の統合により、HTTP イベントコレクター (HEC) を介してリアルタイムのストリーミングデータが Splunk に配信されます。この統合には、プッシュアーキテクチャとプルアーキテクチャの両方のメリットがあります。再試行によるデータ配信が保証され、ほぼリアルタイムで、レイテンシーが低く、複雑性も低くなります。HEC は、HTTP または HTTPS 経由で Splunk にデータを迅速かつ効率的に直接送信します。HEC はトークンベースなので、アプリケーションやサポートファイルに認証情報をハードコーディングする必要はありません。

AWS Firewall Manager ポリシーでは、すべてのアカウントのすべての AWS WAF ウェブ ACL トラフィックのログ記録を設定でき、Firehose 配信ストリームを使用してそのログデータを Splunk に送信し、モニタリング、可視化、分析を行うことができます。ソリューションは次の利点があります。

- すべてのアカウントの AWS WAF ウェブ ACL トラフィックの集中管理とログ記録
- Splunk と 1 つの との統合 AWS アカウント
- スケーラビリティ
- ログデータをほぼリアルタイムで配信
- サーバーレスソリューションの使用によるコストの最適化により、未使用のリソースにお金を払う必要がなくなります。

前提条件と制限

前提条件

- の組織の一部 AWS アカウント であるアクティブな AWS Organizations。

- Firehose でログ記録を有効にするには、次のアクセス許可が必要です。
  - `iam:CreateServiceLinkedRole`
  - `firehose:ListDeliveryStreams`
  - `wafv2:PutLoggingConfiguration`
- AWS WAF とそのウェブ ACLs を設定する必要があります。手順については、「 の開始方法 AWS WAF」を参照してください。
- AWS Firewall Manager を設定する必要があります。手順については、AWS Firewall Manager「 の前提条件」を参照してください。
- の Firewall Manager セキュリティポリシーを設定 AWS WAF する必要があります。手順については、「 AWS Firewall ManagerAWS WAF ポリシーの開始方法」を参照してください。
- Splunk には、Firehose がアクセスできるパブリック HTTP エンドポイントを設定する必要があります。

制約事項

- は、 の 1 つの組織で管理 AWS アカウント する必要があります AWS Organizations。
- ウェブ ACL は、配信ストリームと同じリージョンである必要があります。Amazon CloudFront のログをキャプチャする場合は、米国東部 (バージニア北部) リージョン に Firehose 配信ストリームを作成しますus-east-1。
- Firehose 用 Splunk アドオンは、有料の Splunk Cloud デプロイ、分散型の Splunk Enterprise デプロイ、および単一インスタンスの Splunk Enterprise デプロイで使用できます。このアドオンは、Splunk Cloud の無料トライアルデプロイではサポートされていません。

アーキテクチャ

ターゲットテクノロジースタック

- Firewall Manager
- Firehose
- Amazon Simple Storage Service (Amazon S3)
- AWS WAF
- Splunk

ターゲットアーキテクチャ

次の図は、Firewall Manager を使用してすべての AWS WAF データを一元的にログに記録し、Firehose 経由で Splunk に送信する方法を示しています。

1. AWS WAF ウェブ ACLsファイアウォールログデータを Firewall Manager に送信します。
2. Firewall Manager は、ログデータを Firehose に送信します。
3. Firehose 配信ストリームは、ログデータを Splunk と S3 バケットに転送します。S3 バケットは、Firehose 配信ストリームでエラーが発生した場合のバックアップとして機能します。

自動化とスケール

このソリューションは、組織内のすべての AWS WAF ウェブ ALCs をスケールして対応できるように設計されています。同じ Firehose インスタンスを使用するようにすべてのウェブ ACLs を設定できます。ただし、複数の Firehose インスタンスを設定して使用する場合は、設定できます。

ツール

AWS のサービス

- AWS Firewall Manager は、 のアカウントとアプリケーション全体でファイアウォールルールを一元的に設定および管理するためのセキュリティ管理サービスです AWS Organizations。
- Amazon Data Firehose は AWS のサービス、Splunk などのサポートされているサードパーティーサービスプロバイダーが所有する他の HTTP エンドポイント、カスタム HTTP エンドポイント、HTTP エンドポイントにリアルタイムのストリーミングデータを配信するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS WAF は、保護されたウェブアプリケーションリソースに転送される HTTP および HTTPS リクエストをモニタリングするのに役立つウェブアプリケーションファイアウォールです。

その他のツール

- 「Splunk」 はログデータのモニタリング、視覚化、分析に役立ちます。

エピック

Splunk の設定

タスク	説明	必要なスキル
Splunk App for をインストールします AWS。	1. Splunk ヘビーフォワーダーにサインインします。デフォルトの URL は `http://<IP address>:8000` です。 2. 左側のナビゲーションのアプリの横にあるギアボタンを選択します。 3. その他のアプリを閲覧を選択します。 4. AWS を検索します。 5. Splunk App for AWS で、インストールを選択します。 6. Splunk.com のサインイン認証情報を入力し、利用規約に同意して、[ログインしてインストール] を選択します。 7. [完了] をクリックします。	セキュリティ管理者、Splunk 管理者
のアドオンをインストールします AWS WAF。	前述の手順を繰り返して、Splunk 用の AWS Web Application Firewall アドオンをインストールします。	セキュリティ管理者、Splunk 管理者
Firehose 用の Splunk アドオンをインストールして設定します。	1. Firehose 用の Splunk アドオンをインストールして設定します。インストールと構成の一環として、Splunk プラットフォームに必要な	セキュリティ管理者、Splunk 管理者

タスク	説明	必要なスキル
	場合は HTTP イベントコレクターをセットアップし、ログデータをインデクサーに送信するためのインフラストラクチャを準備します。お使いの Splunk デプロイメントに対応する手順を参照します。 • 「Splunk クラウドデプロイ」(Splunk ドキュメント) • 「分散型 Splunk エンタープライズデプロイ」(Splunk ドキュメント) • 「シングルインスタンスの Splunk エンタープライズデプロイ」(Splunk ドキュメント) **⚠ Important** Splunk アドオンをインストールして設定したら、この手順を停止します。Splunk プラットフォームにデータを送信するように Firehose を設定する手順を実行しないでください。 2. HTTP イベントコレクタートークンと HTTP エンドポイントをメモしておいてく	

タスク	説明	必要なスキル
	ださい。この値は、後で配信ストリームを設定するときに必要になります。	

Firehose 配信ストリームを作成する

タスク	説明	必要なスキル
Firehose に Splunk 送信先へのアクセス権を付与します。	Firehose が Splunk の送信先にアクセスし、ログデータを S3 バケットにバックアップすることを許可するアクセスポリシーを設定します。詳細については、「Firehose に Splunk 送信先へのアクセス権を付与する」を参照してください。	セキュリティ管理者
Firehose 配信ストリームを作成します。	ウェブ ACLs を管理するのと同じアカウントで AWS WAF、Firehose で配信ストリームを作成します。配信ストリームを作成するときは、IAM ロールが必要です。Firehose は IAM ロールを引き受け、指定された S3 バケットへのアクセスを取得します。手順については、「配信ストリームの作成」 を参照してください。次の点に注意してください: • 配信ストリーム名の先頭には、`aws-waf-logs-` を付ける必要があります。	セキュリティ管理者

タスク	説明	必要なスキル
	• ソースには、ダイレクト PUT を選択します。 • S3 バックアップモードの場合は、Backup all events を選択し、既存のバケットを選択するか、新しいバケットを作成します。 • 送信先については、Firehose ドキュメントの「送信先の Splunk を選択する」の指示に従います。Splunk エンドポイントとエンドポイントタイプの値の詳細については、Splunk ドキュメントの「Amazon Data Firehose の設定」を参照してください。 HTTP イベントコレクターで設定したトークンごとに、このプロセスを繰り返します。	
配信ストリームをテストします。	配信ストリームをテストして、適切に設定されていることを確認します。手順については、Firehose ドキュメントの「送信先として Splunk を使用してテストする」を参照してください。	セキュリティ管理者

データを口グに記録するようにFirewall Manager を設定

タスク	説明	必要なスキル
Firewall Manager のポリシーを設定します。	Firewall Manager ポリシーは、ログ記録を有効にし、正しい Firehose 配信ストリームにログを転送するように設定する必要があります。詳細と手順については、「AWS WAF ポリシーのログ記録の設定」を参照してください。	セキュリティ管理者

関連リソース

AWS リソース

- ウェブ ACL トラフィックのログ記録 (AWS WAF ドキュメント)
- AWS WAF ポリシーのログ記録の設定 (AWS WAF ドキュメント)
- チュートリアル: Amazon Data Firehose を使用して Splunk に VPC フローログを送信する (Firehose ドキュメント)
- Amazon Data Firehose を使用して VPC フローログを Splunk にプッシュするにはどうすればよいですか？ (AWS ナレッジセンター)
- Amazon Data Firehose を使用した Splunk へのデータ取り込みの強化 (AWS ブログ記事)

Splunk ドキュメント

- Amazon Data Firehose 用の Splunk アドオン

Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する

作成:エンジェル・エマニュエル・ヘルナンデス・セブリアン

概要

Amazon Web Services (AWS) でホストされている静的コンテンツを配信する場合、Amazon Simple Storage Service (S3) バケットをオリジンとして使用し、Amazon CloudFront を使用してコンテンツを配信することが推奨されます。このソリューションには主に 2 つの利点があります。エッジロケーションで静的コンテンツをキャッシュする利便性と、CloudFront ディストリビューションのウェブアクセスコントロールリスト (ウェブ ACLs) を定義する機能です。これにより、最小限の設定と管理オーバーヘッドでコンテンツへのリクエストを保護するのに役立ちます。

ただし、標準的な推奨アプローチには、アーキテクチャ上の共通の制限があります。環境によっては、仮想プライベートクラウド (VPC) に仮想ファイアウォールアプライアンスをデプロイして、静的コンテンツを含むすべてのコンテンツを検査したい場合があります。標準的なアプローチでは、トラフィックを VPC 経由でルーティングして検査することはありません。このパターンは、アーキテクチャ上の代替ソリューションとなります。S3 バケットの静的コンテンツの配信には引き続き CloudFront ディストリビューションを使用しますが、トラフィックはApplication Load Balancer を使用して VPC 経由でルーティングされます。次に、AWS Lambda 関数が S3 バケットからコンテンツを取得して返します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- S3 バケットにホストされた静的ウェブサイト内容

制約事項

- このパターンのリソースは単一の AWS リージョンにある必要がありますが、異なる AWS アカウントにプロビジョニングできます。
- 制限は、Lambda 関数が受信および送信できるリクエストとレスポンスの最大サイズにそれぞれ適用されます。詳細については、「ターゲットとしての Lambda 関数の制限」 (Elastic Load Balancing ドキュメント)」を参照してください。

- このアプローチを使用するときは、パフォーマンス、スケーラビリティ、セキュリティ、費用対効果のバランスを取ることが重要です。Lambda はスケーラビリティが高いにもかかわらず、Lambda の同時呼び出しの数が最大クォータを超えると、一部のリクエストがスロットされます。詳細については、Lambda のクォータ (Lambda ドキュメント) を参照してください。Lambda を使用する場合は価格設定も考慮する必要があります。Lambda 呼び出しを最小限に抑えるには、CloudFront ディストリビューションのキャッシュを適切に定義します。詳細については、「キャッシュと可用性の最適化」 (CloudFront ドキュメント)」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- CloudFront
- Amazon Virtual Private Cloud (Amazon VPC)
- Application Load Balancer
- Lambda
- Amazon S3

ターゲットアーキテクチャ

次の図表では、CloudFront を使用して S3 バケットから VPC 経由で静的コンテンツを提供する必要がある場合の推奨アーキテクチャを示しています。

1. クライアントは CloudFront ディストリビューションの URL をリクエストして S3 バケット内の特定のウェブサイトファイルを取得します。
2. CloudFront は AWS WAF にリクエストを送信します。AWS WAF は、CloudFront ディストリビューションに適用されたウェブ ACL を使用してリクエストをフィルタリングします。リクエストが有効であると判断された場合、フローは続行されます。リクエストが無効であると判断された場合、クライアントは 403 エラーを受け取ります。
3. CloudFront は内部キャッシュをチェックします。受信したリクエストに一致する有効なキーがある場合、関連する値が応答としてクライアントに送り返されます。そうでなければ、フローは続行されます。
4. CloudFront は、指定した Application Load Balancer の URL にリクエストを転送します。

5. Application Load Balancer には、Lambda 関数に基づいてターゲットグループに関連付けられたリスナーがあります。Application Load Balancer は Lambda 関数を呼び出します。
6. Lambda 関数は S3 バケットに接続して `GetObject` 操作を実行し、コンテンツをレスポンスとして返します。

自動化とスケール

このアプローチを使用して静的コンテンツのデプロイを自動化するには、CI/CD パイプラインを作成して、ウェブサイトをホストする Amazon S3 バケットを更新します。

Lambda 関数は、サービスのクォータと制限の範囲内で、同時リクエストを処理するように自動的にスケーリングします。詳細については、「Lambda 関数のスケーリング」と「Lambda クォータ」(Lambda ドキュメント) を参照してください。CloudFront やApplication Load Balancer など、他の AWS のサービスと特徴量については、AWS がこれらを自動的にスケーリングします。

ツール

- Amazon CloudFront は、世界中のデータセンターネットワークを通じて配信することで、ウェブコンテンツの配信を高速化します。これにより、レイテンシーが減少し、パフォーマンスが向上します。
- 「Elastic Load Balancing (ELB)」は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。このパターンでは、Elastic ロードバランサーを通じてプロビジョニングされた「Application Load Balancer」を使用して、トラフィックを Lambda 関数に転送します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

CloudFront を使用して Amazon S3 から VPC 経由で静的なコンテンツを提供する

タスク	説明	必要なスキル
VPC を作成します。	Application Load Balancer や Lambda 関数など、このパターンでデプロイされたリソースをホストするための VPC を作成します。 手順については、「VPC の作成」（Amazon VPC ドキュメント）を参照してください。	クラウドアーキテクト
AWS WAF Web ACL を作成する	AWS WAF Web ACL を作成する このパターンの後半で、このウェブ ACL を CloudFront ディストリビューションに適用します。手順については、「ウェブ ACL の作成」(AWS WAF ドキュメント) を参照してください。	クラウドアーキテクト
Lambda 関数を作成します。	S3 バケットでホストされている静的コンテンツをウェブサイトとして提供する Lambda 関数を作成します。このパターンの 「追加情報」 セクションに記載されているコードを使用してください。ターゲット S3 バケットを識別するようにコードをカスタマイズします。	AWS 全般
Lambda 関数をアップロードします。	次のコマンドを入力して、Lambda 関数コードを Lambda の.zip ファイルアー	AWS 全般

タスク	説明	必要なスキル
	カイブにアップロードします。 `aws lambda update-function-code \ --function-name \ --zip-file fileb://lambda-alb-s3-website.zip`	
Application Load Balancer を作成します。	Lambda 関数を指すインターネット向けApplication Load Balancer を作成します。手順については、「Lambda 関数のターゲットグループの作成」(Elastic ロードバランサー ドキュメント) を参照してください。高可用性構成の場合は、Application Load Balancer を作成し、それをさまざまなアベイラビリティーゾーンのプライベートサブネットにアタッチします。	クラウドアーキテクト

タスク	説明	必要なスキル
CloudFront ディストリビューションを作成します。	作成したApplication Load Balancer を指す CloudFront ディストリビューションを作成します。 1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/cloudfront/v3/home://https://https://www CloudFront. 2. [Create Distribution] を選択します。 3. [Create Distribution Wizard (ディストリビューションの作成ウィザード)] の最初のページで、[Web (ウェブ)] セクションの [Get Started (今すぐ始める)] を選択します。 4. ディストリビューションの設定項目を指定します。詳細については、「ディストリビューションを作成または更新する場合に指定する値」を参照してください。次の点に注意してください: a. Application Load Balancer をオリジンとして設定します。 b. ディストリビューション設定で、AWS WAF を通じて適用する既存のウェブ ACL を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
	詳細については、「AWS WAF のウェブ ACL」を参照してください。 5. 変更内容を保存します。 6. CloudFront がディストリビューションを作成すると、ディストリビューションの ステータス列の値が、進行中からデプロイ済みに変わります。ディストリビューションを有効にするように選択した場合、ステータスが [Deployed (デプロイ済み)] に切り替わると、リクエストを処理する準備ができています。	

関連リソース

AWS ドキュメント

- 「キャッシュと可用性の最適化」(CloudFront ドキュメント)」を参照してください。
- 「Lambda 関数がターゲットとして機能します」(Elastic ロードバランサードキュメント)
- 「Lambda クォータ」(Lambda ドキュメント)

AWS サービスウェブサイト

- Application Load Balancer
- Lambda
- CloudFront
- Amazon S3
- AWS WAF
- Amazon VPC

追加情報

コード

次の例のLambda 関数が Node.js で記述されます。この Lambda 関数は、ウェブサイトリソースを含む S3 バケットに対して `GetObject` 操作を実行するウェブサーバーとして機能します。

```
/**

 * This is an AWS Lambda function created for demonstration purposes.

 * It retrieves static assets from a defined Amazon S3 bucket.

 * To make the content available through a URL, use an Application Load Balancer with a
 Lambda integration.
 *
 * Set the S3_BUCKET environment variable in the Lambda function definition.
 */

var AWS = require('aws-sdk');

exports.handler = function(event, context, callback) {

    var bucket = process.env.S3_BUCKET;
    var key = event.path.replace('/', '');

    if (key == '') {
        key = 'index.html';
    }

    // Fetch from S3
    var s3 = new AWS.S3();
    return s3.getObject({Bucket: bucket, Key: key},
       function(err, data) {

            if (err) {
                return err;
            }

            var isBase64Encoded = false;
            var encoding = 'utf8';

            if (data.ContentType.indexOf('image/') > -1) {
```

```
                isBase64Encoded = true;
                encoding = 'base64'
            }

            var resp = {
                statusCode: 200,
                headers: {
                    'Content-Type': data.ContentType,
                },
                body: new Buffer(data.Body).toString(encoding),
                isBase64Encoded: isBase64Encoded
            };

            callback(null, resp);
        }
    );
};
```

その他のパターン

- Amazon CloudFront ディストリビューションのアクセスロギング、HTTPS、TLS バージョンを確認する
- gRPC ベースのアプリケーションを Amazon EKS クラスターにデプロイし、Application Load Balancer でアクセスする
- パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする
- Terraform を使用して AWS Wavelength ゾーンにリソースをデプロイする
- Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする
- セルベースのアーキテクチャ用にサーバーレスセルルーターを設定する
- Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める
- Splunk を使用して AWS Network Firewall のログとメトリクスを表示する

データベースとストレージ

トピック

- データベース
- ストレージとバックアップ

データベース

トピック

- リンクサーバーを使用して Amazon EC2 上のMicrosoft SQL サーバーからオンプレミスの Microsoft SQL Server テーブルにアクセスする
- リードレプリカを使用して Amazon RDS Customの Oracle PeopleSoft に HA を追加
- SQL Server データベースを AWS 上の MongoDB Atlas に移行する際のクエリパフォーマンスを評価する
- IaC 原則を使用して Amazon Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する
- AWS Lambda および Task Scheduler を使用して、Amazon EC2 で実行されている SQL Server Express Edition のデータベースタスクを自動化する
- DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する
- 全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント
- Systems Manager と EventBridge を使用した SAP HANA データベースの自動バックアップ
- Python アプリケーションを使用して Amazon DynamoDB の PynamoDB モデルと CRUD 関数を自動的に生成する DynamoDB
- クラウドカストディアンを使用して Amazon RDS へのパブリックアクセスをブロック
- Amazon DynamoDB へのクロスアカウントアクセスを設定する
- AWS 上の SQL Server の Always On アベイラビリティグループで読み取り専用ルーティングを構成する
- pgAdmin の SSH トンネルを使用して接続
- JSON Oracleクエリを PostgreSQL データベース SQL に変換
- を使用してアカウント間で Amazon DynamoDB テーブルをコピーする AWS Backup
- カスタム実装を使用してアカウント間で Amazon DynamoDB テーブルをコピー
- Amazon RDS と Amazon Aurora の詳細なコストと使用状況レポートを作成する
- Aurora PostgreSQL のカスタムエンドポイントを使用して Oracle RAC ワークロードをエミュレートします
- Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする
- 既存の Amazon RDS for PostgreSQL DB インスタンスを暗号化する
- 起動時に Amazon RDS データベースの自動タグ付けを強制する

- DynamoDB テーブルのコストをオンデマンドで見積る
- Amazon DynamoDB テーブルのストレージコストを推定
- AWR レポートを使用して Oracle データベースの Amazon RDS エンジンサイズを推定
- AWS DMS を使用して Amazon RDS for SQL Server テーブルを S3 バケットにエクスポートする
- Aurora PostgreSQL の動的 SQL ステートメントの匿名ブロックを処理
- Aurora PostgreSQL 互換にオーバーロードされた Oracle関数を処理
- DynamoDB でのタグ付けの強制を支援
- AWS DMS と Amazon Aurora によるクロスリージョンディザスタリカバリの実装
- 100 個以上の引数を持つ Oracle 関数とプロシージャを PostgreSQL に移行
- Amazon RDS for Oracle DB インスタンスを AMS を使用する他のアカウントに移行する
- Oracle の OUT バインド変数を PostgreSQL データベースに移行
- 同じホスト名の SAP HSR を使用して SAP HANA を AWS に移行します
- 分散可用性グループを使用して SQL Server を AWS に移行する
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for Oracle に移行
- 暗号化されていない Amazon Aurora インスタンスをモニタリングする
- Amazon CloudWatch を使用して Oracle GoldenGate ログを監視する
- Amazon RDS for Oracle で Oracle Database エンタープライズエディションから標準エディション 2 へリプラットフォームする
- Precisely Connect を使用してメインフレームデータベースを AWS にレプリケート
- Lambda と Secrets Manager を使用して Amazon RDS for PostgreSQL と Aurora PostgreSQL のジョブをスケジュールする
- オンプレミスの SMTP サーバーとデータベースメールを使用して、Amazon RDS for SQL Server データベースインスタンスに通知を送信します。
- AWS の IBM Db2 に SAP のディザスタリカバリをセットアップ
- Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する
- Amazon RDS Custom でアクティブスタンバイデータベースを使用して Oracle E-Business Suite の HA/DR アーキテクチャを設定する
- GTID を使用して Amazon RDS for MySQL と Amazon EC2 上の MySQL との間のデータレプリケーションを設定する
- Amazon RDS Custom for Oracle 上の Oracle PeopleSoft アプリケーションの移行ロール
- アカウント間で Amazon Redshift クラスターから Amazon S3 にデータをアンロードする

- ワークロード別のデータベース移行パターン
- その他のパターン

リンクサーバーを使用して Amazon EC2 上のMicrosoft SQL サーバーからオンプレミスの Microsoft SQL Server テーブルにアクセスする

作成者:ティルマラ・ダサリ (AWS) とエドゥアルド・ヴァレンティム

概要

このパターンは、リンクサーバーを使用して Amazon Elastic Compute Cloud (Amazon EC2) Windows または Linux インスタンスで実行またはホストされている Microsoft SQL Server データベースから、Microsoft Windows 上で実行されているオンプレミスの Microsoft SQL Server データベーステーブルにアクセスする方法を示しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon Linux AMI で実行されている Microsoft SQL Server を使用した Amazon EC2 (Amazon Machine Image)
- オンプレミスのMicrosoft SQL サーバー (Windows) サーバーと Windows または Linux の EC2 インスタンス間の AWS Direct Connect

製品バージョン

- SQL Server 2016 以降

アーキテクチャ

ソーステクノロジースタック

- Windows 上で稼働するオンプレミスのMicrosoft SQL サーバーデータベース
- Windows AMI または Linux AMI 上で動作するMicrosoft SQL サーバーを搭載した Amazon EC2

ターゲットテクノロジースタック

- Amazon Linux AMI で実行されている Microsoft SQL Server を使用した Amazon EC2
- Windows AMI で実行されている Microsoft SQL Server を使用した Amazon EC2

ソースとターゲットのデータベースアーキテクチャ

ツール

- 「Microsoft SQL Server Management Studio (SSMS)」は、SQL Server インフラストラクチャを管理するための統合環境です。SQL Server とやり取りする豊富なスクリプトエディタを備えた、ユーザーインターフェイスとツールグループを備えています。

エピック

Windows SQL Server の SQL Server の認証モードを Windows に変更します。

タスク	説明	必要なスキル
SSMS 経由で Windows SQL サーバーに接続します。		DBA
Windows SQL Server インスタンスのコンテンツ (右クリック) メニューから、SQL Server の認証モードを Windows に変更します。		DBA

Windows MSSQL サービスを再起動します。

タスク	説明	必要なスキル
SQL サービスを再起動します。	1. SSMS オブジェクトエクスプローラーで SQL Server インスタンスを選択します。 2. コンテキスト (右クリック) メニューを開きます。 3. [Restart (再起動)] を選択	DBA

Windows SQL Server で新しいログイン情報を作成し、アクセスするデータベースを選択します。

タスク	説明	必要なスキル
[セキュリティ] タブで、[ログイン] のコンテンツ (右クリック) メニューを開き、新しいログインを選択します。		DBA
[全般] タブで [SQL Server 認証] を選択し、ユーザー名を入力し、パスワードを入力してパスワードを確認して、次回のログイン時にパスワードを変更するオプションをオフにします。		DBA
[サーバーロール] タブで [公開] を選択します。		DBA
[User Mapping] タブで、アクセスするデータベースとスキーマを選択し、データベースを強調表示してデータベースロールを選択します。	public と db_datareader を選択して、データベーステーブルのデータにアクセスします。	DBA
「OK」を選択してユーザーを作成します。		DBA

Windows SQL サーバー IP を Linux SQL Server ホストファイルに追加します。

タスク	説明	必要なスキル
ターミナルウィンドウから Linux SQL Server ボックスに接続します。		DBA

タスク	説明	必要なスキル
/etc/hosts ファイルを開き、SQL Server がインストールされている Windows マシンの IP アドレスを追加します。		DBA
ホストファイルを保存します。		DBA

Linux SQL サーバー上にリンクサーバーを作成します。

タスク	説明	必要なスキル
ストアドプロシージャ master.sys.sp_addlinkedserver と master.dbo.sp_addlinkedsrvlogin を使用してリンクサーバーを作成します。	これらのストアドプロシージャの使用方法の詳細については、「追加情報」セクションを参照してください。	DBA、開発者

作成したリンクサーバーとデータベースを SSMS で検証します。

タスク	説明	必要なスキル
SSMS の Linux SQL Server では、リンクサーバーに移動して更新してください。		DBA
左側のペインで、作成したリンクサーバーとカタログを展開します。	選択した SQL Server データベースがテーブルとビューとともに表示されます。	DBA

Windows SQL サーバーデータベーステーブルにアクセスできることを確認してください。

タスク	説明	必要なスキル
SSMS クエリウィンドウで、「select top 3 * from [sqllin].dms_sample_win.dbo.mlb_data」というクエリを実行します。	FROM 句は computer.database.schema.table という 4 つの部分からなる構文を使用していることに注意してください (たとえば、[sqllin] .master.sys.databases から「SQL2 データベース」という名前を選択)。この例では、ホストファイルに SQL2 のエイリアスを作成したので、角括弧の間に実際の NetBIOS 名を入力する必要はありません。実際の NetBIOS 名を使用する場合は、AWS ではデフォルトで Win-xxxx などの NetBIOS 名が使用され、SQL Server ではダッシュ付きの名前には角括弧が必要であることに注意してください。	DBA、開発者

関連リソース

- 「Linux 上の SQL サーバーのリリースノート」

追加情報

ストアドプロシージャを使用してリンクサーバーを作成する

SSMS は Linux SQL Server 用のリンクサーバーの作成をサポートしていないため、以下のストアドプロシージャを使用して作成する必要があります。

```
EXEC master.sys.sp_addlinkedserver @server= N'SQLLIN' , @srvproduct= N'SQL Server'
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'SQLLIN',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassword='Te
```

注 1: Windows SQL Server で以前に作成したサインイン認証情報をストアドプロシージャ `master.dbo.sp_addlinkedsrvlogin` に入力します。

注 2: `@server` の名前 `SQLLIN` とホストファイルエントリ名 `172.12.12.4 SQLLIN` は同じでなければなりません。

このプロセスを使用して、以下のシナリオで、リンクサーバーを作成できます。

- (このパターンで指定されているように) リンクサーバーを経由して Linux SQL サーバーから Windows SQL サーバーへ
- リンクサーバーを経由して Windows SQL サーバーから Linux SQL サーバーへ
- Linux SQL サーバーからリンクサーバー経由で別の Linux SQL サーバーへ

リードレプリカを使用して Amazon RDS Customの Oracle PeopleSoft に HA を追加

作成者: sampath kathirvel (AWS)

概要

Amazon Web Services (AWS) で「Oracle PeopleSoft」エンタープライズリソースプランニング (ERP) ソリューションを実行するには、「Amazon Relational Database Service (Amazon RDS)」または「Amazon RDS Custom for Oracle」を使用できます。これらは、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシー、カスタムおよびパッケージ化されたアプリケーションをサポートします。移行を計画する際に考慮すべき主な要素については、「AWS 規範ガイダンス」の「Oracle データベースの移行戦略」を参照してください。

この執筆時点では、「RDS Custom for Oracle」は「マルチ AZ」オプションをサポートしていませんが、ストレージ・レプリケーションを使用するHAソリューションとして、「Amazon RDS for Oracle」で利用できます。代わりに、このパターンでは、プライマリデータベースの物理的なコピーを作成して管理するスタンバイデータベースを使用して HA を実現します。このパターンは、Oracle Data Guard を使用してリードレプリカをセットアップし、HA を使って Amazon RDS Custom で PeopleSoft アプリケーションデータベースを実行するステップに焦点を当てます。

同様に、このパターンで、リードレプリカを読み取り専用モードに変更されます。リードレプリカを読み取り専用モードに変更すると、他にも次のような利点があります。

- 読み取り専用のワークロードをプライマリデータベースからオフロードします。
- Oracle Active Data Guard機能によりスタンバイデータベースから正常なブロックを取得することで、破損したブロックを自動的に修復できます。
- Far Sync機能により、REDOログの長距離転送に伴うパフォーマンスのオーバーヘッドなしに、リモート・スタンバイ・データベースを同期状態に保つことができます。

レプリカを読み取り専用モードで使用するには「Oracle Active Data Guard」オプションが必要ですが、これは Oracle Database Enterprise Edition の別途ライセンスされた機能であるため、追加料金がかかります。

前提条件と制限

前提条件

- Amazon RDS Customに既存の PeopleSoft アプリケーション。アプリケーションがない場合は、「Oracle PeopleSoft を Amazon RDS Customに移行」というパターンを参照してください。
- 単一の PeopleSoft アプリケーション層。ただし、このパターンを複数のアプリケーション層で機能するように調整できます。
- Amazon RDS Customには 8 GB 以上のスワップスペースが設定されています。
- リードレプリカを読み取り専用モードに変換し、レポートタスクをスタンバイにオフロードすることに使用するための Oracle Active Data Guard データベースライセンス。詳細については、「Oracle Technology Commercial Price List」を参照してください。

制限事項

- 「RRDS Custom for Oracle」の一般的な制限事項とサポートされていない構成
- Amazon RDS Custom for Oracle リードレプリカに関連する制限

製品バージョン

- Amazon RDS Custom でサポートされている Oracle データベースのバージョンについては、「RDS Custom for Oracle」を参照してください。
- Amazon RDS Customでサポートされている Oracle データベースインスタンスクラスについては、「RDS Custom for Oracle用 DB インスタンスクラスサポート」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon RDS Custom for Oracle
- AWS Secrets Manager
- Oracle Active Data Guard
- Oracle PeopleSoft アプリケーション

ターゲットアーキテクチャ

以下の図は、Amazon RDS Custom DB インスタンスと Amazon RDS Customリードレプリカを示しています。リードレプリカは Oracle Active Data Guard を使用して別のアベイラビリティーゾーンに

レプリケートします。リードレプリカを使用して、プライマリデータベースの読み取りトラフィックをオフロードし、またはレポートを作成することもできます。

AWS で Oracle PeopleSoft を使用する代表的なアーキテクチャについては、「AWS で可用性の高い PeopleSoft アーキテクチャの設定」を参照してください。

ツール

AWS サービス

- 「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。このパターンでは、RDS_DATAGUARD のためシークレット名 `do-not-delete-rds-custom-+<<RDS Resource ID>>+-dg` を含むデータベースユーザーパスワードを Secrets Manager から取得できます。

その他のツール

- 「Oracle Data Guard」 は、スタンバイデータベースの作成、保守、管理とモニタリングに役立ちます。

ベストプラクティス

データ損失ゼロ (RPO=0) を目指すには、`MaxAvailability` Data Guard保護モードとREDO転送 `SYNC+NOAFFIRM` 設定を使用してパフォーマンスを向上させてください。データベース保護モードの選択の詳細については、追加情報セクションを参照してください。

エピック

リードレプリカの作成

タスク	説明	必要なスキル
リードレプリカの作成	Amazon RDS カスタム DB インスタンスのリードレプリカを作成するには、「Amazon RDS ドキュメント」の指示に従い、作成した「Amazon RDS Custom DB インスタンス (前提条件セクションを参照) をソースデータベースとして使用します。 デフォルトでは、Amazon RDS Custom リードレプリカは物理スタンバイとして作成され、マウントされた状態になります。これにより、Oracle Active Data Guard ライセンスへのコンプライアンスが確保されます。 このパターンには、マルチテナントコンテナデータベース (CDB) または非 CDB インスタンスを設定するためのコードを含んでいます。	DBA

Oracle Data Guard の保護モードをMaxAvailabilityに変更します。

タスク	説明	必要なスキル
プライマリデータベースで Data Guard ブローカ設定にアクセスします。	この例では、Amazon RDS Customリードレプリカは `RDS_CUSTOM_ORCL_D` 非 CDB インスタンスと `RDS_CUSTOM_RDSCDB_B` CDB インスタンスに使用されます。非 CDB のデータベースは `orcl_a` (プライマリ) と `orcl_d` (スタンバイ) です。CDB のデータベース名は `rdscdb_a` (プライマリ) と `rdscdb_b` (スタンバイ) です。 RDS Custom リードレプリカには、直接接続することも、プライマリデータベース経由で接続することもできます。データベースのネットサービス名は、`$ORACLE_HOME/network/admin` ディレクトリにある `tnsnames.ora` ファイルにあります。RDS Custom for Oracle は、これらのエントリをプライマリデータベースとリードレプリカに自動的に入力します。 `RDS_DATAGUARD` ユーザーのパスワードは、シークレット名とともに AWS Secrets Manager に保存されます `do-not-de`	DBA

タスク	説明	必要なスキル

	lete-rds-custom-+<<RDS Resource ID>>+-dg。Secrets Manager から取得した SSH (セキュアシェル) キーを使用して RDS カスタムインスタンスに接続する方法の詳細については、「SSH を使用して RDS Custom DB インスタンスに接続する」を参照してください。
	
	Data Guard コマンドライン (dgmgrl) から Oracle Data Guard ブローカ設定にアクセスするには、次のコードを使用します。
	
	非 CDB
	
	`$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_D`
	`DGMGRL for Linux:`
	`Release 19.0.0.0.0 - Production on Fri Sep 30 22:44:49 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.`
	`Welcome to DGMGRL, type "help" for information.`
	`Password:`
	`Connected to "ORCL_D"`
	`Connected as SYSDG.`
	`DGMGRL>`	

タスク	説明	必要なスキル
	DGMGRL> show database orcl_d Database - orcl_d Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Average Apply Rate: 11.00 KByte/s Instance(s): ORCL SUCCESS DGMGRL>	
	CDB -bash-4.2$ dgmgrl C##RDS_DATAGUARD@RDS_CUSTOM_RDSCDB_B DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 11 20:24:11 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for information. Password: Connected to "RDSCDB_B" Connected as SYSDG. DGMGRL>	

タスク	説明	必要なスキル
	DGMGRL> show database rdscdb_b Database - rdscdb_b Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Average Apply Rate: 2.00 KByte/s Real Time Query: OFF Instance(s): RDSCDB Database Status: SUCCESS DGMGRL>	

タスク	説明	必要なスキル
プライマリノードから DGMGRL に接続して、ログ転送設定を変更します。	REDO 転送設定 SYNC+NOAFFIRM に対応するログ転送モードを FastSync に変更します。ロールの切り替え後も設定が有効であることを確認するには、プライマリデータベースとスタンバイデータベースの両方の設定を変更してください。 非 CDB `DGMGRL> DGMGRL> edit database orcl_d set property logxptmode=fastsync; Property "logxptmode" updated DGMGRL> show database orcl_d LogXptMode; LogXptMode = 'fastsync ' DGMGRL> edit database orcl_a set property logxptmode=fastsync; Property "logxptmode" updated DGMGRL> show database orcl_a logxptmode; LogXptMode = 'fastsync ' DGMGRL>`  CDB  `DGMGRL> edit database rdscdb_b set property logxptmode=fastsyn c;DGMGRL> edit database`	DBA

タスク	説明	必要なスキル
	rdscdb_b set property logxptmode=fastsync; Property "logxptmode" updated DGMGRL> show database rdscdb_b LogXptMode; LogXptMode = 'fastsync' DGMGRL> edit database rdscdb_a set property logxptmode=fastsync; Property "logxptmode" updated DGMGRL> show database rdscdb_a logxptmode; LogXptMode = 'fastsync' DGMGRL>	

タスク	説明	必要なスキル

保護モードをMaxAvailabilityに変更します。	保護モードを MaxAvailability プライマリノードから DGMGRL に接続することによる保護モードに変更します。
	
	非 CDB
	
	`DGMGRL> edit configuration set protection mode as maxavailability;`
	`Succeeded.`
	`DGMGRL> show configuration;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`orcl_d - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 38 seconds ago)`
	`DGMGRL>`
	
	CDB
	
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
		DBA

タスク	説明	必要なスキル
	rdscdb_a - Primary database rdscdb_b - Physical standby database Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 57 seconds ago) DGMGRL>	

レプリカのステータスをマウントから読み取り専用に変更し、再実行適用を有効にします。

タスク	説明	必要なスキル
スタンバイデータベースへの再実行適用を停止します。	リードレプリカはデフォルトで MOUNT モードで作成されます。読み取り専用モードで開くには、まずプライマリまたはスタンバイノードから DGMGRL に接続して再実行適用を無効にする必要があります。 非 CDB DGMGRL> show database orcl_dDGMGRL> show database orcl_d Database - orcl_d Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago)	DBA

タスク	説明	必要なスキル
	Average Apply Rate: 11.00 KByte/s Real Time Query: OFF Instance(s): ORCL Database Status: SUCCESS DGMGRL> edit database orcl_d set state=app ly-off; Succeeded. DGMGRL> show database orcl_d Database - orcl_d Role: PHYSICAL STANDBY Intended State: APPLY- OFF Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 42 seconds (computed 1 second ago) Average Apply Rate: (unknown) Real Time Query: OFF Instance(s): ORCL Database Status: SUCCESS DGMGRL> CDB DGMGRL> show configura tionDGMGRL> show configuration Configuration - rds_dg Protection Mode: MaxAvailability Members:	

タスク	説明	必要なスキル
	rdscdb_a - Primary database rdscdb_b - Physical standby database Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 57 seconds ago) DGMGRL> show database rdscdb_b; Database - rdscdb_b Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Average Apply Rate: 2.00 KByte/s Real Time Query: OFF Instance(s): RDSCDB Database Status: SUCCESS DGMGRL> edit database rdscdb_b set state=apply-off; Succeeded. DGMGRL> show database rdscdb_b; Database - rdscdb_b Role: PHYSICAL STANDBY Intended State: APPLY-OFF	

タスク	説明	必要なスキル
	Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Average Apply Rate: (unknown) Real Time Query: OFF Instance(s): RDSCDB Database Status: SUCCESS	

タスク	説明	必要なスキル

リードレプリカインスタンスを読み取り専用モードでオープンします。	TNS エントリを使用してスタンバイデータベースに接続し、プライマリまたはスタンバイノードから接続して読み取り専用モードで開きます。
	
	非 CDB
	
	`$ sqlplus RDS_DATAGUARD@RDS_CUSTOM_ORCL_D as sysdg`
	`-bash-4.2$ sqlplus RDS_DATAGUARD@RDS_CUSTOM_ORCL_D as sysdg`
	`SQL*Plus: Release 19.0.0.0.0 - Production on Fri Sep 30 23:00:14 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982, 2020, Oracle. All rights reserved.`
	`Enter password:`
	`Last Successful login time: Fri Sep 30 2022 22:48:27 +00:00`
	`Connected to:`
	`Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production`
	`Version 19.10.0.0.0`
	`SQL> select open_mode from v$database;`
	`OPEN_MODE`
	`--------------------`
	`MOUNTED`
	`SQL> alter database open read only;`
	`Database altered.`
		DBA

タスク	説明	必要なスキル
	SQL> select open_mode from v$database; OPEN_MODE -------------------- READ ONLY SQL>	
	CDB -bash-4.2$ sqlplus C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_B as sysdg SQL*Plus: Release 19.0.0.0.0 - Productio n on Wed Jan 11 21:14:07 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2022, Oracle. All rights reserved. Enter password: Last Successful login time: Wed Jan 11 2023 21:12:05 +00:00 Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production Version 19.16.0.0.0 SQL> select name,open _mode from v$database; NAME OPEN_MODE --------- --------- ----------- RDSCDB MOUNTED SQL> alter database open read only; Database altered.	

タスク	説明	必要なスキル
	SQL> select name,open _mode from v$database; NAME OPEN_MODE --------- --------- ----------- RDSCDB READ ONLY SQL>	

タスク	説明	必要なスキル

リードレプリカインスタンスで 再実行適用を有効にします。	プライマリまたはスタンバイノードから DGMGR L を使用して、リードレプリカインスタンスで 再実行適用を有効にします。
	
	非 CDB
	
	`$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_D`
	`DGMGRL for Linux:`
	`Release 19.0.0.0.0 -`
	`Production on Fri Sep`
	`30 23:02:16 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982,`
	`2019, Oracle and/or its`
	`affiliates. All rights`
	`reserved.`
	`Welcome to DGMGRL, type`
	`"help" for informati`
	`on.`
	`Password:`
	`Connected to "ORCL_D"`
	`Connected as SYSDG.`
	`DGMGRL>`
	`edit database orcl_d set`
	`state=apply-on;`
	`DGMGRL> edit database`
	`orcl_d set state=app`
	`ly-on;`
	`Succeeded.`
	`DGMGRL> show database`
	`orcl_d`
	`Database - orcl_d`
	`Role: PHYSICAL STANDBY`
	`Intended State: APPLY-`
	`ON`
		DBA

タスク	説明	必要なスキル
	Transport Lag: 0 seconds (computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Average Apply Rate: 496.00 KByte/s Real Time Query: ON Instance(s): ORCL Database Status: SUCCESS DGMGRL>	
	CDB -bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_B -bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_B DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 11 21:21:11 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "RDSCDB_B " Connected as SYSDG.	

タスク	説明	必要なスキル
	DGMGRL> edit database rdscdb_b set state=apply-on; Succeeded. DGMGRL> show database rdscdb_b Database - rdscdb_b Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Average Apply Rate: 35.00 KByte/s Real Time Query: ON Instance(s): RDSCDB Database Status: SUCCESS DGMGRL> show database rdscdb_b Database - rdscdb_b Role: PHYSICAL STANDBY Intended State: APPLY-ON Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Average Apply Rate: 16.00 KByte/s Real Time Query: ON Instance(s): RDSCDB	

タスク	説明	必要なスキル
	`Database Status: SUCCESS DGMGRL>`	

関連リソース

- 「Amazon RDS を Oracle PeopleSoft Database として設定」AWS ホワイトペーパー)
- Oracle Data Guard Broker ガイド (Oracle リファレンスドキュメント)
- 「Oracle Data Guard 概要および管理」(Oracleのリファレンスドキュメント)

追加情報

データベース保護モードを選択

Oracle Data Guardには、可用性、保護、パフォーマンスの要件に基づいてData Guard環境を構成するための 3 つの保護モードが用意されています。次の表は、これら 3 つのモードをまとめたものです。

保護モード	転送設定を再実行	説明
最高のパフォーマンス	ASYNC	プライマリデータベースで発生するトランザクションでは、再実行データが非同期的に送信され、スタンバイデータベースの再実行ログに書き込まれます。したがって、パフォーマンスへの影響は最小限に控えられます。 非同期の口グ配布のため、MaxPerformance は RPO=0 を指定できません。
最大限の保護	SYNC+AFFIRM	プライマリデータベース上のトランザクションでは、トラ

		ンザクションが確認される前に、再実行データが同期的に転送され、ディスク上のスタンバイデータベース再実行ログに書き込まれます。スタンバイデータベースが使用できなくなった場合、プライマリデータベースは自動的にシャットダウンし、トランザクションが確実に保護されます。
最大限の可用性	SYNC+AFFIRM	これは、スタンバイ・データベースから確認応答が受信されない場合を除いて、MaxProtection モードと似ています。この場合は、再実行ストリームを同期化されたスタンバイ・データベースにに再び書き込めるようになるまで、プライマリデータベースの可用性を維持する MaxPerformance モードになっているかのように動作します。

SYNC+NOAFFIRM	プライマリデータベース上のトランザクションでは、再実行はスタンバイデータベースに同期的に転送され、プライマリデータベースは再実行がスタンバイディスクに書き込まれるのではなく、スタンバイデータベースで再実行が受信されたことを確認するのみです。このモード (FastSync とも呼ばれる) は、複数の同時障害が発生する特殊なケースではデータが失われる可能性がありますが、パフォーマンス上のメリットがあります。

RDS Custom for Oracle のリードレプリカは、Oracle Data Guard のデフォルトの保護モードでもある最大パフォーマンス保護モードで作成されます。最大パフォーマンスモードは、プライマリデータベースへのパフォーマンスへの影響を最小限に抑え、秒単位で測定される目標復旧時点 (RPO) の要件を満たすのに役立ちます。

データ損失ゼロ (RPO=0) の目標を達成するには、パフォーマンスを向上させるために、Oracle Data Guard保護モードをREDO転送の SYNC+NOAFFIRM 設定で MaxAvailability にカスタマイズできます。プライマリ・データベースでのコミットは、対応する REDO ベクトルがスタンバイ・データベースに正常に転送された後にのみ確認されるため、プライマリ・インスタンスとレプリカ間のネットワーク遅延は、コミットに敏感なワークロードにとって極めて重要です。リードレプリカを MaxAvailability モードで実行するようにカスタマイズした場合のパフォーマンスへの影響を評価するために、ワークロードの負荷テストを実施することをお勧めします。

リードレプリカをプライマリデータベースと同じアベイラビリティーゾーンにデプロイすると、リードレプリカを別のアベイラビリティーゾーンにデプロイする場合よりもネットワークレイテンシーが低くなります。ただし、プライマリとリードコピーインスタンスの両方が影響を受けるため、同じ可用性ゾーンにマスターインスタンスとリードコピーを配備してもHA要件を満たすことができない場合があります。

SQL Server データベースを AWS 上の MongoDB Atlas に移行する際のクエリパフォーマンスを評価する

Battulga Purevragchaa (AWS)、Krishnakumar Sathyanarayana (PeerIslands US Inc)、Babu Srinivasan (MongoDB) によって作成されました

概要

このパターンは、MongoDB にほぼ現実世界のデータを読み込み、可能な限り本番シナリオに近い状態で MongoDB クエリのパフォーマンスを評価するための指針となります。評価では、リレーショナルデータベースから MongoDB への移行を計画する際に役立つ情報を得ることができます。このパターンでは、「PeerIslandsテストデータジェネレーターとパフォーマンスアナライザー」を使用してクエリのパフォーマンスをテストします。

このパターンは、Microsoft SQL Server を MongoDB に移行する場合に特に役立ちます。スキーマ変換を実行したり、現在の SQL Server インスタンスから MongoDB にデータをロードしたりするのは非常に複雑になる可能性があるためです。代わりに、実際の移行を開始する前に、ほぼ現実世界のデータを MongoDB にロードし、MongoDB のパフォーマンスを理解し、スキーマ設計を微調整することができます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 「MongoDB」アトラスに精通していること
- MongoDB スキーマ
- 一般的なクエリパターン

制約事項

- データのロード時間とパフォーマンスは、MongoDB クラスターインスタンスのサイズによって制限されます。実際のパフォーマンスを理解するために、本番環境での使用が推奨されるインスタンスを選択することをお勧めします。
- PeerIslandsのテストデータジェネレーターとパフォーマンスアナライザーは現在、オンラインのデータロードとクエリのみをサポートしています。オフラインのバッチ処理 (Spark コネクタを使用して MongoDB にデータをロードするなど) はまだサポートされていません。

- PeerIslandsのテストデータジェネレーターとパフォーマンスアナライザーは、コレクション内のフィールドリレーションをサポートしています。コレクション間の関係はサポートしていません。

製品エディション

- このパターンは 「MongoDB アトラス」と「MongoDB エンタープライズアドバンスド」の両方をサポートします。

アーキテクチャ

ターゲットテクノロジースタック

- MongoDB アトラスまたは MongoDB エンタープライズアドバンス

アーキテクチャ

PeerIslandsのテストデータジェネレーターとパフォーマンスアナライザーは、JavaとAngularを使用して構築され、生成されたデータをAmazon Elastic Block Store (Amazon EBS) に保存します。このツールは、テストデータ生成とパフォーマンステストの 2 つのワークフローで構成されています。

- テストデータ生成では、生成する必要のあるデータモデルを JSON で表現したテンプレートを作成します。テンプレートを作成したら、負荷生成設定の定義に従ってターゲットコレクションにデータを生成できます。
- パフォーマンステストでは、プロファイルを作成します。プロファイルは、作成、読み取り、更新、削除 (CRUD) 操作、集約パイプライン、各操作の加重、各段階の所要時間を設定できる多段階のテストシナリオです。プロファイルを作成したら、構成に基づいてターゲットデータベースでパフォーマンステストを実行できます。

PeerIslandsのテストデータジェネレーターとパフォーマンスアナライザーは、データをAmazon EBSに保存するため、ピアリング、許可リスト、プライベートエンドポイントなど、MongoDBがサポートする接続メカニズムを使用してAmazon EBSをMongoDBに接続できます。デフォルトでは、このツールには運用コンポーネントは含まれていません。ただし、必要に応じて Amazon マネージドサービス for Prometheus、Amazon Managed Grafana、Amazon CloudWatch、および AWS Secrets Manager を使用して設定できます。

ツール

- 「PeerIslands テストデータジェネレーターとパフォーマンスアナライザー」には 2 つのコンポーネントが含まれています。Test Data Generator コンポーネントを使用すると、MongoDB スキーマに基づいて、顧客固有の現実世界のデータを生成できます。このツールは完全にUI主導型で、豊富なデータライブラリを備えているため、MongoDBで数十億のレコードを迅速に生成できます。このツールには、MongoDB スキーマのフィールド間の関係を実装する機能もあります。Performance Analyzer コンポーネントを使用すると、顧客固有のクエリや集計を生成し、MongoDB で現実的なパフォーマンステストを実行できます。Performance Analyzer を使用すると、特定のユースケースに合わせた豊富な負荷プロファイルとパラメーター化されたクエリを使用して MongoDB のパフォーマンスをテストできます。

ベストプラクティス

以下のリソースを参照してください。

- 「MongoDB スキーマ設計のベストプラクティス」 (MongoDB 開発者ウェブサイト)
- 「AWS に MongoDB アトラスをデプロイする際のベストプラクティス」 (MongoDB ウェブサイト)
- 「AWS PrivateLink を使用して MongoDB Atlas データプレーンにアプリケーションを安全に接続する」 (AWS ブログ記事)
- 「MongoDB パフォーマンスのベストプラクティスガイド」 (MongoDB ウェブサイト)

エピック

ソースデータを理解する

タスク	説明	必要なスキル
現在の SQL Server ソースのデータベースフットプリントを理解します。	現在の SQL Server フットプリントを把握します。これは、データベースの `INFORMATION` スキーマに対してクエリを実行することで実現できます。テーブルの数と各テーブルのサイズを決	DBA

タスク	説明	必要なスキル
	定します。各テーブルに関連付けられているインデックスを分析します。SQL 分析の詳細については、PeerIslands ウェブサイトの「SQL2Mongo: データ移行の道のり」というブログ記事を参照してください。	
ソーススキーマを理解します。	テーブルスキーマとデータのビジネス表現 (郵便番号、名前、通貨など) を決定します。既存のエンティティ関係 (ER) 図を使用するか、既存のデータベースから ER 図を生成します。詳細については、PeerIslandsウェブサイトのブログ記事「SQL2Mongo: データ移行の道のり」を参照してください。	DBA
クエリパターンを理解します。	使用する SQL クエリのトップ 10 を文書化します。データベースにある performance_schema.events_statements_summary_by_digest テーブルを使用すると、上位のクエリを理解できます。詳細については、PeerIslandsウェブサイトのブログ記事「SQL2Mongo: データ移行の道のり」を参照してください。	DBA

タスク	説明	必要なスキル
SLA のコミットメントを理解します。	データベース運用のターゲットサービスレベルアグリーメント (SLA) を文書化します。一般的な測定値には、クエリの待ち時間や 1 秒あたりのクエリ数などがあります。測定値とその目標は通常、非機能要件 (NFR) 文書に記載されています。	DBA

MongoDB スキーマを定義する

タスク	説明	必要なスキル
ターゲットスキーマを定義します。	ターゲット MongoDB スキーマのさまざまなオプションを定義します。詳細については、MongoDB のドキュメントの「Schemas」を参照してください。テーブルリレーションに基づいたベストプラクティスとデザインパターンを検討します。詳細については、MongoDB「ドキュメントのデータモデルの例とパターン」を参照してください。	MongoDB エンジニア
ターゲットクエリパターンを定義します。	MongoDB クエリとアグリゲーションパイプラインを定義します。これらのクエリは、SQL Server ワークロードについてキャプチャした上位クエリと同等です。MongoDB アグリゲーションパイプライ	MongoDB エンジニア

タスク	説明	必要なスキル
	ンを構築する方法を理解するには、「MongoDB のドキュメント」を参照してください。	
MongoDB インスタンスタイプを定義します。	テストに使用するインスタンスのサイズを決定します。詳細については、「MongoDB のドキュメント」を参照してください。	MongoDB エンジニア

ターゲットデータベースの準備

タスク	説明	必要なスキル
MongoDB アトラスクラスターをセットアップします。	AWS で MongoDB クラスターをセットアップするには、「MongoDB ドキュメント」の指示に従ってください。	MongoDB エンジニア
ターゲットデータベースにユーザーを作成します。	「MongoDB ドキュメント」の指示に従って、MongoDB Atlas クラスターにアクセスとネットワークセキュリティを設定します。	MongoDB エンジニア
AWS で適切なロールを作成し、Atlasのロールベースのアクセス制御を設定します。	必要に応じて、「MongoDB」ドキュメントの指示に従って追加のユーザーを設定します。AWS ロールを使用して認証と認可を設定します。	MongoDB エンジニア
MongoDB アトラスアクセス用のコンパスを設定します。	ナビゲーションとアクセスを容易にするため、「MongoDB	MongoDB エンジニア

タスク	説明	必要なスキル
	Compass GUI ユーティリティ」を設定します。	

テストデータジェネレーターを使用してベースロードを設定します。

タスク	説明	必要なスキル
テストデータジェネレーターをインストールします。	「PeerIsland テストデータジェネレーター」をご使用の環境にインストールします。	MongoDB エンジニア
適切なデータを生成するようにテストデータジェネレーターを設定します。	データライブラリを使用してテンプレートを作成し、MongoDB スキーマの各フィールドに固有のデータを生成します。詳細については、「MongoDB データジェネレーターとパフォーマンス」を参照してください。「アナライザー」ビデオ。	MongoDB エンジニア
テストデータジェネレーターを水平方向にスケールする必要な負荷を生成します。	作成したテンプレートを使用して、必要な並列処理を設定して、ターゲットコレクションに対する負荷生成を開始します。必要なデータを生成するための時間枠とスケールを決定します。	MongoDB エンジニア
MongoDB アトラスでロードを検証します。	MongoDB アトラスに読み込まれたデータを確認します。	MongoDB エンジニア
MongoDB で必要なインデックスを生成します。	クエリパターンに基づいて、必要に応じてインデックスを定義します。詳細について	MongoDB エンジニア

タスク	説明	必要なスキル
	は、「MongoDB のドキュメント」を参照してください。	

パフォーマンステストを実施します。

タスク	説明	必要なスキル
パフォーマンス・アナライザーでロード・プロファイルを設定します。	Performance Analyzer でパフォーマンステストプロファイルを作成するには、特定のクエリとそれに対応する加重、テスト実行時間、ステージを設定します。詳細については、「MongoDB データジェネレーターとパフォーマンス」を参照してください。「アナライザー」ビデオ。	MongoDB エンジニア
パフォーマンステストを実行します。	作成したパフォーマンステストプロファイルを使用して、必要な並列処理を設定して、ターゲットコレクションに対するテストを開始します。パフォーマンステストツールを水平方向にスケールして、MongoDB Atlas に対してクエリを実行します。	MongoDB エンジニア
テスト結果を記録します。	クエリの P95、P99 のレイテンシーを記録します。	MongoDB エンジニア
スキーマとクエリパターンを調整します。	インデックスとクエリパターンを変更して、パフォーマンスの問題に対処します。	MongoDB エンジニア

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。	テストデータジェネレーターとパフォーマンスアナライザーに使用した一時リソースをすべて削除します。	AWS 管理者
パフォーマンステストの結果を更新します。	MongoDB クエリのパフォーマンスを理解し、それを SLA と比較します。必要に応じて、MongoDB スキーマを微調整し、プロセスを再実行します。	MongoDB エンジニア
プロジェクトを完了します。	プロジェクトを終了し、フィードバックを提供します。	MongoDB エンジニア

関連リソース

- GitHub リポジトリ:「S3 からアトラス」
- スキーマ:「MongoDB」スキーマのデザイン
- アグリゲーションパイプライン:「MongoDB」アグリゲーションパイプライン
- MongoDB アトラスのサイジング:「サイジング層の選択」
- ビデオ:「MongoDB データジェネレーター」とパフォーマンス analyzer
- リファレンス:「MongoDB ドキュメンテーション」
- チュートリアル:「MongoDB 開発者ガイド」、「MongoDB ジャンプスタート」
- AWS Marketplace: AWS マーケットプレイスの 「MongoDB アトラス」
- AWS パートナーソリューション:AWS リファレンスデプロイに関する 「MongoDB アトラス」

その他のリソース:

- 「SQL 分析」

- 「MongoDB デベロッパーコミュニティフォーラム」
- 「MongoDB パフォーマンスチューニングに関する質問」
- 「アトラスとRedshift によるオペレーショナル分析」
- 「MongoDB アトラスと AWS Elastic Beanstalk によるアプリケーションのモダナイゼーション」

IaC 原則を使用して Amazon Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する

作成者: Ishwar Chauthaiwale (AWS)、ANKIT JAIN (AWS)、Ramu Jagini (AWS)

概要

Amazon Aurora グローバルデータベースで重要なワークロードを実行する組織では、データベースの更新、移行、スケーリング作業の管理が困難な場合があります。これらのオペレーションをダウンタイムなしでシームレスに実行することは、サービスの可用性を維持し、ユーザーの中断を回避するために不可欠です。

ブルー/グリーンデプロイ戦略は、ブルー (現在の環境) とグリーン (新しい環境) の 2 つの同一の環境を同時に実行できるようにすることで、この課題に対するソリューションを提供します。ブルー/グリーン戦略を使用すると、リスクとダウンタイムを最小限に抑えながら、変更の実装、テストの実行、環境間のトラフィックの切り替えを行うことができます。

このパターンは、Infrastructure as Code (IaC) の原則を使用して、Aurora グローバルデータベースのブルー/グリーンデプロイプロセスを自動化するのに役立ちます。AWS CloudFormation、AWS Lambda、および Amazon Route 53 を使用して、ブルー/グリーンデプロイを簡素化します。信頼性を向上させるために、レプリケーションにグローバルトランザクション識別子 (GTIDsを使用します。GTID ベースのレプリケーションは、バイナリログ (binlog) レプリケーションと比較して、環境間のデータ整合性とフェイルオーバー機能を向上させます。

> **Note**
>
> このパターンは、Aurora MySQL 互換エディション グローバルデータベースクラスターを使用していることを前提としています。代わりに Aurora PostgreSQL 互換を使用している場合は、MySQL コマンドに相当する PostgreSQL を使用してください。

このパターンのステップに従うことで、次のことができます。

- グリーン Aurora グローバルデータベースをプロビジョニングする: CloudFormation テンプレートを使用して、既存のブルー環境をミラーリングするグリーン環境を作成します。
- GTID ベースのレプリケーションを設定する: ブルー環境とグリーン環境を同期させるように GTID レプリケーションを設定します。
- トラフィックをシームレスに切り替える: Route 53 と Lambda を使用して、完全な同期後にトラフィックをブルー環境からグリーン環境に自動的に切り替えます。

- デプロイを確定する: グリーン環境がプライマリデータベースとして完全に動作していることを検証し、レプリケーションを停止して一時的なリソースをクリーンアップします。

このパターンのアプローチには、次の利点があります。

- 重要なデータベースの更新や移行中のダウンタイムを短縮: 自動化により、サービスの中断を最小限に抑えながら、環境間のスムーズな移行が保証されます。
- 迅速なロールバックを有効にする: トラフィックがグリーン環境に切り替わった後に問題が発生した場合は、ブルー環境にすばやく戻り、サービスの継続性を維持できます。
- テストと検証を強化: グリーン環境はライブブルー環境に影響を与えずに完全にテストできるため、本番環境でエラーが発生する可能性が低くなります。
- データの整合性を確保: GTID ベースのレプリケーションはブルー環境とグリーン環境を同期させ、移行中のデータの損失や不整合を防ぎます。
- ビジネス継続性を維持する: ブルー/グリーンデプロイを自動化すると、更新や移行中にサービスを使用できるようにしておくことで、長時間の停止や財務上の損失を回避できます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- ソース Aurora MySQL 互換グローバルデータベースクラスター (ブルー環境) 。グローバルデータベースは、高可用性とディザスタリカバリのためのマルチリージョン設定を提供します。グローバルデータベースクラスターをセットアップする手順については、Aurora ドキュメントを参照してください。
- ソースクラスターで GTID ベースのレプリケーションが有効になっています。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

製品バージョン

- Aurora MySQL 互換 8.0 以降

アーキテクチャ

この図表は、以下を示すものです：

- グローバルデータベースのセットアップ: Aurora グローバルデータベースクラスターは 2 つのクラスターに戦略的にデプロイされます AWS リージョン。この設定により、地理的分散とリージョンの冗長性が可能になり、ディザスタリカバリ機能が強化されます。
- プライマリリージョンからセカンダリリージョンへのレプリケーション: 論理レプリケーションメカニズムにより、プライマリリージョンからセカンダリリージョンへのシームレスなデータ同期が保証されます。このレプリケーションは、地理的距離全体で最小限のレイテンシーでデータの一貫性を維持します。
- クラスター間の GTID ベースのレプリケーション: GTID ベースのレプリケーションは、ブループライマリクラスターとグリーンプライマリクラスター間のトランザクション整合性と順序付けられたデータフローを維持し、信頼性の高いデータ同期を実現します。
- ブループライマリからセカンダリへのレプリケーション: 論理レプリケーションは、ブループライマリクラスターとそのセカンダリクラスター間に堅牢なデータパイプラインを確立します。このレプリケーションにより、継続的なデータ同期と高可用性が可能になります。
- Route 53 DNS 設定: Route 53 ホストゾーンレコードは、すべてのブルーおよびグリーンクラスターデータベースエンドポイントの DNS 解決を管理します。この設定は、シームレスなエンドポイントマッピングを提供し、フェイルオーバーシナリオ中の効率的なトラフィックルーティングを可能にします。

ツール

AWS サービス

- 「Amazon Aurora」はクラウド用に構築されたフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- AWS CloudFormation は、 AWS リソースのモデル化とセットアップに役立つため、それらのリソースの管理に費やす時間を減らし、 が実行されるアプリケーションに集中できます AWS。必要なすべての AWS リソースを記述するテンプレートを作成すると、CloudFormation がそれらのリソースのプロビジョニングと設定を処理します。

- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートするコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

ベストプラクティス

Route 53 のブルー/グリーンデプロイ戦略、GTID ベースのレプリケーション、および加重ルーティングポリシーの理解を深めるために、 AWS ドキュメントを徹底的に確認することをお勧めします。この知識は、データベース移行の効果的な実装と管理、データの一貫性の確保、トラフィックルーティングの最適化に不可欠です。これらの AWS 機能とベストプラクティスを包括的に理解することで、将来の更新を処理し、ダウンタイムを最小限に抑え、回復力のある安全なデータベース環境を維持する準備が整います。

このパターンに を使用する AWS のサービス ためのガイドラインについては、次の AWS ドキュメントを参照してください。

- Amazon Aurora MySQL のベストプラクティス
- AWS CloudFormation ベストプラクティス
- AWS Lambda 関数を使用するためのベストプラクティス
- Amazon Route 53 のベストプラクティス

エピック

グリーン環境を作成する

タスク	説明	必要なスキル
ブルークラスターからスナップショットバックアップを作成します。	ブルー/グリーンデプロイでは、グリーン環境は現在の (ブルー) データベース環境の新しい同一バージョンを表します。グリーン環境を使用して、本番トラフィックを切り替える前に、更新を安全にテストし、変更を検証し、安定性	DBA

タスク	説明	必要なスキル
	を確保します。これは、ライブ環境の中断を最小限に抑えながらデータベースの変更を実装するためのステージングの基盤として機能します。 グリーン環境を作成するには、まず Aurora MySQL 互換グローバルデータベースにプライマリ (ブルー) クラスターのスナップショットを作成します。このスナップショットは、グリーン環境を作成するための基盤として機能します。 スナップショットを作成するには： 1. にサインイン AWS Management Console し、Amazon Relational Database Service (Amazon RDS) コンソールを開きます。 2. プライマリ (青) クラスターを選択します。 3. アクション、スナップショットの作成を選択します。 4. などのスナップショットの名前を指定し`blue-green-demo` 、バックアッププロセスを開始します。	

タスク	説明	必要なスキル
	または、 AWS Command Line Interface （ AWS CLI) を使用してスナップショットを作成することもできます。 `aws rds create-db-cluster-snapshot --db-cluster-snapshot-identifier blue-green-demo --db-cluster-identifier ex-global-cluster --region eu-west-1` 次のステップに進む前に、スナップショットが正常に完了していることを確認してください。	

タスク	説明	必要なスキル
グローバルデータベースとそのリソースの CloudFormation テンプレートを生成します。	CloudFormation IaC ジェネレーターは、既存の AWS リソースから CloudFormation テンプレートを生成するのに役立ちます。この機能を使用して、既存の Aurora MySQL 互換グローバルデータベースとその関連リソースの CloudFormation テンプレートを作成します。このテンプレートは、サブネットグループ、セキュリティグループ、パラメータグループ、およびその他の設定を構成します。 1. CloudFormation ドキュメントの指示に従ってツールに移動し、 AWS 環境に接続します。 2. Aurora グローバルデータベースと関連するリソースを選択して、テンプレートを生成します。	DBA

タスク	説明	必要なスキル
グリーン環境の CloudFormation テンプレートを変更します。	CloudFormation テンプレートをカスタマイズして、グリーン環境の設定を反映します。これには、グリーン環境がブルークラスターとは独立して動作するようにリソース名と識別子を更新することが含まれます。 1. `DBClusterIdentifier` および `DBInstanceIdentifier` プロパティを更新して、グリーン環境を表します。 2. 他のリソース名 (サブネットグループやセキュリティグループなど) を変更して、既存のブルー環境との競合を回避します。 3. Aurora ドキュメントで説明されているように、正しいパラメータを設定して、テンプレートで GTID ベースのレプリケーションを有効にします。 4. `SnapshotIdentifier` プロパティを変更して AWS リージョン、前のステップの 、アカウント ID、スナップショットの名前を指定します。 `SnapshotIdentifier: arn:aws:rds:<region>:<account-id>:sn`	DBA

タスク	説明	必要なスキル
	apshot:<snapshot-name> **Note** SnapshotIdentifier プロパティを使用して DB クラスターを復元する場合は、GlobalClusterIdentifier 、、MasterUsername などのプロパティを指定しないでくださいMasterUserPassword 。	

タスク	説明	必要なスキル
CloudFormation スタックをデプロイして、グリーン環境のリソースを作成します。	このステップでは、カスタマイズされた CloudFormation テンプレートをデプロイして、グリーン環境のリソースを作成します。 CloudFormation スタックをデプロイするには： 1. AWS CloudFormation コンソールを開きます。 2. 右上で、スタックの作成、新しいリソース (標準) を選択します。 3. 変更した CloudFormation テンプレートをアップロードするか、テンプレート URL を指定します。[Next (次へ)] を選択します。 4. などのスタック名を入力しGreenClusterStack 、必要なパラメータ (など) を指定しますGreenClusterIdentifier 。[Next (次へ)] を選択します。 5. 必要に応じて追加のスタックオプションを設定し、CloudFormation が AWS Identity and Access Management (IAM) リソースを作成する場合があることを確認するチェックボッ	DBA

タスク	説明	必要なスキル
	クスをオンにします。[Next (次へ)] を選択します。 6. スタックの詳細を確認します。 7. [送信] を選択します。 CloudFormation は、グリーン環境リソースを作成するプロセスを開始します。このプロセスが完了するまでに数分かかる場合があります。	

タスク	説明	必要なスキル
CloudFormation スタックとリソースを検証します。	CloudFormation スタックのデプロイが完了したら、グリーン環境が正常に作成されたことを確認する必要があります。 1. CloudFormation スタックの出力セクションで、データベースクラスターとデータベースインスタンスのエンドポイントをチェックして、正しい設定を確認します。 2. Amazon RDS コンソールを開き、新しい Aurora データベースクラスター (グリーン環境) が使用可能であることを確認します。 3. サブネットやセキュリティグループなど、関連するすべてのリソースが作成され、グリーン環境にリンクされていることを確認します。 検証後、ブルー環境からのレプリケーションなど、グリーン環境をさらにセットアップする準備が整います。	DBA

GTID ベースのレプリケーションを設定する

タスク	説明	必要なスキル
ブルークラスターの GTID 設定を確認します。	GTIDs は、ブルー環境とグリーン環境間でデータをレプリケートするための信頼性の高い方法を提供します。GTID ベースのレプリケーションは、ブルー環境内のすべてのトランザクションに一意の識別子を割り当てることで、回復力と簡素化されたアプローチを提供します。この方法では、環境間のデータ同期がシームレスで一貫性があり、従来のバイナリログレプリケーションよりも管理が容易になります。 レプリケーションを設定する前に、ブルークラスターで GTID ベースのレプリケーションが適切に有効になっていることを確認する必要があります。このステップでは、ブルー環境の各トランザクションが一意に追跡され、グリーン環境でレプリケートできることを保証します。 GTID が有効になっていることを確認するには： 1. Amazon RDS コンソールで、ブルークラスターに割り当てられたパラメータグループを確認します。	DBA

タスク	説明	必要なスキル
	2. 次のパラメータが設定されていることを確認します。 • `gtid-mode = ON` • `enforce_gtid_consistency = ON` これらの設定により、ブルー環境での今後のすべてのトランザクションの GTID 追跡が有効になります。これらの設定を確認したら、レプリケーションの設定を開始できます。	

タスク	説明	必要なスキル
レプリケーションユーザーを作成します。	ブルー環境からグリーン環境にデータをレプリケートするには、ブルークラスターに専用のレプリケーションユーザーを作成する必要があります。このユーザーは、レプリケーションプロセスの管理を担当します。 レプリケーションユーザーを設定するには： 1. MySQL クライアントを使用してブルークラスターに接続します。 2. 次のコマンドを実行して、レプリケーションユーザーを作成します。 ``` CREATE USER 'repl_use r'@'%' IDENTIFIED BY 'repl_password'; GRANT REPLICATI ON SLAVE ON . TO 'repl_user'@'%'; FLUSH PRIVILEGES; ``` このユーザーには、2 つの環境間でデータをレプリケートするために必要なアクセス許可が付与されました。	DBA

タスク	説明	必要なスキル
グリーンクラスターで GTID ベースのレプリケーションを設定します。	次のステップでは、GTID ベースのレプリケーション用にグリーンクラスターを設定します。この設定により、グリーン環境はブルー環境で発生するすべてのトランザクションを継続的にミラーリングします。 グリーンクラスターを設定するには： 1. MySQL クライアントを使用してグリーンクラスターに接続します。 2. レプリケーションを設定するには、次のコマンドを実行します。 `CHANGE MASTER TO MASTER_HOST='blue-cluster-endpoint', MASTER_USER='repl_user', MASTER_PASSWORD='repl_password', MASTER_AUTO_POSITION=1;` 各パラメータの意味は次のとおりです。 • をブルークラスターのエンドポイント`blue-cluster-endpoint` に置き換えます。 • `MASTER_AUTO_POSITION=1` この設定は、GTI	DBA

タスク	説明	必要なスキル
	D ベースのレプリケーションを使用するように MySQL に指示します。ログと位置を手動で追跡することなく、ブルークラスターのトランザクションをレプリケートするように、グリーンクラスターを自動的に配置します。	
グリーンクラスターでレプリケーションを開始します。	これで、レプリケーションプロセスを開始できます。グリーンクラスターで、 コマンドを実行します。 `START SLAVE;` これにより、グリーン環境はデータの同期を開始し、ブルー環境からのトランザクションの受信と適用を開始できます。	DBA

タスク	説明	必要なスキル
レプリケーションプロセスを確認します。	グリーン環境がブルークラスターからデータを正確にレプリケートしていることを確認するには： 1. グリーンクラスターで次のコマンドを実行して、レプリケーションのステータスを確認します。 `SHOW SLAVE STATUS\G;` 2. 出力を確認して、以下を確認します。 • `Slave_IO_Running = Yes` • `Slave_SQL_Running = Yes` • `Retrieved_Gtid_Set` と `Executed_Gtid_Set` 値はup-to-dateで、ブルークラスターと同期されます。 • `Last_Error` フィールドにレプリケーションエラーはありません。 すべてのインジケータが正しい場合、GTID ベースのレプリケーションはスムーズに機能し、グリーン環境はブルー環境と完全に同期されます。	DBA

ブルークラスターからグリーンクラスターへのトラフィックの切り替え

タスク	説明	必要なスキル
Route 53 の加重ルーティングポリシーを設定します。	ブルー環境とグリーン環境間のデータ整合性を確認したら、ブルークラスターからグリーンクラスターにトラフィックを切り替えることができます。この移行はスムーズで、ダウンタイムを最小限に抑え、アプリケーションのデータベースの整合性を確保する必要があります。これらの要件を満たすには、DNS ルーティングに Route 53 を使用し、Lambda を使用してトラフィックの切り替えを自動化できます。さらに、明確に定義されたロールバックプランにより、問題が発生した場合にブルークラスターに戻すことができます。 最初のステップは、Route 53 で加重ルーティングを設定することです。加重ルーティングを使用すると、ブルークラスターとグリーンクラスター間のトラフィックの分散を制御し、一方の環境から他方の環境にトラフィックを徐々に移行できます。 加重ルーティングを設定するには：	AWS DevOps

タスク	説明	必要なスキル
	1. Route 53 コンソールを開き、ホストゾーンを選択します。 2. データベースに 2 つの DNS レコード (CNAMEs) を作成します。1 つはブルークラスター用、もう 1 つはグリーンクラスター用です。 3. 初期重みを割り当てる： • グリーンクラスターの初期ウェイトを低く設定 (5% など) して、テスト用のトラフィックの一部を送信します。 • ブルークラスターの重み (95% など) を大きく設定して、トラフィックの大部分を保持します。 この設定により、リスクを低減する段階的な移行を実行し、完全に切り替える前にリアルタイムのテストをサポートできます。 加重ルーティングポリシーの詳細については、Route 53 ドキュメントを参照してください。	

タスク	説明	必要なスキル

Lambda 関数をデプロイしてレプリケーションの遅延をモニタリングします。	グリーン環境がブルー環境と完全に同期されるようにするには、クラスター間のレプリケーションラグをモニタリングする Lambda 関数をデプロイします。この関数は、レプリケーションステータス、特に Seconds_Behind_Master メトリクスをチェックして、グリーンクラスターがすべてのトラフィックを処理する準備ができているかどうかを判断できます。
	
	使用できる Lambda 関数の例を次に示します。
	
	`import boto3`
	
	`def check_replication_lag(event, context):`
	`client = boto3.client('rds')`
	`response = client.describe_db_instances(DBInstanceIdentifier='green-cluster-instance')`
	`replication_status = response['DBInstances'][0]['ReadReplicaDBInstanceIdentifiers']`
	`if replication_status:`
	`lag = replication_status[0]['ReplicationLag']`
		AWS DevOps

タスク	説明	必要なスキル
	<pre> return lag return -1</pre> この関数はレプリケーションの遅延をチェックし、 値を返します。ラグがゼロの場合、グリーンクラスターはブルークラスターと完全に同期します。	

タスク	説明	必要なスキル

Lambda を使用して DNS の重み調整を自動化します。	レプリケーションの遅延がゼロになったら、すべてのトラフィックをグリーンクラスターに切り替えます。この移行を自動化するには、Route 53 の DNS 重みを調整してトラフィックの 100% をグリーンクラスターに誘導する別の Lambda 関数を使用します。
	
	トラフィックスイッチを自動化する Lambda 関数の例を次に示します。
	
	`import boto3`
	
	`def switch_traffic(eve nt, context):`
	`route53 = boto3.cli ent('route53')`
	`lag = check_rep lication_lag(event, context)`
	`if lag == 0:`
	`response = route53.change_res ource_record_sets(`
	`HostedZon eId='YOUR_HOSTED_Z ONE_ID',`
	`ChangeBat ch={`
	`'Changes': [`
	`{`
	`'Action': 'UPSERT',`
		AWS DevOps

タスク	説明	必要なスキル

	'ResourceRecordSet':
	{
	'Name': 'db.example.com',
	'Type': 'CNAME',
	'SetIdentifier': 'GreenCluster',
	'Weight': 100,
	'TTL': 60,
	'ResourceRecords': [{'Value': 'green-cluster-endpoint'}]
	}
	},
	{
	'Action': 'UPSERT',
	'ResourceRecordSet':
	{
	'Name': 'db.example.com',
	'Type': 'CNAME',
	'SetIdentifier': 'BlueCluster',
	'Weight': 0,
	'TTL': 60,
	'ResourceRecords':	

タスク	説明	必要なスキル

```
 [{'Value': 'blue-clu
ster-endpoint'}]

 }
                    }
                ]
            }
        )
        return response
```

この関数は、レプリケーションの遅延を確認し、遅延がゼロのときに Route 53 DNS の重みを更新して、トラフィックをグリーンクラスターに完全に切り替えます。

> **Note**
>
> カットオーバープロセス中にブルークラスターで大量の書き込みトラフィックが発生した場合は、カットオーバー中に一時的に書き込み操作を一時停止することを検討してください。これにより、レプリケーションが追いつき、ブルークラスターとグリーンクラスター間のデータの不整合を防ぐことができます。

タスク	説明	必要なスキル
トラフィックスイッチを確認します。	Lambda 関数が DNS の重みを調整したら、すべてのトラフィックがグリーンクラスターに転送され、スイッチが成功したことを確認する必要があります。 検証するには： 1. Route 53 DNS レコードをモニタリングして、トラフィックがグリーンクラスターに転送されていることを確認します。詳細については、Route 53 のドキュメントを参照してください。 2. ユーザーがグリーン環境から提供されていることを確認して、アプリケーションのパフォーマンスを確認します。 3. データベース接続を検証して、グリーンクラスターがすべてのデータベースリクエストを処理していることを確認します。 4. Amazon CloudWatch メトリクスをモニタリングして、レイテンシー、レプリケーションの遅延、パフォーマンスの低下の兆候がないかを確認します。詳細については、Aurora ドキュ	AWS DevOps

タスク	説明	必要なスキル
	メントを参照してください。 すべてが期待どおりに動作している場合、トラフィックスイッチは完了です。	

タスク	説明	必要なスキル
問題が発生した場合は、変更をロールバックします。	トラフィックの切り替え後に問題が発生した場合は、ロールバックプランを作成することが重要です。必要に応じてブルークラスターにすばやく戻る方法は次のとおりです。 1. Route 53 で DNS 重みを元に戻す: 同じ Lambda 関数を使用するか、Route 53 DNS 重みを手動で調整して、トラフィックの 100% をブルークラスターに戻します。 2. アプリケーションのパフォーマンスをモニタリングする: アプリケーションログ、CloudWatch メトリクス、データベースのパフォーマンスをすぐにモニタリングして、ブルー環境への切り替えが問題を解決したことを確認します。 3. 問題を特定して解決する: 別のトラフィック切り替えを試みる前に、グリーンクラスターに関する問題を調査して対処します。 このロールバックプランを実装することで、予期しない問題が発生した場合にユーザーの中断を最小限に抑えることができます。	AWS DevOps

GTID ベースのレプリケーションを検証して停止する

タスク	説明	必要なスキル
グリーンクラスターで GTID ベースのレプリケーションを停止します。	ブルー環境からグリーン環境にトラフィックを切り替えたら、移行の成功を検証し、グリーンクラスターが期待どおりに機能していることを確認します。さらに、グリーン環境がプライマリデータベースとして機能するようになったため、ブルークラスターとグリーンクラスター間の GTID ベースのレプリケーションを停止する必要があります。これらのタスクを完了すると、環境が安全で合理化され、継続的な運用用に最適化されます。 レプリケーションを停止するには： 1. MySQL クライアントを使用してグリーンクラスターに接続します。 2. 次の SQL コマンドを実行して、グリーンクラスターでレプリケーションプロセスを停止します。 `STOP SLAVE;` 3. (オプション) 必要に応じて、レプリケーション設定をリセットして、残ってい	DBA

タスク	説明	必要なスキル
	るレプリケーション設定をクリアできます。 `RESET SLAVE ALL;` レプリケーションを停止すると、グリーンクラスターは完全に独立し、ワークロードのプライマリデータベース環境として機能します。	

タスク	説明	必要なスキル
リソースをクリーンアップします。	ブルークラスターからグリーンクラスターへの移行中に作成された一時的または未使用のリソースをクリーンアップすることで、環境が最適化、安全性、コスト効率を維持できます。クリーンアップには、セキュリティ設定の調整、最終バックアップの取得、不要なリソースの廃止が含まれます。 リソースをクリーンアップするには： 1. セキュリティグループを更新する: 新しいプライマリ環境 (緑) を反映するように、ブルークラスターとグリーンクラスターの両方に関連付けられているセキュリティグループを設定します。不要になった場合はブルー環境へのアクセスを制限し、グリーンクラスターのセキュリティ設定がベストプラクティスに従っていることを確認します。 2. グリーンクラスターの最終バックアップを作成する: 移行が完了したら、グリーンクラスターの最終スナップショットを作成してバックアップとして機能します。このスナップショット	AWS DevOps

タスク	説明	必要なスキル

	を使用して、必要に応じて今後環境を復元できます。
	
	`aws rds create-db-snapshot --db-instance-identifier green-cluster-instance --db-snapshot-identifier green-cluster-final-snapshot`
	
	3. 一時的なリソースの確認と削除: 一時的なセキュリティグループ、スナップショット、その他の設定など、移行中に作成された一時的なリソースを確認します。不要なコストを防ぐために不要になったリソースを削除します。たとえば、不要になったブルークラスターを削除します。
	
	`aws rds delete-db-cluster --db-cluster-identifier blue-cluster-identifier --skip-final-snapshot`
	
	リソースをクリーンアップすることで、安全で合理化された環境を維持し、コストを削減し、必要なインフラストラクチャだけが残るようにします。	

関連リソース

AWS CloudFormation:

- AWS CloudFormation ユーザーガイド
- AWS CloudFormation ベストプラクティス
- IaC Generator を使用して既存のリソースからテンプレートを生成する
- アプリケーション全体を にインポート AWS CloudFormationする (AWS ブログ記事)

Amazon Aurora:

- Amazon Aurora ユーザーガイド
- Amazon Aurora DB クラスターの管理

ブルー/グリーンデプロイ戦略：

- Amazon Aurora ブルー/グリーンデプロイの概要

GTID ベースのレプリケーション：

- GTID ベースのレプリケーションの使用 (Amazon RDS ドキュメント)

AWS Lambda:

- AWS Lambda デベロッパーガイド
- AWS Lambda 関数を使用するためのベストプラクティス

Amazon Route 53:

- Amazon Route 53 デベロッパーガイド
- 加重ルーティング

MySQL クライアントツール：

- PyMYSQL

AWS Lambda および Task Scheduler を使用して、Amazon EC2 で実行されている SQL Server Express Edition のデータベースタスクを自動化する

作成者: Subhani Shaik (AWS)

概要

このパターンは、SQL Server の無料バージョンである SQL Server Express Edition でデータベースタスクをスケジュールおよび管理する方法を示しています。ただし、SQL Server Express エディションには、通常、自動データベースオペレーションを処理する SQL Server エージェントサービスがありません。このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されている SQL Server Express エディションでデータベースタスクを自動化する代わりに、Task Scheduler と Lambda を使用する方法について説明します。

Task Scheduler は、ルーチンタスクの自動実行を容易にする組み込みの Windows システムユーティリティです。自動オペレーションをスケジュールおよび管理するためのメカニズムを提供するため、定期的なプロセスに手動で介入する必要はありません。 AWS Lambdaは、基盤となるインフラストラクチャを管理することなく、イベントに応じてコードを自動的に実行するサーバーレスコンピューティングサービスです。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon Virtual Private Cloud (Amazon VPC) で作成された Virtual Private Cloud (VPC)
- Windows Server を使用する Amazon EC2 インスタンス
- Windows Server で Amazon EC2 インスタンスにアタッチされた Amazon Elastic Block Store (Amazon EBS) ボリューム
- SQL Server Express Edition バイナリ

制約事項

- SQL Server Express Edition の機能制限については、Microsoft のウェブサイトを参照してください。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポ

イントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- SQL Server Express Edition を使用した SQL Server 2016 以降

アーキテクチャ

次の図は、SQL Server Express Edition がインストールされた状態で実行されている Amazon EC2 インスタンスを示しています。インスタンスは、リモートデスクトッププロトコル (RDP) クライアントまたは からアクセスできます AWS Systems Manager Session Manager。 AWS Key Management Service （ AWS KMS) は Amazon EBS ボリュームのデータ暗号化を処理し、data-at-restセキュリティを確保します。インフラストラクチャには AWS Identity and Access Management 、(IAM) も含まれています。これは、Lambda 関数の実行のためのアクセスコントロールを提供し、アクセス許可を管理します。Amazon Simple Storage Service (Amazon S3) は Lambda 関数を保存します。

ツール

AWS のサービス

- Amazon Elastic Block Store (Amazon EBS) は、Amazon EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を許可するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Systems Manager Session Manager はフルマネージド AWS Systems Manager 型のツールです。Session Manager を使用すると、Amazon EC2 インスタンス、エッジデバイス、オンプレミスサーバー、仮想マシン (VMsを管理できます。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

その他のツール

- 「Microsoft SQL Server Management Studio (SSMS)」 は、SQL Server コンポーネントへのアクセス、設定、管理など、SQL Server を管理するためのツールです。
- 「Python」は汎用のコンピュータープログラミング言語です。これを使用して、 でアプリケーションの構築、タスクの自動化、サービスの開発を行うことができますAWS クラウド。
- Task Scheduler は、コンピュータでルーチンタスクを自動的にスケジュールするために使用できる Microsoft ツールです。

ベストプラクティス

- Amazon EC2 のベストプラクティス
- Amazon EC2 に Microsoft SQL Server をデプロイするためのベストプラクティス
- AWS Lambda 関数を使用するためのベストプラクティス
- IAM でのセキュリティのベストプラクティス

エピック

Amazon EC2 インスタンスを作成し、SQL Server Express Edition をインストールする

タスク	説明	必要なスキル
Amazon EC2 インスタンスをデプロイします。	Amazon EC2 インスタンスを作成するには、https://https://console.aws.amazon.com/	DBA、AWS DevOps

タスク	説明	必要なスキル
	ec2/.com で Amazon EC2 コンソールを開き、Windows Server で使用できるインスタンスのリストから Amazon マシンイメージ (AMI) を選択します。 詳細については、 AWS ドキュメントのAmazon EC2 インスタンスを起動する」を参照してください。	

タスク	説明	必要なスキル
SQL Server Express エディションをインストールします。	SQL Server Express Edition をインストールするには、次の手順を実行します。 1. Amazon EC2 インスタンスに接続するには、オプションを選択します。 • オプション A – リモートデスクトッププロトコル (RDP) を使用します。手順については、 AWS ドキュメントの「RDP クライアントを使用して Windows インスタンスに接続する」を参照してください。 • オプション B – Amazon EC2 コンソールと を使用します AWS Systems Manager Session Manager。手順については、 AWS ドキュメントの「Session Manager を使用して Amazon EC2 インスタンスに接続する」を参照してください。 2. 必要な SQL Server Express エディションをダウンロードするには、Microsoft ウェブサイトの SQL Server Downloads にアクセスしてください。 3. SQL Server Express Edition をインストールするに	DBA、AWS DevOps

タスク	説明	必要なスキル
	は、Microsoft ウェブサイトの「SQL Server のインストールを計画する」の手順に従います。	

自動データベースメンテナンスタスクを作成する

タスク	説明	必要なスキル
ルーチンタスクを特定します。	自動化するルーチンタスクを特定します。たとえば、次のタスクは自動化の対象となります。 • データベースバックアップ (完全、差分、トランザクションログ) • インデックスのメンテナンスと再編成 • 統計の更新 • アプリケーション固有のオペレーション • データクリーンアップまたはアーカイブ	DBA
SQL スクリプトを準備します。	SQL スクリプトを準備するには、以下を実行します。 1. メンテナンスタスクごとに SQL クエリを作成します。特定のデータベースバックアップを実行する T-SQL クエリの例を次に示します。 `Backup Database`	DBA

タスク	説明	必要なスキル
	<Database_Name> To Disk='C:\Backups\Database_Name.bak' 2. スクリプトファイルを として保存します<File Name>.sql 。次に、Amazon EC2 インスタンスまたはネットワークファイル共有上のサーバーのローカルドライブ上のアクセス可能な場所にスクリプトを保存します。	

タスク	説明	必要なスキル
アクセス許可を設定します。	アクセス許可を設定するには、以下を実行します。 1. 適切なファイルシステムのアクセス許可を設定します。手順については、Microsoft ウェブサイトの「データベースエンジンアクセスのファイルシステムアクセス許可を設定する」を参照してください。 2. SQL Server サービスアカウントに必要なアクセス権があることを確認します。手順については、Microsoft ウェブサイトの「Windows サービスアカウントとアクセス許可を設定する」を参照してください。 3. リモート共有のネットワーク接続を確認します。詳細については、 AWS ドキュメントの「ファイル共有を使用したデータへのアクセス」を参照してください。	DBA

Task Scheduler を使用してタスクを自動化する

タスク	説明	必要なスキル
バッチファイルを作成します。	• バッチファイルを作成するには、テキストエディタを使用して次のコマンドを入力します。パラメー	AWS DevOps、DBA

タスク	説明	必要なスキル

	タ usernameと を独自の値passwordに置き換えます。次に、ファイルを として保存します<Name>.bat 。
	
	`sqlcmd -S servername -U username -P password -i <T-SQL query path.sql>`
	
	• SQL タスク用のバッチファイルを作成するには、テキストエディタを使用して、次のコマンドを入力します。パラメータ ServerName 、DatabaseName 、username、 を独自の値passwordに置き換えます。次に、ファイルを として保存します<Name>.bat 。
	
	`@echo off`
	`sqlcmd -S [ServerName] -d [DatabaseName] -U username -P password -i "PathToSQLScript\Script.sql" -o "PathToOutput\Output.txt"`	

タスク	説明	必要なスキル
Task Scheduler でタスクを作成します。	Task Scheduler でタスクを作成するには、次のステップを使用します。 1. Task Scheduler を開くには、Windows 検索で taskschd.msc と入力します。 2. アクションメニューを選択し、基本タスクの作成を選択します。 3. 名前 にタスクの名前を指定し、次へ を選択します。 4. Trigger で、タスクを開始する のオプションを選択します。次に次へを選択します。 5. タスクの開始と再帰の情報を入力し、次へを選択します。 6. アクションセクションで、プログラムの開始を選択し、次へを選択します。 7. プログラム/スクリプトで、前のタスクで作成したバッチファイルのパスを指定し、次へを選択します。 8. [Finish] を選択してください。 タスクを手動で実行するには、新しく作成したタスクを	DBA

タスク	説明	必要なスキル
	右クリックし、実行を選択します。	
タスクのステータスを表示します。	Task Scheduler でタスクのステータスを表示するには、次のステップを使用します。 1. Task Scheduler で、Task Scheduler Library に移動し、すべてのタスクを表示します。 2. 前に作成したタスクのステータスを確認するには、タスクを選択し、履歴タブに移動します。	DBA、AWS DevOps

でタスクを自動化する AWS Lambda

タスク	説明	必要なスキル
ソリューションを実装します。	このパターンのソリューションを実装するには、次のステップを使用します。 1. Lambda 関数を作成する。手順については、 AWS ドキュメントの「最初の Lambda 関数を作成する」を参照してください。 2. Lambda 関数をスケジュールします。手順については、 AWS ドキュメントの「スケジュールに従って Lambda 関数を呼び出す」を参照してください。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	3. T-SQL クエリを実行します。詳細については、 AWS ドキュメントの「チュートリアル: Lambda 関数を使用して Amazon RDS データベースにアクセスする」を参照してください。このチュートリアルでは、Lambda 関数から Amazon RDS データベースを接続して SQL クエリを実行する方法について説明します。	

トラブルシューティング

問題	ソリューション
Lambda の問題	の使用時に発生する可能性のあるエラーや問題のヘルプについては AWS Lambda、 AWS ドキュメントの「Lambda の問題のトラブルシューティング」を参照してください。

関連リソース

- Amazon EC2 インスタンスタイプ
- AWS Lambda ドキュメント
- AWS Lambda 料金
- デベロッパー向けタスクスケジューラ (Microsoft ウェブサイト)

DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する

作成者: Jitendra Kumar (AWS)、Oliver Francis (AWS)、Pavithra Balasubramanian (AWS)

概要

このパターンでは、DR Orchestrator Framework を使用して、エラーが発生しやすい手動ステップを調整して自動化し、Amazon Web Services (AWS) リージョン全体でディザスタリカバリを実行する方法について説明します。このパターンは、次のデータベースを対象としています。

- Amazon Relational Database Service (Amazon RDS) for MySQL、Amazon RDS for PostgreSQL、または Amazon RDS for MariaDB
- Amazon Aurora MySQL 互換エディションまたは Amazon Aurora PostgreSQL 互換エディション (一元化されたファイルを使用)
- Amazon ElastiCache (Redis OSS)

DR Orchestrator Framework の機能を実証するには、2 つの DB インスタンスまたはクラスターを作成します。プライマリは にあり AWS リージョン `us-east-1`、セカンダリは にありますus-west-2。これらのリソースを作成するには、aws-cross-region-dr-databases GitHub リポジトリの App-Stackフォルダにある AWS CloudFormation テンプレートを使用します。

前提条件と制限

一般的な前提条件

- プライマリとセカンダリの両方にデプロイされた DR オーケストレーターフレームワーク AWS リージョン
- 2 つの Amazon Simple Storage Service バケット
- 2 つのサブネットと AWS セキュリティグループを持つ Virtual Private Cloud (VPC)

エンジン固有の前提条件

- Amazon Aurora – 少なくとも 1 つの Aurora グローバルデータベースを 2 つで利用できる必要があります AWS リージョン。をプライマリリージョンus-east-1として使用し、 をセカンダリリージョンus-west-2として使用できます。

- Amazon ElastiCache (Redis OSS) – ElastiCache グローバルデータストアは 2 つで利用できる必要があります AWS リージョン。をプライマリリージョンuse `us-east-1`として、 をセカンダリリージョン`us-west-2`として使用できます。

Amazon RDS の制限事項

- DR Orchestrator Framework は、フェイルオーバーまたはフェイルバックを実行する前にレプリケーションの遅延をチェックしません。レプリケーションの遅延は手動でチェックする必要があります。
- このソリューションは、1 つのリードレプリカを持つプライマリデータベースインスタンスを使用してテストされています。複数のリードレプリカを使用する場合は、本番環境に実装する前に、ソリューションを徹底的にテストしてください。

Aurora の制限

- 機能の可用性とサポートは、各データベースエンジンの特定のバージョンと全体で異なります AWS リージョン。クロスリージョンレプリケーションの機能とリージョンの可用性の詳細については、「クロスリージョンリードレプリカ」を参照してください。
- Aurora グローバルデータベースには、サポートされている Aurora DB インスタンスクラスと の最大数に関する特定の設定要件があります AWS リージョン。詳細については、「Amazon Aurora グローバルデータベースの設定要件」を参照してください。
- このソリューションは、1 つのリードレプリカを持つプライマリデータベースインスタンスを使用してテストされています。複数のリードレプリカを使用する場合は、本番環境に実装する前に、ソリューションを徹底的にテストしてください。

ElastiCache の制限事項

- Global Datastore と ElastiCache の設定要件のリージョンの可用性については、ElastiCache ドキュメントの「前提条件と制限」を参照してください。

Amazon RDS product バージョン

Amazon RDS は、次のエンジンバージョンをサポートしています。

- MySQL – Amazon RDS は、MySQL 8.0 および MySQL 5.7 のバージョンの MySQL を実行する DB インスタンスをサポートします。

- PostgreSQL – サポートされているバージョンの Amazon RDS for PostgreSQL については、「利用可能な PostgreSQL データベースバージョン」を参照してください。
- MariaDB – Amazon RDS は、次のバージョンの MariaDB を実行する DB インスタンスをサポートしています。
  - MariaDB 10.11
  - MariaDB 10.6
  - MariaDB 10.5

Aurora 製品バージョン

- Amazon Aurora グローバルデータベースのスイッチオーバーには、Aurora MySQL と MySQL 5.7 との互換性、バージョン 2.09.1 以降が必要です

  詳細については、「Amazon Aurora グローバルデータベースの制限」を参照してください。

ElastiCache (Redis OSS) 製品バージョン

Amazon ElastiCache (Redis OSS) では、次の Redis バージョンがサポートされています。

- Redis 7.1 (拡張)
- Redis 7.0 (拡張)
- Redis 6.2 (拡張)
- Redis 6.0 (拡張)
- Redis 5.0.6 (拡張)

詳細については、「サポートされている ElastiCache (Redis OSS) バージョン」を参照してください。

アーキテクチャ

Amazon RDS アーキテクチャ

Amazon RDS アーキテクチャには、次のリソースが含まれています。

- クライアントの読み取り/書き込みアクセス権を持つプライマリリージョン (`us-east-1`) で作成されたプライマリ Amazon RDS DB インスタンス

- クライアントの読み取り専用アクセスを持つセカンダリリージョン (us-west-2) で作成された Amazon RDS リードレプリカ
- プライマリリージョンとセカンダリリージョンの両方にデプロイされた DR オーケストレーターフレームワーク

図に示す内容は以下のとおりです。

1. プライマリインスタンスとセカンダリインスタンス間の非同期レプリケーション
2. プライマリリージョンのクライアントの読み取り/書き込みアクセス
3. セカンダリリージョンのクライアントの読み取り専用アクセス

Aurora アーキテクチャ

Amazon Aurora アーキテクチャには、次のリソースが含まれています。

- アクティブライターエンドポイントを使用してプライマリリージョン (us-east-1) で作成されたプライマリ Aurora DB クラスター
- 非アクティブライターエンドポイントを持つセカンダリリージョン (us-west-2) で作成された Aurora DB クラスター
- プライマリリージョンとセカンダリリージョンの両方にデプロイされた DR オーケストレーターフレームワーク

図に示す内容は以下のとおりです。

1. プライマリクラスターとセカンダリクラスター間の非同期レプリケーション
2. アクティブライターエンドポイントを持つプライマリ DB クラスター
3. 非アクティブライターエンドポイントを持つセカンダリ DB クラスター

ElastiCache (Redis OSS) アーキテクチャ

Amazon ElastiCache (Redis OSS) アーキテクチャには、次のリソースが含まれています。

- 2 つのクラスターで作成された ElastiCache (Redis OSS) グローバルデータストア：

1. プライマリリージョンのプライマリクラスター (us-east-1)
2. セカンダリリージョンのセカンダリクラスター (us-west-2)

- 2 つのクラスター間の TLS 1.2 暗号化を使用した Amazon クロスリージョンリンク
- プライマリリージョンとセカンダリリージョンの両方にデプロイされた DR オーケストレーターフレームワーク

自動化とスケール

DR Orchestrator Framework はスケーラブルで、複数の AWS データベースのフェイルオーバーまたはフェイルバックを並行してサポートします。

次のペイロードコードを使用して、アカウント内の複数の AWS データベースをフェイルオーバーできます。この例では、3 つの AWS データベース (Aurora MySQL 互換または Aurora PostgreSQL 互換などの 2 つのグローバルデータベース、および 1 つの Amazon RDS for MySQL インスタンス) が DR リージョンにフェイルオーバーします。

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
 databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-mysql-global-
identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-
identifier"
          }
        },
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
 databases (PostgreSQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-postgres-global-
identifier",
```

```
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-postgres-cluster-
identifier"
          }
        },
        {
          "resourceType": "PromoteRDSReadReplica",
          "resourceName": "Promote RDS for MySQL Read Replica",
          "parameters": {
            "RDSInstanceIdentifier": "!Import rds-mysql-instance-identifier",
            "TargetClusterIdentifier": "!Import rds-mysql-instance-global-arn"
          }
        }
      ]
    }
  ]
}
```

ツール

AWS サービス

- 「Amazon Aurora」はクラウド用に構築されたフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- Amazon ElastiCache は、 で分散インメモリキャッシュ環境をセットアップ、管理、スケーリングするのに役立ちます AWS クラウド。このパターンでは、Amazon ElastiCache (Redis OSS) を使用します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。このパターンでは、Lambda 関数は AWS Step Functions によってステップの実行に使用されます。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。このパターンは、Amazon RDS for MySQL、Amazon RDS for PostgreSQL、および Amazon RDS for MariaDB をサポートしています。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立ちます AWS のサービス。このパターンでは、Boto3 APIs はデータベースインスタンスまたはグローバルデータベースとの通信に使用されます。
- AWS Step Functions はサーバーレスオーケストレーションサービスで、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するの

に役立ちます。このパターンでは、Step Functions ステートマシンを使用して、データベースインスタンスまたはグローバルデータベースのクロスリージョンフェイルオーバーとフェイルバックをオーケストレーションして実行します。

コードリポジトリ

このパターンのコードは、GitHub の aws-cross-region-dr-databases リポジトリで入手できます。

エピック

DR オーケストレーターフレームワークをインストールする

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	リポジトリのクローンを作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/aws-cross-region-dr-databases.git`	AWS DevOps、AWS 管理者

Lambda 関数コードを .zip ファイルアーカイブにパッケージ化します。	Lambda 関数のアーカイブファイルを作成して、DR Orchestrator Framework の依存関係を含めます。
	`cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts`
	
	`bash scripts/deploy-orchestrator-sh.sh`
		AWS 管理者

S3 バケットを作成します。	DR Orchestrator Framework を最新の設定とともに保存するには、S3 バケットが必要です。2 つの S3 バケットを作	AWS 管理者

タスク	説明	必要なスキル
	成します。1 つはプライマリリージョン (us-east-1) 、もう 1 つはセカンダリリージョン () ですus-west-2 。 • dr-orchestrator-xxxxxx-us-east-1 • dr-orchestrator-xxxxxx-us-west-2 をランダムな値xxxxxxに置き換えて、バケット名を一意にします。	
サブネットとセキュリティグループを作成します。	プライマリリージョン (us-east-1) とセカンダリリージョン () の両方でus-west-2 、VPC に Lambda 関数をデプロイするための 2 つのサブネットと 1 つのセキュリティグループを作成します。 • subnet-XXXXXXX • subnet-YYYYYYY • sg-XXXXXXXXXXXX	AWS 管理者

タスク	説明	必要なスキル
DR Orchestrator パラメータファイルを更新します。	<YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation フォルダで、次の DR Orchestrator パラメータファイルを更新します。 • Orchestrator-Deployer-parameters-us-east-1.json • Orchestrator-Deployer-parameters-us-west-2.json 次のパラメータ値を使用して、 xと をリソースの名前yに置き換えます。 [{ "ParameterKey": "TemplateStoreS3BucketName", "ParameterValue": "dr-orchestrator-xxxxxx-us-east-1" }, { "ParameterKey": "TemplateVPCId", "ParameterValue": "vpc-xxxxxx" }, {	AWS 管理者

タスク	説明	必要なスキル
	"ParameterKey": "TemplateLambdaSubnetID1", "ParameterValue": "subnet-xxxxx" }, { "ParameterKey": "TemplateLambdaSubnetID2", "ParameterValue": "subnet-yyyyy" }, { "ParameterKey": "TemplateLambdaSecurityGroupID", "ParameterValue": "sg-xxxxxxxxxx" }]	

タスク	説明	必要なスキル
DR Orchestrator Framework コードを S3 バケットにアップロードします。	このコードは、ローカルディレクトリよりも S3 バケットの方が安全です。すべてのファイルとサブフォルダを含むDR-Orchestration-artifacts ディレクトリを S3 バケットにアップロードします。 コードをアップロードするには、次の手順を実行します。 1. AWS Management Console にサインインします。 2. Amazon S3 コンソールに移動します。 3. dr-orchestrator-xxxxxx-us-east-1 bucket を選択します。 4. アップロードを選択し、フォルダの追加を選択します。 5. DR-Orchestration-artifacts フォルダを選択します。 6. [アップロード] を選択します。 7. dr-orchestrator-xxxxxx-us-west-2 バケットを選択します。 8. ステップ 4 ~ 7 を繰り返します。	AWS 管理者

タスク	説明	必要なスキル

DR オーケストレーターフレームワークをプライマリリージョンにデプロイします。	DR Orchestrator Framework をプライマリリージョン (us-east-1) にデプロイするには、次のコマンドを実行します。
	`cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
	`--stack-name dr-orchestrator \`
	`--template-file Orchestrator-Deployer.yaml \`
	`--parameter-overrides file://Orchestrator-Deployer-parameters-us-east-1.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
		AWS 管理者

タスク	説明	必要なスキル
DR オーケストレーターフレームワークをセカンダリリージョンにデプロイします。	セカンダリリージョン (us-west-2) で、次のコマンドを実行します。 ``` cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation aws cloudformation deploy \ --region us-west-2 \ --stack-name dr-orchestrator \ --template-file Orchestrator-Deployer.yaml \ --parameter-overrides file://Orchestrator-Deployer-parameters-us-west-2.json \ --capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \ --disable-rollback ```	AWS 管理者

タスク	説明	必要なスキル
デプロイメントを確認する	AWS CloudFormation コマンドが正常に実行されると、次の出力が返されます。 `Successfully created/ updated stack - dr-orchestrator` または、 AWS CloudFormation コンソールに移動し、dr-orchestrator スタックのステータスを確認できます。	AWS 管理者

データベースインスタンスまたはクラスターを作成する

タスク	説明	必要なスキル
データベースサブネットとセキュリティグループを作成します。	VPC で、プライマリ (us-east-1) リージョンとセカンダリ (us-west-2) リージョンの両方に、DB インスタンスまたはグローバルデータベース用に 2 つのサブネットと 1 つのセキュリティグループを作成します。 • subnet-XXXXXX • subnet-XXXXXX • sg-XXXXXXXXXX	AWS 管理者
プライマリ DB インスタンスまたはクラスターのパラメータファイルを更新します。	<YOUR LOCAL GIT FOLDER>/App-Stack フォルダで、プライマリリージョ	AWS 管理者

タスク	説明	必要なスキル
	ンのパラメータファイルを更新します。 Amazon RDS RDS-MySQL-parameter-us-east-1.json ファイルで、作成したリソースの名前DBSecurityGroup で SubnetIds と を更新します。 `{ "Parameters": { "SubnetIds": "subnet-xxxxxx,subnet-xxxxxx", "DBSecurityGroup": "sg-xxxxxxxxxx", "MySqlGlobalIdentifier":"rds-mysql-instance", "InitialDatabaseName": "mysqldb", "DBPortNumber": "3789", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/rds-mysql-instance-KmsKeyId" } }` Amazon Aurora	

タスク	説明	必要なスキル
	Aurora-MySQL-parameter-us-east-1.json ファイルで、作成したリソースの名前DBSecurityGroup で SubnetIds とを更新します。 `{ "Parameters": { "SubnetIds": "subnet1-xxxxxx,subnet2-xxxxxx", "DBSecurityGroup": "sg-xxxxxxxxxx", "GlobalClusterIdentifier":"dr-globaldb-cluster-mysql", "DBClusterName":"dbcluster-01", "SourceDBClusterName":"dbcluster-02", "DBPortNumber": "3787", "DBInstanceClass": "db.r5.large", "InitialDatabaseName": "sampledb", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/dr-globaldb-cluster-mysql-KmsKeyId" } }` Amazon ElastiCache (Redis OSS)	

タスク	説明	必要なスキル

	ElastiCache-parameter-us-east-1.json ファイルで、作成したリソースの名前DBSecurityGroup で SubnetIds とを更新します。
	
	{
	"Parameters": {
	"CacheNodeType": "cache.m5.large",
	"DBSecurityGroup": "sg-xxxxxxxxxx",
	"SubnetIds": "subnet-xxxxxx,subnet-xxxxxx",
	"EngineVersion": "5.0.6",
	"GlobalReplicationGroupIdSuffix": "demo-redis-global-datastore",
	"NumReplicas": "1",
	"NumShards": "1",
	"ReplicationGroupId": "demo-redis-cluster",
	"DBPortNumber": "3788",
	"TransitEncryption": "true",
	"KMSKeyAliasName": "elasticache/demo-redis-global-datastore-KmsKeyId",
	"PrimaryRegion": "us-east-1",
	"SecondaryRegion": "us-west-2"
	}	

タスク	説明	必要なスキル
	<pre>}</pre>	

タスク	説明	必要なスキル

DB インスタンスまたはクラスターをプライマリリージョンにデプロイします。	インスタンスまたはクラスターをプライマリリージョン (us-east-1) にデプロイするには、データベースエンジンに基づいて次のコマンドを実行します。
	
	Amazon RDS
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
	`--stack-name rds-mysql-app-stack \`
	`--template-file RDS-MySQL-Primary.yaml \`
	`--parameter-overrides file://RDS-MySQL-parameter-us-east-1.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
	
	Amazon Aurora
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
		AWS 管理者

タスク	説明	必要なスキル
	``` --stack-name aurora-my sql-app-stack \ --template-file Aurora- MySQL-Primary.yaml \ --parameter-overrides file://Aurora-MySQ L-parameter-us-eas t-1.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ``` Amazon ElastiCache (Redis OSS) ``` cd <YOUR-LOCAL-GIT-FO LDER>/App-Stack aws cloudformation deploy \ --region us-east-1 -- stack-name elasticac he-ds-app-stack \ --template-file ElastiCache-Primar y.yaml \ --parameter-overrides file://ElastiCache -parameter-us-east -1.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ```	

タスク	説明	必要なスキル
	AWS CloudFormation リソースが正常にデプロイされたことを確認します。	

タスク	説明	必要なスキル
セカンダリ DB インスタンスまたはクラスターのパラメータファイルを更新します。	<YOUR LOCAL GIT FOLDER>/App-Stack フォルダで、セカンダリリージョンのパラメータファイルを更新します。 Amazon RDS RDS-MySQL-parameter-us-west-2.json ファイルで、作成したリソースの名前DBSecurityGroup で SubnetIDs と を更新します。を、プライマリ DB インスタンスの AWS CloudFormation スタックの出力セクションからMySQLKmsKeyId 取得した の値PrimaryRegionKMSKeyArn で更新します。 `{ "Parameters": { "SubnetIds": "subnet-aaaaaaaaa, subnet-bbbbbbbbb", "DBSecurityGroup": "sg-ccccccccc", "MySqlGlobalIdentifier":"rds-mysql-instance", "InitialDatabaseName": "mysqldb", "DBPortNumber": "3789", "PrimaryRegion": "us-east-1",`	AWS 管理者

タスク	説明	必要なスキル
	"SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/rds-mysql-instance-KmsKeyId", "PrimaryRegionKMSKeyArn":"arn:aws:kms:us-east-1:xxxxxxxxx:key/mrk-xxxxxxxxxxxxxxxxx" } } Amazon Aurora Aurora-MySQL-parameter-us-west-2.json ファイルで、作成したリソースの名前DBSecurityGroup で SubnetIDs と を更新します。を、プライマリ DB インスタンスの AWS CloudFormation スタックの出力セクションからAuroraKmsKeyId 取得した の値PrimaryRegionKMSKeyArn で更新します。 { "Parameters": { "SubnetIds": "subnet1-aaaaaaaaa,subnet2-bbbbbbbbb", "DBSecurityGroup": "sg-ccccccccc", "GlobalClusterIdentifier":"dr-globaldb-cluster-mysql",	

タスク	説明	必要なスキル
	<pre>"DBClusterName":"d bcluster-01", "SourceDBClusterNa me":"dbcluster-02", "DBPortNumber": "3787", "DBInstanceClass": "db.r5.large", "InitialDatabaseNa me": "sampledb", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/dr-globaldb-c luster-mysql-KmsKe yId" } }</pre> Amazon ElastiCache (Redis OSS) `ElastiCache-parame ter-us-west-2.json` ファイルで、作成したリソースの名前`DBSecurit yGroup` で `SubnetIDs` とを更新します。を、プライマリ DB インスタンスの AWS CloudFormation スタックの出力セクションから`ElastiCac heKmsKeyId` 取得した の値`PrimaryRegionKMSKe yArn` で更新します。 <pre>{ "Parameters": {	

タスク	説明	必要なスキル
	"CacheNodeType": "cache.m5.large", "DBSecurityGroup": "sg-ccccccccc", "SubnetIds": "subnet-aaaaaaaaa, subnet-bbbbbbbbb", "EngineVersion": "5.0.6", "GlobalReplicationGroupIdSuffix": "demo-redis-global-datastore", "NumReplicas": "1", "NumShards": "1", "ReplicationGroupId": "demo-redis-cluster", "DBPortNumber": "3788", "TransitEncryption": "true", "KMSKeyAliasName": "elasticache/demo-redis-global-datastore-KmsKeyId", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2" } }	

タスク	説明	必要なスキル

DB インスタンスまたはクラスターをセカンダリリージョンにデプロイします。	データベースエンジンに基づいて、次のコマンドを実行します。
	
	Amazon RDS
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-west-2 \`
	`--stack-name rds-mysql-app-stack \`
	`--template-file RDS-MySQL-DR.yaml \`
	`--parameter-overrides file://RDS-MySQL-parameter-us-west-2.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
	
	Amazon Aurora
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-west-2 \`
	`--stack-name aurora-mysql-app-stack \`
	`--template-file Aurora-MySQL-DR.yaml \`
		AWS 管理者

タスク	説明	必要なスキル
	--parameter-overrides file://Aurora-MySQ L-parameter-us-wes t-2.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback Amazon ElastiCache (Redis OSS) cd <YOUR-LOCAL-GIT-FO LDER>/App-Stack aws cloudformation deploy \ --region us-west-2 \ --stack-name elasticac he-ds-app-stack \ --template-file ElastiCache-DR.yaml \ --parameter-overrides file://ElastiCache -parameter-us-west -2.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback AWS CloudFormation リソースが正常にデプロイされたことを確認します。	

関連リソース

- のデータベースのディザスタリカバリ戦略 AWS (AWS 規範ガイダンス戦略)
- でリレーショナルデータベースの DR ソリューションを自動化する AWS (AWS 規範ガイダンスガイド)
- Amazon Aurora Global Database の使用
- グローバルデータストア AWS リージョン を使用した 間のレプリケーション
- でリレーショナルデータベースの DR ソリューションを自動化する AWS (AWS 規範ガイダンスガイド)

全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント

作成者: Parag Nagwekar (AWS)、Arun Chandapillai (AWS)

概要

このパターンは、 AWS Step Functions と AWS アカウント を使用して、さまざまな 間で Amazon Relational Database Service (Amazon RDS) DB インスタンスのレプリケーション、追跡、および ロールバックのプロセスを自動化する方法を示しています AWS Lambda。この自動化を利用すると、組織の規模に関係なく、パフォーマンスへの影響や運用上のオーバーヘッドなしに RDS DB インスタンスの大規模なレプリケーションを実行できます。また、このパターンを使用して、組織が必須のデータガバナンス戦略やコンプライアンス要件に準拠し、さまざまな AWS アカウント および 間でデータをレプリケートして冗長化することもできます AWS リージョン。Amazon RDS データの大規模なクロスアカウントレプリケーションは、非効率的でエラーが発生しやすい手動プロセスであり、コストも時間もかかりますが、このパターンの自動化は、クロスアカウントレプリケーションを安全、効果的、効率的に実現するのに役立ちます。

前提条件と制限

前提条件

- 2 つ AWS アカウント
- ソースで起動および実行されている RDS DB インスタンス AWS アカウント
- 送信先の RDS DB インスタンスのサブネットグループ AWS アカウント
- ソースで作成 AWS アカウント され、送信先アカウントと共有される AWS Key Management Service （ AWS KMS) キー (ポリシーの詳細については、このパターンの追加情報セクションを参照してください ）。
- AWS KMS key 送信先アカウントのデータベースを暗号化 AWS アカウント する送信先の 。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

製品バージョン

- Python 3.9 (使用 AWS Lambda)
- PostgreSQL 11.3、13.x、14.x

アーキテクチャ

テクノロジースタック

- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- AWS Key Management Service (AWS KMS)
- AWS Lambda
- AWS Secrets Manager
- AWS Step Functions

ターゲット アーキテクチャ

次の図は、Step Functions を使用して、ソースアカウント (アカウント A) からターゲットアカウント (アカウント B) への RDS DB インスタンスのスケジュールされたオンデマンドレプリケーションを調整するためのアーキテクチャを示しています。

ソースアカウント (図のアカウント A) では、Step Functions ステートマシンが以下を実行します。

1. アカウント A の RDS DB インスタンスからスナップショットを作成します。
2. アカウント A AWS KMS key から を使用してスナップショットをコピーして暗号化します。転送中の暗号化を確保するために、DB インスタンスが暗号化されているかどうかにかかわらず、スナップショットは暗号化されます。
3. アカウント B にスナップショットへのアクセス権を付与することで、アカウント B と DB スナップショットを共有します。
4. SNS トピックに通知をプッシュし、SNS トピックがアカウント B の Lambda 関数を呼び出します。

宛先アカウント (図のアカウント B) では、Lambda 関数が Step Functions ステートマシンを実行して以下を調整します。

1. 共有スナップショットをアカウント A からアカウント B にコピーし、まずアカウント A AWS KMS key から を使用してデータを復号し、次にアカウント B AWS KMS key の を使用してデータを暗号化します。
2. Secrets Manager からシークレットを読み取り、現在の DB インスタンスの名前を取得します。
3. AWS KMS key Amazon RDS の新しい名前とデフォルトを使用して、スナップショットから DB インスタンスを復元します。
4. 新しいデータベースのエンドポイントを読み取り、Secrets Manager のシークレットを新しいデータベースエンドポイントで更新します。次に、以前の DB インスタンスにタグを付けて後で削除できるようにします。
5. データベースの最新の N 個のインスタンスを保持し、他のすべてのインスタンスを削除します。

ツール

AWS のサービス

- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体の管理に役立ちます AWS リージョン。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。

- AWS Step Functions は、Lambda 関数やその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。

コードリポジトリ

このパターンのコードは、GitHub 内の「Crossaccount RDS Replication」リポジトリで利用できます。

エピック

ワンクリック AWS アカウント で 全体の RDS DB インスタンスのレプリケーションを自動化する

タスク	説明	必要なスキル
CloudFormation スタックをソースアカウントにデプロイする。	1. ソースアカウント (アカウント A) AWS Management Console の にサインインし、CloudFormation コンソールを開きます。 2. ナビゲーションペインで、[Stacks] を選択します。 3. [スタックの作成] を選択し、[既存のリソースを使用 (リソースのインポート)] を選択します。 4. [リソースの識別] ページで [次へ] を選択します。 5. [テンプレートの指定] ページで、[テンプレートのアップロード] を選択します。 6. [ファイルを選択] を選択し、GitHub Crossaccount RDS Replication リポジトリから `Cloudformation-SourceAccountRDS.yaml` ファイルを	クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	選択して、[次へ] を選択します。 7. [スタック名] にスタックの名前を入力します。 8. [パラメータ] セクションで、スタックテンプレートで定義されている次のパラメータを指定します。 • [DestinationAccountNumber] には、移行先の RDS DB インスタンスのアカウント番号を入力します。 • KeyName には、 を入力します AWS KMS key。 • [ScheduleExpression] には cron 式を入力します (デフォルトは毎日午前 12:00)。 • [SourceDBIdentifier] に、ソースデータベースの名前を入力します。 • [SourceDBSnapshotName] には、スナップショットの名前を入力するか、デフォルトをそのまま使用します。 9. [次へ] を選択します。 10[スタックオプションの設定] ページで、デフォルトの値を変更せずに [次へ] を選択します。	

タスク	説明	必要なスキル
	11スタック設定を確認してから、[送信] を選択します。 12スタックの [リソース] タブを選択し、SNS トピックの Amazon リソースネーム (ARN) を書きとめておきます。	

タスク	説明	必要なスキル
CloudFormation スタックをデスティネーションアカウントにデプロイする。	1. 送信先アカウント (アカウント B) AWS Management Console の にサインインし、CloudFormation コンソールを開きます。 2. ナビゲーションペインで、[Stacks] を選択します。 3. [スタックの作成] を選択し、[既存のリソースを使用 (リソースのインポート)] を選択します。 4. [リソースの識別] ページで [次へ] を選択します。 5. [テンプレートの指定] ページで、[テンプレートのアップロード] を選択します。 6. [ファイル] を選択し、GitHub Crossaccount RDS Replication リポジトリから `Cloudformation-DestinationAccountRDS.yaml` ファイルを選択して、[次へ] を選択します。 7. [スタック名] にスタックの名前を入力します。 8. [パラメータ] セクションで、スタックテンプレートで定義されている次のパラメータを指定します。 • [DatabaseName] には、データベースの名前を入力します。	クラウドアーキテクト、DevOps エンジニア、クラウド管理者

タスク	説明	必要なスキル
	• [エンジン] には、ソースデータベースと一致するデータベースエンジンタイプを入力します。 • [DBInstanceClass] には、希望するデータベースインスタンスタイプを入力するか、デフォルトをそのまま使用します。 • [Subnetgroups] には、既存の VPC サブネットグループを入力します。サブネットグループを作成する手順については、Amazon RDS ドキュメントの「ステップ 2: DB サブネットグループを作成する」を参照してください。 • [SecretName] には、パスとシークレット名を入力するか、デフォルトをそのまま使用します。 • [SGID] には、送信先クラスターのセキュリティグループ ID を入力します。 • [KMSKey] には、移行先アカウントの KMS キーの ARN を入力します。 • [NooFolderInstances] には、ロールバックのために残しておきたい RDS	

タスク	説明	必要なスキル
	DB インスタンスの古いコピーの数を入力します。 9. [次へ] を選択します。 10[スタックオプションの設定] ページで、デフォルトの値を変更せずに [次へ] を選択します。 11スタック設定を確認してから、[送信] を選択します。 12スタックの [リソース] タブを選択し、`InvokeStepFunction` の Physical ID と ARN を書きとめます。	
宛先アカウントで RDS DB インスタンスが作成されたことを確認する。	1. にサインイン AWS Management Console し、Amazon RDS コンソールを開きます。 2. ナビゲーションペインで [データベース] を選択し、新しい RDS DB インスタンスが新しいクラスターの下に表示されることを確認します。	クラウド管理者、クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル

Lambda 関数を SNS トピックにサブスクライブする。	送信先アカウント AWS Command Line Interface (アカウント B AWS CLI) の Lambda 関数をソースアカウント (アカウント A) の SNS トピックにサブスクライブするには、次の () コマンドを実行する必要があります。
	
	アカウント A で、次のコマンドを実行します。
	
	`aws sns add-permission \ --label lambda-access --aws-account-id <DestinationAccount> \ --topic-arn <Arn of SNSTopic > \ --action-name Subscribe ListSubscriptionsB yTopic`
	
	アカウント B で、次のコマンドを実行します。
	
	`aws lambda add-permi ssion \ --function-name <Name of InvokeStepFunction > \ --source-arn <Arn of SNSTopic > \ --statement-id function-with-sns \ --action lambda:In vokeFunction \`
		クラウド管理者、クラウドアーキテクト、DBA

タスク	説明	必要なスキル

	`--principal sns.amazonaws.com`
	
	アカウント B で、次のコマンドを実行します。
	
	`aws sns subscribe \ --protocol "lambda" \ --topic-arn <Arn of SNSTopic> \ --notification-endpoint <Arn of InvokeStepFunction>`	

タスク	説明	必要なスキル
ソースアカウントの RDS DB インスタンスを宛先アカウントと同期する。	ソースアカウントで Step Functions ステートマシンを起動して、オンデマンドデータベースレプリケーションを開始します。 1. Step Functions コンソールを開きます。 2. ナビゲーションペインで、[ステートマシン] を選択します。 3. ステートマシンを選択します。 4. [実行] タブで関数を選択し、[実行を開始] を選択してワークフローを開始します。 **Note** スケジューラは、レプリケーションをスケジュールどおりに自動的に実行するのに役立ちますが、スケジューラはデフォルトでオフになっています。スケジューラーの Amazon CloudWatch ルールの名前は、宛先アカウントの CloudFormation スタックの [リソース] タブにあります。CloudWatch	クラウドアーキテクト、DevOps エンジニア、クラウド管理者

タスク	説明	必要なスキル
	Events ルールを変更する方法については、CloudWatch CloudWatch ドキュメントの[CloudWatch Events ルールの削除または無効化]」を参照してください。	
必要に応じて、データベースを以前のコピーのいずれかにロールバックする。	1. [Secrets Manager コンソール]を開きます。 2. シークレットのリストで、CloudFormation テンプレートを使用して作成したシークレットを選択します。アプリケーションはこのシークレットを使用してデスティネーションクラスターのデータベースにアクセスします。 3. 詳細ページからシークレット値を更新するには、[シークレット値] セクションで [シークレットを取得] の値を選択し、[編集] を選択します。 4. データベースエンドポイントの詳細を入力します。	クラウド管理者、DBA、DevOps エンジニア

関連リソース

- [クロスリージョンリードレプリカ] (Amazon RDS ドキュメント)
- [ブルー/グリーンデプロイ (]Amazon RDS ドキュメント)

追加情報

次のポリシー例を使用して、 AWS KMS key を共有できます AWS アカウント。

```
{
    "Version": "2012-10-17",
    "Id": "cross-account-rds-kms-key",
    "Statement": [
        {
            "Sid": "Enable user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<SourceAccount>:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow administration of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<DestinationAccount>:root"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
```

```
                    "arn:aws:iam::<DestinationAccount>:root",
                    "arn:aws:iam::<SourceAccount>:root"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant"
            ],
            "Resource": "*"
        }
    ]
}
```

Systems Manager と EventBridge を使用した SAP HANA データベースの自動バックアップ

作成者: Ambarish Satarkar (AWS)、Gaurav Rath (AWS)

概要

このパターンでは、AWS Systems Manager、Amazon EventBridge、Amazon Simple Storage Service (Amazon S3)、および AWS Backint Agent for SAP HANA エージェント を使用して SAP HANA データベースのバックアップを自動化する方法を説明します。

このパターンでは、`BACKUP DATA` コマンドを使用するシェルスクリプトベースのアプローチが可能になり、多数のシステムにまたがるオペレーティングシステム (OS) インスタンスごとにスクリプトやジョブ設定を管理する必要がなくなります。

> **Note**
>
> 2023 年 4 月、AWS Backup は Amazon Elastic Compute Cloud (Amazon EC2) での SAP HANA データベースのサポートを発表しました。詳細については、「Amazon EC2 インスタンスのバックアップでの SAP HANA データベース」を参照してください。

組織のニーズに応じて、AWS Backup サービスを使用して SAP HANA データベースを自動的にバックアップすることも、このパターンを使用することもできます。

前提条件と制限

前提条件

- Systems Manager 向けに設定されているマネージド型 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで、サポート対象のリリースを含む既存の SAP HANA インスタンス (実行中)
- Systems Manager エージェント (SSM エージェント) 2.3.274.0 以降のバージョンがインストール済み
- パブリックアクセスが有効になっていない S3 バケット
- `hdbuserstore` という名前の `SYSTEM` キー

- オートメーションランブックをスケジュールどおりに実行するための AWS Identity and Access Management (IAM) ロール
- `AmazonSSMManagedInstanceCore` と `ssm:StartAutomationExecution` ポリシーは Systems Manager Automation サービスロールにアタッチされます。

機能制限

- AWS Backint Agent for SAP HANA は重複排除をサポートしていません。
- AWS Backint Agent for SAP HANA はデータ圧縮をサポートしていません。

製品バージョン

AWS Backint Agent は、以下のオペレーティングシステムでサポートされています。

- SUSE Linux Enterprise Server
- SUSE Linux Enterprise Server for SAP
- Red Hat Enterprise Linux for SAP

AWS Backint Agent では、以下のデータベースがサポートされています。

- SAP HANA 1.0 SP12 (シングルノードとマルチノード)
- SAP HANA 2.0 以降 (シングルノードとマルチノード)

アーキテクチャ

ターゲットテクノロジースタック

- AWS Backint Agent
- Amazon S3
- AWS Systems Manager
- Amazon EventBridge
- SAP HANA

ターゲット アーキテクチャ

次の図は、AWS Backint Agent、S3 バケット、Systems Manager と EventBridge をインストールするスクリプトを示しています。これらのスクリプトは、コマンド ドキュメントを使用して定期的なバックアップをスケジュールします。

自動化とスケール

- Systems Manager Automation ランブックを使用すると、複数の AWS Backint Agent をインストールできます。
- Systems Manager ランブックを実行するたびに、ターゲットの選択に基づいて、n 個の SAP HANA インスタンスまでスケールできます。
- EventBridge は SAP HANA のバックアップを自動化できます。

ツール

- 「AWS Backint Agent for SAP HANA」は、既存のワークフローと統合して、設定ファイルで指定した S3 バケットに SAP HANA データベースをバックアップするスタンドアロン アプリケーションです。AWS Backint Agent は SAP HANA データベースの完全バックアップ、増分バックアップ、差分バックアップをサポートしています。SAP HANA データベースサーバーで実行され、バックアップとカタログが SAP HANA データベースから AWS Backint Agent に転送されます。
- Amazon EventBridge は、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。EventBridge は、アプリケーション、Software as a Service (SaaS) アプリケーション、および AWS サービスからのリアルタイムデータのストリームを、AWS Lambda 関数、API デスティネーションを使用した HTTP 呼び出しエンドポイント、または他のアカウントのイベントバスなどのターゲットに配信します。
- Amazon Simple Storage Service (Amazon S3)は、オブジェクトストレージを提供します。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- AWS Systems Manager は、AWS 上のインフラストラクチャを可視化し、制御するためのサービスです。Systems Manager コンソールを使用すると、複数の AWS サービスからの運用データを表示し、AWS リソース全体の運用タスクを自動化できます。

コード

このパターンのコードは、「aws-backint-automated-backup」GitHub リポジトリで利用できます。

エピック

hdbuserstore キーシステムを作成します

タスク	説明	必要なスキル

hdbuserstore キーシステムを作成します	1. /usr/sap/<SID>/HDB <Inst No>/exe に移動します。
	2. 次のコマンドを実行して、XX をSAP HANA データベースのインスタンス番号として指定します。
	`hdbuserstore -i set SYSTEM <hostname>:3XX13@SYSTEMDB SYSTEM`
	たとえば、インスタンス番号 saphanadb の SAP HANA ホスト 00 で、次のコマンドを実行します。
	`hdbuserstore -i set SYSTEM saphanadb:30013@SYSTEMDB SYSTEM`
		AWS 管理者、SAP HANA 管理者

AWS Backint Agent のインストール

タスク	説明	必要なスキル
AWS Backint Agent のインストール	AWS Backint Agent ドキュメントの「AWS Backint Agent for SAP HANA のインストー	AWS 管理者、SAP HANA 管理者

タスク	説明	必要なスキル
	ルと設定」の手順に従ってください。	

Systems Manager コマンドのドキュメントを作成する

タスク	説明	必要なスキル
Systems Manager コマンドのドキュメントを作成する	1. AWS マネジメントコンソールにサインインし、AWS Systems Manager コンソールを開きます。 2. [ドキュメント] を選択し、[自分が所有] を選択します。 3. SAP HANA データベースと同じ AWS リージョンにいることを確認します。 4. [ドキュメント、コマンド、またはセッションを作成] を選択してドキュメントを作成します。 5. スペースを含まない、一意でわかりやすい名前を使用してください (たとえば、SAP HANA-Backup)。 6. [ドキュメントタイプ] が [コマンド ドキュメント] に設定されていることを確認します。 7. [コンテンツ] ヘッダーの下には、いくつかのサンプルコードがあります。JSON コードタイプが選	AWS 管理者、SAP HANA 管理者

タスク	説明	必要なスキル
	択されていることを確認し、このコードを GitHub リポジトリの HDB_Backup_SSM_Document.json ファイル内にあるコードに置き換えます。 8. [ドキュメントの作成] を選択します。 9. [自分が所有] セクションでドキュメントを確認します。	

定期的にバックアップをスケジュールします

タスク	説明	必要なスキル
Amazon EventBridge を使用して定期バックアップをスケジュールします。	1. Amazon EventBridge コンソールを開き、[ルール]、[ルールの作成] の順に選択します。 2. [ルールの詳細を定義] 画面で、ルールについて一意の名前と説明を入力し、デフォルトのイベントバスを使用します。 3. [ルールタイプ] で、[スケジュール]、[次へ] の順に選択します。 4. [スケジュールを定義] 画面で、必要な頻度に基づいて適切なスケジュールパターン、cron 式または rate 式を選択します。	AWS 管理者、SAP HANA 管理者

タスク	説明	必要なスキル
	5. [ターゲットの選択] 画面の [ターゲットタイプ] で、AWS サービスを選択します。[ターゲットの選択] で、[Systems Manager Run Command] を選択します。 6. 先ほど作成したドキュメントを選択します。 7. ターゲットキーとターゲット値に、インスタンス ID を指定します。タグ名とタグ値を使用し、複数のインスタンスを追加できます。 8. [自動化パラメータの設定] で、増分バックアップまたは差分バックアップに [定数] を選択します。完全バックアップを実行する場合は、[パラメータなし] を選択します。 9. [新規ロールの作成]、または [既存ロールの使用] を選択します。既存のロールを使用する場合は、ターゲットを呼び出すために必要なポリシーがあることを確認してください。 10.デフォルトの追加設定はそのままにして、[次へ] を選択します。 11.[タグの設定] 画面はオプションです。[次へ] をクリックします。	

タスク	説明	必要なスキル
	12[確認と作成] 画面で、ルール設定を確認し、[作成] を選択します。ルールは正常に作成されます。 S3 バケットパスから,バックアップの完了を確認できます。 `s3:/<your_bucket_name>/<target folder>/<SID>/usr/sap/<SID>/SYS/global/hdb/backint/DB_<SID>/` SAP HANA バックアップカタログからバックアップを確認することもできます。	

関連リソース

- 「AWS Backint Agent for SAP HANA」
- 「AWS Backint Agent for SAP HANA のインストールと設定]

Python アプリケーションを使用して Amazon DynamoDB の PynamoDB モデルと CRUD 関数を自動的に生成する DynamoDB

作成者: Vijit Vashishtha (AWS)、Dheeraj Alimchandani (AWS)、Dhananjay Karanjkar (AWS)

概要

Amazon DynamoDB データベースオペレーションを効率的に実行するために、エンティティと作成、読み取り、更新、削除 (CRUD) オペレーション関数を要求するのが一般的です。PynamoDB は、Python 3 をサポートする Python ベースのインターフェイスです。また、Amazon DynamoDB トランザクションのサポート、属性値の自動シリアル化と逆シリアル化、Flask や Django などの一般的な Python フレームワークとの互換性などの機能も提供します。このパターンは、PynamoDB モデルと CRUD オペレーション関数の自動作成を合理化するライブラリを提供することで、Python と DynamoDB を操作するデベロッパーに役立ちます。 PynamoDB データベーステーブルに不可欠な CRUD 関数を生成しますが、PynamoDB モデルと CRUD 関数を Amazon DynamoDB テーブルからリバースエンジニアリングすることもできます。このパターンは、Python ベースのアプリケーションを使用してデータベース操作を簡素化するように設計されています。

このソリューションの主な機能は次のとおりです。

- JSON スキーマから PynamoDB モデルへ – JSON スキーマファイルをインポートして Python で PynamoDB モデルを自動的に生成します。
- CRUD 関数の生成 – DynamoDB テーブルで CRUD オペレーションを実行する関数を自動的に生成します。
- DynamoDB からのリバースエンジニアリング – PynamoDB オブジェクトリレーショナルマッピング (ORM) を使用して、既存の Amazon DynamoDB テーブルの PynamoDB モデルと CRUD 関数をリバースエンジニアリングします。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Python バージョン 3.8 以降、ダウンロードしてインストール
- Jinja2 バージョン 3.1.2 以降、ダウンロードしてインストール
- ORM を生成する Amazon DynamoDB テーブル

- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- PynamoDB バージョン 5.4.1 以降、インストール済み

アーキテクチャ

ターゲットテクノロジースタック

- JSON スクリプト
- Python アプリケーション
- PynamoDB モデル
- Amazon DynamoDB データベースインスタンス

ターゲット アーキテクチャ

1. 入力 JSON スキーマファイルを作成します。この JSON スキーマファイルは、 および CRUD 関数から PynamoDB モデルを作成するそれぞれの DynamoDB テーブルの属性を表します。PynamoDB これには、次の 3 つの重要なキーが含まれています。
   - `name` – ターゲット DynamoDB テーブルの名前。
   - `region` – テーブル AWS リージョン がホストされている 。
   - `attributes` – パーティションキー (ハッシュ属性とも呼ばれます)、ソートキー、ローカルセカンダリインデックス、グローバルセカンダリインデックス、非キー属性など、ターゲットテーブルの一部である属性。このツールでは、アプリケーションがターゲットテーブルから直接キー属性を取得するため、入力スキーマはキー以外の属性のみを提供することを想定しています。JSON スキーマファイルで属性を指定する方法の例については、このパターンの???「追加情報」セクションを参照してください。
2. Python アプリケーションを実行し、JSON スキーマファイルを入力として指定します。
3. Python アプリケーションは JSON スキーマファイルを読み取ります。
4. Python アプリケーションは DynamoDB テーブルに接続して、スキーマとデータ型を取得します。アプリケーションは describe_table オペレーションを実行し、テーブルのキー属性とインデックス属性を取得します。
5. Python アプリケーションは、JSON スキーマファイルと DynamoDB テーブルの属性を組み合わせます。Jinja テンプレートエンジンを使用して、PynamoDB モデルと対応する CRUD 関数を生成します。

6. PynamoDB モデルにアクセスして、DynamoDB テーブルで CRUD オペレーションを実行します。

ツール

AWS のサービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。

その他のツール

- Jinja は、テンプレートを最適化された Python コードにコンパイルする拡張可能なテンプレートエンジンです。このパターンでは、Jinja を使用して、テンプレート内にプレースホルダーとロジックを埋め込むことで動的コンテンツを生成します。
- PynamoDB は、Amazon DynamoDB 用の Python ベースのインターフェイスです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub 自動生成 PynamoDB モデルと CRUD 関数リポジトリで使用できます。リポジトリは、コントローラーパッケージとテンプレートの 2 つの主要部分に分かれています。

コントローラーパッケージ

コントローラー Python パッケージには、PynamoDB モデルと CRUD 関数の生成に役立つ主要なアプリケーションロジックが含まれています。以下の要素が含まれます。

- `input_json_validator.py` – この Python スクリプトは、入力 JSON スキーマファイルを検証し、ターゲット DynamoDB テーブルのリストと各テーブルに必要な属性を含む Python オブジェクトを作成します。
- `dynamo_connection.py` – このスクリプトは、DynamoDB テーブルへの接続を確立し、`describe_table`オペレーションを使用してPynamoDB モデルの作成に必要な属性を抽出します。

- `generate_model.py` – このスクリプトには、入力 JSON スキーマファイルと `describe_table`オペレーションに基づいて PynamoDB モデル`GenerateModel`を作成する Python クラスが含まれています。
- `generate_crud.py` – JSON スキーマファイルで定義されている DynamoDB テーブルの場合、このスクリプトは `GenerateCrud`オペレーションを使用して Python クラスを作成します。

テンプレート

この Python ディレクトリには、次の Jinja テンプレートが含まれています。

- `model.jinja` – この Jinja テンプレートには、PynamoDB モデルスクリプトを生成するためのテンプレート式が含まれています。
- `crud.jinja` – この Jinja テンプレートには、CRUD 関数スクリプトを生成するためのテンプレート式が含まれています。

エピック

環境をセットアップする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	次のコマンドを入力して、PynamoDB モデルと CRUD 関数の自動生成リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/amazon-reverse-engineer-dynamodb.git`	アプリ開発者
Python 環境をセットアップします。	1. クローンされたリポジトリの最上位ディレクトリに移動します。	アプリ開発者

タスク	説明	必要なスキル

	`cd amazon-reverse-engineer-dynamodb`
	2. 次のコマンドを入力して、必要なライブラリとパッケージをインストールします。
	`pip install -r requirements.txt`	

PynamoDB モデルと CRUD 関数を生成する

タスク	説明	必要なスキル
JSON スキーマファイルを変更します。	1. クローンされたリポジトリの最上位ディレクトリに移動します。 `cd amazon-reverse-engineer-dynamodb` 2. 任意の工ディタで `test.json` ファイルを開きます。このファイルをリファレンスとして使用して独自の JSON スキーマファイルを作成することも、このファイルの値を環境に合わせて更新することもできます。 3. ターゲット DynamoDB テーブルの名前 AWS リー	アプリ開発者

タスク	説明	必要なスキル
	ジョン、属性値を変更します。 **Note** JSON スキーマファイルに存在しないテーブルを定義した場合、このソリューションはそのテーブルのモデルまたは CRUD 関数を生成しません。 4. `test.json` ファイルを保存して閉じます。このファイルは新しい名前で保存することをお勧めします。	
Python アプリケーションを実行します。	次のコマンドを入力して、PynamoDB モデルと CRUD 関数を生成します。ここで、`<input_schema.json>` は JSON スキーマファイルの名前です。 `python main.py --file <input_schema.json>`	アプリ開発者

PynamoDB モデルと CRUD 関数を検証する

タスク	説明	必要なスキル
生成された PynamoDB モデルを確認します。	1. クローンされたリポジトリの最上位ディレクトリ	アプリ開発者

タスク	説明	必要なスキル
	で、次のコマンドを入力してmodelsリポジトリに移動します。 `cd models` 2. デフォルトでは、このソリューションは PynamoDB モデルファイル に名前を付けますdemo_model.py 。このファイルが存在することを確認します。	
生成された CRUD 関数を確認します。	1. クローンされたリポジトリの最上位ディレクトリで、次のコマンドを入力してcrudリポジトリに移動します。 `cd crud` 2. デフォルトでは、このソリューションはスクリプト に名前を付けますdemo_crud.py 。このファイルが存在することを確認します。 3. demo_crud.py ファイルの Python クラスを使用して、ターゲット DynamoDB テーブルで CRUD オペレーションを実行します。オペレーションが正常に完了したことを確認します。	アプリ開発者

関連リソース

- Amazon DynamoDB のコアコンポーネント (DynamoDB ドキュメント)DynamoDB
- セカンダリインデックスによるデータアクセスの改善 (DynamoDB ドキュメント)

追加情報

JSON スキーマファイルの属性例

```
[
{
"name": "test_table",
"region": "ap-south-1",
"attributes": [
{
"name": "id",
"type": "UnicodeAttribute"
},
{
"name": "name",
"type": "UnicodeAttribute"
},
{
"name": "age",
"type": "NumberAttribute"
}
]
}
]
```

クラウドカストディアンを使用して Amazon RDS へのパブリックアクセスをブロック

アベイ・クマール (AWS) とドワリカ・パトラ (AWS) によって作成されました

概要

多くの組織がワークロードとサービスを複数のクラウドベンダーで実行しています。このようなハイブリッドクラウド環境では、個々のクラウドプロバイダーが提供するセキュリティに加えて、クラウドインフラストラクチャには厳格なクラウドガバナンスが必要です。Amazon Relational Database Service (Amazon RDS) のようなクラウドデータベースは、アクセスや許可に関する脆弱性がないか監視する必要がある重要なサービスの1つです。セキュリティグループを設定することで Amazon RDS データベースへのアクセスを制限できますが、2 つ目の保護レイヤーを追加してパブリックアクセスなどのアクションを禁止することもできます。パブリックアクセスのブロックは、一般データ保護規則 (GDPR)、医療保険の相互運用性と説明責任に関する法律 (HIPAA)、米国国立標準技術研究所 (NIST)、および Payment Card Industry Data Security Standard (PCI DSS) コンプライアンスに役立ちます。

クラウドカストディアンは、Amazon RDS などの Amazon Web Services (AWS) リソースのアクセス制限を適用するために使用できるオープンソースのルールエンジンです。Cloud Custodian では、定義されたセキュリティおよびコンプライアンス基準に照らして環境を検証するルールを設定できます。Cloud Custodian を使用すると、セキュリティポリシー、タグポリシーの順守、未使用リソースのガベージコレクション、コスト管理を実現することで、クラウド環境を管理できます。Cloud Custodian を使用すると、単一のインターフェースでハイブリッドクラウド環境にガバナンスを実装できます。例えば、Cloud Custodian インターフェイスを使用して AWS および Microsoft Azure とやり取りし AWS Config、 AWS セキュリティグループや Azure ポリシーなどのメカニズムを使用する労力を減らすことができます。

このパターンでは、 で Cloud Custodian AWS を使用して Amazon RDS インスタンスにパブリックアクセシビリティの制限を適用する手順を示します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 「キーペア」
- AWS Lambda インストール済み

アーキテクチャ

次の図は、Amazon RDS での Cloud Custodian によるポリシーのデプロイ AWS Lambda、`CreateDBInstance`イベント AWS CloudTrail の開始、Lambda 関数設定の `PubliclyAccessible` false を示しています。

ツール

AWS のサービス

- AWS CloudTrail は、 のガバナンス、コンプライアンス、運用リスクを監査するのに役立ちます AWS アカウント。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。

その他のツール

- 「Cloud Custodian」 は、多くの組織がパブリッククラウドアカウントの管理に使用しているツールとスクリプトを 1 つのオープンソースツールに統合します。ポリシーの定義と実施にはステートレスなルールエンジンを使用し、クラウドインフラストラクチャの指標、構造化された出力、詳細なレポート機能を備えています。サーバーレスランタイムと緊密に統合されているため、運用上のオーバーヘッドを低く抑えながらリアルタイムの修復と対応が可能です。

エピック

のセットアップ AWS CLI

タスク	説明	必要なスキル
をインストールします AWS CLI。	をインストールするには AWS CLI、 AWS ドキュメントの指示に従います。	AWS 管理者

AWS 認証情報を設定します。	や使用する AWS リージョン出力形式など AWS、 が操作 AWS CLI に使用する設定を構成します。
	
	`$>aws configure`
	`AWS Access Key ID`
	`[None]: <your_acc`
	`ess_key_id>`
	`AWS Secret Access Key`
	`[None]: <your_sec`
	`ret_access_key>`
	`Default region name`
	`[None]:`
	`Default output format`
	`[None]:`
	
	詳細については、AWS のドキュメントを参照してください。
		AWS 管理者

IAM ロールを作成します。	Lambda 実行ロールを使用して IAM ロールを作成するには、次のコマンドを実行します。
	
	`aws iam create-role --`
	`role-name lambda-ex --`
	`assume-role-policy-`
	`document '{"Version":`
		AWS DevOps

タスク	説明	必要なスキル
	`"2012-10-17","Statement": [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}`	

クラウドカストディアンをセットアップする

タスク	説明	必要なスキル
クラウドカストディアンをインストールします。	ご使用のオペレーティングシステムと環境に Cloud Custodian をインストールするには、「Cloud Custodian ドキュメント」の指示に従ってください。	DevOps エンジニア
クラウドカストディアンスキーマをチェックします。	ポリシーを実行できる Amazon RDS リソースの完全なリストを確認するには、以下のコマンドを使用します。 `custodian schema aws.rds`	DevOps エンジニア
クラウドカストディアンポリシーを作成します。	クラウドカストディアンポリシーファイルにあるコードを YAML 拡張子を使用して追加情報セクションに保存します。	DevOps エンジニア
パブリックにアクセス可能なフラグを変更するクラウドカストディアンアクションを定義します。	1. カストディアンコード (例:/Users/abcd/custodian/lib/python3.9/	DevOps エンジニア

タスク	説明	必要なスキル
	site-packages/c7n/resources/rds.py) を探します。 2. RDSSetPublicAvailability で rds.py クラスを探し、追加情報セクションの c7n resources rds.py ファイルにあるコードを使用してこのクラスを変更します。	
リサルの実行を行います。	(オプション) リソースに対してアクションを実行せずに、ポリシーによってどのリソースが識別されているかを確認するには、以下のコマンドを使用します。 `custodian run -dryrun <policy_name>.yaml -s <output_directory>`	DevOps エンジニア

ポリシーをデプロイします。

タスク	説明	必要なスキル
Lambda を使用してポリシーをデプロイします。	ポリシーを実行する Lambda 関数を作成するには、次のコマンドを使用します。 `custodian run -s policy.yaml` その後、このポリシーは イベントによって開始されます	DevOps エンジニア

タスク	説明	必要なスキル
	AWS CloudTrail `CreateDBInstance` 。 その結果、 AWS Lambda は条件に一致するインスタンスfalseのパブリックアクセス可能なフラグを に設定します。	

関連リソース

- AWS Lambda website
- Amazon RDS ウェブサイト
- クラウドカストディアンのドキュメント

追加情報

クラウドカストディアンポリシー (YAML ファイル)

```
policies:
  - name: "block-public-access"
    resource: rds
    description: |
      This Enforcement blocks public access for RDS instances.
    mode:
      type: cloudtrail
      events:
        - event: CreateDBInstance # Create RDS instance cloudtrail event
          source: rds.amazonaws.com
          ids: requestParameters.dBInstanceIdentifier
      role: arn:aws:iam::1234567890:role/Custodian-compliance-role
    filters:
      - type: event
        key: 'detail.requestParameters.publiclyAccessible'
        value: true
    actions:
      - type: set-public-access
```

```
        state: false
```

c7n リソース rds.py ファイル

```
@actions.register('set-public-access')
 class RDSSetPublicAvailability(BaseAction):

     schema = type_schema(
         "set-public-access",
         state={'type': 'boolean'})
     permissions = ('rds:ModifyDBInstance',)

     def set_accessibility(self, r):
         client = local_session(self.manager.session_factory).client('rds')
         waiter = client.get_waiter('db_instance_available')
         waiter.wait(DBInstanceIdentifier=r['DBInstanceIdentifier'])
         client.modify_db_instance(
             DBInstanceIdentifier=r['DBInstanceIdentifier'],
             PubliclyAccessible=self.data.get('state', False))


     def process(self, rds):
         with self.executor_factory(max_workers=2) as w:
             futures = {w.submit(self.set_accessibility, r): r for r in rds}
             for f in as_completed(futures):
                 if f.exception():
                     self.log.error(
                         "Exception setting public access on %s  \n %s",
                         futures[f]['DBInstanceIdentifier'], f.exception())
         return rds
```

セキュリティハブの統合

クラウドカストディアンは と統合AWS Security Hubして、セキュリティ検出結果を送信し、修復アクションを試みることができます。詳細については、「 とのクラウドカストディアン統合の発表 AWS Security Hub」を参照してください。

Amazon DynamoDB へのクロスアカウントアクセスを設定する

作成者: Shashi Dalmia (AWS)、Esteban Serna Parra (AWS)、Imhoertha Ojior (AWS)

概要

このパターンでは、リソースベースのポリシーを使用して Amazon DynamoDB へのクロスアカウントアクセスを設定する手順について説明します。DynamoDB を使用するワークロードでは、ワークロード分離戦略を使用してセキュリティの脅威を最小限に抑え、コンプライアンス要件を満たすことがより一般的になっています。ワークロード分離戦略を実装するには、 AWS Identity and Access Management (IAM) アイデンティティベースのポリシーを使用して DynamoDB リソースへのクロスアカウントおよびクロスリージョンアクセスが必要になることがよくあります。これには、IAM アクセス許可の設定と、 間の信頼関係の確立が含まれます AWS アカウント。

DynamoDB のリソースベースのポリシーは、クロスアカウントワークロードのセキュリティ体制を大幅に簡素化します。このパターンでは、別のアカウントの DynamoDB データベーステーブルにデータを書き込む AWS アカウント ように AWS Lambda 関数を 1 つに設定する方法を示すステップとサンプルコードを示します。

前提条件と制限

前提条件

- 2 つの がアクティブです AWS アカウント。このパターンでは、これらのアカウントを「アカウント A」と「アカウント B」と呼びます。
- AWS Command Line Interface （ AWS CLI) をインストールし、アカウント A にアクセスするように設定して、DynamoDB テーブルを作成します。このパターンの他のステップでは、IAM、DynamoDB、および Lambda コンソールの使用手順について説明します。 AWS CLI 代わりに を使用する予定の場合は、両方のアカウントにアクセスするように設定します。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、単一アカウントアーキテクチャを示しています。 AWS Lambda、Amazon Elastic Compute Cloud (Amazon EC2)、DynamoDB はすべて同じアカウントにあります。このシナリオでは、Lambda 関数と Amazon EC2 インスタンスが DynamoDB にアクセスできます。DynamoDB テーブルへのアクセスを許可するには、IAM でアイデンティティベースのポリシーを作成するか、DynamoDB でリソースベースのポリシーを作成できます。

次の図は、マルチアカウントアーキテクチャを示しています。1 つの のリソースが別のアカウントの DynamoDB テーブルにアクセス AWS アカウント する必要がある場合は、DynamoDB でリソースベースのポリシーを設定して、必要なアクセスを許可する必要があります。たとえば、次の図では、リソースベースのポリシーを使用して、アカウント A の DynamoDB テーブルへのアクセスがアカウント B の Lambda 関数に許可されています。

このパターンでは、Lambda と DynamoDB 間のクロスアカウントアクセスについて説明します。両方のアカウントに適切なアクセス許可 AWS のサービス が設定されている場合は、他の にも同様の手順を使用できます。たとえば、アカウント A の Amazon Simple Storage Service (Amazon S3) バケットへのアクセスを Lambda 関数に許可する場合は、Amazon S3 でリソースベースのポリシーを作成し、アカウント B の Lambda 実行ロールにアクセス許可を追加できます。

ツール

AWS のサービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを提供します。
- AWS Identity and Access Management (IAM) は、AWS リソースの使用を認証および許可されているユーザーを制御することで、AWS リソースへのアクセスを安全に管理します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

コード

このパターンには、「追加情報」セクションのサンプルコードが含まれており、アカウント A の DynamoDB テーブルに書き込むようにアカウント B の Lambda 関数を設定する方法を示しています。コードは、説明とテストの目的でのみ提供されています。このパターンを本番環境に実装する場合は、コードを参照として使用し、独自の環境に合わせてカスタマイズします。

ベストプラクティス

- DynamoDB ドキュメントのリソースベースのポリシーのベストプラクティスに従ってください。
- 最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

アカウント B で Lambda 関数の IAM ポリシーとロールを作成する

タスク	説明	必要なスキル
アカウント B でポリシーを作成します。	この IAM ポリシーは、アカウント A の DynamoDB テーブルの PutItem アクションを許可します。 1. でアカウント B にサインインします AWS Management Console。 2. [IAM コンソール] を開きます。 3. ナビゲーションペインで ポリシーを選択してから ポリシーの作成を選択します。 4. アクセス許可の指定ページで、ポリシーエディタで JSON を選択します。 5. 以下のポリシーを入力します。	AWS 全般

タスク	説明	必要なスキル

	<pre>{
	  "Version":
	 "2012-10-17",
	  "Statement": [
	  {
	  "Sid": "Statemen
	t1",
	  "Effect":
	 "Allow",
	  "Action":
	 "dynamodb:PutItem",
	  "Resource":
	 "arn:aws:dynamodb:
	<Region>:<Account-
	A-ID>:table/Table-
	Account-A"
	  }
	  ]
	}</pre>
	6. <Region> と を自分の値<Account-A-ID> に置き換え、次へを選択します。
	7. ポリシー名には、 など、ポリシーの一意の名前を入力しますDynamoDB-PutItem-Policy 。
	8. (オプション) ポリシーの説明を追加します。
	9. [Create policy] を選択します。	

タスク	説明	必要なスキル
アカウント B でロールを作成します。	アカウント B の Lambda 関数は、この IAM ロールを使用してアカウント A の DynamoDB テーブルにアクセスします。 1. [IAM コンソール] を開きます。 2. ナビゲーションペインで ロール を選択してから、ロールを作成する を選択します。 3. [Select trusted entity] (信頼されたエンティティを選択) で、[AWS のサービス] を選択します。 4. ユースケースセクションで、Lambda を選択します。 5. [Next: Permissions] (次へ: アクセス許可) を選択します。 6. [フィルタポリシー] ボックスに DynamoDB と入力します。 7. DynamoDB ポリシーのリストで、 を選択しますDynamoDB-PutItem-Policy 。 8. フィルタポリシーボックスをクリアし、Lambda を入力します。	AWS 全般

タスク	説明	必要なスキル
	9. Lambda ポリシーのリストで、AWSLambdaExecute を選択します。 10[次へ:名前、確認、および作成] を選択します。 11[ロール名] に、DynamoDB-PutItemAccess などのロールの一意の名前を入力します。 12.(オプション) ロールの説明を追加します。 13(オプション) タグをキーバリューのペアとしてアタッチして、メタデータをロールに追加します。 14[ロールの作成] を選択します。 ロールの作成の詳細については、「IAM ドキュメント」を参照してください。	

タスク	説明	必要なスキル
ロールの ARN を書き留めておきます。	1. IAM コンソールを開きます。 2. ナビゲーションペインで [Roles (ロール)] を選択します。 3. 検索ボックスに と入力しDynamoDB-PutItemAccess 、ロールを選択します。 4. ロールの概要ページで、Amazon リソースネーム (ARN) をコピーします。Lambda 関数を設定するときに ARN を使用します。	AWS 全般

アカウント A でDynamoDB テーブルを作成する

タスク	説明	必要なスキル
DynamoDB テーブルを作成します。	DynamoDB テーブルを作成するには、次の AWS CLI コマンドを使用します。 `aws dynamodb create-table \ --table-name Table-Account-A \ --attribute-definitions \ AttributeName=category,AttributeType=S \`	AWS 全般

タスク	説明	必要なスキル

	Attribute Name=item,AttributeType=S \
	--key-schema \
	Attribute Name=category,KeyType=HASH \
	Attribute Name=item,KeyType=RANGE \
	--provisioned-throughput \
	ReadCapacityUnits=5,WriteCapacityUnits=5 \
	--resource-policy \
	'{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Sid": "Statement1",
	"Effect": "Allow",
	"Principal": {
	"AWS": "arn:aws:iam::<Account-B-ID>:role/<Role-Name>"
	},
	"Action": "dynamodb:PutItem",
	"Resource": "arn:aws:dynamodb:<Region>:<Account-A-ID>:table/Table-Account-A"	

タスク	説明	必要なスキル

```
            }

        ]
    }'
```

このコードサンプルで以下を置き換えます。

- <Account-B-ID> はアカウント B の ID です。
- <Role-Name> は、 など、作成した IAM ロールの名前ですDynamoDB-PutItemAccess 。
- <Region> は、DynamoDB テーブルを作成する AWS リージョン です。
- <Account-A-ID> はアカウント A の ID です。

> **Note**
>
> create-table ステートメントでリソースベースのポリシー設定を指定するには、 --resource-policyフラグを使用します。このポリシーは、アカウント A の DynamoDB テーブルの ARN を参照します。

タスク	説明	必要なスキル
	テーブルの作成の詳細については、「DynamoDB のドキュメント」を参照してください。	

アカウント B で Lambda 関数を作成する

タスク	説明	必要なスキル
DynamoDB にデータを書き込むLambda 関数を作成します。	1. でアカウント B にサインインします AWS Management Console。 2. Lambdaのコンソールを開きます。 3. ナビゲーションペインで、[関数]、[関数を作成] の順に選択します。 4. [名前] に `lambda_write_function` と入力します。 5. [ランタイム] には、[Python 3.8] またはそれ以降を選択します。 6. 「デフォルトの実行ロールを変更する」で、「既存のロールを使用する」を選択します。 7. 既存のロールで、 など、作成した IAM ロールを選択しますDynamoDB-`PutItemAccess` 。 8. [Create function (関数の作成)] を選択します。	AWS 全般

タスク	説明	必要なスキル
	9. コードタブで、このパターンの追加情報セクションで提供されているサンプルコードを貼り付けます。このコードサンプルで以下を置き換えます。 • `<Account-A-ID>` はアカウント A の ID です。 • `<Region>` は、DynamoDB テーブルを作成した AWS リージョン です。 10[デプロイ] を選択します。 11[テスト] を選択します。これにより、テストイベントを設定するよう求められます。などの任意の名前で新しいイベントを作成し `MyTestEventForWrite` 、設定を保存します。 12[テスト] を再度選択します。これにより、指定したイベント名で Lambda 関数が実行されます。 13.関数からの出力を確認します。これは、関数がアカウント A の DynamoDB テーブルにアクセスし、それにデータを書き込めたことを示します。 Lambda 関数の作成についての詳細は、「Lambda ドキュ	

タスク	説明	必要なスキル
	メント」を参照してください。	

クリーンアップ

タスク	説明	必要なスキル
リソースの削除	このパターンで作成されたリソースに関連するコストが発生しないようにするには、次の操作を実行してこれらのリソースを削除します。 1. アカウント B で、DynamoDB に接続するために作成した Lambda 関数を削除します。手順については、Lambda ドキュメントを参照してください。 2. アカウント A で、作成した DynamoDB テーブルを削除します。手順については、DynamoDB ドキュメントを参照してください。 3. セキュリティのベストプラクティスについては、不要になった IAM ポリシー (`DynamoDB-PutItem-Policy`) を削除します。詳細については、「IAM ドキュメント」を参照してください。 4. セキュリティのベストプラクティスとして、不	AWS 全般

タスク	説明	必要なスキル
	要になった IAM ロール (DynamoDB-PutItemAccess) を削除します。詳細については、「IAM ドキュメント」を参照してください。	

トラブルシューティング

問題	ソリューション
Lambda 関数を作成すると、ResourceNotFoundException エラーが表示されます。	アカウント A の AWS リージョン と ID が正しく入力されていることを確認します。これらは DynamoDB テーブルの ARN の一部です。

関連リソース

- 「DynamoDB の使用を開始する」(DynamoDB ドキュメント)
- Lambda の開始方法 (Lambda ドキュメント)
- DynamoDB のリソースベースのポリシーの使用 (DynamoDB ドキュメント)DynamoDB
- 「IAM ポリシーの作成」(IAM ドキュメント)
- 「クロスアカウントポリシーの評価ロジック」(IAM ドキュメント)
- 「IAM JSON ポリシー要素のリファレンス」(IAM ドキュメント)

追加情報

「サンプルコード」

```
import boto3
from datetime import datetime

dynamodb_client = boto3.client('dynamodb')
```

```
def lambda_handler(event, context):
     now = datetime.now().isoformat()
     data = dynamodb_client.put_item(TableName='arn:aws:dynamodb:<Region>:<Account-
A-ID>:table/Table-Account-A', Item={"category": {"S": "Fruit"},"item": {"S":
 "Apple"},"time": {"S": now}})
     return data
```

> **Note**
>
> DynamoDB クライアントがインスタンス化されると、テーブル名の代わりに DynamoDB テーブルの ARN が提供されます。これは、Lambda 関数が実行時に正しい DynamoDB テーブルに接続するために必要です。

AWS 上の SQL Server の Always On アベイラビリティグループで読み取り専用ルーティングを構成する

作成者: Subhani Shaik (AWS)

概要

このパターンでは、読み取り専用ワークロードをプライマリレプリカからセカンダリレプリカにオフロードすることで、SQL Server Always On のスタンバイセカンダリレプリカを使用する方法について説明します。

データベースミラーリングには 1 対 1 のマッピングがあります。セカンダリデータベースを直接読み取ることはできないため、スナップショットを作成する必要があります。Always On アベイラビリティグループ機能は Microsoft SQL Server 2012 で導入されました。それ以降のバージョンでは、読み取り専用ルーティングなどの主要な機能が導入されています。Always On アベイラビリティグループでは、レプリカモードを読み取り専用に変更することで、セカンダリレプリカからデータを直接読み取ることができます。

Always On アベイラビリティグループソリューションは、高アベイラビリティ (HA)、ディザスタリカバリ (DR)、およびデータベースミラーリングの代替手段をサポートします。Always On アベイラビリティグループはデータベースレベルで機能し、一連のユーザーデータベースのアベイラビリティを最大化します。

SQL Server は読み取り専用ルーティングメカニズムを使用して、受信した読み取り専用接続をセカンダリ読み取りレプリカにリダイレクトします。そのためには、接続文字列に次のパラメータと値を追加する必要があります。

- `ApplicationIntent=ReadOnly`
- `Initial Catalog=<database name>`

前提条件と制限

前提条件

- 仮想プライベートクラウド (VPC)、2つのアベイラビリティーゾーン、プライベートサブネット、およびセキュリティグループを使用するアクティブな AWS アカウント
- SQL Server 2019 Enterprise Edition Amazon Machine Image を搭載し、インスタンスレベルで構成された Windows サーバーフェイルオーバークラスター (WSFC) 機能、AWS Directory Service for Microsoft Active Directory の `tagechtalk.com` という名のディレクトリの一部であるプライ

マリノード (WSFCNODE1) とセカンダリノード (WSFCNODE2) 間のSQL Server レベルで構成された Always On アベイラビリティグループを備えた 2 台のAmazon Elastic Compute Cloud (Amazon EC2) マシン
- セカンダリレプリカで read-only を許可するように構成された 1 つ以上のノード
- Always On アベイラビリティグループに対して SQLAG1 と名付けられたリスナー
- 2 つのノードに対して同じサービスアカウントで実行されている SQL Server データベースエンジン
- SQL Server Management Studio (SSMS)
- test という名のテストデータベース

製品バージョン

- SQL Server 2014 およびそれ以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EC2
- AWS Managed Microsoft AD
- Amazon FSx

ターゲットアーキテクチャ

次の図は、Always On アベイラビリティグループ (AG) リスナーが、接続内に ApplicationIntent パラメータを含むクエリを適切なセカンダリノードにリダイレクトする方法を示しています。

1. Always On アベイラビリティグループリスナーにリクエストが送信されます。
2. 接続文字列に ApplicationIntent パラメータがない場合、リクエストはプライマリインスタンスに送信されます。
3. 接続文字列に ApplicationIntent=ReadOnly が含まれる場合、リクエストは読み取り専用ルーティング構成でセカンダリインスタンス、つまり Always On アベイラビリティグループの WSFC に送信されます。

ツール

サービス

- AWS Directory Service for Microsoft Active Directory により、ディレクトリ対応型ワークロードと AWS リソースが、AWS クラウドの Microsoft Active Directory を使用できるようになります。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon FSx は、業界標準の接続プロトコルをサポートし、AWS リージョン全体で高いアベイラビリティとレプリケーションを提供するファイルシステムを提供します。

その他のサービス

- SQL Server Management Studio (SSMS) は、SQL Server インスタンスを接続、管理、および管理するためのツールです。
- sqlcmd はコマンドラインユーティリティです。

ベストプラクティス

Always On アベイラビリティグループの詳細については、SQL Server ドキュメントを参照してください。

エピック

読み取り専用ルーティングの設定

タスク	説明	必要なスキル
レプリカを読み取り専用に更新します。	プライマリレプリカとセカンダリレプリカの両方を読み取り専用に更新するには、SSMS からプライマリレプリカに接続し、「追加情報」セクションのステップ 1 コードを実行します。	DBA

タスク	説明	必要なスキル
ルーティング URL を作成する。	両方のレプリカのルーティング URL を作成するには、「追加情報」セクションのステップ 2 コードを実行します。このコードでは、`tagechtalk.com` は AWS Managed Microsoft AD ディレクトリの名前です。	DBA
ルーティングリストを作成する。	両方のレプリカのルーティングリストを作成するには、「追加情報」セクションのステップ 3 コードを実行します。	DBA
ルーティングリストを検証します。	SQL Server Management Studio からプライマリインスタンスに接続し、「追加情報」セクションのステップ 4 コードを実行してルーティングリストを検証します。	DBA

読み取り専用ルーティングのテスト

タスク	説明	必要なスキル
ApplicationIntent パラメータを使用して接続します。	1. SSMS から、Always On アベイラビリティグループの `ApplicationIntent=ReadOnly;Initial Catalog=test` という名のリスナーに接続します。 2. セカンダリレプリカとの接続が確立されます。これを	DBA

タスク	説明	必要なスキル
	テストするには、次のコマンドを実行して接続されているサーバー名を表示します。 `SELECT SERVERPROPERTY('ComputerNamePhysicalNetBios')` 出力には現在のセカンダリレプリカ名 (WSFCNODE2) が表示されます。	

タスク	説明	必要なスキル
フェイルオーバーを実行する。	1. SSMS から Always On アベイラビリティグループのリスナー名に接続します。 2. プライマリデータベースとセカンダリデータベースが同期していて、データが失われていないことを確認します。 3. 現在のプライマリレプリカがセカンダリレプリカになり、セカンダリレプリカがプライマリレプリカになるようにフェイルオーバーを実行します。 4. SSMS から、Always On アベイラビリティグループの `ApplicationIntent=ReadOnly;Initial Catalog=test` という名のリスナーに接続します。 5. セカンダリレプリカとの接続が確立されます。これをテストするには、次のコマンドを実行して接続されているサーバー名を表示します。 `SELECT SERVERPROPERTY('ComputerNamePhysicalNetBios')` 現在のセカンダリレプリカ名 (`WSFCNODE1`) が表示されます。	DBA

sqlcmd コマンドラインユーティリティを使用して接続する

タスク	説明	必要なスキル
sqlcmd を使用して接続します。	sqlcmd から接続するには、コマンドプロンプトの [追加情報] セクションからステップ 5 のコードを実行します。接続後、次のコマンドを実行して、接続されているサーバー名を表示します。 `SELECT SERVERPROPERTY('ComputerNamePhysicalNetBios') .` 出力には現在のセカンダリレプリカ名 (`WSFCNODE1 `) が表示されます。	DBA

トラブルシューティング

問題	ソリューション
リスナーの作成が失敗し、「WSFC クラスターはネットワーク名リソースをオンラインにできませんでした」というメッセージが表示された。	詳細については、Microsoft のブログ記事「リスナーの作成が失敗し、「WSFCクラスターはネットワーク名リソースをオンラインにできませんでした」というメッセージが表示される」を参照してください。
他のリスナーの問題やネットワークアクセスの問題などの潜在的な問題。	Microsoft ドキュメントの「Always On アベイラビリティグループ構成のトラブルシューティング (SQL Server)」を参照してください。

関連リソース

- Always On アベイラビリティグループの読み取り専用ルーティングの構成
- Always On アベイラビリティグループ構成のトラブルシューティング (SQL Server)

追加情報

ステップ 1. レプリカを読み取り専用に更新する

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH (SECONDARY_ROLE
 (ALLOW_CONNECTIONS = READ_ONLY))
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (SECONDARY_ROLE
 (ALLOW_CONNECTIONS = READ_ONLY))
GO
```

ステップ 2. ルーティング URL を作成する

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH (SECONDARY_ROLE
 (READ_ONLY_ROUTING_URL = N'TCP://WSFCNode1.tagechtalk.com:1433'))
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (SECONDARY_ROLE
 (READ_ONLY_ROUTING_URL = N'TCP://WSFCNode2.tagechtalk.com:1433'))
GO
```

ステップ 3 ルーティングリストを作成する

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH
 (PRIMARY_ROLE(READ_ONLY_ROUTING_LIST=('WSFCNODE2','WSFCNODE1')));
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (PRIMARY_ROLE
 (READ_ONLY_ROUTING_LIST=('WSFCNODE1','WSFCNODE2')));
GO
```

ステップ 4。ルーティングリストを検証する

```
SELECT AGSrc.replica_server_name AS PrimaryReplica, AGRepl.replica_server_name AS
 ReadOnlyReplica, AGRepl.read_only_routing_url AS RoutingURL , AGRL.routing_priority
 AS RoutingPriority FROM sys.availability_read_only_routing_lists AGRL INNER JOIN
 sys.availability_replicas AGSrc ON AGRL.replica_id = AGSrc.replica_id INNER JOIN
```

```
sys.availability_replicas AGRepl ON AGRL.read_only_replica_id = AGRepl.replica_id
INNER JOIN sys.availability_groups AV ON AV.group_id = AGSrc.group_id ORDER BY
PrimaryReplica
```

ステップ 5. SQL コマンドユーティリティ

```
sqlcmd -S SQLAG1,1433 -E -d test -K ReadOnly
```

pgAdmin の SSH トンネルを使用して接続

作成者: Jeevan Shetty (AWS) と Bhanu Ganesh Gudivada (AWS)

概要

セキュリティの原因で、データベースをプライベートサブネットに配置することは常に良いことです。データベースに対するクエリは、Amazon Web Services (AWS) クラウドのパブリックサブネットにある Amazon Elastic Compute Cloud (Amazon EC2) 要塞ホストを介して接続することにより実行できます。これには、開発者やデータベース管理者が一般的に使用する pgAdmin や DBeaver などのソフトウェアを Amazon EC2 ホストにインストールする必要があります。

Linux サーバーで pgAdmin を実行し、ウェブブラウザーからアクセスするには、追加の依存関係のインストール、権限の設定、および構成が必要です。

代替のソリューションとして、開発者またはデータベース管理者は pgAdmin を使用して PostgreSQL データベースに接続し、ローカルシステムから SSH トンネルを有効化します。このアプローチでは、pgAdmin はデータベースに接続する前に、パブリックサブネットの Amazon EC2 ホストを仲介ホストとして使用します。アーキテクチャセクションの図表は、セットアップを示します。

> **Note**
>
> PostgreSQL データベースにアタッチされたセキュリティグループが、Amazon EC2 ホストからのポート 5432 で接続を許可していることを確認します。

前提条件と制限

前提条件

- 既存の AWS アカウント
- パブリックサブネットとプライベートサブネットを備えた、仮想プライベートクラウド (VPC)
- セキュリティグループがアタッチされた EC2 インスタンス
- セキュリティグループがアタッチされた Amazon Aurora PostgreSQL 互換エディションデータベース
- トンネルをセットアップするための Secure Shell (SSH) キーペア

製品バージョン

- pgAdmin バージョン 6.2+
- Amazon Aurora PostgreSQL 互換エディションバージョン 12.7+

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EC2
- Amazon Aurora PostgreSQL 互換

ターゲットアーキテクチャ

次の図表では、pgAdmin と SSH トンネルを使用して、インターネットゲートウェイ経由で EC2 インスタンスに接続し、そのEC2 インスタンスがデータベースに接続する方法を示しています。

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

その他のサービス

- 「pgAdmin」は、PostgreSQLのオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

エピック

接続を作成するには

タスク	説明	必要なスキル
サーバーを作成	pgAdmin で作成を選択し、次にサーバーを選択します。サーバーを登録し、接続を設定し、サーバーダイアログを使用して SSH トンネリング経由で接続する pgAdmin の設定に関するその他のヘルプについては、「関連リソース」セクションのリンクを参照してください。	DBA
サーバーの名前を入力します。	[一般] タブで、名前を入力します。	DBA
データベースの詳細を入力します。	[接続タブ] で、次の値を入力します： • ホスト名/アドレス • [ポート] • データベースのメンテナンス • ユーザー名 • [パスワード]	DBA
Amazon EC2 サーバーの詳細を入力します。	[SSH トンネル] タブで、パブリックサブネットにある Amazon EC2 インスタンスの詳細を指定します。 • 「SSH トンネリング」を Yes に設定して、指定されたサーバーに接続する場	DBA

タスク	説明	必要なスキル
	合、pgAdmin が SSH トンネリングを使用するよう指定します。 • 「トンネルホスト」フィールドに、SSH ホストの名前または IP アドレス (10.x.x.x など) を指定します。 • 「トンネルポート」フィールドで、SSH ホストのポート (22 など) を指定します。 • 「ユーザー名」 フィールドに、SSH ホストのログイン権限を持つユーザーの名前 (例: ec2-user) を指定します。 • 認証の種類を「アイデンティティファイル」として指定し、pgAdmin が接続時にプライベートキーファイルを使用するようにします。 • プライバシー強化メール (PEM) ファイルの場所を「 アイデンティティファイル」フィールドに含めます。.pem ファイルは、Amazon EC2 のキーペアです。	
保存して接続します。	[保存] を選択してセットアップを完了し、次に SSH トンネルを使用して Aurora PostgreSQL 互換データベースに接続します。	DBA

関連リソース

- 「サーバーダイアログ」
- 「DNS サーバーに接続する」

JSON Oracleクエリを PostgreSQL データベース SQL に変換

ピネシュ・シンガル (AWS) とロケシュ・グラム (AWS) によって作成された

概要

オンプレミスからAmazon Web Services (AWS) クラウドに移行するためのこの移行プロセスでは、AWS Schema Conversion Tool (AWS SCT) を使用して Oracle データベースのコードを PostgreSQL データベースに変換します。ほとんどのコードは AWS SCT によって自動的に変換されます。ただし、JSON 関連の Oracle クエリは自動的に変換されません。

Oracle 12.2 バージョン以降、Oracle Database は JSON ベースのデータを行ベースのデータに変換するのに役立つさまざまな JSON 関数をサポートしています。ただし、AWS SCT は JSON ベースのデータを PostgreSQL でサポートされている言語に自動的に変換しません。

この移行パターンは、主に、`JSON_OBJECT`、`JSON_ARRAYAGG`、`JSON_TABLE`などの関数を使用する JSON 関連の Oracle クエリを Oracle データベースから PostgreSQL データベースに手動で変換することに重点を置いています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- オンプレミスの Oracle データベースインスタンス (稼働中)
- PostgreSQL または Amazon Aurora PostgreSQL 互換エディションデータベースインスタンス (稼働中) の Amazon Relational Database Service (Amazon RDS)

機能制限

- JSON 関連のクエリには、固定の`KEY`と`VALUE`形式が必要です。この形式を使用しないと、間違った結果が返されます。
- JSON構造の変更により、結果セクションに新しい`KEY`と`VALUE`のペアが追加された場合、SQLクエリで対応するプロシージャまたは関数を変更する必要があります。
- JSON 関連の関数の中には、以前のバージョンの Oracle と PostgreSQL でサポートされていますが、機能が少ないものもあります。

製品バージョン

- インメモリデータベース (バージョン 12.1 以降)
- Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換バージョン 9.5 以降
- AWS SCT 最新バージョン (バージョン 1.0.664 を使用してテスト済み)

アーキテクチャ

ソーステクノロジースタック

- バージョン 19c の Oracle データベースインスタンス

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換データベースインスタンス (バージョン 13)

ターゲットアーキテクチャ

1. AWS SCT と JSON 関数コードを使用して、ソースコードを Oracle から PostgreSQL に変換します。
2. この変換により、PostgreSQL がサポートするマイグレーションされた.sql ファイルが生成されます。
3. 変換されていない Oracle JSON ファンクションコードを PostgreSQL JSON ファンクションコードに手動で変換します。
4. ターゲット Aurora PostgreSQL 互換の DB インスタンスで.sql ファイルを実行します。

ツール

AWS サービス

- Amazon Aurora はフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。

- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。

その他のサービス

- 「Oracle SQL Developer」は、従来のデプロイとクラウドベースのデプロイの両方で Oracle データベースの開発と管理を簡素化する統合開発環境です。
- pgAdminまたはDBeaver。「pgAdmin」はPostgreSQL用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。「DBeaver」は汎用データベースツールです。

ベストプラクティス

Oracle· クエリでは、このJSON_TABLE関数を使用する場合、デフォルトでCASTタイプが使用されます。ベストプラクティスは、PostgreSQLでもCASTを使用し、二重の大なり文字(>>)を使用することです。

詳細については、「追加情報」セクションの「Postgres_SQL_read_JSON」を参照してください。

エピック

Oracle データベースと PostgreSQL データベースに JSON データを生成します。

タスク	説明	必要なスキル
JSON データを Oracle データベースに保存します。	Oracle データベースにテーブルを作成し、そのCLOB列に JSON データを格納します。「追加情報」セクションにある「Oracle_Table_Creation_Insert_Script」を使用してください。	移行エンジニア
JSON データを PostgreSQL データベースに保存します。	PostgreSQL データベースにテーブルを作成し、そのTEXT列に JSON データを格納します。「追加情報」セ	移行エンジニア

タスク	説明	必要なスキル
	クションにある「Postgres_Table_Creation_Insert_Script」を使用してください。	

JSON を行形式に変換します。

タスク	説明	必要なスキル
Oracle データベースの JSON データを変換します。	JSON データを行フォーマットに読み込むための Oracle SQL クエリを記述します。詳細と構文例については、「追加情報」セクションの「Oracle_SQL_READ_JSON」を参照してください。	移行エンジニア
PostgreSQL データベース上の JSON データを変換します。	JSON データを ROW フォーマットに読み込むための PostgreSQL クエリを記述します。詳細と構文例については、「追加情報」セクションの「Postgres_SQL_Read_JSON」を参照してください。	移行エンジニア

SQL クエリを使用して JSON データを手動で変換し、JSON 形式で出力を報告します。

タスク	説明	必要なスキル
Oracle SQL クエリの集計と検証を行います。	JSON データを手動で変換するには、Oracle SQL クエリで結合、集約、検証を実行し、出力を JSON 形式でレポートします。「追加情報」セクションの「Oracle_SQ	移行エンジニア

タスク	説明	必要なスキル
	L_JSON_Aggregation_Join」にあるコードを使用してください。 1. JOIN — JSON 形式のデータが入力パラメータとしてクエリに渡されます。この静的データと Oracle DB テーブル`aws_test_table` 内の JSON データとの間で内部結合が行われます。 2. 検証を伴う集約 — JSON データには、`SUM`と`COUNT`集計に使用される、`accountNumber` 、`parentAccountNumber` 、`businessUnitId` , `positionId` などの値を持つ`KEY`と`VALUE`パラメータがあります。 3. JSON 形式 — 結合と集計の後、`JSON_OBJECT` および`JSON_ARRAYAGG` を使用して JSON 形式でデータがレポートされます。	

タスク	説明	必要なスキル
Postgres SQL クエリの集計と検証を行います。	JSON データを手動で変換するには、PostgreSQL クエリで結合、集約、検証を実行し、出力を JSON 形式でレポートします。「追加情報」セクションの「Postgres_SQL_JSON_Aggregation_Join」にあるコードを使用してください。 1. JOIN — JSON 形式のデータ (tab1) が入力パラメータとしてWITH句クエリに渡されます。この静的データとtabテーブル内の JSON データとの間で JOIN が行われます。aws_test_pg_table テーブルに JSON データを含むWITH句を使用して JOIN も行われます。 2. 集約 — JSON データには、SUMとCOUNTの集計に使用される、accountNumber 、parentAccountNumber 、businessUnitId 、positionId などの値を持つKEYとVALUEパラメータがあります。 3. JSON 形式 — 結合と集計の後、JSON_BUILD_OBJECT およ	移行エンジニア

タスク	説明	必要なスキル
	びJSON_AGGを使用して JSON 形式でデータがレポートされます。	

Oracle プロシージャを JSON クエリを含む PostgreSQL 関数に変換します。

タスク	説明	必要なスキル
Oracle プロシージャ内の JSON クエリを行に変換します。	サンプル Oracle プロシージャでは、前述の Oracle クエリと、「追加情報」セクションの「oracle_Procedure_With_Json_Query」にあるコードを使用してください。	移行エンジニア
JSON クエリを含む PostgreSQL 関数を行ベースのデータに変換します。	PostgreSQL 関数の例については、以前の PostgreSQL クエリと、「追加情報」セクションの「Postgres_function_with_Json_Query」にあるコードを使用してください。	移行エンジニア

関連リソース

- Oracle JSON 関数
- PostgreSQL JSON 関数
- Oracle JSON 関数の例
- PostgreSQL JSON 関数の例
- AWS Schema Conversion Tool

追加情報

JSON コードを Oracle データベースから PostgreSQL データベースに変換するには、次のスクリプトを順番に使用してください。

1. Oracle_Table_Creation_Insert_Script

```
create table aws_test_table(id number,created_on date default sysdate,modified_on
 date,json_doc clob);

REM INSERTING into EXPORT_TABLE
SET DEFINE OFF;
Insert into aws_test_table (ID,CREATED_ON,MODIFIED_ON,json_doc)
values (1,to_date('02-AUG-2022 12:30:14','DD-MON-YYYY HH24:MI:SS'),to_date('02-AUG-2022
 12:30:14','DD-MON-YYYY HH24:MI:SS'),TO_CLOB(q'[{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0100",
    "arrayPattern" : " -'",
    "a]')
|| TO_CLOB(q'[ccount" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
      "accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
```

```
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
        "id" : "0000000046",
]')
|| TO_CLOB(q'[        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}]'));
Insert into aws_test_table (ID,CREATED_ON,MODIFIED_ON,json_doc) values (2,to_date('02-
AUG-2022 12:30:14','DD-MON-YYYY HH24:MI:SS'),to_date('02-AUG-2022 12:30:14','DD-MON-
YYYY HH24:MI:SS'),TO_CLOB(q'[{
  "metadata" : {
    "upperLastNameFirstName" : "PQR XYZ",
    "upperEmailAddress" : "pqr@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "54534343",
    "displayName" : "Xyz, pqr",
    "firstName" : "pqr",
    "lastName" : "Xyz",
    "emailAddress" : "pqr@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0090",
    "arrayPattern" : " -'",
    "account" : {
      "companyId" : "CARS",
      "busin]')
|| TO_CLOB(q'[essUnitId" : 6,
      "accountNumber" : 42001,
      "parentAccountNumber" : 32001,
      "firstName" : "terry",
      "lastName" : "whitlock",
      "street1" : "UO  123",
      "city" : "TOTORON",
      "region" : "NO",
      "postalcode" : "LKM 111",
      "country" : "Canada"
    },
    "products" : [
```

```
      {
        "appUserGuid" : "ia744d7790000016899f8cf3f417d6df6",
        "id" : "0000000014",
        "name" : "ProView eLooseleaf",
      ]')
|| TO_CLOB(q'[  "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}]'));

commit;
```

2。Postgres_テーブル_作成_挿入_スクリプト

```
create table aws_test_pg_table(id int,created_on date ,modified_on date,json_doc text);
insert into aws_test_pg_table(id,created_on,modified_on,json_doc)
values(1,now(),now(),'{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0100",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
      "accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
```

```
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
        "id" : "0000000046",
        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}');


insert into aws_test_pg_table(id,created_on,modified_on,json_doc)
values(2,now(),now(),'{
  "metadata" : {
    "upperLastNameFirstName" : "PQR XYZ",
    "upperEmailAddress" : "pqr@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "54534343",
    "displayName" : "Xyz, pqr",
    "firstName" : "pqr",
    "lastName" : "Xyz",
    "emailAddress" : "a*b**@h**.k**",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0090",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "CARS",
      "businessUnitId" : 6,
      "accountNumber" : 42001,
      "parentAccountNumber" : 32001,
      "firstName" : "terry",
      "lastName" : "whitlock",
      "street1" : "UO  123",
      "city" : "TOTORON",
      "region" : "NO",
      "postalcode" : "LKM 111",
```

```
      "country" : "Canada"
    },
    "products" : [
      {
        "appUserGuid" : "ia744d7790000016899f8cf3f417d6df6",
        "id" : "0000000014",
        "name" : "ProView eLooseleaf",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}');
```

3. Oracle _SQL_Read_JSON

次のコードブロックは、Oracle JSON データを行形式に変換する方法を示しています。

クエリと構文の例

```
SELECT   JSON_OBJECT(
 'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               ) ) ) ) FROM
    (SELECT   tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
```

```
           SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
clerk_count
       FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
 COLUMNS (
         parent_account_number NUMBER PATH
          '$.data.account.parentAccountNumber',
           account_number NUMBER PATH '$.data.account.accountNumber',
           business_unit_id NUMBER PATH '$.data.account.businessUnitId',
           position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
           ) AS tab_data
           INNER JOIN JSON_TABLE ( '{
       "accounts": [{
         "accountNumber": 42000,
         "parentAccountNumber": 32000,
         "businessUnitId": 7
       }, {
         "accountNumber": 42001,
         "parentAccountNumber": 32001,
         "businessUnitId": 6
       }]
     }', '$.accounts[*]' ERROR ON ERROR
     COLUMNS (
     parent_account_number PATH '$.parentAccountNumber',
     account_number PATH '$.accountNumber',
     business_unit_id PATH '$.businessUnitId')
     ) static_data
     ON ( static_data.parent_account_number = tab_data.parent_account_number
          AND static_data.account_number = tab_data.account_number
          AND static_data.business_unit_id = tab_data.business_unit_id )
       GROUP BY
           tab_data.business_unit_id,
           tab_data.parent_account_number,
           tab_data.account_number );
```

JSON ドキュメントはデータをコレクションとして保存します。各コレクションはKEYとVALUEのペアを持つことができます。すべてのVALUEは、入れ子になっているKEYとVALUEのペアを持つことができます。以下の表は、JSON文書から特定のVALUEを読み取るための情報を提供します。

キー	値の取得に使用する階層またはパス	値
profileType	metadata -> profileType	P

positionId	data -> positionId	0100
accountNumber	data -> account -> accountNumber	42000

前の表では、KEYprofileTypeはmetadataKEYのVALUEで
す。KEYpositionIdはdataKEYのVALUEで
す。KEYaccountNumberはaccountKEYのVALUE、accountKEYはdataKEYのVALUEです。

JSON ドキュメントの例

```
{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
"profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
"positionId" : "0100",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
"accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
```

```
        "id" : "0000000046",
        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}
```

JSON ドキュメントから選択したフィールドを取得するために使用される SQL クエリ

```
select parent_account_number,account_number,business_unit_id,position_id from
 aws_test_table aws,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
COLUMNS (
parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
account_number NUMBER PATH '$.data.account.accountNumber',
business_unit_id NUMBER PATH '$.data.account.businessUnitId',
position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'
)) as sc
```

前のクエリでは、JSON_TABLEは JSON データを行形式に変換する Oracle の組み込み関数です。JSON_TABLE 関数には、JSON 形式のパラメータが必要です。

COLUMNSの各項目はあらかじめ定義されたPATHを持ち、そこで与えられたKEYに適切なVALUEが行の形式で返されます。

前のクエリの結果

親口座番号	口座番号	ビジネスユニット ID	ポジション ID
32000	42000	7	0100
32001	42001	6	0090

4。Postgres_sql_read_JSON

クエリと構文の例

```
select *
from (
```

```
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::VARCHAR as positionId
from aws_test_pg_table) d ;
```

Oracle では、PATHは特定のKEYおよびVALUEを識別するために使用されます。しかし、PostgreSQLはJSONからKEYとVALUEを読み取るためにHIERARCHYモデルを使用します。Oracle_SQL_Read_JSONで述べたのと同じJSONデータが、以下の例でも使われています。

CAST タイプの SQL クエリは使用できない

(強制的にCASTと入力すると、クエリは失敗し、構文エラーが発生します)。

```
select *
from (
select (json_doc::json->'data'->'account'->'parentAccountNumber') as
 parentAccountNumber,
(json_doc::json->'data'->'account'->'accountNumber')as accountNumber,
(json_doc::json->'data'->'account'->'businessUnitId') as businessUnitId,
(json_doc::json->'data'->'positionId')as positionId
from aws_test_pg_table) d ;
```

大なり演算子(>)を1つ使うと、そのKEYに定義されたVALUEが返されます。例： KEY: positionId、そして VALUE: "0100"

1 つの大なり演算子 (>) を使用する場合、CAST型は使用できません。

CAST 型の SQL クエリは許可されます

```
select *
from (
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar as positionId
from aws_test_pg_table) d ;
```

タイプCASTを使用するには、二重大なり演算子を使用する必要があります。1 つの大なり演算子を使用すると、クエリは定義されたVALUEを返します (例えばKEY：positionId、およ

びVALUE："0100") を返します。二重大なり演算子 (>>) を使用すると、そのKEYに定義されている実際の値(二重引用符なしのKEY：positionId、およびVALUE：0100など) が返されます。

前のケースでは、parentAccountNumberはINTにCASTと入力し、accountNumberはINTにCASTと入力し、businessUnitIdはINTにCASTと入力し、positionIdはVARCHARにCASTと入力します。

次の表は、1 つの大なり演算子 (>) と二重大なり演算子 (>>) の役割を説明するクエリ結果を示しています。

最初の表のクエリでは、1 つの大なり演算子 (>) を使用しています。各列は JSON 型で、別のデータ型に変換することはできません。

親アカウント番号	AccountNumber	ビジネスユニット ID	ポジション ID
2003565430	2003564830	7	0100
2005284042	2005284042	6	「0090」
2000272719	2000272719	1	0100

2 番目のテーブルでは、クエリは二重大なり演算子 (>>) を使用しています。各列は列の値に基づいてCAST型をサポートします。たとえば、この場合は INTEGER と指定します。

親アカウント番号	AccountNumber	ビジネスユニット ID	ポジション ID
2003565430	2003564830	7	0100
2005284042	2005284042	6	0090
2000272719	2000272719	1	0100

5。Oracle_SQL_JSON _Aggregation_Join

サンプルクエリ

```
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
```

```
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               ) ) ) )
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
 clerk_count
       FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
  COLUMNS (
         parent_account_number NUMBER PATH
          '$.data.account.parentAccountNumber',
           account_number NUMBER PATH '$.data.account.accountNumber',
           business_unit_id NUMBER PATH '$.data.account.businessUnitId',
           position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
           ) AS tab_data
           INNER JOIN JSON_TABLE ( '{
       "accounts": [{
         "accountNumber": 42000,
         "parentAccountNumber": 32000,
         "businessUnitId": 7
       }, {
         "accountNumber": 42001,
         "parentAccountNumber": 32001,
         "businessUnitId": 6
       }]
     }', '$.accounts[*]' ERROR ON ERROR
```

```
      COLUMNS (
      parent_account_number PATH '$.parentAccountNumber',
      account_number PATH '$.accountNumber',
      business_unit_id PATH '$.businessUnitId')
      ) static_data
      ON ( static_data.parent_account_number = tab_data.parent_account_number
           AND static_data.account_number = tab_data.account_number
           AND static_data.business_unit_id = tab_data.business_unit_id )
        GROUP BY
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number
    );
```

行レベルのデータを JSON 形式に変換するために、Oracle には、JSON_OBJECT、JSON_ARRAY、JSON_OBJECTAGG、JSON_ARRAYAGGなどの組み込み関数があります。

- JSON_OBJECTは、KEYとVALUEの 2 つのパラメータを受け入れます。KEYパラメータは本質的にハードコーディングされているか、静的である必要があります。VALUEパラメータはテーブル出力から導出されます。
- JSON_ARRAYAGGはJSON_OBJECTをパラメーターとして受け入れます。これにより、JSON_OBJECT要素のセットをリストとしてグループ化できます。たとえば、複数のレコード (データセット内の複数のKEYとVALUEペア) を持つJSON_OBJECT要素がある場合、JSON_ARRAYAGGはデータセットを追加してリストを作成します。データ構造言語によると、LISTは要素のグループです。このコンテキストでは、LISTはJSON_OBJECT要素のグループです。

次の例は、1 つのJSON_OBJECT要素を示しています。

```
{
   "taxProfessionalCount": 0,
   "attorneyCount": 0,
   "nonAttorneyCount": 1,
   "clerkCount": 0
}
```

次の例には、2 つのJSON_OBJECT要素が示され、LISTが角括弧 ([ ]) で示されています。

```
[
```

```
    {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
,
    {
        "taxProfessionalCount": 2,
        "attorneyCount": 1,
        "nonAttorneyCount": 3,
        "clerkCount":4
      }
]
```

SQL クエリの例

```
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               )
                        )
                                        )
              )
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE   0 END
            )      tax_count,
```

```
           SUM(CASE    WHEN tab_data.position_id = '0100' THEN        1    ELSE
 0 END
           )      attorney_count,

           SUM(CASE    WHEN tab_data.position_id = '0090' THEN        1    ELSE
 0 END
           )      non_attorney_count,

           SUM(CASE    WHEN tab_data.position_id = '0050' THEN        1    ELSE
 0 END
           )      clerk_count

       FROM
           aws_test_table scco,  JSON_TABLE ( json_doc, '$' ERROR ON ERROR
           COLUMNS (
           parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
           account_number NUMBER PATH '$.data.account.accountNumber',
           business_unit_id NUMBER PATH '$.data.account.businessUnitId',
           position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
           ) AS tab_data
           INNER JOIN JSON_TABLE ( '{
       "accounts": [{
         "accountNumber": 42000,
         "parentAccountNumber": 32000,
         "businessUnitId": 7
       }, {
         "accountNumber": 42001,
         "parentAccountNumber": 32001,
         "businessUnitId": 6
       }]
     }', '$.accounts[*]' ERROR ON ERROR
     COLUMNS (
     parent_account_number PATH '$.parentAccountNumber',
     account_number PATH '$.accountNumber',
     business_unit_id PATH '$.businessUnitId')
     ) static_data ON ( static_data.parent_account_number =
tab_data.parent_account_number
                        AND static_data.account_number = tab_data.account_number

                        AND static_data.business_unit_id =
tab_data.business_unit_id )
       GROUP BY
           tab_data.business_unit_id,
           tab_data.parent_account_number,
```

```
            tab_data.account_number
    );
```

前の SQL クエリの出力例

```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": 7,
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

6。 Postgres_SQL_JSON _Aggregation_Join

PostgreSQL の組み込み関数JSON_BUILD_OBJECTとJSON_AGGが、行レベルのデータを JSON 形式に変換します。 PostgreSQL JSON_BUILD_OBJECTおよびJSON_AGGはOracleJSON_OBJECTおよびJSON_ARRAYAGGと同等です。

サンプルクエリ

```
select
```

```
JSON_BUILD_OBJECT ('accountCounts',
    JSON_AGG(
        JSON_BUILD_OBJECT ('businessUnitId',businessUnitId
        ,'parentAccountNumber',parentAccountNumber
        ,'accountNumber',accountNumber
        ,'totalOnlineContactsCount',online_contacts_count,
        'countByPosition',
            JSON_BUILD_OBJECT (
            'taxProfessionalCount',tax_professional_count
            ,'attorneyCount',attorney_count
            ,'nonAttorneyCount',non_attorney_count
            ,'clerkCount',clerk_count
            )
        )
    )
)
from (
with tab as (select * from (
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar as positionId
from aws_test_pg_table) a ) ,
tab1 as ( select
(json_array_elements(b.jc -> 'accounts') ->> 'accountNumber')::integer accountNumber,
(json_array_elements(b.jc -> 'accounts') ->> 'businessUnitId')::integer
 businessUnitId,
(json_array_elements(b.jc -> 'accounts') ->> 'parentAccountNumber')::integer
 parentAccountNumber
from (
select '{
        "accounts": [{
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }, {
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }]
      }'::json as jc) b)
select
tab.businessUnitId::text,
```

```
tab.parentAccountNumber::text,
tab.accountNumber::text,
SUM(1) online_contacts_count,
SUM(CASE WHEN tab.positionId::text = '0095' THEN 1 ELSE 0  END)
  tax_professional_count,
SUM(CASE WHEN tab.positionId::text = '0100' THEN 1 ELSE 0  END)      attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0090' THEN      1  ELSE      0 END)
  non_attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0050' THEN      1  ELSE      0 END)
  clerk_count
from tab1,tab
where tab.parentAccountNumber::INTEGER=tab1.parentAccountNumber::INTEGER
and tab.accountNumber::INTEGER=tab1.accountNumber::INTEGER
and tab.businessUnitId::INTEGER=tab1.businessUnitId::INTEGER
GROUP BY      tab.businessUnitId::text,
            tab.parentAccountNumber::text,
            tab.accountNumber::text) a;
```

前述のクエリの出力例

OracleとPostgreSQLからの出力はまったく同じです。

```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": 7,
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
```

```
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

7. JSON クエリ付きの Oracle_Procedure_

このコードは Oracle プロシージャを JSON SQL クエリを含む PostgreSQL 関数に変換します。クエリが JSON を行に置き換える方法と、その逆の方法を示しています。

```
CREATE OR REPLACE PROCEDURE p_json_test(p_in_accounts_json IN varchar2,
 p_out_accunts_json  OUT varchar2)
IS
BEGIN
/*
p_in_accounts_json paramter should have following format:
       {
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }
*/
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
```

```
                        'clerkCount' VALUE clerk_count
                               ) ) ) )
into p_out_accunts_json
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
 clerk_count
        FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
            COLUMNS (
            parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
            account_number NUMBER PATH '$.data.account.accountNumber',
            business_unit_id NUMBER PATH '$.data.account.businessUnitId',
            position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
            ) AS tab_data
            INNER JOIN JSON_TABLE ( p_in_accounts_json, '$.accounts[*]' ERROR ON ERROR

      COLUMNS (
      parent_account_number PATH '$.parentAccountNumber',
      account_number PATH '$.accountNumber',
      business_unit_id PATH '$.businessUnitId')
      ) static_data
      ON ( static_data.parent_account_number = tab_data.parent_account_number
           AND static_data.account_number = tab_data.account_number
           AND static_data.business_unit_id = tab_data.business_unit_id )
        GROUP BY
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number
    );
EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Error while running the JSON query');
END;
/
```

プロシージャを実行する

次のコードブロックでは、以前に作成した Oracle プロシージャを、プロシージャへの JSON 入力例とともに実行する方法を説明します。また、このプロシージャの結果または出力も表示されます。

```
set serveroutput on;
declare
v_out varchar2(30000);
v_in varchar2(30000):= '{
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }';
begin
  p_json_test(v_in,v_out);
  dbms_output.put_line(v_out);
end;
/
```

プロシージャ出力

```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": 7,
```

```
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

8. JSON クエリ付きポストグレス関数

関数の例

```
CREATE OR REPLACE  FUNCTION f_pg_json_test(p_in_accounts_json  text)
RETURNS text
LANGUAGE plpgsql
AS
$$
DECLARE
 v_out_accunts_json   text;
BEGIN
SELECT
JSON_BUILD_OBJECT ('accountCounts',
    JSON_AGG(
        JSON_BUILD_OBJECT ('businessUnitId',businessUnitId
        ,'parentAccountNumber',parentAccountNumber
        ,'accountNumber',accountNumber
        ,'totalOnlineContactsCount',online_contacts_count,
        'countByPosition',
            JSON_BUILD_OBJECT (
            'taxProfessionalCount',tax_professional_count
            ,'attorneyCount',attorney_count
            ,'nonAttorneyCount',non_attorney_count
            ,'clerkCount',clerk_count
            ))))
INTO v_out_accunts_json
FROM (
WITH tab AS (SELECT * FROM (
SELECT (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER AS
 parentAccountNumber,
```

```
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER AS accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER AS businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar AS positionId
FROM aws_test_pg_table) a ) ,
tab1 AS ( SELECT
(json_array_elements(b.jc -> 'accounts') ->> 'accountNumber')::integer accountNumber,
(json_array_elements(b.jc -> 'accounts') ->> 'businessUnitId')::integer businessUnitId,
(json_array_elements(b.jc -> 'accounts') ->> 'parentAccountNumber')::integer
 parentAccountNumber
FROM (
SELECT p_in_accounts_json::json AS jc) b)
SELECT
tab.businessUnitId::text,
tab.parentAccountNumber::text,
tab.accountNumber::text,
SUM(1) online_contacts_count,
SUM(CASE WHEN tab.positionId::text = '0095' THEN 1 ELSE 0  END)
  tax_professional_count,
SUM(CASE WHEN tab.positionId::text = '0100' THEN 1 ELSE 0  END)      attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0090' THEN      1  ELSE      0 END)
  non_attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0050' THEN      1  ELSE      0 END)
  clerk_count
FROM tab1,tab
WHERE tab.parentAccountNumber::INTEGER=tab1.parentAccountNumber::INTEGER
AND tab.accountNumber::INTEGER=tab1.accountNumber::INTEGER
AND tab.businessUnitId::INTEGER=tab1.businessUnitId::INTEGER
GROUP BY      tab.businessUnitId::text,
            tab.parentAccountNumber::text,
            tab.accountNumber::text) a;
RETURN v_out_accunts_json;
END;
$$;
```

関数を実行する

```
select    f_pg_json_test('{
        "accounts": [{
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }, {
          "accountNumber": 42000,
```

```
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }]
      }')   ;
```

関数の出力

次の出力は、Oracle プロシージャ出力に似ています。違いは、この出力がテキスト形式であることです。

```
{
  "accountCounts": [
    {
      "businessUnitId": "6",
      "parentAccountNumber": "32001",
      "accountNumber": "42001",
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": "7",
      "parentAccountNumber": "32000",
      "accountNumber": "42000",
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

を使用してアカウント間で Amazon DynamoDB テーブルをコピーする AWS Backup

作成者：Ramkumar Ramanujam (AWS)

概要

で Amazon DynamoDB を使用する場合の一般的なユースケースは AWS、開発環境、テスト環境、またはステージング環境の DynamoDB テーブルを、本番環境のテーブルデータにコピーまたは同期することです。標準プラクティスとして、環境ごとに異なる を使用します AWS アカウント。

AWS Backup は、DynamoDB、Amazon Simple Storage Service (Amazon S3)、その他の のデータのクロスリージョンおよびクロスアカウントバックアップと復元をサポートします AWS のサービス。このパターンでは、 AWS Backup クロスアカウントバックアップと復元を使用して DynamoDB テーブルをコピーする手順を示します AWS アカウント。

前提条件と制限

前提条件

- の同じ組織に属する AWS アカウント 2 つのアクティブな AWS Organizations
- 両方のアカウントに DynamoDB テーブルを作成するアクセス許可
- AWS Identity and Access Management ボールトを作成および使用する (IAM) AWS Backup アクセス許可

機能制限

- ソースとターゲットは、 の同じ組織の一部 AWS アカウント である必要があります AWS Organizations。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Backup
- Amazon DynamoDB

ターゲット アーキテクチャ

1. ソースアカウントのバックアップボールトに DynamoDB テーブル AWS Backup バックアップを作成します。
2. バックアップをターゲットアカウントのバックアップ保管庫にコピーします。
3. ターゲットアカウントのバックアップボールトからのバックアップを使用して、ターゲットアカウントの DynamoDB テーブルを復元します。

自動化とスケール

を使用して AWS Backup 、バックアップを特定の間隔で実行するようにスケジュールできます。

ツール

- AWS Backup は、データ保護をクラウド、オンプレミス AWS のサービス、および 間で一元化および自動化するためのフルマネージドサービスです。このサービスを使用すると、バックアップポリシーを設定し、 AWS リソースのアクティビティを 1 か所でモニタリングできます。これにより、以前にサービスごとに実行されたバックアップタスクを自動化および統合でき、カスタムスクリプトや手動プロセスを作成する必要がなくなります。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。

エピック

ソースアカウントとターゲットアカウントの AWS Backup 機能を有効にする

タスク	説明	必要なスキル
DynamoDB およびクロスアカウントバックアップの高度な特徴量を有効にします。	ソースとターゲットの両方で AWS アカウント、次の操作を行います。 1. で AWS Management Console、 AWS Backup コンソールを開きます。 2. 設定 を選択します。	AWS DevOps、移行エンジニア

タスク	説明	必要なスキル
	3. Amazon DynamoDB バックアップの高度な特徴量で、高度な特徴量 が有効になっていることを確認するか、有効化を選択します。 4. クロスアカウント管理のクロスアカウントバックアップで有効化を選択します。	

ソースアカウントとターゲットアカウントにバックアップ保管庫を作成

タスク	説明	必要なスキル
バックアップ保管庫の作成	ソースとターゲットの両方で AWS アカウント、次の操作を行います。 1. AWS Backup コンソールで、バックアップボールトを選択します。 2. バックアップ保管庫を作成を選択します。 3. バックアップ保管庫の Amazon リソースネーム (ARN) をコピーして保存します。 ソースアカウントとターゲットアカウント間で DynamoDB テーブルバックアップをコピーする場合、ソースとターゲットの両方のバックアップ	AWS DevOps、移行エンジニア

タスク	説明	必要なスキル
	ボールトの ARNs が必要です。	

バックアップ保管庫を使用してバックアップと復元を行います。

タスク	説明	必要なスキル
ソースアカウントで、DynamoDB テーブルバックアップを作成します。	ソースアカウントの DynamoDB テーブルのバックアップを作成するには、以下を実行します。 1. AWS Backup ダッシュボードページで、オンデマンドバックアップの作成を選択します。 2. 設定セクションのリソースタイプで DynamoDB を選択し、テーブル名を選択します。 3. バックアップ保管庫 ドロップダウンリストで、ソースアカウントで作成したバックアップ保管庫を選択します。 4. 希望する保存期間 を選択します。 5. オンデマンドバックアップを作成 を選択します。 新しいバックアップジョブが作成されます。	AWS DevOps、移行エンジニア

タスク	説明	必要なスキル
	バックアップジョブのステータスをモニタリングするには、 AWS Backup ジョブページでバックアップジョブタブを選択します。すべてのアクティブ、進行中、完了したバックアップジョブがこのタブに表示されます。	

タスク	説明	必要なスキル
バックアップをソースアカウントからターゲットアカウントにコピーします。	バックアップジョブが完了したら、ソースアカウントのバックアップボールトからターゲットアカウントのバックアップボールトに DynamoDB テーブルのバックアップをコピーします。 バックアップ保管庫をソースアカウントにコピーするには、次の操作を行います。 1. AWS Backup コンソールで、バックアップボールトを選択します。 2. バックアップで、DynamoDB テーブルのバックアップを選択します。 3. アクション、コピーの順に選択します。 4. ターゲットアカウントの AWS リージョン を入力します。 5. 外部保管庫とARNに、ターゲットアカウントで作成したバックアップ保管庫のARNを入力します。 6. ソースアカウントからターゲットアカウントにバックアップをコピーするには、ターゲットアカウントのバックアップ保管庫で、別のアカウントからのアクセスを有効にします。	AWS DevOps、移行エンジニア

タスク	説明	必要なスキル
ターゲットアカウントのバックアップを復元します。	ターゲットで AWS アカウント、次の操作を行います。 1. AWS Backup コンソールで、バックアップボールトを選択します。 2. バックアップで、ソースアカウントからコピーしたバックアップを選択します。 3. アクションで、復旧を選択します。 4. 復元するターゲット DynamoDB テーブルの名前を入力します。	AWS DevOps、移行エンジニア

関連リソース

- DynamoDB AWS Backup での の使用
- でのバックアップコピーの作成 AWS アカウント
- AWS Backup 料金

カスタム実装を使用してアカウント間で Amazon DynamoDB テーブルをコピー

作成者：Ramkumar Ramanujam (AWS)

概要

Amazon Web Services (AWS) で Amazon DynamoDB を使用する場合、通常のユースケースは、開発、テスト、またはステージング環境の DynamoDB テーブルを本番環境のテーブルデータにコピーまたは同期することです。標準的な方法として、環境ごとに異なる AWS アカウントを使用します。

DynamoDB が AWS Backup を使用したクロスアカウントバックアップに適用されます。AWS Backup を使用する際に関連するストレージコストについては、AWS Backup 料金表を参照してください。AWS Backup を使用してアカウントをコピーする場合、ソースアカウントとターゲットアカウントは AWS Organizations 組織の一部である必要があります。AWS Glue などの AWS のサービスを使用したクロスアカウントバックアップと復元には、他にもソリューションがあります。ただし、これらのソリューションを使用すると、デプロイして管理する AWS サービスの数が増えるため、アプリケーションのフットプリントが増加します。

Amazon DynamoDB Streamsを使用して、ソースアカウントのテーブルの変更をキャプチャすることもできます。次に、AWS Lambda 関数を開始し、ターゲットアカウントのターゲットテーブルに対応する変更を加えることができます。ただし、このソリューションは、ソーステーブルとターゲットテーブルを常に同期させる必要があるユースケースにも当てはまります。データが頻繁に更新される開発、テスト、ステージング環境には適用されない場合があります。

このパターンでは、Amazon DynamoDB テーブルをあるアカウントから別のアカウントにコピーするカスタムソリューションを実装する手順を示します。このパターンでは、C#、Java、Python などの一般的なプログラミング言語を使用して実装できます。AWS SDK に適用されている言語を使用することを推薦します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 両方のアカウントの DynamoDB テーブル
- AWS Identity and Access Management (IAM) ロールとポリシー
- C#、Java、Python などの一般的なプログラミング言語を使用して Amazon DynamoDB テーブルにアクセスする方法に関する知識

機能制限

このパターンでは、約 2 GB 以下の DynamoDB テーブルに適用されます。接続やセッションの中断、調整、障害や再試行を処理するロジックを追加すれば、サイズの大きいテーブルにも使用できます。

ソーステーブルから項目を読み取る DynamoDB スキャンオペレーションは、1 回の呼び出しで最大 1 MB のデータしか取得できません。2 GB を超える大きなテーブルでは、この制限によりテーブル全体のコピーを実行する合計時間が長くなる可能性があります。

アーキテクチャ

次の図は、ソースとターゲットの AWS アカウント間のカスタム実装を示しています。IAM ポリシーとセキュリティトークンは、カスタム実装で使用されます。データはソースアカウントの Amazon DynamoDB から読み取られ、ターゲットアカウントの DynamoDB に書き込まれます。

自動化とスケール

このパターンでは、サイズが約 2 GB と小さい DynamoDB テーブルに適用されます。

このパターンを大きなテーブルに適用するには、以下の問題に対処します。

- テーブルコピー操作中は、異なるセキュリティトークンを使用して 2 つのアクティブセッションが維持されます。テーブルコピー操作にトークンの有効期限よりも時間がかかる場合は、セキュリティトークンを更新するロジックを設定する必要があります。
- 十分な読み込みキャパシティーユニット (WCU) と書き込みキャパシティーユニット (WCU) がプロビジョニングされていない場合、ソーステーブルまたはターゲットテーブルの読み込みまたは書き込みが制限される可能性があります。これらの例外を必ずキャッチして処理します。
- その他の障害や例外を処理し、再試行メカニズムを導入して、コピー操作が失敗したところから再試行または続行できるようにします。

ツール

ツール

- Amazon DynamoDB – Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。

- 必要な追加ツールは、実装に選択したプログラミング言語によって異なります。たとえば、C# を使用する場合、Microsoft Visual Studio と次の NuGet パッケージが必要になります。
  - AWSSDK
  - AWSSDK.DynamoDBv2

コード

次の Python コードスニペットは、Boto3 ライブラリを使用して DynamoDB テーブルを削除して再作成します。

IAM ユーザーの AWS_ACCESS_KEY_ID と AWS_SECRET_ACCESS_KEY は長期的な認証情報であるため、使用しません。プログラムで AWS のサービスにアクセスする場合は使用しません。一時的な認証情報についての詳細は、IAM ユーザーガイドを参照してください。

次のコードスニペットで使用されている AWS_ACCESS_KEY_ID、AWS_SECRET_ACCESS_KEY 、および TEMPORARY_SESSION_TOKEN は AWS Security Token Service (AWS STS) から取得された一時的な認証情報です。

```
import boto3
import sys
import json

#args = input-parameters = GLOBAL_SEC_INDEXES_JSON_COLLECTION,
 ATTRIBUTES_JSON_COLLECTION, TARGET_DYNAMODB_NAME, TARGET_REGION, ...

#Input param: GLOBAL_SEC_INDEXES_JSON_COLLECTION
#[{"IndexName":"Test-index","KeySchema":[{"AttributeName":"AppId","KeyType":"HASH"},
{"AttributeName":"AppType","KeyType":"RANGE"}],"Projection":
{"ProjectionType":"INCLUDE","NonKeyAttributes":["PK","SK","OwnerName","AppVersion"]}}]

#Input param: ATTRIBUTES_JSON_COLLECTION
#[{"AttributeName":"PK","AttributeType":"S"},
{"AttributeName":"SK","AttributeType":"S"},
{"AttributeName":"AppId","AttributeType":"S"},
{"AttributeName":"AppType","AttributeType":"N"}]

region = args['TARGET_REGION']
target_ddb_name = args['TARGET_DYNAMODB_NAME']

global_secondary_indexes = json.loads(args['GLOBAL_SEC_INDEXES_JSON_COLLECTION'])
attribute_definitions = json.loads(args['ATTRIBUTES_JSON_COLLECTION'])
```

```
# Drop and create target DynamoDB table
dynamodb_client = boto3.Session(
        aws_access_key_id=args['AWS_ACCESS_KEY_ID'],
        aws_secret_access_key=args['AWS_SECRET_ACCESS_KEY'],
        aws_session_token=args['TEMPORARY_SESSION_TOKEN'],
    ).client('dynamodb')

# Delete table
print('Deleting table: ' + target_ddb_name + ' ...')

try:
    dynamodb_client.delete_table(TableName=target_ddb_name)

    #Wait for table deletion to complete
    waiter = dynamodb_client.get_waiter('table_not_exists')
    waiter.wait(TableName=target_ddb_name)
    print('Table deleted.')
except dynamodb_client.exceptions.ResourceNotFoundException:
    print('Table already deleted / does not exist.')
    pass

print('Creating table: ' + target_ddb_name + ' ...')

table = dynamodb_client.create_table(
    TableName=target_ddb_name,
    KeySchema=[
        {
            'AttributeName': 'PK',
            'KeyType': 'HASH'  # Partition key
        },
        {
            'AttributeName': 'SK',
            'KeyType': 'RANGE'  # Sort key
        }
    ],
    AttributeDefinitions=attribute_definitions,
    GlobalSecondaryIndexes=global_secondary_indexes,
    BillingMode='PAY_PER_REQUEST'
)

waiter = dynamodb_client.get_waiter('table_exists')
waiter.wait(TableName=target_ddb_name)
```

```
print('Table created.')
```

ベストプラクティス

一時的な認証情報

セキュリティのベストプラクティスとして、プログラムで AWS のサービスにアクセスするときは、IAM ユーザーの AWS_ACCESS_KEY_IDと AWS_SECRET_ACCESS_KEY は長期的な認証情報であるため、使用しないでください。AWS サービスにプログラムからアクセスするには、常に一時的な認証情報を使用します。

例として、開発者は開発中にアプリケーションの IAM ユーザーの AWS_ACCESS_KEY_ID と AWS_SECRET_ACCESS_KEY をハードコーディングしますが、変更をコードリポジトリにプッシュする前にハードコードされた値の削除に失敗します。これらの公開された認証情報は、意図しないユーザーや悪意のあるユーザーによって使用される可能性があり、深刻な問題を引き起こす可能性があります (特に公開された認証情報に管理者権限がある場合)。これらの公開後、これらの認証情報を IAM コンソールまたは AWS コマンドラインインターフェイス (AWS CLI) を使用して、直ちに無効化または削除する必要があります。

AWS サービスにプログラムでアクセスするための一時的な認証情報を取得するには、AWS STS を使用します。一時的な認証情報は、指定された時間 (15 分から 36 時間まで) のみ有効です。一時的な認証情報の最大許容期間は、ロール設定やロールチェーンなどの要因によって異なります。AWS STS の詳細については、ドキュメントを参照してください。

エピック

DynamoDB テーブルのセットアップ

タスク	説明	必要なスキル
DynamoDB テーブルを作成します。	ソースとターゲットの両方の AWS アカウントで、インデックスを使用して DynamoDB テーブルを作成します。 キャパシティプロビジョニングをオンデマンドモードに設定します。これにより、DynamoDB はワークロー	アプリ開発者、DBA、移行エンジニア

タスク	説明	必要なスキル
	ドに基づいて読み取り/書き込みキャパシティーを動的にスケーリングできます。 あるいは、4000 の RCU と 4000 WCU のプロビジョニングされたキャパシティを使用することもできます。	
ソーステーブルにデータを入力します。	ソースアカウントの DynamoDB テーブルにテストデータを入力します。テストデータが 50 MB 以上あると、テーブルコピー中に消費される RCU の最大値と平均値を確認するのに役立ちます。その後、必要に応じてキャパシティプロビジョニングを変更できます。	アプリ開発者、DBA、移行エンジニア

DynamoDB テーブルにアクセスするための認証情報をセットアップ

タスク	説明	必要なスキル
ソースとターゲットの DynamoDB テーブルにアクセスするための IAM ロールを作成します。	ソースアカウントの DynamoDB テーブルにアクセス (読み取り) する権限を持つ IAM ロールをソースアカウントで作成します。 ソースアカウントをこのロールの信頼できるエンティティとして追加します。 ターゲットアカウントの DynamoDB テーブルへのアク	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	セス (作成、読み取り、更新、削除) 権限を持つ IAM ロールをターゲットアカウントで作成します。 ターゲットアカウントをこのロールの信頼できるエンティティとして追加します。	

アカウントから別のアカウントにテーブルデータをコピー

タスク	説明	必要なスキル
IAM ロールの一時認証情報を取得します。	ソースアカウントで作成した IAM ロールの一時認証情報を取得します。 ターゲットアカウントで作成された IAM ロールの一時認証情報を取得します。 IAM ロールの一時認証情報を取得する 1 つの方法は、AWS CLI から AWS STS を使用します。 `aws sts assume-role --role-arn arn:aws:iam::<account-id>:role/<role-name> --role-session-name <session-name> --profile <profile-name>` 適切な AWS プロファイル (ソースアカウントまたはター	アプリ開発者、移行エンジニア

タスク	説明	必要なスキル
	ゲットアカウントに対応) を使用します。 一時的なセキュリティ認証情報の使用方法の詳細については、以下を参照してください。 • AWS セキュリティトークンサービスAPI リファレンス • CLI アクセス用の IAM ロール認証情報を取得	
ソースとターゲットの DynamoDB アクセス用に DynamoDB クライアントを初期化します。	AWS SDK によって提供される DynamoDB クライアントを、ソースとターゲットの DynamoDB テーブル用に初期化します。 • ソース DynamoDB クライアントには、ソースアカウントから取得した一時的な認証情報を使用します。 • ターゲット DynamoDB クライアントには、ターゲットアカウントから取得した一時的な認証情報を使用します。 IAM の一時認証情報を使用してリクエストを行う方法についての詳細は、AWS ドキュメント を参照してください。	アプリ開発者

タスク	説明	必要なスキル
ターゲットテーブルをドロップして再作成。	ターゲットアカウントの DynamoDB クライアントを使用して、ターゲットアカウントのターゲット DynamoDB テーブル (およびインデックス) を削除して再作成します。 DynamoDB テーブルからすべてのレコードを削除すると、プロビジョニングされた WCU が消費されるため、コストのかかる操作です。テーブルを削除して再作成することで、このような追加コストを回避できます。 テーブルを作成した後でもインデックスを追加できますが、これには 2 ~ 5 分ほど時間がかかります。テーブルの作成中に インデックスコレクションを `createTable` 呼び出しに渡すことで、インデックスを作成することがより効率的になります。	アプリ開発者

タスク	説明	必要なスキル
テーブルコピーを実行します。	すべてのデータがコピーされるまで、次の手順を繰り返します。 • ソース DynamoDB クライアントを使用して、ソースアカウントのテーブルをスキャンします。DynamoDB スキャンではテーブルから 1 MB のデータしか取得されないため、すべての項目またはレコードが読み取られるまでこの操作を繰り返す必要があります。 • スキャンされた項目の各セットについて、DynamoDB の AWS SDK の `BatchWriteItem` の呼び出しを使用して、ターゲット DynamoDB クライアントを使用して、ターゲットアカウントのテーブルに項目を書き込みます。これにより DynamoDB への `PutItem` リクエストの数が減少されます。 • `BatchWriteItem` 書き込みまたは入力が 25 回、最大 16 MB に制限されています。`BatchWriteItem` を呼び出す前に、スキャンされた項目を 25 個ずつ累積するロジックを追加する必要があります。`BatchWrit`	アプリ開発者

タスク	説明	必要なスキル
	eItem が正常にコピーできなかった項目のリストを返します。このリストを使用して再試行ロジックを追加し、成功しなかった項目だけで、別の BatchWriteItem 呼び出しを実行します。 詳細については、「添付ファイル」セクションの C# のリファレンス実装 (テーブルの削除、作成、入力用) を参照してください。テーブル設定の JavaScript オブジェクト表記 (JSON) ファイルの例も添付されています。	

関連リソース

- Amazon DynamoDB ドキュメント
- AWS アカウント内での IAM ユーザーの作成
- AWS SDK
- AWS リソースを使用した一時的な認証情報の使用

追加情報

このパターンでは、C# を使用して 200,000 項目 (平均項目サイズは 5 KB、テーブルサイズは 250 MB) の DynamoDB テーブルをコピーするために実装されました。ターゲット DynamoDB テーブルは、4000 RCU と 4000 WCU のキャパシティーをプロビジョニングして設定されました。

テーブルの削除と再作成を含むテーブルコピー操作 (ソースアカウントからターゲットアカウントへ) には 5 分かかりました。消費されたキャパシティユニットの総容量：30,000 RCU と約 400,000 WCU。

DynamoDB キャパシティモードの詳細については、AWS ドキュメントの読み取り/書き込みキャパシティモード を参照してください。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon RDS と Amazon Aurora の詳細なコストと使用状況レポートを作成する

作成者:Lakshmanan Lakshmanan (AWS) and Sudarshan Narasimhan

概要

このパターンは、「ユーザー定義のコスト割り当てタグ」を設定して Amazon Relational Database Service (Amazon RDS) または Amazon Aurora クラスターの使用コストを追跡する方法を示しています。これらのタグを使用して、複数のディメンションにわたるクラスターの詳細なコストと使用状況レポートを AWS Cost Explorer で作成できます。たとえば、チーム、プロジェクト、またはコストセンターレベルで使用コストを追跡し、Amazon Athena でデータを分析できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 1 つ以上の「Amazon RDS」または「Amazon Aurora」インスタンス

制約事項

タグ付けの制限については、「AWS Billing ユーザーガイド」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon RDS または Amazon Aurora
- AWS コストと使用状況レポート
- AWS Cost Explorer
- Amazon Athena

ワークフローとアーキテクチャ

タグ付けと分析ワークフローは次のステップで構成されます。

1. データエンジニア、データベース管理者、または AWS 管理者は、Amazon RDS または Aurora クラスターのユーザー定義のコスト配分タグを作成します。

2. AWS 管理者がタグを有効化します。
3. タグはメタデータを AWS Cost Explorer に報告します。
4. データエンジニア、データベース管理者、または AWS 管理者が「毎月のコスト配分レポート」を作成します。
5. データエンジニア、データベース管理者、または AWS 管理者が Amazon Athena を使用して月次コスト配分レポートを分析します。

次の図は、Amazon RDS または Aurora インスタンスの使用コストを追跡するためにタグを適用する方法を示しています。

次のアーキテクチャ図は、コスト配分レポートを Amazon Athena と統合して分析する方法を示しています。

毎月のコスト配分レポートは、指定した Amazon S3 バケットに保存されます。エピックセクションで説明したように、AWS CloudFormation テンプレートを使用して Athena をセットアップすると、テンプレートは AWS Glue クローラー、AWS Glue データベース、Amazon Simple Notification System (Amazon SNS) イベント、AWS Lambda 関数、Lambda 関数 のための AWS Identity and Access Management (IAM) ロールなど、いくつかの追加リソースをプロビジョニングします。新しいコストデータファイルが S3 バケットに到着すると、イベント通知を使用してこれらのファイルを Lambda 関数に転送して処理します。Lambda 関数は AWS Glue クローラージョブを開始して、AWS Glue データカタログのテーブルを作成または更新します。次に、このテーブルを使用して Athena のデータをクエリします。

ツール

- 「Amazon Athena」は、Amazon S3 内のデータを標準 SQL を使用して簡単に分析できるインタラクティブなクエリサービスです。
- 「Amazon Aurora」はクラウド用に構築されたフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。

- 「AWS CloudFormation」は、AWS とサードパーティのリソースを簡単にモデル化、プロビジョニング、管理できるコードとしての infrastructure as code (IaC) サービスです。
- 「AWS Cost Explorer」を使用すると、AWS コストと使用状況を表示および分析できます。

エピック

Amazon RDS または Aurora クラスターのタグを作成してアクティブ化する

タスク	説明	必要なスキル
Amazon RDS または Aurora クラスター用のユーザー定義のコスト配分タグを作成します。	新規または既存の Amazon RDS または Aurora クラスターにタグを追加するには、[Amazon Aurora ユーザーガイド] の「タグの追加、一覧表示、削除」の手順に従ってください。 **Note** Amazon Aurora クラスターの設定方法については、「Amazon Aurora ユーザーガイド」のMySQL と PostgreSQL の手順」を参照してください。	AWS 管理者、データエンジニア、DBA
ユーザー定義のコスト配分タグを有効にします。	「AWS 請求ユーザーガイド」の「ユーザー定義のコスト配分タグの有効化」の手順に従ってください。	AWS 管理者

コストと使用状況レポートの作成

タスク	説明	必要なスキル
クラスターのコストと使用状況レポートを作成して設定する。	1. AWS マネジメントコンソールにサインインし、「AWS 請求コンソール」を開きます。 2. 左のナビゲーションペインで、[コストと使用状況レポート] を選択します。 3. [レポートを作成] を選択します。 4. レポート名を入力し、他のオプションについてはデフォルト設定のままにして、[次へ] を選択します。 5. [設定] を選択し、既存の S3 バケットの詳細を入力します。この画面から新しい S3 バケットを作成することもできます。[Next (次へ)] を選択します。 6. バケットに適用されるデフォルトポリシーを確認し、確認チェックボックスを選択して、[保存] を選択します。 7. レポートパスのプレフィックスに、レポート名に付加するプレフィックスを入力します。 8. [時間の細分化] では、レポート用にデータを収集する頻度に応じて、[毎時]、	アプリオーナー、AWS管理者、DBA、一般AWS、データエンジニア

タスク	説明	必要なスキル
	[毎日]、または [毎月] を選択します。 9. [レポートのバージョン] 管理では、レポートの新しいバージョンを個別に作成するか、既存のレポートを各バージョンで上書きするかを選択します。 10[レポートデータ統合の有効化] には、[Amazon Athena] を選択します。圧縮タイプが Parquet に設定されていることを確認します。 11[Next (次へ)] を選択します。 12.レポートの設定を確認したら、[確認して完了] を選択します。 データは 24 時間後に利用可能になります。	

コストと使用状況のレポートデータを分析します。

タスク	説明	必要なスキル
コストと使用状況のレポートデータを分析します。	1. Athena をセットアップして使用し、レポートデータを分析します。手順については、[AWS コストと使用状況レポートユーザーガイド] の「Amazon Athena を使用したコストと使用状況レ	アプリオーナー、AWS管理者、DBA、一般AWS、データエンジニア

タスク	説明	必要なスキル
	ポートクエリ」を参照してください。「Athena が提供する AWS CloudFormation テンプレート」を使用することをお勧めします。 2. Athena クエリの実行。たとえば、次の SQL クエリの実行により、データ更新のステータスを確認できます。 `select status from cost_and_usage_data_status` 詳細については、[Amazon Athena ユーザーガイド] の「Amazon Athena クエリの実行」を参照してください。 **Note** SQL クエリを実行するときは、ドロップダウンリストから正しいデータベースが選択されていることを確認してください。	

関連リソース

リファレンス

- 「AWS CloudFormation テンプレートを使用してAthena をセットアップする」(推奨)

- 「Athena の手動セットアップ」
- 「Amazon Athena クエリの実行」
- 「他のリソースへのレポートデータのロード」

チュートリアルと動画

- 「Amazon Athena を使用したコストと使用状況レポートの分析」(YouTube ビデオ)

Aurora PostgreSQL のカスタムエンドポイントを使用して Oracle RAC ワークロードをエミュレートします

ハリクリシュナ・ボーガッダ (AWS) によって作成されました

概要

このパターンでは、単一クラスター内のインスタンスにワークロードを分散するカスタムエンドポイントを備えた Amazon Aurora PostgreSQL 互換エディションを使用して Oracle Real Application Clusters (Oracle RAC) ワークロードのサービスをエミュレートする方法を説明します。このパターンは、Amazon Aurora データベース用の 「カスタムエンドポイント」を作成する方法を示しています。カスタムエンドポイントを使用すると、Aurora クラスター内のさまざまな DB インスタンスセットにワークロードを分散し、負荷を分散できます。

Oracle RAC 環境では、「サービス」が 1 つ以上のインスタンスにまたがるため、トランザクションパフォーマンスに基づくワークロードバランシングが容易になります。サービスの特徴量には、エンドツーエンドの無人リカバリ、ワークロードごとのローリング変更、および場所の完全な透明性などがあります。このパターンを使用して、これらの特徴量の一部をエミュレートできます。たとえば、レポートアプリケーションの接続をルーティングする機能をエミュレートできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 「PostgreSQL」 JDBC ドライバー
- 「Aurora PostgreSQL 互換データベース」
- Oracle RAC データベースが Aurora PostgreSQL 互換データベースに移行されました

制約事項

- カスタムエンドポイントに適用される制限については、Amazon RDS ドキュメントの 「カスタムエンドポイントのプロパティの指定」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- 3 ノードの Oracle RAC データベース

ターゲットテクノロジースタック

- 2 つのリードレプリカを備えた Aurora PostgreSQL 互換データベース

ソースアーキテクチャ

次の図は、3 ノード Oracle RAC データベースのアーキテクチャを示します。

ターゲット アーキテクチャ

次の図は、2 つのリードレプリカを含む Aurora PostgreSQL 互換データベースのアーキテクチャを示します。3 つの異なるアプリケーション/サービスがカスタムエンドポイントを使用しており、これらは異なるアプリケーションユーザーにサービスを提供し、プライマリレプリカとリードレプリカ間でトラフィックと負荷をリダイレクトします。

ツール

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。
- 「Amazon CloudWatch」は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムに監視します。
- 「Amazon Relational Database Service (Amazon RDS)」を使用して、AWS クラウドでの PostgreSQL リレーショナルデータベースをセットアップ、運用、スケーリングできます。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

エピック

Aurora PostgreSQL 互換クラスターを作成する

タスク	説明	必要なスキル
クラスターを作成する。	クラスターを作成するには、Amazon RDS ドキュメントの「DB クラスターを作成して Aurora PostgreSQL DB クラスターのデータベースに接続する」を参照してください。	AWS 管理者
ワークロードのカスタムパラメータグループを作成します。	パラメータグループを作成するには、Amazon RDS ドキュメントの「DB クラスターパラメータグループの作成」を参照してください。	AWS 管理者
イベント通知とアラームを作成します。	イベント通知と Amazon CloudWatch アラームを使用して、クラスターの状態が変化したときに通知したり、事前定義されたしきい値に達したときにメトリックスをキャプチャしたりできます。 CloudWatch アラームを作成するには、「静的しきい値に基づいて CloudWatch アラームを作成する」を参照してください。 イベント通知を作成するには、「イベントでトリガーする CloudWatch イベントルー	AWS 管理者

タスク	説明	必要なスキル
	ルの作成」を参照してください。	

レプリカを Aurora PostgreSQL 互換の DB クラスターに追加する

タスク	説明	必要なスキル
クラスターにリードレプリカを追加します。	1. 「リードレプリカの作成」します。 2. **Note** DB クラスターと同じアベイラビリティーゾーンにリードレプリカを追加します。: フェイルオーバーノードで満たす必要がある要件がある場合は、別のアベイラビリティーゾーンを使用できます。	AWS 管理者
リードレプリカのエンドポイントをメモしておきます。	リードレプリカのエンドポイントを文書化して、後でカスタムエンドポイントを作成する際に使用できるようにします。	AWS 管理者

カスタムエンドポイントの作成

タスク	説明	必要なスキル
カスタムエンドポイントの名前を入力します。	必要なエンドポイントごとに、ワークロードまたはアプリケーションに関連する一意のエンドポイント名を作成します。	AWS 管理者
エンドポイントメンバーを追加します。	リードレプリカエンドポイントをカスタムグループに追加します。詳細については、Amazon RDS ドキュメントの「カスタムエンドポイントの編集」を参照してください。	AWS 管理者
(オプション) future インスタンスをクラスターに追加します。	カスタムグループにさらにレプリカまたはエンドポイントを追加する場合は、Amazon RDS ドキュメントの「Aurora レプリカを DB クラスターに追加する」を参照してください。	AWS 管理者
エンドポイントを作成します。	エンドポイントを作成するには、Amazon RDS ドキュメントの「カスタムエンドポイントの作成」を参照してください。	AWS 管理者

カスタムエンドポイントを使用してアプリケーション接続をテストします。

タスク	説明	必要なスキル
カスタムエンドポイントの詳細を、ワークロードを指定するアプリケーションと共有します。	テストする予定のレポートアプリケーションのデータベース接続の詳細に、カスタムエンドポイントの詳細を追加します。	AWS 管理者
カスタムエンドポイントを使用してワークロードConnectします。	レポートアプリケーションでカスタムエンドポイントの詳細を検証します。	AWS 管理者
データベースから接続の詳細を確認します。	1. アプリケーションのユーザー名と接続数をテストします。 2. ワークロード全体の負荷分散をチェックして、接続が異なるカスタムエンドポイント (プライマリとリードレプリカ) に分散されていることを確認します。	AWS 管理者

関連リソース

- 「Aurora エンドポイントのタイプ」
- 「カスタムエンドポイントのメンバーシップルール」
- 「カスタムエンドポイントのエンドツーエンド AWS CLI の例」
- 「Oracle RAC の代替としてのAmazon Aurora」
- 「Oracleから PostgreSQL に移行する際の課題とその克服方法」

Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする

作成者： Rohit Kapoor (AWS)

概要

Amazon Relational Database Service (Amazon RDS) は PostgreSQL DB インスタンスの SSL 暗号化をサポートしています。SSL を使用して、アプリケーションと Amazon RDS for PostgreSQL DB インスタンスとの間の PostgreSQL 接続を暗号化できます。デフォルトでは、Amazon RDS for PostgreSQL は SSL/TLS を使用し、すべてのクライアントが SSL/TLS 暗号化を使用して接続することを想定しています。Amazon RDS for PostgreSQL は TLS バージョン 1.1 および 1.2 をサポートします。

このパターンは、Amazon RDS for PostgreSQL DB インスタンスに対して暗号化された接続を有効にする方法を示しています。同じプロセスを使用して、Amazon Aurora PostgreSQL 互換エディションの暗号化接続を有効化できます。

前提条件と制限

- アクティブな AWS アカウント
- Amazon RDS for PostgreSQL DB インスタンス
- SSL バンドル

アーキテクチャ

ツール

- pgAdmin は、オープンソースの PostgreSQL 向けの管理・開発プラットフォームです。pgAdmin を使用すると、Linux、Unix、macOS、および Windows で PostgreSQL 10 以降のデータベースオブジェクトを管理できます。
- PostgreSQL エディタ は、クエリを作成、開発、実行したり、要件に応じてコードを編集したりする際に役立つ、よりユーザーフレンドリーなインターフェイスを提供します。

ベストプラクティス

- セキュアでないデータベース接続をモニタリングします。
- データベースアクセス権を確認します。
- バックアップとスナップショットが保管中の場合に暗号化されていることを確認します。
- データベースアクセスをモニタリングします。
- 無制限のアクセスグループを避けてください。
- Amazon GuardDuty で通知機能を強化しましょう。
- ポリシーの遵守状況を定期的にモニタリングします。

エピック

信頼できる証明書をダウンロードして信頼ストアにインポートする

タスク	説明	必要なスキル
信頼できる証明書をコンピュータにロードします。	コンピュータの信頼されたルート証明機関ストアに証明書を追加するには、次の手順に従います。(これらの手順では、例として Windows Server を使用しています)。 1. Windows サーバーで、[スタート] から [ファイル名を指定して実行] を選択し、mmc と入力します。 2. コンソールの [ファイル] メニューで、[スナップインの追加と削除] を選択します。 3. [利用できるスナップイン] で、[証明書] を選択し、[追加] を選択します。	DevOps エンジニア、移行エンジニア、DBA

タスク	説明	必要なスキル
	4. [このスナップインは常に証明書を管理します] で、[コンピュータアカウント]、[次へ] の順に選択します。 5. [ローカルコンピュータ] を選択して、[完了] を選択します。 6. コンソールに追加するスナップインがなくなった場合は、[OK] を選択します。 7. コンソールツリーで、[証明書] をダブルクリックします。 8. [信頼されたルート証明機関] を右クリックします。 9. [すべてのタスク]、[インポート] を選択して、ダウンロードした証明書をインポートします。 10[証明書のインポート] ウィザードの指示に従って操作します。	

強制的に SSL 接続する

タスク	説明	必要なスキル
パラメータグループを作成し、rds.force_ssl パラメータを設定します。	PostgreSQL DBインスタンスにカスタムパラメータグループがある場合は、パラメータグループを編集し、`rds.force_ssl` を 1 に変更します。	DevOps エンジニア、移行エンジニア、DBA

タスク	説明	必要なスキル
	DB インスタンスが `rds.force_ssl` を有効にしていないデフォルトのパラメータグループを使用している場合は、新しいパラメータグループを作成します。新しいパラメータグループは Amazon RDS API を使用するか、以下の手順に従って手動で変更できます。 パラメータグループを作成するには： 1. AWS マネジメントコンソールにサインインし、DB インスタンスをホストする AWS リージョンの Amazon RDS コンソールを開きます。 2. ナビゲーションペインで、[パラメータグループ] を選択します。 3. [パラメータグループの作成] を選択し、次の値を設定します。 • [パラメータグループファミリー] で、postgres14 を選択します。 • [グループ名] に pgsql--ssl <database_instance> と入力します。 • [説明] には、追加するパラメータグループの説	

タスク	説明	必要なスキル
	明を自由形式で入力します。 • [作成] を選択します。 4. 先ほど作成したパラメータグループを選択します。 5. [パラメータグループのアクション] で、[編集]を選択します。 6. rds.force_ssl を検索し、その設定を 1 に変更します。 **Note** このパラメータを変更する前に、クライアント側のテストを実施します。 7. [Save changes] (変更の保存) をクリックします。 パラメータグループを PostgreSQL DB インスタンスに関連付けるには： 1. Amazon RDS コンソールのナビゲーションペインで、[データベース] を選択し、PostgreSQL DB インスタンスを選択します。 2. [変更] を選択します。 3. [追加設定] で新しいパラメータグループを選択し、[続行] を選択します。	

タスク	説明	必要なスキル
	4. [変更のスケジュール] で、[すぐに適用] を選択します。 5. [DB インスタンスの変更] を選択します。 詳細については、「Amazon RDS ドキュメント」を参照してください。	
強制的に SSL 接続するようにします。	移行元の Amazon RDS for PostgreSQL インスタンスに接続します。SSL を使用しない接続は拒否され、エラーメッセージが表示されます。詳細については、「Amazon RDS ドキュメント」を参照してください。	DevOps エンジニア、移行エンジニア、DBA

SSL 拡張機能のインストール

タスク	説明	必要なスキル
SSL 拡張機能をインストールします。	1. DBA として psql または pgAdmin 接続を開始します。 2. ssl_is_used () 関数を呼び出して、SSL が使用されているかどうかを調べます。 `select ssl_is_used();` この関数は、この接続が SSL を使用している場合に	DevOps エンジニア、移行エンジニア、DBA

タスク	説明	必要なスキル

	t を返し、それ以外の場合に f を返します。
	3. SSL 拡張機能をインストールします。
	`create extension sslinfo;`
	`show ssl;`
	`select ssl_cipher();`
	詳細については、「Amazon RDS ドキュメント」を参照してください。	

PostgreSQL クライアントを SSL 向けに設定します

タスク	説明	必要なスキル
SSL 向けにクライアントを設定します。	SSL を使用すると、TLS プロトコルを使用する暗号化接続をサポートする PostgreSQL サーバーを起動できます。サーバーは、同じ TCP ポート上の標準接続と SSL 接続をリッスンし、SS Lを使用するかどうかを接続クライアントとネゴシエートします。デフォルトでは、このクライアントオプションは有効になっています。 psql クライアントを使用している場合：	DevOps エンジニア、移行エンジニア、DBA

タスク	説明	必要なスキル

	1. Amazon RDS 証明書がローカルコンピュータに読み込まれていることを確認します。
	2. 以下を追加して SSL クライアント接続を開始します。
	
	`psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser sslmode=verify-full sslrootcert=rds-ssl-ca-cert.pem`
	`select ssl_cipher();`
	
	他の PostgreSQL クライアントの場合:
	
	• それぞれのアプリケーション公開キーパラメータを変更します。これはオプションとして、または GUI ツールの接続ページのプロパティとして使用できます。
	
	これらのクライアントについては、以下のページを確認してください。
	
	• pgAdmin ドキュメント
	• JDBC ドキュメント	

トラブルシューティング

問題	ソリューション
SSL 証明書をダウンロードできません。	ウェブサイトへの接続を確認してから、証明書をローカルコンピュータにダウンロードしてください。

関連リソース

- 「Amazon RDS for PostgreSQL ドキュメント」
- 「PostgreSQL DB インスタンスで SSL を使用する」 (Amazon RDS ドキュメント)
- 「SSL による TCP/IP 接続のセキュリティ保護」 (PostgreSQL ドキュメント)
- 「SSL を使用する」 (JDBC ドキュメント)

既存の Amazon RDS for PostgreSQL DB インスタンスを暗号化する

作成者: Piyush Goyal (AWS)、Shobana Raghu (AWS)、Yaser Raja (AWS)

概要

このパターンでは、AWS クラウド内の PostgreSQL DB インスタンス用の既存の Amazon Relational Database Service (Amazon RDS) を最小限のダウンタイムで暗号化する方法について説明します。このプロセスは、Amazon RDS for MySQL DB インスタンスでも機能します。

Amazon RDS DB インスタンスの暗号化は、DB インスタンスの作成時に有効にすることができますが、作成後に暗号化を有効にすることはできません。ただし、DB インスタンスのスナップショットを作成し、そのスナップショットの暗号化済みコピーを作成すると、暗号化されていない DB インスタンスに暗号化を追加できます。この暗号化されたスナップショットから DB インスタンスを復元することで、元の DB インスタンスの暗号化されたコピーを取得できます。プロジェクトがこのアクティビティの間 (少なくとも書き込みトランザクションの場合) ダウンタイムを許可されている場合は、これがすべての作業になります。DB インスタンスの暗号化された新しいコピーが使用可能になると、アプリケーションを新しいデータベースに接続できます。　ただし、プロジェクトでこのアクティビティによる大幅なダウンタイムが許容されない場合は、ダウンタイムを最小限に抑えるための代替アプローチが必要です。　このパターンでは、AWS Database Migration Service (AWS DMS) を使用してデータを移行し、継続的に複製します。これにより、新しい暗号化されたデータベースへのカットオーバーを最小限のダウンタイムで行うことができます。

Amazon RDS の暗号化された DB インスタンスでは、Amazon RDS DB インスタンスをホストしているサーバーでデータを暗号化するために、業界標準の AES-256 暗号化アルゴリズムを使用します。データが暗号化されると、Amazon RDS はパフォーマンスの影響を最小限に抑えながら、データへのアクセスと復号化の認証を透過的に処理します。暗号化を使用するために、データベースのクライアントアプリケーションを変更する必要はありません。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 暗号化されていないAmazon RDS for PostgreSQL DB インスタンス
- AWS DMS タスクの使用経験 (作成、変更、または停止) (AWS DMS ドキュメントの「AWS DMS タスクの使用」を参照)
- データベースを暗号化するための AWS Key Management Service (AWS KMS) に関する知識 (AWS KMS ドキュメントを参照)

機能制限

- Amazon RDS DB インスタンスの暗号化は、DB インスタンスの作成時にのみ有効にすることができます。作成後に暗号化を有効にすることはできません。
- ログに記録されていないテーブルのデータは、スナップショットを使用しても復元されません。詳細については、「PostgreSQL を使用するためのベストプラクティス」を参照してください。
- 暗号化されていない DB インスタンスのリードレプリカを暗号化することや、暗号化されている DB インスタンスのリードレプリカを暗号化しないようにすることはできません。
- 暗号化されていないバックアップやスナップショットを、暗号化された DB インスタンスに復元することはできません。
- AWS DMS はシーケンスを自動的に転送しないため、これを処理するには追加の手順が必要です。

詳細については、Amazon RDS ドキュメントの「Amazon RDS 暗号化 DB インスタンスの制限事項」を参照してください。

アーキテクチャ

ソースアーキテクチャ

- 暗号化されていない RDS DB インスタンス

ターゲット アーキテクチャ

- 暗号化されていない RDS DB インスタンス
  - ターゲット RDS DB インスタンスは、ソース RDS DB インスタンスの DB スナップショットコピーを復元することによって作成されます。
  - AWS KMS キーは、スナップショットの復元中の暗号化に使用されます。
  - AWS DMS レプリケーションタスクを使用してデータを移行します。

ツール

暗号化を有効化するためのツール:

- 暗号化のための AWS KMS キー – 暗号化された DB インスタンスを作成するときは、カスタマーマネージドキーまたは Amazon RDS の AWS マネージドキーを選択して、DB インスタンスを暗号化できます。カスタマーマネージドキーのキー識別子を指定しない場合、Amazon RDS は新しい DB インスタンスに AWS マネージドキーを使用します。Amazon RDS は、Amazon RDS 用の AWS マネージドキーを AWS アカウントに作成します。AWS アカウントには、AWS リージョンごとに Amazon RDS の AWS マネージドキーがあります。Amazon RDS の暗号化に KMS キーを使用する方法の詳細については、「Amazon RDS リソースの暗号化」を参照してください。

継続的なレプリケーションに使用されるツール:

- AWS DMS - AWS Database Migration Service (AWS DMS) を使用して、ソース DB からターゲット DB に変更を複製します。ダウンタイムを最小限に抑えるために、ソース DB とターゲット DB を同期させることが重要です。AWS DMS の設定とタスクの作成については、「AWS DMS のドキュメント」を参照してください。

エピック

ソース DB インスタンスのスナップショットを作成し、暗号化します

タスク	説明	必要なスキル
ソース PostgreSQL DB インスタンスの詳細を確認します。	Amazon RDS コンソールで、ソース PostgreSQL DB インスタンスを選択します。[設定] タブで、インスタンスの暗号化が有効になっていないことを確認します。画面の図については、「追加情報」セクションを参照してください。	DBA
DB スナップショットを作成します。	暗号化するインスタンスの DB スナップショットを作成します。スナップショットを作成するのにかかる時間は、データベースのサイズによって異なります。手順については、Amazon RDS ドキュメン	DBA

タスク	説明	必要なスキル
	トの「DB スナップショットの作成」を参照してください。	
スナップショットを暗号化します。	Amazon RDS コンソールのナビゲーションペインで、[スナップショット] を選択し、作成した DB スナップショットを選択します。[アクション] では、[スナップショットのコピー] を選択します。対応するフィールドで、送信先の AWS リージョンと DB スナップショットのコピー名を指定します。[暗号化を有効にする] チェックボックスを選択します。[マスターキー] では、DB スナップショットの暗号化に使用する KMS キー識別子を指定します。[スナップショットをコピー] を選択します。詳細については、Amazon RDS のドキュメントの「スナップショットのコピー」を参照してください。	DBA

ターゲット DB インスタンスを準備します

タスク	説明	必要なスキル
DB スナップショットを復元します。	Amazon RDS コンソールで、[スナップショット] タブを選択します。作成した暗号化スナップショットを選択します。[アクション]、[スナップショットの復元] の順に選択し	DBA

タスク	説明	必要なスキル
	ます。DB インスタンス識別子として、新しい DB インスタンスの一意の名前を入力します。インスタンスの詳細を確認してから、[DB インスタンスの復元] を選択します。新しい、暗号化された DB インスタンスがスナップショットから作成されます。詳細については、Amazon RDS ドキュメントの「DB スナップショットからの復元」を参照してください。	
AWS DMS を使用してデータを移行します。	AWS DMS コンソールで、AWS DMS タスクを作成します。[移行タイプ] では、[既存のデータを移行して進行中の変更を複製する] を選択します。[タスク設定] の [ターゲットテーブル作成モード] では、[切り詰め] を選択します。詳細については、AWS DMS ドキュメントの「タスクの作成」を参照してください。	DBA
データ検証を有効にします。	[タスク設定] で、[検証を有効にする] を選択します。これにより、ソースデータとターゲットデータを比較して、データが正確に移行されたことを確認できます。	DBA

タスク	説明	必要なスキル
ターゲット DB インスタンスの制約を無効にします。	ターゲット DB インスタンスで、トリガーと外部キーの制約をすべて無効にしてから、AWS DMS タスクを開始します。トリガーと外部キーの制約を無効にする詳細については、AWS DMS ドキュメントを参照してください。	DBA
データを検証します。	フルロードが完了したら、ターゲット DB インスタンスのデータを検証して、ソースデータと一致するかどうかを確認します。詳細については、AWS DMS ドキュメントの「AWS DMS のデータ検証」を参照してください。	DBA

ターゲット DB インスタンスへのカットオーバー

タスク	説明	必要なスキル
ソース DB インスタンスに対する書き込み操作を停止します。	ソース DB インスタンスでの書き込み操作を停止して、アプリケーションのダウンタイムを開始できるようにします。AWS DMS がパイプライン内のデータのレプリケーションを完了したことを検証します。ターゲット DB インスタンスで、トリガーと外部キーを有効にします。	DBA
データベースシーケンスの更新	ソースデータベースにシーケンス番号が含まれている場合	DBA

タスク	説明	必要なスキル
	は、ターゲットデータベースのシーケンスを検証して更新します。	
アプリケーションのエンドポイントを設定します。	アプリケーション接続で、新しい Amazon RDS DB インスタンスのエンドポイントを使用するように設定します。現在、DB インスタンスは暗号化されています。	DBA、アプリ所有者

関連リソース

- 「AWS DMS タスクの作成」
- 「Amazon CloudWatch を使用したレプリケーション タスクのモニタリング」
- 「AWS DMS タスクのモニタリング」
- 「Amazon RDS 暗号化キーを更新する」

追加情報

ソース PostgreSQL DB インスタンスの暗号化の確認:

このパターンに関するその他の注意事項:

- `rds.logical_replication` パラメータを 1 に設定して、PostgreSQL でのレプリケーションを有効にします。

重要な注意: レプリケーションスロットは、ファイルが外部から (たとえば `pg_recvlogical` によって) 消費されるまで、抽出、変換、ロード(ETL)ジョブを介して、あるいは AWS DMS 経由で先書きログ (WAL) ファイルを保持します。`rds.logical_replication` パラメータ値を 1 に設定すると、AWS DMS は `wal_level`、`max_wal_senders`、`max_replication_slots`、および `max_connections` パラメータを設定します。論理レプリケーションスロットが存在するが、replicationスロットによって予約された WAL ファイルにコンシューマが存在しない場合、トラ

ンザクション ログ ディスクの使用量が増加し、使用可能なストレージ容量が継続的に減少することがあります。この問題を解決するための詳細情報と手順については、AWS サポートナレッジセンターで「Amazon RDS for PostgreSQL で「デバイスに空き領域がありません」または「ディスクがいっぱいです」という エラーが発生する原因を特定する方法」を参照してください。

- DB スナップショットを作成した後、ソース DB インスタンスに加えたスキーマの変更は、ターゲット DB インスタンスには反映されません。
- 暗号化された DB インスタンスを作成したら、その DB インスタンスで使用されている KMS キーを変更することはできません。したがって、暗号化された DB インスタンスを作成する前に、KMS キーの要件を必ず確認してください。
- AWS DMS タスクを実行する前に、ターゲット DB インスタンスのトリガーと外部キーを無効にする必要があります。タスクが完了した後、これらを再度有効にすることができます。

起動時に Amazon RDS データベースの自動タグ付けを強制する

作成者：Susanne Kangnoh (AWS) と Archit Mathur (AWS)

概要

Amazon Relational Database Service (Amazon RDS) は、Amazon Web Services (AWS) クラウドでリレーショナルデータベースを簡単にセットアップし、運用し、拡張することのできるウェブサービスです。業界スタンダードのリレーショナルデータベース向けに、費用対効果に優れたエクステンションを備え、一般的なデータベース管理タスクを管理します。

タグ付けを使用すると、さまざまな方法で AWS リソースを分類できます。リレーショナルデータベースのタグ付けは、アカウントに多数のリソースがあり、特定のリソースをすばやく識別する場合に便利です。Amazon RDS タグを使用して、RDS DB インスタンスにカスタムメタデータを追加できます。　各タグは、ユーザー定義のキーと値で構成されます。組織の要件に適合する一貫したタグのセットを作成することをお勧めします。

このパターンでは、RDS DB インスタンスのモニタリングとタグ付けに役立つ AWS CloudFormation テンプレートについて説明します。このテンプレートは、AWS CloudTrail CreateDBInstance イベントをモニタリングする Amazon CloudWatch Events イベントを作成します。(CloudTrail は、Amazon RDS の API コールをイベントとしてキャプチャします。) このイベントを検出すると、定義したタグキーと値を自動的に適用する AWS Lambda 関数を呼び出します。テンプレートでは、Amazon Simple Notification Service (Amazon SNS) を使用して、インスタンスがタグ付けされたという通知も送信されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon Simple Storage Service (Amazon S3) バケットに、Lambda コードをアップロードします。
- タグ付けの通知を受信するメールアドレス

機能制限

- このソリューションは、CloudTrail の CreateDBInstance イベントをサポートします。他のイベントの通知は作成されません。

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- AWS CloudFormation テンプレートは、さまざまな AWS リージョンとアカウントに複数回使用できます。各リージョンまたはアカウントで、テンプレートを 1 回実行するだけで済みます。

ツール

AWS サービス

- AWS CloudTrail – AWS CloudTrail は、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査を行えるように支援する AWS のサービスです。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを配信します。CloudWatch Events は、運用上の変更が生じると同時にそれらを認識して対応し、環境に応答するためのメッセージを送信する、機能をアクティブ化する、変更を行う、および状態情報を収集することによって、必要に即した是正措置を講じます。
- AWS Lambda – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS –Amazon Simple Notification Service (Amazon SNS) は、アプリケーション、エンドユーザー、およびデバイスでクラウドから通知を瞬時に送受信できるようにするウェブサービスです。

コード

このパターンには、次の 2 つのファイルを含む添付ファイルが含まれます。

- `index.zip`は、このパターンの Lambda コードを含む圧縮ファイルです。
- `rds.yaml` は、Lambda コードをデプロイする CloudFormation テンプレートです。

これらのファイルの使用方法については、「エピック」セクションを参照してください。

エピック

Lambda コードをデプロイします。

タスク	説明	必要なスキル
S3 バケットにコードをアップロードします。	新しい S3 バケットを作成するか、既存の S3 バケットを使用して、添付 `index.zip` ファイル (Lambda コード) をアップロードします。このバケットは、モニタリングするリソース (RDS DB インスタンス) と同じ AWS リージョンに存在する必要があります。	クラウドアーキテクト
CloudFormation テンプレートをデプロイする。	S3 バケットと同じ AWS リージョンで Cloudformation コンソールを開き、添付ファイルで提供されている `rds.yaml` ファイルをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を入力します。	最初のエピックで作成または選択した S3 バケットの名	クラウドアーキテクト

タスク	説明	必要なスキル
	前を入力します。この S3 バケットには,Lambda コードの.zip ファイルが含まれており、CloudFormation テンプレートおよびモニタリングする RDS DB インスタンスと同じ AWS リージョンに存在する必要があります。	
S3 キーを入力します。	S3 バケット内の Lambda コードの.zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例：`index.zip` または `controls/index.zip`)。	クラウドアーキテクト
メールアドレスを入力します。	違反の通知を受信するメールアドレスを入力します。	クラウドアーキテクト
ログレベルを指定します。	ログ記録レベルと詳細レベルを定義します。`Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定し、デバッグのみに使用されます。`Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。`Warning` 潜在的に危険有害な状況を示します。	クラウドアーキテクト

タスク	説明	必要なスキル
RDS DB インスタンスのタグキーと値を入力します。	RDS インスタンスに自動適用したい必須のタグキーと値を入力します。詳細については、AWS ドキュメントの「Amazon RDS リソースのタグ付け」を参照してください。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
メールサブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定されたメールアドレスにサブスクリプションのメールメッセージが送信されます。インスタンスがタグ付けされたときに通知を受信するには、このメールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「バケットの作成」 (Amazon S3 ドキュメント)
- 「Amazon RDS リソースのタグ付け」 (Amazon Aurora ドキュメント)
- 「オブジェクトのアップロード」 (Amazon S3 ドキュメント)
- 「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成」 (Amazon CloudWatch ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

DynamoDB テーブルのコストをオンデマンドで見積る

作成者: Moinul Al-Mamun (AWS)

概要

Amazon DynamoDB は、ペタバイトスケールでもミリ秒単位 1 桁のレイテンシーを実現する NoSQL トランザクションデータベースです。この Amazon Web Services (AWS) のサーバーレスサービスは、一貫したパフォーマンスとスケーラビリティにより人気が高まっています。 基盤となるインフラストラクチャをプロビジョニングする必要はありません。1 つのテーブルが最大でペタバイトまで増加する可能性があります。

オンデマンドキャパシティモードでは、アプリケーションがテーブルに対して実行するデータの読み取りと書き込みに対して、リクエストごとに料金を支払います。AWS の料金は、1 か月あたりの累積読み取りリクエスト単位 (RRU) と書き込みリクエスト単位 (WRU) に基づいています。DynamoDB では、1 か月にわたるテーブルサイズの継続的なモニタリングに基づいてストレージ料金が決定されます。ポイントインタイムリカバリ (PITR) による継続的バックアップのサポートを提供します。DynamoDB では、1 か月にわたる PITR 対応テーブルサイズの継続的なモニタリングに基づいてバックアップ料金が決定されます。

プロジェクトの DynamoDB コストの見積もりでは、製品ライフサイクルのさまざまな段階で消費される RRU、WRU、およびストレージの値を計算することが重要です。大まかなコスト見積もりには AWS 料金見積りツールを使用できますが、テーブルに必要な RRU、WRU、およびストレージ要件のおおよその値を提供する必要があります。プロジェクトの開始時に、これらの値を見積もるのが難しい場合があります。AWS 料金見積りツールでは、データの増加率や項目サイズは考慮されません。また、ベーステーブルおよびグローバルセカンダリインデックス (GSI) の読み取りと書き込みの回数は個別に考慮されません。AWS 料金見積りツールを使用するには、これらすべての要素を見積もり、WRU、RRU、ストレージサイズの大まかな数値を想定してコストを見積もる必要があります。

このパターンは、DynamoDB の基本的なコスト要因 (書き込み、読み取り、ストレージ、バックアップ、リカバリの費用など) を見積もるメカニズムと再利用可能な Microsoft Excel テンプレートが提供されます。AWS 料金見積りツールよりも詳細で、ベーステーブルと GSI の要件を個別に検討します。また、項目データの月次増加率を考慮して、3 年間の費用を予測します。

前提条件と制限

前提条件

- DynamoDB と DynamoDB データモデル設計に関する基本的な知識

- DynamoDB の料金、WRU、RRU、ストレージ、バックアップとリカバリに関する基本的な知識 (詳細については、「オンデマンドキャパシティの料金」を参照してください)
- DynamoDB のデータ、データモデル、項目サイズに関する知識
- DynamoDB GSI に関する知識

機能制限

- テンプレートではおおよその計算のみで、すべての構成に適応はしていません。より正確な見積もりを得るには、ベーステーブルと GSI の各項目のサイズを個別に試算する必要があります。
- より正確な見積もりでは、各項目の書き込み (挿入、更新、削除) 回数と読み取り回数の平均を考慮する必要があります。
- このパターンでは、固定データ増加の仮定に基づいて、今後数年間の書き込み、読み取り、ストレージ、バックアップ、リカバリ費用のみの見積もりが可能です。

ツール

AWS サービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。

その他のツール

- AWS 料金見積りツールは、ユースケースの見積りを作成するために使用できるウェブベースの計画ツールです。

ベストプラクティス

コストを低く抑えるには、次の DynamoDB 設計のベストプラクティスを考慮してください。

- パーティションキーの設計 — カーディナリティの高いパーティションキーを使用して負荷を均等に分散します。
- 隣接関係リストの設計パターン — この設計パターンは、一対多および多対多リレーションシップを管理する場合に使用します。
- スパースインデックス — GSI にはスパースインデックスを利用します。GSI を作成する際、パーティションキーおよびソートキー (オプション) を指定します。対応する GSI パーティションキー

を含むベーステーブルの項目だけが、スパースインデックスに表示されます。これにより GSI を小さく抑えることができます。

- インデックスの多重定義 — さまざまなタイプの項目のインデックス作成に、同じ GSI を使用します。
- GSI 書き込みシャーディング — うまくシャーディングしてパーティション全体にデータを分散することで、クエリを効率的かつ高速に行うことができます。
- 大きなアイテム — テーブル内にメタデータのみを保存し、blob を Amazon S3 に保存し、リファレンスを DynamoDB に保持します。大きな項目を複数の項目に分割し、ソートキーを使用して効率的にインデックスを作成します。

設計のベストプラクティスについての詳細は、Amazon DynamoDB の「デベロッパーガイド」を参照してください。

エピック

DynamoDB データモデルから項目情報を抽出します。

タスク	説明	必要なスキル
項目サイズを取得します。	1. テーブルに格納するアイテムの種類を確認してください。 2. 各項目のサイズをキロバイト単位で計算するには、各プロパティのキーと値のサイズを合計します。 3. ベーステーブルと各 GSI の項目サイズを計算します。	データエンジニア
書き込み費用を見積もります。	オンデマンドキャパシティモードでの書き込み費用を見積もるには、1 か月に消費される WRU の値を測定する必要があります。そのため、以下の要素を考慮する必要があります。	データエンジニア

タスク	説明	必要なスキル

	• 1 か月間に実行される、各アイテムの作成、更新、および削除オペレーションの数。
	• 使用可能な GSI の数。 各インデックスを個別に検討してください。
	  • インデックス項目の平均サイズ。
	  • インデックスの同期回数。
	• 毎月、コンポーネントや製品などの新しい項目が何個テーブルに追加されますか？ 追加数は毎月異なりますが、ビジネスケースに基づいて平均的な増加率を想定できます。
	
	詳細については、「追加情報」セクションをご覧ください。	

タスク	説明	必要なスキル
読み取り費用を見積もります。	オンデマンドモードでの読み取り費用を見積もるには、まず 1 か月に消費される RRU の値を測定する必要があります。そのため、以下の要素を考慮する必要があります。 • 使用可能な GSI の数。 各インデックスを個別に検討してください。 • インデックス項目の平均サイズ。 • 1 製品あたり 1 か月あたりの平均読み取り回数。 • DynamoDB テーブルで使用可能な (コンポーネントまたは製品) の合計数。	データエンジニア、アプリ開発者

タスク	説明	必要なスキル
ストレージのサイズと費用を見積もります。	最初に、テーブルの項目サイズに基づいて、月間平均ストレージ要件を見積もります。次に、ストレージサイズに AWS リージョンの GB あたりのストレージ料金を掛けてストレージ費用を計算します。 書き込み費用を見積もるためのデータを既に入力している場合は、ストレージサイズの計算のためにデータを再度入力する必要はありません。それ以外の場合、ストレージサイズの見積りでは、以下の要素を考慮する必要があります。 • テーブル設計に基づくモジュール (製品) 内のデータの項目数。 • 平均の項目サイズ (KB) • 使用可能な GSI の数。 各インデックスを個別に検討してください。 • インデックス項目の平均サイズ。 • 毎月、新しい項目が何個テーブルに追加されますか？ 新製品の数は毎月異なりますが、ビジネスケースに基づいて平均的な増加率を想定できます。この例で	データエンジニア

タスク	説明	必要なスキル
	は、毎月平均 1,000 万件の新製品を使用しています。	

Excel テンプレートに項目とオブジェクトの情報を入力します。

タスク	説明	必要なスキル
添付ファイルセクションから Excel テンプレートをダウンロードし、ユースケーステーブルに合わせて調整します。	1. Excel テンプレートをダウンロードします。 2. テーブル設計に基づいてビジネスモジュールと GSI を調整します	データエンジニア
Excel テンプレートに情報を入力します。	1. シート内のアイテム情報を更新します。オレンジ色のセルのデータのみ更新します。 2. オブジェクト番号の調整: 1 か月あたりどれくらいの量をテーブルに追加しますか? 3. お住まいの AWS リージョンの 100 万単位の WRU と RRU の料金を更新します。 4. お使いの AWS リージョンのストレージ料金とバックアップ料金を GB 単位で更新します。 5. お使いの AWS リージョンの GB あたりのリカバリ料金を更新します。	データエンジニア

タスク	説明	必要なスキル
	テンプレートには、情報、メタデータ、リレーションシップの 3 つの項目またはエンティティが含まれます。2 つの GSI があります。ユースケースに合わせて、さらに項目が必要な場合は、新しい行を作成してください。さらに GSI が必要な場合は、既存の GSI ブロックをコピーして貼り付け、必要な数の GSI ブロックを作成します。次に、「合計」列と「合計」列の計算を調整します。	

関連リソース

リファレンス

- 「Amazon DynamoDB のオンデマンドキャパシティ料金表」
- 「DynamoDB 用の AWS 料金見積りツール」
- 「DynamoDB を使用した設計とアーキテクチャに関するベストプラクティス」
- 「DynamoDB の使用開始」

ガイドとパターン

- 「Amazon DynamoDB によるデータのモデリング」
- 「Amazon DynamoDB テーブルのストレージ費用を見積もる」

追加情報

費用計算の例

DynamoDB データモデルの設計では、1 つの製品について 3 つの項目が表示され、平均項目サイズは 4 KB です。DynamoDB ベーステーブルに新しい製品を追加すると、項目数 x (項目サイズ/1 KB 書き込み単位) = 3 x (4/1) = 12 WRU が消費されます。この例では、1 KB を書き込むと、製品は 1 WRU を消費します。

費用計算の例

RRU の見積もりを求めるには、各項目の 1 か月にわたる読み取り回数の平均を考慮してください。たとえば、情報項目は 1 か月に平均で 10 回読み取られ、メタデータ項目は 2 回、リレーションシップアイテムは 5 回読み取られます。このテンプレート例では、すべてのコンポーネントの合計 RRU = 毎月作成される新しいコンポーネントの数 x コンポーネントあたりの RRU = 1,000 万 x 17 RRU = 1 か月あたり 1 億 7,000 万 RRU です。

毎月、新しいもの (コンポーネントまたは製品) が追加され、製品の総数は時間の経過に伴い増えていきます。そのため、RRU の要件も時間に伴い増加していきます。

- 最初の 1 か月の RRU の消費量は 1 億 7,000 万になります。
- 2 か月目の RRU の消費量は 2 x 1 億 7,000 万 = 3 億 4,000 万になります。
- 3 か月目の RRU の消費量は 3 x 1 億 7,000 万 = 5 億 1,000 万になります。

次のグラフは、毎月の RRU 消費量と費用予測を示しています。

注意：グラフ内の料金は説明の目的でのみ使用されています。ユースケースに適応した正確な予測を作成するには、AWS 料金ページを確認し、その料金を Excel シートに入力してください。

ストレージ、バックアップ、リカバリ費用の計算例

DynamoDB ストレージ、バックアップ、リカバリはすべて相互に接続されています。バックアップはストレージに関係し、リカバリはバックアップサイズに関係します。テーブルのサイズが大きくなると、それに対応するストレージ、バックアップ、およびリカバリの費用も比例して増加します。

ストレージのサイズと費用

ストレージ費用は、データの増加率に応じて時間の経過に伴い増加していきます。例えば、ベーステーブルと GSI にあるコンポーネントまたは製品の平均サイズが 11 KB で、データベーステーブルに毎月 1,000 万個の新しい製品が追加されると仮定します。その場合、DynamoDB テーブルのサイズは (11 KB x 1000 万) /1024/1024 = 1 か月あたり 105 GB 増加します。最初の月のテーブルストレージのサイズは 105 GB になり、2 か月目には 105 + 105 = 210 GB というようになります。

- 最初の月には、AWS リージョンのストレージ費用は 105 GB x 1 GB あたりのストレージ料金になります。
- 2 か月目には、お住まいのリージョンのストレージ費用は 210 GB x 1 GB あたりのストレージ料金になります。
- 3 か月目には、そのリージョンのストレージ費用は 315 GB x 1 GB あたりのストレージ料金になります。

今後 3 年間のストレージサイズと費用については、「ストレージサイズと予測」セクションを参照してください。

バックアップの費用

バックアップの費用は、データの増加率に応じて時間の経過に伴い増加していきます。ポイントインタイムリカバリ (PITR) による継続的なバックアップを有効にする場合、継続的なバックアップの料金は、ストレージの月間平均 GB 単位に基づいて課金されます。1 か月あたりの平均バックアップサイズは、テーブルのストレージサイズと同じですが、実際のサイズは若干異なる場合があります。新しい製品が毎月追加されるため、ストレージの合計サイズとバックアップサイズは時間の経過の伴い増加していきます。例えば、最初の月の平均バックアップサイズは 105 GB でしたが、2 か月目には 210 GB に増える可能性があります。

- 最初の月のバックアップ費用は、お使いの AWS リージョンの連続バックアップ料金 (GB あたり 105 GB x) になります。
- 2 か月目のバックアップ費用は、お住まいのリージョンの 1 GB あたり 210 GB/月 x 継続バックアップ料金になります。
- 3 か月目のバックアップ費用は、お住まいのリージョンの 1 GB あたりの継続バックアップ料金が 315 GB/月 * になります。
- 以降も同様になります

バックアップ費用は、「ストレージサイズとコストの予測」セクションのグラフに含まれています。

リカバリ費用

PITR を有効にした状態で継続的なバックアップを行う場合、リカバリ操作料金はそのサイズに基づいて計算されます。リカバリするたびに、リカバリデータのギガバイト数に基づいて支払いが発生します。テーブルのサイズが大きく、1 か月に複数回リカバリを実行すると、コストが高くなります。

リストア費用を見積もるために、この例では毎月月末に PITR 復元を実行することを想定しています。この例では、月間平均バックアップサイズをその月のリカバリデータサイズとして使用しています。最初の月の平均バックアップサイズは 105 GB で、月末の復元データサイズは 105 GB です。2 か月目は 210 GB になり、以降も同様になります。

リカバリ費用は、データの増加率に応じて時間の経過に伴い増加していきます。

- 最初の月には、105 GB x AWS リージョンの GB あたりの復元料金になります。
- 2 か月目のリカバリ費用は、210 GB * お住まいのリージョンの GB あたりの復元料金になります。
- 3 か月目のリカバリ費用は、315 GB * お住まいのリージョンの GB あたりの復元料金になります。

詳細については、Excel テンプレートの [ストレージ、バックアップ、リカバリ] タブと次のセクションのグラフを参照してください。

ストレージのサイズとコストの予測

テンプレートでは、実際の課金ストレージサイズは、標準テーブルクラスの毎月 25 GB の無料利用枠を差し引いて計算されます。シートには、月次値ごとに分割された予測グラフが表示されます。

次のグラフの例では、今後 36 か月間の月間ストレージサイズ (GB)、請求可能なストレージ費用、オンデマンドバックアップ費用、およびリカバリ費用を予測しています。コストの見積り グラフから、ストレージ、バックアップ、リカバリの費用は、ストレージサイズの増加に比例して増加することが分かります。

注意：グラフ内の料金は説明の目的でのみ使用されています。ユースケースに適応した正確な料金を作成するには、AWS 料金ページを確認し、その料金を Excel シートに入力してください。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon DynamoDB テーブルのストレージコストを推定

作成者: Moinul Al-Mamun (AWS)

概要

Amazon DynamoDB は、ペタバイトスケールでもミリ秒単位 1 桁のレイテンシーを実現する NoSQL トランザクションデータベースです。のこの一般的なサーバーレスサービスは、一貫したパフォーマンスとスケーラビリティ AWS を提供します。ストレージをプロビジョニングする必要はなく、単一のテーブルは最大ペタバイトまで拡張できます。

DynamoDB は、テーブルのサイズを 1 か月間継続的にモニタリングしてストレージ料金を決定します。 AWS その後、ストレージの平均サイズをギガバイト単位で請求します。テーブルが時間の経過とともに大きくなればなるほど、ストレージコストも増加します。ストレージコストを計算するには、 AWS 料金計算ツールを使用できますが、グローバルセカンダリインデックス (GSIs) など、テーブルのおおよそのサイズを指定する必要があります。これは、プロジェクトの開始時に見積もるのが本当に難しいものです。また、 AWS 料金計算ツールでは、データの増加率は考慮されません。

このパターンでは、DynamoDB ストレージのサイズとコストを計算するためのメカニズムと、再利用可能な Microsoft Excel テンプレートを提供します。ベーステーブルと GSI のストレージ要件を別々に考慮します。個々のアイテムのサイズと時間の経過に伴うデータの増加率を考慮して、ストレージサイズを計算します。

見積もりを求めるには、次の 2 つの情報をテンプレートに挿入します：

- ベーステーブルと GSI の個々の項目サイズ (キロバイト)
- 1ヶ月間に、テーブルに追加できる新しいオブジェクトまたは製品の平均数 (例、1,000 万)

テンプレートは、次の例に示すように、今後 3 年間のストレージとコストの予測グラフを生成します。

前提条件と制限

前提条件

- DynamoDB ストレージと料金など、DynamoDB の基本知識
- DynamoDB のデータ、データモデル、項目サイズに関する知識
- DynamoDB グローバルセカンダリインデックス (GSI) に関する知識

機能制限

- このテンプレートでは、おおよその計算が可能ですが、すべての構成に適しているわけではありません。より正確な推定を行うには、ベーステーブルと GSI の各項目のサイズを個別に測定する必要があります。
- このパターンでは、固定データ増加の仮定に基づいて、今後数年間のストレージサイズとコストのみを見積もることができます。

ツール

AWS サービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。

その他のツール

- AWS 料金計算ツールは、 AWS ユースケースの見積りを作成するために使用できるウェブベースの計画ツールです。

エピック

DynamoDB データモデルから項目情報を抽出します。

タスク	説明	必要なスキル
項目サイズを取得します。	1. テーブルに保存するさまざまな項目タイプの数を決定します。 2. 各項目のサイズをキロバイト単位で計算するには、各属性の `Key`および `Value`サイズを追加します。 3. ベーステーブルと各 GSI のアイテムサイズを計算します。	データエンジニア

タスク	説明	必要なスキル
1ヶ月間に追加されたオブジェクトの数を取得します。	1ヶ月間に DynamoDB テーブルに追加されるコンポーネントまたはオブジェクトの平均数を推定します。	データエンジニア

Excel テンプレートに項目とオブジェクトの情報を入力します。

タスク	説明	必要なスキル
Excel スプレッドシートをダウンロードして調整します。	1. 添付のドキュメントから Excel テンプレートをダウンロードします。 2. テーブル設計に基づいてビジネスモジュールと GSI を調整します	データエンジニア
エクセルのテンプレートに情報を入力します。	1. シートの項目情報を更新します。 2. オブジェクト番号を調整: テーブルには毎月どれくらい追加できますか? 3. の GB/月あたりのストレージ料金を更新します AWS リージョン。	データエンジニア

関連リソース

- Amazon DynamoDB のオンデマンドキャパシティの料金
- AWS DynamoDB の料金計算ツール

追加情報

添付のテンプレートでは、Standard ストレージテーブルクラスのストレージサイズとコストのみを予測していることに注意してください。ストレージコストの予測に基づき、個々のアイテムのサイズと製品またはオブジェクトの増加率を考慮して、次が推定できっます：

- データエクスポートコスト
- バックアップとリカバリコスト
- データストレージの要件。

Amazon DynamoDB データストレージのコスト

DynamoDB はテーブルサイズの継続的監視を行って、ストレージ料金を決定します。DynamoDB は、データの RAW バイトサイズと有効化した特徴量に応じて、項目あたりのストレージオーバーヘッドを加算して、請求対象データのサイズを測定します。詳細については、「Amazon DynamoDB 開発者ガイド」を参照してください。

データストレージの料金は、テーブルクラスによって異なります。DynamoDB スタンダードテーブルクラスを使用している場合、毎月保存される最初の 25 GB は無料です。さまざまな の標準および標準低頻度アクセステーブルクラスのストレージコストの詳細については AWS リージョン、「オンデマンドキャパシティの料金」を参照してください。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWR レポートを使用して Oracle データベースの Amazon RDS エンジンサイズを推定

作成者: Abhishek Verma (AWS) および Eduardo Valentim (AWS)

概要

Oracle データベースを Amazon Relational Database Service (Amazon RDS) または Amazon Aurora に移行する場合、ターゲットデータベースの CPU、メモリ、ディスク I/O を計算することが重要な要件です。Oracle 自動ワークロードリポジトリ (AWR) レポートを分析することにより、ターゲットデータベースに必要なキャパシティを推定できます。このパターンでは、AWR レポートを使用して、これらの値を推定する方法を説明します。

ソース Oracle データベース は、オンプレミスで、または Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされることが可能です。あるいは、Amazon RDS for Oracle DB インスタンス でも可能です。ターゲットデータベースは、Amazon RDS または Aurora データベースのどれでも構いません。

> **Note**
>
> ターゲットデータベースエンジンが Oracle の場合、容量の見積もりはより正確になります。その他の Amazon RDS データベースでは、データベースアーキテクチャの違いによりエンジンサイズが異なる場合があります。

Oracle データベースを移行する前にパフォーマンステストを実行することを推奨します。

前提条件と制限

前提条件

- AWR レポートをダウンロードするには、Oracle データベースエンタープライズエディションライセンスと Oracle 診断パックライセンスが必要です。

製品バージョン

- バージョン 11g (バージョン 11.2.0.3.v1 以降) および 12.2、18c、19c までのすべての Oracle Database エディション。

- このパターンでは、Oracleエンジニアドシステムズ や Oracleクラウドインフラストラクチャ（OCI）が、カバーされていません。

アーキテクチャ

ソーステクノロジースタック

次のいずれかです:

- オンプレミスの Oracle データベース
- EC2 インスタンス上の Oracle データベース
- Amazon RDS for Oracle DB インスタンス。

ターゲットテクノロジースタック

- Amazon RDS データベース または Amazon Aurora クラスター

ターゲット アーキテクチャ

完全な移行プロセスについては、「AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行する」のパターンを参照してください。

自動化とスケール

移行する 複数の Oracle データベースがあり、追加のパフォーマンスメトリクスを使用したい場合、ブログ記事「Oracle のパフォーマンスメトリックスに基づきAmazon RDS インスタンスを大規模に適切なサイズ」 で説明されている手順に従ってプロセスを自動化できます。

ツール

- 「Oracle 自動ワークロードリポジトリ (AWR)」 は Oracle データベースにビルドインされているリポジトリです。システム・アクティビティとワークロード・データを定期的に収集して保存し、自動データベース診断モニター (ADDM) によって分析します。AWR は、システムパフォーマンスデータのスナップショットを定期的に(デフォルトでは 60 分ごと) 作成し、その情報を保存します(デフォルトでは最大 8 日間)。 AWR のビューとレポートを使用して、このデータを分析できます。

ベストプラクティス

- ターゲットデータベースのリソース要件を計算するために、単一の AWR レポート、複数の AWR レポート、または動的 AWR ビューを使用できます。ピークロードの時に、複数の AWR レポートを使用して、ピーク時のロードを処理するのに必要なリソースを推定することを推奨します。さらに、動的ビューでは、リソース要件をより正確に計算することを支援するデータポイントがより多く提供されます。
- IOPSの推定は移行する予定のデータベースについてのみ行い、他のデータベースやそのディスクを使うプロセスについては行いません。
- データベースが使用する I/O の量を計算するには、AWR レポートのロードプロファイルセクションの情報を使用しないでください。その代わり、可能な場合は、 I/O プロファイルセクションを使用するか、インスタンスアクティビティステータスセクションに飛んで、物理的な読み書き操作の合計値を確認します。
- CPU 使用率を推定する場合、オペレーティングシステム (OS) の統計の代わりに、データベースメトリクスメソッドを使用することをお勧めします。理由は、それがデータベースのみが使用する CPU に基づいているためです。(OS 統計情報には他のプロセスによる CPU 使用量も含まれます。) 移行後のパフォーマンスを向上させるには、ADDM レポートの CPU 関連の推奨事項も確認する必要があります。
- 適切なインスタンスタイプを決定する際には、特定のインスタンスサイズの I/O スループット制限 (Amazon Elastic Block Store (Amazon EBS) のスループットとネットワークスループット) を考慮します。
- 移行前にパフォーマンステストを実行して、エンジンサイズを検証します。

エピック

レポートを作成する

タスク	説明	必要なスキル
AWR レポートを有効にします。	レポートを有効にするには、「Oracle ドキュメント」の手順に従ってください。	DBA
保持期間を確認します。	AWR レポートの保持期間を確認するには、次のクエリを使用します。	DBA

タスク	説明	必要なスキル
	`SQL> SELECT snap_interval,retention FROM dba_hist_wr_control;`	
スナップショットを生成します。	AWR スナップショットの間隔がピークワークロードの急増を捉えるほどきめ細かくない場合、AWR レポートを手動で生成できます。手動 AWR スナップショットを生成するには、次のクエリを使用します。 `SQL> EXEC dbms_workload_repository.create_snapshot;`	DBA
最近のスナップショットをチェックします。	最近の AWR スナップショットを確認するには、次のクエリを使用します。 `SQL> SELECT snap_id, to_char(begin_interval_time,'dd/MON/yy hh24:mi') Begin_Interval, to_char(end_interval_time,'dd/MON/yy hh24:mi') End_Interval FROM dba_hist_snapshot ORDER BY 1;`	DBA

ディスク I/O 要件を推定します

タスク	説明	必要なスキル
メソッドを選択します。	IOPS は、ストレージデバイスの 1 秒あたりの入力操作と出力操作の標準的な尺度であり、読み取り操作と書き込み操作の両方が含まれます。 オンプレミスデータベースを AWS に移行する場合、データベースが使用するピークディスク I/O を決定する必要があります。 次の方法を使用して、ターゲットデータベースのディスク I/O を推定することができます: • AWR レポートのロードプロファイルセクション • AWR レポートのインスタンスアクティビティ統計セクション (Oracle データベース12c 以降の場合、このセクションを使用します) • AWR レポートの I/O プロファイルセクション (12c 以前のバージョンの Oracle データベースの場合、このセクションを使用します) • AWR ビュー 次のステップでこの 4 つのメソッドについて説明します。	DBA

タスク	説明	必要なスキル
オプション 1: 負荷プロファイルを使用します。	次のテーブルでは、AWR レポートのロードプロファイルセクションの例を示しています。 ⚠ Important より正確な情報を得るには、ロードプロファイルの代わりにオプション 2 (I/O プロファイル) またはオプション 3 (インスタンスアクティビティ統計) を使用することをお勧めします。 1 秒あたり / トランザクションあたり / 実行あたり / 呼び出しあたり DB 時間: 26.6 0.2 0.00 0.02	DBA

タスク	説明	必要なスキル
	DB CPU 18.0 0.1 0.00 0.01 バックグラウンド CPU 0.2 0.0 0.00 0.00 やり直しのサイズ (バイト): 2,45 17,0 .9 ロジカルリード (ブロック): 3,37 23,4 .5 ブロッ 21,6 150.	

タスク	説明	必要なスキル
	ク変更: 物理的読み取り(ブロック): 13,5 94.4 物理的書き込み(ブロック): 3,46 24.1 読み取りIOリクエ 3,58 24.9	

タスク	説明	必要なスキル
	スト: 書き込みIOリクエスト: 574.1 4.0 読み取りIO (MB) 106.1 0.7 書き込みIO (MB) 27.1 0.2 IMスキャン行: 0.0 0.0 セッショ	

タスク	説明	必要なスキル
	ンのロジカル読み取り IM: ユーザーの呼び出し: 解析 (SQL ハード解析 (SQL SQL ワークエリ	

タスク	説明	必要なスキル
	ア (MB) ログオン: 1.7 0.0 実行 (SQL 136,(950.4 ロールバック: 22.9 0.2 トランザクション: 143.8 この情報に基づいて、IOPS とスループットを次のように計算できます： IOPS = 読み取りI /O リクエスト: + 書き込み I/O リクエスト = 3,586.8 + 574.7 = 4134.5 スループット = 物理読み取り (ブロック) + 物理的書き	

タスク	説明	必要なスキル
	込み (ブロック) = 13,575.1 + 3,467.3 = 17,042.4 Oracle のブロックサイズが 8 KB なので、合計スループットは次のように計算できます: メガバイト単位の合計スループットは、 17042.4 * 8 * 024 / 1024 / 1024 = 133.2 MB **⚠ Warning** ロードプロファイルを使用してインスタンスサイズを推定しないでください。それは、インスタンスのアクティビティ統計や I/O プロファイルほど正確ではありません。	

タスク	説明	必要なスキル
オプション 2: インスタンスアクティビティ統計を使用します。	12c 以前のバージョンの Oracle Database を使用している場合、AWR レポートのインスタンスアクティビティ統計セクションを使用して IOPS とスループットを推定できます。次の表に、この構文の例を示します。 Statist 合計 1 秒あたり トランザクションあたり 物理的読み取り、IO リクエストの合計 2,547,,217 3,610. 25.11 物理的読み取りの合計バイト数 80,776,124, 114,48 796,149 8	DBA

タスク	説明	必要なスキル
	物理的書き込みの I/O リクエストの合計数 534、 757.1 5.27 08 物理的書き込みの合計バイト数 25,517 8,849, 36,165 1.84 251,508 8 この情報に基づいて、合計 IOPS とスループットを次のように計算できます： 合計 IOPS = 3,610.28 + 757.11 = 4367 合計 Mbps = 114,482,426.26 + 36,165,631.84 = 150648058.1/ 1024 / 1024 = 143 Mbps	

タスク	説明	必要なスキル
オプション 3: I/O プロファイルを使用します。	Oracle Database 12c では、AWR レポートに I/O Profiles セクションが含まれています。このセクションでは、すべての情報が 単一のテーブルに表示され、データベースのパフォーマンスに関するより正確なデータが得られます。次の表に、この構文の例を示します。 1 秒あたりの読み取り+書き込み / 1 秒あたりの読み取り数 / 1 秒あたりの書き込み数 リクエスト合計: 4,367. / 3,610. / 757.1 データベースリクエスト: 4,161. / 3,586. / 574.7 最適化さ 0.0 / 0.0 / 0.0	DBA

タスク	説明			必要なスキル
	れたリクエスト:			
	やり直しリクエスト:	179.3	2.8	176.6
	合計 (MB):	143.7	109.2	34.5
	データベース (MB):	133.1	106.1	27.1
	最適化された合計 (MB):	0.0	0.0	0.0
	やり直し (MB):	7.6	2.7	4.9
	データベース (ブ	17,042	13,575	3,467.3

タスク	説明	必要なスキル
	ロック): バッファキャッシュ経由 (ブロック): 5,898. 5,360. 537.6 ダイレクト (ブロック): 11,143 8,214. 2,929.7 このテーブルでは、スループットと合計 IOPS について次の値を示します： スループット = 143 MBPS (5 行目の 2 列目の合計とラベル付きから) IOPS = 4,367.4 (最初の行のリクエストの合計 とラベル付き、2 番目の列)	

タスク	説明	必要なスキル
オプション 4: AWR ビューを使用します。	AWR ビューを使用しても、同じ IOPS とスループット情報が表示されます。この情報を取得するには、次のクエリを使用します： `break on report compute sum of Value on report select METRIC_NAME,avg(AVERAGE) as "Value" from dba_hist_sysmetric_summary where METRIC_NAME in ('Physical Read Total IO Requests Per Sec','Physical Write Total IO Requests Per Sec') group by metric_name;`	DBA

CPU 要件の推定

タスク	説明	必要なスキル
メソッドを選択します。	ターゲットデータベースに必要な CPU は、次の 3 つの方法で推定することができます: • プロセッサの実際に使用可能なコアを使用して • OS 統計に基づく活用コアを使用して • データベース統計に基づく活用コアを使用して	DBA

タスク	説明	必要なスキル
	使用されるコアを確認する場合、OS 統計の代わりに、データベースメトリクスメソッドを使用することを推奨します。理由は、移行を計画しているデータベースのみが使用する CPU に基づいているためです。(OS 統計情報には他のプロセスによる CPU 使用量も含まれます。) 移行後のパフォーマンスを向上させるには、ADDM レポートの CPU 関連の推奨事項も確認する必要があります。 CPU 世代に基づいて、要件を推定することもできます。異なる世代の CPU を使用している場合、ホワイトペーパー「ワークロードパフォーマンスを最適化するための vCPUs 数の謎を解き明かす」の手順に従うことで、必要なターゲットデータベースを推定することができます。	

タスク	説明	必要なスキル
オプション 1: 利用可能なコアに基づいて要件を推定します。	AWR レポートで: • CPU は論理 CPU と仮想 CPU を指します。 • コアは物理 CPU チップセットに搭載されているプロセッサの数です。 • ソケットは、チップをボードに接続する物理デバイスです。マルチコアプロセッサには、複数の CPU コアを搭載したソケットがあります。 使用可能なコア数は次の 2 つの方法のいずれかで推定できます： • OSコマンドを使用して • AWR レポートを使用して OS コマンドを使用して使用可能なコア数を推定するには 次のコマンドを使用して、プロセッサのコアをカウントします。 `$ cat /proc/cpuinfo | grep "cpu cores"|uniq cpu cores : 4 cat /proc/cpuinfo | egrep "core id|physical id" | tr -d "\n" | sed s/physical/\\nphys`	DBA

タスク	説明	必要なスキル
	ical/g | grep -v ^$ | sort | uniq | wc -l 以下のコマンドを使用して、プロセッサのソケット数をカウントします。 grep "physical id" / proc/cpuinfo | sort -u physical id : 0 physical id : 1 **Note** CPU 使用率を抽出するために、 nmon や sar などの OS コマンドを使用することはお勧めしません。この理由は、これらの計算には他のプロセスの CPU 使用率が含まれて、データベースが使用する実際の CPU 使用率を反映していない可能性があるためです。 AWR レポートを使用して使用可能なコアを推定するには AWR レポートの最初のセクションから CPU 使用率を導き出すこともできます。以下はレポートからの抜粋です。	

タスク	説明	必要なスキル
	D DB インスタンス Inst 起動時間 [Rel (リリース) RA N Id num X <DB XXX> 1 2021年9月5日23:(12.1.0 いいえ ホスト名 プラットフォーム Cpu コア ソケット メモリ (GB) < hose > Linux x86 64 ビット 80 80 2 441.7 この例では、CPU 数は 80 で、これは論理 (仮想) CPU であることを示しています。また、この構成には 2 つのソケットがあり、各ソケットに 1 つの物理プロセッサ (合計で 2 つの物理プロセッサ) があ	

タスク	説明	必要なスキル
	り、物理プロセッサまたはソケットごとに 40 コアがあることも確認できます。	

タスク	説明	必要なスキル
オプション 2: OS 統計を使用して CPU 使用率を推定します。	OS で(sarやその他のホスト OS ユーティリティを使用して)、またはAWR レポートのオペレーティングシステム統計セクションから IDLE/(IDLE+BUSY) 値をレビューすることで、OS の CPU 使用率統計をチェックできます。v$osstat から直接消費された CPU の秒数を確認できます。AWR レポートと Statspack レポートでも、オペレーティングシステム統計セクションでこのデータが表示されます。 同じボックスに複数のデータベースがある場合、BUSY_TIME の v$osstat 値はすべて同じになります。 Statistic 値 最終値 FREE_MRY_BYT 6,810,67,248 12,280,799,232 INACTIVMEMORTES 175、62333、632 160、380、53、568 SWAP_F_BYTES 17,145,64,336 17,145,872,384 BUSY_T 1,305,56,937	DBA

タスク	説明		必要なスキル
	IDLE_TI	4,312,71 ,839	
	IOWAIT_ ME	53,417,1 4	
	NICE_TI	29,815	
	SYS_TIM	148,567, 70	
	USER_T	1,146,91 ,783	
	LOAD	25	29
	VM_IN_E ES	593,920	
	OUT_BY	327,680	
	PHYSIC/ MEMOR TES	474、36 17、152	
	NUM_CU	80	
	NUM_CF ORES	80	
	NUM_CF OCKETS	2	
	GLOBAL CEIVE_S E_MAX	4,194,30	

タスク	説明	必要なスキル
	GLOBAL_ND_SIZE_MAX 2,097,15 TCP_RE_VE_SIZE_EFAULT 87,380 TCP_RE_VE_SIZE_MAX 6,291,45 TCP_RE_VE_SIZE_MIN 4,096 TCP_SE_SIZE_DEFAULT 16,384 TCP_SE_SIZE_MA 4,194,30 TCP_SE_SIZE_MI 4,096 システムに他の CPU 使用者がいない場合は、次の式を使用して CPU 使用率を計算します: 使用率 = ビジータイム/合計時間 ビジータイム = 要件 = v$osstat.Busy_Time	

タスク	説明	必要なスキル
	C = 合計時間 (ビジー+アイドル) C = 容量 = v$OSTAT.Busy_Time + v$OSTAT.Idle_Time 使用率 = BUSY_TIME / (BUSY_TIME + IDLE_TIME) = -1,305,569,937/(1,305,569,937 + 4,312,718,839) = 23% 利用	

タスク	説明	必要なスキル
オプション 3: データベースメトリクスを使用して CPU 使用率を推定します。	システム内で複数のデータベースが実行されている場合、レポートの先頭に表示されるデータベースメトリックを使用できます。 スナップ ID / スナップタイム / セッション / カーソル/セッション スナップを開始: 1846 / 2020年9月28日09:0(/ 1226 / 35.8 エンドスナップ: 1854 / 2020年10月6日13:0(/ 1876 / 41.1 経過時間: / 11,7((分) / /	DBA

タスク	説明	必要なスキル
	DB 時間: 312,(0 (分) CPU 使用率メトリクスを取得するには、次の式を使用します: データベースの CPU 使用率 (使用可能な CPU パワーの%) = CPU 時間 /N UM_CPUS / 経過時間 CPU 使用率は CPU 時間で表され、CPU の待機時間の代わりに、 CPU に費やされた時間を示します。この計算結果は次のようになります : = 312,625.40/11,759.64/80 = CPU の 33% が使用されています コア数 (33%) * 80 = 26.4 コア 合計コア数 = 26.4 * (120%) = 31.68 コア これら 2 つの値のうち大きい方を使用して、Amazon RDS または Aurora DB インスタンスの CPU 使用率を計算できます。	

タスク	説明	必要なスキル
	Note IBM AIX では、計算された使用率がオペレーティングシステムまたはデータベースの値と一致しません。これらの値は、他のオペレーティングシステムでは一致します。	

メモリ要件の推定

タスク	説明	必要なスキル
メモリ統計情報を使用してメモリ要件を推定します。	ソースデータベースのメモリを計算し、それをターゲットデータベースでマッチさせるのに、AWR レポートを使用できます。また、コストを節約するのに、既存のデータベースのパフォーマンスを確認してメモリ要件を減らすか、パフォーマンスを向上させるために要件を増やす必要もあります。それには、AWR の応答時間とアプリケーションのサービスレベルアグリーメント (SLA) を詳細に分析する必要があります。Oracle のシステムグローバルエリア (SGA) とプログラムグローバルエリア (PGA) の使用量の合計を、Oracle の推定メモリ使用	DBA

タスク	説明	必要なスキル
	率として使用します。OS にさらに20% を追加して目標メモリサイズ要件を決定します。Oracle RAC では、すべての RAC ノードの推定メモリ使用率の合計を使用して、合計メモリを減らします。理由は、メモリが共通ブロックに格納されるためです。 1. インスタンス効率パーセンテージのメトリックを確認します。この表では、次の用語を使用します： • Buffer Nowait % は、物理的な I/O を実行せずにバッファキャッシュで特定のブロックが見つかった回数の割合で、パフォーマンス向上には 100% を目標とします。 • Buffer Nowait % は 100% 近くになるはずです。 • Latch Hit % は 100% に近いはずです。 • Non-Parse CPU の %は、非解析アクティビティに費やされた CPU 時間の割合です。この値は 100% に近いはずです。 インスタンス効率の割合 (目標 100%)	

タスク	説明	必要なスキル
	Buffer Nowa %: 99.99 Redo NoWa %: 100.00 Buffer Hit %: 99.84 インメモリソート %: 100.00 ライブラリー %: 748.7 ソフト解析 %: 99.81 Execu to Parse %: 96.61 ラッチヒット %: 100.00 Parse CPU to Parse Elaps %: 72.73 Non-Parse CPU の率 %: 99.21 フラッ 0.00	

タスク	説明	必要なスキル
	シュキャッシュのヒット率%: この例では、すべてのメトリクスに問題がないように見えるため、既存のデータベースの SGA と PGA をキャパシティプランニングの要件として使用できます。 2. メモリ統計セクションを確認し、SGA/PGA を計算します。 開始 修了 ホストメモリ(MB): 452,387 452,387.3 SGAの使用(MB): 220,544 220,544.0	

タスク	説明	必要なスキル
	PGA 使用量 (MB): 36,874.9 45,270.0 使用中のインスタンスメモリの合計 = SGA + PGA = 220 GB + 45 GB = 265 GB バッファーの 20% を追加: 合計インスタンスのメモリ = 1.2 * 265 GB = 318 GB SGA と PGA がホストメモリの 70% を占めるため、必要な合計メモリ量は次のようになります: ホストメモリの総容量 = 318/0.7 = 464 GB **Note** Amazon RDS for Oracle に移行すると、PGA と SGA は事前定義された式に基づいて事前に計算されます。事前に計算された値が、推定値に近いことを確保します。	

ターゲットデータベースの DB インスタンスタイプを決定

タスク	説明	必要なスキル
ディスク I/O、CPU、メモリの見積もりに基づいて DB インスタンスタイプを決定します。	前のステップの見積もりに基づいて、ターゲットの Amazon RDS または Aurora データベースの容量は次のようになるはずです: • CPU の 68 コア • スループットの 143 MBPS • ディスク I/O の 4367 IOPS • 464 GB のメモリ ターゲットの Amazon RDS または Aurora データベースでは、これらの値を db.r5.16x large インスタンスタイプにはマッピングできます。このインスタンスタイプは 32 コアのキャパシティ、512 GB の RAM、13,600 Mbps のスループットがあります。 詳細については、AWS ブログ記事「Oracle のパフォーマンスメトリクスに基づく Amazon RDS インスタンスを大規模に適切なサイズ」を参照してください。	DBA

関連リソース

- 「Aurora DB インスタンスクラス」 (Amazon Aurora ドキュメント)
- 「Amazon RDS DB インスタンスストレージ」 (Amazon RDS ドキュメント)

- 「AWS マイナーツール」 (GitHub リポジトリ)

AWS DMS を使用して Amazon RDS for SQL Server テーブルを S3 バケットにエクスポートする

作成者: Subhani Shaik (AWS)

概要

SQL Server 用 Amazon Relational Database Service (Amazon RDS) は、Amazon Web Services (AWS) クラウド上の他の DB エンジンにリンクされたサーバーへのデータのロードをサポートしていません。代わりに、AWS Database Migration Service (AWS DMS) を使用して Amazon RDS for SQL Server テーブルを Amazon Simple Storage Service (Amazon S3) バケットにエクスポートできます。これにより、データは他の DB エンジンで利用できるようになります。

AWS DMS は、AWS にデータベースを簡単かつ安全に移行します。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS は、広く普及しているほとんどの商用データベースとオープンソースデータベース間のデータ移行にご利用いただけます。

このパターンでは、AWS DMS エンドポイントの構成時に AWS Secrets Manager を使用します。Secrets Manager は、アプリケーション、サービス、IT リソースへのアクセスに必要なシークレットの保護に役立ちます。このサービスを使うと、ライフサイクルを通じてデータベース認証情報、API キー、その他のシークレットをローテーション、管理、取得することができます。ユーザーとアプリケーションは Secrets Manager を呼び出すことでシークレットを取得できるため、機密情報をハードコーディングする必要がなくなります。Secrets Manager には、Amazon RDS、Amazon Redshift、Amazon DocumentDB の統合機能が組み込まれたシークレットローテーションが用意されています。また、このサービスは API キーや OAuth トークンなど、他のタイプのシークレットにも拡張できます。Secrets Manager を使用すると、AWS クラウド、サードパーティサービス、およびオンプレミスのリソースに対するきめ細かな権限設定やシークレットローテーション監査の一元化を通して、シークレットへのアクセスを制御できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- S3 バケット
- 仮想プライベートクラウド (VPC)
- DB サブネット

- Amazon RDS for SQL Server
- Amazon RDS インスタンスに代わって S3 バケットへのアクセス (オブジェクトのリスト、取得、および配置) を行う AWS Identity and Access Management (IAM) ロール。
- RDS インスタンスの認証情報を保存する Secrets Manager。

アーキテクチャ

テクノロジースタック

- Amazon RDS for SQL Server
- AWS DMS
- Amazon S3
- AWS Secrets Manager

ターゲットアーキテクチャ

次の図は、AWS DMS を使用して Amazon RDS インスタンスから S3 バケットにデータをインポートするアーキテクチャを示しています。

1. ソースエンドポイントを介してソース Amazon RDS インスタンスに接続する AWS DMS 移行タスク
2. ソース Amazon RDS インスタンスからデータをコピーする
3. ターゲットエンドポイントを介してターゲット S3 バケットに接続する AWS DMS 移行タスク
4. コピーしたデータを CSV (カンマ区切り値) 形式で S3 バケットにエクスポートする

ツール

AWS サービス

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。

その他のサービス

- 「Microsoft SQL Server Management Studio (SSMS)」 は、SQL Server コンポーネントへのアクセス、設定、管理など、SQL Server を管理するためのツールです。

エピック

Amazon RDS for SQL Server インスタンスの構成

タスク	説明	必要なスキル
Amazon RDS for SQL Server インスタンスを作成します。	1. AWS マネジメントコンソールを開いて [RDS] を選択し、[標準作成] オプションを使用して必要なエディション (SQL Server Express エディション、SQL Server スタンダードエディション、SQL Server エンタープライズエディションなど) で Amazon RDS インスタンスを作成します。バージョンは、2016 以降を選択してください。 2. テンプレートで [開発/テスト] を選択します。	DBA、DevOps エンジニア

タスク	説明	必要なスキル
インスタンスの認証情報を設定する。	1. インスタンスの名前を入力します。 2. Amazon RDS インスタンスの ユーザー名とパスワードを入力します。	DBA、DevOps エンジニア

タスク	説明	必要なスキル
インスタンスクラス、ストレージ、自動スケーリング、可用性を構成します。	1. DB インスタンスクラスを次のリストから選択します: [標準]、[メモリ最適化]、[バースト可能] クラス。この DB インスタンスに予定されているワークロードに必要な計算、ネットワーク、メモリ容量を割り当てる DB インスタンスタイプを選択します。詳細については、ドキュメントを参照してください。 2. 次のリストからストレージタイプを選択します: [汎用 SSD]、[プロビジョンド IOPS SSD]、または [マグネティック]。必要に応じてデフォルトのストレージサイズを割り当てます。 3. キャパシティプランニングに基づいて Amazon RDS ストレージを増やすには、[自動スケーリングを有効にする] を選択します。 4. レプリケーションインスタンスを備えたマルチ AZ 配置は AWS DMS でサポートされています。アベイラビリティゾーン、内部ハードウェア、またはネットワークで障害が発生した場合、AWS DMS はスタンバイインスタンスを作成し、スタンバイレプリカへの自	DBA、DevOps エンジニア

タスク	説明	必要なスキル
	動フェイルオーバーにより高可用性 (HA) を実現します。インポートのサイズに応じて、適切なオプションを選択します。	
VPC、サブネットグループ、パブリックアクセス、セキュリティグループを指定します。	必要に応じて [VPC]、[DB サブネットグループ]、[VPC セキュリティグループ] を選択し、Amazon RDS インスタンスを作成します。次のようなベストプラクティスに従ってください。 • RDS DB インスタンスへのパブリックアクセスを有効にしない。 • CIDR 0.0.0.0/0 をセキュリティグループで使用しない。 • RDS インスタンスへのアクセスには、必要な IP アドレスとポート情報のみを使用する。	DBA、DevOps エンジニア

タスク	説明	必要なスキル
モニタリング、バックアップ、メンテナンスを構成する。	1. 必要なバックアップオプションを指定します。デフォルトでは、自動バックアップは 7 日の保持期間で有効になっています。 2. 適切な [自動マイナーバージョンアップグレード] と [メンテナンスウィンドウ] 設定を選択し、保留中の変更またはメンテナンスを Amazon RDS によるデータベースに適用します。 3. [データベースの作成] を選択します。	DBA、DevOps エンジニア

データベースと例データのセットアップ

タスク	説明	必要なスキル
テーブルを作成し、例データをロードします。	新しいデータベース内にテーブルを作成します。[追加情報] セクションのサンプルコードを使用して、テーブルにデータをロードします。	DBA、DevOps エンジニア

認証情報の設定

タスク	説明	必要なスキル
シークレットを作成します。	1. コンソールで[Secrets Manager] コンソールを選択し、[新しいシークレットを保存] を選択します。	DBA、DevOps エンジニア

タスク	説明	必要なスキル
	2. Amazon RDS for SQL Server データベースのユーザー名とパスワードを入力します。 このシークレットは AWS DMS ソースエンドポイントに使用されます。	

データベースと S3 バケット間のアクセスを設定する

タスク	説明	必要なスキル
Amazon RDS にアクセスするための IAM ロールを作成します。	1. コンソールで [IAM] を選択し、S3 バケットに Amazon RDS への読み取り/書き込みアクセスを許可する IAM ロールを作成します。 2. [機能] で [S3 統合] を選択します。	DBA、DevOps エンジニア

S3 バケットの作成

タスク	説明	必要なスキル
S3 バケットを作成する。	Amazon RDS for SQL Server からのデータを保存するには、コンソールで [S3] を選択し、[バケットの作成] を選択します。S3 バケットがパブリックにアクセス可能でないことを確認します。	DBA、DevOps エンジニア

AWS DMS と S3 バケット間のアクセスを設定する

タスク	説明	必要なスキル
AWS DMS が Amazon S3 にアクセスするための IAM ロールを作成します。	AWS DMS が S3 バケットのオブジェクトをリスト、取得、および配置できるようにする IAM ロールを作成します。	DBA、DevOps エンジニア

AWS DMS を構成する

タスク	説明	必要なスキル
AWS DMS ソースエンドポイントを作成します。	1. コンソールで [データベース移行サービス] を選択し、[エンドポイント] を選択します。[RDS DB インスタンスを選択する] チェックボックスを選択して、ソースエンドポイントを作成します。 2. ソースエンジンには、[Microsoft SQL サーバー] を選択します。 3. [エンドポイントデータベースにアクセスする] で [AWS Secrets Manager] を選択し、先に作成したシークレットと IAM ロール、およびデータベース名を入力します。 4. ソースエンドポイントをテストします。	DBA、DevOps エンジニア
AWS DMS ターゲットエンドポイントを作成する。	ターゲットエンドポイントを作成し、ターゲットエンジン	DBA、DevOps エンジニア

タスク	説明	必要なスキル
	として Amazon S3 を選択します。 先に作成した IAM ロールの S3 バケット名とフォルダ名を指定します。	
AWS DMS レプリケーションインスタンスを作成する。	同じ VPC、サブネット、セキュリティグループで AWS DMS レプリケーションインスタンスを作成します。インスタンスクラスのオプションの選択に関する詳細については、 のドキュメントを参照してください。	DBA、DevOps エンジニア
AWS DMS 移行タスクを作成します。	Amazon RDS for SQL Server から S3 バケットにデータをエクスポートするには、データベース移行タスクを作成します。[Migration type (移行タイプ)] で [Migrate existing data (既存のデータを移行する)] を選択します。作成した AWS DMS エンドポイントとレプリケーションインスタンスを選択します。	DBA、DevOps エンジニア

データを S3 バケットにエクスポートする

タスク	説明	必要なスキル
データベース移行タスクを実行します。	データベース移行タスクを開始して、SQL Server テーブルデータをエクスポートしま	DBA、DevOps エンジニア

タスク	説明	必要なスキル
	す。このタスクでは、Amazon RDS for SQL Server のデータを CSV 形式で S3 バケットにエクスポートします。	

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースを削除します。	追加コストが発生しないように、コンソールを使用して次の順序でリソースを削除します。 1. 移行タスク 2. レプリケーションインスタンス 3. エンドポイント 4. S3 バケット 5. Database instance	DBA、DevOps エンジニア

関連リソース

- AWS DMS
- Amazon S3
- Amazon RDS for SQL Server
- Amazon S3 統合

追加情報

データベースとテーブルを作成し、例データを読み込むには、次のコードを使用します。

```
--Step1: Database creation in RDS SQL Server
```

```
CREATE DATABASE [Test_DB]
 ON  PRIMARY
( NAME = N'Test_DB', FILENAME = N'D:\rdsdbdata\DATA\Test_DB.mdf' , SIZE = 5120KB ,
 FILEGROWTH = 10%)
 LOG ON
( NAME = N'Test_DB_log', FILENAME = N'D:\rdsdbdata\DATA\Test_DB_log.ldf' , SIZE =
 1024KB , FILEGROWTH = 10%)
GO

--Step2: Create Table
USE Test_DB
GO
Create Table Test_Table(ID int, Company Varchar(30), Location Varchar(20))

--Step3: Load sample data.
USE Test_DB
GO
Insert into Test_Table values(1,'AnyCompany','India')
Insert into Test_Table values(2,'AnyCompany','USA')
Insert into Test_Table values(3,'AnyCompany','UK')
Insert into Test_Table values(4,'AnyCompany','Hyderabad')
Insert into Test_Table values(5,'AnyCompany','Banglore')
```

Aurora PostgreSQL の動的 SQL ステートメントの匿名ブロックを処理

作成者: anuradha chintha (AWS)

概要

このパターンでは、動的 SQL ステートメントで匿名ブロックを処理する場合に発生するエラーを回避する方法を示しています。AWS Schema Conversion Tool を使用して、Oracle データベースを Aurora PostgreSQL 互換エディションデータベースに変換するとエラーメッセージが表示されます。このエラーを回避するには、OUT バインド変数の値を知っている必要がありますが、SQL ステートメントの実行後まで OUT のバインド変数の値を知ることはできません。。このエラーは、AWS Schema Conversion Tool (AWS SCT) が動的 SQL ステートメント内部のロジックを理解していないことが原因です。AWS SCT は PL/SQL コード (つまり、関数、プロシージャ、パッケージ) の動的 SQL ステートメントを変換できません。

前提条件と制限

前提条件

- アクティブなAWS アカウント。
- 「Aurora PostgreSQL データベース (DB) インスタンス」
- Oracle DV インスタンスの Amazon Relational Database Service (Amazon RDS)の起動
- 「PostgreSQL インタラクティブターミナル (psql)」
- 「SQL *Plus」
- ターゲットデータベースにある AWS_ORACLE_EXT のスキーマ (「AWS SCT 拡張パック」の一部)
- 「AWS Schema Conversion Tool (AWS SCT)」 の最新バージョンおよび必要なドライバー

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース 10.g 以降のバージョン

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL
- Amazon RDS for PostgreSQL

- AWS Schema Conversion Tool (AWS SCT)

移行アーキテクチャ

次の図表は、AWS SCT と Oracle 0UT のバインド変数を使用して、アプリケーションコードをスキャンして埋め込み SQL ステートメントを探し、そのコードを Aurora データベースが使用することができる互換性のある形式に変換する方法を示しています。

この図表は、次のワークフローを示しています：

1. Aurora PostgreSQL をターゲットデータベースとして使用して、ソースデータベースの AWS SCT レポートを生成します。
2. 動的 SQL コードブロック (AWS SCT がエラーを発生させたブロック) 内の、匿名ブロックを特定します。
3. コードブロックを手動で変換し、ターゲットデータベースにコードをデプロイします。

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- 「OracleのAmazon Relational Database Service (Amazon RDS)」 によって、AWS クラウドで Oracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。
- 「AWS Schema Conversion Tool (AWS SCT)」 によって、ソースデータベーススキーマと大部分のデータベースコードオブジェクトを、ターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行を予測可能にします。

その他のツール

- 「pgAdmin」を使用して、データベースサーバーに接続して操作できます。
- 「Oracle SQL Developer」 は、Oracle データベース内のデータベースを開発および管理するために使用できる統合開発環境です。このパターンには 「SQL *Plus」または Oracle SQL Developer のいずれも使用できます。

エピック

Oracle ソースデータベースを設定する

タスク	説明	必要なスキル
Amazon RDS または Amazon EC2 で Oracle インスタンスを作成します。	Amazon RDS で Oracle DB インスタンスを作成するには、Amazon RDS ドキュメントの 「Oracle DB インスタンスの作成と Oracle DB インスタンスのデータベースへの接続」を参照してください。 Amazon Elastic Compute Cloud (Amazon EC2) で Oracle DB インスタンスを作成するには、AWS 規範ガイダンス ドキュメントの 「Oracle 向けAmazon EC2」を参照してください。	DBA
移行のデータベーススキーマとオブジェクトを作成します。	Amazon Cloud Directory を使用して、データベーススキーマを作成することができます。詳細については、クラウドディスカバリドキュメントの 「スキーマの作成」 を参照してください。	DBA
インバウンドとアウトバウンドのセキュリティグループを設定します。	セキュリティグループを作成して設定するには、Amazon RDS ドキュメントの「セキュリティグループによるアクセス制御」を参照してください。	DBA

タスク	説明	必要なスキル
データベースが実行されていることを確認します。	データベースのステータスを確認するには、Amazon RDS ドキュメントの「Amazon RDS イベントの表示」を参照してください。	DBA

ターゲットの Aurora PostgreSQL データベースを設定

タスク	説明	必要なスキル
Amazon RDS に Aurora PostgreSQL インスタンスを作成します。	Aurora PostgreSQL インスタンスを作成するには、Amazon RDS ドキュメントの「DB クラスターを作成して Aurora PostgreSQL DB クラスターのデータベースに接続」を参照してください。	DBA
インバウンドとアウトバウンドのセキュリティグループを設定します。	セキュリティグループを作成および設定するには、Aurora ドキュメントの「セキュリティグループの設定によりVPCのDBクラスターへのアクセスを提供する」を参照してください。	DBA
Aurora PostgreSQL データベースが実行されていることを確認します。	データベースのステータスを確認するには、Aurora ドキュメントの「Amazon RDS イベントの表示」を参照してください。	DBA

AWS SCT のセットアップ

タスク	説明	必要なスキル
AWS SCTをソースデータベースに接続する。	AWS SCT をソースデータベースに接続するには、AWS SCT ドキュメントの「ソースとして PostgreSQL に接続する」を参照してください。	DBA
AWS SCT をターゲットデータベースに接続します。	AWS SCT をターゲットデータベースに接続するには、AWS スキーマ変換ツールユーザーガイドの「AWS スキーマ変換ツールとは?」を参照してください。	DBA
AWS SCT でデータベーススキーマを変換し、自動変換されたコードを SQL ファイルとして保存します。	AWS SCT の変換されたファイルを保存するには、 AWS スキーマ変換ツールユーザーガイドの「AWS SCT での変換済みスキーマの保存と適用」を参照してください。	DBA

コードの移行

タスク	説明	必要なスキル
手動変換用の SQL ファイルを取得します。	AWS SCT で変換されたファイルで、手動変換が必要な SQL ファイルを引き出します。	DBA
スクリプトを更新します。	SQL ファイルを手動で更新します。	DBA

関連リソース

- 「Amazon RDS」
- 「Amazon Aurora の特徴量」

追加情報

次のサンプルコードは、Oracle ソースデータベースの設定方法を示しています：

```
CREATE or replace PROCEDURE calc_stats_new1 (
  a NUMBER,
  b NUMBER,
  result out NUMBER)
IS
BEGIN
result:=a+b;
END;
/
```

```
set serveroutput on ;

DECLARE
  a NUMBER := 4;
  b NUMBER := 7;
  plsql_block VARCHAR2(100);
  output number;
BEGIN
  plsql_block := 'BEGIN calc_stats_new1(:a, :b,:output); END;';
  EXECUTE IMMEDIATE plsql_block USING a, b,out output;
  DBMS_OUTPUT.PUT_LINE('output:'||output);

END;
```

次のサンプルコードは、ターゲット Aurora PostgreSQL データベースの設定方法を示しています:

```
 w integer,
 x integer)
RETURNS integer
AS
$BODY$
DECLARE
```

```
begin
return w + x ;
end;
$BODY$
LANGUAGE  plpgsql;


CREATE OR REPLACE FUNCTION test_pg.init()
RETURNS void
AS
$BODY$
BEGIN
if aws_oracle_ext.is_package_initialized
      ('test_pg' ) then
      return;
    end if;
    perform aws_oracle_ext.set_package_initialized
      ('test_pg' );

PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_output', NULL::INTEGER);
PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_status', NULL::text);
END;
$BODY$
LANGUAGE  plpgsql;


DO $$
declare
v_sql text;
v_output_loc int;
a integer :=1;
b integer :=2;
BEGIN
perform  test_pg.init();
--raise notice 'v_sql %',v_sql;
execute 'do $a$ declare v_output_l int; begin select * from test_pg.calc_stats_new1('||
a||','||b||') into v_output_l;
PERFORM aws_oracle_ext.set_package_variable(''test_pg'', ''v_output'', v_output_l) ;
 end; $a$'  ;
v_output_loc := aws_oracle_ext.get_package_variable('test_pg', 'v_output');
raise notice 'v_output_loc %',v_output_loc;
END ;
$$
```

Aurora PostgreSQL 互換にオーバーロードされた Oracle関数を処理

作成者: Sumana Yanamandra (AWS)

概要

オンプレミスの Oracle データベースから Amazon Aurora PostgreSQL 互換 エディションに移行するコードには、オーバーロードされた関数が含まれる場合があります。これらの関数の定義は同じです。つまり、関数名が同じで、入力 (IN) パラメータの数とデータ型は同じですが、データ型や 出力 (OUT) パラメータの数が異なることがあります。

これらのパラメータの不一致により、実行する関数を判断するのが難しくなる可能性があり、そのためPostgreSQLで問題を引き起こす可能性があります。このパターンでは、データベースコードを Aurora PostgreSQL 互換に移行する際に、オーバーロードされた関数を処理する方法を示しています。

前提条件と制限

前提条件

- ソースデータベースとしての Oracle データベースインスタンス
- ターゲットデータベースとしての Aurora PostgreSQL 互換 DB インスタンス (Aurora ドキュメントの「説明」 を参照)

製品バージョン

- Oracle データベース 9.i 以降
- Oracle SQL 開発者用バージョン 18.4.0.376
- pgAdmin 4 のクライアント
- Aurora PostgreSQL 互換バージョン 11 以降 (Aurora ドキュメントの「Amazon Aurora PostgreSQL のバージョンの識別」 を参照)

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。

その他のツール

- 「Oracle SQL Developer」 は、従来のデプロイメントとクラウドデプロイメントの両方で Oracle データベースの SQL を操作するための無料の統合開発環境です。
- 「pgAdmin」は、PostgreSQLのオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

エピック

シンプルな関数を作成する

タスク	説明	必要なスキル
PostgreSQL で、1 つの入力パラメータと 1 つの出力パラメータを持つ関数を作成します。	次の例は、Aurora PostgreSQL 互換の `test_overloading` という名前の関数を示しています。この関数には 2 つのパラメーターがあります。1 つは入力テキストパラメータ、もう 1 つは出力テキストパラメーターです。 `CREATE OR REPLACE FUNCTION public.test_overloading( str1 text, OUT str2 text) LANGUAGE 'plpgsql' COST 100 VOLATILE AS $BODY$ DECLARE BEGIN str2 := 'Success';`	データエンジニア、Aurora PostgreSQL 互換

タスク	説明	必要なスキル

	`RETURN ;`
	`EXCEPTION`
	`WHEN others THEN`
	`RETURN ;`
	`END;`
	`$BODY$;`	

PostgreSQL で関数を実行します。	前のステップで作成した関数を実行します。
	`select public.test_overloading('Test');`
	次の出力が表示されるはずです。
	`Success`
		データエンジニア、Aurora PostgreSQL 互換

関数をオーバーロードする

タスク	説明	必要なスキル
同じ関数名を使用して、PostgreSQL でオーバーロードされた関数を作成します。	Aurora PostgreSQL 互換で、前の関数と同じ関数名を使用するオーバーロード関数を作成します。次の例にも `test_overloading` という名前が付けられ、3 つのパラメーターがあります。1 つは入力テキストパラメータ、1 つは出力テキストパラメータ、最後の 1 つは出力整数パラメータです。	データエンジニア、Aurora PostgreSQL 互換

タスク	説明	必要なスキル

	CREATE OR REPLACE
	 FUNCTION public.te
	st_overloading(
	str1 text,
	OUT str2 text,
	OUT num1
	 integer)
	LANGUAGE 'plpgsql'
	
	COST 100
	VOLATILE
	AS $BODY$
	DECLARE
	str3 text;
	
	BEGIN
	
	str2 :=
	 'Success';
	num1 := 100;
	
	RETURN ;
	EXCEPTION
	WHEN others THEN
	RETURN ;
	END;
	$BODY$;	

タスク	説明	必要なスキル

PostgreSQL で関数を実行します。	この関数を実行して、次のエラーメッセージが表示されて失敗します。
	
	`ERROR: cannot change return type of existing function`
	`HINT: Use DROP FUNCTION test_overloading(text) first.`
	
	これは、Aurora PostgreSQL 互換で関数のオーバーロードを直接に適用されない原因です。2 番目のバージョンの関数では入力パラメータは同じでも、出力パラメータの数が異なるため、実行する関数を特定できません。
		データエンジニア、Aurora PostgreSQL 互換

回避策を適用する

タスク	説明	必要なスキル
INOUT を最初の出力パラメータに追加します。	回避策として、最初の出力パラメータを INOUT として表現することにより関数コードを変更します。 `CREATE OR REPLACE FUNCTION public.test_overloading( str1 text, INOUT str2 text,`	データエンジニア、Aurora PostgreSQL 互換

タスク	説明	必要なスキル

	<pre>OUT num1
	integer)
	LANGUAGE 'plpgsql'
	
	COST 100
	VOLATILE
	AS $BODY$
	DECLARE
	str3 text;
	BEGIN
	
	str2 :=
	'Success';
	num1 := 100;
	
	RETURN ;
	EXCEPTION
	WHEN others THEN
	RETURN ;
	END;
	$BODY$;</pre>	

修正済みの関数を実行します。	次のクエリを使用して、更新された関数を実行します。この関数の 2 番目の引数には NULL 値を渡します。この理由は、エラーを回避するためにこのパラメータを INOUT と宣言したためです。
	
	<pre>select public.te
	st_overloading('Te
	st', null);</pre>
	これで、この関数が正常に作成されました。
	
	<pre>Success, 100</pre>
		データエンジニア、Aurora PostgreSQL 互換

タスク	説明	必要なスキル
結果を検証します。	オーバーロードされた関数が付いたコードが正常に変換されたことを確認します。	データエンジニア、Aurora PostgreSQL 互換

関連リソース

- 「Amazon Aurora PostgreSQL の操作」(Aurora ドキュメント)
- 「Oracle での関数のオーバーロード」(Oracle ドキュメント)
- 「PostgreSQL の関数のオーバーロード」(PostgreSQL ドキュメント)

DynamoDB でのタグ付けの強制を支援

作成者: Mansi Suratwala (AWS)

概要

このパターンは、事前定義の Amazon DynamoDB タグがAmazon Web Services (AWS) クラウドの DynamoDB リソースから見つからず、または削除された場合、自動通知を設定します。

DynamoDB は、高速で予測可能なパフォーマンスとスケーラビリティを実現するフルマネージド NoSQL データベースサービスです。DynamoDB では、分散データベースの運用とスケーリングに伴うユーザーの管理上の負担を軽減できます。DynamoDB を使用して、ハードウェアのプロビジョニング、セットアップと構成、レプリケーション、ソフトウェアパッチ適用、クラスタースケーリングなどを配慮しなくても良いです。

このパターンでは、Amazon CloudWatch Events イベントと AWS Lambda 関数を作成する AWS CloudFormation テンプレートを使用します。このイベントでは、AWS CloudTrail を使用して、新規または既存の DynamoDB タグ情報を監視します。定義済みのタグが見つからない、または削除された場合、CloudWatch は Lambda 関数をトリガーし、違反を通知する Amazon Simple Notification Service (Amazon SNS) 通知を送信します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- バケットLambda 関数を実行するための Python スクリプトを含む Lambda .zip ファイルの、Amazon Simple Storage Service (Amazon S3)

機能制限

- このソリューションは、`TagResource`または `UntagResource` のCloudTrail イベントが発生した場合にのみ機能します。他のイベントの通知は作成されません。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon DynamoDB

- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon S3
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

AWS CloudFormation テンプレートは、さまざまな AWS リージョンとアカウントに複数回使用できます。各リージョンまたはアカウントで、テンプレートを 1 回実行するだけで済みます。

ツール

ツール

- 「Amazon DynamoDB 」 — DynamoDBは、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。
- 「AWS CloudTrail」 — CloudTrailは、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査について支援する AWS のサービスです。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- 「Amazon CloudWatch Events」 — CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- AWS Lambda – Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- 「Amazon S3」 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS –Amazon Simple Notification Service (Amazon SNS) は、アプリケーション、エンドユーザー、およびデバイスでクラウドから通知を瞬時に送受信できるようにするウェブサービスです。

コード

- プロジェクトの .zip ファイルは添付ファイルとして入手できます。

エピック

S3 バケットを定義

タスク	説明	必要なスキル
S3 バケットを削除します。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを選択、作成します。この S3 バケットは Lambda コードの.zip ファイルをホストします。S3 バケットは、監視対象の DynamoDB リソースと同じ AWS リージョンに存在している必要があります。	クラウドアーキテクト

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
S3 バケットに Lambda コードをアップロードします。	「添付ファイル」セクションにある Lambda コードの.zip ファイルを S3 バケットにアップロードします。S3 バケットは、監視中の DynamoDB リソースと同じリージョンに存在している必要があります。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをデプロイします。	AWS CloudFormation コンソールで、添付ファイルセクションで提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を提供します。	クラウドアーキテクト

AWS CloudFormation テンプレートのパラメータを入力します

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成または選択した S3 バケットの名前を入力します。	クラウドアーキテクト
Amazon S3 キーを指定します。	S3 バケット内の Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例: `<folder>/<file-name>.zip`)。	クラウドアーキテクト
E メールアドレスを提供	Amazon SNS 通知を受信するための有効なメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベン	クラウドアーキテクト

タスク	説明	必要なスキル
	トを指定します。 `Warning` 潜在的に有害な状況を示します。	
必要な DynamoDB タグキーを入力します。	タグは必ず、それらの間にスペースを入れずにカンマで区切ります(例: `ApplicationId,CreatedBy,Environment,Organization`)。CloudWatch Events イベントはこれらのタグを検索し、見つからない場合は通知を送信します。	クラウドアーキテクト

サブスクリプションを確認します。

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定したメールアドレスに購読メールメッセージが送信されます。違反通知を受信するには、このEメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「S3 バケットの作成」
- ファイルを S3 バケットにアップロードする
- DynamoDB でのタグ付けのリソース
- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS DMS と Amazon Aurora によるクロスリージョンディザスタリカバリの実装

マークハドソン (AWS) によって作成された

概要

自然災害や人為的災害はいつでも発生し、特定の AWS リージョンで実行されているサービスとワークロードの可用性に影響を与える可能性があります。リスクを軽減するには、AWS サービスに組み込まれているクロスリージョン機能を組み込んだディザスタリカバリ (DR) 計画を立てる必要があります。本質的にクロスリージョン機能を備えていない AWS サービスの場合、DR プランは AWS リージョン間のフェイルオーバーを処理するソリューションも提供する必要があります。

このパターンでは、1 つのリージョンにある 2 つの Amazon Aurora MySQL対応版データベースクラスターを含むディザスタリカバリ設定について説明します。DR 要件を満たすため、データベースクラスターは Amazon Aurora Global Database機能を使用するように設定されており、1 つのデータベースが複数の AWS リージョンにまたがっています。AWS Database Migration Service (AWS DMS) タスクはローカルリージョン内のクラスター間でデータをレプリケートします。ただし、AWS DMS は現在、リージョン間のタスクフェイルオーバーをサポートしていません。このパターンには、この制限を回避し、両方のリージョンで AWS DMS を個別に設定するために必要なステップが含まれています。

前提条件と制限

前提条件

- 「Amazon Aurora Global Database」をサポートするプライマリ AWS リージョンとセカンダリ AWS リージョンを選択しました。
- プライマリリージョンの 1 つのアカウントにある 2 つの独立した Amazon Aurora MySQL 対応版データベースクラスター。
- Database インスタンスクラスは db.r5 以上 (推奨)
- 既存のデータベースクラスター間で継続的なレプリケーションを実行するプライマリリージョンの AWS DMS タスク。
- データベースインスタンスを作成するための要件を満たすための DR リージョンのリソースが整っている。VPC での DB インスタンスの操作の詳細については、「VPC でDB インスタンスを使用する」を参照してください。

機能制限

- Amazon Aurora Global Databaseの制限の全リストについては、「Amazon Aurora Global Databaseの制限」を参照してください。

製品バージョン

- Amazon Aurora MySQL 互換エディション 5.7 または 8.0 詳細については、「Amazon Auroraバージョン」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Aurora MySQL 互換エディションのグローバルデータベースクラスター
- AWS DMS

ターゲット アーキテクチャ

次の図は、2 つの AWS リージョンのグローバルデータベースを示しています。1 つはプライマリメインデータベースとレポーターデータベース、もう 1 つはセカンダリメインデータベースとレポーターデータベースです。

自動化とスケール

AWS CloudFormation を使用して、仮想プライベートクラウド (VPC)、サブネット、パラメータグループなどの前提となるインフラストラクチャをセカンダリリージョンに作成できます。AWS CloudFormation を使用して DR リージョンにセカンダリクラスターを作成し、グローバルデータベースに追加することもできます。CloudFormation テンプレートを使用してプライマリリージョンでデータベースクラスターを作成した場合は、それらを更新するか、追加のテンプレートを追加してグローバルデータベースリソースを作成できます。詳細については、「2 つの DB インスタンスを持つ Amazon Aurora DB クラスターの作成」と「Aurora MySQL 用のグローバルデータベースクラスターの作成」を参照してください。

最後に、フェイルオーバーイベントとフェイルバックイベントが発生した後に、CloudFormation を使用してプライマリリージョンとセカンダリリージョンに AWS DMS タスクを作成できます。詳細については、「AWS::DMS::ReplicationTask」を参照してください。

ツール

- Amazon Aurora は、MySQL および PostgreSQL と互換性のあるフルマネージドリレーショナルデータベースエンジンです。このパターンでは Amazon Aurora MySQL–Compatible Edition を使用しています。
- Amazon Aurora グローバルデータベースは、グローバルに分散されたアプリケーション向けに設計されています。1 つの Amazon Aurora Global Databaseが複数の AWS リージョンにまたがる場合があります。データベースのパフォーマンスに影響を与えずにデータを複製します。また、各リージョンで低レイテンシーで高速なローカル読み取りが可能になり、リージョン全体の障害からのディザスタリカバリが可能になります。
- AWS DMS は、1 回限りの移行または継続的なレプリケーションを提供します。継続的なレプリケーションタスクにより、ソースデータベースとターゲットデータベースの同期が保たれます。セットアップ後、進行中のレプリケーションタスクはソースの変更を最小限のレイテンシーでターゲットに継続的に適用します。データの検証や変換などの AWS DMS のすべての機能は、どのレプリケーションタスクでも利用できます。

エピック

プライマリリージョンの既存のデータベースクラスターを準備します。

タスク	説明	必要なスキル
データベースクラスターパラメータグループを変更します。	既存のデータベースクラスターパラメータグループで、`binlog_format` パラメータを「行」の値に設定して行レベルのバイナリロギングを有効にします。 AWS DMS では、継続的なレプリケーションまたは変更データキャプチャ (CDC) を実行する場合、MySQL 互換データベースの行レベルのバイナリロギングが必要です。詳細については、「AWS が管理する MySQL 対応データベース	AWS 管理者

タスク	説明	必要なスキル
	を AWS DMS のソースとして使用する」を参照してください。	

タスク	説明	必要なスキル

データベースのバイナリログの保持期間を更新します。	エンドユーザーデバイスにインストールした MySQL クライアントまたは Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用して、Amazon Relational Database Service (Amazon RDS) が提供する次のストアドプロシージャを実行します。ここで、XXはログを保持する時間数です。
	`call mysql.rds_set_configuration('binlog retention hours', XX)`
	次のコマンドを実行してこの設定を確認します。
	`call mysql.rds_show_configuration;`
	AWS が管理する MySQL 互換データベースは、バイナリログをできるだけ早く消去します。したがって、保持期間は、AWS DMS タスクが実行される前にログが消去されないように十分な長さにする必要があります。通常は 24 時間で十分ですが、この値は DR リージョンで AWS DMS タスクをセットアップするのに必要な時間に基づいている必要があります。
		DBA

プライマリリージョンの既存の AWS DMS タスクを更新する

タスク	説明	必要なスキル

AWS DMS タスクの ARN を記録します。	Amazon リソースネーム (ARN) を使用して、後で使用できるようにAWS DMS タスク名を取得します。AWS DMS タスク ARN を取得するには、コンソールでタスクを表示するか、次のコマンドを実行します。
	`aws dms describe-replication-tasks`
	ARN は次のようになります。
	`arn:aws:dms:us-east-1:<accountid>:task:AN6HFFMPM246XOZVEUHCNSOVF7MQCLTOZUIRAMY`
	最後のコロンの後の文字は、後のステップで使用されるタスク名に対応しています。
		AWS 管理者

既存の AWS DMS タスクを変更してチェックポイントを記録します。	AWS DMS は情報を含むチェックポイントを作成し、レプリケーションエンジンは変更ストリームの復旧ポイントを確認します。チェックポイント情報を記録するには、コンソールで次のステップを実施します。	AWS 管理者

タスク	説明	必要なスキル
	1. AWS DMS タスクを停止します。 2. タスクの JSON エディタを使用して TaskRecoveryTableEnabled パラメータを「true」に設定します。 3. AWS DMS タスクを開始します。	
チェックポイント情報を検証します。	クラスターのライターエンドポイントに接続された MySQL クライアントを使用して、レポーターデータベースクラスターの新しいメタデータテーブルをクエリし、そのテーブルが存在し、レプリケーション状態情報が含まれていることを確認します。以下のコマンドを実行してください。 `select * from awsdms_control.awsdms_txn_state;` ARN のタスク名は、この表のTask_Name 列にあるはずです。	DBA

両方の Amazon Aurora クラスターを DR リージョンに拡張する

タスク	説明	必要なスキル
DR リージョンに基本インフラを作成します。	Amazon Aurora クラスターの作成とアクセスに必要な基本コンポーネントを作成します。 • 仮想プライベートクラウド (VPC) • サブネット • セキュリティグループ • ネットワークアクセスコントロールリスト • サブネットグループ • DB パラメータグループ • DB クラスターのパラメータグループ 両方のパラメータグループの設定がプライマリリージョンの設定と一致していることを確認します。	AWS 管理者
両方の Amazon Aurora クラスターに DR リージョンを追加します。	メインクラスターとレポーターの Amazon Aurora クラスターにセカンダリージョン (DR リージョン) を追加します。詳細については、「Amazon Aurora Global DatabaseにAWS リージョンを追加」を参照してください。	AWS 管理者

フェイルオーバーを実行

タスク	説明	必要なスキル
AWS DMS タスクを停止します。	プライマリリージョンの AWS DMS タスクはフェイルオーバーが発生すると正しく機能しなくなるため、エラーを回避するために停止する必要があります。	AWS 管理者
マネージドフェイルオーバーを実行してください。	メインデータベースクラスターの DR リージョンへのマネージドフェイルオーバーを実行します。詳細については、「Amazon Aurora Global Databaseのマネージドプラン済みフェイルオーバーを実行」を参照してください。メインデータベースクラスターのフェイルオーバーが完了したら、レポーターデータベースクラスターで同じアクティビティを実行します。	AWS 管理者、データベース管理者
メインデータベースにデータをロードします。	DR データベースクラスター内のメインデータベースのライターノードにテストデータを挿入します。このデータは、レプリケーションが正常に機能していることを確認するために使用されます。	DBA
AWS DMS レプリケーションインスタンスを作成します。	DR リージョンに AWS DMS レプリケーションインスタンスを作成するには、「レプリ	AWS 管理者、データベース管理者

タスク	説明	必要なスキル
	ケーションインスタンスの作成」を参照してください。	
AWS DMS ソースおよびターゲットエンドポイントを作成します。	DR リージョンに AWS DMS ソースエンドポイントとターゲットエンドポイントを作成するには、「ソースエンドポイントとターゲットエンドポイントの作成」を参照してください。ソースは、メインデータベースクラスターのライターインスタンスを指している必要があります。ターゲットは、Reporter Database クラスターのライターインスタンスを指します。	AWS 管理者、データベース管理者
レプリケーションチェックポイントを取得します。	レプリケーションチェックポイントを取得するには、MySQL クライアントを使用して DR リージョンのレポーターデータベースクラスターのライターノードに対して以下を実行してメタデータテーブルをクエリします。 `select * from awsdms_control.awsdms_txn_state;` 表で、2 番目のエピックで取得したプライマリリージョンに存在する AWS DMS タスクの ARN に対応する task_name 値を見つけます。	DBA

タスク	説明	必要なスキル
AWS DMS タスクを作成します。	コンソールを使用して、DR リージョンに AWS DMS タスクを作成します。タスクでは、「データ変更のみ複製」の移行方法を指定します。詳細については、「タスクの作成」を参照してください。 1. タスク設定で、ウィザードを使用して以下を指定します。 • ソーストランザクションの CDC 開始モード — カスタム CDC 開始モードを有効にします。 • ソーストランザクションのカスタム CDC スタートポイント — リカバリチェックポイントを指定する 2. 「リカバリチェックポイント」ボックスに、`awsdms_txn_state` テーブルのデータベースクエリによって以前に取得したレプリケーションチェックポイント値を入力します。 3. タスク設定セクションで JSON エディターを選択し、「TaskRecoveryTableEnabled」パラメーターを「true」に設定します。	AWS 管理者、データベース管理者

タスク	説明	必要なスキル
	AWS DMS タスクの「移行タスクを開始する」設定を「作成時に自動」に設定します。	
AWS DMS タスクの ARN を記録します。	ARN を使用して、後で使用するための AWS DMS タスク名を取得します。AWS DMS タスク ARN を取得するには、次のコマンドを実行します。 `aws dms describe-replication-tasks`	AWS 管理者、データベース管理者
複製されたデータを検証します。	DR リージョンのレポーターデータベースクラスターにクエリを実行して、メインデータベースクラスターに読み込んだテストデータが複製されていることを確認します。	DBA

フェイルバックを実行する

タスク	説明	必要なスキル
AWS DMS タスクを停止します。	DR リージョンの AWS DMS タスクは、フェイルバックが発生すると正しく機能しなくなるため、エラーを回避するために停止する必要があります。	AWS 管理者
マネージドフェイルバックを実行してください。	メインデータベースクラスターをプライマリリージョンにフェイルバックします。詳細については、「Amazon	AWS 管理者、データベース管理者

タスク	説明	必要なスキル
	Aurora Global Databaseのマネージドプラン済みフェイルオーバーを実行」を参照してください。メインデータベースクラスターのフェイルバックが完了したら、レポーターデータベースクラスターで同じアクティビティを実行します。	
レプリケーションチェックポイントを取得します。	レプリケーションチェックポイントを取得するには、MySQL クライアントを使用して DR リージョンのレポーターデータベースクラスターのライターノードに対して以下を実行してメタデータテーブルをクエリします。 `select * from awsdms_control.awsdms_txn_state;` 表で、4 番目のエピックで取得した DR リージョンに存在する AWS DMS タスクの ARN に対応する`task_name`値を見つけます。	DBA

タスク	説明	必要なスキル
AWS DMS ソースとターゲットのエンドポイントを更新します。	データベースクラスターがフェイルバックしたら、プライマリリージョンのクラスターをチェックして、どのノードがライターインスタンスであるかを判断します。次に、プライマリリージョンの既存の AWS DMS ソースエンドポイントとターゲットエンドポイントがライターインスタンスを指していることを確認します。そうでない場合は、ライターインスタンスのドメインネームシステム (DNS) 名でエンドポイントを更新します。	AWS 管理者

タスク	説明	必要なスキル
AWS DMS タスクを作成します。	コンソールを使用して、プライマリリージョンに AWS DMS タスクを作成します。タスクでは、「データ変更のみ複製」の移行方法を指定します。詳細については、「タスクの作成」を参照してください。 1. タスク設定で、ウィザードを使用して以下を指定します。 • ソーストランザクションの CDC 開始モード — カスタム CDC 開始モードを有効にします。 • ソーストランザクションのカスタム CDC スタートポイント — リカバリチェックポイントを指定する 2. 「リカバリチェックポイント」ボックスに、 `awsdms_txn_state` テーブルのデータベースクエリによって以前に取得したレプリケーションチェックポイント値を入力します。 3. また、タスク設定セクションで JSON エディターを選択し、「TaskRecoveryTableEnabled」パラ	AWS 管理者、データベース管理者

タスク	説明	必要なスキル
	メーターを「true」に設定します。 4. 最後に、AWS DMS タスクの「移行タスクを開始する」設定を「作成時に自動」に設定します。	
AWS DMS タスク Amazon リソースネーム (ARN) を記録します。	ARN を使用して、後で使用するための AWS DMS タスク名を取得します。AWS DMS タスク ARN を取得するには、次のコマンドを実行します。 `aws dms describe-replication-tasks` タスク名は、別のマネージドフェイルオーバーを実行するときや DR シナリオ中に必要になります。	AWS 管理者、データベース管理者
AWS DMS タスクを削除します。	プライマリリージョンの元の (現在停止している) AWS DMS タスクと、セカンダリリージョンの既存の AWS DMS タスク (現在停止中) を削除します。	AWS 管理者

関連リソース

- Amazon Aurora DB クラスターの設定
- Amazon Aurora Global Database の使用
- Amazon Aurora MySQL の操作
- AWS DMS レプリケーション インスタンスを使用する

- AWS DMS エンドポイントの使用
- AWS DMS タスクでの作業
- AWS CloudFormation とは

追加情報

この例では Amazon Aurora Global Databaseを DR に使用しています。これは、目標復旧時間 (RTO) が 1 秒、目標復旧時点 (RPO) が 1 分未満であるためです。どちらも従来の複製ソリューションよりも低く、DR シナリオに最適です。

Amazon Aurora Global Databaseには、他にも次のような多くの利点があります。

- ローカルレイテンシーによるグローバル読み取り — 世界中の消費者は、ローカルリージョンの情報にローカルレイテンシーでアクセスできます。
- スケーラブルなセカンダリ Amazon Aurora DB クラスター — セカンダリクラスターは個別にスケーリングでき、最大 16 の読み取り専用レプリカを追加できます。
- プライマリからセカンダリの Amazon Aurora DB クラスターへの迅速なレプリケーション-Aurora Global Database によるレプリケーションは、プライマリ DB クラスターのパフォーマンスにほとんど影響しません。これはストレージレイヤーで発生し、クロスリージョンレプリケーションのレイテンシーは通常 1 秒未満です。

このパターンでは、レプリケーションにも AWS DMS を使用します。Amazon Aurora データベースにはリードレプリカを作成できるため、レプリケーションプロセスと DR セットアップを簡素化できます。ただし、データ変換が必要な場合や、ターゲットデータベースがソースデータベースにない追加のインデックスを必要とする場合に、AWS DMS は複製によく使用されます。

100 個以上の引数を持つ Oracle 関数とプロシージャを PostgreSQL に移行

作成者: Srinivas Potlachervoo (AWS)

概要

このパターンでは、100 を超える引数を持つ Oracle データベースの関数とプロシージャを PostgreSQL に移行する方法を示しています。たとえば、このパターンを使用して Oracle の関数とプロシージャを以下の PostgreSQL 互換の AWS データベースサービスのいずれかに移行できます。

- PostgreSQL の Amazon Relational Database Service (Amazon RDS)
- Amazon Aurora PostgreSQL 互換エディション

PostgreSQLには、100 個以上の引数を持つ関数やプロシージャが適用されません。回避策として、ソース関数の引数と一致するタイプフィールドを持つ新しいデータ型を定義できます。次に、カスタムデータ型を引数として使用する PL/pgSQL 関数を作成して実行できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 「Amazon RDS Oracle Database DB インスタンス」
- 「Amazon RDS for PostgreSQL DB インスタンス」 または 「Aurora PostgreSQL 互換 DB インスタンス」

製品バージョン

- Amazon RDS Oracle DB インスタンス バージョン 10.2 以降
- Amazon RDS PostgreSQL DB インスタンス バージョン 9.4 以降、または Aurora PostgreSQL 互換 DB インスタンスバージョン 9.4 以降
- Oracle SQL Developer バージョン 18 以降
- pgAdmin バージョン 4 以降

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS Oracle DB インスタンス バージョン 10.2 以降

ターゲットテクノロジースタック

- Amazon RDS PostgreSQL DB インスタンス バージョン 9.4 以降、または Aurora PostgreSQL 互換 DB インスタンス バージョン 9.4 以降

ツール

AWS サービス

- 「PostgreSQL の Amazon Relational Database Service (Amazon RDS)」 を使用して、AWS クラウドで PostgreSQL リレーショナルデータベース (DB) をセットアップ、運用、スケーリングができます。
- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。

その他のサービス

- 「Oracle SQL Developer」 は、従来のデプロイとクラウドベースのデプロイの両方で Oracle データベースの開発と管理を簡素化する統合開発環境です。
- 「pgAdmin」は PostgreSQL のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

ベストプラクティス

作成するデータ型が、ソース Oracle 関数またはプロシージャに含まれる型フィールドと一致することを保証します。

エピック

100 個以上の引数を持つ Oracle の関数またはプロシージャを実行する

タスク	説明	必要なスキル
100個以上の引数を持つ既存のOracle/plSQL 関数またはプロシージャを作成または識別する。	100個以上の引数を持つOracle/PLSQL 関数またはプロシージャを作成します。 -または- 100個以上の引数を持つ既存のOracle/plSQL 関数またはプロシージャを特定します。 詳細については、Oracle データベースのドキュメントのセクション「14.7 関数作成ステートメント」と「14.11 プロシージャステートメントの作成」を参照してください。	Oracle/PL/SQL の知識
Oracle/PLSQL 関数またはプロシージャをコンパイルします。	Oracle/PLSQL 関数またはプロシージャをコンパイルします。 詳細については、Oracle データベースドキュメントの「関数のコンパイル」を参照してください。	Oracle/PL/SQL に関する知識
Oracle/plSQL 関数を実行します。	Oracle/PLSQL 関数またはプロシージャを実行します。次に、出力を保存します。	Oracle/PL/SQL の知識

ソース関数またはプロシージャの引数と一致する新しいデータ型を定義します。

タスク	説明	必要なスキル
PostgreSQL で新しいデータ型を定義します。	ソース Oracle 関数、またはプロシージャの引数と同じフィールドをすべて含む、新しいデータ型を PostgreSQL で定義します。 詳細については、PostgreSQL のドキュメントの「型の作成」を参照してください。	PostgreSQL PL/pgSQL の知識

新しい型引数を含む PostgreSQL 関数を作成します。

タスク	説明	必要なスキル
新しいデータ型を含む PostgreSQL 関数を作成します。	新しい TYPE 引数を含む PostgreSQL 関数を作成します。 サンプル関数を確認するには、このパターンの「追加情報」セクションを参照してください。	PostgreSQL PL/pgSQL の知識
PostgreSQL 関数をコンパイルします。	PostgreSQL で関数をコンパイルします。新しいデータ型フィールドがソース関数またはプロシージャの引数と一致すれば、関数は正常にコンパイルされます。	PostgreSQL PL/pgSQL の知識
PostgreSQL 関数を実行します。	PostgreSQL 関数を実行します。	PostgreSQL PL/pgSQL の知識

トラブルシューティング

問題	ソリューション
関数は、次のエラーを返します： エラー:「<ステート>」付近の構文エラー	関数のステートメントが、すべてセミコロン (;) で終わっていることを確認してください。
関数は、次のエラーを返します： エラー:「<statement>」 が既知の変数ではありません	関数ボディで使用されている変数が、関数の DECLARE セクションに確実にリストされているようにします。

関連リソース

- 「Amazon Aurora PostgreSQL との連携」(Amazon Aurora ユーザーガイド)
- 「型の作成」(PostgreSQL ドキュメント)

追加情報

タイプ引数を含むPostgreSQL関数の例

```
CREATE OR REPLACE FUNCTION test_proc_new
(
    IN p_rec type_test_proc_args
)
RETURNS void
AS
$BODY$
BEGIN

    /*
    **************
    The body would contain code to process the input values.
    For our testing, we will display couple of values.
    ***************
    */
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_acct_id: ', p_rec.p_acct_id);
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_ord_id: ', p_rec.p_ord_id);
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_ord_date: ', p_rec.p_ord_date);
```

```
END;
$BODY$
LANGUAGE plpgsql
COST 100;
```

Amazon RDS for Oracle DB インスタンスを AMS を使用する他のアカウントに移行する

作成者: Pinesh Singal (AWS)

概要

このパターンは、Oracle DB インスタンス用の Amazon Relational Database Service (Amazon RDS) を、ある AWS アカウントから別の AWS アカウントに移行する方法を示しています。このパターンは、ソース AWS アカウントが AWS Managed Services (AMS) を使用していないが、ターゲットアカウントが AMS を使用するシナリオに適用されます。AWS マネジメントコンソールを使用してデータベース操作を実行する代わりに、AMS の変更要求 (RFC)を使用して移行を完了できます。この方法では、トランザクション数が多い数テラバイトの Oracle ソースデータベースのダウンタイムを最小限にできます。たとえば、400 ~ 900 GB のデータベースのダウンタイムは約 2 ~ 3 時間かかる可能性があります。データベースの移行時間は、Amazon RDS for Oracle DB インスタンスのサイズに直接比例します。

> **Important**
>
> このパターンでは、ソースアカウントの Amazon RDS for Oracle DB インスタンスのデータベーススナップショットを作成し、そのスナップショットを AMS を使用しているターゲットアカウントにコピーしてから、RFCs を生成してそのスナップショットから新しい DB インスタンスを作成する必要があります。

前提条件と制限

前提条件

- ソースアカウントのアクティブな AWS アカウント
- ターゲットアカウントに AMS を使用するアクティブな AWS アカウント
- 稼働中の Amazon RDS for Oracle DB インスタンス

制限

- ソースアカウントの DB インスタンスの同じプロパティまたは設定が、AMS の新しいターゲット DB インスタンスにコピーされます。

- この移行アプローチで使用される RFC メソッドでは、Amazon RDS for Oracle をサポートする機能が限定されています。データベース移行をする AWS CloudFormation テンプレートを使用することで、Amazon RDS for Oracle の全機能にアクセスできます。
- 移行は予定されたダウンタイム中に完了する必要があるため、アプリケーションは数時間停止する可能性があります。ダウンタイム中は、ソースアカウントの DB インスタンスを停止し、ターゲットアカウントの新しい DB インスタンスに移行します。
- この移行方法は、同じ AWS アカウント内のある AWS リージョンから別のリージョンへの DB インスタンスの移行には適用されません。

製品バージョン

- Oracle Database Standard Edition 2 (SE2) 12.1.0.2.v2 インスタンスおよび Amazon RDS for Oracle 以降
- Amazon RDS for Oracle 11g はサポートされなくなりました (詳細については、Amazon RDS ドキュメントの Amazon RDS for Oracle を参照してください)。

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for Oracle の Oracle Database SE 2 12.1.0.2.v2 インスタンス
- Amazon RDS サブネットグループ
- Amazon RDS オプショングループ (必要な場合)
- Amazon RDS パラメータグループ (必要な場合)
- Amazon Virtual Private Cloud (Amazon VPC) セキュリティグループ
- AWS マネージドキーまたはカスタマーマネージドキーを搭載した AWS Key Management Service (AWS KMS)
- AWS Identity and Access Management (IAM) ロール (必要な場合)

ターゲットテクノロジースタック

- Amazon RDS for Oracle の Oracle Database SE 2 12.1.0.2.v2 インスタンス
- Amazon RDS サブネットグループ
- Amazon RDS オプショングループ (必要な場合)

- Amazon RDS パラメータグループ (必要な場合)
- Amazon VPC セキュリティグループ
- AWS Managed Services (AMS)
- AWS マネージドキーとカスタマーマネージドキーを搭載した AWS KMS
- IAM ロール (必要な場合)

ソースとターゲットの移行アーキテクチャ

次の図は、ある AWS アカウントの Amazon RDS for Oracle DB インスタンスを、AMS を使用する別の AWS アカウントの Amazon RDS for Oracle DB インスタンスに移行を示しています。

この図表は、次のワークフローを示しています：

1. ソースアカウントの Amazon RDS for Oracle DB インスタンスのデータベーススナップショットを作成します。
2. ターゲットアカウントの AMS にスナップショットをコピーします。
3. ターゲットアカウントのスナップショットから新しい Amazon RDS for Oracle DB インスタンスを作成します。

自動化とスケール

CloudFormation テンプレートと AMS で RFC を作成するを使用して、移行を自動化とスケールできます。CloudFormation を使用すると、スナップショットから Amazon RDS for Oracle DB インスタンスを作成するときに DB インスタンスを設定および復元する機能など、Amazon RDS for Oracle のすべての機能を使用できます。

ツール

- 「OracleのAmazon Relational Database Service (Amazon RDS)」によって、AWS クラウドで Oracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- AWS Managed Services (AMS) は、AWS インフラストラクチャをより効率的かつ安全に運用する上で役立ちます。

エピック

ターゲットアカウントでカットオーバーを準備する

タスク	説明	必要なスキル
カスタム AWS KMS キーを作成します。	1. KMS キーを作成 という自動 RFC を作成して、ターゲットアカウントからカスタム KMS キーを作成します。 2. **Note** カスタム KMS キーをソースアカウントと共有します。: Amazon RDS のデフォルトの AWS マネージドキー () を使用する Amazon RDS for Oracle DB インスタンスを共有することはできませんaws/rds。代わりに、KMS キーから DB インスタンスを再暗号化して DB インスタンスを共有します。	AWS、AMS
セキュリティグループを作成します。	セキュリティグループを作成 という自動 RFC を作成して、ターゲットアカウントから VPC のセキュリティグループを作成します。 以下を必ず指定します。	AWS、AMS

タスク	説明	必要なスキル
	• 新しいセキュリティグループ名 • TCP と UDP のイングレス/エグレスルール • 標準タグ	
(オプション) Amazon RDS リソースを確認します。	Amazon RDS for Oracle DB インスタンスが作成されると、次のリソースが作成されます。 • Amazon RDS サブネットグループ (サブネット ID に基づく) • Amazon RDS オプショングループ (ソース DB インスタンスのスナップショットに基づく) • Amazon RDS パタメータグループ (DB インスタンスのスナップショットに基づく) DB インスタンスの作成時に作成された Amazon RDS リソースを確認すると、Oracle DB インスタンスに接続し、Amazon RDS コンソールでサブネットグループ、オプショングループ、およびパラメータグループを検索できます。	AWS

ソースアカウントでカットオーバーする

タスク	説明	必要なスキル
アプリケーションを停止します。	アプリケーションとその依存サービスを停止します。ソースアカウントのデータベースへのトラフィックをすべて停止する必要があります。	アプリ所有者
手動スナップショットを撮ります。	手動でソースアカウントの Amazon RDS for Oracle DB インスタンスの DB スナップショットを作成します。	AWS
DB インスタンスを停止します。	Amazon RDS for Oracle DB インスタンスを停止します。	AWS
スナップショットをコピーします。	同じソースアカウントに DB スナップショットをコピーし、ターゲットアカウントから共有されているカスタム KMS キーを使用して、コピーした DB スナップショットファイルを再暗号化します。	AWS
スナップショットを共有します。	ターゲットアカウントと新しいスナップショット (カスタム KMS キーを使用してコピー) を共有します。	AWS

ターゲットアカウントでカットオーバーする

タスク	説明	必要なスキル
スナップショットをコピーします。	RDS スナップショットをコピー という自動 RFC を作成	AWS、AMS

タスク	説明	必要なスキル
	して DB スナップショットを同じターゲットアカウントにコピーし、再暗号化用に作成したデフォルトの AWS マネージド KMS キーを使用します。 これは、ターゲットアカウントを新しいスナップショットの所有者にし、必要に応じて、スナップショットから作成された Amazon RDS for Oracle DB インスタンスを有効化して、オプショングループに関連付ける場合に必要です。	
スナップショットから DB インスタンスを作成します。	スナップショットから DB を作成という自動 RFC を作成して、スナップショットから Amazon RDS for Oracle DB インスタンスを作成します。 以下を必ず指定します。 • 前の手順で作成した新しいスナップショットのID • VPC ID • サブネット ID • RDS インスタンス ID • 標準タグ	AWS、AMS

タスク	説明	必要なスキル
インスタンスをセキュリティグループにアタッチし、設定を更新します。	1. その他を更新という手動 RFC を作成して、以前に作成した Amazon RDS for Oracle DB インスタンスを、以前に作成した VPC セキュリティグループにアタッチします。 2. Amazon RDS for Oracle DB インスタンスの設定に追加変更をします。	AWS、AMS

タスク	説明	必要なスキル
DB インスタンスをテストします。	同じセキュリティグループでホストされているインスタンスまたはアプリケーションサーバーにログインし、telnet を使用して 1521 ポートに接続して、新しい Amazon RDS for Oracle DB インスタンスのエンドポイント接続をテストします。詳細については、Amazon RDS ドキュメントの Amazon RDS DB インスタンスへ接続するを参照してください。 **Note** プライマリユーザーのログイン認証情報が利用可能な場合は、任意の SQL クライアント (Oracle SQL Developer など) からログインして Amazon RDS for Oracle DB インスタンスをテストできます。	AWS、DBA

関連リソース

- AWS Managed Services (AWS ドキュメント)
- RFC の仕組み (AWS Managed Services ドキュメント)
- 暗号化されたスナップショットを共有する (Amazon RDS ユーザーガイド)

- 暗号化された Amazon RDS DB スナップショットを別のアカウントと共有する方法とは (AWS ナレッジセンター)
- Amazon Relational Database Service (Amazon RDS) とは （ Amazon RDS ユーザーガイド ）
- Amazon RDS for Oracle (Amazon RDS ユーザーガイド)
- AMS コンソールを使用する (AWS Managed Services ドキュメント)

追加情報

移行をロールバックする

移行をロールバックする場合は、以下の手順を完了します。

1. ターゲットアカウントから手動で RFC (その他を更新) を作成して、ターゲットアカウントで作成したデータベーススタックを削除します。
2. ソースアカウントの Amazon RDS for Oracle DB インスタンスを指すようにアプリケーション設定を更新します。
3. ソースアカウントで Amazon RDS for Oracle DB インスタンスを起動します。

Oracle の OUT バインド変数を PostgreSQL データベースに移行

作成者: Bikash Chandra Rout (AWS) と Vinay Paladi (AWS)

概要

このパターンでは、Oracleデータベース 0UT のバインド変数を、次の PostgreSQL 互換 AWS データベースサービスのいずれか1つに移行する方法を示します：

- PostgreSQL の Amazon Relational Database Service (Amazon RDS)
- Amazon Aurora PostgreSQL 互換エディション

PostgreSQLは、0UTのバインド変数をサポートしていません。Python ステートメントで同じ機能を取得するには、代わりに、GET と SET のパッケージ変数を使用するカスタム PL/pgSQL 関数を作成できます。これらの変数を適用するために、このパターンで提供されるサンプルのラッパー関数スクリプトでは、「AWS Schema Conversion Tool (AWS SCT) 拡張パック」 を使用しています。

> **Note**
>
> Oracle EXECUTE IMMEDIATEステートメントが最大 1 行を返すことができるSELECT ステートメントである場合は、次の操作を行うことをお勧めします。

- OUT バインド変数 (定義) をINTO 句に入れます。
- IN バインド変数をUSING 句に 入れます

詳細については、Oracle ドキュメントの「EXECUTE IMMEDIATE ステートメント」 を参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスのデータセンターにある Oracle データベース 10g (またはより新しい) のソースデータベース
- 「Amazon RDS for PostgreSQL DB インスタンス」 または 「Aurora PostgreSQL 互換 DB インスタンス」

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース 10g (またはそれ以降) データベース

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL DB インスタンスまたは Aurora PostgreSQL DB インスタンス

ターゲットアーキテクチャ

次の図は、Oracle Database の0UTバインド変数を PostgreSQL 互換の AWS データベースに移行するためのワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. AWS SCT は、ソースデータベーススキーマとカスタムコードの大部分を、ターゲット PostgreSQL 互換 AWS データベースと互換性のある形式に変換します。
2. 自動的に変換できないいずれかのデータベースオブジェクトには、PL/pgSQL 関数によってフラグが立てられます。フラグが立てられたオブジェクトが手動で変換され、移行が完了します。

ツール

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- PostgreSQL の Amazon Relational Database Service (Amazon RDS) を使用して、 AWS Cloud.でリレーショナルデータベース (DB) のセットアップ、運用、スケーリングができます。
- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコードを、ターゲットデータベースと互換性のある形式に自動的に変換することにより、異種データベースの移行をサポートします。
- 「pgAdmin」は.PostgreSQLのオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

エピック

カスタム PL/pgSQL 関数と AWS SCT を使用して Oracle OUT バインド変数を移行

タスク	説明	必要なスキル
PostgreSQL 互換 AWS データベースに接続します。	DB インスタンスの作成後に、標準の SQL クライアントアプリケーションを使用して DB クラスターのデータベースに接続できます。たとえば、「pgAdmin」 を使用して DB インスタンスに接続できます。 詳細については、以下のいずれかを参照してください： • Amazon RDS ユーザーガイドの「Amazon RDS DB インスタンスに接続」 • Amazon Aurora ユーザーガイドの「Amazon Aurora DB クラスターに接続」	移行エンジニア
このパターンのラッパー関数スクリプトの例を、ターゲットデータベースのメインスキーマに追加します。	このパターンの追加情報セクションから、PL/pgSQL ラッパー関数スクリプトの例をコピーします。次に、その関数をターゲットデータベースのメインスキーマに追加します。 詳細については、PostgreSQL のドキュメントの「機能の作成」を参照してください。	移行エンジニア

タスク	説明	必要なスキル
(オプション)ターゲットデータベースのメインスキーマの検索パスを更新して、Test_PG スキーマを含むようにします。	パフォーマンス向上のために、PostgreSQL の search_path 変数を更新して Test_PG スキーマ名が含まれるようにします。検索パスにスキーマ名を含めると、PL/pgSQL 関数を呼び出すたびに名前を指定する必要はありません。 詳細については、PostgreSQL のドキュメントのセクション「5.9.3 スキーマ検索パス」を参照してください。	移行エンジニア

関連リソース

- AWS スキーマ変換ツール
- 「アウトバインド変数」(Oracle ドキュメント)
- 「バインド変数を使用して SQL クエリのパフォーマンスを向上」(Oracle ブログ)

追加情報

PL/pgSQL 関数の例

```
/* Oracle */

CREATE or replace PROCEDURE test_pg.calc_stats_new1 (
                                                      a NUMBER,
                                                      b NUMBER,
                                                      result out NUMBER
                                                    )

IS
BEGIN
result:=a+b;
END;
```

```
/
/* Testing */
set serveroutput on
DECLARE
  a NUMBER := 4;
  b NUMBER := 7;
  plsql_block VARCHAR2(100);
  output number;
BEGIN
  plsql_block := 'BEGIN test_pg.calc_stats_new1(:a, :b,:output); END;';
  EXECUTE IMMEDIATE plsql_block USING a, b,out output;  -- calc_stats(a, a, b, a)
  DBMS_OUTPUT.PUT_LINE('output:'||output);
END;

output:11


PL/SQL procedure successfully completed.

--Postgres--

/* Example : 1 */
CREATE OR REPLACE FUNCTION test_pg.calc_stats_new1(
                                                    w integer,
                                                    x integer
                                                  )
RETURNS integer
AS
$BODY$
begin
      return w + x ;
end;
$BODY$
LANGUAGE  plpgsql;


CREATE OR REPLACE FUNCTION aws_oracle_ext.set_package_variable(
                                                                 package_name name,
                                                                 variable_name name,
                                                                 variable_value
 anyelement
                                                               )
    RETURNS void
    LANGUAGE 'plpgsql'
```

```
    COST 100
    VOLATILE
AS $BODY$
begin
    perform set_config
      ( format( '%s.%s',package_name, variable_name )
      , variable_value::text
      , false );
  end;
$BODY$;

CREATE OR REPLACE FUNCTION aws_oracle_ext.get_package_variable_record(
                                                                        package_name
 name,
                                                                        record_name name
                                                                      )
RETURNS text
LANGUAGE 'plpgsql'
    COST 100
    VOLATILE
AS $BODY$
begin
    execute 'select ' || package_name || '$Init()';

    return aws_oracle_ext.get_package_variable
        (
         package_name := package_name
        , variable_name := record_name || '$REC' );
end;
$BODY$;

--init()--
CREATE OR REPLACE FUNCTION test_pg.init()
RETURNS void
AS
$BODY$
BEGIN
if aws_oracle_ext.is_package_initialized('test_pg' ) then
      return;
    end if;
    perform aws_oracle_ext.set_package_initialized
      ('test_pg' );
PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_output', NULL::INTEGER);
```

```
PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_status', NULL::text);
END;
$BODY$
LANGUAGE  plpgsql;

/* callable for 1st Example */

DO $$
declare
v_sql text;
v_output_loc int;
a integer :=1;
b integer :=2;
BEGIN
perform  test_pg.init();
--raise notice 'v_sql %',v_sql;
execute 'do $a$ declare v_output_l int; begin select * from test_pg.calc_stats_new1('||
a||','||b||') into v_output_l;
PERFORM aws_oracle_ext.set_package_variable(''test_pg'', ''v_output'', v_output_l) ;
 end; $a$'  ;
v_output_loc := aws_oracle_ext.get_package_variable('test_pg', 'v_output');
raise notice 'v_output_loc %',v_output_loc;
END ;
$$

/*In above Postgres example we have set the value of v_output using v_output_l in the
 dynamic anonymous block to mimic the
  behaviour of oracle out-bind variable .*/

--Postgres Example : 2 --
CREATE OR REPLACE FUNCTION test_pg.calc_stats_new2(
 w integer,
 x integer,
inout status text,
 out result integer)
AS
$BODY$
DECLARE
begin
result :=  w + x ;
status := 'ok';
end;
$BODY$
LANGUAGE  plpgsql;
```

```
/* callable for 2nd Example */
DO $$
declare
v_sql text;
v_output_loc int;
v_staus text:= 'no';
a integer :=1;
b integer :=2;
BEGIN
perform  test_pg.init();
execute 'do $a$ declare v_output_l int; v_status_l text; begin select * from
 test_pg.calc_stats_new2('||a||','||b||','''||v_staus||''') into v_status_l,v_output_l;
PERFORM aws_oracle_ext.set_package_variable(''test_pg'', ''v_output'', v_output_l) ;
PERFORM aws_oracle_ext.set_package_variable(''test_pg'', ''v_status'', v_status_l) ;
end; $a$'  ;
v_output_loc := aws_oracle_ext.get_package_variable('test_pg', 'v_output');
v_staus := aws_oracle_ext.get_package_variable('test_pg', 'v_status');
raise notice 'v_output_loc %',v_output_loc;
raise notice 'v_staus %',v_staus;
END ;
$$
```

同じホスト名の SAP HSR を使用して SAP HANA を AWS に移行します

作成者：パラディープ・プリヤンパッタ (AWS)

概要

SAP HANA の Amazon Web Services (AWS) への移行は、バックアップと復元、エクスポートとインポート、SAP HANA システムレプリケーション (HSR) など、複数のオプションを使用して実行できます。どのオプションを選択するかは、ソースとターゲットの SAP HANA データベース間のネットワーク接続、ソースデータベースのサイズ、ダウンタイムの考慮事項、およびその他の要因によって異なります。

SAP HANA ワークロードをAWSに移行するための SAP HSR オプションは、ソースシステムとターゲットシステム間のネットワークが安定しており、SAP が SAP HSR のネットワークスループット要件として規定しているように、データベース全体（ SAP HANA DBレプリケーションスナップショット ）を1日以内に完全にレプリケートできる場合にうまく機能します。このアプローチによるダウンタイム要件は、ターゲット AWS 環境でのテイクオーバー、SAP HANA DB バックアップ、および移行後のタスクの実行に限定されます。

SAP HSR は、プライマリ (ソース) システムとセカンダリ (ターゲット) システム間のレプリケーショントラフィックに、異なるホスト名 (異なる IP アドレスにマップされたホスト名) の使用をサポートしています。そのためには、`global.ini`の`[system_replication_hostname_resolution]`セクションで特定のホスト名セットを定義してください。このセクションでは、プライマリサイトとセカンダリサイトのすべてのホストを各ホストで定義する必要があります。詳細な設定手順については、「SAP ドキュメンテーション」を参照してください。

この設定で重要な 1 つは、プライマリシステムのホスト名はセカンダリシステムのホスト名と異なっていなければならないということです。そうしないと、次のようなエラーが発生する可能性があります。

- `"each site must have a unique set of logical hostnames"`
- `"remoteHost does not match with any host of the source site. All hosts of source and target site must be able to resolve all hostnames of both sites correctly"`

ただし、移行後のステップの数は、ターゲット AWS 環境で同じ SAP HANA DB ホスト名を使用することで減らすことができます。

このパターンは、SAP HSR オプションを使用するときに、ソース環境とターゲット環境で同じホスト名を使用する場合の回避策となります。このパターンでは、SAP HANA ホスト名の名前変更オプションを使用できます。SAP HSR のホスト名が一意になるように、ターゲット SAP HANA DB に一時的なホスト名を割り当てます。移行によってターゲット SAP HANA 環境でのテイクオーバーマイルストーンが完了すると、ターゲットシステムのホスト名をソースシステムのホスト名に戻すことができます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 仮想プライベートクラウド (VPC) と、仮想プライベートネットワーク (VPN) エンドポイントまたはルーターを備えています。
- AWS Client VPN または は、ソースからターゲットにファイルを転送するように AWS Direct Connect 設定されています。
- ソース環境とターゲット環境の両方にある SAP HANA データベース。ターゲット SAP HANA DB パッチレベルは、同じ SAP HANA プラットフォームエディション内のソース SAP HANA DB パッチレベル以上である必要があります。たとえば、HANA 1.0 システムと HANA 2.0 システムの間でレプリケーションを設定することはできません。詳細については、SAP ノート：1999880 — よくある質問：SAP HANA システムレプリケーションの質問 15 を参照してください。
- ターゲット環境の SAP アプリケーションサーバー。
- ターゲット環境の、Amazon Elastic Block Store (Amazon EBS) ボリューム。

制約事項

次の SAP ドキュメントリストは、SAP HANA の動的階層化やスケールアウト移行に関する制約など、この回避策に関連する既知の問題について説明しています。

- 2956397 — SAP HANA データベースシステムの名前を変更できませんでした
- 2222694 — HANA システムの名前を変更しようとすると、「ソースファイルは元の sidadm ユーザーによって所有されていません (uid = xxxx)」というエラーが表示される
- 2607227 — hdblcm： 登録_名前変更_システム：SAP HANA インスタンスの名前変更に失敗した
- 2630562 — HANA ホスト名の名前変更に失敗し、HANA が起動しない
- 2935639 — sr_register が global.ini セクションの system_replication_hostname_resolution で指定されているホスト名を使用していない

- 2710211 — エラー：ソースシステムとターゲットシステムの論理ホスト名が重複している
- 2693441 — エラーにより SAP HANA システムの名前を変更できなかった
- 2519672 — HANA プライマリとセカンダリのシステム PKI、SSFS データ、キーが異なるか、確認できない
- 2457129 — 動的階層化がランドスケープの一部である場合、SAP HANA システムホストの名前変更が許可されない
- 2473002 — HANA システムレプリケーションによるスケールアウトシステムの移行 (スケールアウト SAP HANA システムでのこのホスト名変更アプローチの使用には、SAP による制限はありません。 ただし、この手順は個々のホストで繰り返す必要があります。 このアプローチには、他のスケールアウト移行の制限も適用されます。)

製品バージョン

- このソリューションは SAP HANA DB プラットフォームエディション 1.0 と 2.0 に適用されます。

アーキテクチャ

ソースセットアップ

SAP HANA データベースがソース環境にインストールされます。すべての SAP アプリケーションサーバー接続と DB インターフェースは、クライアント接続に同じホスト名を使用します。次の図は、`hdbhost`ソースホスト名とそれに対応する IP アドレスの例を示しています。

ターゲットセットアップ

AWS クラウド ターゲット環境は、同じホスト名を使用して SAP HANA データベースを実行します。AWS のターゲット環境には以下が含まれます。

- SAP HANA データベース
- SAP アプリケーションサーバー
- EBS ボリューム

中間設定

次の図では、 AWS ソースとターゲットのホスト名が一意temp-hostになるように、ターゲット環境のホスト名を一時的に に変更しています。移行がターゲット環境上でテイクオーバーマイルストーンを完了すると、ターゲットシステムの仮想ホスト名は元の名前hdbhostを使用してリネームされます。

中間設定には、以下のオプションのいずれかが含まれます。

- AWS Client VPN クライアント VPN エンドポイントを使用する
- AWS Direct Connect ルーターへの接続

AWS ターゲット環境の SAP アプリケーションサーバーは、レプリケーションのセットアップ前またはテイクオーバー後にインストールできます。ただし、レプリケーションをセットアップする前にアプリケーションサーバーをインストールしておくと、インストール中のダウンタイムの短縮、高可用性の設定、およびバックアップに役立ちます。

ツール

AWS のサービス

- AWS Client VPN は、オンプレミスネットワークの AWS リソースとリソースに安全にアクセスできるマネージドクライアントベースの VPN サービスです。
- AWS Direct Connect は、標準イーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、ネットワークパス内のインターネットサービスプロバイダーをバイパスして AWS のサービス、パブリックに直接仮想インターフェイスを作成できます。
- Amazon Elastic Block Store (Amazon EBS) は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルのストレージボリュームを提供します。EBS ボリュームの動作は、未初期化のブロックデバイスに似ています。これらのボリュームは、デバイスとしてインスタンスにマウントできます。

その他のツール

- SAP アプリケーションサーバー — SAP アプリケーションサーバーは、プログラマーがビジネスロジックを表現する方法を提供します。SAP アプリケーションサーバーは、ビジネスロジックに

基づいてデータ処理を実行します。実際のデータは、別のコンポーネントであるデータベースに格納されます。

- 「SAP HANA コックピット」と「SAP HANA スタジオ」 — SAP HANA コックピットと SAP HANA Studio はどちらも SAP HANA データベースへの管理インターフェースを提供します。SAP HANA Studio では、SAP HANA 管理コンソールは SAP HANA データベース管理に関連するコンテンツを提供するシステムビューです。
- SAP HANA システムレプリケーション — SAP HANA システムレプリケーション (SAP HSR) は、SAP HANA データベースをレプリケートするために SAP が提供している標準手順です。SAP HSR に必要な実行ファイルは、SAP HANA サーバーカーネル自体に含まれています。

エピック

ソース環境とターゲット環境を準備する

タスク	説明	必要なスキル
SAP HANA データベースをインストールして構成します。	ソース環境とターゲット環境で、SAP HANA DB がベストプラクティスの SAP HANA に従ってインストールおよび設定されていることを確認します。詳細については、「SAP HANA on AWS」を参照してください。	SAP 基本管理
IP アドレスをマップします。	ターゲット環境では、一時的なホスト名が内部 IP アドレスに割り当てられていることを確認します。 1. EC2、インスタンス、アクション、ネットワーク、IP アドレスの管理、新しい IP アドレスの割り当てに移動して、AWS マネジメントコンソールの EC2 インスタ	AWS 管理

タスク	説明	必要なスキル
	ンスにセカンダリ IPv4 アドレスを割り当てます。 2. EC2 ネットワークアダプター (NIC) に同じアドレスを割り当てるには、オペレーティングシステムからルートユーザーとして`ip addr add <IP>/32 dev eth0`コマンドを実行し、`<IP>`をステップ 1 の IP アドレスに置き換えます。	
ターゲットホスト名を解決します。	セカンダリ SAP HANA DB で、`/etc/hosts` ファイル内の関連するホスト名を更新して、SAP HANA レプリケーションネットワークのホスト名 (`hdbhost`と`temp-host`) が両方解決されていることを確認します。	Linux アドミニストレーション
ソースとターゲットの SAP HANA データベースをバックアップします。	SAP HANA Studio または SAP HANA コックピットを使用して SAP HANA データベースのバックアップを実行します。	SAP 基本管理

タスク	説明	必要なスキル
エクスチェンジシステム PKI 証明書。	(SAP HANA 2.0 以降にのみ適用) プライマリデータベースとセカンダリデータベースの間のファイルシステム (SSFS) ストア内のシステム公開鍵基盤 (PKI) セキュアストア内の証明書を交換します。詳細については、SAP Note 2369981 — SAP HANA システムレプリケーションによる認証に必要な設定手順を参照してください。	SAP 基本管理

ターゲット SAP HANA データベースの名前を変更します。

タスク	説明	必要なスキル
ターゲットクライアント接続を停止します。	ターゲット環境で SAP アプリケーションサーバーとその他のクライアント接続をシャットダウンします。	SAP 基本管理

ターゲット SAP HANA DB の名前を一時的なホスト名に変更します。	1. ルートユーザーとして、常駐hdblcmを使用してターゲット SAP HANA DB のホスト名を一時的なホスト名に変更します。
	`root $> cd /hana/shared/<SID/hdblcm`
	`root $> ./hdblcm`
	2. `9 | rename_system | Rename the SAP HANA`
		SAP 基本管理

タスク	説明	必要なスキル
	Database System を選択します。 3. 新しい名前 temp-host を指定します。 4. 必要に応じて、他のオプションを検証できます。ただし、ホスト名の変更と SID の変更を混同しないように注意してください (SAP Note 2598814 — hdblcm： SID の名前変更が失敗する)。 SAP HANA DBの停止と開始はhdblcmによって制御されます。	
レプリケーションネットワークを割り当てる。	ソースシステムのglobal.ini ファイルの[system_replication_hostname_resolution] ヘッダーの下に、ソースとターゲットのレプリケーションネットワークの詳細を指定します。次に、エントリをターゲットシステム上のglobal.ini ファイルにコピーします。	SAP 基本管理

タスク	説明	必要なスキル
プライマリでのレプリケーションを有効にします。	次のコマンドを実行して、ソース SAP HANA DB でレプリケーションを有効にします。 `hdbnsutil -sr_enable --name=siteA`	SAP 基本管理

ターゲット SAP HANA データベースをセカンダリシステムとして登録します。	ターゲット SAP HANA DB を SAP HSR のソースとなるセカンダリシステムとして登録するには、「非同期」レプリケーションを選択します。
	
	`(sid)adm $> HDB stop`
	`(sid)adm $> hdbnsutil -sr_register –name=siteB –remotehost=hdbhost /`
	`--remoteInstance=00 –replicationMode=async –operationMode=logreplay`
	`(sid)adm $> HDB start`
	
	または、登録する–onlineオプションを選択できます。その場合は、SAP HANA DB を停止して起動する必要はありません。
		SAP 基本管理

タスク	説明	必要なスキル

同期を検証します。	ソース SAP HANA DB で、すべてのログがターゲットシステムに適用されていることを確認します (非同期レプリケーションであるため)。
	
	レプリケーションを確認するには、ソース上で以下のコマンドを実行します。
	
	`(sid)adm $> cdpy`
	`(sidadm $> python systemReplicationStatus.py`
		SAP 基本管理

ソース SAP アプリケーションと SAP HANA データベースをシャットダウンします。	移行カットオーバー中に、ソースシステム (SAP アプリケーションと SAP HANA データベース) をシャットダウンします。	SAP 基本管理
ターゲットでテイクオーバーを実行します。	AWS 上のターゲットでテイクオーバーを実行するには、コマンド`hdbnsutil -sr_takeover` を実行します。	SAP 基本管理

タスク	説明	必要なスキル
ターゲット SAP HANA DB で、レプリケーションをオフにします。	レプリケーションメタデータをクリアするには、コマンドhdbnsutil -sr_disable を実行してターゲットシステム上のレプリケーションを停止します。 **Note** これは、SAP Note 2693441「」 – エラーにより SAP HANA システムの名前を変更できませんでした。	SAP 基本管理
ターゲット SAP HANA データベースをバックアップします。	テイクオーバーが成功したら、SAP HANA DB の完全バックアップを実行することをお勧めします。	SAP 基本管理

ターゲットシステムの元のホスト名に戻す。

タスク	説明	必要なスキル

ターゲットの SAP HANA DB ホスト名を元のホスト名に戻します。	1. 対象のSAP HANA DBホスト名を元の仮想ホスト名に戻すには、常駐hdblcmを使用します。
	
	`root $> cd /hana/shared/<SID>/hdblcm`
	`root $> ./hdblcm`
	
	2. 9 | rename_system | Rename the SAP HANA
		SAP 基本管理

タスク	説明	必要なスキル
	Database System を選択します。 3. 新しい名前hdbhostを指定します。 必要に応じて、他のオプションを検証できます。ただし、ホスト名の変更と SID の変更を混同しないように注意してください (SAP Note 2598814 — hdblcm： SID の名前変更が失敗する)。	
HD ユーザーストアを調整します。	hdbuserstore の詳細をソースschema/user の詳細に合わせて調整します。詳細な手順については、SAP のドキュメントを参照してください。 このステップを検証するには、コマンドR3trans -dを実行します。結果には SAP HANA データベースへの接続が成功したことが反映されているはずです。	SAP 基本管理
クライアント接続を開始します。	ターゲット環境で SAP アプリケーションサーバーとその他のクライアント接続を起動します。	SAP 基本管理

関連リソース

SAP リファレンス

SAP ドキュメンテーションリファレンスは SAP によって頻繁に更新されます。最新情報を入手するには、SAP Note 2407186 — SAP HANA の高可用性に関するハウツーガイドとホワイトペーパーを参照してください。

追加のSAPメモ

- 2550327 — SAP HANA システムの名前を変更する方法
- 1999880 — よくある質問：SAP HANA システムレプリケーション
- 2078425 — SAP HANA プラットフォームライフサイクル管理ツール (hdblcm) のトラブルシューティングノート
- 2592227 — HANA システムの FQDN サフィックスの変更
- 2048681 — SSH または ルート認証情報なしで、複数のホストシステムで SAP HANA プラットフォームのライフサイクル管理管理タスクを実行する

SAP ドキュメント

- システムレプリケーションネットワーク接続
- システムレプリケーションのホスト名解決

AWS リファレンス

- SAP HANA を他のプラットフォームから に移行する AWS

追加情報

ホスト名のリネーム活動の一部としてhdblcmによって実行された変更は、以下の詳細ログに集約されます。

分散可用性グループを使用して SQL Server を AWS に移行する

プラビーン・マーサラ (AWS) によって作成されました

概要

Microsoft SQL Server Always On 可用性グループは、SQL Server に高可用性 (HA) およびディザスタリカバリ (DR) ソリューションを提供します。可用性グループは、読み取り/書き込みトラフィックを受け入れるプライマリレプリカと、読み取りトラフィックを受け入れる最大 8 つのセカンダリレプリカで構成されます。可用性グループは 2 つ以上のノードで構成される Windows Server フェールオーバークラスター (WSFC) 上に構成されます。

Microsoft SQL Server Always On 分散可用性グループは、2 つの独立した WFSC 間で 2 つの別々の可用性グループを構成するソリューションを提供します。分散可用性グループに含まれる可用性グループは、同じデータセンター内にある必要はありません。1 つの可用性グループをオンプレミスに配置し、もう 1 つの可用性グループを別のドメインの Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の Amazon Web Services (AWS) クラウドに配置できます。

このパターンは、分散可用性グループを使用して、既存の可用性グループの一部であるオンプレミスの SQL Server データベースを、Amazon EC2 に可用性グループが設定された SQL Server に移行する手順の概要を示しています。このパターンに従うことで、カットオーバー中のダウンタイムを最小限に抑えながら、データベースを AWS クラウドに移行できます。データベースは、カットオーバー直後から AWS で高い可用性を発揮します。このパターンを使用して、同じバージョンの SQL Server を維持したまま、基盤となるオペレーティングシステムをオンプレミスから AWS に変更することもできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Direct Connect または AWS Site-to-Site VPN
- オンプレミスと AWS の 2 つのノードに同じバージョンの SQL Server がインストールされている

製品バージョン

- SQL Server バージョン 2016 以降)
- SQL Server Enterprise Edition

アーキテクチャ

ソーステクノロジースタック

- オンプレミスで Always On 可用性グループを使用する Microsoft SQL Server データベース

ターゲットテクノロジースタック

- AWS クラウド上の Amazon EC2 にある Always On アベイラビリティグループを備えた Microsoft SQL Server データベース

移行アーキテクチャ

用語

- WSFC 1 — オンプレミスの WSFC
- WSFC 2 — AWS クラウド上の WSFC
- AG 1 — WSFC 1 にある最初のアベイラビリティグループ
- AG 2 — WSFC 2 にある 2 番目のアベイラビリティグループ
- SQL Server プライマリレプリカ — AG 1 のノードで、すべての書き込みのグローバルプライマリと見なされます。
- SQL Server フォワーダー — SQL Server プライマリレプリカからデータを非同期で受信する AG 2 のノード
- SQL Server セカンダリレプリカ — プライマリレプリカまたはフォワーダーからデータを同期的に受信する AG 1 または AG 2 のノード

ツール

- AWS Direct Connect, –標準イーサネット光ファイバケーブルを介して、内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、パブリック AWS サービスへ仮想インターフェイスを直接作成できるため、ネットワークパスのインターネットサービスプロバイダーを回避できます。

- 「Amazon EC2」— Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動し、スケールアウトまたはスケールインできます。
- AWS Site-to-Site VPN — AWS Site-to-Site VPN は、Site-to-Site 仮想プライベートネットワークの作成をサポートします。AWS で起動するインスタンスと独自のリモートネットワーク間でトラフィックを渡すように VPN を設定できます。
- マイクロソフトSQLサーバー管理スタジオ — マイクロソフトSQLサーバー管理スタジオ(SSMS)は、SQL Server インフラストラクチャを管理するための統合環境です。SQL Server とやり取りする豊富なスクリプトエディターを備えたユーザーインターフェイスとツールグループを備えています。

エピック

AWS に 2 つ目のアベイラビリティグループを設定する

タスク	説明	必要なスキル
AWS に WSFC を作成します。	HA 用に 2 つのノードを備えた Amazon EC2 インスタンスに WSFC 2 を作成します。このフェールオーバークラスターを使用して、AWS に 2 つ目の可用性グループ (AG 2) を作成します。	システム管理者、システム運用管理者
WSFC 2 に 2 つ目の可用性グループを作成します。	SSMS を使用して、WSFC 2 の 2 つのノードに AG 2 を作成します。WSFC 2 の最初のノードがフォワーダーとして機能します。WSFC 2 の 2 番目のノードは AG 2 のセカンダリレプリカとして機能します。 この段階では、AG 2 にはデータベースはありません。これ	DBA、開発者

タスク	説明	必要なスキル
	が分散可用性グループの設定の出発点です。	
AG 2 で復旧オプションなしでデータベースを作成します。	オンプレミス可用性グループ (AG 1) のデータベースをバックアップします。 復旧オプションなしで、AG 2 のフォワーダーとセカンダリレプリカの両方にデータベースを復元します。データベースを復元するときは、データベースデータファイルとログファイルを保存するのに十分なディスク容量がある場所を指定します。 この段階では、データベースは復元中の状態です。これらは AG 2 または分散可用性グループには含まれておらず、同期も行われていません。	DBA、開発者

分散可用性グループの設定

タスク	説明	必要なスキル
AG 1 に分散可用性グループを作成します。	AG 1 に分散可用性グループを作成するには、`DISTRIBUTED` オプションを指定して`CREATE AVAILABILITY GROUP`を使用します。 1. AG 1 と AG 2 に はLISTENER_URL エンド	DBA、開発者

タスク	説明	必要なスキル
	ポイントアドレスを使用してください。 2. AVAILABILITY-MODE では、ネットワークの遅延を避けるためにASYNCHRONOUS_COMMIT を使用します。これはデータベースのパフォーマンスには影響しません。 3. FAILOVER_MODE の場合は、MANUAL を使用します。これは分散可用性グループで機能する唯一の可用性モードです。 4. AG 2 でデータベースを手動で復元し、大規模なデータベースをより細かく制御するには、MANUALまたはSEEDING_MODE を使用します。	

タスク	説明	必要なスキル
AG 2 に分散可用性グループを作成します。	AG 2 に分散可用性グループを作成するには、DISTRIBUTED オプションを指定してALTER AVAILABILITY GROUPを使用します。 1. AG 1 と AG 2 にはLISTENER_URL エンドポイントアドレスを使用してください。 2. AVAILABILITY-MODE では、ネットワークの遅延を避けるためにASYNCHRONOUS_COMMIT を使用します。これはデータベースのパフォーマンスには影響しません。 3. FAILOVER_MODE の場合は、MANUAL を使用します。これは分散可用性グループで機能する唯一の可用性モードです。 4. AG 2 でデータベースを手動で復元し、大規模なデータベースをより細かく制御するには、MANUALまたはSEEDING_MODE を使用します。 分散可用性グループは AG 1 と AG 2 の間に作成されます。	DBA、開発者

タスク	説明	必要なスキル
	AG 2 のデータベースは、AG 1 から AG 2 へのデータフローに参加するようにまだ設定されていません。	
AG 2 のフォワーダーとセカンダリレプリカにデータベースを追加します。	AG 2 のフォワーダーとセカンダリレプリカの両方でSET HADRAVAILABILITY GROUPオプションを指定してALTER DATABASEを使用して、データベースを分散可用性グループに追加します。 これにより、AG 1 と AG 2 のデータベース間の非同期データフローが開始されます。 グローバルプライマリは書き込みを行い、AG 1 のセカンダリレプリカにデータを同期的に送信し、AG 2 のフォワーダーにデータを非同期で送信します。AG 2 のフォワーダーは、AG 2 のセカンダリレプリカに同期的にデータを送信します。	DBA、開発者

AG 1 と AG 2 間の非同期データフローを監視します。

タスク	説明	必要なスキル
DMV と SQL サーバーログを使用します。	動的管理ビュー (DMV) と SQL Server ログを使用して、2 つの可用性グループ間	DBA、開発者

タスク	説明	必要なスキル
	のデータフローの状態を監視します。 監視の対象となる DMV には、`sys.dm_hadr_availability_replica_states` と`sys.dm_hadr_automatic_seeding` があります。 フォワーダー同期の状態については、フォワーダーの SQL Server ログで「同期状態」を監視します。	

最終移行のためのカットオーバーアクティビティを実行

タスク	説明	必要なスキル
プライマリレプリカへのすべてのトラフィックを停止します。	AG 1 のプライマリレプリカへの受信トラフィックを停止して、データベースへの書き込みアクティビティが発生せず、データベースを移行できる状態にします。	アプリ所有者、開発者
AG 1 の分散可用性グループの可用性モードを変更します。	プライマリレプリカで、分散可用性グループのアベイラビリティモードを同期に設定します。 アベイラビリティモードを同期に変更すると、データは AG 1 のプライマリレプリカか	DBA、開発者

タスク	説明	必要なスキル
	ら AG 2 のフォワーダーに同期的に送信されます。	
両方の可用性グループの LSN を確認します。	AG 1 と AG 2 の両方の最新のログシーケンス番号 (LSN) を確認します。AG 1 のプライマリレプリカでは書き込みが行われていないため、データは同期されており、両方の可用性グループの最後の LSN は一致しているはずです。	DBA、開発者
AG 1 をセカンダリロールに更新します。	AG 1 をセカンダリロールに更新すると、AG 1 はプライマリレプリカのロールを失い、書き込みを受け付けなくなり、2 つの可用性グループ間のデータフローが停止します。	DBA、開発者

2 番目の可用性グループへのフェイルオーバー

タスク	説明	必要なスキル
AG 2 に手動でフェールオーバーします。	AG 2 のフォワーダーで、データが失われないように分散可用性グループを変更します。AG 1 と AG 2 の最後の LSN が一致することはすでに確認済みなので、データ損失は問題になりません。 AG 2 のフォワーダでのデータ損失を許可すると、AG 1 と AG 2 の役割が変わります。	DBA、開発者

タスク	説明	必要なスキル
	• AG 2 はプライマリレプリカとセカンダリレプリカを含むアベイラビリティグループになります。 • AG 1 はフォワーダーとセカンダリレプリカを含むアベイラビリティグループになります。	
AG 2 の分散可用性グループの可用性モードを変更します。	AG 2 のプライマリレプリカで、アベイラビリティモードを非同期に変更します。 これにより、AG 2 から AG 1 へのデータ移動が、同期から非同期に変更されます。このステップは、AG 2 と AG 1 の間でネットワーク遅延が発生してもそれを回避するために必要であり、データベースのパフォーマンスには影響しません。	DBA、開発者

タスク	説明	必要なスキル
新しいプライマリレプリカへのトラフィックの送信を開始します。	AG 2 のリスナー URL エンドポイントを使用してデータベースにトラフィックを送信するように接続文字列を更新します。 AG 2 は、AG 2 の独自のセカンダリレプリカにデータを送信するとともに、書き込みを受け付けて AG 1 のフォワーダーにデータを送信するようになりました。データは AG 2 から AG 1 に非同期的に移動します。	アプリ所有者、開発者

カットオーバー後のアクティビティの実行

タスク	説明	必要なスキル
分散可用性グループを AG 2 にドロップします。	予定された時間だけ移行を監視します。次に、AG 2 に分散可用性グループをドロップして、AG 2 と AG 1 の間の分散可用性グループの設定を削除します。これにより、分散可用性グループの設定が削除され、AG 2 から AG 1 へのデータフローが停止します。 この時点で、AG 2 は AWS での可用性が高く、プライマリレプリカは書き込みを行い、セカンダリレプリカは同じ可用性グループ内にあります。	DBA、開発者

タスク	説明	必要なスキル
オンプレミスサーバーを廃止します。	AG 1 の一部である WSFC 1 のオンプレミスサーバーの使用を停止します。	システム管理者、システム運用管理者

関連リソース

- 分散可用性グループ
- SQL Docs： 分散可用性グループ
- SQL Docs： Always On 可用性グループ：高可用性と災害復旧のためのソリューション

SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for Oracle に移行

作成者: Ramu Jagini (AWS)

概要

このパターンでは、オンプレミスの Oracle 8i または 9i データベースを、 Oracle データベースの Amazon Relational Database Service (Amazon RDS) に移行する方法について説明してます。このパターンを使用して、Quest SharePlex を同期レプリケーションに使用すると、ダウンタイムを短縮して移行を完了できます。

AWS Database Migration Service (AWS DMS) はソース環境として Oracle 8i または 9i が適用されないため、移行には中間 Oracle データベースインスタンスを使用する必要があります。「SharePlex 7.6.3」 を使用して、以前の Oracle データベースバージョンから新しい Oracle データベースバージョンに複製できます。中間 Oracle データベースインスタンスは SharePlex 7.6.3 のターゲットとして互換性があり、AWS DMS またはそれ以降の SharePlex リリースのソースとしても適用されます。この適用により、Amazon RDS for Oracle ターゲット環境へのデータの以降のレプリケーションが可能になります。

いくつかの廃止されたデータ型や特徴量が、Oracle 8i または 9i から最新バージョンの Oracle Database への移行に影響する可能性があることを考慮します。この影響を軽減するために、このパターンでは Oracle 11.2.0.4 を中間データベースバージョンとして使用し、Amazon RDS for Oracle ターゲット環境への移行前にスキーマコードを最適化します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミス環境のソース Oracle 8i または 9i データベース
- Amazon Elastic Compute Cloud (Amazon EC2) でのステージングの「Oracle Database 12c Release 2」 (12CR2)
- Quest SharePlex 7.6.3 (商用グレード)

制約事項

- RDS for Oracle の制限事項

製品バージョン

- ソースデータベース用の Oracle 8i または 9i
- ステージングデータベースの Oracle 12CR2 (Amazon RDS for Oracle バージョンと一致する必要があります)
- ターゲットデータベースの Oracle 12CR2 以降 (Amazon RDS for Oracle)

アーキテクチャ

ソーステクノロジースタック

- Oracle 8i または 9i データベース
- SharePlex

ターゲットテクノロジースタック

- 「Amazon RDS for Oracle」

移行アーキテクチャ

次の図表では、オンプレミス環境から AWS クラウドの Oracle 8i または 9i データベースをオンプレミス環境から AWS クラウドの Amazon RDS for Oracle DB インスタンスに移行する方法を示します。

この図表は、次のワークフローを示しています：

1. Oracle ソースデータベースをアーカイブログモード、強制ロギング、および補足ロギングでイネーブルにします。
2. リカバリマネージャー (RMAN) のポイントインタイムリカバリと 「FLASHBACK_SCN」 を使用して Oracle ステージングデータベースを Oracle ソースデータベースから復元します。
3. FLASHBACK_SCN (RMAN で使用されている) を使用して、 Oracle ソースデータベースから REDO ログを読み取るように SharePlex を設定します。
4. SharePlex レプリケーションを開始して、Oracle ソースデータベースから Oracle ステージングデータベースにデータを同期します。

5. EXPDP と FLASHBACK_SCN を伴うIMPDP を使用して Amazon RDS for Oracle のターゲットデータベースを復元します。
6. FLASHBACK_SCN (EXPDP で使われている) を使用して、AWS DMS とそのソースタスクを Oracle ステージングデータベースとして設定し、Amazon RDS for Oracle をターゲットデータベースとして設定します。
7. AWS DMS タスクを開始して、Oracle ステージングデータベースから Oracle ターゲットデータベースにデータを同期します。

ツール

- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「Quest SharePlex」 は、ダウンタイムを最小限に抑え、データ損失なしでデータを移動できる Oracle 間のデータ複製ツールです。
- 「リカバリマネージャー (RMAN)」 は、データベースのバックアップとリカバリのタスクを実行する Oracle データベースクライアントです。データベースファイルのバックアップ、リストア、リカバリを大幅に簡素化します。
- 「データパンプエクスポート」 を使用して、データとメタデータをダンプファイルセットと呼ばれるオペレーティングシステムファイルのセットにアップロードできます。ダンプファイルセットは、「データパンプインポート」 ユーティリティまたは 「DBMS_DATAPUMP」 パッケージでのみインポートできます。

エピック

Amazon EC2 に SharePlex と Oracle ステージングデータベースをセットアップします

タスク	説明	必要なスキル
EC2 インスタンスを作成します。	1. EC2 インスタンスを作成します。 2. Oracle 12CR2 を EC2 インスタンスにインストールして Oracle ステージング	Oracle 管理

タスク	説明	必要なスキル
	データベースとして使用します。	
ステージングデータベースを準備します。	Oracle 8i または 9i データベースソース環境から RMAN バックアップを取得して、Oracle 12CR2 のアップグレードとして復元できるように Oracle ステージングデータベースを準備します。 詳細については、Oracle ドキュメントの「Oracle 9i リカバリマネージャーユーザーズガイド」と「データベースバックアップ およびリカバリーユーザーズガイド」を参照してください。	Oracle 管理
SharePlex を設定します。	SharePlex ソースをオンプレミスの Oracle 8i または 9i データベースとして設定し、ターゲットを Amazon EC2 でホストされている Oracle 12CR2 ステージングデータベースとして設定します。	SharePlex、Oracle管理

ターゲット環境として Amazon RDS for Oracleを設定します。

タスク	説明	必要なスキル
Oracle DB インスタンスを作成します。	Amazon RDS for Oracle データベースを作成し、Oracle 12CR2 をデータベースに接続します。	DBA

タスク	説明	必要なスキル
	詳細については、Amazon RDS ドキュメントの「Oracle DB インスタンスを作成して Oracle DB インスタンスのデータベースに接続」を参照してください。	
Amazon RDS for Oracle をステージングデータベースから復元します。	1. FLASHBACK_SCN を使用して、 Oracle ステージングデータベースサーバーから EXPDP バックアップを作成します。 2. Amazon RDS for Oracle をステージングデータベースから復元します。 詳細については、Oracle ドキュメントの「54 DBMS_DATAPUMP」を参照してください。	DBA

AWS DMS のセットアップ

タスク	説明	必要なスキル
データベースのエンドポイントを作成します。	Oracle ステージングデータベース用のソースエンドポイントと、 Amazon RDS for Oracle データベースのターゲットエンドポイントを作成します。 詳細については、 AWS ナレッジセンターの「AWS	DBA

タスク	説明	必要なスキル
	DMS を使用してソースエンドポイントまたはターゲットエンドポイントを作成する方法」 を参照してください。	
レプリケーションインスタンスを作成します。	AWS DMS を使用して、Oracle ステージングデータベースの Amazon RDS for Oracle データベースへのレプリケーションインスタンスを起動します。 詳細については、AWS ナレッジセンターの「AWS DMS レプリケーションインスタンスを作成する方法」 を参照してください。	DBA
レプリケーションタスクを作成します。	EXPDP から FLASHBACK_SCN を使用して、変更データキャプチャ (CDC) の AWS DMS レプリケーションタスクを作成します (全ロードは EXPDP ですでに行われているため)。 詳細については、AWS DMS のドキュメントの「タスクを作成 」を参照してください。	DBA

Amazon RDS for Oracle へのカットオーバー

タスク	説明	必要なスキル
アプリケーションワークロードを停止します。	予定されているカットオーバー期間中、アプリケーションサーバーとそのアプリケーションを停止します。	アプリ開発者、DBA
オンプレミスの Oracle ステージングデータベースと EC2 インスタンスの同期を検証します。	オンプレミスのソースデータベースでいくつかのログスイッチを実行して、SharePlex レプリケーションインスタンスから Amazon EC2 の Oracle ステージングデータベースへのレプリケーションタスクのすべてのメッセージが掲示されていることを確認します。 詳細については、Oracle ドキュメントの「6.4.2 ログファイルの切り替え」を参照してください。	DBA
Oracle ステージングデータベースと Amazon RDS for Oracle データベースとの同期を検証します。	すべての AWS DMS タスクに遅延やエラーがないことを確認し、タスクの検証状態を確認します。	DBA
SharePlex と Amazon RDS のレプリケーションを停止します。	SharePlex と AWS DMS の両方のレプリケーションにエラーが表示されない場合、両方のレプリケーションを停止します。	DBA
アプリケーションを Amazon RDS に再マップします。	Amazon RDS for Oracle エンドポイントの詳細をアプリケーションサーバーとそのア	アプリ開発者、DBA

タスク	説明	必要なスキル
	プリケーションと共有し、それからアプリケーションを起動して業務を再開します。	

AWS ターゲット環境のテスト

タスク	説明	必要なスキル
AWS で Oracle ステージングデータベース環境をテストします。	1. SharePlex レプリケーションをテストし、Oracle ステージングデータベースに同期ギャップやレプリケーションエラーがないことを確認します。 2. オンプレミス環境で定義されたベンチマークを通じて、アプリケーションが期待どおりに動作することを確認します。	SharePlex、Oracle管理
Amazon RDS 環境をテストします。	1. レプリケーション後に、Amazon RDS に伝達されるすべてのデータに エラーがないことを確認します。 2. 別のアプリケーションを Amazon RDS DB インスタンスにポイントし、パフォーマンステストを実行して予想される動作を検証します。 詳細については、Amazon RDS ドキュメントの	Oracle 管理

タスク	説明	必要なスキル
	「Amazon RDS for Oracle」を参照してください。	

関連リソース

- 「自信を持って移行」
- 「Amazon EC2」
- 「Amazon RDS for Oracle」
- 「AWS Database Migration Service」
- 「AWS DMS 移行のデバッグ: 問題が発生した場合の対処方法 (パート 1)」
- 「AWS DMS 移行のデバッグ: 問題が発生した場合の対処方法 (パート 2)」
- 「AWS DMS 移行のデバッグ: 問題が発生した場合の対処法」「(パート 3)」
- 「SharePlex によるデータベースレプリケーション」
- 「SharePlex: あらゆる環境に対応するデータベースレプリケーション」

暗号化されていない Amazon Aurora インスタンスをモニタリングする

作成者: Mansi Suratwala (AWS)

概要

このパターンは、Amazon Web Services (AWS) CloudFormation テンプレートを提供して、暗号化をオンにせずに Amazon Aurora インスタンスが作成されたときに自動通知を設定するためにデプロイできます。

Aurora はフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。Aurora では、既存のアプリケーションのほとんどを変更せずに、ほんの少しのワークロードで MySQL のスループットの 5 倍、PostgreSQL のスループットの 3 倍を実現します。

CloudFormation テンプレートは Amazon CloudWatch Events イベントと AWS Lambda 関数を作成します。このイベントでは、AWS CloudTrail を使用して 任意のAurora インスタンスの作成や既存インスタンスの特定時点での復元をモニタリングします。Cloudwatch Events は、暗号化が有効になっているかどうかをチェックする Lambda 関数を呼び出します。暗号化が有効になっていない場合、Lambda 関数から Amazon Simple Notification Service (Amazon SNS) 通知が送信されます。

前提条件と制限

前提条件

- アクティブなAWS アカウント

機能制限

- このサービスコントロールは Amazon Aurora インスタンスでのみ機能します。他のAmazon Relational Database Service (Amazon RDS) インスタンスをサポートしません。
- CloudFormation テンプレートは、`CreateDBInstance` と `RestoreDBClusterToPointInTime` にのみデプロイする必要があります。

製品バージョン

- Amazon Aurora でサポートされている PostgreSQL のバージョン
- Amazon Aurora でサポートされている MySQL バージョン

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Aurora
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

CloudFormation テンプレートは、さまざまな国/地域とアカウントに複数回使用できます。各リージョンまたはアカウントで 1 回のみ実行できます。

ツール

ツール

- 「Amazon Aurora」— Amazon Aurora はフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- 「AWS CloudTrail」— AWS CloudTrail は、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査を管理します。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- 「Amazon CloudWatch Events」— Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- 「Amazon S3」—「Amazon Simple Storage Service (Amazon S3)」は、スケーラビリティの高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップとデータレイクなど、幅広いストレージソリューションに使用できます。

- 「Amazon SNS」—「Amazon Simple Notification Service (Amazon SNS)」は、Lambda、HTTP、Eメール、モバイルプッシュ通知とモバイルテキストメッセージ (SMS) を使用してメッセージを配信するマネージドサービスです。

コード

プロジェクトの .zip ファイルは添付ファイルとして入手できます。

エピック

Lambda スクリプト用の S3 バケットを作成する

タスク	説明	必要なスキル
S3 バケットを定義します。	Amazon S3 コンソールを開き、S3 バケットを選択、または作成します。この S3 バケットは Lambda コードの.zip ファイルをホストします。S3 バケットは、Aurora と同じ国/地域に存在する必要があります。S3 バケット名の先頭にスラッシュを含めることはできません。	クラウドアーキテクト

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
Lambda コードをアップロードします。	「添付ファイル」セクションにある Lambda コードの.zip ファイルを S3 バケットにアップロードします。	クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	CloudFormation コンソールで、このパターンへの添付ファイルとして提供されている `RDS_Aurora_Encryption_At_Rest.yml` CloudFormation テンプレートをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を指定します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。	クラウドアーキテクト
S3 キーを指定します。	S3 バケットの Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例: `<directory>/<file-name>.zip`)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況	クラウドアーキテクト

タスク	説明	必要なスキル
	に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。 `Warning` 潜在的に有害な状況を示します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定されたメールアドレスに購読メールメッセージが送信されます。通知を受信するには、このメールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「S3 バケットの作成」
- 「S3 バケットへのファイルアップロード」
- Amazon Aurora DB クラスターの作成
- 「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon CloudWatch を使用して Oracle GoldenGate ログを監視する

作成者：Chithra Krishnamurthy (AWS)

概要

Oracle GoldenGate は、Oracle データベース用の Amazon Relational Database Service (Amazon RDS) または Amazon Elastic Compute Cloud (Amazon EC2) でホストされている Oracle データベース間のリアルタイムレプリケーションを提供しています。一方向レプリケーションと双方向レプリケーションの両方をサポートしています。

Oracle GoldenGate をレプリケーションに使用する場合は、Oracle GoldenGate プロセスが稼働していて、ソースデータベースとターゲットデータベースが同期していることを確認してください。

このパターンでは、GoldenGate エラーログの Amazon CloudWatch モニタリングを実装する手順と、レプリケーションを迅速に再開するための適切なアクションを実行できるような、STOP または ABEND など特定のイベントの通知を送信するようにアラームを設定する方法について説明します。

前提条件と制限

前提条件

- GoldenGate は EC2 インスタンスにインストールし設定されているため、これらの EC2 インスタンスで CloudWatch モニタリングをセットアップできます。AWS リージョン全体で GoldenGate をモニタリングして双方向レプリケーションを行う場合は、GoldenGate プロセスが実行されている各 EC2 インスタンスに CloudWatch エージェントをインストールする必要があります。

制限事項

- このパターンでは、CloudWatch を使用して GoldenGate プロセスをモニタリングする方法を説明します。CloudWatch は、レプリケーション中のレプリケーションラグやデータ同期の問題をモニタリングしません。「GoldenGate のドキュメント」で説明されているように、レプリケーションラグやデータ関連のエラーをモニタリングするには、別の SQL クエリを実行する必要があります。

製品バージョン

- このドキュメントは、Linux x86-64 上の Oracle 用 Oracle GoldenGate 19.1.0.0.4 の実装を基にしています。ただし、このソリューションは GoldenGate のすべての主要バージョンに適用できます。

アーキテクチャ

ターゲットテクノロジースタック

- EC2 インスタンスにインストールされた Oracle 用 GoldenGate バイナリ
- Amazon CloudWatch
- Amazon Simple Notification Service (Amazon SNS)

ターゲットアーキテクチャ

ツール

AWS サービス

- Amazon CloudWatch は、このようなパターンで GoldenGate エラーログをモニタリングするために使用されるモニタリングサービスです。
- Amazon SNS は、このようなパターンでメール通知を送信するために使用されるメッセージ通知サービスです。

その他のツール

- 「Oracle GoldenGate」は、Oracle データベース用の Amazon RDS for Oracle または Amazon EC2 でホストされている Oracle データベースに使用できるデータレプリケーションツールです。

ハイレベルな実装ステップ

1. CloudWatch エージェント用に AWS Identity and Access Management (IAM) ロールを作成します。
2. GoldenGate エラーログが生成される EC2 インスタンスに IAM ロールを添付します。
3. EC2 インスタンスに CloudWatch エージェントをインストールします。
4. CloudWatch エージェント設定ファイルの `awscli.conf` と `awslogs.conf` を設定します。

5. CloudWatch エージェントを起動します。
6. ロググループにメトリックスフィルタを作成します。
7. Amazon SNS をセットアップします。
8. 次に、メトリクスフィルターのアラームを作成します。Amazon SNS は、これらのフィルタがイベントをキャプチャするとメールアラートを送信します。

詳細な手順については、次のセクションを参照してください。

エピック

ステップ 1. CloudWatch エージェント用の IAM ロールを作成

タスク	説明	必要なスキル
IAM ロールを作成します。	AWS リソースへのアクセスには権限が必要なので、各サーバーが CloudWatch エージェントを実行するのに必要な権限を含む IAM ロールを作成します。 IAM ロールを作成するには 1. AWS マネジメントコンソールにサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。 2. ナビゲーションペインで ロール を選択してから、ロールを作成する を選択します。 3. [信頼できるエンティティタイプ] には、[AWS サービス] を選択します。	AWS 全般

タスク	説明	必要なスキル
	4. [ユースケース] には [EC2] を選択し、[次へ] を選択します。 5. ポリシーのリストで、[CloudWatchAgentServerPolicy] の横にあるチェックボックスを選択します。必要に応じて、検索ボックスを使用してポリシーを見つけます。 6. [Next (次へ)] を選択します。 7. [Role name] (ロール名) に、新しいロールの名前 (例えば `goldengate-cw-monitoring-role` または適宜の別の名前) を入力します。 8. (オプション) [Role description] (ロールの説明) に、説明を入力します。 9. [ポリシー名] の下に [CloudWatchAgentServerPolicy] が表示されていることを確認します。 10. (オプション) 1 つ以上のタグキーと値のペアを追加して、このロールのアクセスを整理、追跡、制御し、[ロールの作成] を選択します。	

ステップ 2. GoldenGate EC2 インスタンスに IAM ロールを添付

タスク	説明	必要なスキル
GoldenGate エラーログが生成される EC2 インスタンスに IAM ロールを添付します。	GoldenGate によって生成されたエラーログは CloudWatch に入力してモニタリングする必要があるため、ステップ 1 で作成した IAM ロールを GoldenGate が実行されている EC2 インスタンスに添付する必要があります。 IAM ロールをインスタンスにアタッチするには 1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。 2. ナビゲーションペインでインスタンスを選択し、GoldenGate が実行されているインスタンスを見つけます。 3. インスタンスを選択し、[アクション]、[セキュリティ]、[IAM ロールの変更] の順に選択します。 4. インスタンスにアタッチする最初のステップで作成した IAM ロールを選択して、[保存] を選択します。	AWS 全般

ステップ 3 ~ 5。Goldengate EC2 インスタンスに CloudWatch Logs エージェントをインストールして設定します。

タスク	説明	必要なスキル
GoldenGate EC2 インスタンスに CloudWatch エージェントをインストールします。	エージェントをインストールするには、以下のコマンドを実行します。 `sudo yum install -y awslogs`	AWS 全般

エージェント設定ファイルを編集します。	1. 以下のコマンドを実行してください。
	`sudo su -`
	2. このファイルを編集して、必要に応じて AWS リージョンを更新します。
	`cat /etc/awslogs/conf [plugins] cwlogs = cwlogs [default] region = us-east-1`
	3. `/etc/awslogs/awslogs.conf` ファイルを編集して、ファイル名、ロググループ名、日付/時刻の形式を更新します。`ggserror.log` で日付の形式と一致する日付/時刻を指定する必要があります。そうしないと、ログストリームは CloudWatch に流れません。以下に例を示します。
		AWS 全般

タスク	説明	必要なスキル

	`datetime_format = %Y-%m-%dT%H:%M:%S%z`
	`file = /u03/oracle/oragg/ggserr.log`
	`log_group_name = goldengate_monitor`	

CloudWatch エージェントを起動します。	エージェントを開始するには、次のコマンドを実行します。 `$ sudo service awslogsd start` エージェントを起動した後、CloudWatch コンソールでロググループを表示できます。ログストリームにはファイルのコンテンツが含まれています。	AWS 全般

ステップ 6. ロググループのメトリクスフィルターの作成

タスク	説明	必要なスキル
キーワードの「ABEND」と「STOPPED」のために、メトリックスフィルターを作成します。	ロググループのメトリックスフィルタを作成すると、エラーログでフィルタが特定されるたびに、アラームが起動し、Amazon SNS 設定に基づき、メール通知が送信されます。 メトリックスフィルターを作成するには	CloudWatch

タスク	説明	必要なスキル
	1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。 2. ロググループの名前を選択します。 3. [アクション]、[メトリクスフィルターの作成] の順に選択します。 4. フィルタパターンには、ABEND などのパターンを指定します。 5. [次へ] を選択し、メトリクスフィルターの名前を入力します。 6. [Metric details (メトリクスの詳細)] の [Metric namespace (メトリクス名前空間)] で、メトリクスを発行する CloudWatch 名前空間の名前を入力します。名前空間がまだ存在しない場合は、[新規作成] が選択されていることを確認します。 7. メトリクスフィルターでフィルター内のキーワードの出現回数をカウントする場合は、[メトリクス値] に「1」と入力します。 8. 単位をなしに設定します。 9. [Create metric filter] (メトリクスフィルターの作成) を	

タスク	説明	必要なスキル
	選択します。ナビゲーションペインから作成したメトリクスフィルターを見つけることができます。 10STOPPED パターン用のメトリクスフィルターをもう 1 つ作成します。1 つのロググループ内には、複数のメトリックスフィルターを作成し、アラームを個別に設定できます。	

ステップ 7. Amazon SNS をセットアップする

タスク	説明	必要なスキル
SNS トピックを作成します。	このステップでは、Amazon SNS を設定して、メトリックスフィルタのアラームを作成します。 SNS トピックを作成するには 1. https://console.aws.amazon.com/sns/home「https://www.com で Amazon SNS コンソールにサインインします。 2. [トピックの作成] で、goldengate-alert などのトピック名を入力し、[次のステップ] を選択します。	Amazon SNS

タスク	説明	必要なスキル
	3. [Type (タイプ)] で、[Standard (標準)] を選択してください。 4. フォームの最下部までスクロールし、[トピックの作成] を選択します。コンソールに新しいトピックの [詳細] ページが表示されます。	

タスク	説明	必要なスキル
サブスクリプションを作成します。	トピックのサブスクリプションを作成するには 1. 左のナビゲーションペインで、[サブスクリプション] を選択します。 2. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。 3. [サブスクリプションの作成] ページで、[トピック ARN] フィールドを選択し、AWS のトピックを一覧表示します。 4. 上記のステップで作成したトピックを選択します。 5. [プロトコル] で [E メール] を選択します。 6. [エンドポイント] に、通知を受信するために使用できる E メールアドレスを入力します。 7. [サブスクリプションの作成] を選択し、コンソールは新しいサブスクリプションの [詳細] ページを開きます。 8. メールボックスに AWS 通知からのメッセージが届いているか確認し、E メール内の [サブスクリプションの確認] を選択します。	Amazon SNS

タスク	説明	必要なスキル
	Amazon SNS がウェブブラウザを開き、サブスクリプション ID とともにサブスクリプションの確認を表示します。	

ステップ 8: メトリックスフィルタの通知を送信するアラームを作成します。

タスク	説明	必要なスキル
SNS トピック用のアラームを作成します。	ロググループのメトリクスフィルターに基づいてアラームを作成するには 1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。 2. ナビゲーションペインで、[Logs] (ログ)、[Log groups] (ロググループ) の順に選択します。 3. メトリクスフィルターを含むロググループを選択します。 4. [Metric filters] (メトリクスフィルター) を選択します。 5. メトリクスフィルタータブで、アラームのベースにするメトリクスフィルターのボックスを選択します。 6. [アラームの作成] を選択します。	CloudWatch

タスク	説明	必要なスキル
	7. 条件については、各セクションで以下のように指定します。 • [Threshold type] で [静的] を選択します。 • [Whenever <metric-name> is . . .] の場合は、[Greater] を選択します。 • それなら。。 、0 を指定します。 8. [Next (次へ)] を選択します。 9. 通知中 • [Alarm state trigger] (アラーム状態トリガー) で、[In alarm] (アラーム状態) を選択します。 • Send notification to (通知の宛先) に、既存の SNS トピックを選択します。 • 電子メールボックスで、前のステップで作成した Amazon SNS トピックを選択します。 10.[Next (次へ)] を選択します。 11.[名前と説明] にアラームの名前と説明を入力します。 **Note** 説明には、通知 E メールがわかりやす	

タスク	説明	必要なスキル
	いようにインスタンス名を指定できます。 12[プレビューと作成] で、設定が正しいことを確認し、[アラームの作成] を選択します。 これらの手順を実行すると、モニタリングしている GoldenGate エラーログファイル (`ggserr.log`) でこれらのパターンが検出されるたびに、電子メール通知が届きます。	

トラブルシューティング

問題	ソリューション
GoldenGate エラーログからのログストリームは CloudWatch に流れません。	`/etc/awslogs/awslogs.conf` ファイルを確認して、ファイル名、ロググループ名、日付/時刻の形式を確認します。`ggserror.log` の日付形式と一致する日付/時刻を指定する必要があります。そうしないと、ログストリームは CloudWatch に流れません。

関連リソース

- 「Amazon CloudWatch ドキュメント」
- 「CloudWatch エージェントを使用したメトリクスとログの収集」
- 「Amazon SNS のドキュメント」

Amazon RDS for Oracle で Oracle Database エンタープライズエディションから標準エディション 2 へリプラットフォームする

作成者: Lanre (Lan-Ray) showunmi (AWS) と Tarun Chawla (AWS)

概要

Oracle Database エンタープライズエディション (EE) は、多くの企業でアプリケーションを実行する人気の選択肢です。ただし、アプリケーションによっては Oracle Database EE の機能をほとんどまたは全く使用しないため、莫大なライセンスコストが発生する正当性が欠落しています。Amazon RDS に移行するときに、このようなデータベースを Oracle Database 標準エディション 2 (SE2) にダウングレードすることで、コスト削減を実現できます。

このパターンは、オンプレミスから Amazon RDS for Oracle に移行する際に Oracle Database EE から Oracle Database SE2 にダウングレードする方法を説明しています。このパターンに示されている手順は、EE Oracle データベースがすでに Amazon RDS または Amazon Elastic Compute Cloud」(Amazon EC2) インスタンスで実行中の場合にも適用されます。

詳細については、Oracle データベースを AWS の標準エディション 2 へのダウングレードの評価方法に関する AWS 規範ガイダンスガイドを参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Oracle Database エンタープライズエディション
- Oracle SQL Developer または SQL*Plus など、Oracle データベースで SQL コマンドを接続および実行するクライアントツール
- 評価を実行するデータベースユーザー。たとえば、次のいずれか。
  - AWS Schema Conversion Tool (AWS SCT) 評価を実行する十分な特権があるユーザー
  - Oracle データベースディクショナリテーブルで SQL クエリを実行する十分な特権があるユーザー
- データベース移行を実行するデータベースユーザー。たとえば、次のいずれか。
  - AWS Database Migration Service (AWS DMS)を実行する十分な特権があるユーザー
  - Oracle Data Pump のエクスポートとインポートを実行する十分な権限があるユーザー
  - Oracle GoldenGate を実行する十分な特権があるユーザー

制限

- Amazon RDS for Oracle には最大データベースサイズがあります。詳細については、Amazon RDS DB インスタンスストレージを参照してください。

製品バージョン

本書で説明する一般的な論理は、Oracle の 9i 以降のバージョンに適用されます。サポートされているセルフマネージドデータベースと Amazon RDS for Oracle データベースのバージョンについては、AWS DMSドキュメントを参照してください。

AWS SCT がサポートされていない場合に、機能の使用状況を特定するには、ソースデータベースで SQL クエリを実行します。AWS DMS と Oracle Data Pump がサポートされていない Oracle の旧バージョンから移行するには、Oracle エクスポートユーティリティとインポートユーティリティを使用します。

サポートされているバージョンとエディションの最新リストについては、 AWS ドキュメントの Oracle on Amazon RDS を参照してください。価格設定とサポートされているインスタンスクラスの詳細については、Amazon RDS for Oracle の価格設定を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスまたは Amazon EC2 で実行中の Oracle Database エンタープライズエディション

ネイティブ Oracle ツールを使用したターゲットテクノロジースタック

- Oracle Database SE2 を実行中の Amazon RDS for Oracle

1. Oracle Data Pump を使用してデータをエクスポートします。
2. データベースリンクで Amazon RDS にダンプファイルをコピーします。
3. Oracle Data Pump を使用して Amazon RDS にダンプファイルをインポートします。

AWS DMS を使用したターゲットテクノロジースタック

- Oracle Database SE2 を実行中の Amazon RDS for Oracle

- AWS DMS

1. Oracle Data Pump と FLASHBACK_SCN を使用してデータをエクスポートします。
2. データベースリンクで Amazon RDS にダンプファイルをコピーします。
3. Oracle Data Pump を使用して Amazon RDS にダンプファイルをインポートします。
4. AWS DMS 変更データキャプチャ (CDC) を使用します。

ツール

AWS サービス

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。このパターンでは Amazon RDS for Oracle を使用します。
- AWS SCT は、ソース Oracle データベースのデータベーススキーマをAmazon RDS for Oracle と互換性のある形式に自動的に評価、変換、コピーするプロジェクトベースのユーザーインターフェイスを提供します AWS SCT を使用すると、ライセンスタイプを Oracle のエンタープライズから標準工ディション 2 に変更することで実現できる潜在的なコスト削減を分析できます。AWS SCT レポートの ライセンス評価とクラウドサポートセクションには、使用中の Oracle 機能に関する詳細情報が表示されるため、Amazon RDS for Oracle への移行時に情報に基づき決定できます。

その他のツール

- ネイティブ Oracle インポートとエクスポートのユーティリティは、Oracle データの Oracle データベース内外への移動をサポートします。Oracle は、オリジナルのエクスポートとインポート (以前のリリースの場合) と Oracle Data Pump のエクスポートとインポート (Oracle Database 10g リリース 1 以降で使用可能) という 2 つのタイプのデータベースのインポートとエクスポートユーティリティを提供しています。
- Oracle GoldenGate は、リアルタイムレプリケーション機能を提供しているため、初回のロード後にターゲットデータベースを同期できます。このオプションは、稼働開始時のアプリケーションのダウンタイムの削減に役立ちます。

エピック

移行前の評価をする

タスク	説明	必要なスキル
アプリケーションのデータベース要件を検証します。	アプリケーションを Oracle Database SE2 上で実行することが認定されていることを確認します。ソフトウェアベンダー、開発者に直接確認、またはアプリケーションドキュメントを参照します。	アプリ開発者、DBA、アプリ所有者
EE 機能の使用状況をデータベースで直接調査します。	EE 機能の使用を確認するには、以下のいずれかを実行します。 • Oracle EE データベース用の AWS SCT 評価レポートを生成します。ライセンスのタイプを変更する場合は、このレポートで、現在の機能からどの機能を削除する必要があるかがわかります。 • Oracle サポートアカウントがある場合は、サポートドキュメント 1317265.1 のスクリプト `options_packs_usage_statistics.sql` を取得して実行し、Oracle データベースで使用中のオプションと機能のレポートを生成します。 • DBA_FEATURE_USAGE_STATISTICS をクエリし	アプリ所有者、DBA、アプリ開発者

タスク	説明	必要なスキル
	て、使用中のすべての機能の詳細を表示します。	
運用アクティビティの EE 機能の用途を特定します。	データベースまたはアプリケーション管理者は、運用アクティビティの EE のみの機能に頼る場合があります。一般的な例には、オンライン保守アクティビティ (インデックスの再構築、テーブルの移動) 、バッチジョブによる並列処理の使用などがあります。 このような依存関係は、できる限り操作を変更することで軽減できます。これらの機能の用途を特定し、コストと利点の比較に基づき決定します。 Oracle Database EE と SE2 の機能の比較表をガイドとして使用して、Oracle Database SE2 で使用できる機能を特定します。	アプリ開発者、DBA、アプリ所有者

タスク	説明	必要なスキル
EE Oracle データベースのワークロードパターンを確認します。	Oracle Database SE2 は、いつでも使用量を最大 16 の CPU スレッドに自動的に制限します。 お使いの Oracle EE データベースに Oracle Diagnostic Pack の使用ライセンスが付与されている場合は、自動ワークロードリポジトリ (AWR) ツールまたは DBA_HIST_* ビューを使用してデータベースのワークロードパターンを分析し、SE2 にダウングレードしたときに 16 CPU スレッドの最大制限がサービスレベルに悪影響を及ぼすかどうかを判断します。 評価は、処理が日末、月末、または年末など、アクティビティのピーク期間を対象とするようにします。	アプリ所有者、DBA、アプリ開発者

AWS にターゲットインフラストラクチャを準備する

タスク	説明	必要なスキル
ネットワークインフラストラクチャをデプロイして設定します。	仮想プライベートクラウド (VPC) とサブネット、セキュリティグループ、ネットワークアクセスコントロールリストを作成します。	AWS 管理者、クラウドアーキテクト、ネットワーク管理者、DevOps エンジニア

タスク	説明	必要なスキル
Amazon RDS for Oracle SE2 データベースをプロビジョニングします。	ターゲットの Amazon RDS for Oracle SE2 データベースをプロビジョニングして、アプリケーションのパフォーマンス、可用性、セキュリティ要件を満たします。本稼働のワークロードにはマルチ AZ 設定をお勧めします。ただし、移行パフォーマンスを向上させるため、マルチ AZの有効化をデータ移行後まで延期できます。	クラウド管理者、クラウドアーキテクト、DBA、DevOps エンジニア、AWS 管理者
Amazon RDS 環境をカスタマイズします。	カスタムパラメータとオプションを設定し、追加監視を有効化します。詳細については、Amazon RDS for Oracle に移行するためのベストプラクティスを参照してください。	AWS 管理者、AWS システム管理者、クラウド管理者、DBA、クラウドアーキテクト

移行ドライラン、アプリケーションテストを実行する

タスク	説明	必要なスキル
データを移行 (ドライラン) します。	特定の環境に最適なアプローチを使用して、ソース Oracle EE データベースから Amazon RDS for Oracle SE2 データベースインスタンスにデータを移行します。サイズ、複雑性、使用可能ダウンタイムウィンドウなどの要因に基づき移行戦略を選択します。次	DBA

タスク	説明	必要なスキル
	のいずれか、または組み合わせを使用します。 • Oracle Data Pump (推奨)、Oracle インポート/エクスポートユーティリティ、Oracle GoldenGate などのネイティブ Oracle ツール。 • CDC による連続レプリケーションで全負荷を使用する AWS DMS。	
ターゲットデータベースを検証します。	データベースストレージとコードオブジェクトの移行後の検証を実行します。移行ログを確認し、特定された問題を修正します。詳細については、ガイド AWS クラウドへの Oracle データベースの移行を参照してください。	DBA
アプリケーションをテストします。	アプリケーション管理者およびデータベース管理者は、機能テスト、パフォーマンステスト、運用テストを適宜実施する必要があります。詳細については、Amazon RDS for Oracle に移行するためのベストプラクティスを参照してください。 最後に、ステークホルダーからテスト結果の承認を得ます。	アプリ開発者、アプリ所有者、DBA、移行エンジニア、移行リード

カットオーバー

タスク	説明	必要なスキル
Oracle データベース EE からのデータをリフレッシュします。	アプリケーションの可用性要件に基づきデータのリフレッシュ方法を選択します。詳細については、AWS への Oracle Database の移行戦略の移行方法を参照してください。 たとえば、継続的なレプリケーションを搭載した Oracle GoldenGate または AWS DMS などのツールを使用して、ダウンタイムほぼゼロを達成できます。ダウンタイム期間が許可する場合は、Oracle Data Pump またはオリジナルのエクスポート/インポートユーティリティなどのオフラインメソッドを使用して、最終データカットオーバーを実行できます。	アプリ所有者、カットオーバーリード、DBA、移行エンジニア、移行リード
アプリケーションをターゲットデータベースインスタンスにポイントします。	Amazon RDS for Oracle SE2 データベースを指すアプリケーションと他のクライアントの接続パラメータを更新します。	アプリ開発者、アプリ所有者、移行エンジニア、移行リード、カットオーバーリード
移行後のアクティビティを実行します。	マルチ AZ の有効化、データ検証、その他のチェックなど、データ移行後のタスクを実行します。	DBA、移行エンジニア

タスク	説明	必要なスキル
カットオーバー後の監視を実行します。	Amazon CloudWatch、Amazon RDS Performance Insights などのツールを使用して Amazon RDS for Oracle SE2 データベースを監視します。	アプリ開発者、アプリ所有者、AWS 管理者、DBA、移行エンジニア

関連リソース

AWS 規範ガイダンス

- 「AWS クラウドへの Oracle データベースの移行」(ガイド)
- Oracle データベースの標準エディション 2 へのダウングレードを評価する (ガイド)
- オンプレミス Oracle データベースを Amazon RDS for Oracle へ移行する (パターン)
- Oracle Data Pump を使用してオンプレミス Oracle データベースを Amazon RDS for Oracle へ移行する (パターン)

ブログ記事

- AWS DMS を使用したダウンタイムほぼゼロでの Oracle データベースの移行
- Amazon RDS for Oracle を使用した Oracle SE のパフォーマンス管理の分析
- Amazon RDS for Oracle による Oracle SE の SQL 計画の管理
- Oracle 標準エディションでのテーブルパーティションの実装: パート 1

Precisely Connect を使用してメインフレームデータベースを AWS にレプリケート

作成者：Lucio Pereira (AWS), Balaji Mohan (AWS) と Sayantan Giri (AWS)

概要

このパターンでは、Precisely Connect を使用してメインフレームデータベースから Amazon データストアにほぼリアルタイムでデータをレプリケーションする手順の概要を示しています。Amazon Managed Streaming for Apache Kafka (Amazon MSK) によるイベントベースのアーキテクチャと、クラウド内のカスタムデータベースコネクタを実装して、スケーラビリティ、耐障害性、パフォーマンスを向上させます。

Precisely Connect は、従来のメインフレームシステムからデータをキャプチャしてクラウド環境に統合するレプリケーションツールです。データは、低レイテンシーで高スループットの異種データパイプラインによるほぼリアルタイムのメッセージフローを使用して、変更データキャプチャ (CDC) を通じてメインフレームから AWS に複製されます。

このパターンには、マルチリージョンのデータ複製とフェールオーバールーティングによる回復力のあるデータパイプラインの災害復旧戦略も含まれます。

前提条件と制限

前提条件

- AWS クラウドに複製する既存のメインフレームデータベース (IBM DB2、IBM 情報管理システム (IMS)、仮想ストレージアクセスメソッド (VSAM) など)
- アクティブなAWSアカウント
- 企業環境から AWS への AWS Direct Connect または AWS 仮想プライベートネットワーク (AWS VPN)
- レガシープラットフォームからアクセス可能なサブネットを備えた仮想プライベートクラウド

アーキテクチャ

ソーステクノロジースタック

次のデータベースのうち少なくとも 1 つを含むメインフレーム環境

- IBM IMS データベース
- IBM DB2 データベース
- VSAM ファイル

ターゲットテクノロジースタック

- Amazon MSK
- Amazon Elastic Kubernetes Service (Amazon EKS) と Amazon EKS Anywhere
- Docker
- 次のような AWS リレーショナルまたは NoSQL データベース：
  - Amazon DynamoDB
  - Oracle の Amazon Relational Database Service (Amazon RDS)、Amazon RDS for PostgreSQL、または Amazon Aurora
  - Amazon ElastiCache for Redis
  - Amazon Keyspaces (Apache Cassandra 向け)

ターゲット アーキテクチャ

メインフレームデータの AWS データベースへのレプリケーション

次の図表では、メインフレームデータを DynamoDB、Amazon RDS、Amazon ElastiCache、Amazon Keyspaces などの AWS データベースに複製する方法を示しています。レプリケーションは、オンプレミスのメインフレーム環境では Precisely Capture and Publisher を使用し、オンプレミスの分散環境の Amazon EKS Anywhere では Precisely Dispatcher を使用し、AWS クラウドでは Precisely Apply Engine とデータベースコネクタを使用して、ほぼリアルタイムで行われます。

この図表は、次のワークフローを示しています：

1. Precisely Capture は CDC ログからメインフレームデータを取得し、そのデータを内部の一時ストレージに保持します。
2. Precisely Publisher は内部データストレージの変更を監視し、TCP/IP 接続を介して CDC レコードを Precisely Dispatcher に送信します。

3. Precisely Dispatcher はパブリッシャーから CDC レコードを受信し、Amazon MSK に送信します。発送者は、ユーザー設定と複数のワーカータスクに基づいて Kafka キーを作成し、データを平行でプッシュします。レコードが Amazon MSK に保存されると、発想者はパブリッシャーに確認応答を送り返します。
4. Amazon MSK は CDC レコードをクラウド環境に保持します。トピックのパーティションサイズは、使用するトランザクション処理システム (TPS) のスループット要件によって異なります。Kafka キーは、さらなる変換やトランザクションの順序付けには必須です。
5. Precisely Apply Engine は Amazon MSK からの CDC レコードを受信し、ターゲットデータベースの要件に基づいてデータを (フィルタリングやマッピングなどによって) 変換します。Precisely SQD スクリプトにはカスタマイズされたロジックを追加できます。(SQD は Precisely 独自の言語です)。Precisely Apply エンジンは、各 CDC レコードを Apache Avro または JSON 形式に変換し、要件に基づいてさまざまなトピックに配信します。
6. ターゲット Kafka トピックは、ターゲットデータベースに基づいて複数のトピックの CDC レコードを保持し、Kafka は定義済みの Kafka キーに基づいてトランザクションの順序付けを容易にします。パーティションキーは対応するパーティションと連動し、順次処理をサポートします。
7. データベースコネクタ (カスタマイズされた Java アプリケーション) は Amazon MSK の CDC レコードを受信し、ターゲットデータベースに保存します。
8. 要件に基づいて、ターゲットデータベースを選択できます。このパターンでは NoSQL データベースとリレーショナルデータベースをサポートします。

ディザスタリカバリ

ビジネス継続性は組織の成功の鍵です。AWS クラウドは高可用性 (HA) とディザスタリカバリ (DR) を実現する機能を提供し、組織のフェイルオーバープランとフォールバックプランをサポートします。このパターンでは、アクティブ/パッシブの DR 戦略に従い、RTO と RPO の要件を満たす DR 戦略を実装するための大まかなガイダンスを提供します。

次の図は、 のワークフローです。

図に示す内容は以下のとおりです。

1. AWS リージョン 1 で障害が発生した場合、半自動フェイルオーバーが必要です。リージョン 1 で障害が発生した場合、システムは Precisely Dispatcher をリージョン 2 に接続するためのルーティング変更を開始する必要があります。

2. Amazon MSK はリージョン間のミラーリングを通じてデータを複製します。そのため、フェイルオーバー時には、リージョン 2 の Amazon MSK クラスターをプライマリリーダーに昇格させる必要があります。
3. Precisely Apply Engine とデータベースコネクタは、どのリージョンでも動作するステートレスアプリケーションです。
4. データベースの同期は、ターゲットデータベースによって異なります。たとえば、DynamoDB はグローバルテーブルを使用でき、ElastiCache はグローバルデータストアを使用できます。

データベースコネクタによる低レイテンシーで高スループットの処理

このパターンでは、データベースコネクタが重要なコンポーネントです。コネクタはリスナーベースのアプローチに従い、Amazon MSK からデータを収集し、ミッションクリティカルなアプリケーション (階層 0 と 1) の高スループットで低レイテンシーの処理を通じてトランザクションをデータベースに送信します。次の図は、このプロセスを示したものです。

このパターンでは、マルチスレッド処理エンジンを通じてシングルスレッドで消費するカスタマイズされたアプリケーションの開発をサポートします。

1. コネクタのメインスレッドは Amazon MSK の CDC レコードを消費し、スレッドプールに送信して処理します。
2. スレッドプールのスレッドは CDC レコードを処理し、ターゲットデータベースに送信します。
3. すべてのスレッドがビジー状態の場合、CDC レコードはスレッドキューによって保留されます。
4. メインスレッドは、スレッドキューからすべてのレコードがクリアされるのを待って、Amazon MSK にオフセットをコミットします。
5. 子スレッドは障害を処理します。処理中に障害が発生した場合、失敗したメッセージは DLQ (デッドレターキュー) トピックに送信されます。
6. 子スレッドは、メインフレームのタイムスタンプに基づいて条件付き更新 (DynamoDB ドキュメントの条件式 を参照) を初期化し、データベースの重複や順序の狂った更新を回避します。

マルチスレッド機能を備えた Kafka コンシューマーアプリケーションを実装する方法については、Confluent ウェブサイトのブログ投稿Apache Kafka コンシューマーによるマルチスレッドメッセージ消費 を参照してください。

ツール

AWS サービス

- 「Amazon Managed Streaming for Apache Kafka (Amazon MSK)」 は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションを、構築および実行することを支援するフルマネージドサービスです。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、 で Kubernetes を実行する際に役立ちます。独自の Kubernetes コントロールプレーンまたはノードをインストールおよび維持する必要はありません。
- Amazon EKS Anywhere を使用すると、自社のデータセンターで実行される Kubernetes クラスターをデプロイ、使用、管理できます。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon ElastiCache は、AWS クラウドのインメモリ分散キャッシュ環境のセットアップ、管理、スケーリングに役立ちます。
- Amazon Keyspaces (Apache Cassandra に向け) は、AWS クラウドの Cassandra ワークロードの移行、実行、スケーリングを支援するマネージド型データベースサービスです。

その他のツール

- Precisely Connect は、VSAMデータセットやIBMメインフレームデータベースなどの従来のメインフレームシステムのデータを次世代のクラウドおよびデータプラットフォームに統合します。

ベストプラクティス

- 最適なパフォーマンスとコストのバランスを取るために、Kafka パーティションとマルチスレッドコネクタの最適な組み合わせを検出します。Precisely Capture インスタンスと Dispatcher インスタンスが複数あると、MIPS (1 秒あたり百万命令) の消費量が増えるため、コストが増加する可能性があります。
- データ操作や変換のロジックをデータベースコネクタに追加しません。そのためには、マイクロ秒単位の処理時間を実現する Precisely Apply エンジンを使用します。

- 接続を頻繁にウォームアップして待ち時間を短縮するために、データベースコネクターでデータベースへのリクエストまたはヘルスチェックコール (ハートビート) を定期的に作成します。
- スレッドプール検証ロジックを実装して、スレッドキュー内の保留中のタスクを把握し、すべてのスレッドが完了するまで待ってから次の Kafka ポーリングを行います。これにより、ノード、コンテナ、またはプロセスがクラッシュした場合にデータが失われることを回避します。
- ヘルスエンドポイントを通じてレイテンシーメトリクスを公開し、ダッシュボードやトレースメカニズムを通じてオブザーバビリティ機能を強化します。

エピック

ソース環境 (オンプレミス) の準備

タスク	説明	必要なスキル
メインフレームプロセス (バッチまたはオンラインユーティリティ) を設定して、メインフレームデータベースから CDC プロセスを開始します。	1. メインフレーム環境を特定します。 2. CDC プロセスに関係するメインフレームデータベースを特定します。 3. メインフレーム環境で、CDC ツールを起動してソースデータベースの変更をキャプチャするプロセスを開発します。手順については、メインフレームのマニュアルを参照します。 4. 設定を含んで、CDC プロセスを文書化します。 5. テスト環境と実稼働環境の両方にプロセスを導入します。	メインフレームエンジニア
メインフレームデータベースのログストリームを有効にします。	1. CDC ログをキャプチャするようにメインフレーム環境でログストリームを設定します。手順については、メ	メインフレーム DB スペシャリスト

タスク	説明	必要なスキル
	インフレームのマニュアルを参照します。 2. ログストリームをテストして、必要なデータがキャプチャされていることを確認します。 3. テスト環境および本番環境にログストリームをデプロイします。	

タスク	説明	必要なスキル
キャプチャーコンポーネントを使用して CDC レコードをキャプチャします。	1. Precisely Capture コンポーネントをメインフレーム環境にインストールして設定します。手順については、Preciselyのドキュメントを参照してください。 2. 構成をテストして、キャプチャーコンポーネントが正しく動作することを確認します。 3. キャプチャーした CDC レコードをキャプチャーコンポーネントを通じてレプリケーションするレプリケーションプロセスを設定します。 4. ソースデータベースごとにキャプチャー構成を文書化します。 5. キャプチャーコンポーネントが長期にわたってログを適切に収集するように監視システムを開発します。 6. テスト環境と実稼働環境にインストールと構成をデプロイします。	メインフレームエンジニア、Precisely Connect SME

タスク	説明	必要なスキル
キャプチャコンポーネントをリッスンするようにパブリッシャーコンポーネントを設定します。	1. Precisely Publisher コンポーネントをメインフレーム環境にインストールして設定します。手順については、Preciselyのドキュメントを参照してください。 2. 設定をテストして、パブリッシャーコンポーネントが正しく動作することを確認します。 3. CDC レコードをパブリッシャーから Precisely Dispatcher コンポーネントに公開するためのレプリケーションプロセスを設定します。 4. パブリッシャーの設定を文書化します。 5. パブリッシャーコンポーネントが長期にわたって適切に動作することを確認するための監視システムを開発します。 6. テスト環境と実稼働環境にインストールと構成をデプロイします。	メインフレームエンジニア、Precisely Connect SME

タスク	説明	必要なスキル
オンプレミスの分散環境で Amazon EKS Anywhere をプロビジョニングします。	1. Amazon EKS Anywhere をオンプレミスインフラストラクチャにインストールし、正しく設定されていることを確認します。手順については、Amazon EKS Anywhere のドキュメントを参照してください。 2. Kubernetes クラスターに、ファイアウォールを含む安全なネットワーク環境を設定します。 3. Amazon EKS Anywhere クラスターへのサンプルアプリケーションのデプロイを実装してテストします。 4. クラスターに自動スケール機能を実装します。 5. ネットワークのバックアップとリカバリーの手順を開発し、実施します。	DevOps エンジニア

タスク	説明	必要なスキル
発想者コンポーネントを分散環境にデプロイして設定し、AWS クラウドにトピックを公開します。	1. Precisely Dispatcher コンポーネントを設定してコンテナ化します。手順については、Preciselyのドキュメントを参照してください。 2. ディスパッチャー Docker イメージをオンプレミスの Amazon EKS Anywhere 環境にデプロイします。 3. AWS クラウドとディスパッチャー間の安全な接続を設定します。 4. 発送者コンポーネントが長期にわたって適切に動作することを確認するためのモニタリングシステムを開発します。 5. テスト環境と実稼働環境にインストールと構成をデプロイします。	DevOpsエンジニア、Precisely Connect SME

ターゲット環境 (AWS) の準備

タスク	説明	必要なスキル
指定された AWS リージョンに Amazon EKS クラスターをプロビジョニングします。	1. AWS アカウントにサインインし、Amazon EKS クラスターの作成と管理に必要な権限があることを確認するように設定します。 2. 選択した AWS リージョンに 仮想プライベートクラウド (VPC)とサブネットを	DevOps エンジニア、ネットワーク管理者

タスク	説明	必要なスキル
	作成します。手順については、Amazon EKS のドキュメントを参照してください。 3. Amazon EKS クラスターと VPC の他のリソース間の通信を可能にするために必要なネットワークセキュリティグループを作成して設定します。詳細については、Amazon EKS のドキュメント参照してください。 4. Amazon EKS クラスターを作成し、正しいノードグループ サイズとインスタンスタイプで設定します。 5. サンプルアプリケーションをデプロイして Amazon EKS クラスターを検証します。	

タスク	説明	必要なスキル
MSK クラスターをプロビジョニングし、該当する Kafka トピックを設定します。	1. AWS アカウントを設定して、MSK クラスターの作成と管理に必要な権限があることを確認します。 2. MSK クラスターと VPC の他のリソース間の通信を可能にするために必要なネットワークセキュリティグループを作成して設定します。詳細については、Amazon VPC のドキュメント参照してください。 3. MSK クラスターを作成し、アプリケーションが使用する Kafka トピックを含むように設定します。詳細については、Amazon MSK のドキュメント参照してください。	DevOps エンジニア、ネットワーク管理者

タスク	説明	必要なスキル
レプリケーションされた Kafka トピックを聞くように Apply Engine コンポーネントを設定します。	1. Precisely Apply Engine コンポーネント を設定してコンテナ化します。 2. アプライエンジンの Docker イメージを AWS アカウントの Amazon EKS クラスターにデプロイします。 3. MSK トピックを聞くように適用エンジンを設定します。 4. Apply Engine で、フィルタリングと変換を処理する SQD スクリプトを開発して設定します。詳細については、Preciselyのドキュメントを参照してください。 5. テスト環境および本番環境に Apply Engine をデプロイします。	Precisely Connect SME

タスク	説明	必要なスキル
AWS クラウドで DB インスタンスをプロビジョニングします。	1. DB クラスターとテーブルの作成と管理に必要な権限があることを確認するように AWS アカウントを設定します。手順について、使用する AWS データベースサービスの AWS ドキュメントを参照してください。(リンクについては、リソース セクションを参照してください)。 2. 選択された AWS リージョンに VPC とサブネットを作成します。 3. DB インスタンスと VPC の他のリソース間の通信を可能にするために必要なネットワークセキュリティグループを作成して設定します。 4. データベースを作成し、アプリケーションが使用するテーブルを含むように設定します。 5. データベーススキーマを設計、検証します。	データエンジニア、DevOps エンジニア

タスク	説明	必要なスキル
Apply Engine が公開するトピックを聞くためのデータベースコネクタを設定してデプロイします。	1. Kafka トピックを前のステップで作成した AWS データベースに接続するデータベースコネクタを設計します。 2. ターゲットデータベースに基づいてコネクタを開発します。 3. Apply Engine によって公開された Kafka トピックを聞くようにコネクタを設定します。 4. コネクタを Amazon EKS クラスターにデプロイします。	アプリ開発者、クラウドアーキテクト、データエンジニア

事業継続とディザスタリカバリの設定

タスク	説明	必要なスキル
ビジネスアプリケーションのディザスタリカバリの目標を定義します。	1. ビジネスニーズと影響分析に基づいて CDC パイプラインの RPO と RTO の目標を定義します。 2. すべての関係者がディザスタリカバリ計画を了解できるように、コミュニケーションと通知の手順を定義します。 3. ディザスタリカバリ計画の実施に必要な予算とリソースを決定します。	クラウドアーキテクト、データエンジニア、アプリオーナー

タスク	説明	必要なスキル
	4. RPO と RTO の目標を含むディザスタリカバリ目標を文書化します。	
定義した RTO/RPO に基づいてディザスタリカバリ戦略を設計します。	1. 重要度とリカバリ要件に基づいて、CDC パイプラインに最も適したディザスタリカバリ戦略を決定します。 2. ディザスタリカバリのアーキテクチャとトポロジーを定義します。 3. CDC パイプラインのフェールオーバーとフェールバックの手順を定義して、バックアップリージョンに迅速かつシームレスに切り替えられるようにします。 4. ディザスタリカバリの戦略と手順を文書化し、すべての利害関係者が設計を明確に理解していることを確認します。	クラウドアーキテクト、データエンジニア

タスク	説明	必要なスキル
ディザスタリカバリのクラスタと構成をプロビジョニングします。	1. ディザスタリカバリのセカンダリ AWS リージョンをプロビジョニングします。 2. セカンダリ AWS リージョンで、プライマリ AWS リージョンと同じ環境を作成します。 3. プライマリーリージョンとセカンダリージョンの間で Apache Kafka MirrorMaker を設定します。詳細については、Amazon MSK のドキュメント参照してください。 4. セカンダリリージョンで、スタンバイアプリケーションを設定します。 5. プライマリーリージョンとセカンダリージョンの間のデータベースレプリケーションを設定します。	DevOpsエンジニア、ネットワーク管理者、クラウドアーキテクト

タスク	説明	必要なスキル
ディザスタリカバリ の CDC パイプラインをテストします。	1. CDC パイプラインのディザスタリカバリテストの範囲と目標 (テストシナリオや達成すべき RTO など) を定義します。 2. ディザスタリカバリテストを実施するためのテスト環境とインフラストラクチャを識別します。 3. 障害シナリオをシミュレートするためのテストデータセットとスクリプトを準備します。 4. データの整合性と一貫性を検証して、データ損失がないことを確認します。	アプリオーナー、データエンジニア、クラウドアーキテクト

関連リソース

「AWS リソース」

- Amazon DynamoDB
- Amazon DynamoDB による条件式
- Amazon EKS
- Amazon EKS Anywhere
- Amazon ElasticCache
- Amazon Keyspaces
- Amazon MSK
- Amazon RDS と Amazon Aurora
- Amazon VPC

Precisely Connect リソース

- Precisely Connect の概要
- Precisely Connect による変更データキャプチャ

コンフルエントリソース

- Apache Kafka コンシューマーによるマルチスレッドメッセージ消費

Lambda と Secrets Manager を使用して Amazon RDS for PostgreSQL と Aurora PostgreSQL のジョブをスケジュールする

作成者: Yaser Raja (AWS)

概要

オンプレミスデータベースと Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされているデータベースの場合、データベース管理者は cron ユーティリティを使用してジョブをスケジュールすることがよくあります。

例えば、データ抽出用のジョブやデータ消去用のジョブは、cron を使用して簡単にスケジュールできます。これらのジョブでは、通常、データベースの認証情報はハードコーディングされるか、プロパティファイルに保存されます。ただし、Amazon Relational Database Service (Amazon RDS) または Amazon Aurora PostgreSQL 互換エディションに移行すると、ホストインスタンスにログインして cron ジョブをスケジュールすることができなくなります。

このパターンでは、 AWS Lambda と を使用して AWS Secrets Manager 、移行後に Amazon RDS for PostgreSQL および Aurora PostgreSQL 互換データベースのジョブをスケジュールする方法について説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換データベース

制限

- ジョブは Lambda 関数のタイムアウト制限である 15 分以内に完了する必要があります。その他の制限については、AWS Lambda 「 ドキュメント」を参照してください。
- Job コードは Lambda がサポートする言語で記述する必要があります。

アーキテクチャ

ソーステクノロジースタック

このスタックには、Bash、Python、Java などの言語で記述されたジョブが含まれています。データベースの認証情報はプロパティファイルに保存され、ジョブは Linux cron を使用してスケジュールされます。

ターゲットテクノロジースタック

このスタックには、Secrets Manager に保存されている認証情報を使用してデータベースに接続し、アクティビティを実行する Lambda 関数があります。Lambda 関数は、Amazon CloudWatch Events を使用してスケジュールされた間隔で開始されます。

ターゲットアーキテクチャ

ツール

- Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。CloudWatch Events が発生すると、運用上の変更が認識されます。オペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、修正アクションを実行します。CloudWatch Events を使用して、cron式 や rate式により特定の時間に自己トリガーする自動アクションをスケジュールすることもできます。
- AWS Lambda はサーバーのプロビジョニングや管理をする必要がなく、コードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。使用したコンピューティング時間に対してのみお支払いいただきます- コードが実行中でなければ料金はかかりません。Lambda を使用すると、ほぼすべてのタイプのアプリケーションまたはバックエンドサービスのコードを管理なしで実行できます。Lambda は、高可用性コンピューティングインフラストラクチャでコードを実行し、サーバーとオペレーティングシステムのメンテナンス、容量のプロビジョニングと自動スケーリング、コードのモニタリング、ログ記録など、すべてのコンピューティングリソースを管理します。必要なのは、Lambda がサポートする言語のいずれかでコードを提供することだけです。
- AWS Secrets Manager は、アプリケーション、サービス、IT リソースにアクセスするためのシークレットを保護するのに役立ちます。データベース認証情報、API キー、その他のシークレットをライフサイクル全体にわたって簡単にローテーション、管理、取得できます。ユーザーとアプリケーションは Secrets Manager API を呼び出してシークレットを取得するため、機密情報をプ

レーンテキストでハードコーディングする必要がなくなります。Secrets Manager には、Amazon RDS、Amazon Redshift、Amazon DocumentDB の統合機能が組み込まれたシークレットローテーションが用意されています。このサービスは API キーや OAuth トークンなど、他のタイプのシークレットにも拡張できます。Secrets Manager を使用すると、きめ細かなアクセス許可を使用してシークレットへのアクセスを制御し、シークレットローテーションを一元的に監査して AWS クラウド、、サードパーティーのサービス、オンプレミスのリソースを監査できます。

エピック

Secrets Manager にデータベース認証情報を保存する

タスク	説明	必要なスキル
Lambda 関数用のデータベースユーザーを作成します。	アプリケーションのさまざまな部分に別々のデータベースユーザーを使用することをお勧めします。cron ジョブ用に別のデータベースユーザーが既に存在する場合は、そのユーザーを使用してください。それ以外の場合は、新しいデータベースユーザーを作成します。詳細については、PostgreSQL ユーザーとロールの管理」(AWS ブログ記事) を参照してください。	DBA
Secrets Manager でデータベース認証情報をシークレットとして保存します。	「AWS Secrets Manager データベースシークレットを作成する」(Secrets Manager ドキュメント) の指示に従います。	DevOps、DBA

Lambda 関数のコードを作成する

タスク	説明	必要なスキル
Lambda でサポートされているプログラミング言語を選択します。	サポートされている言語のリストについては、「Lambda ランタイム」(Lambda ドキュメント) を参照してください。	開発者
Secrets Manager からデータベース認証情報を取得するロジックを記述します。	サンプルコードについては、「 を使用して Lambda 関数にデータベース認証情報を安全に提供する方法 AWS Secrets Manager」 (AWS ブログ記事) を参照してください。	開発者
スケジュールされたデータベースアクティビティを実行するロジックを記述します。	オンプレミスで使用しているスケジューリングジョブの既存のコードを Lambda 関数に移行します。詳細については、「Lambda 関数のデプロイ」(Lambda ドキュメント) を参照してください。	開発者

コードをデプロイして Lambda 関数を作成する

タスク	説明	必要なスキル
Lambda 関数デプロイパッケージを作成します。	このパッケージには、コードとその依存関係が含まれます。詳細については、「デプロイパッケージ」(Lambda ドキュメント) を参照してください。	開発者
Lambda 関数を作成します。	Lambda コンソールで、関数の作成を選択し、関数名を	DevOps

タスク	説明	必要なスキル
	入力し、ランタイム環境を選択し、関数の作成を選択します。	
デプロイパッケージをアップロードする	作成した Lambda 関数を選択し、設定を開きます。コードをコードセクションに直接記述することも、デプロイパッケージをアップロードすることもできます。パッケージをアップロードするには、[Function code] セクションに移動し、[コードエントリタイプ] を選択して.zip ファイルをアップロードし、パッケージを選択します。	DevOps
Lambda 関数を要件に合わせて設定します。	例えば、Timeout パラメータを Lambda 関数にかかると予想される時間に設定できます。詳細については、「関数オプションの設定 (Lambda ドキュメント)」を参照してください。	DevOps
Lambda 関数ロールが Secrets Manager にアクセスするためのアクセス権限を設定します。	手順については、「 AWS Lambda 関数でシークレットを使用する」(Secrets Manager ドキュメント) を参照してください。	DevOps
Lambda 関数をテストします。	Lambda 関数を手動で開始して、正常に動作することを確認します。	DevOps

CloudWatch イベントを使用して Lambda 関数をスケジュールします

タスク	説明	必要なスキル
Lambda 関数をスケジュールに従って実行するルールを作成します。	CloudWatch イベントを使用して Lambda 関数をスケジュールします。手順については、「Schedule Lambda functions using CloudWatch Events」(CloudWatch イベントチュートリアル) を参照してください。	DevOps

関連リソース

- AWS Secrets Manager
- Lambda の使用開始
- イベントでトリガーする CloudWatch Events のルールの作成
- AWS Lambda 制限
- サーバーレスアプリケーションから AWS データベースをクエリする (ブログ記事)

オンプレミスの SMTP サーバーとデータベースメールを使用して、Amazon RDS for SQL Server データベースインスタンスに通知を送信します。

ニシャド・マンカー (AWS) によって作成されました

概要

「データベースメール」 (Microsoft ドキュメント) は、簡易メール転送プロトコル (SMTP) サーバーを使用して Microsoft SQL Server データベースから通知や警告などの電子メールメッセージを送信します。Microsoft SQL Server 用 Amazon Relational Database Service (Amazon RDS) のドキュメントには、Amazon Simple Email Service (Amazon SES) をデータベースメールの SMTP サーバーとして使用する方法が記載されています。詳細については、「Amazon RDS for SQL Server でのデータベースメールの使用」を参照してください。代替設定として、このパターンでは、オンプレミスの SMTP サーバーをメールサーバーとして使用して Amazon RDS for SQL Server データベース (DB) インスタンスから E メールを送信するようにデータベースメールを設定する方法を説明します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- SQL Server のスタンダードエディションまたはエンタープライズエディションを実行する Amazon RDS DB インスタンス
- オンプレミスの SMTP サーバーの IP アドレスまたはホスト名
- SMTP サーバーの IP アドレスから Amazon RDS for SQL Server DB インスタンスへの接続を許可するインバウンド 「セキュリティグループルール」
- オンプレミスネットワークと Amazon RDS DB インスタンスを含む仮想プライベートクラウド (VPC) 間の接続 (「AWS Direct Connect」 接続など)

機能制限

- SQL Server のエクスプレスエディションはサポートされていません。
- 制限に関する詳細については、Amazon RDSドキュメントの Amazon RDS for SQL Server Database Mail の使用に関する 「制限」 を参照してください。

製品バージョン

- 「RDS でサポートされる SQL Server バージョン」 のスタンダードエディションとエンタープライズエディション

アーキテクチャ

ターゲットテクノロジースタック

- Amazon RDS for SQL Server データベースインスタンス
- Amazon Route 53 転送ルール
- データベースメール
- オンプレミスの SMTP サーバー
- Microsoft SQL Server Management Studio (SSMS)

ターゲットアーキテクチャ

次の図は、このパターンのターゲットアーキテクチャを示しています。データベースインスタンスに関する通知またはアラートを開始するイベントまたはアクションが発生すると、Amazon RDS for SQL Server はデータベースメールを使用して E メール通知を送信します。データベースメールはオンプレミスの SMTP サーバーを使用して E メールを送信します。

ツール

AWS サービス

- 「Microsoft SQL サーバーの Amazon Relational Database Service (Amazon RDS)」 を使用して、AWS クラウドで SQL サーバーリレーショナルデータベースをセット、運用、スケーリングすることを支援します。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

その他のツール

- 「データベースメール」 は、通知やアラートなどの E メールメッセージを SQL Server データベースエンジンからユーザーに送信するツールです。

- 「Microsoft SQL Server Management Studio (SSMS)」 は、SQL Server コンポーネントへのアクセス、設定、管理など、SQL Server を管理するためのツールです。このパターンでは、SSMS を使用して SQL コマンドを実行し、Amazon RDS for SQL Server DB インスタンスにデータベースメールを設定します。

エピック

オンプレミスの SMTP サーバーとのネットワーク接続を有効にします。

タスク	説明	必要なスキル
RDS DB インスタンスからマルチ AZ を削除します。	マルチゾーン RDS DB インスタンスを使用している場合は、マルチ AZ インスタンスをシングル AZ インスタンスに変換します。データベースメールの設定が完了したら、DB インスタンスをマルチ AZ 配置に戻します。次に、プライマリノードとセカンダリノードの両方に、データベースメールの設定が実行します。説明については、「Microsoft SQL Server DB インスタンスからのマルチ AZ の削除」 を参照してください。	DBA
Amazon RDS エンドポイントまたは IP アドレスの許可リストを、オンプレミスの SMTP サーバー上に作成します。	SMTP サーバーは AWS ネットワークの外部にあります。オンプレミスの SMTP サーバーで、Amazon RDS インスタンスまたは Amazon RDS でホストされている Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのアウトバウンドエン	DBA

タスク	説明	必要なスキル
	ドポイントまたは IP アドレスと通信することを許可する許可リストを作成します。この手順は組織によって異なります。DB インスタンスエンドポイントの詳細については、「DB インスタンスのエンドポイントとポート番号の検索」を参照してください。	

タスク	説明	必要なスキル
ポート 25 の制限を解除します。	デフォルトでは、AWS は EC2 インスタンスのポート 25 を制限します。ポート 25 の制限を解除するには、次の手順を実行します。 1. AWS アカウントでサインインし、「E メール送信制限解除のリクエストフォーム」 を開きます。 2. リクエストに関する最新情報を AWS サポート から連絡できるように、E メールアドレスを入力します。 3. ユースケースの説明フィールドに必要な情報を入力します。 4. [送信] を選択します。 Note • 複数の AWS リージョンにインスタンスがある場合は、リージョンごとに個別のリクエストを送信します。 • リクエストの処理には最大 48 時間かかります。	AWS 全般

タスク	説明	必要なスキル
Route 53 ルールを追加して、SMTP サーバーの DNS クエリを解決します。	Route 53 を使用して、AWS リソースとオンプレミスの SMTP サーバー間の DNS クエリを解決します。DNS クエリを SMTP サーバードメイン (`example.com` など) に転送するルールを作成する必要があります。手順については、Route 53 ドキュメントの「転送ルールの作成」を参照してください。	ネットワーク管理者

Amazon RDS for SQL Server DB インスタンスでのデータベースメールのセットアップ

タスク	説明	必要なスキル
データベースメールを有効化します。	データベースメールのパラメータグループを作成し、`database mail xps` パラメータを 1 に設定し、データベースメールパラメータグループをターゲット RDS DB インスタンスに関連付けます。手順については、Amazon RDS ドキュメントの「データベースメールの有効化」を参照してください。この説明のデータベースメールの設定セクションには進まないでください。オンプレミスの SMTP サーバーの設定は Amazon SES とは異なります。	DBA

タスク	説明	必要なスキル
DB インスタンスに接続します。	踏み台ホストから、Microsoft SQL Server Management Studio (SSMS) を使用して Amazon RDS for SQL Server データベースインスタンスに接続します。説明については、「Microsoft SQL Server データベースエンジンを実行する DB インスタンスに接続する」 を参照してください。エラーが発生した場合は、「関連リソース」 セクションの「接続トラブルシューティングに関する参考資料」を参照してください。	DBA

タスク	説明	必要なスキル
プロファイルを作成します。	SSMS で、以下の SQL ステートメントを入力してデータベースメールプロファイルを作成します。以下の値を置き換えます: • `profile_name` の場合、新しいプロファイルの名前を入力します。 • `description` に、新しいプロファイルの簡単な説明を入力します。 このストアドプロシージャとその引数の詳細については、Microsoft のドキュメントの「sysmail_add_profile_sp」を参照してください。 `EXECUTE msdb.dbo.sysmail_add_profile_sp @profile_name = 'SQL Alerts profile', @description = 'Profile used for sending outgoing notifications using OM SMTP Server.';`	DBA

タスク	説明	必要なスキル
プリンシパルをプロファイルに追加します。	次の SQL ステートメントを入力して、パブリック・プリンシパルまたはプライベート・プリンシパルをデータベース・メール・プロファイルに追加します。プリンシパルは、SQL Server リソースをリクエストできるエンティティです。以下の値を置き換えます: • `profile_name` には、事前に作成したプロファイルの名前を入力します。 • `principal_name` には、データベースユーザーまたはロールの名前を入力します。この値は、SQL Server 認証ユーザー、Windows 認証ユーザー、または Windows 認証グループにマッピングする必要があります。 このストアドプロシージャとその引数の詳細については、Microsoft のドキュメントの「sysmail_add_profile_sp」を参照してください。 `EXECUTE msdb.dbo.sysmail_add_principalprofile_sp @profile_name = 'SQL Alerts profile',`	DBA

タスク	説明	必要なスキル
	`@principal_name = 'public', @is_default = 1 ;`	

タスク	説明	必要なスキル
アカウントを作成します。	以下の SQL ステートメントを入力して、データベースメールアカウントを作成します。以下の値を置き換えます: • account_name に、新しいアカウントの名前を入力します。 • description に、新しいアカウントの簡単な説明を入力します。 • email_address に、データベースメールメッセージの送信元の電子メールアドレスを入力します。 • display_address に、このアカウントの送信メッセージに使用する表示名 (SQL Server Automated Notification など) を入力します。email_address に入力した値を使用することもできます。 • mailserver_name には、SMTP メールサーバーの名前または IP アドレスを入力します。 • port については、25 の値はそのままにします。 • enable_ssl では、1 値をのままにするか、データベースメールに SSL 0 を使用して通信を暗号化させた	DBA

タスク	説明	必要なスキル
	くない場合は入力してください。 • username に、SMTP メールサーバーにログオンするためのユーザー名を入力します。サーバーが認証を必要としない場合は、NULL と入力します。 • password の場合、SMTP メールサーバーにログオンするためのパスワードを入力します。サーバーが認証を必要としない場合は、NULL と入力します。 このストアドプロシージャとその引数の詳細については、Microsoft 「のドキュメントの sysmail_add_account_sp」 を参照してください。 `EXECUTE msdb.dbo.sysmail_add_account_sp @account_name = 'SQL Alerts account', @description = 'Database Mail account for sending outgoing notifications.', @email_address = 'xyz@example.com', @display_name = 'xyz@example.com', @mailserver_name = 'test_smtp.example.com',`	

タスク	説明	必要なスキル

	<pre>@port = 25,
	@enable_ssl = 1,
	@username = 'SMTP-username',
	@password = 'SMTP-password';</pre>	

プロファイルにアカウントを追加します。	以下の SQL ステートメントを入力して、データベースメールアカウントをデータベースメールプロファイルに追加します。以下の値を置き換えます:
	• `profile_name` には、事前に作成したプロファイルの名前を入力します。
	• `account_name` の場合、事前に作成したアカウント名を入力します。
	
	このストアドプロシージャとその引数の詳細については、Microsoft のドキュメントの「sysmail_add_profileaccount_sp」を参照してください。
	<pre>EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
	@profile_name = 'SQL Alerts profile',
	@account_name = 'SQL Alerts account',
	@sequence_number = 1;</pre>
		DBA

タスク	説明	必要なスキル
(オプション) RDS DB インスタンスにマルチ AZ を追加します。	マルチ AZ をデータベースミラーリング (DBM) または AlwaysOn 可用性グループ (AG) で追加する場合は、「Microsoft SQL Server DB インスタンスへのマルチ AZ の追加」を参照してください。	DBA

関連リソース

- 「Amazon RDS for SQL Server でのデータベースメールの使用」(Amazon RDS ドキュメント)
- 「添付ファイルの処理」(Amazon RDS ドキュメント)
- 「SQL Server DB インスタンスへのトラブルシューティング接続」(Amazon RDS ドキュメント)
- 「Amazon RDS DB インスタンス」に接続できない (Amazon RDS ドキュメント)

AWS の IBM Db2 に SAP のディザスタリカバリをセットアップ

作成者: Ambarish Satarkar (AWS) と Debasis Sahoo (AWS)

概要

このパターンでは、Amazon Web Services（AWS）クラウドで稼働する IBM Db2 をデータベースプラットフォームとして使用する、SAP ワークロードのディザスタリカバリ（DR）システムをセットアップする手順の概要を示します。目的は、ディザスタが発生した場合でも、事業を継続できる低コストのソリューションを提供することです。

このパターンでは、「パイロットライト方式」を使用します。AWS にパイロットライト DR を実装することで、ダウンタイムを削減し、ビジネスの継続性を維持できます。パイロットライトアプローチは、SAP システムやスタンバイ Db2 データベースなど、本番環境と同期する最小限の DR 環境を AWS に設定することに重点を置いています。

このソリューションはスケーラブルです。必要に応じて本格的なディザスタリカバリ環境に拡張できます。

前提条件と制限

前提条件

- レプリケーションインスタンスは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されます。
- IBM Db2 データベース
- SAP 製品可用性マトリックス (PAM) が適用されるオペレーティングシステム
- 本番データベースホストとスタンバイデータベースホストで異なる物理データベースホスト名
- 各 AWS リージョンでは、「クロスリージョンレプリケーション (CRR)」が有効になっている Amazon Simple Storage Service (Amazon S3) バケット

製品バージョン

- IBM Db2 データベースバージョン 11.5.7 以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EC2
- Amazon Simple Storage Service (Amazon S3)
- Amazon Virtual Private Cloud (VPC ピアリング)
- Amazon Route 53
- IBM Db2 高可用性ディザスタリカバリ (HADR)

ターゲットアーキテクチャ

このアーキテクチャーは、データベースプラットフォームとして Db2 を使用する SAP ワークロード用の DR ソリューションを実装します。本番データベースは AWS リージョン 1 にデプロイされ、スタンバイデータベースは 2 番目のリージョンにデプロイされます。スタンバイデータベースは DR システムを指します。Db2 Database には、複数のスタンバイデータベース (最大 3 つ) が適用されます。Db2 HADR を使用して DR データベースを設定し、本番データベースとスタンバイデータベース間のログ配布を自動化します。

リージョン 1 が使用できなくなるようなディザスタが発生した場合、DR リージョンのスタンバイデータベースが本番データベースの役割を引き継ぎます。SAP アプリケーションサーバーは、目標復旧時間 (RTO) の要件を満たすために、事前に構築することも、「AWS エラスティックディザスタリカバリ」 や Amazon マシンイメージ (AMI) を使用して構築することもできます。このパターンでは AMI を使用します。

Db2 HADR は、本番がプライマリサーバーとして機能し、すべてのユーザーがそのサーバーに接続されます。本番とスタンバイのセットアップを実装しています。すべてのトランザクションがログファイルに書き込まれ、TCP/IP を使用してスタンバイサーバーに転送されます。スタンバイサーバーは、転送されたログレコードをロールフォワードしてローカルデータベースを更新します。これにより、本番サーバーとの同期が保持されます。

VPC ピアリングは、本番リージョンと DR リージョンのインスタンスが相互に通信できるようにするために使用されます。Amazon Route 53 は、エンドユーザーをインターネットアプリケーションにルーティングします。

1. リージョン 1 にアプリケーションサーバーの 「AMI を作成」し、その AMI をリージョン 2 に「コピーします」 。ディザスタが発生した場合、AMI を使用してリージョン 2 のサーバーを起動します。
2. 本番データベース (リージョン 1) とスタンバイデータベース (リージョン 2) の間で Db2 HADR レプリケーションを設定します。

3. ディザスタが発生した時、本番インスタンスに合わせて EC2 インスタンスタイプを変更します。
4. リージョン 1 では、LOGARCHMETH1 が db2remote: S3 path に設定されます。
5. リージョン 2 では、LOGARCHMETH1 が db2remote: S3 path に設定されます。
6. クロスリージョンレプリケーションは S3 バケット間で実行されます。

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。このパターンでは、「VPC ピアリング」を使用します。

ベストプラクティス

- ネットワークは HADR のレプリケーションモードを決定する上で重要な役割を果たします。AWS リージョン間の DR には、Db2 HADR ASYNC モードまたは SUPERASYNC モードを使用することを推奨します。
- Db2 HADR のレプリケーションモードの詳細については、「IBM のドキュメント」 を参照してください。
- 既存の SAP システムの「新しい AMI を作成」するには、AWS マネジメントコンソールまたは AWS コマンドラインインターフェイス (AWS CLI) を使用できます。その後、AMI を使用して既存の SAP システムを復元したり、クローンを作成したりできます。
- AWS Systems Manager Automation は、Amazon EC2 インスタンスおよび他の AWS リソースの一般的なメンテナンスとデプロイのタスクを簡略化します。
- AWS では、AWS のインフラストラクチャとアプリケーションを監視および管理するための複数のネイティブサービスを提供します。Amazon CloudWatch や AWS CloudTrail などのサービスを

使用して、基盤となるインフラストラクチャと API オペレーションをそれぞれ監視できます。詳細については、「SAP on AWS — IBM Db2 HADR with Pacemaker」を参照してください。

エピック

環境の準備

タスク	説明	必要なスキル
システムとログをチェックします。	1. 本番環境の SAP on Db2 システムがセットアップされていることを確認します。 2. ログバックアップが有効になって、S3 バケットにログを保存するように設定されていることを確認します。Db2 パラメータ `LOGARCHMETH1` で確認できます。 3. 追加のアプリケーションサーバーの AMI を作成します。	AWS 管理者、SAP ベーシス管理者

サーバーとレプリケーションをセットアップする

タスク	説明	必要なスキル
SAP サーバーとデータベースサーバーを作成します。	1. DR リージョンのインフラストラクチャをデプロイするには、AWS CloudFormation スクリプトを使用するか、本番インスタンスの AMI を使用します。パイロットライトアプローチの一部として、本番インスタンスと同じファミ	SAP ベーシス管理者

タスク	説明	必要なスキル
	リーの小規模な EC2 インスタンスを使用できます。たとえば、本番インスタンスタイプが `r6i.12xlarge` の場合、DR ビルドの `r6i.xlarge` のインスタンスタイプを使用できます。ただし、本番環境のデータベースバックアップを復元するには、必ず DR インスタンスに同じストレージ容量を割り当てます。 2. `/sapmnt/<SID>/` の Amazon Elastic File System (Amazon EFS) マウントポイントを作成し、プライマリシステムから「複製されるように設定」されることを保証します。 3. 本番システムからデータベースの完全バックアップ (オンラインまたはオフライン) を行います。このバックアップを使用して DR データベースを構築します。 4. DR システムでは、「HA/DR 目的のバックアップ/復元によるシステムコピーを使用」を用いて、 SAP ソフトウェアプロビジョニングマネージャー (SWPM) のシステムコピー方法を使用	

タスク	説明	必要なスキル
	してSAP システムを構築します。 5. SWPM から依頼された場合、本番環境から取得したバックアップを使用して DR 内のデータベースを復元します。DR データベースはロールフォワード保留状態になります。 ロールフォワード保留状態は、フルバックアップの復元後にデフォルトで設定されます。ロールフォワード保留状態は、データベースが復元中で、何らかの変更を適用する必要がある可能性があることを示します。詳細については、「IBM ドキュメント」を参照してください。	

タスク	説明	必要なスキル

設定を確認します。	1. HADR のログアーカイブを設定するには、本番データベースと DR データベースの両方が、すべてのログアーカイブの場所から自動的にログを取得できる必要があります。DR データベースの LOGARCHMETH1 パラメータが本番データベースと同じ場所に設定されていることを検証します。リージョンの制限により同じ場所がアクセス不能な場合、DR システムがプライマリシステムからログを自動的に取得できることを確保します。
	2. データベースレプリケーションを有効にするための TCP/IP ポートを有効にするには、次の 2 つのエントリを追加することで、本番ホストの /etc/services と DR ホストを変更します。コードでは、<SID> が Db2 データベースのシステム ID (SID)を指します (例: PR1)。
	`<SID>_HADR_1 55001/tcp # DB2 HADR Port1`
	`<SID>_HADR_2 55002/tcp # DB2 HADR Port2`
		AWS 管理者、SAP ベーシス管理者

タスク	説明	必要なスキル
	両方のポートでは、プライマリとスタンバイ間のインバウンドトラフィックとアウトバウンドトラフィックが許可されていることを確認します。 3. 本番ホストと DR ホストで `/etc/hosts` をチェックインして、本番ホストとスタンバイホストの両方のホスト名が正しい IP アドレスを指していることを確認します。	

タスク	説明	必要なスキル

本番 DB から DR DB へのレプリケーションを設定します (ASYNC モードを使用)。	1. 本番データベースで、次のコマンドを実行してパラメータを更新します。
	
	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_HOST HOST1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_SVC <SID>_HADR_1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_HOST HOST2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_SVC <SID>_HADR_2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_INST db2<sid>`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_TIMEOUT 120`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SYNCMODE ASYNC`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SPOOL_LIMIT 1000`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_PEER_WINDOW 240`
	`db2 UPDATE DB CFG FOR <SID> USING indexrec RESTART logindexbuild ON`
	
	2. DR データベースで、次のコマンドを実行してパラメータを更新します。
		SAP ベーシス管理者

タスク	説明	必要なスキル

	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_HOST HOST2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_SVC <SID>_HADR_2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_HOST HOST1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_SVC <SID>_HADR_1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_INST db2<sid>`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_TIMEOUT 120`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SYNCMODE ASYNC`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SPOOL_LIMIT 1000`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_PEER_WINDOW 240`
	`db2 UPDATE DB CFG FOR <SID> USING indexrec RESTART logindexbuild ON`
	
	これらのパラメータは、HADR 関連の情報を両方のデータベースに提供するために必要です。Db2 データベースでは、HADR は以前に設定された各パラメータの値に基づいてアク	

タスク	説明	必要なスキル

	ティブ化されます。パラメータの詳細については、「IBM のドキュメント」を参照してください。
	3. 次のコマンドを使用して、新しく作成したスタンバイデータベースで HADR を最初に起動します。
	`db2 deactivate db <SID>`
	`db2 start hadr on db <SID> as standby`
	4. 次のコマンドを使用して、本番データベースで HADR を起動します。
	`db2 deactivate db <SID>`
	`db2 start hadr on db <SID> as primary`
	5. 本番とスタンバイの Db2 データベースが同期していて、ログ配信が進行中であるかどうかを確認します。
	HADR レプリケーションステータスを監視するには、以下の db2pd コマンドを使用します。
	`db2pd -d <SID> -hadr`
	HADR モニタリングの詳細については、「IBM ドキュ	

タスク	説明	必要なスキル
	メント」 を参照してください。	

DR フェイルオーバータスクをテスト

タスク	説明	必要なスキル
DR テストの本番ビジネスのダウンタイムを計画します。	DR フェイルオーバーシナリオをテストするために、必ず本番環境で必要な業務停止時間を計画するようにします。	SAP ベーシス管理者
テストユーザーを作成します。	DR ホストで検証できるテストユーザー (または任意のテスト変更) を作成して、DR フェイルオーバー後のログの複製を確認します。	SAP ベーシス管理者
コンソールで、本番環境の EC2 インスタンスを停止します。	このステップでは、ディザスタシナリオを想定して不適切なシャットダウンが開始されます。	AWS システム管理者
DR EC2 インスタンスを要件に合わせてスケールアップします。	EC2 コンソールで、DR リージョンのインスタンスタイプを変更します。 1. インスタンスを停止:インスタンスが実行中の場合、そのインスタンスを先に停止する必要があります。EC2 コンソールでインスタンスを選択し、 [停止] を選択します。	SAP ベーシス管理者

タスク	説明	必要なスキル

	2. インスタンスタイプを変更: EC2 コンソールでインスタンスを選択し、[アクション] [インスタンス設定] [インスタンスタイプを変更] を選択します。プライマリインスタンスと一致するインスタンスタイプを選択し、[適用] を選択します。
	3. インスタンスを起動: インスタンスタイプの変更が完了したら、EC2 コンソールからインスタンスを選択して、次に [開始] を選択してインスタンスを起動します。
	4. データベースを削除するには、次のコマンドを使用します。
	`db2start`
	`db2 start HADR on db <SID> as standby`	

タスク	説明	必要なスキル

テイクオーバーを初期化します。	DR システム (host2) から、テイクオーバープロセスを開始し、DR データベースをプライマリとして起動します。
	
	`db2 takeover hadr on database <SID> by force`
	
	オプションとして、以下のパラメータを設定して、インスタンスタイプに基づいてデータベースのメモリ割り当てを自動的に調整できます。INSTANCE_MEMORY の値は、Db2 データベースに割り当てられるメモリの専用部分に基づいて決定できます。
	
	`db2 update db cfg for <SID> using INSTANCE_MEMORY <FIXED VALUE> IMMEDIATE;`
	`db2 get db cfg for <SID> | grep -i DATABASE_MEMORY AUTOMATIC IMMEDIATE;`
	`db2 update db cfg for <SID> using self_tuning_mem ON IMMEDIATE;`
	
	次のコマンドを使用して変更を確認します。
	
	`db2 get db cfg for <SID> | grep -i MEMORY`
		SAP ベーシス管理者

タスク	説明	必要なスキル
	`db2 get db cfg for <SID> | grep -i self_tuning_mem`	
DR リージョンで SAP のアプリケーションサーバーを起動します。	本番システムで作成した AMI を使用して、DR リージョンに「新しい追加のアプリケーションサーバーを起動」します。	SAP ベーシス管理者
SAP アプリケーションを起動する前に検証を行います。	1. `/etc/hosts` および `/etc/fstab` のエントリを検証します。 2. DR システムに `/sapmnt/<SID>/` をマウントします。 3. DRファイルシステム `/sapmnt/<SID>/` が本番環境 `/sapmnt/<SID>/` と同期していることを確認します。 4. `<sid>adm` ユーザーにログインして `R3trans -d` を実行し、`trans.log` ファイルの出力を確認します。`trans.log` ファイルが `R3trans -d` コマンドを実行した同じ場所に生成されます。	AWS 管理者、SAP ベーシス管理者

タスク	説明	必要なスキル

DR システムで SAP アプリケーションを起動します。	<sid>adm ユーザーを使用して、 DR システムで SAP アプリケーションを起動します。次のコードを使用したら、XX が SAP ABAP SAP セントラルサービス (ASCS)を表し、YY がSAPアプリケーションサーバーのインスタンス数を表します。
	`sapconrol -nr XX -function StartService <SID>`
	`sapconrol -nr XX -function StartSystem`
	`sapconrol -nr YY -function StartService <SID>`
	`sapconrol -nr YY -function StartSystem`
		SAP ベーシス管理者

SAP 検証を実行します。	DR テストとして実施され、エビデンスを提供したり、DR 領域へのデータ複製の成功を確認したりします。	テストエンジニア

DR フェイルバックタスクの実行

タスク	説明	必要なスキル
本番環境の SAP サーバーとデータベースサーバーを起動します。	コンソールで、SAP と本番システムのデータベースをホストするEC2インスタンスを起動します。	SAP ベーシス管理者

タスク	説明	必要なスキル

本番データベースを起動し、HADR を設定します。	本番システム (host1) にログインし、以下のコマンドを使用して DB がリカバリモードになっていることを確認します。
	`db2start`
	`db2 start HADR on db P3V as standby`
	`db2 connect to <SID>`
	HADRの状態が connected であることを確認します。レプリケーションステータスが peer であるはずです。
	`db2pd -d <SID> -hadr`
	データベースに不一致がなく、connected と peer のステータスではない場合、現在アクティブなデータベース (DR リージョンの host2)とデータベースを同期(host1 で)するために、バックアップと復元が必要になることがあります。その場合、DB バックアップを host2 DR のリージョンのデータベースから host1 本番リージョンのデータベースに復元します。
		SAP ベーシス管理者

タスク	説明	必要なスキル
データベースを本番リージョンにフェイルバックします。	通常の業務シナリオでは、このステップがスケジュール済みのダウンタイムに実行されます。DR システムで実行されているアプリケーションが停止され、データベースが本番リージョン (リージョン 1) にフェイルバックされ、本番リージョンからのオペレーションが再開されます。 1. DR リージョンの SAP アプリケーションサーバーに口グインし、SAP アプリケーションを停止します。 2. DRシステムから `/sapmnt/<SID>` をアンマウントし、本番システムの `/sapmnt/<SID>` に変更がリバースレプリケートされることを確認します。 3. 本番リージョンのデータベースサーバー (`host1`) に口グインし、テイクオーバーを実行します。 `db2 takeover hadr on database <SID>` 4. HADR のステータスを確認します。`HADR_ROLE` が `host1` の `PRIMARY` で、または `host2` の `StandBy` にあるはずです。	SAP ベーシス管理者

タスク	説明	必要なスキル
	`db2pd -d <SID> -hadr`	
SAP アプリケーションを起動する前に検証を行います。	1. /etc/hosts および /etc/fstab のエントリを検証します。 2. 本番システムに /sapmnt/<SID>/ をマウントします。 3. DR システム /sapmnt/<SID>/ と同期していることを確認します。 4. <sid>adm ユーザーにログインして R3trans -d を実行し、trans.log ファイルの出力を確認します。trans.log ファイルが R3trans -d コマンドを実行した同じ場所に生成されます。	AWS 管理者、SAP ベーシス管理者

タスク	説明	必要なスキル

SAP アプリケーションを起動します。	1. <sid>adm ユーザーを使用して、本番システムで SAP アプリケーションを起動します。次のコードを使用します。その内、XX が SAP ASCS サーバーのインスタンス数を表し、YY がSAP アプリケーションサーバーのインスタンス数を表します。
	`sapconrol -nr XX -function StartService <SID>`
	`sapconrol -nr XX -function StartSystem`
	`sapconrol -nr YY -function StartService <SID>`
	`sapconrol -nr YY -function StartSystem`
	2. アプリケーションサーバーが使用可能であることを確認するには、SAP にログインし、SICK と SM51 のトランザクションを使用してチェックを行います。
		SAP ベーシス管理者

トラブルシューティング

問題	ソリューション
HADR 関連の問題をトラブルシューティングするためのキーログファイルとコマンド	• `db2 get db cfg | grep -i hadr` • `db2pd -d sid -hadr`

問題	ソリューション
	• Db2diag.log (このファイルは通常 db2dump ディレクトリにあり、db2dump パスがパラメータ DIAGPATH によって定義されます)。
Db2 UDB の HADR 問題のトラブルシューティングに関する SAP ノート	「SAP ノート 1154013-DB6:HADR 環境のDBの問題」 を参照してください。(このノートにアクセスするには SAP ポータルの認証情報が必要です)。

関連リソース

- 「AWS 上の Db2 データベースのディザスタリカバリアプローチ」 (ブログ記事)
- 「SAP on AWS — ペースメーカー搭載の IBM Db2 HADR」
- 「DB2 データベース間で HADR レプリケーションをセットアップするステップバイステップの手順」
- 「DB2 HADR ウィキ」

追加情報

このパターンを使用して、Db2 データベースで稼働している SAP システムのディザスタリカバリシステムを設定できます。ディザスタの時、定義された目標復旧時間 (RTO) と目標復旧時点 (RPO) の要件の範囲でビジネスを継続できる必要があります:

- RTO とは、サービスが中断してから復旧するまでに経過した時間の、許容される最大値のことです。これにより、サービスが使用不可のときに許容される時間枠が決まります。
- RPO とは、データが最後に復旧した時点を起点とする経過時間の、許容される最大値のことです。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

HADR に関するよくある質問については、「SAP ノート #1612105-DB6: Db2 高可用性ディザスタリカバリ (HADR) に関する FAQ」 を参照してください。(このノートにアクセスするには SAP ポータルの認証情報が必要です)。

Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Aarti Rajput (AWS)、Ashish Bhatt (AWS)、Aniket Dekate (AWS)、Naveen Suthar (AWS)、Nadeem Rahaman (AWS)、R"ka Modi (AWS)、Tamilselvan P (AWS)

概要

このパターンは、データベース移行を信頼性が高く自動化された方法で管理するための継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを確立するためのものです。Infrastructure as Code (IaC) ツールである Terraform を使用して、必要なインフラストラクチャのプロビジョニング、データの移行、スキーマの変更のカスタマイズを行うプロセスについて説明します。

具体的には、このパターンは、オンプレミスの Microsoft SQL Server データベースを Amazon Relational Database Service (Amazon RDS) に移行する CI/CD パイプラインを設定します AWS。このパターンを使用して、仮想マシン (VM) または別のクラウド環境にある SQL Server データベースを Amazon RDS に移行することもできます。

このパターンは、データベースの管理とデプロイに関連する以下の課題に対処します。

- 手動データベースデプロイは、時間がかかり、エラーが発生しやすく、環境間の一貫性がありません。
- インフラストラクチャのプロビジョニング、データ移行、スキーマの変更を調整することは複雑で管理が難しい場合があります。
- 本稼働システムでは、データの整合性を確保し、データベースの更新中のダウンタイムを最小限に抑えることが重要です。

このパターンには以下の利点があります。

- データベース移行用の CI/CD パイプラインを実装することで、データベースの変更を更新およびデプロイするプロセスを合理化します。これにより、エラーのリスクが軽減され、環境間の一貫性が確保され、ダウンタイムが最小限に抑えられます。
- 信頼性、効率、コラボレーションの向上に役立ちます。データベースの更新中の市場投入までの時間を短縮し、ダウンタイムを短縮します。
- データベース管理に最新の DevOps プラクティスを採用するのに役立ちます。これにより、ソフトウェア配信プロセスの俊敏性、信頼性、効率が向上します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ローカルマシンにインストールされた Terraform 0.12 以降 (手順については、Terraform ドキュメントを参照してください)
- HashiCorp の Terraform AWS Provider バージョン 3.0.0 以降 (このプロバイダーの GitHub リポジトリを参照)
- 最小特権 AWS Identity and Access Management (IAM) ポリシー (ブログ記事「最小特権 IAM ポリシーを記述するためのテクニック」を参照)

アーキテクチャ

このパターンでは、データベース移行プロセスの完全なインフラストラクチャを提供する次のアーキテクチャを実装します。

このアーキテクチャでは、以下のようになります。

- ソースデータベースは、オンプレミス、仮想マシン (VM)、または別のクラウドプロバイダーによってホストされている SQL Server データベースです。この図は、ソースデータベースがオンプレミスデータセンターにあることを前提としています。
- オンプレミスのデータセンターと AWS は、VPN または AWS Direct Connect 接続を介して接続されます。これにより、ソースデータベースと AWS インフラストラクチャ間の安全な通信が可能になります。
- ターゲットデータベースは、データベースプロビジョニングパイプラインを使用して の仮想プライベートクラウド (VPC) AWS 内でホストされる Amazon RDS データベースです。
- AWS Database Migration Service （ AWS DMS) は、オンプレミスデータベースを にレプリケートします AWS。これは、ソースデータベースのターゲットデータベースへのレプリケーションを設定するために使用されます。

次の図は、プロビジョニング、セットアップ、 AWS DMS 検証を含む、さまざまなレベルのデータベース移行プロセスで設定されたインフラストラクチャを示しています。

このプロセスでは、次の操作を行います。

- 検証パイプラインはすべてのチェックを検証します。必要なすべての検証が完了すると、統合パイプラインは次のステップに進みます。
- DB プロビジョニングパイプラインは、データベースに提供された Terraform コードに対して Terraform アクションを実行するさまざまな AWS CodeBuild ステージで構成されます。これらのステップが完了すると、ターゲットにリソースがデプロイされます AWS アカウント。
- AWS DMS パイプラインは、テストを実行し、IaC を使用して移行を実行するための AWS DMS インフラストラクチャをプロビジョニングするさまざまな CodeBuild ステージで構成されます。

ツール

AWS のサービス および ツール

- AWS CodeBuild は、ソースコードをコンパイルし、テストを実行し、ready-to-deployソフトウェアパッケージを生成するフルマネージドの継続的インテグレーションサービスです。
- AWS CodePipeline は、完全マネージド型の継続的デリバリーサービスであり、リリースパイプラインを自動化して、アプリケーションとインフラストラクチャを迅速かつ確実に更新するのに役立ちます。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。
- Amazon Simple Storage Service (Amazon S3) は、スケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。
- AWS Database Migration Service （ AWS DMS ） は、データストアを に移行する AWS クラウドか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役立ちます。

その他のサービス

- Terraform は HashiCorp の IaC ツールで、クラウドおよびオンプレミスのリソースの作成と管理に役立ちます。

コードリポジトリ

このパターンのコードは、GitHub Database Migration DevOps Framework using Terraform サンプルリポジトリで入手できます。

ベストプラクティス

- データベース移行の自動テストを実装して、スキーマの変更とデータの整合性が正しいことを確認します。これには、ユニットテスト、統合テスト、end-to-endテストが含まれます。
- 特に移行前に、データベースの堅牢なバックアップおよび復元戦略を実装します。これにより、データの整合性が確保され、障害が発生した場合にフォールバックオプションが提供されます。
- 移行中に障害や問題が発生した場合にデータベースの変更を元に戻すための堅牢なロールバック戦略を実装します。これには、以前のデータベース状態にロールバックしたり、個々の移行スクリプトを元に戻したりすることが含まれます。
- データベース移行の進行状況とステータスを追跡するためのモニタリングとログ記録のメカニズムを設定します。これにより、問題をすばやく特定して解決できます。

エピック

ローカルワークステーションをセットアップする

タスク	説明	必要なスキル
ローカルワークステーションで Git をセットアップおよび設定します。	Git ドキュメントの手順に従って、ローカルワークステーションに Git をインストールして設定します。	DevOps エンジニア
プロジェクトフォルダを作成し、GitHub リポジトリからファイルを追加します。	1. このパターンの GitHub リポジトリを開きます。 2. コードを選択してクローンオプションを表示し、HTTPS タブで提供されている URL をコピーします。 3. ワークステーションにプロジェクトのフォルダを作成します。 4. ターミナルを開き、このフォルダに移動します。	DevOps エンジニア

タスク	説明	必要なスキル

	5. GitHub リポジトリのクローンを作成します。
	`git clone <github-repository-url>`
	ここで、 `<github-repository-url>` はステップ 2 でコピーした URL です。
	6. クローン作成が完了したら、プロジェクトフォルダのクローンされたリポジトリに移動します。
	`cd <folder-name>/aws-terraform-db-migration-framework-samples`
	7. このプロジェクトを任意の統合開発環境 (IDE) で開きます。	

ターゲットアーキテクチャのプロビジョニング

タスク	説明	必要なスキル
必要なパラメータを更新します。	`ssm-parameters.sh` ファイルには、必要な AWS Systems Manager パラメータがすべて保存されます。これらのパラメータは、プロジェクトのカスタム値で設定できます。	DevOps エンジニア

タスク	説明	必要なスキル
	ローカルワークステーションの setup/db-ssm-params フォルダで、ファイルを開きssm-parameters.sh 、CI/CD パイプラインを実行する前にこれらのパラメータを設定します。	
Terraform 構成を初期化します。	db-cicd-integration フォルダに次のコマンドを入力して、Terraform 設定ファイルを含む作業ディレクトリを初期化します。 `terraform init`	DevOps エンジニア
Terraform プランをプレビューしてください。	Terraform プランを作成するには、次のコマンドを入力します。 `terraform plan -var-file="terraform.sample"` Terraform は設定ファイルを評価して、宣言されたリソースのターゲット状態を判断します。次に、ターゲットの状態を現在の状態と比較し、プランを作成します。	DevOps エンジニア
プランを検証してください。	計画を確認し、ターゲットに必要なアーキテクチャが設定されていることを確認しますAWS アカウント。	DevOps エンジニア

タスク	説明	必要なスキル
ソリューションをデプロイします。	1. プランを適用するには、次のコマンドを入力します。 `terraform apply -var-file="terraform.sample"` 2. yes を入力して確定します。Terraform は、構成ファイルに宣言されている目標状態を達成するために、インフラストラクチャを作成、更新、または破棄します。シーケンスの詳細については、このパターンの「アーキテクチャ」セクションを参照してください。	DevOps エンジニア

デプロイメントを確認する

タスク	説明	必要なスキル
デプロイを検証します。	`db-cicd-integration` パイプラインのステータスを確認して、データベースの移行が完了したことを確認します。 1. にサインインし AWS Management Console、AWS CodePipeline コンソールを開きます。	DevOps エンジニア

タスク	説明	必要なスキル
	2. ナビゲーションペインで [Pipelines] (パイプライン) を選択します。 3. パイプラインを選択しますdb-cicd-integration 。 4. パイプラインの実行が正常に完了したことを確認します。	

使用後のインフラストラクチャのクリーンアップ

タスク	説明	必要なスキル
インフラストラクチャをクリーンアップします。	1. プロジェクトが完了したら、 コマンドを使用して作成したインフラストラクチャをクリーンアップします。 `terraform destroy --var-file=terrafo rm.sample` 2. yes を入力して確定します。	DevOps エンジニア

関連リソース

AWS ドキュメント

- Terraform 製品の開始方法

Terraformのドキュメント

- Terraform のインストール
- Terraform バックエンド設定
- Terraform AWS プロバイダーのドキュメント

Amazon RDS Custom でアクティブスタンバイデータベースを使用して Oracle E-Business Suite の HA/DR アーキテクチャを設定する

作成者: Simo Cunningham (AWS)、Nitin Saxena

概要

このパターンでは、Amazon Relational Database Service (Amazon RDS) Custom で Amazon RDS Custom リードレプリカデータベースを別の Amazon Web Services (AWS) アベイラビリティーゾーンに設定し、それをアクティブなスタンバイデータベースに変換することで、高可用性 (HA) とディザスタリカバリ (DR) を実現する Oracle E-Business ソリューションを構築する方法について説明します。Amazon RDS Custom リードレプリカの作成は、AWS マネジメントコンソールを通じて完全に自動化されています。

このパターンでは、HA/DR アーキテクチャの一部にもなり得るアプリケーション層や共有ファイルシステムを追加する手順については説明していません。これらのトピックの詳細については、Oracle サポートに関する注意事項の 1375769.1、1375670.1、および 1383621.1 (セクション 5、「クローニングの詳細オプション」) を参照してください。(アクセスには Oracle サポートアカウントが必要です。)

E-Business Suite システムを Amazon Web Services (AWS) の単層、シングル AZ アーキテクチャに移行するには、「Oracle E-Business Suite を Amazon RDS Custom に移行」というパターンを参照してください。

Oracle E-Business Suite は、財務、人事、サプライチェーン、製造などの企業全体のプロセスを自動化するためのエンタープライズリソースプランニング (ERP) ソリューションです。クライアント、アプリケーション、データベースという 3 層のアーキテクチャで構成されています。以前は、E-Business Suite データベースを自己管理型の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行する必要がありましたが、Amazon RDS Custom の恩恵を受けることができるようになりました。

前提条件と制限

前提条件

- Amazon RDS Custom にインストールされている既存の E-Business Suite。「Oracle E-Business Suite を Amazon RDS Customに移行」のパターンを参照してください。

- リードレプリカを読み取り専用に変更し、それを使用してレポートをスタンバイにオフロードする場合は、「Oracle Active Data Guardデータベースライセンス」（Oracle Technology の商用価格表を参照）が必要です。

制約事項

- Amazon RDS Custom 上の Oracle データベースに関する制限とサポートされていない構成
- Amazon RDS Custom for Oracle リードレプリカに関連する制限

製品バージョン

Amazon RDS Custom がサポートする Oracle Database のバージョンとインスタンスクラスについては、「Amazon RDS Custom for Oracle の要件と制限」を参照してください。

アーキテクチャ

次の図は、アクティブ/パッシブのセットアップに複数のアベイラビリティーゾーンとアプリケーション層を含む E-Business Suite on AWS の代表的なアーキテクチャを示しています。データベースは Amazon RDS Custom DB インスタンスと Amazon RDS Custom リードレプリカを使用します。リードレプリカは Active Data Guard を使用して、別のアベイラビリティーゾーンに複製します。リードレプリカを使用して、プライマリデータベースの読み取りトラフィックをオフロードし、またはレポートを作成することもできます。

詳細については、Amazon RDS ドキュメントの「Amazon RDS Custom for Oracle 用リードレプリカの使用」を参照してください。

Amazon RDS Custom リードレプリカは、デフォルトではマウント状態で作成されます。ただし、読み取り専用のワークロードの一部をスタンバイデータベースにオフロードしてプライマリデータベースの負荷を軽減したい場合は、「エピック」セクションの手順に従って、マウントされたレプリカのモードを手動で読み取り専用に変更できます。一般的な使用例は、スタンバイデータベースからレポートを実行することです。読み取り専用に変更するには、アクティブ/スタンバイデータベースライセンスが必要です。

AWS でリードレプリカを作成すると、システムは Oracle Data Guard ブローカーを内部で使用します。 この設定は、以下のように自動的に生成され、最大パフォーマンスモードで設定されます。

```
DGMGRL> show configuration
```

```
Configuration - rds_dg
  Protection Mode: MaxPerformance
  Members:
  vis_a - Primary database
    vis_b - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS   (status updated 58 seconds ago)
```

ツール

AWS サービス

- 「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。データベース管理のタスクとオペレーションを自動化し、データベース管理者としてデータベース環境とオペレーティングシステムへのアクセスおよびカスタマイズを可能にします。

その他のツール

- Oracle Active Data Guard は、スタンバイデータベースの作成と管理に役立ちます。このパターンでは、Oracle Data Guard を使用して Amazon RDS Custom 上にアクティブ スタンバイデータベースを設定します。

エピック

リードレプリカの作成

タスク	説明	必要なスキル
Amazon RDS Custom DB インスタンスのリードレプリカを作成します。	リードレプリカを作成するには、「Amazon RDS ドキュメント」の指示に従い、作成した Amazon RDS Custom DB インスタンス (「前提条件」セクションを参照) をソースデータベースとして使用します。	DBA

タスク	説明	必要なスキル
	デフォルトでは、Amazon RDS Custom リードレプリカは物理スタンバイとして作成され、マウントされた状態になります。これにより、Oracle Active Data Guard ライセンスへのコンプライアンスが確保されます。リードレプリカを読み取り専用モードに変換するには、次の手順に従います。	

リードレプリカを読み取り専用のアクティブスタンバイに変更します。

タスク	説明	必要なスキル
Amazon RDS Custom リードレプリカに接続します。	以下のコマンドを使用して、物理スタンバイデータベースをアクティブスタンバイデータベースに変換します。 **Important** これらのコマンドには、Oracle アクティブスタンバイライセンスが必要です。ライセンスを取得するには、Oracle の担当者にお問い合わせください。 `$ sudo su - rdsdb`	DBA

タスク	説明	必要なスキル
	-bash-4.2$ sql SQL> select process,s tatus,sequence# from v $managed_standby; PROCESS STATUS SEQUENCE# --------- ------------ ---------- ARCH CLOSING 3956 ARCH CONNECTED 0 ARCH CLOSING 3955 ARCH CLOSING 3957 RFS IDLE 0 RFS IDLE 3958 MRP0 APPLYING_LOG 3958 SQL> select name, database_role, open_mode from v $database; NAME DATABASE_ ROLE OPEN_MODE --------- --------- ------- --------- ----------- VIS PHYSICAL STANDBY MOUNTED SQL> alter database recover managed standby database cancel; Database altered. Open the standby database	

タスク	説明	必要なスキル

	`SQL> alter database open;`
	`Database altered.`
	`SQL> select name, database_role, open_mode from v$database;`
	
	`NAME DATABASE_ROLE OPEN_MODE`
	`--------- ---------- ----------`
	`VIS PHYSICAL STANDBY READ ONLY`	

リアルタイムログを適用することで、メディアリカバリを開始します。	リアルタイムログ適用機能を有効にするには、以下のコマンドを使用します。これらのコマンドはスタンバイ (リードレプリカ) をアクティブスタンバイデータベースとして変換して検証するため、接続して読み取り専用クエリを実行できます。
	
	`SQL> alter database recover managed standby database using current logfile disconnect from session;`
	`Database altered`
		DBA

タスク	説明	必要なスキル

データベースのステータスをチェックします。	データベースのステータスを確認するには、次のコマンドを使用します。
	
	`SQL> select name, database_role, open_mode from v$database;`
	`NAME DATABASE_ROLE OPEN_MODE`
	`--------- ---------- ---------- ----------`
	`VIS PHYSICAL STANDBY READ ONLY WITH APPLY`
		DBA

タスク	説明	必要なスキル

REDO 適用モードをチェックします。	REDO 適用モードを確認するには、次のコマンドを使用します。
	
	`SQL> select process,status,sequence# from v$managed_standby;`
	`PROCESS STATUS SEQUENCE#`
	`--------- ------------ ----------`
	`ARCH CLOSING 3956`
	`ARCH CONNECTED 0`
	`ARCH CLOSING 3955`
	`ARCH CLOSING 3957`
	`RFS IDLE 0`
	`RFS IDLE 3958`
	`MRP0 APPLYING_LOG 3958`
	
	`SQL> select open_mode from v$database;`
	`OPEN_MODE`
	`--------------------`
	`READ ONLY WITH APPLY`
		DBA

関連リソース

- 「Oracle E-Business Suite を Amazon RDS Customに移行」 (AWS 規範ガイダンス)
- 「「Amazon RDS Customでの作業」 (Amazon RDS ドキュメント)
- 「Amazon RDS Custom for Oracle リードレプリカの使用」 (Amazon RDS ドキュメント)

- 「Amazon RDS Custom for Oracle — データベース環境における新しい制御機能」(AWS ニュース ブログ)
- 「Oracle E-Business Suite on AWS への移行」(AWS ホワイトペーパー)
- 「AWS での Oracle E-Business Suite アーキテクチャ」(AWS ホワイトペーパー)

GTID を使用して Amazon RDS for MySQL と Amazon EC2 上の MySQL との間のデータレプリケーションを設定する

作成者: Rajesh Madiwale (AWS)

概要

このパターンでは、MySQL ネイティブグローバルトランザクション識別子 (GTID) レプリケーションを使用して、MySQL DB インスタンス用 Amazon Relational Database Service (Amazon RDS) と Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の MySQL データベースとの間で、Amazon Web Services (AWS) クラウド上のデータレプリケーションを設定する方法を示しています。

GTID を使用すると、トランザクションは、オリジンサーバーでコミットされ、レプリカにより適用されるときに識別および追跡されます。フェイルオーバー中に新しいレプリカを起動する際、ログファイルを参照する必要はありません。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- デプロイされた Amazon Linux インスタンス

制限事項

- この設定では、内部チームが読み取り専用クエリを実行する必要があります。
- ソースの MySQL バージョンとターゲットの MySQL バージョンが同じでなければなりません。
- レプリケーションは、同じ AWS リージョンと仮想プライベートクラウド (VPC) 内に設定されます。

製品バージョン

- Amazon RDS バージョン 5.7.23 以降 (GTID をサポートするバージョン)

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for MySQL

ターゲットテクノロジースタック

- Amazon EC2 上の MySQL

ターゲット アーキテクチャ

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Relational Database Service (Amazon RDS) for MySQL を使用して、AWS クラウドでリレーショナルデータベースをセットアップ、運用、スケーリングできます。

その他のサービス

- グローバルトランザクション ID (GTID) はコミットされた MySQL トランザクションに対して生成される一意の ID です。
- mysqldump は、ソースデータベースのオブジェクト定義とテーブルデータを再現するために実行できる SQL ステートメントを生成することで、論理バックアップを実行するクライアントユーティリティです。
- mysql は MySQL のコマンドラインクライアントです。

エピック

Amazon RDS for MySQL DB インスタンスを作成して準備する

タスク	説明	必要なスキル
RDS for MySQL インスタンスを作成する。	RDS for MySQL インスタンスを作成するには、次のタスク	DBA、DevOps エンジニア

タスク	説明	必要なスキル
	で説明するパラメータ値を使用して、Amazon RDS ドキュメントの手順に従います。	
DB パラメータグループで GTID 関連の設定を有効にする。	Amazon RDS for MySQL DB パラメータグループで、次のパラメータを有効にします。 `enforce_gtid_consistency` を on に、`gtid-mode` を on に設定する。	DBA
Amazon RDS for MySQL インスタンスを再起動する。	パラメータの変更を反映するには再起動が必要です。	DBA

ユーザーを作成し、レプリケーション権を付与する。	次のコマンドを使用して をインストールします。
	
	`CREATE USER 'repl'@'%' IDENTIFIED BY 'xxxx';`
	`GRANT REPLICATI`
	`ON slave ON *.* TO 'repl'@'%' ;`
	`FLUSH PRIVILEGES;`
		DBA

Amazon EC2 インスタンスに MySQL をインストールして準備する

タスク	説明	必要なスキル
Amazon Linux に LAMP をインストールする。	次のコマンドを使用して をインストールします。 `sudo yum update`	DBA

タスク	説明	必要なスキル

	`sudo wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm`
	`sudo yum localinstall mysql57-community-release-el7-11.noarch.rpm`
	`sudo yum install mysql-community-server`
	`sudo systemctl start mysqld`	

EC2 インスタンスで MySQL にログインし、データベースを作成する。	データベース名は、Amazon RDS for MySQL のデータベース名と同じである必要があります。次の例では、ユーザー名は `replication` です。 `create database replication;`	DBA

タスク	説明	必要なスキル

MySQL 構成ファイルを編集し、データベースを再起動する。	以下のパラメータを入力して、/etc/ にある my.conf ファイルを編集します。
	`server-id=3`
	`gtid_mode=ON`
	`enforce_gtid_consistency=ON`
	`replicate-ignore-db=mysql`
	`binlog-format=ROW`
	`log_bin=mysql-bin`
	その後、mysqld サービスを再起動します。
	`systemctl mysqld restart`
		DBA

レプリケーションの設定

タスク	説明	必要なスキル
Amazon RDS for MySQL データベースからデータダンプをエクスポートする。	Amazon RDS for MySQL からダンプをエクスポートするには、次のコマンドを使用します。 `mysqldump --single-transaction -h mydb.xxxxxxx.amazonaws.com -uadmin -p --databases replication > replication-db.sql`	DBA

タスク	説明	必要なスキル
Amazon EC2 の MySQL データベースにある.sql ダンプファイルを復元する。	Amazon EC2 の MySQL データベースにダンプをインポートするには、次のコマンドを使用します。 `mysql -D replication -uroot -p < replication-db.sql`	DBA

Amazon EC2 の MySQL データベースをレプリカとして構成する。	レプリケーションを開始してレプリケーションステータスを確認するには、Amazon EC2 の MySQL データベースにログインし、次のコマンドを使用します。
	`CHANGE MASTER TO MASTER_HOST="mydb.xxxxxxx.amazonaws.com", MASTER_USER="repl", MASTER_PASSWORD="rep123", MASTER_PORT=3306, MASTER_AUTO_POSITION = 1;`
	`START SLAVE;`
	`SHOW SLAVE STATUS\G`
		DBA

関連リソース

- Linux インスタンス用 Amazon EC2 ユーザーガイド
- MySQL Yum リポジトリを使用して MySQL を Linux にインストールする
- グローバルトランザクション識別子を使ったレプリケーション
- Amazon RDS for MySQL の GTID ベースレプリケーションを使用する

Amazon RDS Custom for Oracle 上の Oracle PeopleSoft アプリケーションの移行ロール

作成者: sampath kathirvel (AWS)

概要

Amazon Web Services (AWS) で Oracle PeopleSoft エンタープライズリソースプランニング (ERP) ソリューションを実行するには、Amazon Relational Database Service (Amazon RDS) または Amazon RDS Custom for Oracle を使用できます。これらは、基盤となるオペレーティングシステム (OS) とデータベース環境へのアクセスを必要とするレガシー、カスタム、およびパッケージ化されたアプリケーションをサポートします。移行を計画する際に考慮すべき主な要素については、「AWS 規範ガイダンス」 の「Oracle データベースの移行戦略」を参照してください。

このパターンは、読み込みレプリカデータベースを備えたプライマリデータベースとして Amazon RDS Custom で実行されている PeopleSoft アプリケーションデータベースの Oracle Data Guard スイッチオーバーまたはロール移行を実行するステップに焦点を当てています。このパターンには、ファストスタートフェイルオーバー (FSFO) を構成するステップが含まれています。このプロセス中、Oracle Data Guard 構成内のデータベースは引き続き新しい役割で機能します。Oracle Data Guard のスイッチオーバーの一般的なユースケースとしては、ディザスタリカバリ (DR) ドリル、データベースの定期メンテナンスアクティビティ、スタンバイファーストパッチ適用ローリングパッチなどがあります。詳細については、ブログ記事「Amazon RDS Custom のデータベースパッチのダウンタイムを削減する」を参照してください。

前提条件と制限

前提条件

- 読み込みレプリカパターンを使用して Amazon RDS Custom でOracle PeopleSoft に HA を追加する」パターンを完了する。

機能制限

- RDS Custom for Oracle で制限されている構成およびサポートされていない構成
- Amazon RDS Custom for Oracle リードレプリカに関連する制限

製品バージョン

- Amazon RDS Custom でサポートされている Oracle データベースのバージョンについては、「RDS Custom for Oracle」を参照してください。
- Amazon RDS Customでサポートされている Oracle データベースインスタンスクラスについては、「RDS Custom for Oracle用 DB インスタンスクラスサポート」を参照してください。

アーキテクチャ

テクノロジースタック

- Amazon RDS Custom for Oracle

ターゲットアーキテクチャ

次の図は、Amazon RDS Custom DB インスタンスと Amazon RDS Custom 読み込みレプリカを示しています。Oracle Data Guard は、DR のフェイルオーバー中にロールを移行します。

AWS で Oracle PeopleSoft を使用する代表的なアーキテクチャについては、「AWS で可用性の高い PeopleSoft アーキテクチャの設定」を参照してください。

ツール

サービス

- Amazon RDS Custom for Oracle は、基盤となるOSとデータベース環境へのアクセスを必要とするレガシー、カスタム、およびパッケージアプリケーション向けのマネージドデータベースサービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。このパターンでは、シークレット名 `do-not-delete-rds-custom-+<<RDS Resource ID>>+-dg` を使用して Secrets Manager から `RDS_DATAGUARD` のデータベースユーザーパスワードを取得します。

その他のサービス

- Oracle Data Guard は、スタンバイ・データベースの作成、保守、管理、モニタリングに役立ちます。このパターンでは、ロールの移行に Oracle Data Guard Maximum Performance を使用します（Oracle Data Guard のスイッチオーバー）。

ベストプラクティス

本番環境へのデプロイでは、オブザーバーインスタンスをプライマリノードやリードレプリカノードとは別の 3 番目のアベイラビリティゾーンで起動することをお勧めします。

エピック

ロールの移行を開始する

タスク	説明	必要なスキル
プライマリとレプリカの両方のデータベースの自動化を一時停止する。	RDS Custom の自動化フレームワークはロール移行プロセスを妨げませんが、Oracle Data Guard のスイッチオーバー中は自動化を一時停止することをお勧めします。 RDS Custom データベースの自動化を一時停止して再開するには、「RDS Custom 自動化の一時停止と再開」の手順に従ってください。	クラウド管理者、DBA
Oracle Data Guard のステータスをチェックします。	Oracle Data Guard のステータスを確認するには、プライマリデータベースにログインします。このパターンには、マルチテナントコンテナデータベース (CDB) または 非 CDB インスタンスを使用するコードが含まれています。 非 CDB `-bash-4.2$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_A`	DBA

タスク	説明	必要なスキル
	DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 28 20:55:50 2022 Version 19.10.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "ORCL_A" Connected as SYSDG. DGMGRL> show configura tion Configuration - rds_dg Protection Mode: MaxAvailability Members: orcl_a - Primary database orcl_d - Physical standby database Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 59 seconds ago) DGMGRL> CDB CDB-bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_A DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 18 06:13:07 2023	

タスク	説明	必要なスキル
	Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "RDSCDB_A " Connected as SYSDG. DGMGRL> show configura tion Configuration - rds_dg Protection Mode: MaxAvailability Members: rdscdb_a - Primary database rdscdb_b - Physical standby database Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 52 seconds ago) DGMGRL>	

タスク	説明	必要なスキル
インスタンスロールを検証します。	AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。データベースの [レプリケーション] セクションの [接続とセキュリティ] タブで、プライマリとレプリカのインスタンスロールを確認します。 プライマリロールは Oracle Data Guard プライマリデータベースと一致し、レプリカロールは Oracle Data Guard のフィジカルスタンバイデータベースと一致する必要があります。	クラウド管理者、DBA

タスク	説明	必要なスキル

スイッチオーバーを実行します。	スイッチオーバーを実行するには、プライマリノードから DGMGRL に接続します。
	
	非 CDB
	
	`DGMGRL> switchover to orcl_d;`
	`Performing switchover NOW, please wait...`
	`Operation requires a connection to database "orcl_d"`
	`Connecting ...`
	`Connected to "ORCL_D"`
	`Connected as SYSDG.`
	`New primary database "orcl_d" is opening...`
	`Operation requires start up of instance "ORCL" on database "orcl_a"`
	`Starting instance "ORCL"...`
	`Connected to an idle instance.`
	`ORACLE instance started.`
	`Connected to "ORCL_A"`
	`Database mounted.`
	`Database opened.`
	`Connected to "ORCL_A"`
	`Switchover succeeded, new primary is "orcl_d"`
	`DGMGRL>`
	
	CDB
	
	`DGMGRL> switchover to rdscdb_b`
		DBA

タスク	説明	必要なスキル
	Performing switchover NOW, please wait... New primary database "rdscdb_b" is opening... Operation requires start up of instance "RDSCDB" on database "rdscdb_a" Starting instance "RDSCDB"... Connected to an idle instance. ORACLE instance started. Connected to "RDSCDB_A" Database mounted. Database opened. Connected to "RDSCDB_A" Switchover succeeded, new primary is "rdscdb_b"	

タスク	説明	必要なスキル

Oracle Data Guard の接続を検証します。	スイッチオーバー後、プライマリノードから DGMGRL への Oracle Data Guard の接続を確認します。
	
	非 CDB
	
	`DGMGRL> show configura tion;`
	`Configuration - rds_dg`
	`Protection Mode:`
	` MaxAvailability`
	`Members:`
	`orcl_d - Primary`
	` database`
	`orcl_a - Physical`
	` standby database`
	`Fast-Start Failover:`
	` Disabled`
	`Configuration Status:`
	`SUCCESS (status updated`
	` 60 seconds ago)`
	`DGMGRL>`
	
	`DGMGRL> show configura tion lag;`
	`Configuration - rds_dg`
	`Protection Mode:`
	` MaxAvailability`
	`Members:`
	`orcl_d - Primary`
	` database`
	`orcl_a - Physical`
	` standby database`
	`Transport Lag: 0`
	` seconds (computed 0`
	` seconds ago)`
	`Apply Lag: 0 seconds`
	` (computed 0 seconds`
	` ago)`
		DBA

タスク	説明	必要なスキル
	Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 44 seconds ago) DGMGRL>	

CDB

```
DGMGRL> show configura
tion
DGMGRL> show configura
tion
Configuration - rds_dg
  Protection Mode:
 MaxAvailability
  Members:
  rdscdb_b - Primary
 database
    rdscdb_a - Physical
 standby database
Fast-Start Failover:
 Disabled
Configuration Status:
SUCCESS   (status
 updated 52 seconds ago)
DGMGRL>

DGMGRL> show configura
tion lag
Configuration - rds_dg
  Protection Mode:
 MaxAvailability
  Members:
  rdscdb_b - Primary
 database
    rdscdb_a - Physical
 standby database
              Transport
 Lag:      0 seconds
```

タスク	説明	必要なスキル
	(computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 53 seconds ago) DGMGRL>	
Amazon RDS コンソールでインスタンスロールを検証します。	ロールの切り替えを実行すると、Amazon RDS コンソールで、[データベース] の [接続とセキュリティ] タブの [レプリケーション] セクションに新しいロールが表示されます。[レプリケーションの状態] が「空」から [レプリケーション中] に更新されるまでに数分かかる場合があります。	DBA

FSFO を構成する

タスク	説明	必要なスキル
スイッチオーバーをリセットします。	スイッチオーバーをプライマリノードに再設定します。	DBA
オブザーバーをインストールして起動する。	オブザーバープロセスは DGMGRL クライアントコンポーネントで、通常はプライマリデータベースやスタンバイデータベースとは別のマ	DBA

タスク	説明	必要なスキル
	シンで動作します。オブザーバー用の ORACLE HOME インストールは、Oracle Client Administrator でもかまいませんし、Oracle Database エンタープライズエディションまたはパーソナルエディションのいずれかをインストールすることも可能です。ご使用のデータベースリリースに合わせたオブザーバーのインストールについては、「オブザーバーのインストールと起動」を参照してください。オブザーバープロセスのハイアベイラビリティー構成には、次の操作を実行します。 • オブザーバーを実行している EC2 インスタンスの EC2 インスタンス自動リカバリを有効にします。OS のスタートアップの一環として、オブザーバーのスタートアッププロセスを自動化する必要があります。 • EC2 インスタンスにオブザーバーをデプロイし、サイズ 1 の Amazon EC2 Auto Scaling グループを構成します。EC2 インスタンスに障害が発生した場合、Auto Scaling グループは自動的に別の EC2 インスタンスをスピンアップします。	

タスク	説明	必要なスキル
	Oracle 12c リリース 2 以降では、最大 3 つのオブザーバーをデプロイできます。1 つのオブザーバーがプライマリオブザーバーで、残りはバックアップオブザーバーです。プライマリオブザーバーに障害が発生すると、バックアップオブザーバーの 1 つがプライマリの役割を果たします。	

タスク	説明	必要なスキル

オブザーバーホストから DGMGRL に接続します。	オブザーバーホストには、プライマリデータベース接続とスタンバイデータベース接続用の `tnsnames.ora` エントリが構成されています。データ損失が FFastStartFailOverLagLimit 構成 (秒単位) の範囲内であれば、maximum performance 保護モードで FSFO を有効にできます。ただし、データ損失をゼロ (RPO=0) にするには maximum availability 保護モードを使用する必要があります。
	
	非 CDB
	
	`DGMGRL> show configuration;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`orcl_d - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 58 seconds ago)`
	`DGMGRL> show configuration lag`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
		DBA

タスク	説明	必要なスキル
	Members: orcl_a - Primary database orcl_d - Physical standby database Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 5 seconds ago) DGMGRL>	
	CDB -bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_A DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 18 06:55:09 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "RDSCDB_A " Connected as SYSDG. DGMGRL> show configura tion Configuration - rds_dg	

タスク	説明	必要なスキル

	``
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_a - Primary
	 database
	    rdscdb_b - Physical
	 standby database
	Fast-Start Failover:
	 Disabled
	Configuration Status:
	SUCCESS (status
	 updated 18 seconds ago)
	DGMGRL>
	``	

タスク	説明	必要なスキル

スタンバイデータベースをフェイルオーバーターゲットに変更します。	プライマリノードまたはオブザーバーノードから 1 つのスタンバイデータベースに接続します。(構成には複数のスタンバイデータベースがあってもかまいませんが、この時点で接続する必要があるのは 1 つだけです)。
	
	非 CDB
	
	`DGMGRL> edit database orcl_a set property FastStartFailoverTarget='orcl_d';`
	`Property "faststartfailovertarget" updated`
	`DGMGRL> edit database orcl_d set property FastStartFailoverTarget='orcl_a';`
	`Property "faststartfailovertarget" updated`
	`DGMGRL> show database orcl_a FastStartFailoverTarget;`
	`FastStartFailoverTarget = 'orcl_d'`
	`DGMGRL> show database orcl_d FastStartFailoverTarget;`
	`FastStartFailoverTarget = 'orcl_a'`
	`DGMGRL>`
	
	CDB
		DBA

タスク	説明	必要なスキル

	DGMGRL> edit database
	 orcl_a set property
	 FastStartFailoverT
	arget='rdscdb_b';
	Object "orcl_a" was not
	 found
	DGMGRL> edit database
	 rdscdb_a set property
	 FastStartFailoverT
	arget='rdscdb_b';
	Property "faststar
	tfailovertarget"
	 updated
	DGMGRL> edit database
	 rdscdb_b set property
	 FastStartFailoverT
	arget='rdscdb_a';
	Property "faststar
	tfailovertarget"
	 updated
	DGMGRL> show database
	 rdscdb_a FastStart
	FailoverTarget;
	  FastStartFailoverT
	arget = 'rdscdb_b'
	DGMGRL> show database
	 rdscdb_b FastStart
	FailoverTarget;
	  FastStartFailoverT
	arget = 'rdscdb_a'
	DGMGRL>	

タスク	説明	必要なスキル
DGMGRL への接続にファストスタートフェイルオーバーのしきい値を構成します。	Oracle 19c のデフォルト値は 30 秒で、最小値は 6 秒です。値を小さくすると、フェイルオーバー時の目標復旧時間 (RTO) が短くなる可能性があります。値を高くすると、プライマリデータベースで不必要なフェイルオーバーの一時エラーが発生する可能性が低減されます。 RDS Custom for Oracle 自動化フレームワークは、データベースの状態をモニタリングし、数秒ごとに修正アクションを実行します。そのため、FastStartFailoverThreshold は 10 秒より大きい値に設定することをお勧めします。次の例では、しきい値を 35 秒に構成しています。 非 CBD または CDB `DGMGRL> edit configuration set property FastStartFailoverThreshold=35; Property "faststartfailoverthreshold" updated DGMGRL> show configuration FastStart FailoverThreshold; FastStartFailover Threshold = '35'`	DBA

タスク	説明	必要なスキル
	DGMGRL>	

タスク	説明	必要なスキル

プライマリノードまたはオブザーバーノードから DGMGRL に接続して FSFO を有効にします。	データベースでフラッシュバックデータベースが有効になっていない場合は、警告メッセージ ORA-16827 が表示されます。オプションのフラッシュバックデータベースは、FastStartFailoverAutoReinstate 構成プロパティが TRUE (デフォルト) に構成されている場合、障害が発生したプライマリデータベースをフェイルオーバー前のある時点まで自動的に復元するのに役立ちます。
	
	非 CDB
	
	`DGMGRL> enable fast_start failover;`
	`Warning: ORA-16827: Flashback Database is disabled`
	`Enabled in Zero Data Loss Mode.`
	`DGMGRL>`
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`Warning: ORA-16819: fast-start failover observer not started`
	`orcl_d - (*) Physical standby database`
		DBA

タスク	説明	必要なスキル

	Warning: ORA-16819:
	 fast-start failover
	 observer not started
	Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	WARNING (status updated
	 29 seconds ago)
	DGMGRL>
	
	CDB
	
	DGMGRL> enable fast_star
	t failover;
	Warning: ORA-16827:
	 Flashback Database is
	 disabled
	Enabled in Zero Data
	 Loss Mode.
	DGMGRL> show configura
	tion;
	Configuration - rds_dg
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_a - Primary
	 database
	  Warning: ORA-16819
	: fast-start failover
	 observer not started
	  rdscdb_b - (*)
	 Physical standby
	 database
	Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	WARNING (status
	 updated 11 seconds ago)	

タスク	説明	必要なスキル
	DGMGRL>	

タスク	説明	必要なスキル

FSFO モニタリング用のオブザーバーを起動し、ステータスを確認します。	FSFO を有効にする前でも後でもオブザーバーを起動できます。FSFO がすでに有効になっている場合、オブザーバーはただちにプライマリおよびターゲットのスタンバイデータベースの状態と接続のモニタリングを開始します。FSFO が有効になっていない場合、オブザーバーは FSFO が有効になるまでモニタリングを開始しません。
	
	オブザーバーを起動すると、前の `show configuration` コマンドで確認したように、プライマリ DB 構成がエラーメッセージなしで表示されます。
	
	非 CDB
	
	`DGMGRL> start observer;`
	`[W000 2022-12-0`
	`1T06:16:51.271+00:00]`
	`FSFO target standby is`
	`orcl_d`
	`Observer 'ip-10-0-`
	`1-89' started`
	`[W000 2022-12-0`
	`1T06:16:51.352+00:00]`
	`Observer trace level is`
	`set to USER`
	
	`DGMGRL> show configura`
	`tion`
	`Configuration - rds_dg`
		DBA

タスク	説明	必要なスキル
	Protection Mode: MaxAvailability Members: orcl_a - Primary database orcl_d - (*) Physical standby database Fast-Start Failover: Enabled in Zero Data Loss Mode Configuration Status: SUCCESS (status updated 56 seconds ago) DGMGRL> DGMGRL> show observer Configuration - rds_dg Primary: orcl_a Active Target: orcl_d Observer "ip-10-0-1-89" - Master Host Name: ip-10-0-1-89 Last Ping to Primary: 1 second ago Last Ping to Target: 1 second ago DGMGRL> CDB DGMGRL> start observer; Succeeded in opening the observer file "/home/oracle/fsfo_ip-10-0-1-56.dat". [W000 2023-01-18T07:31:32.589+00:00] FSFO target standby is rdscdb_b	

タスク	説明	必要なスキル

	Observer 'ip-10-0-1-56' started
	The observer log file is '/home/oracle/observer_ip-10-0-1-56.log'.
	
	DGMGRL> show configuration
	Configuration - rds_dg
	  Protection Mode: MaxAvailability
	  Members:
	  rdscdb_a - Primary database
	    rdscdb_b - (*) Physical standby database
	Fast-Start Failover: Enabled in Zero Data Loss Mode
	Configuration Status: SUCCESS (status updated 12 seconds ago)
	DGMGRL>
	
	DGMGRL> show observer;
	Configuration - rds_dg
	  Primary: rdscdb_a
	  Active Target: rdscdb_b
	Observer "ip-10-0-1-56" - Master
	  Host Name: ip-10-0-1-56
	  Last Ping to Primary: 1 second ago
	  Last Ping to Target: 2 seconds ago
	DGMGRL>	

タスク	説明	必要なスキル
フェイルオーバーを検証します。	このシナリオでは、プライマリ EC2 インスタンスを手動で停止することでフェイルオーバーテストを実行できます。EC2 インスタンスを停止する前に、構成に基づいて `tail` コマンドを使用してオブザーバーログファイルをモニタリングします。DGMGRL を使用してスタンバイデータベース `orcl_d` にユーザー `RDS_DATAGUARD` でログインし、Oracle Data Guard のステータスを確認します。`orcl_d` が新しいプライマリデータベースであることが表示されるはずです。 **Note** このフェイルオーバーテストシナリオでは、`orcl_d`は非 CDB データベースです。 フェイルオーバー前は、フラッシュバックデータベースは `orcl_a` で有効になっていました。元のプライマリデータベースがオンラインに戻り、`MOUNT` の状態で起動すると、オブザーバーはそのデータベースを新しいスタンバイデータベースに復元します。	DBA

タスク	説明	必要なスキル

	復元されたデータベースは、新しいプライマリデータベースの FSFO ターゲットとして機能します。詳細はオブザーバーログで確認できます。
	
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_d - Primary database`
	`Warning: ORA-16824: multiple warnings, including fast-start failover-related warnings, detected for the database`
	`orcl_a - (*) Physical standby database (disabled)`
	`ORA-16661: the standby database needs to be reinstated`
	`Fast-Start Failover: Enabled in Zero Data Loss Mode`
	`Configuration Status: WARNING (status updated 25 seconds ago)`
	`DGMGRL>`
	
	observer.log の出力例を次に示します。
	
	`$ tail -f /tmp/observer.log`	

タスク	説明	必要なスキル
	Unable to connect to database using rds_custom_orcl_a [W000 2023-01-18T07:50:32.589+00:00] Primary database cannot be reached. [W000 2023-01-18T07:50:32.589+00:00] Fast-Start Failover threshold has expired. [W000 2023-01-18T07:50:32.590+00:00] Try to connect to the standby. [W000 2023-01-18T07:50:32.590+00:00] Making a last connection attempt to primary database before proceeding with Fast-Start Failover. [W000 2023-01-18T07:50:32.591+00:00] Check if the standby is ready for failover. [S002 2023-01-18T07:50:32.591+00:00] Fast-Start Failover started... 2023-01-18T07:50:32.591+00:00 Initiating Fast-Start Failover to database "orcl_d"... [S002 2023-01-18T07:50:32.592+00:00] Initiating Fast-start Failover. Performing failover NOW, please wait...	

タスク	説明	必要なスキル
	Failover succeeded, new primary is "orcl_d" 2023-01-18T07:55:3 2.101+00:00 [S002 2023-01-1 8T07:55:32.591+00:00] Fast-Start Failover finished... [W000 2023-01-1 8T07:55:32.591+00:00] Failover succeeded. Restart pinging. [W000 2023-01-1 8T07:55:32.603+00:00] Primary database has changed to orcl_d. [W000 2023-01-1 8T07:55:33.618+00:00] Try to connect to the primary. [W000 2023-01-1 8T07:55:33.622+00: 00] Try to connect to the primary rds_custo m_orcl_d. [W000 2023-01-1 8T07:55:33.634+00: 00] The standby orcl_a needs to be reinstated [W000 2023-01-1 8T07:55:33.654+00:00] Try to connect to the new standby orcl_a. [W000 2023-01-1 8T07:55:33.654+00: 00] Connection to the primary restored! [W000 2023-01-1 8T07:55:35.654+00: 00] Disconnecting from database rds_custo m_orcl_d.	

タスク	説明	必要なスキル
	`[W000 2023-01-18T07:55:57.701+00:00] Try to connect to the new standby orcl_a. ORA-12170: TNS:Connect timeout occurred`	

Oracle PeopleSoft アプリケーションとデータベース間の接続を構成する

タスク	説明	必要なスキル
プライマリデータベースでサービスを作成して開始します。	プライマリデータベースエンドポイントとスタンバイデータベースエンドポイントの両方を含む TNS エントリを構成に使用することで、ロールの移行中にアプリケーション構成が変更されるのを防ぐことができます。読み取り/書き込みワークロードと読み取り専用ワークロードの両方をサポートする 2 つのロールベースのデータベースサービスを定義できます。次の例では、`orcl_rw` はプライマリデータベースでアクティブな読み取り/書き込みサービスです。`orcl_ro` は読み取り専用モードで開かれた、スタンバイデータベースでアクティブな読み取り専用サービスです。 `SQL> select name,open _mode from v$database;`	DBA

タスク	説明	必要なスキル
	NAME OPEN_MODE --------- --------- ----------- ORCL READ WRITE SQL> exec dbms_serv ice.create_service ('orcl_rw','orcl_r w'); PL/SQL procedure successfully completed . SQL> exec dbms_serv ice.create_service ('orcl_ro','orcl_r o'); PL/SQL procedure successfully completed . SQL> exec dbms_serv ice.start_service('orcl_rw'); PL/SQL procedure successfully completed . SQL>	

タスク	説明	必要なスキル
スタンバイデータベースでサービスを起動する。	読み取り専用のスタンバイデータベースでサービスを起動するには、次のコードを使用します。 `SQL> select name,open _mode from v$database; NAME OPEN_MODE --------- ---------- ----------- ORCL READ ONLY WITH  APPLY SQL> exec dbms_serv ice.start_service( 'orcl_ro'); PL/SQL procedure  successfully completed . SQL>`	DBA

タスク	説明	必要なスキル
プライマリ DB の再起動時にサービスを自動的に開始します。	プライマリデータベースの再起動時にサービスを自動的に開始するには、次のコードを使用します。 `SQL> CREATE OR REPLACE TRIGGER TrgDgServices after startup on database DECLARE db_role VARCHAR(30); db_open_mode VARCHAR(30); BEGIN SELECT DATABASE_ROLE, OPEN_MODE INTO db_role, db_open_mode FROM V$DATABASE; IF db_role = 'PRIMARY' THEN DBMS_SERV 2 ICE.START_SERVICE('orcl_rw'); END IF; IF db_role = 'PHYSICAL STANDBY' AND db_open_mode LIKE 'READ ONLY%' THEN DBMS_SERVICE.START_SERVICE('orcl_ro'); END IF; END; / Trigger created. SQL>`	DBA

タスク	説明	必要なスキル
読み取り/書き込みデータベースと読み取り専用データベース間の接続を構成します。	次のアプリケーション構成例を読み取り/書き込み接続と読み取り専用接続に使用できます。 `ORCL_RW = (DESCRIPTION = (CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_ TIMEOUT=3) (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST=devpsftdb.******.us-west-2.rds.amazonaws.com) (PORT=1521)) (ADDRESS = (PROTOCOL = TCP)(HOST=psftread.******.us-west-2.rds.amazonaws.com) (PORT=1521))) (CONNECT_DATA=(SERVICE_NAME = orcl_rw)))` `ORCL_RO = (DESCRIPTION = (CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_ TIMEOUT=3) (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST=devpsftdb.******.us-west-2.rds.amazonaws.com) (PORT=1521))`	DBA

タスク	説明	必要なスキル
	(ADDRESS = (PROTOCOL = TCP)(HOST=psftread.******.us-west-2.rds.amazonaws.com)(PORT=1521))) (CONNECT_DATA=(SERVICE_NAME = orcl_ro)))	

関連リソース

- Amazon RDS Custom for Oracle でData Guardのハイアベイラビリティを有効にする (AWS テクニカルガイド)
- Amazon RDS を Oracle PeopleSoft Databaseとして構成する (AWS ホワイトペーパー)
- Oracle Data Guard Broker ガイド (Oracle リファレンスドキュメント)
- Data Guard の概要および管理 (Oracle リファレンスドキュメント)
- Oracle Data Guard 固有の FAN および FCF 構成要件 (Oracle リファレンスドキュメント)

アカウント間で Amazon Redshift クラスターから Amazon S3 にデータをアンロードする

作成者: Andrew Kamel (AWS)

概要

アプリケーションをテストするときは、テスト環境に本稼働データを含めると便利です。本番稼働用データを使用すると、開発中のアプリケーションのより正確な評価を行うことができます。

このパターンは、本番環境の Amazon Redshift クラスターから、Amazon Web Services () の開発環境の Amazon Simple Storage Service (Amazon S3) バケットにデータを抽出しますAWS。

パターンは、以下を含む、DEV アカウントと PROD アカウントの両方のセットアップをステップします。

- 必要なリソース
- AWS Identity and Access Management (IAM) ロール
- Amazon Redshift 接続をサポートするサブネット、セキュリティグループ、仮想プライベートクラウド (VPC) のネットワーク調整
- アーキテクチャをテストするための Python ランタイムを使用する AWS Lambda 関数の例

Amazon Redshift クラスターへのアクセスを許可するために、このパターンでは AWS Secrets Manager を使用して関連する認証情報を保存します。利点は、Amazon Redshift クラスターの場所を知ることなく、Amazon Redshift クラスターに直接接続するために必要なすべての情報があることです。さらに、シークレットの使用をモニタリングできます。

Secrets Manager に保存されているシークレットには、Amazon Redshift クラスターのホスト、データベース名、ポート、および関連する認証情報が含まれます。

このパターンを使用する際のセキュリティ上の考慮事項については、「ベストプラクティス」セクションを参照してください。

前提条件と制限

前提条件

- PROD アカウントで実行されている Amazon Redshift クラスター

- DEV アカウントで作成された S3 バケット
- DEV アカウントと PROD アカウント間の VPC ピアリング。それに応じてルートテーブルが調整されます。
- 両方のピア接続された VPCs で有効になっている DNS ホスト名と DNS 解決

機能制限

- クエリするデータの量によっては、Lambda 関数がタイムアウトすることがあります。

  実行に最大 Lambda タイムアウト (15 分) よりも時間がかかる場合は、Lambda コードに非同期アプローチを使用します。このパターンのコード例では、Python 用の psycopg2 ライブラリを使用していますが、現在非同期処理をサポートしていません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

次の図は、DEV アカウントと PROD アカウントを使用したターゲットアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. DEV アカウントの Lambda 関数は、PROD アカウントの Secrets Manager の Amazon Redshift 認証情報にアクセスするために必要な IAM ロールを引き受けます。

   次に、Lambda 関数は Amazon Redshift クラスターシークレットを取得します。
2. DEV アカウントの Lambda 関数は、この情報を使用して、ピア接続された VPCs を介して PROD アカウントの Amazon Redshift クラスターに接続します。

   次に、Lambda 関数はアンロードコマンドを送信して、PROD アカウントの Amazon Redshift クラスターをクエリします。
3. PROD アカウントの Amazon Redshift クラスターは、DEV アカウントの S3 バケットにアクセスするための関連する IAM ロールを引き受けます。

Amazon Redshift クラスターは、クエリされたデータを DEV アカウントの S3 バケットにアンロードします。

Amazon Redshift からのデータのクエリ

次の図は、Amazon Redshift 認証情報を取得して Amazon Redshift クラスターに接続するために使用するロールを示しています。ワークフローは Lambda 関数によって開始されます。

この図表は、次のワークフローを示しています：

1. `CrossAccount-SM-Read-Role` DEV アカウントの は、PROD `SM-Read-Role` アカウントの を引き受けます。
2. `SM-Read-Role` ロールは、アタッチされたポリシーを使用して Secrets Manager からシークレットを取得します。
3. 認証情報は、Amazon Redshift クラスターへのアクセスに使用されます。

Amazon S3 へのデータのアップロード

次の図は、データを抽出して Amazon S3 にアップロードするためのクロスアカウント読み取り/書き込みプロセスを示しています。ワークフローは Lambda 関数によって開始されます。パターンは Amazon Redshift の IAM ロールを連鎖します。Amazon Redshift クラスターから送信されるアンロードコマンドは、 を引き受け`CrossAccount-S3-Write-Role`、次に を引き受けます`S3-Write-Role`。このロールの連鎖により、Amazon Redshift は Amazon S3 にアクセスできます。

ワークフローには以下のステップが含まれます。

1. `CrossAccount-SM-Read-Role` DEV アカウントの は、PROD `SM-Read-Role` アカウントの を引き受けます。
2. は、Secrets Manager から Amazon Redshift 認証情報`SM-Read-Role`を取得します。
3. Lambda 関数は Amazon Redshift クラスターに接続し、クエリを送信します。
4. Amazon Redshift クラスターは を引き受けます`CrossAccount-S3-Write-Role`。
5. は、DEV アカウントの を`CrossAccount-S3-Write-Role`引き受け`S3-Write-Role`ます。
6. クエリ結果は、DEV アカウントの S3 バケットにアンロードされます。

ツール

AWS のサービス

- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。
- Amazon Simple Storage Service (Amazon S3) は、データ量にかかわらず、保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このパターンのコードは、GitHub unload-redshift-to-s3-python リポジトリで入手できます。

ベストプラクティス

セキュリティに関する免責事項

このソリューションを実装する前に、以下の重要なセキュリティ上の推奨事項を考慮してください。

- 開発アカウントと本番稼働用アカウントを接続すると、スコープが拡大し、全体的なセキュリティ体制が軽減される可能性があることに注意してください。このソリューションを一時的にのみデプロイし、必要なデータを抽出してから、デプロイされたリソースをすぐに破棄することをお勧めします。リソースを破棄するには、Lambda 関数を削除し、このソリューション用に作成された IAM ロールとポリシーを削除し、アカウント間で付与されたネットワークアクセスを取り消す必要があります。
- 本番環境から開発環境にデータをコピーする前に、セキュリティチームとコンプライアンスチームに相談してください。個人を特定できる情報 (PII)、保護対象医療情報 (PHI)、その他の機密データや規制対象データは、通常、この方法でコピーしないでください。公開されている非機密情報 (たとえば、ショップのフロントエンドからの公開株式データ) のみをコピーします。可能な限り本番稼働用データを使用する代わりに、データのトークン化または匿名化、または合成テストデータの

生成を検討してください。AWS セキュリティ原則の 1 つは、データを人から遠ざけることです。つまり、デベロッパーは本番稼働用アカウントでオペレーションを実行しないでください。

- 本番環境の Amazon Redshift クラスターからデータを読み取ることができるため、開発アカウントの Lambda 関数へのアクセスを制限します。
- 本番環境の中断を回避するには、次の推奨事項を実装します。
  - テストおよび開発アクティビティには、専用の開発アカウントを別途使用します。
  - 厳格なネットワークアクセスコントロールを実装し、アカウント間のトラフィックを必要なもののみに制限します。
  - 本番環境とデータソースへのアクセスをモニタリングおよび監査します。
  - 関連するすべてのリソースとサービスに対して、最小特権のアクセスコントロールを実装します。
  - AWS Secrets Manager シークレットや IAM ロールのアクセスキーなどの認証情報を定期的に確認して更新します。
- この記事で使用されているサービスについては、次のセキュリティドキュメントを参照してください。
  - AWS Lambda セキュリティ
  - Amazon Redshift セキュリティ
  - Amazon S3 セキュリティ
  - AWS Secrets Manager セキュリティ
  - IAM セキュリティのベストプラクティス

セキュリティは、本番稼働用のデータとリソースにアクセスする際の最優先事項です。常にベストプラクティスに従い、最小特権のアクセスコントロールを実装し、セキュリティ対策を定期的に見直して更新します。

エピック

Amazon Redshift からデータをクエリする

タスク	説明	必要なスキル
Amazon Redshift クラスターのシークレットを作成します。	Amazon Redshift クラスターのシークレットを作成するには、次の手順を実行します。	DevOps エンジニア

タスク	説明	必要なスキル
	1. PROD アカウントで、にサインインし AWS Management Console、「https://https://console.aws.amazon.com/secretsmanager/:///https://www. 2. 新しいシークレットの保存を選択します。 3. Amazon Redshift データウェアハウスの認証情報を選択します。 4. ユーザー名とパスワードに、インスタンスの値を入力し、暗号化キーの値を確認または選択します。 5. シークレットがアクセスする Amazon Redshift データウェアハウスを選択します。 6. シークレット名`Redshift-Creds-Secret` に を入力します。 7. 残りの作成ステップをデフォルトの選択肢で完了し、保存を選択します。 8. シークレットを表示し、シークレットを識別するために生成されたシークレット ARN 値を書き留めます。	

タスク	説明	必要なスキル
Secrets Manager にアクセスするためのロールを作成します。	ロールを作成するには、以下を実行します。 1. PROD アカウントで、「https://https://console.aws.amazon.com/iam/.com で IAM コンソールを開きます。 2. [ポリシー] を選択します。 3. [Create policy] を選択します。 4. JSON タブを選択し、次のような IAM ポリシーを入力します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "secretsmanager:GetResourcePolicy", "secretsmanager:GetSecretValue", "secretsmanager:DescribeSecret", "secretsmanager:ListSecretVersionIds" ],`	DevOps エンジニア

タスク	説明	必要なスキル
	"Resource": ["<Redshift-Creds-Secret-ARN>"] }, { "Effect": "Allow", "Action": "secretsmanager:ListSecrets", "Resource": "*" }] } を、Amazon Redshift クラスターの情報と認証情報を含む Secrets Manager シークレットの Amazon リソースネーム (ARN) `Redshift-Creds-Secret-ARN` に置き換えます。	

データを Simple Storage Service (Amazon S3) にアップロードする

タスク	説明	必要なスキル
S3 バケットにアクセスするためのロールを作成します。	S3 バケットにアクセスするためのロールを作成するには、次の手順を実行します。	DevOps エンジニア

タスク	説明	必要なスキル
	1. DEV アカウントで、IAM コンソールを開きます。 2. [ポリシー] を選択します。 3. [Create policy] を選択します。 4. JSON タブを選択し、次のような IAM ポリシーを入力します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "kmsstmt", "Effect": "Allow", "Action": [ "kms:Decrypt", "kms:Encrypt", "kms:GenerateDataKey" ], "Resource": [ "<kms-key-arn>" ] }, { "Sid": "s3stmt", "Effect": "Allow", "Action": [ "s3:PutObject",`	

タスク	説明	必要なスキル

	<pre>"s3:Get*",
	"s3:List*"
	],
	"Resource": [
	"arn:aws:
	s3:::mybucket",
	"arn:aws:
	s3:::mybucket/*"
	]
	}
	]
	}</pre>
	を、アクセスする S3 バケットの名前`mybucket`に置き換えます。また、S3 バケットが暗号化されている場合は、 を S3 バケットの暗号化に使用される AWS Key Management Service (AWS KMS) キーの ARN `kms-key-arn` に置き換えます。それ以外の場合は、ポリシーの AWS KMS セクションは必要ありません。
	5. 「ポリシーの確認」を選択し、「ポリシー名`S3-Write-Policy` 」と入力してから、「ポリシーの作成」を選択します。
	6. ナビゲーションペインで [Roles (ロール)] を選択します。
	7. [Create role] を選択します。	

タスク	説明	必要なスキル
	8. 信頼されたエンティティロールで、カスタム信頼ポリシーを選択します。 9. 次へ: アクセス許可を選択し、作成した S3-Write-Policy ポリシーを選択します。 10.をロール名`S3-Write-Role` として入力し、ロールの作成を選択します。	

タスク	説明	必要なスキル
Amazon Redshift ロールを作成します。	Amazon Redshift ロールを作成するには、次の手順を実行します。 1. PROD アカウントで、IAM コンソールを開きます。 2. [ポリシー] を選択します。 3. [Create policy] を選択します。 4. JSON タブを選択し、次のような IAM ポリシーを入力します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "CrossAccountPolicy", "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "S3-Write-Role-ARN" } ] }` を DEV アカウントの `S3-Write-Role` の ARN `S3-Write-Role-ARN` に置き換えます。	DevOps エンジニア

タスク	説明	必要なスキル
	5. 「ポリシーの確認」を選択し、「ポリシー名`S3-Write-Role-Assume-Policy` 」と入力してから、「ポリシーの作成」を選択します。 6. ナビゲーションペインで ロール を選択してから、ロールを作成する を選択します。 7. 信頼できるエンティティタイプとして AWS サービスを選択し、Redshift、Redshift - カスタマイズ可能を選択します。 8. 次へ: アクセス許可を選択し、作成した S3-Write-Role-Assume-Policy ポリシーを選択します。 9. ロール名`CrossAccount-S3-Write-Role` としてを入力し、ロールの作成を選択します。 10. IAM ロールを Amazon Redshift クラスターに関連付けます。	

Lambda 関数をデプロイします

タスク	説明	必要なスキル
Lambda 関数をデプロイします。	ピア接続された VPC に Lambda 関数をデプロイするには、次の手順を実行します。 1. Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。 2. 関数を選択します。 3. [Create function (関数の作成)] を選択します。 4. [基本的な情報] の [関数名] に、関数の名前を入力します。 5. Runtime で、Python 3.8 を選択します。 6. デフォルトの実行ロールの変更を展開し、次の操作を行います。 a. [既存のロールを使用する] を選択します。 b. 既存のロールで、前に作成した CrossAccount-RM-Read-Role Lambda ロールを選択します。 7. 詳細設定を展開し、次の操作を行います。 a. VPC を有効にするチェックボックスをオンにします。	DevOps エンジニア

タスク	説明	必要なスキル
	b. VPC の場合は、DEV アカウントのピア接続された VPC を選択します。 c. Subnets で、プライベートサブネットを選択します。 d. [Security groups (セキュリティグループ)] で、デフォルトのセキュリティグループを選択します。 8. [Create function (関数の作成)] を選択します。 9. Lambda 関数にレイヤーとして psycopg2 ライブラリを追加します。 **Note** psycopg2-lambda-layer リポジトリからデプロイ済みの Layer を使用できます。 AWS リージョン および Python ランタイムに基づいて URL を使用してください。	

アーキテクチャをテストする

タスク	説明	必要なスキル

必要なリソースをインポートします。	必要なリソースをインポートするには、次のコマンドを実行します。
	
	`import ast`
	`import boto3`
	`import psycopg2`
	`import base64`
	`from botocore.exceptions import ClientError`
		アプリ開発者

Lambda ハンドラー関数を実行します。	Lambda 関数は、クロスアカウントアクセスと一時的な認証情報管理に AWS Security Token Service（AWS STS) を使用します。関数は AssumeRole API オペレーションを使用して、IAM `sm_read_role` ロールのアクセス許可を一時的に引き受けます。
	
	Lambda 関数を実行するには、次のコード例を使用します。
	
	`def lambda_handler(event, context):`
	`  sts_client = boto3.client('sts')`
	
	`  # Secrets Manager Configurations`
		アプリ開発者

タスク	説明	必要なスキル

	secret_name = "redshift_creds"
	sm_region = "eu-west-1"
	sm_read_role = "arn:aws:iam::PROD_ACCOUNT_NUMBER:role/SM-Read-Role"
	
	# S3 Bucket Configurations
	s3_bucket_path = "s3://mybucket/"
	s3_bucket_region = "eu-west-1"
	s3_write_role = "arn:aws:iam::DEV_ACCOUNT_NUMBER:role/S3-Write-Role"
	
	# Redshift Configurations
	sql_query = "select * from category"
	redshift_db = "dev"
	redshift_s3_write_role = "arn:aws:iam::PROD_ACCOUNT_NUMBER:role/CrossAccount-S3-Write-Role"
	
	chained_s3_write_role = "%s,%s" % (redshift_s3_write_role, s3_write_role)
	
	assumed_role_object = sts_client.assume_role(
	RoleArn=sm_read_role,	

タスク	説明	必要なスキル

	RoleSessionName="CrossAccountRoleAssumption",
	ExternalId="YOUR_EXTERNAL_ID",
	)
	credentials = assumed_role_object['Credentials']
	
	secret_dict = ast.literal_eval(get_secret(credentials, secret_name, sm_region))
	execute_query(secret_dict, sql_query, s3_bucket_path, chained_s3_write_role, s3_bucket_region, redshift_db)
	
	return {
	'statusCode': 200
	}	

タスク	説明	必要なスキル

シークレットを取得します。	Amazon Redshift シークレットを取得するには、次のサンプルコードを使用します。
	
	`def get_secret(credentials, secret_name, sm_region):`
	`  # Create a Secrets Manager client`
	`  session = boto3.session.Session()`
	`  sm_client = session.client(`
	`  service_name='secretsmanager',`
	`  aws_access_key_id=credentials['AccessKeyId'],`
	`  aws_secret_access_key=credentials['SecretAccessKey'],`
	`  aws_session_token=credentials['SessionToken'],`
	`  region_name=sm_region`
	`  )`
	
	`  try:`
	`  get_secret_value_response = sm_client.get_secret_value(`
	`  SecretId=secret_name`
	`  )`
	`  except ClientError as e:`
	`  print(e)`
	`  raise e`
		アプリ開発者

タスク	説明	必要なスキル

	<pre> else:
	        if 'SecretSt
	ring' in get_secre
	t_value_response:
	            return
	 get_secret_value_r
	esponse['SecretStr
	ing']
	        else:
	            return
	 base64.b64decode(g
	et_secret_value_re
	sponse['SecretBina
	ry'])</pre>	

タスク	説明	必要なスキル

アンロードコマンドを実行します。	S3 バケットにデータをアンロードするには、次のサンプルコードを使用します。
	
	`def execute_query(secret_dict, sql_query, s3_bucket_path, chained_s3_write_role, s3_bucket_region, redshift_db):`
	`  conn_string = "dbname='%s' port='%s' user='%s' password='%s' host='%s'" \`
	`  % (redshift_db,`
	`  secret_dict["port"],`
	`  secret_dict["username"],`
	`  secret_dict["password"],`
	`  secret_dict["host"])`
	`  con = psycopg2.connect(conn_string)`
	`  unload_command = "UNLOAD ('{}') TO '{}' IAM_ROLE '{}' DELIMITER '|' REGION '{}';" \`
	`  .format(sql_query,`
	`  s3_bucket_path + str(datetime.datetime.now()) + ".csv",`
		アプリ開発者

タスク	説明	必要なスキル

	chained_s3_write_role,
	
	s3_bucket_region)
	
	# Opening a cursor and run query
	cur = con.cursor()
	cur.execute(unload_command)
	
	print(cur.fetchone())
	cur.close()
	con.close()	

クリーンアップ

タスク	説明	必要なスキル
Lambda 関数を削除する	予期しないコストが発生しないようにするには、リソースと DEV アカウントと PROD アカウント間の接続を削除します。 Lambda 関数を削除するには、次の手順を実行します。 1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/://www.com で開きます。 2. 作成した Lambda 関数を見つけて選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	3. [アクション] を選択し、[削除] を選択します。 4. 削除を確定します。	

タスク	説明	必要なスキル
IAM ロールとポリシーを削除します。	DEV アカウントと PROD アカウントから IAM ロールとポリシーを削除します。 DEV アカウントで、次の操作を行います。 1. [IAM コンソール] を開きます。 2. 次のロールを削除します。 • `S3-Write-Role` • `CrossAccount-RM-Read-Role` (Lambda ロール) 3. 関連付けられたポリシーを削除します。 • `S3-Write-Policy` • PROD CrossAccountロールを引き受けるクロスアカウントポリシー PROD アカウントで、次の操作を行います。 1. [IAM コンソール] を開きます。 2. 次のロールを削除します。 • `SM-Read-Role` • `CrossAccount-S3-Write-Role` 3. 関連付けられたポリシーを削除します。	DevOps エンジニア

タスク	説明	必要なスキル
	• Secrets Manager にアクセスするための CrossAccount ポリシー • `S3-Write-Role-Assume-Policy`	
Secrets Manager でシークレットを削除します。	シークレットを削除するには、次の手順を実行します。 1. PROD アカウントで、Secrets Manager コンソールを開きます。 2. という名前のシークレットを見つけて選択しますRedshift-Creds-Secret 。 3. [Actions] (アクション) を選択してから、[Delete secret] (シークレットの削除) を選択します。 4. 削除を確定します。	DevOps エンジニア

タスク	説明	必要なスキル
VPC ピアリングとセキュリティグループのルールを削除します。	VPC ピアリングとセキュリティグループのルールを削除するには、次の手順を実行します。 1. PROD アカウントで、Amazon EC2「https://https://console.aws.amazon.com/ec2/:///https://https://https://

タスク	説明	必要なスキル
S3 バケットからデータを削除します。	Amazon S3 からデータを削除するには、次の手順を実行します。 1. DEV アカウントで、Amazon S3「https://https://console.aws.amazon.com/s3/:///https://https://https://https://https://https://https://https://https://https://https://https://https:// 2. データストレージに使用したバケットを見つけます。 3. バケット内のオブジェクトを削除するか、不要になった場合はバケット全体を削除します。	DevOps エンジニア
AWS KMS キーをクリーンアップします。	暗号化用のカスタム AWS KMS キーを作成した場合は、次の操作を行います。 1. AWS KMS コンソールを https://console.aws.amazon.com/kms/:///https://https://https://https://https://https 2. このパターン用に作成されたキーを見つけます。 3. キーの削除をスケジュールします。（キーの削除には必須の待機期間があります）。	DevOps エンジニア

タスク	説明	必要なスキル
Amazon CloudWatch logsを確認して削除します。	CloudWatch ログを削除するには、次の手順を実行します。 1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。 2. Lambda 関数または Amazon Redshift クラスターによって作成されたロググループを確認します。 3. これらのロググループが不要になった場合は削除します。	DevOps エンジニア

関連リソース

- 「Amazon CloudWatch ドキュメント」
- 「IAM ドキュメント」
- Lambda ドキュメント
- Amazon Redshift ドキュメント
- 「Amazon S3 ドキュメント」
- AWS Secrets Manager ドキュメント
- AWS セキュリティ原則

追加情報

Amazon Redshift から Amazon S3 にデータをアンロードした後、Amazon Athena を使用してデータを分析できます。

Amazon Athena はビッグデータクエリサービスであり、大量のデータにアクセスする必要がある場合に役立ちます。サーバーやデータベースをプロビジョニングしなくても Athena を使用できます。Athena は複雑なクエリをサポートしており、さまざまなオブジェクトで実行できます。

ほとんどの と同様に AWS のサービス、Athena を使用する主な利点は、複雑さを増すことなくクエリを実行する方法に大きな柔軟性があることです。Athena を使用すると、データ型を変更せずに、CSV や JSON などのさまざまなデータ型を Amazon S3 でクエリできます。外部を含むさまざまなソースからデータをクエリできます AWS。Athena は、サーバーを管理する必要がないため、複雑さを軽減します。Athena は、クエリを実行する前にデータをロードまたは変更することなく、Amazon S3 から直接データを読み取ります。

ワークロード別のデータベース移行パターン

トピック

- IBM
- Microsoft
- 該当なし
- オープンソース
- Oracle
- SAP

IBM

- AWS DMS を使用して Db2 データベースを Amazon EC2 から Aurora MySQL 互換のデータベースに移行する
- ログ配信を使用して Db2 for LUW を Amazon EC2 に移行することで、システム停止時間を短縮する
- 高可用性ディザスタリカバリ機能を備えた Db2 for LUW を Amazon EC2 に移行する
- AWS DMS と AWS SCT を使用して IBM Db2 on Amazon EC2 から Aurora PostgreSQL 互換に移行する
- IBM WebSphere アプリケーションサーバーから Amazon EC2 の Apache Tomcat に移行
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

Microsoft

- Microsoft ワークロードの検出と AWS への移行を加速する
- リンクサーバーを使用して Amazon EC2 上のMicrosoft SQL サーバーからオンプレミスの Microsoft SQL Server テーブルにアクセスする
- SQL Server データベースを AWS 上の MongoDB Atlas に移行する際のクエリパフォーマンスを評価する
- AWS Lambda および Task Scheduler を使用して、Amazon EC2 で実行されている SQL Server Express Edition のデータベースタスクを自動化する
- Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションへのデータベース移行をサポートするように Python と Perl アプリケーションを変更する
- AWS 上の SQL Server の Always On アベイラビリティグループで読み取り専用ルーティングを構成する
- Microsoft Excel と Python を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon S3 にエクスポートする
- AWS DMS を使用して Amazon RDS for SQL Server テーブルを S3 バケットにエクスポートする
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- EC2 Windows インスタンスを AWS Managed Services アカウントに取り込み、移行
- メッセージキューを Microsoft Azure Service Bus から Amazon SQS に移行
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon EC2 から Amazon DocumentDB に移行します
- AWS DMS と AWS SCT を使用して Microsoft SQL Server データベースを Aurora MySQL に移行
- .NET アプリケーションを Microsoft Azure App Serviceから AWS Elastic Beanstalk に移行
- オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する
- オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- リンクされたサーバーを使用して、オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- ネイティブバックアップと復元メソッドを使用して、オンプレミスの Microsoft SQL サーバーデータベースを Amazon RDS for SQL Server に移行
- AWS DMS を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する

- AWS SCT データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- オンプレミス Microsoft SQL Server データベースを、Linux を実行中の Amazon EC2 上の Microsoft SQL Server に移行する
- Rclone を使用して Microsoft Azure Blob から Amazon S3 にデータを移行する
- でリレーショナルデータベースを MongoDB Atlas に移行する AWS
- 分散可用性グループを使用して SQL Server を AWS に移行する
- ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行
- AWS Cloud でオンプレミスワークロードをリホストする: 移行チェックリスト
- Microsoft SQL サーバーを AWS クラウドに移行した後に接続エラーを解決する
- オンプレミスの SMTP サーバーとデータベースメールを使用して、Amazon RDS for SQL Server データベースインスタンスに通知を送信します。
- Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する
- Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする

該当なし

- へのリホスト移行中にファイアウォールリクエストの承認プロセスを作成する AWS
- 既存の Amazon RDS for PostgreSQL DB インスタンスを暗号化する
- Amazon DynamoDB テーブルのストレージコストを推定
- AWS DMS と Amazon Aurora によるクロスリージョンディザスタリカバリの実装

オープンソース

- Python アプリケーションを使用して Amazon DynamoDB の PynamoDB モデルと CRUD 関数を自動的に生成する DynamoDB
- pgAdmin の SSH トンネルを使用して接続
- Aurora PostgreSQL-Compatible でのアプリケーションユーザーとロールの作成
- Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする
- AWS SCT と AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行する AWS CLIAWS CloudFormation
- ネイティブツールを使用して オンプレミスの MariaDB Amazon RDS for MariaDB に移行する
- オンプレミス MySQL データベースを Amazon EC2 に移行する
- オンプレミス MySQL データベースを Amazon RDS for MySQL に移行する
- オンプレミス MySQL データベースを Aurora MySQL に移行する
- オンプレミス PostgreSQL データベースを Aurora PostgreSQL に移行する
- Couchbase Server データベースを Amazon EC2 に移行する
- IBM WebSphere アプリケーションサーバーから Amazon EC2 上の Apache Tomcat への自動スケーリングによる移行
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for Oracle に移行
- Oracle GlassFish から AWS Elastic Beanstalk へ移行する
- pglogic を使用して Amazon EC2 上の PostgreSQL から Amazon RDS for PostgreSQL に移行する
- AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する
- AWS App2Container を使用したオンプレミスの Java プリケーションの AWS への移行
- Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミスの MySQL データベースを Aurora MySQL に移行する
- Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行
- 100 個以上の引数を持つ Oracle 関数とプロシージャを PostgreSQL に移行
- Redis ワークロードを AWS 上の Redis Enterprise Cloud に移行
- 暗号化されていない Amazon Aurora インスタンスをモニタリングする
- RHEL ソースサーバーを再起動した後、SELinux を無効にせずに AWS レプリケーションエージェントを自動的に再起動する

- Lambda と Secrets Manager を使用して Amazon RDS for PostgreSQL と Aurora PostgreSQL のジョブをスケジュールする
- GTID を使用して Amazon RDS for MySQL と Amazon EC2 上の MySQL との間のデータレプリケーションを設定する
- pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する
- アカウント間で Amazon Redshift クラスターから Amazon S3 にデータをアンロードする

Oracle

- リードレプリカを使用して Amazon RDS Customの Oracle PeopleSoft に HA を追加
- JSON Oracleクエリを PostgreSQL データベース SQL に変換
- Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換
- PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート
- Aurora PostgreSQL のカスタムエンドポイントを使用して Oracle RAC ワークロードをエミュレートします
- AWR レポートを使用して Oracle データベースの Amazon RDS エンジンサイズを推定
- Aurora PostgreSQL の動的 SQL ステートメントの匿名ブロックを処理
- Aurora PostgreSQL 互換にオーバーロードされた Oracle関数を処理
- Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行
- Aurora PostgreSQL 互換のファイルエンコーディングを使用して BLOB ファイルを TEXT にロード
- Amazon RDS for Oracle DB インスタンスを AMS を使用する他のアカウントに移行する
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- Amazon RDS for Oracle データベースを別の に移行 AWS アカウント し、 AWS リージョン を使用して継続的なレプリケーション AWS DMS を行う
- Amazon RDS for Oracle DB インスタンスを別の VPC へ移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon EC2 に移行する
- Logstash を使用して、オンプレミス Oracle データベースを Amazon OpenSearch Service へ移行する
- AWS DMS と AWS SCT を使用してオンプレミスの Oracle データベースを Amazon RDS for MySQL に移行する
- オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- データベースリンクを経由した直接 Oracle Data Pump Import を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle バイスタンダーと AWS DMS を使用して、オンプレミスの Oracle データベースを Amazon RDS for PostgreSQL に移行する

- オンプレミスの Oracle データベースを Oracle Amazon EC2 に移行する
- AWS DMS と AWS SCT を使用して、Oracle データベースを Amazon EC2 から Amazon RDS for MariaDB に移行する
- AWS DMS を使用して Oracle データベースを Amazon EC2 から Amazon RDS for Oracle に移行する
- AWS DMS を使用して Oracle データベースを Amazon DynamoDB に移行する
- Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行
- Oracle Data Pump と AWS DMS を使用して Oracle JD Edwards EnterpriseOne データベースを AWS に移行します
- AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行
- AWS DMS を使用して Oracle PeopleSoft データベースを AWS に移行する
- オンプレミスの Oracle データベースから Aurora PostgreSQL にデータを移行する
- Amazon RDS for Oracle から Amazon RDS for MySQL に移行する
- マテリアライズドビューと AWS DMS を使用して Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- Oracle GoldenGate を使用して Oracle Database から Amazon RDS for PostgreSQL に移行
- AWS DMS と AWS SCT を使用して Oracle on Amazon EC2 から Amazon RDS for MySQL に移行する
- AWS DMS を使用して Oracle から Amazon DocumentDB に移行する
- Amazon ECS で Oracle WebLogic から Apache Tomcat (ToMee) に移行する
- 関数ベースのインデックスを Oracle から PostgreSQL に移行する
- レガシーアプリケーションを Oracle Pro*C から ECPG に移行する
- Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行
- Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する
- Oracle E-Business Suite を Amazon RDS Custom に移行
- エクステンションを使用して Oracle のネイティブ関数を PostgreSQL に移行

- Oracle の OUT バインド変数を PostgreSQL データベースに移行
- Oracle PeopleSoft を Amazon RDS Custom に移行
- Oracle ROWID 機能を AWS の PostgreSQL に移行
- Oracle SERIALLY_REUSABLE プラグマパッケージを PostgreSQL に移行
- 仮想生成列をOracleから PostgreSQL に移行
- Amazon CloudWatch を使用して Oracle GoldenGate ログを監視する
- Amazon RDS for Oracle で Oracle Database エンタープライズエディションから標準エディション 2 へリプラットフォームする
- Amazon RDS Custom でアクティブスタンバイデータベースを使用して Oracle E-Business Suite の HA/DR アーキテクチャを設定する
- Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする
- Amazon RDS Custom for Oracle 上の Oracle PeopleSoft アプリケーションの移行ロール
- Oracle から Amazon Aurora PostgreSQL への移行後にデータベースオブジェクトを検証する

SAP

- Systems Manager と EventBridge を使用した SAP HANA データベースの自動バックアップ
- オンプレミスの SAP ASE データベースを Amazon EC2 に移行
- AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する
- AWS SCT と AWS DMS を使用して Amazon EC2 上の SAP ASE を Amazon Aurora PostgreSQL 互換の Amazon Aurora PostgreSQL 互換に移行します
- 同じホスト名の SAP HSR を使用して SAP HANA を AWS に移行します
- アプリケーション移行サービスを使用することで、同種の SAP 移行のカットオーバー時間を短縮する
- AWS の IBM Db2 に SAP のディザスタリカバリをセットアップ

その他のパターン

- Amazon EKS コンテナから Amazon Neptune データベースにアクセスする
- Athena による Amazon DynamoDB テーブルへのアクセス、クエリ、結合
- Athena でのML 予測のため、Amazon DynamoDB 内のデータを集約
- EC2 インスタンスを AMS アカウントの S3 バケットへの書き込みアクセスを許可
- Amazon Athena と Amazon QuickSight を使用してネストされた JSON データを分析および視覚化する
- AWS Directory Service を使用して Amazon EC2 の Microsoft SQL Server を認証する
- AWS Batch を使用して Amazon RDS for PostgreSQL DBインスタンスのバックアップを自動化します
- DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする
- 暗号化されていない Amazon RDS DB インスタンスとクラスターを自動的に修正する
- AWS Systems Manager メンテナンスウィンドウを使用して Amazon RDS DB インスタンスを自動的に停止および起動する
- DevOps プラクティスと AWS Cloud9 を使用して、マイクロサービスで疎結合アーキテクチャを構築する
- AWS Mainframe Modernization と を使用して COBOL Db2 プログラムを構築する AWS CodeBuild
- Amazon DataZone を使用してエンタープライズデータメッシュを構築する AWS CDKAWS CloudFormation
- Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションへのデータベース移行をサポートするように Python と Perl アプリケーションを変更する
- Python を使用して EBCDIC データを AWS 上の ASCII に変換およびアンパックします
- Teradata NORMALIZE 時間的特徴量を Amazon Redshift SQL に変換
- Teradata RESET WHEN 特徴量を Amazon Redshift SQL に変換
- Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換
- Aurora PostgreSQL-Compatible でのアプリケーションユーザーとロールの作成
- Microsoft Excel と Python を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成
- で Kinesis Data Streams と Firehose を使用して DynamoDB レコードを Amazon S3 に配信する AWS CDK

- Amazon EC2 にプライベート静的 IP を使用して Cassandra クラスターをデプロイしてリバランスを回避する
- RAG および ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する
- PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート
- Amazon RDS for SQL Server で透過的なデータ暗号化を有効にする
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon S3 にエクスポートする
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行
- Aurora PostgreSQL 互換のファイルエンコーディングを使用して BLOB ファイルを TEXT にロード
- AWS Secrets Manager を使用して認証情報を管理
- AWS DMS を使用して Db2 データベースを Amazon EC2 から Aurora MySQL 互換のデータベースに移行する
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon EC2 から Amazon DocumentDB に移行します
- AWS DMS と AWS SCT を使用して Microsoft SQL Server データベースを Aurora MySQL に移行
- でセルフホスト型 MongoDB 環境を MongoDB Atlas に移行する AWS
- AWS SCT データ抽出エージェントを使用して Teradata データベースを Amazon Redshift に移行
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- AWS SCT と AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行する AWS CLIAWS CloudFormation
- Amazon RDS DB インスタンスを別の VPC またはアカウントに移行する
- Amazon RDS for Oracle データベースを別の に移行 AWS アカウント し、 AWS リージョン を使用して継続的なレプリケーション AWS DMS を行う
- Amazon RDS for Oracle DB インスタンスを別の VPC へ移行する
- Amazon Redshift クラスターを中国の AWS リージョンに移行する
- ネイティブツールを使用して オンプレミスの MariaDB Amazon RDS for MariaDB に移行する
- オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する
- オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する

- リンクされたサーバーを使用して、オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- ネイティブバックアップと復元メソッドを使用して、オンプレミスの Microsoft SQL サーバーデータベースを Amazon RDS for SQL Server に移行
- AWS DMS を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- AWS SCT データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- オンプレミス Microsoft SQL Server データベースを、Linux を実行中の Amazon EC2 上の Microsoft SQL Server に移行する
- オンプレミス MySQL データベースを Amazon EC2 に移行する
- オンプレミス MySQL データベースを Amazon RDS for MySQL に移行する
- オンプレミス MySQL データベースを Aurora MySQL に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon EC2 に移行する
- Logstash を使用して、オンプレミス Oracle データベースを Amazon OpenSearch Service へ移行する
- AWS DMS と AWS SCT を使用してオンプレミスの Oracle データベースを Amazon RDS for MySQL に移行する
- オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- データベースリンクを経由した直接 Oracle Data Pump Import を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle バイスタンダーと AWS DMS を使用して、オンプレミスの Oracle データベースを Amazon RDS for PostgreSQL に移行する
- オンプレミスの Oracle データベースを Oracle Amazon EC2 に移行する
- オンプレミス PostgreSQL データベースを Aurora PostgreSQL に移行する
- オンプレミスの SAP ASE データベースを Amazon EC2 に移行
- オンプレミスの ThoughtSpot Falcon データベースを Amazon Redshift に移行する
- AWS SCT データ抽出エージェントを使用してオンプレミスの Vertica データベースを Amazon Redshift に移行する

- AWS DMS と AWS SCT を使用して、Oracle データベースを Amazon EC2 から Amazon RDS for MariaDB に移行する
- AWS DMS を使用して Oracle データベースを Amazon EC2 から Amazon RDS for Oracle に移行する
- AWS DMS を使用して Oracle データベースを Amazon DynamoDB に移行する
- Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行
- Oracle Data Pump と AWS DMS を使用して Oracle JD Edwards EnterpriseOne データベースを AWS に移行します
- AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行
- AWS DMS を使用して Oracle PeopleSoft データベースを AWS に移行する
- Couchbase Server データベースを Amazon EC2 に移行する
- オンプレミスの Oracle データベースから Aurora PostgreSQL にデータを移行する
- Starburst AWS クラウド を使用してデータを に移行する
- ログ配信を使用して Db2 for LUW を Amazon EC2 に移行することで、システム停止時間を短縮する
- 高可用性ディザスタリカバリ機能を備えた Db2 for LUW を Amazon EC2 に移行する
- Amazon RDS for Oracle から Amazon RDS for MySQL に移行する
- カウチベースサーバーから AWS 上のカウチベースカペラへの移行
- AWS DMS と AWS SCT を使用して IBM Db2 on Amazon EC2 から Aurora PostgreSQL 互換に移行する
- マテリアライズドビューと AWS DMS を使用して Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- Oracle GoldenGate を使用して Oracle Database から Amazon RDS for PostgreSQL に移行
- AWS DMS と AWS SCT を使用して Oracle on Amazon EC2 から Amazon RDS for MySQL に移行する
- AWS DMS を使用して Oracle から Amazon DocumentDB に移行する

- pglogic を使用して Amazon EC2 上の PostgreSQL から Amazon RDS for PostgreSQL に移行する
- AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する
- 関数ベースのインデックスを Oracle から PostgreSQL に移行する
- レガシーアプリケーションを Oracle Pro*C から ECPG に移行する
- オンプレミスの Cloudera ワークロードを AWS 上の Cloudera データプラットフォームに移行する
- Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミスの MySQL データベースを Aurora MySQL に移行する
- Oracle ビジネスインテリジェンス 12c をオンプレミスサーバーから AWS クラウドに移行
- Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行
- Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する
- Oracle E-Business Suite を Amazon RDS Custom に移行
- Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行
- エクステンションを使用して Oracle のネイティブ関数を PostgreSQL に移行
- Oracle PeopleSoft を Amazon RDS Custom に移行
- Oracle ROWID 機能を AWS の PostgreSQL に移行
- Oracle SERIALLY_REUSABLE プラグマパッケージを PostgreSQL に移行
- Redis ワークロードを AWS 上の Redis Enterprise Cloud に移行
- でリレーショナルデータベースを MongoDB Atlas に移行する AWS
- AWS SCT と AWS DMS を使用して Amazon EC2 上の SAP ASE を Amazon Aurora PostgreSQL 互換の Amazon Aurora PostgreSQL 互換に移行します
- 仮想生成列をOracleから PostgreSQL に移行
- 組織間でデータを共有するための最小限の実行可能なデータスペースを設定する
- Amazon ElastiCache クラスターの保存時の暗号化をモニタリングする
- セキュリティグループの ElastiCache クラスターをモニタリング
- Amazon Athena を使用して SQL で Amazon DynamoDB テーブルをクエリする Amazon Athena
- アプリケーション移行サービスを使用することで、同種の SAP 移行のカットオーバー時間を短縮する
- コンテナを再起動せずにデータベースの認証情報をローテーションする
- AWS Fargate を使用してメッセージ駆動型の大規模なワークロード実行する

- トラステッドコンテキストを使用して、AWS の Db2 フェデレーションデータベースのユーザーアクセスを保護し、合理化する
- AWS で可用性の高い PeopleSoft アーキテクチャを設定する
- Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS
- PGO を使用して Amazon EKS への PostgreSQL デプロイを合理化する
- 大規模な Db2 z/OS データを CSV ファイルで Amazon S3 に転送する
- pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する
- Oracle から Amazon Aurora PostgreSQL への移行後にデータベースオブジェクトを検証する
- 新しい Amazon Redshift クラスターが VPC で起動することを検証

ストレージとバックアップ

トピック

- EC2 インスタンスを AMS アカウントの S3 バケットへの書き込みアクセスを許可
- Snowflake Snowpipe、Amazon S3、Amazon SNSAmazon SNS Data Firehose を使用して Snowflake データベースへのデータストリームの取り込みを自動化する
- 既存および新規の Amazon EBS ボリュームを自動的に暗号化する
- AWS クラウド上の Stromasys Charon-SSP エミュレーターで Sun SPARC サーバーをバックアップ
- Veeam Backup & Replication を使用してデータを Amazon S3 にバックアップおよびアーカイブする
- VMware Cloud on AWS 向けVeritas NetBackup の構成
- AWS CLI を使用して S3 バケットから別のアカウントとリージョンにデータをコピーする
- S3 バッチレプリケーションを使用して S3 バケットから別のアカウントとリージョンにデータをコピーする
- Amazon S3 用 AWS PrivateLink と DistCP を使用して、オンプレミスの Hadoop 環境から Amazon S3 にデータを移行する
- その他のパターン

EC2 インスタンスを AMS アカウントの S3 バケットへの書き込みアクセスを許可

作成者: Mansi Suratwala (AWS)

概要

AWS Managed Services (AMS) は、 AWS インフラストラクチャをより効率的かつ安全に運用するのに役立ちます。AMS アカウントには、 AWS リソースの標準化された管理のためのセキュリティガードレールがあります。1 つのガードレールは、デフォルトの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイルでは、Amazon Simple Storage Service (Amazon S3) バケットへの書き込みアクセスが許可されていないことです。ただし、組織には複数の S3 バケットがあり、EC2 インスタンスによるアクセスをより細かく制御する必要がある場合があります。たとえば、EC2 インスタンスのデータベースバックアップを S3 バケットに保存できます。

このパターンでは、変更リクエスト (RFCs) を使用してEC2 インスタンスが AMS アカウントの S3 バケットに書き込みアクセスできるようにする方法について説明します。RFC は、管理対象環境を変更するためにユーザーまたは AMS が作成したリクエストで、特定の操作の「変更タイプ」(CT) ID が含まれます。

前提条件と制限

前提条件

- AMS Advanced アカウント。詳細については、AMS ドキュメントの「AMS オペレーションプラン」を参照してください。
- RFCs を送信するための AWS Identity and Access Management (IAM) `customer-mc-user-role`ロールへのアクセス。
- AWS Command Line Interface （ AWS CLI ）。AMS アカウントの EC2 インスタンスでインストールおよび設定されます。
- AMS で RFC を作成して提出する方法に対する理解。詳細については、AMS ドキュメントの「AMS 変更タイプとは」を参照してください。
- 手動変更タイプと自動変更タイプ (CT) に対する理解。詳細については、AMS ドキュメントの「自動 CT と手動 CTs」を参照してください。

アーキテクチャ

テクノロジースタック

- AMS
- AWS CLI
- Amazon EC2
- Amazon S3
- IAM

ツール

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Managed Services (AMS) は、AWS インフラストラクチャをより効率的かつ安全に運用するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

エピック

RFC で S3 バケットを作成

タスク	説明	必要なスキル
自動 RFC を使用して S3 バケットを作成します。	1. AMS アカウントにサインインし、変更タイプの選択ページを選択し、RFC を選択して、RFC の作成を選択します。 2. S3 バケットの作成自動 RFC を送信します。	AWS システム管理者、AWS 開発者

タスク	説明	必要なスキル
	Note S3 バケットの名前を必ず記録してください。	

EC2 インスタンスに関連付ける IAM インスタンスプロファイルロールを作成します。

タスク	説明	必要なスキル
手動 RFC を提出して IAM ロールを作成します。	AMS アカウントがオンボードされると、 という名前のデフォルトの IAM インスタンスプロファイル`customer-mc-ec2-instance-profile` が作成され、AMS アカウントの各 EC2 インスタンスに関連付けられます。ただし、インスタンスプロファイルには S3 バケットへの書き込みアクセス許可はありません。 書き込みアクセス許可を追加するには、IAM リソースの作成手動 RFC を送信して、、`customer_ec2_instance_` 、`customer_deny_policy` の 3 つのポリシーを持つ IAM ロールを作成します `customer_ec2_s3_integration_policy` 。	AWS システム管理者、AWS 開発者

タスク	説明	必要なスキル
	Important customer_ec2_instance_ および customer_deny_policy ポリシーは AMS アカウントに既に存在します。ただし、次のサンプルポリシーcustomer_ec2_s3_integration_policy を使用してを作成する必要があります。	

```
{
  "Version": "2012-10-
17",
   "Statement": [
    {
      "Sid": "",
       "Effect":
 "Allow",
       "Principal": {
         "Service":
 "ec2.amazonaws.com"
      },
       "Action":
 "sts:AssumeRole"
    }
  ]
}

Role Permissions:
{
```

タスク	説明	必要なスキル
	"Version": "2012-10-17", "Statement": [{ "Action": ["s3:ListBucket", "s3:GetBucketLocation"], "Resource": "arn:aws:s3:::", "Effect": "Allow" }, { "Action": ["s3:GetObject", "s3:PutObject", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload"], "Resource": "arn:aws:s3:::/*", "Effect": "Allow" }] }	

タスク	説明	必要なスキル
IAM インスタンスプロファイルを置き換えるには、手動 RFC を提出してください。	手動 RFC を提出して、ターゲット EC2 インスタンスを新しい IAM インスタンスプロファイルに関連付けます。	AWS システム管理者、AWS 開発者
S3 バケットへのコピー操作をテストします。	で次のコマンドを実行して、S3 バケットへのコピーオペレーションをテストします AWS CLI。 `aws s3 cp test.txt s3://<S3 bucket>/test2.txt`	AWS システム管理者、AWS 開発者

関連リソース

- 「Amazon EC2 インスタンス (CLI) の IAM インスタンスプロファイルを作成する」
- S3 バケットの作成 (Amazon S3 コンソール、 AWS SDKs、または を使用 AWS CLI)

Snowflake Snowpipe、Amazon S3、Amazon SNSAmazon SNS Data Firehose を使用して Snowflake データベースへのデータストリームの取り込みを自動化する

ビカシュ・チャンドラ・ラウト (AWS) によって作成されました

概要

このパターンでは、Amazon Web Services (AWS) クラウドのサービスを使用してデータの継続的なストリームを処理し、Snowflake データベースにロードする方法について説明します。このパターンでは、Amazon Data Firehose を使用してデータを Amazon Simple Storage Service (Amazon S3) に配信し、Amazon Simple Notification Service (Amazon SNS) を使用して新しいデータを受信したときに通知を送信し、Snowflake Snowpipe を使用してデータを Snowflake データベースにロードします。

このパターンに従うことで、継続的に生成されたデータを数秒で分析でき、複数の手動COPYコマンドを回避し、ロード時の半構造化データを完全にサポートできます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Firehose 配信ストリームに継続的にデータを送信しているデータソース。
- Firehose 配信ストリームからデータを受信している既存の S3 バケット。
- アクティブな Snowflake アカウント。

機能制限

- Snowflake Snowpipe は Firehose に直接接続しません。

アーキテクチャ

テクノロジースタック

- Amazon Data Firehose

- Amazon SNS
- Amazon S3
- Snowflake Snowflake Snowpipe
- Snowflake データレイク

ツール

- Amazon Data Firehose は、Amazon S3、Amazon Redshift、Amazon OpenSearch Service、Splunk、カスタム HTTP エンドポイント、またはサポートされているサードパーティーサービスプロバイダーが所有する HTTP エンドポイントなどの送信先にリアルタイムのストリーミングデータを配信するためのフルマネージドサービスです。
- Amazon Simple Storage Service (Amazon S3) は、インターネット用のストレージです。
- Amazon Simple Notification Service (Amazon SNS) は、サブスクライブしているエンドポイントやクライアントへのメッセージ配信や送信を調整および管理します。
- Snowflake – Snowflakeは、Software as a Service (SaaS) として提供される分析データウェアハウスです。
- Snowflake Snowpipe — Snowpipe は、スノーフレークの段階でファイルが利用可能になるとすぐにファイルからデータをロードします。

エピック

Snowflake Snowpipeをセットアップする

タスク	説明	必要なスキル
Snowflake で CSV ファイルの作成	Snowflake にサインインし、`CREATE FILE FORMAT` コマンドを実行して、指定されたフィールド区切り文字を持つ CSV ファイルを作成します。このコマンドおよびその他の Snowflake コマンドの詳細については、「追加情報」セクションを参照してください。	開発者

タスク	説明	必要なスキル
外部の Snowflake ステージを作成します。	CREATE STAGE コマンドを実行して、前に作成した CSV ファイルを参照する外部 Snowflake ステージを作成します。重要: S3 バケットの URL、 AWS アクセスキー、シー AWS クレットアクセスキーが必要です。SHOW STAGES コマンドを実行して、Snowflake ステージが作成されていることを確認します。	開発者
Snowflake ターゲットテーブルを作成します。	CREATE TABLE コマンドを実行して Snowflake テーブルを作成します。	開発者
パイプを作成します。	CREATE PIPE コマンドを実行し、 auto_ingest=true が コマンドにあることを確認します。SHOW PIPES コマンドを実行して、パイプが作成されていることを確認します。notification_channel 列値をコピーして保存します。この値は Amazon S3 イベント通知の設定に使用されます。	開発者

S3 バケットを設定する

タスク	説明	必要なスキル
S3 バケットの 30 日間のライフサイクルポリシーを作成します。	にサインイン AWS Management Console し、Amazon S3 コンソールを開きます。Firehose のデータを含む S3 バケットを選択します。次に、S3 バケットの管理タブを選択し、ライフサイクルルールの追加を選択します。[ライフサイクルルール] ダイアログボックスにルールの名前を入力し、バケットの 30 日間のライフサイクルルールを設定します。このストーリーやその他のストーリーに関するヘルプは、「関連リソース」セクションを参照してください。	システム管理者、開発者
S3 バケット用の IAM ポリシーを作成します。	AWS Identity and Access Management (IAM) コンソールを開き、ポリシーを選択します。[Create policy] (ポリシーの作成) を選択し、[JSON] タブを選択します。追加情報セクションからポリシーをコピーして JSON フィールドに貼り付けます。このポリシーは、 `PutObject` および `DeleteObject` アクセス許可と、、`GetObjectVersion` 、および `GetObject` `ListBucket` アクセス許可を付与しま	システム管理者、開発者

タスク	説明	必要なスキル
	す。ポリシーの確認を選択し、ポリシー名を入力し、ポリシーの作成を選択します。	
ポリシーを IAM ロールにアサインします。	IAM コンソールを開き、ロールを選択し、ロールの作成を選択します。信頼されたエンティティとして別の AWS アカウントを選択します。AWS アカウント ID を入力し、外部 ID を要求する を選択します。後で変更するプレースホルダー ID を入力します。Next を選択し、前に作成した IAM ポリシーを割り当てます。IAM ロールを作成します。	システム管理者、開発者
IAM ロールの Amazon リソースネーム (ARN) をメモします。	IAM コンソールを開き、ロールを選択します。前に作成した IAM ロールを選択し、ロール ARN をコピーして保存します。	システム管理者、開発者

Snowflakeでストレージインテグレーションをセットアップします。

タスク	説明	必要なスキル
Snowflake でクラウドストレージの統合を作成します。	Snowflake にサインインし、 `CREATE STORAGE INTEGRATION` コマンドを実行します。これにより、信頼関係が変更され、Snowflake へのアクセス権が付与さ	システム管理者、開発者

タスク	説明	必要なスキル
	れ、Snowflakeステージの外部 ID が提供されます。	
Snowflake アカウントの IAM ユーザーを取得します。	`DESC INTEGRATION` コマンドを実行して、IAM ロールの ARN を取得します。 **Important** `<integration_name>` は、前に作成した Snowflake ストレージ統合の名前です。	システム管理者、開発者
2 つの列の値を記録します。	列と `storage_aws_external_id` 列の値をコピー`storage_aws_iam_user_arn` して保存します。	システム管理者、開発者

Snowflake Snowpipeに S3 バケットへのアクセスを許可する

タスク	説明	必要なスキル
IAM ロールポリシーを変更します。	IAM コンソールを開き、[Roles] (ロール) を選択します。前に作成した IAM ロールを選択し、信頼関係タブを選択します。[Edit trust relationship (信頼関係の編集)] を選択します。を、前にコピーしたstorage_aws_external_id 値snowflake_external_id に置き換	システム管理者、開発者

タスク	説明	必要なスキル
	えます。を、前にコピーしたstorage_aws_iam_user_arn 値snowflake_user_arn に置き換えます。次に、信頼ポリシーの更新を選択します。	

S3 バケットの SNS 通知をオンにして設定する

タスク	説明	必要なスキル
S3 バケットのイベント通知をオンにします。	Amazon S3 コンソールを開き、 バケットを見つけます。プロパティを選択し、詳細設定 でイベントを選択します。通知を追加を選択し、このイベントの名前を入力します。名前を入力しない場合は、グローバル一意識別子 (GUID) が使用されます。	システム管理者、開発者
通知用に Amazon SNS トピックを設定する	イベント で ObjectCreate (All) を選択し、次に Send to ドロップダウンリストで SQS キューを選択します。SNS リストで、SQS キュー ARN を追加を選択し、前にコピーしたnotification_channel 値を貼り付けます。次に、[保存] を選択します。	システム管理者、開発者
Snowflake SQS キューを SNS トピックにサブスクライブする。	Snowflake SQS キューを作成した SNS トピックにサブスクライブします。このステップのヘルプについては、「関連	システム管理者、開発者

タスク	説明	必要なスキル
	リソース」セクションを参照してください。	

Snowflake ステージインテグレーションを確認してください。

タスク	説明	必要なスキル
Snowpipeをチェックしてテストしてください。	Snowflakeにサインインし、スノーフレークステージを開きます。S3 バケットにファイルをドロップし、Snowflake テーブルに読み込まれるかどうかを確認します。S3 バケットに新しいオブジェクトが表示されると、Amazon S3 は SNS 通知を Snowpipe に送信します。	システム管理者、開発者

関連リソース

- Managing your storage lifecycle
- Snowflake SQS キューを Amazon SNS トピックへサブスクライブします

追加情報

ファイル形式の作成：

```
CREATE FILE FORMAT <name>
TYPE = 'CSV'
FIELD_DELIMITER = '|'
SKIP_HEADER = 1;
```

外部ステージの作成：

```
externalStageParams (for Amazon S3) ::=
```

```
  URL = 's3://[//]'

  [ { STORAGE_INTEGRATION =  } | { CREDENTIALS = ( {  { AWS_KEY_ID = `` AWS_SECRET_KEY
 = `` [ AWS_TOKEN = `` ] } | AWS_ROLE = ``  } ) ) }` ]
  [ ENCRYPTION = ( [ TYPE = 'AWS_CSE' ] [ MASTER_KEY = '' ] |
                   [ TYPE = 'AWS_SSE_S3' ] |
                   [ TYPE = 'AWS_SSE_KMS' [ KMS_KEY_ID = '' ] |
                   [ TYPE = NONE ] )
```

テーブルの作成：

```
CREATE [ OR REPLACE ] [ { [ LOCAL | GLOBAL ] TEMP[ORARY] | VOLATILE } | TRANSIENT ]
 TABLE [ IF NOT EXISTS ]
  <table_name>
    ( <col_name> <col_type> [ { DEFAULT <expr>
                                | { AUTOINCREMENT | IDENTITY } [ ( <start_num> ,
 <step_num> ) | START <num> INCREMENT <num> ] } ]
                                 /* AUTOINCREMENT / IDENTITY supported only for numeric
 data types (NUMBER, INT, etc.) */
                               [ inlineConstraint ]
      [ , <col_name> <col_type> ... ]
      [ , outoflineConstraint ]
      [ , ... ] )
  [ CLUSTER BY ( <expr> [ , <expr> , ... ] ) ]
  [ STAGE_FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>'
                           | TYPE = { CSV | JSON | AVRO | ORC | PARQUET | XML }
 [ formatTypeOptions ] } ) ]
  [ STAGE_COPY_OPTIONS = ( copyOptions ) ]
  [ DATA_RETENTION_TIME_IN_DAYS = <num> ]
  [ COPY GRANTS ]
  [ COMMENT = '<string_literal>' ]
```

ショーステージ：

```
SHOW STAGES;
```

パイプを作成します：

```
CREATE [ OR REPLACE ] PIPE [ IF NOT EXISTS ]
  [ AUTO_INGEST = [ TRUE | FALSE ] ]
  [ AWS_SNS_TOPIC =  ]
  [ INTEGRATION = '' ]
```

```
  [ COMMENT = '' ]
  AS
```

パイプを表示：

```
SHOW PIPES [ LIKE '<pattern>' ]
           [ IN { ACCOUNT | [ DATABASE ] <db_name> | [ SCHEMA ] <schema_name> } ]
```

ストレージインテグレーションの作成：

```
CREATE STORAGE INTEGRATION <integration_name>
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = '<iam_role>'
  STORAGE_ALLOWED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/')
  [ STORAGE_BLOCKED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/') ]
```

例：

```
create storage integration s3_int
  type = external_stage
  storage_provider = s3
  enabled = true
  storage_aws_role_arn = 'arn:aws:iam::001234567890:role/myrole'
  storage_allowed_locations = ('s3://amzn-s3-demo-bucket1/mypath1/', 's3://amzn-s3-
demo-bucket2/mypath2/')
  storage_blocked_locations = ('s3://amzn-s3-demo-bucket1/mypath1/sensitivedata/',
 's3://amzn-s3-demo-bucket2/mypath2/sensitivedata/');
```

このステップに関する詳細については、Snowflake のドキュメントにある Amazon S3 にアクセスするための Snowflake ストレージ統合の設定を参照してください。

インテグレーションの説明：

```
DESC INTEGRATION <integration_name>;
```

S3 バケットポリシー

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:GetObjectVersion",
              "s3:DeleteObject",
              "s3:DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "/*"
                    ]
                }
            }
        }
    ]
}
```

既存および新規の Amazon EBS ボリュームを自動的に暗号化する

作成者: Tony DeMarco (AWS)、Josh Joy (AWS)

概要

Amazon Elastic Block Store (Amazon EBS) ボリュームの暗号化は、組織のデータ保護戦略にとって重要です。これは、アーキテクチャが適切に設計された環境を確立するための重要なステップです。暗号化されていない既存の EBS ボリュームまたはスナップショットを直接暗号化する方法はありませんが、新しいボリュームまたはスナップショットを作成することで暗号化できます。詳細については、Amazon EC2 ドキュメントの「EBS リソースの暗号化」を参照してください。このパターンにより、新規および既存の EBS ボリュームを暗号化するための予防的かつ検出的な制御が可能になります。このパターンでは、アカウント設定の設定、自動修復プロセスの作成、アクセス制御の実装を行います。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント
- macOS、Linux、または Windows にインストールおよび設定済みの AWS コマンドラインインターフェイス (AWS CLI)
- macOS、Linux、または Windows にインストールおよび設定されている jq
- AWS Identity and Access Management (IAM) アクセス許可は、AWS CloudFormation、Amazon Elastic Compute Cloud (Amazon EC2)、AWS Systems Manager、AWS Config、および AWS Key Management Service (AWS KMS) への読み取りおよび書き込みアクセス権を持つようにプロビジョニングされます
- AWS Organizations は、サービスコントロールポリシーの要件であるすべての機能を有効にして設定
- AWS Config はターゲットアカウントで有効

制限

- ターゲット AWS アカウントには、encrypted-volumes という名前の AWS Config ルールがあってはなりません。この解決策はこの名前のルールをデプロイします。この名前のルールが既に存在していると、デプロイが失敗し、同じルールを複数回処理することに関連して不要な料金が発生する可能性があります。

- この解決策では、すべての EBS ボリュームを同じ AWS KMS キーで暗号化します。
- アカウントの EBS ボリュームの暗号化を有効にする場合、この設定はリージョン固有になります。AWS リージョンに対して有効にした場合、そのリージョン内の個々のボリュームまたはスナップショットに対して無効にすることはできません。詳細については、Amazon EC2 のドキュメントで「デフォルトでの暗号化」を参照してください。
- 暗号化されていない既存の EBS ボリュームを修正するときは、EC2 インスタンスが使用されていないことを確認してください。この自動化によってインスタンスがシャットダウンされ、暗号化されていないボリュームがデタッチされ、暗号化されたボリュームがアタッチされます。修正中はダウンタイムが発生します。これが組織にとって重要なインフラストラクチャである場合は、インスタンスで実行されているアプリケーションの可用性に影響を与えないように、手動または自動の高可用性構成を実施してください。重要なリソースは、標準のメンテナンス時間帯にのみ修正することをお勧めします。

アーキテクチャ

自動化ワークフロー

1. AWS Config は暗号化されていない EBS ボリュームを検出します。
2. 管理者は AWS Config を使用して修復コマンドを Systems Manager に送信します。
3. Systems Manager の自動化は、暗号化されていない EBS ボリュームのスナップショットを作成します。
4. Systems Manager の自動化では、AWS KMS を使用してスナップショットの暗号化されたコピーを作成します。
5. Systems Manager の自動化は次のことを行います。
   a. 影響を受ける EC2 インスタンスが稼働している場合、そのインスタンスを停止します。
   b. ボリュームの新しい暗号化されたコピーを EC2 インスタンスにアタッチする
   c. EC2 インスタンスを元の状態に戻す

ツール

AWS サービス

- AWS CLI – AWS コマンドラインインターフェイス (AWS CLI) では、AWS サービスのパブリックアプリケーションプログラミングインターフェイス (API) に直接アクセスできます。AWS CLI を

使用してサービスの機能を調べ、シェルスクリプトを開発してリソースを管理できます。低レベルの同等の API コマンドに加えて、複数の AWS サービスでは AWS CLI のカスタマイズを提供します。カスタマイズには、複雑な API によるサービスの使用を簡略化する高レベルのコマンドが含まれます。

- AWS CloudFormation – AWS CloudFormation は、リソースをモデル化してセットアップするためのサービスです。必要なすべての AWS リソース (Amazon EC2 インスタンスなど) を説明するテンプレートを作成すると、CloudFormation がそれらのリソースをプロビジョニングして構成します。
- AWS Config は、AWS アカウントにおける AWS Config の設定を詳細に表示します。これには、リソース間の関係と設定の履歴が含まれるため、時間の経過と共に設定と関係がどのように変わるかを確認できます。
- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) は、ソフトウェアシステムの構築とホストに使用する、サイズ変更可能なコンピューティング容量を提供する Web サービスです。
- AWS KMS – AWS Key Management Service (AWS KMS) は、クラウド向けに拡張された暗号化およびキー管理サービスです。AWS KMS キーと機能は、他の AWS のサービスで使用され、それを使用して、AWS を使用するお客様の独自のアプリケーションのデータを保護できます。
- AWS Organizations – AWS Organizations は、作成して一元管理している複数の AWS アカウントを組織に統合するためのアカウント管理サービスです。
- AWS Systems Manager Automation – Systems Manager Automation は、Amazon EC2 インスタンスやその他の AWS リソースの一般的なメンテナンスとデプロイメントのタスクを簡素化します。

その他のサービス

- jq – jq は軽量で柔軟なコマンドライン JSON プロセッサです。このツールを使用して、AWS CLI 出力から重要な情報を抽出します。

コード

- このパターンのコードは、GitHub 内の「Automatically remediate unencrypted EBS Volumes using customer KMS keys」リポジトリで利用できます。

エピック

暗号化されていないボリュームの自動修正

タスク	説明	必要なスキル
スクリプトと CloudFormation テンプレートをダウンロードする。	GitHub からシェルスクリプト、JSON ファイル、および CloudFormation テンプレートをダウンロードします。顧客の KMS キーリポジトリを使用して、暗号化されていない EBS ボリュームを自動的に修正します。	AWS 管理者、AWS 全般
AWS KMS キーの管理者を特定する。	1. AWS マネジメントコンソールにサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。 2. AWS KMS キー管理者になるユーザーまたはロールを特定します。この目的で新しいユーザーまたはロールを作成する必要がある場合は、今すぐ作成してください。詳細については、IAM ドキュメントの「IAM ID」を参照してください。この自動化により、新しい AWS KMS キーが作成されます。 3. 識別されたら、ユーザーまたはロールの Amazon リソースネーム (ARN) をコピーします。詳細について	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	は、IAM ドキュメントの「IAM ARN」を参照してください。この ARN は次のステップで使用します。	

タスク	説明	必要なスキル
Stack1 CloudFormation テンプレートをデプロイする。	1. AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation/) を開きます。 2. CloudFormation で、`Stack1.yaml` テンプレートをデプロイします。次のデプロイメントの詳細に留意してください。 • スタックにはわかりやすい名前を付けてください。次のステップで必要になるため、スタック名を書きとめておきます。 • キー管理者の ARN を Stack1 の唯一のパラメータフィールドに貼り付けます。このユーザーまたはロールは、スタックによって作成された AWS KMS キーの管理者になります。 CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントの「AWS CloudFormation テンプレートの使用」を参照してください。	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
Stack2 CloudFormation テンプレートをデプロイする。	CloudFormation で、`Stack2.yaml` テンプレートをデプロイします。次のデプロイメントの詳細に留意してください。 • スタックにはわかりやすい名前を付けてください。 • Stack2 の唯一のパラメータには、前のステップで作成したスタックの名前を入力します。これにより、Stack2 は前のステップでスタックによってデプロイされた新しい AWS KMS キーとロールを参照できます。	AWS 管理者、AWS 全般
テスト用に暗号化されていないボリュームを作成する。	暗号化されていない EBS ボリュームで EC2 インスタンスを作成します。手順については、Amazon EC2 のドキュメントで「Amazon EBS ボリュームの作成」を参照してください。インスタンスタイプは関係なく、インスタンスへのアクセスも必要ありません。t2.micro インスタンスを作成して無料利用枠のままにしておくことができ、key pair を作成する必要はありません。	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
AWS Config ルールをテストする。	1. https://console.aws.amazon.com/config/ で AWS Config コンソールを開きます。[ルール] ページで、encrypted-volumes ルールを選択します。 2. 暗号化されていない新しいテストインスタンスが非準拠リソースのリストに表示されていることを確認します。ボリュームがすぐに表示されない場合は、数分待ってから結果を更新します。AWS Config ルールは、インスタンスとボリュームが作成された直後にリソースの変更を検出します。 3. リソースを選択し、[修復] を選択します。 Systems Manager では、修復の進行状況とステータスを次のように表示できます。 1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/) を開きます。 2. ナビゲーションペインで [オートメーション] を選択します。	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	3. Execution ID リンクを選択すると、手順とステータスが表示されます。	
追加のアカウントまたは AWS リージョンを設定する。	ユースケースの必要に応じて、追加のアカウントや AWS リージョンについてもこのエピックを繰り返してください。	AWS 管理者、AWS 全般

EBS ボリュームのアカウントレベルの暗号化を有効にする

タスク	説明	必要なスキル
有効スクリプトを実行する。	1. Bash シェルで、cd コマンドを使用してクローンされたリポジトリに移動します。 2. 次のコマンドを入力して enable-ebs-encryption-for-account スクリプトを実行します。 `./Bash/enable-ebs-encryption-for-account.sh`	AWS 管理者、AWS 全般、bash
設定が更新されたことを確認する。	1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	2. 画面の右側にある設定で、データ保護とセキュリティを選択します。 3. EBS 暗号化セクションで、常に新しい EBS ボリュームの暗号化が有効になっていること、およびデフォルトの暗号化キーが前に指定した ARN に設定されていることを確認します。 **Note** [新しい EBS ボリュームを常に暗号化する] 設定がオフになっているか、キーが alias/aws/ebs に設定されたままの場合は、シェルスクリプトを実行したのと同じアカウントと AWS リージョンにログインしていることを確認し、シェルにエラーメッセージがないか確認してください。	

タスク	説明	必要なスキル
追加のアカウントまたは AWS リージョンを設定する。	ユースケースの必要に応じて、追加のアカウントや AWS リージョンについてもこのエピックを繰り返してください。	AWS 管理者、AWS 全般

暗号化されていないインスタンスの作成を防ぐ

タスク	説明	必要なスキル
サービスコントロールポリシーを作成する。	1. https://console.aws.amazon.com/organizations/v2/ で AWS Organizations コンソールを開きます。 2. 新しいサービスコントロールポリシーを作成します。詳細については、AWS Organizations ドキュメントの「サービスコントロールポリシーの作成、更新、削除」を参照してください。 3. `DenyUnencryptedEC2.json` の内容をポリシーに追加して保存します。この JSON ファイルは、最初のエピックで GitHub リポジトリからダウンロードしました。 4. このポリシーを組織ルートまたは必要な組織単位 (OU) に添付します。詳細については、AWS Organizations	AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	ドキュメントの「サービスコントロールポリシーのアタッチとデタッチ」を参照してください。	

関連リソース

AWS サービスのドキュメント

- AWS CLI
- AWS Config
- AWS CloudFormation
- Amazon EC2
- AWS KMS
- AWS Organizations
- AWS Systems Manager Automation

その他のリソース

- jq マニュアル (jq ウェブサイト)
- jq ダウンロード (GitHub)

AWS クラウド上の Stromasys Charon-SSP エミュレーターで Sun SPARC サーバーをバックアップ

ケビン・ヤン（AWS）、ルイス・ラモス（Stromasys）、ロヒト・ダルジ（AWS）によって作成されました

概要

このパターンでは、オンプレミス環境からAmazon Web Services (AWS) クラウドへの移行後に Sun Microsystems SPARC サーバーをバックアップするための 4 つのオプションが用意されています。これらのバックアップオプションは、組織の目標復旧時点 (RPO) と目標復旧時間 (RTO) を満たし、自動化されたアプローチを採用し、全体的な運用コストを削減するバックアップ計画を実装するのに役立ちます。このパターンは、4 つのバックアップオプションの概要と実装手順を示しています。

「Stromasys Charon-SSP エミュレータでゲスト」としてホストされている Sun SPARC サーバーを使用する場合、次の 3 つのバックアップオプションのいずれかを使用できます。

- Backup オプション1: Stromasys仮想テープ — Charon-SSP仮想テープ特徴量を使用してSun SPARCサーバーにバックアップ設備を設定し、「AWS Systems Manager Automation」 を使用してバックアップファイルをAmazonSimple 「Amazon Simple Storage Service (Amazon S3)」「Storage Service Glacier」 にアーカイブします。
- Backup オプション2: Stromasysスナップショット — Charon-SSPスナップショット特徴量を使用して、Charon-SSP内のSun SPARCゲストサーバーのバックアップファシリティを設定します。
- Backup オプション 3: Amazon Elastic Block Store (Amazon EBS) ボリュームスナップショット — Amazon Elastic Compute Cloud (Amazon EC2) で Charon-SSP エミュレーターをホストしている場合、「Amazon EBS ボリュームスナップショット」を使用して Sun SPARC ファイルシステムのバックアップを作成できます。

ハードウェア上でゲストとしてホストされているSun SPARCサーバーとAmazon EC2上のCharon-SSPを使用する場合、以下のバックアップオプションを使用できます。

- Backup オプション 4: AWS Storage Gateway 仮想テープライブラリ (VTL) — 「Storage Gateway」VTL テープゲートウェイを備えたバックアップアプリケーションを使用して、Sun SPARC サーバーをバックアップします。

Sun SPARC サーバーのブランドゾーンとしてホストされている Sun SPARC サーバーを使用する場合は、バックアップオプション 1、2、4 を使用できます。

「Stromasys」は、従来の SPARC、Alpha、VAX、および PA-RISC の重要なシステムをエミュレートするためのソフトウェアとサービスを提供しています。Stromasys エミュレーションを使用して AWS クラウドに移行する方法の詳細については、AWS ブログの 「Stromasys による SPARC、Alpha、またはその他のレガシーシステムの AWS へのリホスト」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 既存の Sun SPARC サーバー。
- Charon-SSP の既存のライセンス。Charon-SSPのライセンスはAWS Marketplace から、ストロマシス仮想環境 (VE) のライセンスはストロマシスから入手できます。詳細については、「ストロマシスの営業担当」にお問い合わせください。
- Sun SPARC サーバーと Linux バックアップに精通していること。
- Charon-SSP エミュレーションテクノロジーに精通していること。これについて詳しくは、Stromasys ドキュメンテーションの「Stromasys レガシーサーバーエミュレーション」を参照してください。
- Sun SPARC サーバーのファイルシステムに仮想テープ機能またはバックアップアプリケーションを使用する場合は、Sun SPARC サーバーファイルシステムのバックアップ機能を作成して設定する必要があります。
- RPO と RTO について理解していること。詳細については、AWS Well-Architected フレームワーク ドキュメントの 「信頼性の柱」 ホワイトペーパーの 「ディザスタリカバリ」を参照してください。
- Backup オプション 4 を使用するには、次のものが必要です。
  - Storage ゲートウェイ VTL テープゲートウェイをサポートするソフトウェアベースのバックアップアプリケーション。詳細については、AWS Storage Gateway ドキュメントの「VTL デバイスの操作」を参照してください。
  - Bacula Director または類似のバックアップアプリケーションがインストールされ、設定されている。詳細については、「Bacula ディレクター」を参照してください。

次の表に、このパターンの 4 つのバックアップオプションについての情報を示します。

バックアップオプション	Crash-consistent を実現できるか?	アプリケーションの一貫性を実現できるか?	仮想バックアップアプライアンスソリューション?	一般的なユースケース
オプション 1 — Stromasys 仮想テープ	可能 Sun SPARC ファイルシステムのスナップショットを自動化して、仮想テープにデータをバックアップできます。例えば、UFS または ZFS スナップショットを使用できます。	可能 このバックアップオプションには、処理中のトランザクションをフラッシュしたり、ファイルシステムのスナップショット中に読み取り専用または一時的なオフラインモードを設定したり、アプリケーションデータダンプを実行したりするための自動スクリプトが必要です。また、アプリケーションのダウンタイムまたは読み取り専用モードが必要になる場合があります。	可能	Sun SPARC サーバーのファイルシステムを.tar ファイルまたは.zip ファイルでバックアップします。 アプリケーションデータのバックアップ
オプション 2 — Stromasys スナップショット	可能 この特徴量を有効にするには、	可能 このバックアップオプションで	いいえ	Sun SPARC サーバースナップショット

「Charon-SSP Manager」を設定するか、コマンドラインのスタートアップ引数を使用する必要があります。 また、Linuxコマンドを実行して、Sun SPARCゲストサーバーの状態をスナップショットファイルに保存するようにCharon-SSPエミュレータに指示する必要があります。 **Important** Sun SPARCゲストサーバーをシャットダウンする必要があります。	は、仮想ディスクとメモリーダンプを含む、エミュレートされたゲストサーバーのスナップショットが作成されます。 **Important** スナップショットの作成中に Sun SPARC ゲストサーバーをシャットダウンする必要があります。	アプリケーションデータのバックアップ

オプション 3 — Amazon EBS ボリュームスナップショット	可能 AWS Backup を使用して Amazon EBS スナップショットを自動化できます。	可能 このバックアップオプションでは、処理中のトランザクションをフラッシュし、Amazon EBS ボリュームスナップショット中に EC2 インスタンスを読み取り専用または一時的に停止するように設定する自動スクリプトが必要です。 **Important** このバックアップオプションでは、アプリケーションの一貫性を実現するために、アプリケーションのダウンタイムまたは読み取り専用モードが	いいえ	Sun SPARC サーバーファイルシステムスナップショット アプリケーションデータのバックアップ

必要になる場合があります。

オプション 4 — AWS Storage Gateway VTL	可能 バックアップエージェントを使用して Sun SPARC ファイルシステムのバックアップデータを VTL に自動的にバックアップできます。	可能 このバックアップオプションでは、ファイルシステムのスナップショットまたはアプリケーションデータダンプ中に、処理中のトランザクションをフラッシュし、読み取り専用または一時的なオフラインモードを設定する自動スクリプトが必要です。 ⚠ Important このバックアップオプションでは、アプリケーションのダウンタイムまたは読み取り専用モードが必要になる場合	可能	大きなフリートの Sun SPARC サーバーのファイルシステムの大規模なバックアップ アプリケーションデータのバックアップ

があります。

制約事項

- このパターンの方法を使用して個々の Sun SPARC サーバーをバックアップできますが、クラスター内で動作するアプリケーションがある場合は、これらのバックアップオプションを共有データに使用することもできます。

ツール

Backup オプション 1: Stromasys 仮想テープ

- 「Stromasys Charon-SSPエミュレーター」 — Charon-SSPエミュレーターは、標準の64ビットx86互換コンピューターシステム内にオリジナルのSPARCハードウェアの仮想レプリカを作成します。SunOS や Solaris などのオペレーティングシステム (OS)、その階層化製品、アプリケーションなど、元の SPARC バイナリコードを実行します。
- 「Amazon EC2」 – Amazon Elastic Compute Cloud (Amazon EC2) は、ソフトウェアシステムを構築してホストするための、自在に拡張および縮小できるコンピューティング能力を提供するウェブサービスです。
- 「Amazon EFS」– Amazon Elastic File System (Amazon EFS) は、AWS クラウドサービスとオンプレミスリソースで使用できる、シンプルでサーバーレスな、一度設定すればあとは作業不要の伸縮自在なファイルシステムです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。
- 「Amazon S3 Glacier」– Amazon Simple Storage Service Glacierは、データのアーカイブと長期バックアップ用の、安全で耐久性が高く、非常に低コストの Amazon S3 ストレージクラスです。
- 「AWS Systems Manager Automation」 — オートメーションは、AWS Systems Manager の一機能であり、EC2 インスタンスおよび他の AWS リソースの一般的なメンテナンスとデプロイのタスクを簡素化します。

Backup オプション 2: Stromasys スナップショット

- 「Stromasys Charon-SSPエミュレーター」— Charon-SSPエミュレーターは、標準の64ビットx86互換コンピューターシステム内にオリジナルのSPARCハードウェアの仮想レプリカを作成します。SunOS や Solaris などの OS、それらのレイヤード製品、アプリケーションを含むオリジナルの SPARC バイナリコードを実行します。
- 「Amazon EC2」– Amazon Elastic Compute Cloud (Amazon EC2) は、ソフトウェアシステムを構築してホストするための、自在に拡張および縮小できるコンピューティング能力を提供するウェブサービスです。
- 「Amazon EFS」– Amazon Elastic File System (Amazon EFS) は、AWS クラウドサービスとオンプレミスリソースで使用できる、シンプルでサーバーレスな、一度設定すればあとは作業不要の伸縮自在なファイルシステムです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。
- 「Amazon S3 Glacier」– Amazon Simple Storage Service Glacierは、データのアーカイブと長期バックアップ用の、安全で耐久性が高く、非常に低コストの Amazon S3 ストレージクラスです。
- 「AWS Systems Manager Automation」— オートメーションは、AWS Systems Manager の一機能であり、EC2 インスタンスおよび他の AWS リソースの一般的なメンテナンスとデプロイのタスクを簡素化します。

Backup オプション 3: Amazon EBS ボリュームスナップショット

- 「Stromasys Charon-SSPエミュレーター」— Charon-SSPエミュレーターは、標準の64ビットx86互換コンピューターシステム内にオリジナルのSPARCハードウェアの仮想レプリカを作成します。SunOS や Solaris などの OS、それらのレイヤード製品、アプリケーションを含むオリジナルの SPARC バイナリコードを実行します。
- 「AWS Backup」– AWS Backupは、フルマネージド型のデータ保護サービスで、クラウド内、およびオンプレミス間で簡単に一元化およびを自動化できます。
- 「Amazon EBS」– Amazon Elastic Block Store (Amazon EBS)は、EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。
- 「Amazon EC2」– Amazon Elastic Compute Cloud (Amazon EC2) は、ソフトウェアシステムを構築してホストするための、自在に拡張および縮小できるコンピューティング能力を提供するウェブサービスです。

Backup オプション 4: AWS Storage Gateway VTL

- 「Stromasys Charon-SSPエミュレーター」— Charon-SSPエミュレーターは、標準の64ビットx86互換コンピューターシステム内にオリジナルのSPARCハードウェアの仮想レプリカを作成します。SunOS や Solaris などの OS、それらのレイヤード製品、アプリケーションを含むオリジナルの SPARC バイナリコードを実行します。
- 「Bacula — Bacula」 は、オープンソースのエンタープライズレベルのコンピューターバックアップシステムです。既存のバックアップアプリケーションがテープゲートウェイをサポートしているかどうかの詳細については、AWS Storage Gateway ドキュメントの 「テープゲートウェイでサポートされているサードパーティバックアップアプリケーション」 を参照してください。
- 「Amazon EC2」– Amazon Elastic Compute Cloud (Amazon EC2) は、ソフトウェアシステムを構築してホストするための、自在に拡張および縮小できるコンピューティング能力を提供するウェブサービスです。
- 「Amazon RDS for MySQL」— Amazon Relational Database Service (Amazon RDS) は、MySQL の複数のバージョンを実行する DB インスタンスをサポートします。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。
- 「Amazon S3 Glacier」– Amazon Simple Storage Service Glacierは、データのアーカイブと長期バックアップ用の、安全で耐久性が高く、非常に低コストの Amazon S3 ストレージクラスです。
- AWS Storage Gateway - Storage Gateway は、オンプレミスのソフトウェアアプライアンスをクラウドベースのストレージと接続し、お客様のオンプレミスの IT 環境と AWS のストレージインフラストラクチャとの間にデータセキュリティ機能を備えたシームレスな統合を実現するサービスです

エピック

Backup オプション 1 — Stromasys 仮想テープバックアップを作成する

タスク	説明	必要なスキル
仮想テープファイルストレージ用の Amazon EFS 共有ファイルシステムを作成します。	AWS マネジメントコンソールにサインインするか、AWS CLI を使用して Amazon EFS ファイルシステムを作成します。	クラウドアーキテクト

タスク	説明	必要なスキル
	この件の詳細については、Amazon EFS ドキュメントの「Amazon EFS ファイルシステムの作成」を参照してください。	
共有ファイルシステムをマウントするように Linux ホストを設定します。	Amazon EC2 Linux インスタンスに Amazon EFS ドライバーをインストールし、起動時に Amazon EFS 共有ファイルシステムをマウントするように Linux OS を設定します。 詳細については、Amazon EFS ドキュメントの「EFS マウントヘルパーを使用したファイルシステムのマウント」を参照してください。	DevOps エンジニア
Charon-SSP エミュレーターをインストールします。	Amazon EC2 Linux インスタンスに Charon-SSP エミュレーターをインストールします。 詳細については、Stromasys ドキュメントの「Charon-SSP 用のAWS Sクラウドインスタンスのセットアップ」を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル
Sun SPARC ゲストサーバーごとに、共有ファイルシステムに仮想テープファイルコンテナーを作成します。	`touch <vtape-container-name>` コマンドを実行して、Charon-SSP エミュレータにデプロイされた各 Sun SPARC ゲストサーバーの共有ファイルシステムに仮想テープファイルコンテナを作成します。	DevOps エンジニア
Sun SPARCゲストサーバー用の仮想テープデバイスを作成するようにCharon-SSPマネージャーを設定します。	Charon-SSP Managerにログインして仮想テープデバイスを作成し、各Sun SPARCゲストサーバーの仮想テープコンテナファイルを使用するように設定します。 これについて詳しくは、「Stromasysのドキュメントにある Charon-SSP 5.2」 for Linuxユーザーガイドを参照してください。	DevOps エンジニア
仮想テープデバイスが Sun SPARC ゲストサーバーで使用できることを確認します。	Sun SPARC の各ゲストサーバーにログインし、`mt -f /dev/rmt/1` コマンドを実行して、仮想テープデバイスが OS で設定されていることを確認します。	DevOps エンジニア

タスク	説明	必要なスキル
Systems Manager Automation ランブックとオートメーションを作成します。	Systems Manager 自動化ランブックを作成し、バックアッププロセスをスケジュールするためのメンテナンスウィンドウとアソシエーションを Systems Manager で設定します。 詳細については、AWS Systems Manager ドキュメントの「自動化チュートリアル」と「メンテナンスウィンドウの設定」を参照してください。	クラウドアーキテクト
ローテーションされた仮想テープコンテナファイルをアーカイブするように Systems Manager Automation を設定します。	追加情報セクションの戻るオプション 1 のコードサンプルを使用して、ローテーションされた仮想テープコンテナファイルを Amazon S3 と Amazon S3 Glacier にアーカイブするための Systems Manager 自動化ランブックを作成します。	クラウドアーキテクト
Systems Manager Automation ランブックをデプロイして、アーカイブとスケジューリングを行います。	Systems Manager オートメーションランブックをデプロイし、Systems Manager で自動的に実行されるようにスケジュールします。 この件の詳細は、Systems Manager のドキュメントで「オートメーションチュートリアル」を参照してください。	クラウドアーキテクト

Backup オプション 2 — Stromasys スナップショットを作成する

タスク	説明	必要なスキル
仮想テープファイルストレージ用の Amazon EFS 共有ファイルシステムを作成します。	AWS マネジメントコンソールにサインインするか、AWS CLI を使用して Amazon EFS ファイルシステムを作成します。 詳細については、Amazon EFS ドキュメントの「Amazon EFS ファイルシステムの作成」を参照してください。	クラウドアーキテクト
共有ファイルシステムをマウントするように Linux ホストを設定します。	Amazon EC2 Linux インスタンスに Amazon EFS ドライバーをインストールし、スタートアップ時に Amazon EFS 共有ファイルシステムをマウントするように Linux OS を設定します。 詳細については、Amazon EFS ドキュメントの「EFS マウントヘルパーを使用したファイルシステムのマウント」を参照してください。	DevOps エンジニア
Charon-SSP エミュレーターをインストールします。	Amazon EC2 Linux インスタンスに Charon-SSP エミュレーターをインストールします。 詳細については、Stromasys ドキュメントの「Charon-SSP 用のAWS Sクラウドインスタ	DevOps エンジニア

タスク	説明	必要なスキル
	ンスのセットアップ」を参照してください。	
Sun SPARC ゲストサーバーをスナップショットオプションで起動するように設定します。	Charon-SSP マネージャを使用して、Sun SPARC ゲストサーバーごとにスナップショットオプションを設定します。 これについて詳しくは、「Stromasysのドキュメントにあるcharon-SSP 5.2」for Linuxユーザーガイドを参照してください。	DevOps エンジニア
Systems Manager Automation ランブックを作成します。	追加情報セクションのBackup オプション 2 のコードサンプルを使用して、メンテナンス時間中に Sun SPARC ゲストサーバーでスナップショットコマンドをリモートで実行するための Systems Manager Automation Runbook を作成します。	クラウドアーキテクト

タスク	説明	必要なスキル
Systems Manager 自動化ランブックをデプロイし、Amazon EC2 Linux ホストとの関連付けを設定します。	システムマネージャー自動化ランブックをデプロイし、バックアッププロセスをスケジュールするためのメンテナンスウィンドウとアソシエーションをシステムマネージャーでセットアップします。 詳細については、AWS Systems Manager ドキュメントの「自動化チュートリアル」と「メンテナンスウィンドウの設定」を参照してください。	クラウドアーキテクト
スナップショットを長期保存にアーカイブします。	追加情報セクションのランブックサンプルコードを使用して、スナップショットファイルを Amazon S3 と Amazon S3 Glacier にアーカイブするための Systems Manager 自動化ランブックを作成します。	クラウドアーキテクト

バックアップオプション 3 – Amazon EBS ボリュームスナップショットを作成

タスク	説明	必要なスキル
Charon-SSP エミュレーターをインストールします。	Amazon EC2 Linux インスタンスに Charon-SSP エミュレーターをインストールします。 詳細については、Stromasys ドキュメントの「Charon-SSP	DevOps エンジニア

タスク	説明	必要なスキル
	用のAWS Sクラウドインスタンスのセットアップ」を参照してください。	
Sun SPRAC ゲストサーバー用の EBS ボリュームを作成します。	AWS マネジメントコンソールにサインインし、Amazon EBS コンソールを開いて、Sun SPRAC ゲストサーバー用の EBS ボリュームを作成します。 詳細については、Stromasys ドキュメントの「Charon-SSP用のAWS Sクラウドインスタンスのセットアップ」を参照してください。	クラウドアーキテクト
Amazon EC2 Linux インスタンスに EBS ボリュームをアタッチします。	Amazon EC2 コンソールで、EBS ボリュームを Amazon EC2 Linux インスタンスにアタッチします。 詳細については、Amazon EC2 ドキュメントの「インスタンスに Amazon EBS ボリュームのアタッチ」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
EBS ボリュームを Charon-SSP エミュレーターで SCSI ドライブとしてマッピングします。	EBS ボリュームを Sun SPARC ゲストサーバーの SCSI ドライブとしてマッピングするように、Charon-SSP マネージャーを設定します。 これに関する詳細は、「Stromasysドキュメンテーション のCharon-SSP」 V5.2 for Linuxガイドの SCSIストレージ設定セクションを参照してください。	AWS DevOps
EBS ボリュームのスナップショットを作成するための AWS Backup スケジュールを設定します。	EBS ボリュームのスナップショットを作成するための AWS Backup ポリシーとスケジュールを設定します。 詳細については、AWS Developer Center ドキュメントの「AWS Backup を使用した Amazon EBS のバックアップと復元」チュートリアルを参照してください。	AWS DevOps

Backup オプション 4 — AWS Storage Gateway VTL の作成

タスク	説明	必要なスキル
テープゲートウェイデバイスを作成します。	AWS マネジメントコンソールにサインインし、AWS Storage Gateway コンソールを開いて、VPC にテープゲートウェイデバイスを作成します。	クラウドアーキテクト

タスク	説明	必要なスキル
	詳細については、AWS Storage Gateway ドキュメントの「ゲートウェイの作成」を参照してください。	
Bacula カタログの Amazon RDS DB インスタンスを作成します。	Amazon RDS コンソールを開き、Amazon RDS for MySQL DB インスタンスを作成します。 この件の詳細については、Amazon RDS ドキュメントの「MySQL DB インスタンスを作成して MySQL DB インスタンス上のデータベースに接続する」を参照してください。	クラウドアーキテクト
バックアップアプリケーションコントローラを VPC にデプロイします。	EC2 インスタンスに Bacula をインストールし、バックアップアプリケーションコントローラをデプロイし、テープゲートウェイデバイスに接続するようにバックアップストレージを設定します。`Bacula-storage-daemon-config.txt` ファイル (添付) にある Bacula Director ストレージデーモン設定のサンプルを使用できます。 詳細については、「Bacula ディレクター」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
Sun SPARC ゲストサーバーにバックアップアプリケーションを設定します。	`SUN-SPARC-Guest-Bacula-Config.txt` ファイル (添付) のサンプル Bacula 設定を使用して、Sun SPARC ゲストサーバにバックアップアプリケーションをインストールしてセットアップするように 2 台目のクライアントを設定します。	DevOps エンジニア
バックアップ構成とスケジュールを設定します。	`Bacula-Director-Config.txt` ファイル (添付) にあるサンプルの Bacula Director 設定を使用して、バックアップアプリケーションコントローラにバックアップ設定とスケジュールを設定します。 詳細については、「Bacula ディレクター」を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル
バックアップ構成とスケジュールが正しいことを検証します。	「Bacula のドキュメント」の指示に従って、Sun SPARC ゲストサーバーでのセットアップの検証とバックアップテストを行います。 例えば、次のコマンドを使用して構成ファイルを検証できます。 • `bacula-dir -t -c bacula-dir.conf` • `bacula-fd -t -c bacula-fd.conf` • `bacula-sd -t -c bacula-sd.conf`	DevOps エンジニア

関連リソース

- 「チャロン仮想SPARC (VEライセンス付き) 」
- 「チャロン-仮想-SPARC」
- 「Bacula エンタープライズエディションでのクラウドサービスとオブジェクトストレージの使用」
- 「ディザスタリカバリ (DR) 目標」
- 「Charon のレガシーシステムエミュレーションソリューション」

追加情報

Backup オプション 1 — Stromasys 仮想テープの作成

次の Systems Manager Automation ランブックコードのサンプルを使用すると、自動的にバックアップを開始してからテープを交換できます。

```
...
```

```
# example backup script saved in SUN SPARC Server
 #!/usr/bin/bash
 mt -f  rewind
 tar -cvf
 mt -f  offline
...
         mainSteps:
         - action: aws:runShellScript
           name:
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               # Validate tape backup container file exists
               if [ ! -f {{TapeBackupContainerFile}} ]; then
                 logger -s -p local3.warning "Tape backup container file is not exists
 - {{TapeBackupContainerFile}}, create a new one"
                 touch {{TapeBackupContainerFile}}
               fi
         - action: aws:runShellScript
           name: startBackup
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               user={{BACKUP_USER}}
               keypair={{KEYPAIR_PATH}}
               server={{SUN_SPARC_IP}}
               backup_script={{BACKUP_SCRIPT}}
               ssh -i $keypair $user@$server -c "/usr/bin/bash $backup_script"
         - action: aws:runShellScript
           name: swapVirtualDiskContainer
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               mv {{TapeBackupContainerFile}} {{TapeBackupContainerFile}}.$(date +%s)
               touch {{TapeBackupContainerFile}}
         - action: aws:runShellScript
           name: uploadBackupArchiveToS3
           inputs:
```

```
            onFailure: Abort
            timeoutSeconds: "1200"
            runCommand:
            - |
              aws s3 cp {{TapeBackupContainerFile}} s3://{{BACKUP_BUCKET}}/
{{SUN_SPARC_IP}}/$(date '+%Y-%m-%d')/
 ...
```

Backup オプション 2: Stromasys スナップショット

次の Systems Manager 自動化ランブックコードのサンプルを使用して、バックアッププロセスを自動化できます。

```
     ...

        mainSteps:
        - action: aws:runShellScript
          name: startSnapshot
          inputs:
            onFailure: Abort
            timeoutSeconds: "1200"
            runCommand:
            - |
              # You may consider some graceful stop of the application before taking a
 snapshot
              # Query SSP PID by configuration file
              # Example: ps ax | grep ssp-4 | grep Solaris10.cfg | awk '{print $1"
"$5}' | grep ssp4 | cut -f1 -d" "
              pid=`ps ax | grep ssp-4 | grep {{SSP_GUEST_CONFIG_FILE}} | awk '{print
$1" "$5}' | grep ssp4 | cut -f1 -d" "`
              if [ -n "${pid}" ]; then
                kill -SIGTSTP ${pid}
              else
                echo "No PID found for SPARC guest with config
 {{SSP_GUEST_CONFIG_FILE}}"
                exit 1
              fi
        - action: aws:runShellScript
          name: startBackup
          inputs:
            onFailure: Abort
            timeoutSeconds: "1200"
            runCommand:
```

```
              - |
                # upload snapshot and virtual disk files into S3
                aws s3 sync {{SNAPSHOT_FOLDER}} s3://{{BACKUP_BUCKET}}/$(date '+%Y-%m-
%d')/
                aws s3 cp {{VIRTUAL_DISK_FILE}} s3://{{BACKUP_BUCKET}}/$(date '+%Y-%m-
%d')/
          - action: aws:runShellScript
            name: restratSPARCGuest
            inputs:
              onFailure: Abort
              timeoutSeconds: "1200"
              runCommand:
              - |
                /opt/charon-ssp/ssp-4u/ssp4u -f {{SSP_GUEST_CONFIG_FILE}} -d -a
 {{SPARC_GUEST_NAME}} --snapshot {{SNAPSHOT_FOLDER}}
 ...
```

Backup オプション 4: AWS Storage Gateway VTL

Solaris 非大域ゾーンを使用して仮想化されたレガシー Sun SPARC サーバーを実行する場合、Sun SPARC サーバーで実行されている非大域ゾーンにもバックアップアプリケーション手法を適用できます (たとえば、バックアップクライアントは非大域ゾーン内で実行できます)。ただし、バックアップクライアントを Solaris ホストで実行して、非大域ゾーンのスナップショットを作成することもできます。その後、そのスナップショットはテープにバックアップできます。

次の設定例では、Solaris 非大域ゾーンをホストするファイルシステムを Solaris ホストのバックアップ設定に追加しています。

```
FileSet {
   Name = "Branded Zones"
   Include {
     Options {
       signature = MD5
     }
     File = /zones
   }
 }
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Veeam Backup & Replication を使用してデータを Amazon S3 にバックアップおよびアーカイブする

作成者: Jeanna James (AWS)、Anthony Fiore (AWS) (AWS)、William Quigley (AWS)

概要

このパターンは、Veeam Backup & Replicationによって作成されたバックアップを、Veeamのスケールアウトバックアップリポジトリ機能を使用して、サポートされている Amazon Simple Storage Service (Amazon S3) のオブジェクトストレージクラスに送信するプロセスを詳しく説明しています。

Veeam は、お客様固有のニーズに最適な複数の Amazon S3 ストレージクラスをサポートしています。ストレージのタイプは、バックアップまたはアーカイブデータのデータアクセス、耐障害性、コスト要件に基づいて選択できます。たとえば、30 日以上使用する予定のないデータを Amazon S3 の低頻度アクセス (IA) に保存して、コストを抑えることができます。データを 90 日以上アーカイブする予定の場合は、Amazon Simple Storage Service Glacier （ Amazon S3 Glacier Flexible Retrieval ） のフレキシブルリトリーブや、Veeam のアーカイブ階層による S3 Glacier Deep Archive を使用できます。S3 オブジェクトロックを使用して、Amazon S3 内で変更不可能なバックアップを作成することもできます。

このパターンでは、テープゲートウェイを使用して Veeam Backup & Replication を設定する方法について説明していません AWS Storage Gateway。このトピックについては、Veeamウェブサイトの「AWS VTL ゲートウェイを使用した Veeam の Backup とレプリケーション-デプロイガイド」 を参照してください。

> **Warning**
>
> このシナリオでは、プログラムによるアクセスと長期的な認証情報を持つ AWS Identity and Access Management (IAM) ユーザーが必要です。これはセキュリティリスクをもたらします。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。アクセスキーは、必要に応じて更新できます。詳細については、「IAM ユーザーガイド」の「アクセスキーの更新」を参照してください。

前提条件と制限

前提条件

- Veeam Availability SuiteまたはVeeam Backup Essentialsを含むVeeamBackup &レプリケーションがインストールされています（「無料トライアル」に登録できます）
- Veeam ユニバーサルライセンス（VUL）を含む、エンタープライズまたはエンタープライズプラス機能を備えた Veeam Backup &レプリケーションライセンス
- Amazon S3 バケットにアクセスできるアクティブな IAM ユーザー
- アーカイブ階層を使用している場合、Amazon Elastic Compute Cloud (Amazon EC2) と Amazon Virtual Private Cloud (Amazon VPC) にアクセスできるアクティブな IAM ユーザー
- パブリックインターネット接続またはパブリック AWS Direct Connect 仮想インターフェイス (VIF) を介したトラフィックのバックアップと復元に使用できる帯域幅 AWS のサービス で、オンプレミスから へのネットワーク接続
- オブジェクトストレージリポジトリと正しく通信できるように、以下のネットワークポートとエンドポイントが開かれました。
  - Amazon S3 ストレージ — TCP — ポート 443: Amazon S3 ストレージとの通信に使用されます。
  - Amazon S3 ストレージ - クラウドエンドポイント - AWS リージョン および `*.amazonaws.com`用 AWS GovCloud (US) Regions、または中国リージョン`*.amazonaws.com.cn`用: Amazon S3 ストレージとの通信に使用されます。接続エンドポイントの完全なリストについては、 AWS ドキュメントのAmazon S3 エンドポイント」を参照してください。
  - Amazon S3 ストレージ — TCP HTTP — ポート 80: 証明書のステータスを確認するために使用されます。証明書検証エンドポイント (証明書失効リスト (CRL) の URL と Online Certificate Status Protocol (OCSP) サーバー) は変更される可能性があることを考慮します。実際のアドレス一覧は、証明書自体に記載されています。
  - Amazon S3 ストレージ – 証明書検証エンドポイント – `*.amazontrust.com`: 証明書のステータスの検証に使用されます。証明書検証エンドポイント (CRL URL と OCSP サーバー) は変更される可能性があることを考慮します。実際のアドレス一覧は、証明書自体に記載されています。

機能制限

- Veeamは、VeeamオブジェクトストレージリポジトリとしてS3バケットのS3ライフサイクルポリシーをサポートしていません。これには、Amazon S3 ストレージクラスの移行と S3 ライフサイクルの有効期限ルールを含むポリシーが含まれます。これらのオブジェクトを管理するのはVeeamだけである必要があります。S3 ライフサイクルポリシーを有効にすると、データ損失などの予期しない結果が生じる可能性があります。

製品バージョン

- Veeam Backup & Replication v9.5 Update 4 以降（バックアップのみまたは容量階層）
- Veeam Backup & Replication v10 以降（バックアップ、または容量階層と S3 Object Lock）
- Veeam Backup & Replication v11 以降（バックアップまたはキャパシティ層、アーカイブまたはアーカイブ層、S3オブジェクトロック）
- Veeam Backup & Replication v12 以降（パフォーマンス階層、バックアップまたはキャパシティ階層、アーカイブまたはアーカイブ層、S3オブジェクトロック）
- S3 Standard
- S3 Standard – IA
- S3 1 ゾーン - IA
- S3 Glacier Flexible Retrieval (v11 以降のみ)
- S3 Glacier Deep Archive (v11 以降のみ)
- S3 Glacier Instant Retrieval (v12 以降のみ)

アーキテクチャ

ソーステクノロジースタック

- VeeamBackup サーバーまたは Veeam ゲートウェイサーバーから Amazon S3 に接続するオンプレミスの Veeam バックアップ&レプリケーションインストール

ターゲットテクノロジースタック

- Amazon S3
- Amazon VPC と Amazon EC2 (アーカイブ階層を使用する場合)

ターゲットアーキテクチャ SOBR

次の図は、スケールアウトバックアップリポジトリ (SOBR) アーキテクチャを示しています。

Veeam Backup and Replication ソフトウェアは、システム障害、アプリケーションエラー、偶発的な削除などの論理的エラーからデータを保護します。この図では、バックアップは最初にオンプレミスで実行され、セカンダリコピーは直接 Amazon S3 に送信されます。バックアップは、データのポイントインタイムコピーです。

このワークフローは、Amazon S3 へのバックアップの階層化またはコピーに必要な 3 つの主要コンポーネントと、1 つのオプションコンポーネントで構成されています。

- Veeam Backup & Replication (1) — バックアップインフラストラクチャ、設定、ジョブ、リカバリータスク、およびその他のプロセスの調整、制御、管理を行うバックアップサーバー。
- Veeam ゲートウェイサーバー (図には示されていません) — Veeam バックアップサーバーが Amazon S3 へのアウトバウンド接続を持たない場合に必要な、オプションのオンプレミスゲートウェイサーバー。
- スケールアウト・バックアップ・リポジトリ (2) — データの多層ストレージ用の水平スケーリングをサポートするリポジトリ・システム。スケールアウトバックアップリポジトリは、データへの高速アクセスを提供する 1 つ以上のバックアップリポジトリで構成され、長期ストレージ (容量階層) とアーカイブ (アーカイブ層) 用に Amazon S3 オブジェクトストレージリポジトリで拡張できます。Veeamはスケールアウトバックアップリポジトリを使用して、ローカル (パフォーマンス階層) とAmazon S3 オブジェクトストレージ (容量層とアーカイブ層) の間でデータを自動的に階層化します。

> **Note**
>
> Veeam Backup & Replication v12.2 以降、Direct to S3 Glacier 機能は S3 容量階層をオプションにします。SOBR は、パフォーマンス階層と S3 Glacier アーカイブ階層で設定できます。この設定は、キャパシティ階層のローカル (オンプレミス) ストレージに多額の投資を行い、クラウドでの長期アーカイブ保持のみを必要とするユーザーに役立ちます。詳細については、Veeam Backup & Replication のドキュメントを参照してください。

- Amazon S3 (3) – スケーラビリティ、データの可用性、セキュリティ、パフォーマンスを提供する AWS オブジェクトストレージサービス。

ターゲットアーキテクチャ DTO

次の図は、ダイレクト・トゥ・オブジェクト（DTO）アーキテクチャを示しています。

この図では、バックアップデータは最初にオンプレミスに保存されることなく Amazon S3 に直接送信されます。セカンダリコピーは S3 Glacier に保存できます。

自動化とスケール

VeeamHub GitHub リポジトリで提供されている AWS CloudFormation テンプレートを使用して、IAM リソースと S3 バケットの作成を自動化できます。テンプレートには標準オプションと不変オプションの両方が含まれています。

ツール

ツールと AWS のサービス

- 「Veeam Backup & Replication」は、仮想ワークロードと物理ワークロードを保護、バックアップ、レプリケーション、復元するための Veeam のソリューションです。
- AWS CloudFormation は、AWS リソースのモデル化とセットアップ、迅速かつ一貫したプロビジョニング、ライフサイクル全体の管理に役立ちます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。スタックは、複数の と にまたがって管理 AWS アカウント およびプロビジョニングできます AWS リージョン。
- Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動できます。
- AWS Identity and Access Management (IAM) は、へのアクセスを安全に制御するためのウェブサービスです AWS のサービス。IAM を使用すると、ユーザー、アクセスキーなどのセキュリティ認証情報、およびユーザーとアプリケーションがアクセスできる AWS リソースを制御するアクセス許可を一元管理できます。
- Amazon Simple Storage Service (Amazon S3)は、オブジェクトストレージを提供します。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- 「Amazon S3 Glacier (S3 Glacier)」は、低コストでデータのアーカイブおよび長期バックアップを行うための、安全性と耐久性に優れたサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークで AWS リソースを起動 AWS クラウド できる の論理的に分離されたセクションをプロビジョニングします。仮想

ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるというメリットがあります。

コード

「VeeamGitHub リポジトリ」 で提供されている CloudFormation テンプレートを使用して、このパターンの IAM リソースと S3 バケットを自動的に作成します。これらのリソースを手動で作成したい場合は、エピック セクションの手順に従ってください。

ベストプラクティス

- IAMのベストプラクティスに従い、Veeam Backup & Replication のバックアップを Amazon S3 に書き込むために使用するIAMユーザーなど、長期にわたるIAMユーザーの認証情報を定期的にローテーションすることを強くお勧めします。詳細については、IAM ドキュメントの「セキュリティのベストプラクティス」を参照してください。

エピック

Amazon S3 ストレージをアカウントに設定する

タスク	説明	必要なスキル
IAM ユーザーを作成します。	「IAM ドキュメントの指示」 に従って IAM ユーザーを作成します。このユーザーには AWS コンソールアクセスは許可されません。また、このユーザーのアクセスキーを作成する必要があります。Veeam はこのエンティティを使用して で認証 AWS し、S3 バケットの読み取りと書き込みを行います。ユーザーに必要以上の権限が付与されないように、最小特権（つまり、タスクの実行に必要な権限のみを付与）する必	AWS 管理者

タスク	説明	必要なスキル
	要があります。Veeam IAM ユーザーにアタッチする IAM ポリシーの例については、「追加情報」 セクションを参照してください。 **Note** または、VeeamHub GitHub リポジトリで提供されている CloudFormation テンプレートを使用して、このパターンの IAM ユーザーと S3 バケットを作成することもできます。	

タスク	説明	必要なスキル
S3 バケットを作成する。	1. にサインイン AWS Management Console し、Amazon S3 コンソールを開きます。 2. ターゲットストレージとして使用する既存の S3 バケットがまだない場合は、バケットの作成を選択し、バケット名 AWS リージョン、およびバケット設定を指定します。 • S3 バケットの「Block Public Access オプション」 を有効にし、組織の要件に合わせてアクセスポリシーとユーザー権限ポリシーを設定することをお勧めします。例については、「Amazon S3documentation」を参照してください。 • 「すぐに使用する予定がない場合でも、S3 オブジェクトロック」 を有効にすることをお勧めします。この設定は S3 バケットの作成時にのみ有効にできます。 詳細については、Amazon S3 ドキュメントの「バケットの作成」 を参照してください。	AWS 管理者

Amazon S3 および S3 Glacier Flexible Retrieval (または S3 Glacier Deep Archive) を Veeam Backup & Replication に追加する

タスク	説明	必要なスキル
新規オブジェクトリポジトリウィザードを起動します。	Veeam でオブジェクトストレージとスケールアウトバックアップリポジトリを設定する前に、容量とアーカイブ階層に使用する Amazon S3 および S3 Glacier ストレージリポジトリを追加する必要があります。次のエピックでは、これらのストレージリポジトリをスケールアウトバックアップリポジトリに接続します。 1. Veeam コンソールで、バックアップ・インフラストラクチャー・ビューを開きます。 2. インベントリペイン で、Backup リポジトリ」ノードを選択し、リポジトリを追加を選択します。 3. Backup リポジトリを追加ダイアログで、オブジェクトストレージ、Amazon S3 を選択します。	AWS 管理者、アプリ所有者
キャパシティティアに Amazon S3 ストレージを追加します。	1. Amazon クラウドストレージサービスダイアログボックスで、Amazon S3 を選択します。 2. ウィザードの名前ステップで、オブジェクトストレー	AWS 管理者、アプリ所有者

タスク	説明	必要なスキル
	ジの名前と、作成者や作成日などの簡単な説明を指定します。 3. ウィザードのアカウントステップで、オブジェクトストレージアカウントを指定します。 • 認証情報では、最初の工ピックで作成した IAM ユーザーを選択して Amazon S3 オブジェクトストレージにアクセスします。 • AWS リージョンでは、S3 バケット AWS リージョン がある を選択します。 4. ウィザードの Bucket ステップで、オブジェクトストレージ設定を指定します。 • データセンターリージョンで、S3 バケット AWS リージョン がある を選択します。 • バケットでは、最初の工ピックで作成された S3 バケットを選択します。 • Folder では、オブジェクトストレージリポジトリをマップするクラウドフォルダを作成または選択します。	

タスク	説明	必要なスキル
	• 不変性を有効にする場合は、最近のバックアップを X 日間不変にするを選択し、バックアップをロックする期間を設定します。不変性を有効にすると、Veeam から Amazon S3 への API 呼び出すの数が増えるため、コストが増加することに注意してください。 5. ウィザードの Summary ステップで設定情報を確認し、Finish を選択します。	

タスク	説明	必要なスキル
アーカイブ層に S3 Glacier ストレージを追加します。	アーカイブ階層を作成する場合は、「追加情報」 セクションに詳述されている IAM 権限を使用してください。 1. 前述のように、新規オブジェクトリポジトリウィザードを起動します。 2. Amazon クラウドストレージサービスダイアログボックスで、Amazon S3 Glacier を選択します。 3. ウィザードの名前ステップで、オブジェクトストレージの名前と、作成者や作成日などの簡単な説明を指定します。 4. ウィザードのアカウントステップで、オブジェクトストレージアカウントを指定します。 • 認証情報では、最初のエピックで作成した IAM ユーザーを選択して、S3 Glacier オブジェクトストレージにアクセスします。 • AWS リージョンでは、S3 バケット AWS リージョン がある を選択します。 5. ウィザードの Bucket ステップで、オブジェクト	AWS 管理者、アプリ所有者

タスク	説明	必要なスキル
	ストレージ設定を指定します。 • データセンターリージョンで、 を選択します AWS リージョン。 • Bucket で、バックアップデータを保存する S3 バケットを選択します。これは、容量階層に使用したのと同じバケットでもかまいません。 • Folder では、オブジェクトストレージリポジトリをマップするクラウドフォルダを作成または選択します。 • イミュータビリティを有効にする場合は、最近のバックアップをリテンションポリシーの全期間にわたってイミュータブルにするを選択します。不変性を有効にすると、Veeam から Amazon S3 への API 呼び出すの数が増えるため、コストが増加することに注意してください。 • S3 Glacier ディープアーカイブをアーカイブストレージクラスとして使用する場合は、ディープアーカイブストレージ	

タスク	説明	必要なスキル
	クラスを使用を選択します。 6. ウィザードの Proxy Appliance ステップで、Amazon S3 から S3 Glacier にデータを転送するために使用される補助インスタンスを設定します。 S3 デフォルト設定を使用できます。または各設定を手動で設定できます。手動で設定するには: • [Customize] (カスタマイズ) を選択してください。 • EC2 インスタンスタイプでは、スケールアウトバックアップリポジトリのアーカイブ階層にバックアップファイルを転送する際の速度とコストの要件に基づいて、プロキシアプライアンスのインスタンスタイプを選択します。 • Amazon VPC の場合は、ターゲットインスタンスの VPC を選択します。 • サブネットで、プロキシアプライアンスのサブネットを選択します。 • セキュリティグループでは、セキュリティグループを選択して、プロキシ	

タスク	説明	必要なスキル
	アプライアンスに関連します。 • リダイレクターポートには、プロキシアプライアンスとバックアップインフラストラクチャコンポーネント間のリクエストをルーティングするための TCP ポートを指定します。 • OK を選択して設定を確定します。 7. ウィザードの Summary ステップで設定情報を確認し、Finish を選択します。	

スケールアウトバックアップリポジトリの追加

タスク	説明	必要なスキル
新規スケールアウトBackup リポジトリ」ウィザードを起動します。	1. Veeam コンソールで、バックアップ・インフラストラクチャー・ビューを開きます。 2. インベントリペインでスケールアウトリポジトリを選択し、スケールアウトリポジトリを追加を選択します。	APP オーナー、AWS システム管理者
スケールアウト型バックアップリポジトリを追加し、容量とアーカイブ階層を設定します。	1. ウィザードの名前ステップで、スケールアウトバックアップリポジトリの名前と簡単な説明を指定します。	APP オーナー、AWS システム管理者

タスク	説明	必要なスキル
	2. 必要に応じて、パフォーマンスエクステントを追加します。既存の Veeam ローカルバックアップリポジトリをパフォーマンス階層として使用することもできます。 Veeam バージョン 12 以降、ローカルのパフォーマンス階層をバイパスして、ダイレクト・トゥ・オブジェクト（DTO）バックアップのパフォーマンス範囲としてS3バケットを追加できるようになりました。 3. Advanced を選択し、スケールアウト・バックアップ・リポジトリの追加オプションを指定します。 • マシンごとのバックアップファイルを使用を選択してマシンごとに個別のバックアップファイルを作成し、それらのファイルを複数のストリームでバックアップリポジトリに同時に書き込みます。このオプションは、ストレージとコンピュートリソースをより有効に活用するために推奨されます。 • インクリメンタルバックアップの復元ポイントを	

タスク	説明	必要なスキル
	含むエクステントがオフラインになった場合に備えて、必要なエクステントがオフラインのときにフルバックアップを実行を選択してフルバックアップファイルを作成します。このオプションでは、フルバックアップファイルをホストするためにスケールアウトバックアップリポジトリに空きスペースが必要です。 4. ウィザードのポリシーステップで、リポジトリのバックアップ配置ポリシーを指定します。 • 同じチェーンに属するフルバックアップファイルとインクリメンタルバックアップファイルを同じパフォーマンス範囲で一緒に保存するには、Data locality を選択します。新しいバックアップチェーンに属するファイルは、同じパフォーマンス範囲または別のバックアップチェーンに保存できます (パフォーマンスの範囲として重複排除ストレージアプライアンスを使用している場合を除く)。	

タスク	説明	必要なスキル

	• フルバックアップファイルとインクリメンタルバックアップファイルを異なるパフォーマンス範囲で保存するには、パフォーマンスを選択します。このオプションには、高速で信頼性の高いネットワーク接続が必要です。パフォーマンスを選択すると、保存するバックアップファイルのタイプをパフォーマンスの範囲ごとに制限できます。たとえば、フルバックアップファイルを 1 つのエクステントに保存し、インクリメンタルバックアップファイルを他のエクステントに保存できます。ファイルタイプを選択するには:
	  • [Customize] (カスタマイズ) を選択してください。
	  • Backup 配置設定ダイアログボックスで、パフォーマンス範囲を選択し、編集を選択します。
	  • エクステントに保存するバックアップファイルのタイプを選択します。	

タスク	説明	必要なスキル
	5. ウィザードのキャパシティ階層のステップで、スケールアウトバックアップリポジトリにアタッチする長期ストレージ階層を設定します。 • オブジェクトストレージでスケールアウトバックアップリポジトリの容量を拡張するを選択します。オブジェクトストレージリポジトリでは、前のエピックで追加したキャパシティティアのAmazon S3 ストレージを選択します。 • ウィンドウを選択して、データを移動またはコピーする時間枠を選択します。 • すべてまたは最近作成されたバックアップファイルのみを容量の範囲内でコピーするには、バックアップを作成したらすぐにオブジェクトストレージにコピーするを選択します。 • 使用頻度の低いバックアップチェーンを容量の範囲まで転送するには、オペレーションリストア期間が終了したらバックアップをオブジェクトス	

タスク	説明	必要なスキル
	トレージに移動を選択します。X 日以上経過したバックアップファイルの移動フィールドで、バックアップファイルをオフロードするまでの期間を指定します。(使用頻度の低いバックアップチェーンを作成日にオフロードするには、0 日を指定します)。スケールアウトバックアップリポジトリが指定したしきい値に達した場合に、オーバーライドを選択してバックアップファイルをより早く移動することもできます。 • オブジェクトストレージにアップロードされたデータを暗号化を選択し、パスワードを指定して、すべてのデータとそのメタデータをオフロード用に暗号化します。パスワードの追加またはパスワードの管理を選択して新しいパスワードを指定します。 6. ウィザードの容量階層のステップで、スケールアウトバックアップリポジトリにアタッチする長期ストレージ階層を設定します。	

タスク	説明	必要なスキル
	(Amazon S3 Glacier ストレージの追加をスキップした場合、このステップは表示されません)。 • GFS 完全バックアップをオブジェクトストレージにアーカイブを選択します。オブジェクトストレージリポジトリには、前のエピックで追加した Amazon S3 Glacier ストレージを選択します。 • N 日以上経過したアーカイブ GFS バックアップの場合は、ファイルをアーカイブ範囲に移動する時間枠を選択します。(使用頻度の低いバックアップチェーンを作成日にアーカイブするには、0 日を指定します)。 7. ウィザードの概要ステップで、スケールアウトバックアップリポジトリの設定を確認し、完了を選択します。	

関連リソース

- で IAM ユーザーを作成する AWS アカウント (IAM ドキュメント)
- 「バケットの作成」 (Amazon S3 ドキュメント)
- 「Amazon S3 ストレージへのパブリックアクセスのブロック」 (Amazon S3 ドキュメント)
- 「S3 オブジェクトロックを使用する」 (Amazon S3 ドキュメント)

- 「Veeam 技術ドキュメント」
- 「S3 オブジェクトストレージに接続するためのセキュア IAM ポリシーを作成する方法」 (Veeam ドキュメント)

追加情報

以下のセクションでは、このパターンの 「エピック」 セクションでIAMユーザーを作成するときに使用できるサンプルIAMポリシーを紹介します。

キャパシティ階層の IAM ポリシー

> **Note**
>
> サンプルポリシーの S3 バケットの名前を <yourbucketname>から、Veeam 容量階層バックアップに使用する S3 バケットの名前に変更します。また、 AWS ブログ記事Amazon S3 オブジェクトの意図しない暗号化の防止」で説明されているように、ポリシーは Veeam で使用される特定のリソース (次のポリシーのResource仕様で示される) に制限する必要があります。また、ポリシーの最初の部分ではクライアント側の暗号化が無効になることに注意してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        },
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
```

```
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:PutObjectLegalHold",
                "s3:GetBucketVersioning",
                "s3:GetObjectLegalHold",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObject*",
                "s3:GetObject*",
                "s3:GetEncryptionConfiguration",
                "s3:PutObjectRetention",
                "s3:PutBucketObjectLockConfiguration",
                "s3:DeleteObject*",
                "s3:DeleteObjectVersion",
                "s3:GetBucketLocation"

            ],
            "Resource": [
                "arn:aws:s3:::<yourbucketname>",
                "arn:aws:s3:::<yourbucketname>/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

アーカイブ階層の IAM ポリシー

> **Note**
>
> サンプルポリシーの S3 バケットの名前を <yourbucketname>から、Veeam アーカイブ階層のバックアップに使用する S3 バケットの名前に変更します。

既存の VPC、サブネット、およびセキュリティグループを使用するには:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:RestoreObject",
                "s3:ListBucket",
                "s3:AbortMultipartUpload",
                "s3:GetBucketVersioning",
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObjectRetention",
                "s3:GetObjectVersion",
                "s3:PutObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:DeleteObjectVersion",
                "s3:ListBucketVersions"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>",
                "arn:aws:s3:::<bucket-name>/*"
            ]
        }
    ]
}

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2Permissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:CreateKeyPair",
                "ec2:DescribeKeyPairs",
                "ec2:RunInstances",
```

```
                "ec2:DeleteKeyPair",
                "ec2:DescribeVpcAttribute",
                "ec2:CreateTags",
                "ec2:DescribeSubnets",
                "ec2:TerminateInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:DescribeVpcs"
            ],
            "Resource": "arn:aws:ec2:<region>:<account-id>:*"
        }
    ]
}
```

新しい VPC、サブネット、およびセキュリティグループを作成するには

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:RestoreObject",
                "s3:ListBucket",
                "s3:AbortMultipartUpload",
                "s3:GetBucketVersioning",
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObjectRetention",
                "s3:GetObjectVersion",
                "s3:PutObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:DeleteObjectVersion",
                "s3:ListBucketVersions"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>",
                "arn:aws:s3:::<bucket-name>/*"
```

```
            ]
        }
    ]
}

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2Permissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:CreateKeyPair",
                "ec2:DescribeKeyPairs",
                "ec2:RunInstances",
                "ec2:DeleteKeyPair",
                "ec2:DescribeVpcAttribute",
                "ec2:CreateTags",
                "ec2:DescribeSubnets",
                "ec2:TerminateInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:DescribeVpcs",
                "ec2:CreateVpc",
                "ec2:CreateSubnet",
                "ec2:DescribeAvailabilityZones",
                "ec2:CreateRoute",
                "ec2:CreateInternetGateway",
                "ec2:AttachInternetGateway",
                "ec2:ModifyVpcAttribute",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSecurityGroup",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:DescribeRouteTables",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

VMware Cloud on AWS 向けVeritas NetBackup の構成

作成者: Shubham Salani (AWS)

概要

> 注意: 2024 年 4 月 30 日現在、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなりました。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、AWS 担当者にお問い合わせください。

多くの企業が、オンプレミスの VMware vSphere ベースのワークロードのバックアップとリカバリソリューションとして Veritas NetBackup を使用しています。企業がワークロードを Amazon Web Services (AWS) インフラストラクチャ上の VMware Cloud にあるソフトウェア定義のデータセンター (SDDC) に移行した後、NetBackup を統合するための明確なリフトアンドシフト手順はありません。このパターンは、AWS アカウントで Veritas NetBackup をセットアップし、VMware SDDC のワークロードをバックアップできるように構成する方法を示しています。

このパターンには、ワークロードを移行する手順は含まれていません。詳細は、「Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX」を参照してください。ワークロードを VMware Cloud on AWS に設定するときは、ストレッチクラスター (VMware ドキュメント) を使用してください 。この構成では、クラスターは 1 リージョン内の 2 つの AWS アベイラビリティゾーンにまたがります。これにより、アベイラビリティゾーンのいずれかが利用できなくなった場合でも、高い可用性と耐障害性が得られます。Elastic DRS と vSAN 監視ホスト (VMware ドキュメント) は、データをフォールトドメインと呼ばれる 3 番目のアベイラビリティゾーンにシームレスにコピーします。このパリティソリューションにより、障害発生時にデータを復元しやすくなります。このアプローチでは 3 つのアベイラビリティゾーンが必要なため、VMware Cloud 環境用の AWS リージョンを選択するときは、そのリージョンに 3 つ以上のアベイラビリティゾーンがあることを確認してください。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。

このパターンでは、各 SDDC にプロキシサーバーであるバックアップホストがあります。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用して、NetBackup マスターサーバーとメディアサーバーを、SDDC ごとに 1 つずつ、個別の仮想プライベートクラウド (VPC) で設定します。エラスティックネットワークインターフェイスは高帯域幅と低レイテンシーを提供するため、それらを使用してバックアップホストと対応する NetBackup マスターおよびメディアサーバー間の接続を構成します。EC2 インスタンスは、バックアップの最初のポイントである Amazon Elastic Block Store (Amazon EBS) ボリュームにバックアップを送信します。AWS DataSync を使用することで、SDDC の EBS ボリュームを同期した状態に保つことができます。

また、AWS Transit Gateway とインターフェイス VPC エンドポイントを使用して、EBS ボリュームを Amazon Simple Storage Service (Amazon S3) など別のストレージサービスに接続できます。企業はデータ保持ポリシーに従って、S3 Intelligent-Tiering S3 Glacier ストレージクラスを使用してストレージコストを最適化できます。詳細については、Amazon S3 ストレージクラス (Amazon S3 ドキュメント) を参照してください。

前提条件と制限

前提条件

- VMware Cloud on AWS 環境で、2 つのアベイラビリティゾーンにまたがるストレッチクラスターを使用してること。
- バックアップホストが、VMware Virtual Machine Disk File (VMDK) ファイルがデプロイされているデータストアにアクセスできる VMware Cloud on AWS SDDC 上に配置されている。
- 仮想マシン (VM) をバックアップおよび復元するために、NetBackup クライアントで HotAdd トランスポートモードが有効になっている。また、ユーザー指定のファイルおよびフォルダからの復元が許可されている。

制約事項

- NetBackup マスターサーバーは、SDDC 内の vCenter バックアップホストのプライベート IP アドレスに対して DNS 解決を使用する必要があります。
- NetBackup マスターサーバーとバックアップホストのホストファイルには、次のものが含まれている必要があります。
  - マスターサーバーのプライベート IP アドレスとプライベート DNS 名。
  - バックアップホストのプライベート IP アドレスとプライベート DNS 名。
- S3 バケットにインターフェイス VPC エンドポイントを構成する場合、Classless Inter-Domain Routing (CIDR) ブロックソースからの HTTPS を許可するように SDDC Compute ゲートウェイファイアウォールを構成する必要があります。詳細については、「S3 エンドポイントを使用して S3 バケットにアクセスする」（VMware ドキュメント）を参照してください。
- VMware Cloud on AWS では NetBackup の次の機能をサポートしていません。
  - VM テンプレートのバックアップまたは復元
  - NetBackup vSphere クライアント (HTML5 プラグイン) の使用
  - バックアップまたは復元用の VM のロックとロック解除
  - vSAN データストアへのバックアップの保存が不可

- ネットワークブロックデバイス (NBD)、NBDSSL、SAN トランスポートモード

製品バージョン

- VMware Cloud on AWS SDDC バージョン 1.0 以降
- Veritas NetBackup Version 8.1.2 以降
- Linux バージョン 6.8 以降
- VMware vSphere バージョン 6.0 以降

アーキテクチャ

次の図は、VMware Cloud on AWS 向け NetBackup の構成を示しています。NetBackup マスターサーバーとメディアサーバーは別々の VPC にデプロイされ、Elastic Network Interface によって SDDC のバックアップホストに接続されます。NetBackup マスターサーバーとメディアサーバーは、バックアップを Amazon EBS ボリュームに保存します。オプションで、AWS Transit Gateway と AWS PrivateLink インターフェイスの VPC エンドポイントを使用して、Amazon S3 バケットに追加のストレージを構成できます。

ツール

サービスとツール

- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。
- PrivateLink は、仮想プライベートクラウド (VPC) から他の VPC 内のサービスへの単方向のプライベート接続の確立に役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

その他のサービス

- VMware Cloud on AWS は、Amazon Web Services (AWS) と VMware が共同開発した統合クラウドサービスです。
- NetBackup for VMware は、VMware ESXi ホストで実行されている VMware 仮想化マシンをバックアップおよび復元します。

エピック

NetBackup サーバーの構成

タスク	説明	必要なスキル
ファイアウォールルールを更新します。	ファイアウォールルールを更新して、VMware Cloud on AWS SDDC と NetBackup マスターおよびメディアサーバー間の接続を確立します。以下の操作を実行します。 1. https://vmc.vmware.com/ で VMware Cloud on AWS にログインします 2. [ネットワークとセキュリティ] タブで [ゲートウェイファイアウォール] を選択します。 3. 「ゲートウェイファイアウォール」ページで [コンピュートゲートウェイ] を選択します。 4. [ルールの追加] を選択し、必要なファイアウォールポート設定を含む新しいルールを作成します。詳細については、「NetBackup ファイアウォールのポー	ネットワーク管理者、クラウド管理者

タスク	説明	必要なスキル
	ト要件 (Veritas ドキュメント)」を参照してください。	
NetBackup マスターサーバーとメディアサーバーを起動します。	1. AWS マネジメントコンソールにサインインし、Amazon EC2 コンソール (https://console..amazon.com/ec2/) を開きます。 2. EC2 インスタンス (Amazon EC2 ドキュメント) を起動し、以下の詳細を使って構成を行います。 a. NetBackup マスターサーバーとメディアサーバーには、`NBU-Linux-GA-8-1-2-Setup-f032d23e-881b-4dee-ba70-b9ca3e915910-ami-072509a7ffc156938.4` Amazon マシンイメージ (AMI) を選択します。この事前設定された AMI は、AWS Marketplace から入手できます。 b. インスタンスタイプの選択 NetBackup はマスターサーバーとメディアサーバーに `m5.2xlarge` を推奨しています。	クラウド管理者、Backup 管理者

タスク	説明	必要なスキル
NetBackup のバックアップホストを構成します。	1. https://vmc.vmware.com/で VMware Cloud on AWS にログインします 2. SDDC を選択します。 3. [VCENTER を開く] タブを選択します。SDC vCenter が開きます。 4. バックアップホストの完全修飾ドメイン名 (FQDN) を書き留めておきます。 5. NetBackup 管理コンソールにログインします。詳細については、「NetBackup 管理コンソールへのログイン」(Veritas ドキュメント) を参照してください。 6. マスターサーバーとメディアサーバーを選択し、[VMware アクセスホスト] を選択します。 7. バックアップホストの FQDN を入力します。 8. [適用]、[OK] の順に選択します。	クラウド管理者、Backup 管理者

(オプション) Amazon S3 ストレージの設定

タスク	説明	必要なスキル
Amazon S3 でストレージを構成します。	1. Amazon S3 クラウドストレージオプション (Veritas ドキュメント) を確認し、	クラウド管理者、AWS 全般

タスク	説明	必要なスキル
	要件に適したストレージクラスを選択します。 2. 「NetBackup でのクラウドストレージの構成」(Veritas ドキュメント) に記載されている手順に従って、クラウドストレージに Amazon S3 を使用するように NetBackup を構成します。	

関連リソース

ドキュメント

- インターフェイス VPC エンドポイントの作成 (AWS PrivateLink のドキュメント)

Veritas ドキュメント

- NetBackup ファイアウォールのポート要件

VMware ドキュメント

- コンテンツライブラリの OVF テンプレートから VM をデプロイする
- VMware Cloud on データ転送料金: その仕組み (VMware のブログ記事)
- VMware Cloud on : ストレッチクラスター

AWS CLI を使用して S3 バケットから別のアカウントとリージョンにデータをコピーする

Appasaheb Bagali (AWS) と Purushotham G K (AWS) によって作成された

概要

このパターンは、AWS ソースアカウントの Amazon Simple Storage Service (Amazon S3) バケットから、同じ AWS リージョンまたは別のリージョンにある、別の AWS アカウントの宛先 S3 バケットにデータを移行する方法を示しています。

ソース S3 バケットでは、添付のリソースポリシーを使用して AWS Identity and Access Management (IAM) アクセスを許可されます。送信先アカウントのユーザーは、ソースバケットに対する`PutObject`と`GetObject`権限を持つロールを引き受ける必要があります。最後に、`copy`と`sync`コマンドを実行して、送信元 S3 バケットから送信先 S3 バケットにデータを転送します。

S3 バケットにアップロードするオブジェクトは、アカウントが所有します。複数のアカウントやリージョンにわたってオブジェクトをコピーする場合は、コピー先のアカウントにコピーされたオブジェクトの所有権を付与します。オブジェクトの所有権は、「アクセス制御リスト (ACL)」を`bucket-owner-full-control`に変更することで変更できます。ただし、ACL は複数のオブジェクトでは管理が難しい場合があるため、プログラムによるクロスアカウント権限を移行先のアカウントに付与することをお勧めします。

> **Warning**
>
> このシナリオでは、プログラムによるアクセスと長期的な認証情報を持つ IAM ユーザーが必要です。これはセキュリティリスクをもたらします。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。アクセスキーは、必要に応じて更新できます。詳細については、「IAM ユーザーガイド」の「アクセスキーの更新」を参照してください。

このパターンでは、1 回限りの移行について説明します。ソースバケットから宛先バケットへの新しいオブジェクトの継続的かつ自動的な移行が必要なシナリオでは、代わりに S3 バッチレプリケーションを使用できます。パターン「S3 バッチS3レプリケーションを使用して S3 バケットから別のアカウントとリージョンにデータをコピーする」を参照してください。

前提条件と制限

- 同じまたは異なる AWS リージョンにある 2 つのアクティブな AWS アカウント。
- ソースアカウント内の既存の S3 バケット。
- 送信元または送信先 Amazon S3 バケットで「デフォルト暗号化が有効である場合は、AWS Key Management Service (AWS KMS) Key アクセス許可を変更する必要があります。詳細については、このトピックの「AWS re： Post 記事」を参照してください。
- クロスアカウントアクセス許可に精通している。

アーキテクチャ

ツール

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

ベストプラクティス

- IAM でのセキュリティのベストプラクティス (IAM ドキュメント)
- 最小権限権限の適用 (IAM ドキュメント)

エピック

宛先の AWS アカウントに IAM ユーザーとロールを作成します。

タスク	説明	必要なスキル
IAM ユーザーを作成してアクセスキーを取得します。	1. AWS マネジメントコンソールにサインインして、プログラムによるアクセス権	AWS DevOps

タスク	説明	必要なスキル
	を持つ IAM ユーザーを作成します。詳細な手順については、IAM ドキュメントの「IAM ユーザーの作成」を参照してください。このユーザーにはポリシーをアタッチする必要はありません。 2. このユーザーのアクセスキーとシークレットキーを生成します。手順については、AWS ドキュメントの「AWS アカウントとアクセスキー」を参照してください。	

タスク	説明	必要なスキル
アイデンティティベースの IAM ポリシーを作成します。	以下の権限を使用して、S3MigrationPolicy という名前の IAM ID ベースのポリシーを作成します。詳細なステップについては、IAM ドキュメントの「IAM ポリシーの作成」を参照してください。 { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["s3:ListBucket", "s3:GetObject", "s3:GetObjectTagging", "s3:GetObjectVersion", "s3:GetObjectVersionTagging"], "Resource": ["arn:aws:s3:::amazon-s3-demo-source-bucket", "arn:aws:s3:::amazon-	AWS DevOps

タスク	説明	必要なスキル

	s3-demo-source-bucket/*"
	            ]
	        },
	        {
	            "Effect": "Allow",
	            "Action": [
	"s3:ListBucket",
	"s3:PutObject",
	"s3:PutObjectAcl",
	"s3:PutObjectTagging",
	"s3:GetObjectTagging",
	"s3:GetObjectVersion",
	"s3:GetObjectVersionTagging"
	            ],
	            "Resource": [
	"arn:aws:s3:::amazon-s3-demo-destination-bucket",
	"arn:aws:s3:::amazon-s3-demo-destination-bucket/*"
	            ]
	        }
	    ]
	}	

タスク	説明	必要なスキル
	Note ユースケースに応じて、レプリケート元バケット名とレプリケート先バケット名を変更します。 この ID ベースのポリシーにより、このロールを引き受けるユーザーは、ソースバケットとターゲットバケットにアクセスできます。	

タスク	説明	必要なスキル

IAM ロールを作成します。	以下の信頼ポリシーを使用してS3MigrationRole という名前の IAM ロールを作成し、先に作成したS3MigrationPolicy をアタッチします。詳細な手順については、IAM ドキュメントの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Principal": {
	"AWS": "arn:aws:iam::<destination_account>:user/<user_name>"
	},
	"Action": "sts:AssumeRole",
	"Condition": {}
	}
	]
	}
	
	Note
	ユースケースに従って、信頼ポリシーの送信先 IAM ロール
		AWS DevOps

タスク	説明	必要なスキル
	またはユーザー名の Amazon リソースネーム (ARN) を変更します。 この信頼ポリシーにより、新しく作成された IAM ユーザは`S3MigrationRole` であるとみなされます。	

S3 バケットポリシーを作成してソースアカウントでアタッチします

タスク	説明	必要なスキル
S3 バケットポリシーを作成してアタッチします。	ソースの AWS マネジメントコンソールにサインインして、[Amazon S3 コンソール] を開きます。ソースの S3 バケットを選択して [アクセス権限]を選択します。[バケットポリシー] の下で、[編集]を選択し、以下のバケットポリシーを貼り付けます。[Save] を選択します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "DelegateS3Access", "Effect": "Allow",`	クラウド管理者

タスク	説明	必要なスキル

	<pre>"Principa
	l": {"AWS": "arn:aws:
	iam::<destination_
	account>:role/<Rol
	eName>"},
	"Action":
	["s3:ListBucket",
	
	"s3:GetObject",
	
	"s3:GetObjectTaggi
	ng",
	
	"s3:GetObjectVersi
	on",
	
	"s3:GetObjectVersi
	onTagging"
	
	],
	"Resource":
	[
	
	"arn:aws:s3:::amazon-
	s3-demo-source-bucket/
	*",
	
	"arn:aws:s3:::amazon-
	s3-demo-source-bucket"
	]
	}
	]
	}</pre>	

	Note 送信先アカウントの AWS アカウント ID を必ず含め、要件に従ってバケットポリシーテ	

タスク	説明	必要なスキル
	ンプレートを設定します。 このリソースベースのポリシーでは、デスティネーションロール`S3MigrationRole` がソースアカウントの S3 オブジェクトにアクセスすることを許可します。	

送信先 S3 バケットの設定

タスク	説明	必要なスキル
宛先 S3 バケットを作成します。	送信先アカウントの AWS マネジメントコンソールにサインインして、Amazon S3 コンソールを開き、「バケットの作成」を選択します。要件に従って S3 バケットを作成します。詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	クラウド管理者

送信先 S3 バケットにデータをコピーする

タスク	説明	必要なスキル
新しく作成したユーザー認証情報を使用して AWS CLI を設定します。	1. AWS CLI の最新リリースをインストールします。指示については、「AWS CLI の最新バージョンをインス	AWS DevOps

タスク	説明	必要なスキル
	トールまたは更新する」を参照してください。 2. $ aws configure を実行し、作成したユーザの AWS アクセス・キーで CLI を更新します。詳細については、AWS CLI ドキュメントの「設定と機密ファイル設定」を参照してください。	

タスク	説明	必要なスキル
S3 の移行ロールを引き受けてください。	1. AWS CLI を使用してS3MigrationRole を仮定する： ``` aws sts assume-role \ --role-arn "arn:aws:iam::<destination_account>:role/S3MigrationRole" \ --role-session-name AWSCLI-Session ``` このコマンドは複数の情報を出力します。認証情報ブロック内にはAccessKeyId 、SecretAccessKey 、SessionToken が必要です。この例では、環境変数RoleAccessKeyID 、RoleSecretKey とRoleSessionToken を使用しています。有効期限フィールドのタイムスタンプは UTC タイムゾーンであることに注意してください。タイムスタンプは IAM ロールの一時認証情報がいつ期限切れになるかを示します。一時認証情報の有効期限が切れると、sts:AssumeRole API を再度呼び出す必要があります。	AWS 管理者

タスク	説明	必要なスキル

	2. IAM ロールを引き受ける環境変数を 3 つ作成します。これらの環境変数には以下の出力が入力されます。
	
	`# Linux`
	`export AWS_ACCES`
	`S_KEY_ID=RoleAcces`
	`sKeyID`
	`export AWS_SECRE`
	`T_ACCESS_KEY=RoleS`
	`ecretKey`
	`export AWS_SESSI`
	`ON_TOKEN=RoleSessi`
	`onToken`
	`# Windows`
	`set AWS_ACCES`
	`S_KEY_ID=RoleAcces`
	`sKeyID`
	`set AWS_SECRE`
	`T_ACCESS_KEY=RoleS`
	`ecretKey`
	`set AWS_SESSI`
	`ON_TOKEN=RoleSessi`
	`onToken`
	
	3. IAM ロールを引き受けることを検証するには、以下のコマンドを実行します。
	
	`aws sts get-caller-identity`
	
	詳細については、「AWS Knowledge Center」を参照してください。	

タスク	説明	必要なスキル

送信元 S3 バケットから送信先 S3 バケットにデータをコピーして同期します。	ロールS3MigrationRole を引き受けたら、コピー (cp) コマンドまたは 同期 (sync) コマンドを使用してデータをコピーできます。
	
	コピーします。(詳細については、「AWS CLI コマンドリファレンス」を参照してください。)
	
	`aws s3 cp s3://amazon-s3-demo-source-bucket/ \ s3://amazon-s3-demo-destination-bucket/ \ --recursive --source-region SOURCE-REGION-NAME --region DESTINATION-REGION-NAME`
	
	同期します。(詳細については、「AWS CLI コマンドリファレンス」を参照してください。)
	
	`aws s3 sync s3://amazon-s3-demo-source-bucket/ \ s3://amazon-s3-demo-destination-bucket/ \ --source-region SOURCE-REGION-NAME`
		クラウド管理者

タスク	説明	必要なスキル
	`--region DESTINATION-REGION-NAME`	

トラブルシューティング

問題	ソリューション
PutObject オペレーションを呼び出すときにエラー (AccessDenied) が発生しました： アクセスが拒否されましたListObjects 。	• ロール を引き受けたことを確認しますS3MigrationRole 。 • aws sts get-caller-identity を実行して、使用したロールを確認します。出力にS3MigrationRole のARNが表示されない場合は、もう一度その役割を引き受けて再試行してください。

関連リソース

- S3 バケットの作成 (Amazon S3 ドキュメント)
- Amazon S3 バケットポリシーとユーザーポリシー (Amazon S3 ドキュメント)
- IAM ID (ユーザー、グループ、ロール)(IAM ドキュメント)
- cp コマンド (AWS CLI ドキュメント)
- 同期コマンド (AWS CLI ドキュメント)

S3 バッチレプリケーションを使用して S3 バケットから別のアカウントとリージョンにデータをコピーする

作成者: Appasaheb Bagali (AWS)、Lakshmikanth B D (AWS)、Purushotham G K (AWS)、Shubham Harsora (AWS)、Suman Rajotia (AWS)

概要

このパターンでは、Amazon Simple Storage Service (Amazon S3) バッチレプリケーションを使用して、バケットをセットアップした後、手動操作なしで S3 バケットの内容を別の S3 バケットに自動的にコピーする方法について説明します。レプリケート元バケットとレプリケート先バケットは、同じ または異なる AWS アカウント または リージョンにあることができます。

S3 バッチレプリケーションを使用すると、レプリケーション設定が配置される前に存在していた Amazon S3 オブジェクト、以前にレプリケートされたオブジェクト、およびレプリケーションに失敗したオブジェクトをレプリケートできます。このメソッドは S3 バッチオペレーションジョブを使用します。ジョブが終了すると、完了レポートが表示されます。

S3 バッチレプリケーションは、レプリケート元バケットからレプリケート先バケットへの新しいオブジェクトの継続的かつ自動的な移行を必要とするシナリオで使用できます。1 回限りの移行では、代わりに AWS Command Line Interface （ AWS CLI) を使用できます。パターン「 を使用して S3 バケットから別のアカウントとリージョンにデータをコピーする AWS CLI」を参照してください。

前提条件と制限

- ソース AWS アカウント。
- 送信先 AWS アカウント。
- オブジェクト (ファイルまたはフォルダ) がいくつかあるソースアカウントの S3 バケット。
- 送信先アカウントの 1 つ以上の S3 バケット。
- 送信元バケットと送信先バケットで有効になっている S3 バージョニング。
- AWS Identity and Access Management 送信元アカウントと送信先アカウントに IAM ポリシー、IAM ロール、および S3 バケットポリシーを作成する (IAM) アクセス許可。
- S3 バッチレプリケーションジョブがアクティブな間は、Amazon S3 ライフサイクルルールが無効になります。 S3 これにより、レプリケート元バケットとレプリケート先バケット間のパリティが確保されます。それ以外の場合、レプリケート先バケットはレプリケート元バケットの正確なレプリカではない可能性があります。

アーキテクチャ

ツール

AWS サービス

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

ベストプラクティス

re AWS :Invent 2022 の次の動画では、規制コンプライアンス、データ保護、アプリケーションパフォーマンスの向上のために Amazon S3 レプリケーションを使用するためのベストプラクティスについて説明します。

https://www.youtube-nocookie.com/embed/hrJEbISBL04?controls=0

エピック

ソースアカウントでクロスアカウントレプリケーション用の IAM ポリシーとロールを作成する

タスク	説明	必要なスキル
クロスアカウントレプリケーション用の IAM ポリシーを作成します。	AWS ソースアカウントで： 1. [IAM コンソール] を開きます。 2. 新しい IAM ポリシーを作成します。 3. ポリシーエディタセクションで JSON を選択し、次のコードを貼り付けます。 `{`	クラウド管理者、AWS 管理者

タスク	説明	必要なスキル
	"Version": "2012-10-17", "Statement": [{ "Sid": "GetSourceBucketConfiguration", "Effect": "Allow", "Action": ["s3:ListBucket", "s3:GetBucketLocation", "s3:GetBucketAcl", "s3:GetReplicationConfiguration", "s3:GetObjectVersionForReplication", "s3:GetObjectVersionAcl", "s3:GetObjectVersionTagging"], "Resource": ["arn:aws:s3:::source-bucket-name", "arn:aws:s3:::source-bucket-name/*"] }, {	

タスク	説明	必要なスキル
	"Sid": "ReplicateToDestinationBuckets", "Effect": "Allow", "Action": ["s3:List*", "s3:*Object", "s3:ReplicateObject", "s3:ReplicateDelete", "s3:ReplicateTags"], "Resource": ["arn:aws:s3:::destination-bucket-name*", "arn:aws:s3:::destination-bucket-name/*"] }, { "Sid": "PermissionToOverrideBucketOwner", "Effect": "Allow", "Action": [	

タスク	説明	必要なスキル
	"s3:ObjectOwnerOverrideToBucketOwner"], "Resource": ["arn:aws:s3:::destination-bucket-name*", "arn:aws:s3:::destination-bucket-name/*"] }] } このポリシーには、次の 3 つのステートメントが含まれます。 • GetSourceBucketConfiguration は、レプリケート元バケットでのレプリケーション用のレプリケーション設定とオブジェクトバージョンへのアクセスを提供します。 • ReplicateToDestinationBuckets は、レプリケート先バケットにレプリケートするためのアクセスを提供します。	

タスク	説明	必要なスキル
	配列で複数の宛先バケットを指定できます。 • `PermissionToOverrideBucketOwner` は、レプリケート先バケット`ObjectOwnerOverrideToBucketOwner` がレプリケート元アカウントからレプリケートされたレプリケート先アカウントのオブジェクトを所有できるように、 へのアクセスを提供します。 4. 次へ を選択し、 などのポリシー名を指定し`cross-account-bucket-replication-policy` 、ポリシーの作成 を選択します。 詳細については、IAM ドキュメントの「IAM ポリシーの作成」を参照してください。	

タスク	説明	必要なスキル
クロスアカウントレプリケーション用の IAM ロールを作成します。	AWS ソースアカウントで： 1. IAM コンソールで、以下の情報を含む IAM ロールを作成します。 a. [信頼できるエンティティタイプ] には、[AWS サービス] を選択します。 b. サービスで、S3 を選択します。 c. ユースケースでは、S3 バッチオペレーションを選択します。 d. 前のステップで作成したポリシーを選択します。 2. cross-account-bucket-replication-role などのロール名を指定し、ロールの作成を選択します。 詳細については、IAM ドキュメントの「IAM ロールの作成」を参照してください。	クラウド管理者、AWS 管理者

ソースアカウントにレプリケーションルールを作成する

タスク	説明	必要なスキル
ソースアカウントのソースバケットに対してレプリケーションルールを作成します。	AWS ソースアカウントで： 1. Amazon S3 コンソールを開きます。	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	2. ソースバケットに移動し、管理タブを選択します。 3. 次の設定でレプリケーションルールを作成します。 a. などのルール名を指定しますs3-replication-rule 。 b. [Status] で、[Enabled] を選択します。 c. ルールスコープで、バケット内のすべてのオブジェクトに適用を選択します。 d. 送信先 で、別のアカウントのバケットを指定 を選択し、送信先 AWS アカウント 番号とバケット名を入力します。 e. オブジェクトの所有権を送信先バケット所有者に変更するオプションを選択します。 f. IAM ロールの場合は、ソースアカウントで前に作成したロールを選択します。 g. その他のレプリケーションオプションについては、使用可能なすべてのオプションを選択します。これにより、コンテンツの迅速なレプリケー	

タスク	説明	必要なスキル
	ト、Amazon CloudWatch メトリクスによるレプリケーションの進行状況のモニタリング、削除マーカーのレプリケート、メタデータの変更のレプリケートを行うことができます。 h. [Save] を選択します。 4. 複数のレプリケート先バケットがある場合は、追加のレプリケーションルールを作成します。 詳細については、Amazon S3 ドキュメントの「レプリケート元バケットとレプリケート先バケットが異なるアカウントによって所有されている場合のレプリケーションの設定」を参照してください。	

バケットポリシーを送信先バケットに適用する

タスク	説明	必要なスキル
バケットポリシーを送信先バケットに適用します。	このステップは、送信先アカウントで AWS 送信先バケットごとに個別に実行する必要があります。 AWS 送信先アカウント：	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル
	1. Amazon S3 コンソールを開き、送信先バケットに移動し、アクセス許可タブを選択します。 2. 次の JSON コードを指定してバケットポリシーを編集し、ポリシーを保存します。	

```
{
    "Version":
 "2012-10-17",
    "Id": "PolicyFo
rDestinationBucket",
    "Statement": [
        {
            "Sid":
 "Permissions on
 objects and buckets",
            "Effect":
 "Allow",
            "Principa
l": {
                "AWS":
 "arn:aws:iam::Sour
ceAWSAccountNumber
:role/IAM-Role-cre
ated-in-step1-in-s
ource-account"
            },
            "Action": [

 "s3:List*",

 "s3:GetBucketVersi
oning",

 "s3:PutBucketVersi
oning",
```

タスク	説明	必要なスキル

	"s3:ReplicateDelete",
	"s3:ReplicateObject"
	],
	"Resource":
	[
	"arn:aws:s3:::destination-bucket",
	"arn:aws:s3:::destination-bucket/*"
	]
	},
	{
	"Sid":
	"Permission to
	override bucket owner",
	"Effect":
	"Allow",
	"Principal": {
	"AWS":
	"arn:aws:iam::SourceAWSAccountNumber:role/IAM-Role-created-in-step1-in-source-account"
	},
	"Action":
	"s3:ObjectOwnerOverrideToBucketOwner",
	"Resource": "arn:aws:s3:::destination-bucket/*"
	}
	]
	}	

タスク	説明	必要なスキル
	このポリシーには、次の 2 つのステートメントが含まれます。 • `Permissions on objects and buckets` は、レプリケート先バケットがソースアカウントで定義されたロールに基づいてコンテンツをレプリケートできることを示します。ロールは、ソースバケットにアクセス許可を付与します。 • `Permission to override bucket owner` は、レプリケート先バケットにレプリケート元アカウントの所有権を上書きするアクセス許可があることを示します。	

Amazon S3 クロスアカウントレプリケーションをテストする

タスク	説明	必要なスキル
レプリケーションが正しく機能することを確認します。	1. オブジェクトをソースバケットに追加します。 2. 新しいオブジェクトが送信先アカウントの S3 バケットに表示されることを確認します。 3. CloudWatch メトリクスを表示します。	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	a. ソースバケットで、メトリクスタブを選択します。 b. 「レプリケーションメトリクス」セクションで、レプリケーションルールを選択します。 c. [Display charts (チャートの表示)] を選択します。グラフには、レプリケーション保留中のオペレーション、レプリケーションレイテンシー、レプリケーション保留中のバイト数が表示され、レプリケーションの状態が反映されます。 詳細については、Amazon S3 ドキュメントの「Amazon Amazon CloudWatch によるメトリクスのモニタリング」を参照してください。 Amazon S3	

関連リソース

- IAM はいつ使用できますか？ (IAM ドキュメント)
- IAM の仕組み (IAM ドキュメント)
- IAM ロールの作成 (IAM ドキュメント)
- 「IAM ポリシーの作成」(IAM ドキュメント)
- アクセス管理の概要: アクセス許可とポリシー (IAM ドキュメント)

- Amazon S3 バケットの作成、設定、および操作 (Amazon S3 ドキュメント)
- Amazon S3 でのオブジェクトのアップロード、ダウンロード、および操作 (Amazon S3 ドキュメント)
- オブジェクトの複製 (Amazon S3 ドキュメント)

Amazon S3 用 AWS PrivateLink と DistCP を使用して、オンプレミスの Hadoop 環境から Amazon S3 にデータを移行する

作成者: Jason Owens (AWS)、Andres Cantor (AWS)、Jeff Klopphenstein (AWS)、Bruno Rocha" (AWS)、Samuel Schmidt (AWS)

概要

このパターンは、Apache オープンソースツール DistCp と AWS PrivateLink for Amazon Simple Storage Service (Amazon S3) を使用して、オンプレミスの Apache Hadoop 環境からAmazon Web Services (AWS) クラウドにほぼすべての量のデータを移行する方法を示しています。パブリックインターネットまたはプロキシソリューションを使用してデータを移行する代わりに、Amazon S3 用 AWS PrivateLink を使用して、オンプレミスデータセンターと Amazon Virtual Private Cloud (Amazon VPC) 間のプライベートネットワーク接続を介して Amazon S3 にデータを移行できます。Amazon Route 53 で DNS エントリを使用するか、オンプレミス Hadoop クラスターのすべてのノードで /etc/hosts ファイルにエントリを追加すると、自動的に正しいインターフェイスエンドポイントに誘導されます。

このガイドでは、DistCP を使用して AWS クラウドにデータを移行する方法について説明します。DistCp は最も一般的に使用されるツールですが、他の移行ツールも使用できます。例えば、AWS Snowball や AWS Snowmobile などのオフライン AWS ツール、または AWS Storage Gateway や AWS DataSync などのオンライン AWS ツールを使用できます。さらに、Apache NiFi のような他のオープンソースツールを使用することもできます。

前提条件と制限

前提条件

- オンプレミスデータセンターと AWS クラウドの間に Private Network 接続を持つアクティブな AWS アカウント
- Hadoop、DistCp を使用してオンプレミスにインストール
- Hadoop 分散ファイルシステム (HDFS) の移行データにアクセスできる Hadoop ユーザー
- インストールおよび設定済みの AWS コマンドラインインターフェイス (AWS CLI)
- S3 バケットにオブジェクトを入れる権限

制限

仮想プライベートクラウド (VPC) 制限は、AWS PrivateLink for Amazon S3 に適用されます。詳細については、AWS PrivateLink ガイドの「インターフェイスエンドポイントのプロパティと制限」および「AWS PrivateLink クォータ」を参照してください。

AWS PrivateLink for Amazon S3 では、以下をサポートしていません。

- 連邦情報処理規格 (FIPS) エンドポイント
- ウェブサイトエンドポイント
- レガシーグローバルエンドポイント

アーキテクチャ

ソーステクノロジースタック

- DistCP がインストールされた Hadoop クラスター

ターゲットテクノロジースタック

- Amazon S3
- Amazon VPC

ターゲットアーキテクチャ

この図は、Hadoop 管理者が DistCP を使用して、AWS Direct Connect などのプライベートネットワーク接続を介してオンプレミス環境から Amazon S3 インターフェイスエンドポイントを介して Amazon S3 にデータをコピーする方法を示しています。

ツール

AWS サービス

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

その他のツール

- Apache Hadoop DistCp (分散コピー) は、大規模なクラスター間およびクラスター内をコピーするために使用されるツールです。DistCp は、配信、エラー処理とリカバリ、およびレポート作成に Apache MapReduce を使用しています。

エピック

データを AWS クラウドに移行

タスク	説明	必要なスキル
AWS PrivateLink for Amazon S3 のエンドポイントの作成	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで [エンドポイント] を選択し、[Create endpoint (エンドポイントの作成)] を選択します。 3. [Service category] で、[AWS services] を選択します。 4. 検索ボックスに「s3」と入力し、Enter キーを押します。 5. 検索結果で、[タイプ] 列の値が [インターフェイス] である com.amazonaws.<you	AWS 管理者

タスク	説明	必要なスキル
	r-aws-region>.s3 サービス名を選択します。 6. [VPC] で、ユーザーの VPC を選択します。[Subnet] (サブネット) で、サブネットを選択します。 7. [セキュリティグループ] で、TCP 443 を許可するセキュリティグループを選択または作成します。 8. 要件に基づいてタグを追加し、[エンドポイントの作成] を選択します。	
エンドポイントを確認し、DNS エントリを見つけます。	1. Amazon VPC コンソールを開いて [エンドポイント] を選択し、先ほど作成したエンドポイントを選択します。 2. [詳細] タブで、DNS 名の最初の DNS エントリを探します。これはリージョナル DNS エントリです。この DNS 名を使用すると、アベイラビリティゾーン固有の DNS エントリが交互にリクエストされます。 3. [サブネット] タブを選択します。エンドポイントの Elastic Network Interface のアドレスは、各アベイラビリティゾーンで確認できます。	AWS 管理者

タスク	説明	必要なスキル

ファイアウォールルールとルーティング設定を確認してください。	ファイアウォールルールが開かれていて、ネットワーク設定が正しく設定されていることを確認するには、Telnet を使用してポート 443 のエンドポイントをテストします。以下に例を示します。
	
	`$ telnet vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com 443`
	
	`Trying 10.104.88.6...`
	
	`Connected to vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com.`
	
	`...`
	
	`$ telnet vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com 443`
	
	`Trying 10.104.71.141...`
	
	`Connected to vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com.`
	
	Note
	リージョンエントリを使用すると、テストに
		ネットワーク管理者、AWS 管理者

タスク	説明	必要なスキル
	成功すると、Amazon VPC コンソールの選択したエンドポイントのサブネットタブに表示される 2 つの IP アドレスが DNS で交互に切り替わっていることがわかります。	

タスク	説明	必要なスキル
名前解決の設定を変更します。	Hadoop が Amazon S3 インターフェイスエンドポイントにアクセスできるように名前解決を設定する必要があります。エンドポイント名自体を使用することはできません。代わりに、`<your-bucket-name>.s3.<your-aws-region>.amazonaws.com` または `*.s3.<your-aws-region>.amazonaws.com` を解決する必要があります。この命名制限の詳細については、「Introducing the Hadoop S3A client」(Hadoop ウェブサイト) を参照してください。 次の設定オプションのいずれかを選択します。 • オンプレミス DNS を使用してエンドポイントのプライベート IP アドレスを解決します。すべてのバケットまたは選択したバケットの動作をオーバーライドできます。詳細については、「Secure hybrid access to Amazon S3 using AWS PrivateLink」(AWS ブログ記事) の「Option 2: Access Amazon S3 using Domain Name System Response	AWS 管理者

タスク	説明	必要なスキル
	Policy Zones (DNS RPZ)」を参照してください。 • VPC のリゾルバーインバウンドエンドポイントにトラフィックを条件付きで転送するようにオンプレミス DNS を設定します。トラフィックは Route 53 に転送されます。詳細については、「Secure hybrid access to Amazon S3 using AWS PrivateLink」(AWS ブログ記事) の「Option 3: Forwarding DNS requests from on premises using Amazon Route 53 Resolver Inbound Endpoints」を参照してください。 • Hadoop クラスター内のすべてのノードの /etc/hosts ファイルを編集します。これはテスト用の一時的な解決策であり、本番環境にはお勧めしません。/etc/hosts ファイルを編集するには、`<your-bucket-name>.s3.<your-aws-region>.amazonaws.com` または `s3.<your-aws-region>.amazonaws.com` のエントリを追加します。/etc/hosts ファイルには、1 つのエントリに複数の IP	

タスク	説明	必要なスキル
	アドレスを指定することはできません。アベイラビリティゾーンのいずれかから 1 つの IP アドレスを選択する必要があります。選択すると、単一障害点になります。	

タスク	説明	必要なスキル
Amazon S3 の認証の設定。	Hadoop を使用して Amazon S3 への認証を行うには、一時的なロール認証情報を Hadoop 環境にエクスポートすることをお勧めします。詳細については、「Authenticating with S3」(Hadoop ウェブサイト) を参照してください。長時間実行されるジョブの場合は、ユーザーを作成し、S3 バケットにのみデータを保存する権限を持つポリシーを割り当てることができます。アクセスキーとシークレットキーは Hadoop に保存でき、DistCP ジョブ自体と Hadoop 管理者のみがアクセスできます。シークレットの保存について詳しくは、「Storing secrets with Hadoop Credential Providers」(Hadoop ウェブサイト) を参照してください。その他の認証方法の詳細については、AWS IAM アイデンティティセンター (AWS Single SAWS Single Sign-On の後継) の「How to get credentials of an IAM role for use with CLI access to an AWS account」を参照してください。 一時的な認証情報を使用するには、一時的な認証情報を認証情報ファイルに追加する	AWS 管理者

タスク	説明	必要なスキル

	か、以下のコマンドを実行して認証情報を環境にエクスポートします。
	
	`export AWS_SESSION_TOKEN=SECRET-SESSION-TOKEN`
	`export AWS_ACCESS_KEY_ID=SESSION-ACCESS-KEY`
	`export AWS_SECRET_ACCESS_KEY=SESSION-SECRET-KEY`
	
	従来のアクセスキーとシークレットキーの組み合わせを使用している場合は、以下のコマンドを実行します。
	
	`export AWS_ACCESS_KEY_ID=my.aws.key`
	`export AWS_SECRET_ACCESS_KEY=my.secret.key`
	
	Note
	アクセスキーとシークレットキーの組み合わせを使用する場合は、DistCp コマンドの認証情報プロバイダーを から に変更`"org.apache.hadoop.fs.s3a.TemporaryA`	

タスク	説明	必要なスキル
	WSCredentialsProvider" します"org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider" 。	

タスク	説明	必要なスキル
DistCp を使用してデータを転送します。	DistCp を使用してデータを転送するには、次のコマンドを実行します。 `hadoop distcp -Dfs.s3a.aws.credentials.provider=\"org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider" \ -Dfs.s3a.access.key="${AWS_ACCESS_KEY_ID}" \ -Dfs.s3a.secret.key="${AWS_SECRET_ACCESS_KEY}" \ -Dfs.s3a.session.token="${AWS_SESSION_TOKEN}" \ -Dfs.s3a.path.style.access=true \ -Dfs.s3a.connection.ssl.enabled=true \ -Dfs.s3a.endpoint=s3.<your-aws-region>.amazonaws.com \ hdfs:///user/root/ s3a://<your-bucket-name>` **Note** AWS PrivateLink for Amazon S3 で DistCp コマンドを使用すると、エンドポイントの AWS リージョンは	移行エンジニア、AWS 管理者

タスク	説明	必要なスキル
	自動的に検出されません。Hadoop 3.3.2 以降のバージョンでは、S3 バケットの AWS リージョンを明示的に設定するオプションを有効にすることでこの問題を解決しています。詳細については、「S3A to add option fs.s3a.endpoint.region to set AWS region」(Hadoop Web サイト) を参照してください。 その他の S3A プロバイダーの詳細については、「General S3A Client configuration」(Hadoop ウェブサイト) を参照してください。例えば、暗号化を使用する場合、暗号化の種類に応じて、上記の一連のコマンドに次のオプションを追加できます。 `-Dfs.s3a.server-side-encryption-algorithm=AES-256 [or SSE-C or SSE-KMS]`	

タスク	説明	必要なスキル
	Note S3A でインターフェイスエンドポイントを使用するには、インターフェイスエンドポイントへの S3 リージョン名 (など`s3.<your-aws-region>.amazonaws.com`) の DNS エイリアスエントリを作成する必要があります。手順については、「Amazon S3 の認証の設定」セクションを参照してください。この回避策は Hadoop 3.3.2 以前のバージョンで必要です。S3A の今後のバージョンでは、この回避策は必要なくなります。 Amazon S3 の署名に問題がある場合は、Signature Version 4 (SigV4) 署名を使用するオプションを追加します。 `-Dmapreduce.map.java.opts="-Dcom.amazonaws.services.s3.enableV4=true"`	

その他のパターン

- をインストールして IBM z/OS AWS のサービス からアクセスする AWS CLI
- CodeBuild と CloudWatch イベントを使用して、CodeCommit から Amazon S3 へのイベント駆動型バックアップを自動化する
- DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする
- Systems Manager と EventBridge を使用した SAP HANA データベースの自動バックアップ
- BMC AMI Cloud Data を使用してメインフレームデータを Amazon S3 にバックアップおよびアーカイブする
- AWS Glue を使用して Amazon S3 から Amazon Redshift にデータを段階的にロードする ETL サービスパイプラインを構築
- を使用して Amazon Bedrock でモデル呼び出しログ記録を設定する AWS CloudFormation
- Python を使用して EBCDIC データを AWS 上の ASCII に変換およびアンパックします
- Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換
- Amazon ECS タスク定義を作成し、Amazon EFS を使用して EC2 インスタンスにファイルシステムをマウントする
- で Kinesis Data Streams と Firehose を使用して DynamoDB レコードを Amazon S3 に配信する AWS CDK
- Terraform と DRA を使用して高性能データ処理用の Lustre ファイルシステムをデプロイする
- Amazon DynamoDB テーブルのストレージコストを推定
- Security Hub AWS Organizations を使用して でパブリック Amazon S3 バケットを特定する
- Amazon RDS for Oracle DB インスタンスを AMS を使用する他のアカウントに移行する
- AWS を使用してオンプレミス SFTP サーバーを に移行する AWS Transfer for SFTP
- AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行
- Rclone を使用して Microsoft Azure Blob から Amazon S3 にデータを移行する
- Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行
- AWS の大規模移行における共有ファイルシステムの移行
- AWS SFTP を使用して小規模なデータセットをオンプレミスから Amazon S3 に移行する
- 暗号化されていない Amazon Aurora インスタンスをモニタリングする
- Transfer Family を使用して、メインフレームファイルを Amazon S3 に直接移動する
- AWS Lambda オートメーションを使用して AWS アカウント から同じ の Amazon EC2 AWS Managed Microsoft AD エントリを削除する

- AWS Fargate 搭載の Amazon EKS で Amazon EFS を使用して、永続的なデータストレージでステートフルワークロードを実行する
- S3 バケットを AWS CloudFormation スタックとして正常にインポートしました
- AWS DataSync を使用して、さまざまな AWS リージョンの Amazon EFS ファイルシステム間でデータを同期する
- AWS アカウントまたは組織の EBS スナップショットの詳細を表示

デベロッパーツール

トピック

- DevOps
- インフラストラクチャ
- ウェブおよびモバイルアプリ

DevOps

トピック

- Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する
- Terraform を使用してロードバランサーエンドポイントが変更されたときの CloudFront 更新を自動化する
- GitHub Actions を使用して Python AWS CDK アプリケーションの Amazon CodeGuru レビューを自動化する
- AWS リソース評価を自動化する
- オープンソースツールを使用して SAP システムを自動的にインストール
- AWS CDK を使用して AWS Service Catalog ポートフォリオと製品のデプロイを自動化する
- CodeBuild と CloudWatch イベントを使用して、CodeCommit から Amazon S3 へのイベント駆動型バックアップを自動化する
- AWS CloudFormation スタックと関連リソースの削除を自動化する
- AWS Service Catalog と を使用して、Gitflow 環境にホットフィックスソリューションをデプロイするための動的パイプライン管理を自動化する AWS CodePipeline
- Terraform を使用して Amazon Managed Grafana での Amazon MWAA カスタムメトリクスの取り込みと視覚化を自動化する
- 自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する
- AWS CodePipeline および AWS CodeBuild を使用して、スタックセットデプロイを自動化
- Cloud Custodian と AWS CDK を使用して、Systems Manager の AWS マネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする
- AWS CDK を使用してマイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する
- DevOps プラクティスと AWS Cloud9 を使用して、マイクロサービスで疎結合アーキテクチャを構築する
- GitHub Actions と Terraform を使用して Docker イメージを構築して Amazon ECR にプッシュする
- AWS CodeCommit、AWS CodePipeline、AWS Device Farm を使用して iOS アプリを構築してテストする
- AWS CDK アプリケーションまたは CloudFormation テンプレートのベストプラクティスを cdk-nag ルールパックを使用して確認する
- Amazon EKS で実行されているアプリケーションの相互 TLS 認証を設定する

- AWS CloudFormation を使用して AppStream 2.0 リソースの作成を自動化する
- Firelens ログルーターを使用して Amazon ECS 用のカスタムログパーサーを作成する
- CodePipeline と HashiCorp Packer を使用してパイプラインと AMI を作成
- CodePipeline を使用してパイプラインを作成し、アーティファクトの更新をオンプレミスの EC2 インスタンスにデプロイ
- Java および Python プロジェクト用の動的 CI パイプラインを自動的に作成
- Terraform を使用して CloudWatch Synthetics Canary をデプロイする
- Amazon ECS に Java マイクロサービス用の CI/CD パイプラインをデプロイ
- ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation
- AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイする
- AWS CodePipeline CI/CD パイプラインで AWS Glue ジョブをデプロイする
- EC2 インスタンスプロファイルを使用して AWS Cloud9 から Amazon EKS クラスターをデプロイする
- AWS CodePipeline、AWS CodeCommit、AWS CodeBuild を使用して、複数の AWS リージョンにコードをデプロイする
- Terraform を使用して Amazon Redshift SQL クエリを実行する
- AWS Organizations 内の組織全体の AWS Backup レポートを CSV ファイルとしてエクスポートする
- Amazon EC2 インスタンスのリストのタグを CSV ファイルにエクスポートする
- Troposphere を使用して AWS Config マネージドルールを含む AWS CloudFormation テンプレートを生成します
- SageMaker ノートブックインスタンスに、別の AWS アカウントの CodeCommit リポジトリへの一時的なアクセスを許可する
- マルチアカウント DevOps 環境に GitHub Flow 分岐戦略を実装する
- マルチアカウント DevOps 環境に Gitflow 分岐戦略を実装する
- マルチアカウント DevOps 環境のトランク分岐戦略を実装する
- AWS インフラストラクチャをデプロイする前に、一元化されたカスタム Checkov スキャンを実装してポリシーを適用する
- CodeCommit 内のモノリポジトリの変更を自動的に検出し、さまざまな CodePipeline パイプラインを開始する
- AWS CloudFormation を使用して Bitbucket リポジトリを AWS Amplify と統合する

- Step Functions と Lambda プロキシ関数を使用して AWS アカウント間で CodeBuild プロジェクトを起動する
- Application Recovery Controller を使用して EMR クラスターのマルチ AZ フェイルオーバーを管理する
- AWS コードサービスと AWS KMS マルチリージョンキーを使用して、複数のアカウントとリージョンへのマイクロサービスのブルー/グリーンデプロイを管理
- AWS CloudFormation と AWS Config を使用して Amazon ECR リポジトリのワイルドカード権限をモニタリングする
- AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する
- AWS CodeCommit イベントからカスタムアクションを実行する
- GitHub Actions を使用して AWS CloudFormation テンプレートに基づいて AWS Service Catalog 製品をプロビジョニングする
- ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする
- Amazon CloudWatch メトリクスを CSV ファイルに公開
- AWS Lambda オートメーションを使用して 全体の Amazon EC2 エントリを AWS アカウント から削除 AWS Managed Microsoft AD する
- AWS Lambda オートメーションを使用して AWS アカウント から同じ の Amazon EC2 AWS Managed Microsoft AD エントリを削除する
- pytest フレームワーク AWS Glue を使用して で Python ETL ジョブのユニットテストを実行する
- Amazon S3 に Helm v3 チャートリポジトリを設定する
- AWS CodePipeline と AWS CDK を使用して CI/CD パイプラインを設定する
- cert-managerと Let's Encrypt を使用して Amazon EKS のアプリケーションのエンドツーエンド暗号化を設定
- Flux を使用して Amazon EKS マルチテナントアプリケーションのデプロイを簡素化する
- カスタムリソースを使用して複数の E メールエンドポイントを SNS トピックにサブスクライブする
- AWS Fargate WaitCondition フックコンストラクトを使用してリソースの依存関係とタスク実行を調整する
- AWS CodePipeline のサードパーティの Git ソースリポジトリを使用する
- AWS CodePipeline を使用して Terraform の構成を検証するための CI/CD パイプラインを作成する
- その他のパターン

Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する

作成者: Ishwar Chauthaiwale (AWS) と Anand Bukkapatnam Tirumala (AWS)

概要

クラウドネイティブソリューションでは、一般的なインフラストラクチャオペレーションを自動化することは、効率的で安全で費用対効果の高い環境を維持する上で重要な役割を果たします。オペレーションを手動で処理するには時間がかかり、人為的ミスが発生しやすくなります。さらに、さまざまなレベルの AWS 専門知識を持つチームメンバーは、セキュリティプロトコルへの準拠を確保しながら、これらのタスクを実行する必要があります。このパターンは、Amazon Bedrock を使用して自然言語処理 (NLP) を通じて一般的な AWS インフラストラクチャオペレーションを自動化する方法を示しています。

このパターンは、組織が複数の環境に生成 AI ベースのインフラストラクチャをデプロイするための再利用可能なモジュール式で安全なコードを開発するのに役立ちます。コードとしてのインフラストラクチャ (IaC) と自動化に重点を置くことで、バージョン管理、一貫したデプロイ、エラーの削減、プロビジョニングの高速化、コラボレーションの改善など、DevOps の主な利点を提供します。

このパターンは、チームがキーに関連するオペレーションを管理できるようにする安全なアーキテクチャを実装しています。これには、以下 AWS のサービス が含まれます。

- Amazon Simple Storage Service (Amazon S3) バケットのバージョニング管理
- Amazon Relational Database Service (Amazon RDS) スナップショットの作成
- Amazon Elastic Compute Cloud (Amazon EC2) インスタンス管理

このアーキテクチャでは、Amazon Virtual Private Cloud (Amazon VPC) エンドポイントとプライベートネットワークを使用して安全な通信を実現し、 AWS Lambda プライベートサブネット内のタスクエグゼキュターとして機能します。Amazon S3 は、データ管理を提供し、適切なアクセスコントロールを確保するために、包括的な AWS Identity and Access Management (IAM) ロールとアクセス許可を実装します。このソリューションにはチャット履歴機能はなく、チャットは保存されません。

前提条件と制限

- アクティブ AWS アカウント。

- アクセスをセキュア化および制御するために、適切なアクセスコントロール対策を実施する必要があります。アクセスコントロールの例としては AWS Systems Manager、 の使用、基盤モデルアクセス、デプロイ用の IAM ロール、サービスベースのロール、Amazon S3 バケットへのパブリックアクセスの無効化、デッドレターキューの設定などがあります。
- AWS Key Management Service（ AWS KMS) カスタマーマネージドキー。
- AWS Command Line Interface（ AWS CLI) バージョン 2 以降、デプロイ環境にインストールおよび設定済み。
- Terraform AWS Provider バージョン 4 以降がインストールされ、設定されています。
- Terraform バージョン 1.5.7 以降がインストールされ、設定されています。
- Amazon Bedrock でエージェントのアクショングループの OpenAPI スキーマを確認して定義し、不正アクセスから保護し、データの整合性を維持します。
- 必要な Amazon Titan Text Embeddings v2 と Claude 3.5 Sonnet または Claude 3 Haiku 基盤モデルのいずれか AWS アカウント に対して で有効になっているアクセス。 https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.htmlデプロイの失敗を回避するには、ターゲットデプロイ AWS リージョン が必要なモデルをサポートしていることを確認します。
- AWS Well Architected Framework のベストプラクティスに従う設定済み仮想プライベートクラウド (VPC)。
- Amazon Responsible AI ポリシーのレビューが完了しました。

製品バージョン

- Amazon Titan Text Embeddings v2
- Anthropic Claude 3.5 Sonnet または Claude 3 Haiku
- Terraform AWS プロバイダーバージョン 4 以降
- Terraform バージョン 1.5.7 以降

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

ソリューションアーキテクチャは、自然言語リクエストを処理し、対応する AWS オペレーションを実行するために連携する複数のレイヤーで構成されます。

1. ユーザーは Amazon Bedrock チャットコンソールからオペレーションリクエストを行います。
2. チャットボットはリクエスト処理に Amazon Bedrock ナレッジベースを使用します。自然言語処理用の Amazon Titan Text Embeddings v2 モデルを実装しています。
3. ユーザープロンプトにアクションリクエストが含まれている場合、Amazon Bedrock アクショングループは実行ロジックに Anthropic Claude 3 Haiku または Claude 3.5 Sonnet モデル (選択に応じて) のいずれかを使用し、OpenAPI スキーマを介してオペレーションを定義します。
4. アクショングループは、安全なサービス通信 AWS PrivateLink のために を使用して Amazon VPC エンドポイントに到達します。
5. AWS Lambda 関数は、Amazon Bedrock サービスの Amazon VPC エンドポイントを介して到達します。
6. Lambda 関数は、プライマリ実行エンジンです。リクエストに基づいて、Lambda 関数は API を呼び出して に対してアクションを実行します AWS のサービス。Lambda 関数は、オペレーションのルーティングと実行も処理します。
7. Lambda 関数から API リクエスト AWS のサービス を取得すると、対応するオペレーションが実行されます。
8. Lambda 関数は、Amazon Bedrock によって理解される出力ペイロードを計算します。
9. このペイロードは、安全なサービス通信のために PrivateLink を使用して Amazon Bedrock に送信されます。Amazon Bedrock で使用される大規模言語モデル (LLM) は、このペイロードを理解し、人間が理解できる形式に変換します。
10. その後、出力は Amazon Bedrock チャットコンソールにユーザーに表示されます。

このソリューションでは、次のプライマリオペレーションを有効にします。

- Amazon S3 – バージョン管理のためにバケットのバージョニングを有効にします。
- Amazon RDS – バックアップ用のデータベーススナップショットを作成します。
- Amazon EC2 – インスタンスを一覧表示し、インスタンスの開始と停止を制御します。

ツール

AWS のサービス

- Amazon Bedrock は、主要な AI スタートアップと Amazon の高性能基盤モデル (FMs) を統一された API を通じて使用できるようにするフルマネージドサービスです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を許可するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon OpenSearch Serverless は、Amazon OpenSearch Service のオンデマンドサーバーレス設定です。
- AWS PrivateLink は、仮想プライベートクラウド (VPCs) から VPC 外のサービスへの単方向のプライベート接続を作成するのに役立ちます。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間が短縮され、 AWS リソースを大規模に安全に管理できます。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

その他のツール

- Git はオープンソースの分散バージョン管理システムです。
- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンのコードは、GitHub aws-samples/infra-ops-orchestrator リポジトリで入手できます。

ベストプラクティス

- Lambda 実行ログを定期的にモニタリングします。詳細については、「Lambda 関数をモニタリングおよびトラブルシューティングする」を参照してください。ベストプラクティスの詳細については、「 AWS Lambda 関数を使用するためのベストプラクティス」を参照してください。
- セキュリティ設定を定期的に見直して、組織の要件に準拠していることを確認します。詳細については、「セキュリティのベストプラクティス」を参照してください。
- 最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

解決策をデプロイする

タスク	説明	必要なスキル

リポジトリをクローン作成します。	ローカルマシンでリポジトリのクローンを作成するには、次のコマンドを実行します。
	`git clone "git@github.com:aws-samples/infra-ops-orchestrator.git"`
	`cd infra-ops-orchestrator`
		AWS DevOps、DevOps エンジニア

環境変数を編集します。	クローンされたリポジトリのルートディレクトリで `terraform.tfvars` ファイルを編集します。で示されているプレースホルダーを確認	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	し[XXXXX]、環境に応じて更新します。	

インフラストラクチャを作成します。	インフラストラクチャを作成するには、次のコマンドを実行します。
	`terraform init`
	`terraform plan`
	実行計画を慎重に確認してください。計画された変更が許容できる場合は、次のコマンドを実行します。
	`terraform apply --auto-approve`
		AWS DevOps、DevOps エンジニア

ソリューションにアクセスする

タスク	説明	必要なスキル
ソリューションにアクセスします。	デプロイが成功したら、次のステップに従ってチャットベースのインターフェイスを使用します。 1. Infrastructure Orchestrator Assistant にアクセスするには、Amazon Bedrock のアクセス許可を持つ IAM ロール AWS Management Console を使用して にサ	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	インインし、https://https://console.aws.amazon.com/bedrock/://www..com で Amazon Bedrock コンソールを開きます。左側のナビゲーションペインで [エージェント] を選択します。次に、「 エージェント」セクションの「インフラストラクチャオーケストレーターアシスタント」を選択します。 2. 以下の提案については、ターゲットリソースが AWS 環境に存在することを確認し、次のオペレーション例を試してください。 • 「RDS インスタンスのスナップショットを作成する [instance-name]」と尋ねて、Amazon RDS インスタンスのスナップショットバックアップを作成します。 • 「バケットのバージョニングを有効にする [bucket-name]」と尋ねて、Amazon S3 バケットでバージョニングを有効にします。 • 「すべての Amazon EC2 インスタンスを一覧表示する」と尋ねて Amazon	

タスク	説明	必要なスキル
	EC2 インスタンスを一覧表示します。 • Amazon EC2インスタンスの開始 [instance-id]」またはEC2 インスタンスの停止 [instance-id]」を尋ねて、Amazon EC2 インスタンスを開始または停止します。 注: 括弧内の値は、 AWS 環境の実際のリソース名または IDsに置き換えてください。	

リソースをクリーンアップする

タスク	説明	必要なスキル

作成したリソースを削除します。	このパターンで作成されたすべてのインフラストラクチャを削除するには、次のコマンドを実行します。
	`terraform plan -destroy`
	破棄計画を慎重に確認してください。計画された削除が許容できる場合は、次のコマンドを実行します。
	`terraform destroy`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	注: このコマンドは、このパターンによって作成されたすべてのリソースを完全に削除します。コマンドは、リソースを削除する前に確認を求められます。	

トラブルシューティング

問題	ソリューション
エージェントの動作	この問題の詳細については、Amazon Bedrock ドキュメントの「エージェントの動作のテストとトラブルシューティング」を参照してください。
Lambda ネットワークの問題	これらの問題の詳細については、Lambda ドキュメントの「Lambda でのネットワーク問題のトラブルシューティング」を参照してください。
IAM 許可	これらの問題の詳細については、IAM ドキュメントの「IAM のトラブルシューティング」を参照してください。

関連リソース

- Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成
- Amazon Bedrock でエージェントのアクショングループの OpenAPI スキーマを定義する
- バケットでのバージョニングの有効化
- Amazon Bedrock エージェントの仕組み
- Amazon Bedrock ナレッジベースを使用してデータを取得し、AI レスポンスを生成する
- 経由でサービスに安全にアクセスする AWS PrivateLink

- Amazon EC2 インスタンスの停止と起動
- アクショングループを使用して、エージェントが実行するアクションを定義する

Terraform を使用してロードバランサーエンドポイントが変更されたときの CloudFront 更新を自動化する

作成者: Tamilselvan P (AWS)、Mohan Annam (AWS)、および Naveen Suthar (AWS)

概要

Amazon Elastic Kubernetes Service (Amazon EKS) のユーザーが Helm チャートを使用してイングレス設定を削除して再インストールすると、新しい Application Load Balancer (ALB) が作成されます。Amazon CloudFront は古い ALB の DNS レコードを参照し続けるため、この問題が発生します。その結果、このエンドポイント宛てのサービスは到達できません。(この問題のあるワークフローの詳細については、「追加情報」を参照してください。)

この問題を解決するために、このパターンでは Python で開発されたカスタム AWS Lambda 関数の使用について説明します。この Lambda 関数は、Amazon EventBridge ルールを使用して新しい ALB がいつ作成されたかを自動的に検出します。を使用して AWS SDK for Python (Boto3)、関数は新しい ALB の DNS アドレスで CloudFront 設定を更新し、トラフィックが正しいエンドポイントにルーティングされるようにします。

この自動化されたソリューションは、追加のルーティングやレイテンシーなしでサービスの継続性を維持します。このプロセスは、基盤となるインフラストラクチャが変更されても、CloudFront が常に正しい ALB DNS エンドポイントを参照するのに役立ちます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Helm を使用して Amazon EKS にデプロイされるテストと検証用のサンプルウェブアプリケーション。詳細については、Amazon EKS ドキュメントの「Amazon EKS に Helm を使用してアプリケーションをデプロイする」を参照してください。
- Helm イングレスコントローラーによって作成された ALB に呼び出しをルーティングするように CloudFront を設定します。詳細については、Amazon EKS ドキュメントの「Helm を使用した Install AWS Load Balancer Controller」および CloudFront ドキュメントの「Application Load Balancer へのアクセスの制限」を参照してください。
- Terraform がローカルワークスペースにインストールおよび設定されている。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Terraform バージョン 1.0.0 以降
- Terraform AWS Provider バージョン 4.20 以降

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

このソリューションでは、次のステップを実行します。

1. Amazon EKS Ingress Controller は、Helm の再起動またはデプロイが行われるたびに、新しい Application Load Balancer (ALB) を作成します。
2. EventBridge は ALB 作成イベントを検索します。
3. ALB 作成イベントは Lambda 関数をトリガーします。
4. Lambda 関数は python 3.9 に基づいてデプロイされており、boto3 API を使用して を呼び出します AWS のサービス。Lambda 関数は、ロードバランサーの作成イベントから受信した最新のロードバランサー DNS 名で CloudFront エントリを更新します。

ツール

AWS のサービス

- Amazon CloudFront は、世界中のデータセンターネットワークを通じて配信することで、ウェブコンテンツの配信を高速化します。これにより、レイテンシーが減少し、パフォーマンスが向上します。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。

- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。
- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンのコードは、GitHub aws-cloudfront-automation-terraform-samples リポジトリで入手できます。

エピック

ローカルワークステーションをセットアップする

タスク	説明	必要なスキル
Git CLI をセットアップおよび設定します。	ローカルワークステーションに Git コマンドラインインターフェイス (CLI) をインストールして設定するには、Git ドキュメントの「開始方法 – Git のインストール」の手順に従います。	DevOps エンジニア
プロジェクトフォルダを作成し、ファイルを追加します。	1. パターンの GitHub リポジトリに移動し、コードボタンを選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	2. クローンダイアログで、HTTPS タブを選択します。ウェブ URL を使用してクローンを作成するで、表示される URL をコピーします。 3. ローカルマシンにフォルダを作成します。プロジェクト名で名前を付けます。 4. ローカルマシンでターミナルを開き、このフォルダに移動します。 5. このパターンの git リポジトリのクローンを作成するには、次のコマンドを実行します。 `git clone https://github.com/aws-samples/aws-cloudfront-automation-terraform-samples` 6. リポジトリのクローンを作成したら、次のコマンドを使用してクローンディレクトリに移動します。 `cd <directory name>/cloudfront-update` このプロジェクトを任意の統合開発環境 (IDE) で開きます。	

Terraform 設定を使用してターゲットアーキテクチャをプロビジョニングする

タスク	説明	必要なスキル

ソリューションをデプロイします。	ターゲットにリソースをデプロイするには AWS アカウント、次のステップを使用します。
	
	1. `cloudfront-update` フォルダに移動します。
	2. で `terraform.tfvars` ファイルを更新しますcloudfront_distribution_id 。
	3. AWS プロファイル AWS リージョン の を設定するには、次のコマンドを実行します。
	
	`export AWS_REGION={{ REGION }}`
	
	4. Terraform を初期化するには、次のコマンドを実行します。
	
	`terraform init`
	
	5. Terraform を検証するには、次のコマンドを実行します。
	
	`terraform validate`
	
	6. Terraform 実行プランを作成するには、次のコマンドを実行します。
		DevOps エンジニア

タスク	説明	必要なスキル

	`terraform plan`
	7. からアクションを適用するにはterraform plan、次のコマンドを実行します。
	`terraform apply`	

デプロイメントを確認する

タスク	説明	必要なスキル
デプロイを検証します。	1. にサインイン AWS Management Console し、Amazon CloudFront コンソールを https://console.aws.amazon.com/cloudfront/v4/home:// www.com で開きます。 2. 左側のナビゲーションペインで、ディストリビューションを選択し、CloudFront ディストリビューションを開きます。 3. オリジンタブで、オリジン名とオリジンマッピングに更新された ALB DNS レコードがあることを確認します。	DevOps エンジニア

インフラストラクチャをクリーンアップする

タスク	説明	必要なスキル
インフラストラクチャをクリーンアップします。	前に作成したインフラストラクチャをクリーンアップするには、次の手順を実行します。 1. 次のコマンドを実行します。`terraform destroy` 2. destroy コマンドを確認するには、 と入力しますyes。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
プロバイダー認証情報の検証中にエラーが発生しました。	ローカルマシンから Terraform applyまたは destroy コマンドを実行すると、次のようなエラーが発生することがあります。 `Error: configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: operation error STS: GetCallerIdentity, https response error StatusCode: 403, RequestID: 123456a9-fbc1-40ed-b8d8-513d0133ba7f, api error InvalidClientTokenId: The security token included in the request is invalid.` このエラーは、ローカルマシンの設定で使用されている認証情報のセキュリティトークンの有効期限が切れていることが原因です。

問題	ソリューション
	エラーを解決するには、 AWS Command Line Interface （ AWS CLI) ドキュメントの「設定の設定と表示」を参照してください。

関連リソース

AWS リソース

- Application Load Balancer へのアクセスを制限する
- AWS Load Balancerコントローラーを使用してインターネットトラフィックをルーティングする

Terraformのドキュメント

- AWS プロバイダー
- 「Terraform のインストール」
- リモート状態

追加情報

問題のあるワークフロー

この図表は、次のワークフローを示しています：

1. ユーザーがアプリケーションにアクセスすると、呼び出しは CloudFront に送信されます。
2. CloudFront は、呼び出しをそれぞれの Application Load Balancer (ALB) にルーティングします。
3. ALB には、アプリケーションポッドの IP アドレスであるターゲット IP アドレスが含まれます。そこから、ALB はユーザーに期待される結果を提供します。

ただし、このワークフローは問題を示しています。アプリケーションのデプロイは Helm チャートを通じて行われます。デプロイがある場合、または誰かが Helm を再起動すると、それぞれの進入も再作成されます。その結果、外部ロードバランサーコントローラーは ALB を再作成します。また、再作成のたびに、ALB は別の DNS 名で再作成されます。このため、CloudFront にはオリジン設定に

古いエントリがあります。この古いエントリのため、アプリケーションにはユーザーが到達できません。この問題により、ユーザーのダウンタイムが発生します。

代替ソリューション

もう 1 つの可能な解決策は、ALB の外部 DNS を作成し、CloudFront の Amazon Route 53 プライベートホストゾーンエンドポイントを指すことです。ただし、このアプローチではアプリケーションフローに別のホップが追加されるため、アプリケーションのレイテンシーが発生する可能性があります。このパターンの Lambda 関数ソリューションは、現在のフローを中断しません。

GitHub Actions を使用して Python AWS CDK アプリケーションの Amazon CodeGuru レビューを自動化する

作成者: Vanitha Dontireddy (AWS) と Sarat Chandra Pothula (AWS)

概要

このパターンは、GitHub Actions を通じてオーケストレーションされた Python AWS Cloud Development Kit (AWS CDK) アプリケーションの Amazon CodeGuru 自動コードレビューの統合を示しています。このソリューションは、Python で定義されたサーバーレスアーキテクチャ AWS CDK をデプロイします。開発パイプライン内のエキスパートコード分析を自動化することで、このアプローチは Python AWS CDK プロジェクトに対して次のことを実行できます。

- コードの品質を向上させます。
- ワークフローを合理化します。
- サーバーレスコンピューティングの利点を最大化します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface （ AWS CLI) バージョン 2.9.11 以降、インストールおよび設定済み。
- PR ワークフローが正しく動作するように、アクティブな GitHub アカウントと、読み取りおよび書き込みワークフローのアクセス許可と GitHub Actions によるプルリクエスト (PR) の作成を持つ GitHub リポジトリ。
- ソリューションを にデプロイするための GitHub Actions の OpenID Connect (OIDC) ロール AWS アカウント。ロールを作成するには、 AWS CDK コンストラクトを使用します。

制約事項

- Amazon CodeGuru Profiler は、すべての Java 仮想マシン (JVM) 言語 (Scala や Kotlin など) で記述されたアプリケーション、ランタイム、Python 3.6 以降をサポートしています。
- Amazon CodeGuru Reviewer は、、Bitbucket AWS CodeCommit、GitHub、GitHub Enterprise Cloud、および GitHub GitHub Enterprise Server のソースプロバイダーからのみ、Java および

Python コードリポジトリとの関連付けをサポートします。さらに、Amazon Simple Storage Service (Amazon S3) リポジトリは GitHub Actions を通じてのみサポートされます。

- 継続的インテグレーションと継続的デプロイ (CI/CD) パイプライン中に結果を自動的に出力する方法はありません。代わりに、このパターンでは GitHub Actions を代替方法として使用して、検出結果を処理および表示します。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、このソリューションのアーキテクチャを示しています。

図に示すように、開発者がレビュー用のプルリクエスト (PR) を作成すると、GitHub Actions は次のステップをトリガーします。

1. IAM ロールの前提 – パイプラインは、GitHub Secrets で指定された IAM ロールを使用してデプロイタスクを実行します。
2. コード分析
   - CodeGuru Reviewer は、Amazon S3 バケットに保存されているコードを分析します。欠陥を特定し、修正と最適化に関する推奨事項を提供します。
   - CodeGuru Security は、ポリシー違反と脆弱性をスキャンします。
3. 結果のレビュー
   - パイプラインは、コンソール出力の結果ダッシュボードへのリンクを出力します。
   - 重要な検出結果が検出されると、パイプラインはすぐに失敗します。
   - 重要度が高い、正常、または低い検出結果の場合、パイプラインは次のステップに進みます。
4. PR の承認
   - レビュー担当者は、PR を手動で承認する必要があります。
   - PR が拒否されると、パイプラインは失敗し、以降のデプロイステップが停止します。
5. CDK デプロイ — PR の承認後、CDK デプロイプロセスが開始されます。以下の AWS のサービス および リソースが設定されます。

- CodeGuru Profiler
- AWS Lambda 関数
- Amazon Simple Queue Service (Amazon SQS) キュー

6. データ生成のプロファイリング – CodeGuru Profiler 用に十分なプロファイリングデータを生成するには：
   - パイプラインは、Amazon SQS キューに定期的にメッセージを送信することで、Lambda 関数を複数回呼び出します。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- CDK Toolkit は、 AWS CDK アプリの操作に役立つコマンドラインクラウド開発キットです。
- Amazon CodeGuru Profiler は、ライブアプリケーションからランタイムパフォーマンスデータを収集し、アプリケーションのパフォーマンスを微調整する上で役立つ推奨事項を提供します。
- Amazon CodeGuru Reviewer は、プログラム分析と機械学習を使用して、開発者が見つけるのが難しい潜在的な欠陥を検出します。次に、CodeGuru Profiler は Java および Python コードを改善するための提案を提供します。
- Amazon CodeGuru Security は、機械学習を使用してセキュリティポリシー違反と脆弱性を検出する静的アプリケーションセキュリティツールです。セキュリティリスクに対処するための提案を提供し、アプリケーションのセキュリティ体制を追跡できるようにメトリクスを生成します。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Queue Service (Amazon SQS) 」は、安全で耐久性があり、配信ソフトウェアシステムとコンポーネントを統合および分離できる利用可能なホスト型キューを提供します。
- 「Amazon Simple Storage Service (Amazon S3) 」は、どのようなデータの量であっても、保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- GitHub Actions は、GitHub リポジトリと緊密に統合された継続的インテグレーションおよび継続的デリバリー (CI/CD) プラットフォームです。GitHub Actions を使用して、ビルド、テスト、デプロイパイプラインを自動化できます。

コードリポジトリ

このパターンのコードは、GitHub amazon-codeguru-suite-cdk-python リポジトリで入手できます。

ベストプラクティス

- を使用してクラウドインフラストラクチャを開発およびデプロイするためのベストプラクティス AWS CDKに従ってください。
- AWS のサービス GitHub Actions ワークフローで を使用する場合は、IAM のセキュリティのベストプラクティスに従います。
  - リポジトリコードに認証情報を保存しないでください。
  - 一時的な認証情報を受け取る IAM ロールを引き受け、可能であれば一時的な認証情報を使用します。
  - GitHub Actions ワークフローで使用される IAM ロールに最小特権を付与します。GitHub Actions ワークフローでアクションを実行するために必要なアクセス許可のみを付与します。
  - GitHub Actions ワークフローで使用される IAM ロールのアクティビティをモニタリングします。
  - 使用する長期的な認証情報は定期的にローテーションします。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
AWS 認証情報を設定します。	スタックをデプロイする AWS アカウント および AWS リージョン を定義する変数をエクスポートするには、次のコマンドを実行します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	`export CDK_DEFAU LT_ACCOUNT=<12-digit AWS account number>`
	
	`export CDK_DEFAU LT_REGION=<AWS Region>`
	
	の AWS 認証情報 AWS CDK は、環境変数を通じて提供されます。	

リポジトリをクローン作成します。	ローカルマシンでリポジトリのクローンを作成するには、次のコマンドを実行します。 `git clone https://g ithub.com/aws-samp les/amazon-codeguru- suite-cdk-python.git`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

CDK Toolkit をインストールします。	CDK Toolkit がインストールされていることを確認し、バージョンを確認するには、次のコマンドを実行します。
	`cdk --version`
	CDK Toolkit のバージョンが 2.27.0 より前の場合は、次のコマンドを入力してバージョン 2.27.0 に更新します。
	`npm install -g aws-cdk@2.27.0`
	CDK Toolkit がインストールされていない場合は、次のコマンドを実行してインストールします。
	`npm install -g aws-cdk@2.27.0 --force`
		AWS DevOps、DevOps エンジニア

必要な依存ファイルをインストールします。	必要なプロジェクトの依存関係をインストールするには、次のコマンドを実行します。
	`python -m pip install --upgrade pip`
	`pip install -r requirements.txt`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

CDK 環境をブートストラップします。	AWS CDK 環境をブートストラップするには、次のコマンドを実行します。
	`npm install`
	`npm run cdk bootstrap "aws://${ACCOUNT_NUMBER}/${AWS_REGION}"`
	環境を正常にブートストラップすると、次の出力が表示されます。
	`# Bootstrapping environment aws://{account}/{region}...`
	`# Environment aws://{account}/{region} bootstrapped`
		AWS DevOps、DevOps エンジニア

CDK アプリをデプロイする

タスク	説明	必要なスキル
AWS CDK アプリケーションを合成します。	AWS CDK アプリケーションを合成するには、次のコマンドを実行します。 `cdk synth` このコマンドの詳細については、 AWS CDK ドキュメントの「cdk 合成」を参照してください。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
リソースのデプロイ	リソースをデプロイするには、次のコマンドを実行します。 `cdk deploy --require-approval never` **Note** `--require-approval never` フラグは、CDK がすべての変更を自動的に承認して実行することを意味します。これには、CDK が通常手動レビューを必要とするものとしてフラグ付けする変更 (IAM ポリシーの変更やリソースの削除など) が含まれます。本番環境で `--require-approval never`フラグを使用する前に、CDK コードと CI/CD パイプラインが十分にテストされ、安全であることを確認してください。	AWS DevOps、DevOps エンジニア

GitHub シークレットと個人用アクセストークンを作成する

タスク	説明	必要なスキル
GitHub で必要なシークレットを作成します。	GitHub Actions ワークフローがリポジトリのコードに機密情報を公開することなく AWS リソースに安全にアクセスできるようにするには、シークレットを作成します。`ROLE_TO_ASSUME` 、、の GitHub でシークレットを作成するには`CodeGuruReviewArtifactBucketName` `AWS_ACCOUNT_ID` 、GitHub Actions ドキュメントの「リポジトリのシークレットの作成」の手順に従います。 変数の詳細については、以下を参照してください。 • `AWS_ACCOUNT_ID` – パイプラインが実行される AWS アカウント ID。 • `CodeGuruReviewArtifactBucketName` – CodeGuru Reviewer アーティファクトが保存されている S3 バケットの名前。このパターンでは、バケット名 を使用します`codeguru-reviewer-build-artifacts-<ACCOUNT_ID>-<REGION>` 。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	• AWS_REGION – リソース AWS リージョン が配置されている 。 • ROLE_TO_ASSUME – パイプラインが引き受ける IAM ロールの名前。このパターンでは、ロール名を使用しますgithubActionsDeployRole 。	
GitHub 個人用アクセストークンを作成します。	GitHub Actions ワークフローが GitHub を認証して操作するための安全な方法を設定するには、以下を実行します。 1. リポジトリへの読み取りおよび書き込みアクセス権を持つ GitHub 個人用アクセストークンを作成するには、GitHub ドキュメントの「個人用アクセストークンの管理」の手順に従います。 2. このトークンを GitHub Actions のリポジトリシークレットとして保存するには、GitHub Actions ドキュメントの「リポジトリのシークレットの作成」の手順に従います。	AWS DevOps、DevOps エンジニア

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップします。	Python AWS CDK アプリケーションをクリーンアップするには、次のコマンドを実行します。 `cdk destroy --all`	DevOps エンジニア

トラブルシューティング

問題	ソリューション
ダッシュボードの検出結果へのリンクを表示します。	CI/CD パイプライン中に結果を出力する方法はありません。代わりに、このパターンでは GitHub Actions を代替方法として使用して、検出結果を処理および表示します。

関連リソース

AWS リソース

- AWS クラウド開発キット
- Amazon CodeGuru ドキュメント
- Amazon S3
- AWS Identity and Access Management
- Amazon Simple Queue Service
- とは AWS Lambda

GitHub ドキュメント

- Amazon Web Services での OpenID Connect の設定

- GitHub Actions
- ワークフローの再利用
- ワークフローのトリガー

AWS リソース評価を自動化する

作成者: Naveen Suthar (AWS)、Arun Bagal (AWS)、Manish Garg (AWS)、Sandeep Gawande (AWS)

概要

このパターンでは、AWS Cloud Development Kit (AWS CDK) を使用してリソース評価機能を設定する自動化アプローチを説明します。このパターンを使用すると、運用チームはリソース監査の詳細を自動的に収集し、AWS アカウントにデプロイされたすべてのリソースの詳細を単一のダッシュボードに表示できます。このパターンは、以下に示すユースケースで役立ちます。

- infrastructure as code (IaC) ツールを特定し、HashiCorp Terraform、AWS CloudFormation、AWS CDK、AWS コマンドラインインターフェイス (AWS CLI) など、さまざまな IaC ソリューションによって作成されたリソースを孤立化します。
- リソース監査情報を取得する

このソリューションは、リーダーシップチームが単一ダッシュボードから AWS アカウントのリソースとアクティビティに関する洞察を得る上でも役立ちます。

> **Note**
>
> Amazon QuickSight は有料サービスです。実行してデータを分析しダッシュボードを作成する前に、Amazon QuickSight の価格設定を確認します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Identity and Access Management (IAM) ロールとプロビジョンリソースへのアクセス権限
- Amazon Simple Storage Service (Amazon S3) と Amazon Athena へのアクセス権で作成された Amazon QuickSight アカウント
- AWS CDK バージョン 2.55.1 以降がインストールされている
- Python 3.9 以降がインストールされている

制限

- このソリューションは単一 AWS アカウントにデプロイされます。
- このソリューションは、AWS CloudTrail がすでにセットアップされ、S3 バケットにデータを保存していない限り、デプロイ前に発生したイベントを追跡しません。

製品バージョン

- AWS CDK バージョン 2.55.1 またはそれ以降
- Python バージョン 3.9 以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Athena
- AWS CloudTrail
- AWS Glue
- AWS Lambda
- Amazon QuickSight
- Amazon S3

ターゲットアーキテクチャ

AWS CDK コードは、AWS アカウントへのリソース評価機能の設定に必要なすべてのリソースをデプロイします。次の図は、CloudTrail ログを AWS Glue、Amazon Athena と QuickSight に送信するプロセスを示しています。

1. CloudTrail は、ストレージ用にログを S3 バケットに送信します。
2. イベント通知は、ログを処理してフィルタリングされたデータを生成する Lambda 関数を呼び出します。
3. フィルタリングされたデータは、別の S3 バケットに保存されます。
4. S3 バケット内のフィルタリングされたデータに AWS Glue クローラーを設定して、AWS Glue データカタログテーブルにスキーマを作成します。

5. フィルタリングされたデータを Amazon Athena によってクエリする準備ができました。
6. クエリされたデータには、QuickSight が視覚化のためにアクセスします。

自動化とスケール

- AWS Organizations に組織全体の CloudTrail トレイルがある場合は、このソリューションを 1 つの AWS アカウントから複数の AWS アカウントに拡張できます。CloudTrail を組織レベルでデプロイすることで、このソリューションを使用して、必要なすべてのリソースのリソース監査詳細を取得することもできます。
- このパターンでは、AWS サーバーレスリソースを使用してソリューションをデプロイします。

ツール

AWS サービス

- Amazon Athena はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接分析できます。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングをし、AWS アカウントと AWS リージョンにわたってライフサイクル全体のリソースを管理できます。
- AWS CloudTrail は、AWS アカウントのガバナンス、コンプライアンス、運用面のリスクの監査をサポートします。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの確実な分類、整理、強化、移動をサポートできます。このパターンでは、AWS Glue クローラーと AWS Glue データカタログテーブルを使用します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon QuickSight は、単一ダッシュボードでデータを視覚化、分析、レポートできるクラウドスケールのビジネスインテリジェンス (BI) サービスです。
- Amazon Simple Storage Service (Amazon S3) は、データ量にかかわらず、保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このパターンのコードは、GitHub 内の「infrastructure-assessment-iac-automation」リポジトリで利用できます。

コードリポジトリには、以下のファイルとフォルダが含まれます。

- `lib` フォルダ — AWS CDK は、AWS リソースの作成に使用される Python ファイルをコンストラクトします。
- `src/lambda_code` — Lambda 関数で実行される Python コード
- `requirements.txt` — インストールする必要があるすべての Python 依存関係のリスト
- `cdk.json` — リソースの起動に必要な値を提供する入力ファイル

ベストプラクティス

Lambda 関数のモニタリングとアラートを設定します。詳細については、Lambda 関数をモニタリングおよびトラブルシューティングするを参照してください。Lambda 関数を使用する際の一般的なベストプラクティスについては、AWS ドキュメントを参照してください。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
ローカルマシンにリポジトリを複製します。	リポジトリを複製するには、`git clone https://github.com/aws-samples/infrastructure-assessment-iac-automation.git` コマンドを実行します。	AWS DevOps、DevOps エンジニア
Python 仮想環境を設定し、必要な依存関係をインストールします。	Python 仮想環境をセットアップするには、次のコマンドを実行します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	`cd infrastructure-assessment-iac-automation`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	
	必要な依存関係を設定するには、`pip install -r requirements.txt` コマンドを実行します。	

AWS CDK 環境をセットアップし、AWS CDK コードを合成します。	1. AWS アカウントで AWS CDK 環境を設定するには、`cdk bootstrap aws://ACCOUNT-NUMBER/REGION` コマンドを実行します。 2. コードを AWS CloudFormation スタック設定に変換するには、`cdk synth` コマンドを実行します。	AWS DevOps、DevOps エンジニア

ローカルマシンで AWS 認証情報をセットアップする

タスク	説明	必要なスキル
スタックをデプロイするアカウントとリージョンの変数をエクスポートします。	環境変数を使用して AWS CDK の AWS 認証情報を提供には、次のコマンドを実行します。 `export CDK_DEFAULT_ACCOUNT=<12 Digit AWS Account Number>`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`export CDK_DEFAU LT_REGION=<region>`	
AWS CLI プロファイルをセットアップします。	アカウントの AWS CLI プロファイルを設定するには、AWS ドキュメントの指示に従います。	AWS DevOps、DevOps エンジニア

リソース評価ツールを設定してデプロイする

タスク	説明	必要なスキル
アカウントにリソースをデプロイします。	AWS CDK を使用して AWS アカウントにリソースをデプロイするには、以下を実行します。 1. クローンしたリポジトリのルートにある `cdk.json` ファイルに、以下のパラメータを入力します。 • `s3_context` • `ct_context` • `kms_context` • `lambda_context` • `glue_context` • `qs_context` これらの値は、リソースの設定と命名法を定義します。デフォルト値が設定されており、必要に応じて変更できます。	AWS DevOps

タスク	説明	必要なスキル
	Note S3 バケットが既に存在するというエラーを回避するには、`ct` および `output` セクションで `s3_context` に一意の名前を指定してください。 2. リソースをデプロイするには、`cdk deploy` コマンドを実行します。 `cdk deploy` コマンドは、ログイベントに CloudTrail リソースを作成し、ログファイルを入力 S3 バケットに保存します。トレイルのログファイルは Lambda 関数によって処理されます。フィルタリングされた結果は出力 S3 バケットに保存され、Amazon Athena と Amazon QuickSight が使用する準備ができます。	

タスク	説明	必要なスキル
AWS Glue クローラーを実行し、データカタログテーブルを作成します。	AWS Glue クローラーは、データスキーマを動的に保つために使用されます。このソリューションは、AWS Glue クローラースケジューラーの定義に従ってクローラーを定期的に実行することで、AWS Glue データカタログテーブルのパーティションを作成および更新します。出力 S3 バケットでデータが使用できるようになると、次のステップを使用して AWS Glue クローラーを実行し、テスト用の Data Catalog テーブルスキーマを作成します。 1. AWS マネジメントコンソールにサインインし、AWS Glue コンソールにナビゲートします。 2. ナビゲーションペインの [Data Catalog (データカタログ)] で、[Crawlers (クローラー)] を選択します。 3. `iac-tool-qa-resource-iac-json-crawler` クローラーを選択します。 4. クローラーを実行します。 5. クローラーが正常に実行されると、AWS Glue データカタログテーブルが作成されます。AWS QuickSight	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	はこのテーブルを使用してデータを視覚化します。 **Note** AWS CDK コードは、AWS Glue クローラを特定の時間に実行するように設定しますが、オンデマンドで実行することもできます。	
QuickSight コンストラクトをデプロイします。	1. QuickSight コンストラクトをデプロイするには、`resource_iac_tool_stack.py` の`#QuickSight setup - start` と`#QuickSight setup – ends` 間のコードのコメントを解除します。 2. コメントを解除したら、`cdk deploy` コマンドを実行して QuickSight アカウントで `QuickSight DataSource` と `QuickSight DataSet` を作成します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
QuickSight ダッシュボードを作成します。	QuickSight ダッシュボードと分析の例を作成するには、以下を実行します。 1. QuickSight コンソールに移動し、リソースがデプロイされている AWS リージョンを選択します。 2. ナビゲーションペインで、[データセット] を選択し、`ct-operations-iac-ds` という名前のデータセットが Amazon QuickSight データセットに作成されていることを確認します。 データセットが表示されない場合は、QuickSight コンストラクトを再デプロイします。 3. `ct-operations-iac-ds` データセットを選択し、[USE IN ANALYSIS (分析で使用)] を選択します。 4. デフォルトシートを選択します。 5. 左側のフィールドリストからそれぞれの列を選択します。 6. 必要な列を選択したら、適切なビジュアルタイプを選択してデータを表示します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	詳細については、Amazon QuickSight での分析の開始および Amazon QuickSight でのビジュアルタイプを参照してください。	

ソリューション内のすべての AWS リソースをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	1. ソリューションによってデプロイされた AWS リソースを削除するには、`cdk destroy` コマンドを実行します。 2. 2 つの S3 バケットからすべてのオブジェクトを削除してから、バケットを削除します。 詳細については、バケットを削除するを参照してください。	AWS DevOps、DevOps エンジニア

AWS リソース評価ツールの自動化のトップに追加機能を設定する

タスク	説明	必要なスキル
手動で作成したリソースを監視してクリーンアップします。	(オプション) アプリケーションに IaC ツールを使用してリソースを作成するというコンプライアンス要件がある場合は、AWS リソース評価ツールの自動化を使用して手動でプ	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	ロビジョニングされたリソースを取得することで、コンプライアンスを達成できます。また、このツールを使用して、リソースを IaC ツールにインポートまたはリソースを再作成できます。手動でプロビジョニングされたリソースを監視するには、以下の高レベルのタスクを実行します。 1. AWS リソース評価ツールの自動化をデプロイします。 2. Lambda 関数をセットアップして、Athena テーブルに毎日クエリを実行し、手動でプロビジョニングされたリソースの関連データを検索し、それをカンマ区切り値 (CSV) ファイルにエクスポートします。 3. Lambda 関数を実行後、必要なデータを含む通知をそれぞれのステークホルダーに送信できます。 4. 長期間保存する場合は、.csv ファイルを S3 バケットに保存できます。 5. .csv ファイルの情報に基づき、手動で作成したリソースを削除、または既存の IaC ソリューションにインポートします。	

トラブルシューティング

問題	ソリューション
AWS CDK はエラーを返します。	AWS CDK の問題に関するヘルプは、一般的な AWS CDK 問題をトラブルシューティングするを参照してください。

関連リソース

- Python で Lambda 関数を構築する
- AWS CDK の使用を開始する
- Python の AWS CDK を使用する
- CloudTrail 口グ証跡を作成する
- Amazon QuickSight を開始する

追加情報

複数アカウント

複数アカウントの AWS CLI 認証情報を設定するには、AWS プロファイルを使用します。詳細については、AWS CLI をセットアップする」の複数のプロファイルの設定するセクションを参照してください。

AWS CDK コマンド

AWS CDK を使用する際は、以下の便利なコマンドに注意してください。

- アプリ内のすべてのスタックを一覧表示

```
cdk ls
```

- 合成された AWS CloudFormation テンプレートを発行

```
cdk synth
```

- スタックをデフォルトの AWS アカウントとリージョンにデプロイ

```
cdk deploy
```

- デプロイされたスタックを現在の状態と比較

```
cdk diff
```

- AWS CDK ドキュメントを開く

```
cdk docs
```

オープンソースツールを使用して SAP システムを自動的にインストール

作成者：Guilherme Sesterheim (AWS)

概要

このパターンでは、オープンソースツールを使用して以下のリソースを作成することで SAP システムのインストールを自動化する方法を示しています。

- SAP S/4HANA 1909 データベース
- SAP ABAP セントラルサービス (ASCS) インスタンス
- SAP プライマリアプリケーションサーバー (PAS) インスタンス

HashiCorp Terraform が SAP システムのインフラストラクチャを作成し、Ansible がオペレーティングシステム (OS) を設定し、SAP アプリケーションをインストールします。Jenkins がインストールを実行します。

この設定により、SAP システムのインストールが反復可能なプロセスになり、デプロイの効率と品質が向上します。

> **Note**
>
> このパターンで提供されているサンプルコードは、高可用性 (HA) システムと非 HA システムの両方で機能します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- すべての SAP メディアファイルを含めるAmazon Simple Storage Service (Amazon S3) バケット
- アクセスキーとシークレットキー を持ち、以下の権限を持つ AWS アイデンティテとアクセス管理 (IAM) プリンシパルです。
  - 読み取り専用権限：Amazon Route 53、AWS Key Management Service (AWS KMS)
  - 読み取り権限と書き込み権限：Amazon S3、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic File System (Amazon EFS)、IAM、Amazon CloudWatch、Amazon DynamoDB

- ルート 53 プライベートホストゾーン
- Amazon マーケットプレイスの HA およびアップデートサービス 8.2 付けの Red Hat Enterprise Linux for SAP のAmazon マシンイメージ (AMI) についての説明
- AWS KMS カスタマー管理キー
- Secure Shell (SSH) キーペア
- Jenkins をインストールしたホスト名からのポート 22 で SSH 接続を許可する Amazon EC2 セキュリティグループ (多分、ホスト名がlocalhost)
- インストール、設定されたHashiCorpによる バグラント
- インストール、設定された Oracle による VirtualBox
- Git、テラフォーム、アンシブル、ジェンキンスに精通

制約事項

- この特定のシナリオで完全にテストされているのは SAP S/4HANA 1909 だけです。別のバージョンの SAP HANA を使用する場合、このパターンの Ansible コード例を変更する必要があります。
- このパターンのサンプル手順は Mac OS と Linux オペレーティングシステムで機能します。一部のコマンドは UNIX ベースの端末でしか実行できません。ただし、別のコマンドと Windows OS を使用しても同様の結果が取得されます。

製品バージョン

- SAP S/4HANA 1909
- Red Hat Enterprise Linux (RHEL) バージョン 6 以上

アーキテクチャ

次の図表では、オープンソースツールを使用して AWS アカウントで SAP システムのインストールを自動化するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. Jenkins は Terraform と Ansible のコードを実行することで SAP システムのインストールをオーケストレーションします。
2. Terraform コードは SAP システムのインフラストラクチャーを構築します。

3. Ansible コードが OS を設定し、SAP アプリケーションをインストールします。
4. 定義されたすべての前提条件を含む SAP S/4HANA 1909 データベース、ASCS インスタンス、および PAS インスタンスが Amazon EC2 インスタンスにインストールされます。

> **Note**
>
> このパターンのセットアップ例では、AWS アカウントに Amazon S3 バケットを自動的に作成して Terraform 状態ファイルを保存します。

テクノロジースタック

- Terraform
- Ansible
- Jenkins
- SAP S/4HANA 1909 データベース
- SAP ASCS インスタンス
- SAP PAS インスタンス
- Amazon EC2

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。仮想サーバーを必要な数だけ起動して、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、ユーザーのデータを保護するために使用される、暗号化キーの作成と制御を容易にするマネージドサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するのに役立つコマンドラインインターフェイスアプリケーションです。
- Ansible は、アプリケーション、構成、IT インフラストラクチャの自動化に役立つオープンソースのコード設定 (CaC) ツールです。
- Jenkins は、デベロッパーがソフトウェアを確実に構築、テスト、デプロイできるようにするオープンソースのオートメーションサーバーです。

コード

このパターンのコードでは、GitHub 内の「aws-install-sap-with-jenkins-ansible」リポジトリで利用できます。

エピック

AWS の前提条件の設定

タスク	説明	必要なスキル
Amazon S3 バケットにファイルを追加します。	SAP メディアファイルをすべて含め、Amazon S3 バケットを作成します。 **Important** Launch Wizard ドキュメントの S/4HANA の AWS Launch Wizard のフォルダ階層に従っていることを確認してください。	クラウド管理者
VirtualBox をインストールします。	Oracle で VirtualBox をインストールして設定します。	DevOps エンジニア

タスク	説明	必要なスキル
Vagrant をインストールします。	HashiCorp による Vagrantのインストールと設定を行います。	DevOps エンジニア

タスク	説明	必要なスキル
AWS アカウントの設定	1. アクセスキーとシークレットキー 付けの IAM プリンシパルを持って、次の権限があることを検証します。 • 読み取り専用権限：Amazon Route 53、AWS Key Management Service (AWS KMS) • 読み取り権限と書き込み権限：Amazon S3、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic File System (Amazon EFS)、IAM、Amazon CloudWatch、Amazon DynamoDB 2. IAM プリンシパルアクセスキーとシークレットキーを保存し、後で参照します。 3. Route 53 プライベートホストゾーン をまだ持っていない場合、作成します。後で参照できるように、ゾーン名 (sapteam.net) など) を保存します。 4. Amazon マーケットプレイスの HA およびアップデートサービス 8.2 AMI付けの Red Hat Enterprise Linux for SAP を購読します。後で参照できるように、 AMI	AWS 全般

タスク	説明	必要なスキル
	ID（たとえば ami-00000000）を保存します。 5. カスタマーマネージドAWS KMS キー.を作成。後で参照できるように、KMS キーの Amazon リソースネーム (ARN) を保存します。 **Note** AWS KMS カスタマーマネージドキー ARN の例を次に示します。arn:aws:kms:us-east-1:123412341234:key/uuid 6. SSH キーペア を作成します。後で参照できるように、キーペアの名前と.pem ファイルを保存します。 7. Jenkins をインストールするhostnameからポート 22 で SSH 接続を許可する Amazon EC2 セキュリティグループ を作成します。後で参照できるように、セキュリティグループ ID を保存します。	

タスク	説明	必要なスキル
	Note ホスト名は、ほとんどの場合 localhost です。	

SAP インストールを構築して実行

タスク	説明	必要なスキル
GitHub から リポジトリのクローンを作成します。	GitHub の aws-install-sap-with-jenkins-ansible リポジトリをクローンします。	DevOps エンジニア
Jenkins サービスを開始します。	Linux ターミナルを開く 次に、クローンされたコードリポジトリフォルダーを含むローカルフォルダーにナビゲーションし、以下のコマンドを実行します。 `sudo vagrant up` **Note** Jenkins の起動には約 20 分かかります。コマンドが成功する場合、サービスがアップし、実行されていますというメッセージが返されます。	DevOps エンジニア

タスク	説明	必要なスキル
ウェブブラウザで Jenkins を開き、ログインします。	1. ウェブブラウザで http://localhost:5555 を入力します。Jenkins が開きます。 2. ユーザー名は admin で、パスワードが my_secret_pass_from_vault を使用して Jenkins にログインします。	DevOps エンジニア

タスク	説明	必要なスキル
SAP システムのインストールパラメータを設定します。	1. ジェンキンスでジェンキンスの管理を選択します。次に、認証情報の管理を選択します。設定できる認証情報変数のリストが表示されます。 2. 次の認証情報変数をすべて設定します。 • AWS_ACCOUNT_CREDENTIALS には、IAM プリンシパルのアクセスキー ID とシークレットアクセスキー ID を入力します。 • HA とアップデートサービス 8.2 AMI の AMI_ID の SAP の Red Hat Enterprise Linux を入力します。 • KMS_KEY_ARN には、AWS KMS カスタマー管理キーの ARN を入力します。 • SSH_KEYPAIR_NAME の場合、.pem ファイルタイプを入力せずに、 SSH キーペアの名前を入力します。 • SSH_KEYPAIR_FILE の場合、キーペアの.pem ファイルのフルネーム (たとえば mykeypair.pem) を入力します。キーペアの.pem ファイルも必ず Jenkins にアップロードします。	AWS システム管理者、DevOps エンジニア

タスク	説明	必要なスキル
	• S3_ROOT_FOLDER_INSTALL_FILES, の場合、SAP メディアファイルを含む名称とAmazon S3 バケット(該当する場合)(たとえば、s3://my-media-bucket/S4H1909) を入力します。 • PRIVATE_DNS_ZONE_NAME, の場合、Route 53 のプライベートホストゾーンの名前 (たとえば myprivatecompanyurl.net) を入力します。 • VPC_ID の場合、SAP リソースを作成している Amazon VPC の VPC ID (たとえば、vpc-12345) を入力します。 • SUBNET_IDS の場合、テスト環境 (未来の HA 機能用) で実行されている場合、 2 つのパブリックサブネット ID を入力します。実稼働環境で作業する場合、拠点ホストで 2 つのプライベートサブネットを使用するのがベストプラクティスです。 • SECURITY_GROUP_IDの場合、Jenkins をインストールしたホスト名からポート 22 で SSH 接続を許可する Amazon EC2 セキュリティグループの ID を入力します。	

タスク	説明	必要なスキル
	Note 必要に応じて、ユースケースに基づいて、その他の不要なパラメータを設定できます。たとえば、SAP システムのインスタンスの SAP システム ID (SID)、デフォルトパスワード、名前、タグを変更できます。 すべての必須変数には、名前の先頭に (必須) が付いています。	

タスク	説明	必要なスキル
SAP システムインストールを実行します。	1. ジェンキンスでは、ジェンキンスホームを選択します。次に、SAP HANA +ASCS+PAS 3 インスタンス] を選択します。 2. スピンアップしてインストールを選択します。次に、メインを選択します。 3. 今すぐビルドを選択します。 パイプラインステップの詳細については、AWS ブログのオープンソースツールによる SAP インストールの自動化のパイプラインステップの理解 セクションを参照してください。 **Note** エラーが発生した場合は、表示される赤いエラーボックスにカーソルを移動し、ログを選択します。エラーが発生したパイプラインステップのログが表示されます。ほとんどのエラーは、誤ったパラメータ設定が原因で発生します。	DevOps エンジニア、AWS システム管理者

関連リソース

- SAP 向け DevOps — SAP インストール：2 か月から 2 時間へ (DevOps エンタープライズ提出ビデオライブラリ)

AWS CDK を使用して AWS Service Catalog ポートフォリオと製品のデプロイを自動化する

作成者: Sandeep Gawande (AWS)、RAJNEESH TYAGI (AWS)、および Viyoma Sachdeva (AWS)

概要

AWS Service Catalog は、組織の AWS 環境での使用が承認された IT サービスまたは製品のカタログを一元管理するのに役立ちます。ポートフォリオとは、製品の集合で、設定情報も含まれます。AWS Service Catalogでは、組織のユーザータイプごとにカスタマイズしたポートフォリオを作成し、適切なポートフォリオへのアクセス権を選択的に付与できます。そうすれば、ユーザーはポートフォリオ内から必要な製品をすばやくデプロイできます。

マルチリージョンアーキテクチャやマルチアカウントアーキテクチャなどの複雑なネットワークインフラストラクチャを使用している場合は、Service Catalog ポートフォリオを単一の中央アカウントで作成および管理することをお勧めします。このパターンでは、AWS Cloud Development Kit (AWS CDK) を使用して、中央アカウントでの Service Catalog ポートフォリオの作成を自動化し、エンドユーザーにそのポートフォリオへのアクセスを許可し、オプションで 1 つ以上のターゲット AWS アカウントに製品をプロビジョニングする方法を説明します。このすぐに使用できるソリューションにより、ソースアカウントに Service Catalog ポートフォリオが作成されます。また、オプションで AWS CloudFormation スタックを使用してターゲットアカウントに製品をプロビジョニングし、製品の TagOptions を設定するのに役立ちます。

- AWS CloudFormation StackSets — StackSets を使用して、複数の AWS リージョンとアカウントにわたってService Catalog 製品を起動できます。このソリューションでは、このソリューションをデプロイするときに製品を自動的にプロビジョニングできます。詳細については、AWS CloudFormation StackSets (Service Catalog ドキュメント) と StackSets の概念 (CloudFormation ドキュメント) を参照してください。
- TagOption ライブラリ — TagOption ライブラリを使用して、プロビジョニングされた製品のタグを管理できます。TagOption は AWS Service Catalog で管理されるキーと値のペアです。これは AWS のタグではありませんが、TagOption に基づいて AWS のタグを作成するためのテンプレートとして機能します。詳細については、TagOption ライブラリー との共有(Service Catalog ドキュメント) を参照してください。

前提条件と制限

前提条件

- Service Catalog ポートフォリオを管理するためのソースアカウントとして使用するアクティブな AWS アカウント。
- このソリューションを使用して 1 つ以上のターゲットアカウントに製品をプロビジョニングする場合、ターゲットアカウントはすでに存在し、アクティブである必要があります。
- AWS Service Catalog、AWS CloudFormationおよび AWS IAM にアクセスするための AWS Identity and Access Management (IAM) 権限。

製品バージョン

- AWS CDK バージョン 2.27.0

アーキテクチャ

ターゲットテクノロジースタック

- 一元管理された AWS アカウントのService Catalog ポートフォリオ
- ターゲットアカウントにデプロイされたService Catalog 製品

ターゲット アーキテクチャ

1. ポートフォリオ (または ソース) アカウントで、config.json ファイルを AWS アカウント、AWS リージョン、IAM ロール、ポートフォリオ、ユースケースの製品情報で更新します。
2. AWS CDK アプリケーションをデプロイします。
3. AWS CDK アプリケーションはデプロイメント IAM ロールを引き受け、config.json ファイルに定義されているService Catalog ポートフォリオと製品を作成します。

   ターゲットアカウントに製品をデプロイするように StackSets を設定した場合、プロセスは続行されます。製品をプロビジョニングするように StackSets を設定していない場合、プロセスは完了です。
4. AWS CDK アプリケーションはStackSet 管理者ロールを引き受け、config.json ファイルで定義した AWS CloudFormationスタックセットをデプロイします。
5. ターゲットアカウントでは、StackSets が StackSet の実行ロールを引き受け、製品をプロビジョニングします。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK Toolkit は、 AWS CDKアプリケーションの操作に役立つコマンドラインクラウド開発キットです。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Service Catalog では、AWS で承認された IT サービスのカタログを一元管理できます。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-cdk-servicecatalog-automation」リポジトリで利用できます。コードリポジトリには以下のファイルとフォルダが含まれています。

- cdk-sevicecatalog-app — このフォルダには、このソリューションの AWS CDK アプリケーションが含まれています。
- config — このフォルダには、Service Catalog ポートフォリオに製品をデプロイするための config.json ファイルと CloudFormation テンプレートが含まれています。
- config/config.json — このファイルにはすべての設定情報が含まれています。このファイルを更新して、ユースケースに合わせてこのソリューションをカスタマイズします。
- config/templates — このフォルダには、サービスセンター製品のCloudFormation テンプレートが含まれています。
- setup.sh — このスクリプトはソリューションをデプロイします。
- uninstall.sh — このスクリプトは、スタックと、このソリューションのデプロイ時に作成されたすべての AWS リソースを削除します。

これらのファイルを使用するには、エピックセクションの指示に従ってください。

ベストプラクティス

- このソリューションのデプロイに使用する IAM ロールは、最小権限の原則 (IAM ドキュメント) に従う必要があります。
- AWS CDK でクラウドアプリケーションを開発するためのベストプラクティス (AWS ブログ記事) に従ってください。
- AWS CloudFormation のベストプラクティス (クラウドフォーメーションドキュメント) に従ってください。

エピック

環境をセットアップします。

タスク	説明	必要なスキル

AWS CDK Toolkitをインストールします。	AWS CDK Toolkit がインストールされていることを確認します。次のコマンドを入力して、インストールされているかどうかを確認し、バージョンを確認します。
	`cdk --version`
	がインストールされていない場合は、次のコマンドを実行してインストールします。
	`npm install -g aws-cdk@2.27.0`
	AWS CDK Toolkit のバージョンが 2.27.0 より前の場合は、次のコマンドを入力してバージョン 2.27.0 に更新します。
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`npm install -g aws-cdk@2.27.0 --force`	

リポジトリをクローン作成します。	次のコマンドを入力します。追加情報 セクションのリポジトリのクローン では、リポジトリの URL を含むコマンド全体をコピーできます。これにより、GitHub から aws-cdk-サービスカタログ-自動化 リポジトリがクローンされます。
	`git clone <repository-URL>.git`
	これにより、ターゲットディレクトリに `cd aws-cdk-servicecatalog-automation` フォルダーが作成されます。次のコマンドを入力して、このフォルダーに移動します。
	`cd aws-cdk-servicecatalog-automation`
		AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

AWS 認証情報の設定	以下のコマンドを入力します。これらは、スタックをデプロイする AWS アカウントとリージョンを定義する以下の変数をエクスポートします。
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	`export CDK_DEFAULT_REGION=<AWS Region>`
	AWS CDK の AWS 認証情報は、環境変数を通じて提供されます。
		AWS DevOps、DevOps エンジニア

エンドユーザーIAM ロールのアクセス許可を設定する	IAM ロールを使用してポートフォリオとその中の製品へのアクセスを許可する場合、ロールには servicecatalog.amazonaws.comサービスプリンシパルが引き受ける権限が必要です。これらのアクセス権限を付与する方法については、Service Catalog による信頼できるアクセスの有効化 (AWS Organizations ドキュメント) を参照してください。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
StackSets に必要な IAM ロールを設定します。	StackSets を使用してターゲットアカウントに製品を自動的にプロビジョニングする場合は、スタックセットを管理および実行する IAM ロールを設定する必要があります。 1. ソースアカウントで、`AWSCloudFormationStackSetAdministrationRole` が既に存在しているかどうかを確認します。ソースアカウントで、`AWSCloudFormationStackSetExecutionRole` が既に存在しているかどうかを確認します。これらのロールが既にある場合、次のエピックまでスキップできます。 2. 自己管理権限の付与 (IAM ドキュメント)の手順に従って、ポートフォリオアカウントにスタックセット管理ロールを作成し、各ターゲットアカウントに実行ロールを作成します。	AWS DevOps、DevOps エンジニア

ソリューションのカスタマイズとデプロイ

タスク	説明	必要なスキル
CloudFormation テンプレートの準備	この `config/templates` フォルダーで、ポートフォ	アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	リオに含めたい製品の CloudFormation テンプレートを作成します。詳細については、Working with AWS CloudFormation テンプレート (CloudFormation ドキュメント) を参照してください。	

タスク	説明	必要なスキル
設定ファイルをカスタマイズします。	`config` フォルダーで config.json ファイルを開き、ユースケースに適したパラメーターを定義します。特に明記されていない限り、以下のパラメータは必須です。 `portfolios` セクションで、以下のパラメータを定義して 1 つ以上のService Catalog ポートフォリオを作成します。 • ポートフォリオの名前。 • `providerName` — ポートフォリオを管理する個人、チーム、または組織の名前。 • `description` — ポートフォリオに関する簡単な説明。 • `roles` — (オプション) このポートフォリオにアクセスできるはずの IAM ロールの名前。このロールを持つユーザーは、このポートフォリオの製品にアクセスできます。 • `users` — (オプション) このポートフォリオとその製品にアクセスできるはずの IAM ユーザーの名前。 • `groups` — (オプション) このポートフォリオとその製	アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	品にアクセスできるはずの IAM ユーザーの名前。 **⚠ Warning** IAM ユーザーは長期認証情報を保有するため、セキュリティ上のリスクが生じます。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。 **⚠ Important** `roles`、`users`、`groups`はすべてオプションパラメータですが、これらのパラメータのいずれかを定義しない場合、Service Catalog コンソールでポートフォリオ製品を表示することはできません。これらのパラメータのうち少なくとも 1 つを定義します。詳細については、Service Catalog	

タスク	説明	必要なスキル

	エンドユーザーへの権限の付与 (Service Catalog ドキュメント) を参照してください。
	
	• (オプション) tagOption セクションで、製品の TagOptions を定義します。
	  • key — TagOption キーの名前
	  • value — TagOption に使用できる文字列値
	
	詳細については、TagOption ライブラリー との共有(Service Catalog ドキュメント) を参照してください。
	• productsセクションで、次のパラメーターの値を指定します。
	  • portfolioName — 製品を追加するポートフォリオの名前。 – 指定できるフォーマットは 1 つだけです。
	  • productName – 製品の名前。
	  • owner – 製品の所有者。
	  • productVersionName — 文字列値内の製品バージョンの名前 (v1 など)。	

タスク	説明	必要なスキル

	• `templatePath` — 製品の CloudFormation テンプレートのファイルパス。
	• `deployWithStackSets` — (オプション) StackSets を使用してポートフォリオ内の製品を自動的にプロビジョニングする 1 つ以上のアカウントとリージョンを指定します。このデプロイオプションを使用する場合、このセクションの以下のパラメータはすべて必須です。
	  • `accounts` — ターゲットアカウント。
	  • `regions` – ターゲットリージョンです。
	  • `stackSetAdministrationRoleName` — StackSets 設定の管理に使用される IAM ロールの名前。この値を変更しないでください。ロールにはこの正確な名前が必要です。
	  • `stackSetExecutionRoleName` — スタックインスタンスをデプロイするターゲットアカ	

タスク	説明	必要なスキル
	ウントの IAM ロールの名前。この値を変更しないでください。ロールにはこの正確な名前が必要です。 完成した設定ファイルの例については、追加情報 セクションのサンプル設定ファイルを参照してください。	
ソリューションをデプロイします。	次のコマンドを入力します。これにより、AWS CDK アプリケーションがデプロイされ、config.json ファイルに指定されているService Catalog ポートフォリオと製品がプロビジョニングされます。 `sh +x setup.sh`	アプリ開発者、AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
デプロイメントを確認する	以下を実行して、デプロイが正常に完了したことを確認します。 1. 設定ファイルで定義した 1 つ以上のポートフォリオにアクセスできる認証情報を使用して AWS マネジメントコンソールにサインインします。 2. Service Catalog コンソール (https://console.aws.amazon.com/servicecatalog/) を開きます。 3. ナビゲーションペインで、プロビジョニング 内の製品を選択します。ポートフォリオに指定した製品のリストが表示されていることを確認します。 4. 製品の起動 (Service Catalog ドキュメント)の手順に従って、使用可能な製品のいずれかを起動します。使用可能な製品バージョンとタグが、設定ファイルに指定した値と一致していることを確認します。 5. StackSets を使用して 1 つ以上のターゲットアカウントに製品を自動的にプロビジョニングすることを選択した場合は、次の操作を行います。	AWS 全般

タスク	説明	必要なスキル
	a. いずれかのターゲットアカウントでプロビジョニングされた製品を閲覧する権限を付与する認証情報を使用してサインインします。 b. Service Catalog コンソールのナビゲーションペインのプロビジョニングで、プロビジョニング済み製品を選択します。 c. 予想される製品がリストに表示されていることを確認します。	

タスク	説明	必要なスキル
(オプション) ポートフォリオと製品を更新します。	このソリューションを使用してポートフォリオや製品を更新したり、新しい製品をプロビジョニングしたりする場合： 1. config.json ファイルに必要な変更を加えます。 2. 必要に応じて、`config/template` フォルダー内の CloudFormation テンプレートを追加または変更します。 3. ソリューションをデプロイする たとえば、ポートフォリオを追加したり、リソースをさらにプロビジョニングできます。AWS CDK アプリケーションは変更のみを実装します。以前にデプロイしたポートフォリオや製品に変更がなければ、再デプロイしても影響はありません。	アプリ開発者、AWS DevOps、DevOps エンジニア

ソリューションをクリーンアップする

タスク	説明	必要なスキル
(オプション) このソリューションによってデプロイされ	プロビジョニング済み製品を削除する場合は、プロビジョニング済み製品の削除	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
た AWS リソースを削除します。	(Service Catalog ドキュメント)の手順に従ってください。 このソリューションで作成されたリソースをすべて削除する場合は、次のコマンドを入力します。 `sh uninstall.sh`	

関連リソース

- AWS Service Catalog コンストラクトライブラリ (AWS API リファレンス)
- StackSets の概念 (CloudFormation ドキュメント)
- AWS Service Catalog (AWS マーケティング)
- AWS CDK でのService Catalog の使用 (AWS ワークショップ)

追加情報

リポジトリをクローンします

リポジトリのクローンを作成するには、次の コマンドを入力します。

```
git clone https://github.com/aws-samples/aws-cdk-servicecatalog-automation.git
```

サンプル設定ファイル

以下は、サンプル値を含む config.json ファイルのサンプルです。

```
{
    "portfolios": [
        {
            "displayName": "EC2 Product Portfolio",
            "providerName": "User1",
            "description": "Test1",
            "roles": [
```

```
                "<Names of IAM roles that can access the products>"
            ],
            "users": [
                "<Names of IAM users who can access the products>"
            ],
            "groups": [
                "<Names of IAM user groups that can access the products>"
            ]
        },
        {
            "displayName": "Autoscaling Product Portfolio",
            "providerName": "User2",
            "description": "Test2",
            "roles": [
                "<Name of IAM role>"
            ]
        }
    ],
    "tagOption": [
        {
            "key": "Group",
            "value": [
                "finance",
                "engineering",
                "marketing",
                "research"
            ]
        },
        {
            "key": "CostCenter",
            "value": [
                "01",
                "02",
                "03",
                "04"
            ]
        },
        {
            "key": "Environment",
            "value": [
                "dev",
                "prod",
                "stage"
            ]
```

```
        }
    ],
    "products": [
        {
            "portfolioName": "EC2 Product Profile",
            "productName": "Ec2",
            "owner": "owner1",
            "productVersionName": "v1",
            "templatePath": "../../config/templates/template1.json"
        },
        {
            "portfolioName": "Autoscaling Product Profile",
            "productName": "autoscaling",
            "owner": "owner1",
            "productVersionName": "v1",
            "templatePath": "../../config/templates/template2.json",
            "deployWithStackSets": {
                "accounts": [
                    "012345678901",
                ],
                "regions": [
                    "us-west-2"
                ],
                "stackSetAdministrationRoleName":
 "AWSCloudFormationStackSetAdministrationRole",
                "stackSetExecutionRoleName": "AWSCloudFormationStackSetExecutionRole"
            }
        }
    ]
}
```

CodeBuild と CloudWatch イベントを使用して、CodeCommit から Amazon S3 へのイベント駆動型バックアップを自動化する

作成者: Kirankumar Chandrashekar (AWS)

概要

Amazon Web Services (AWS) クラウドでは、AWS CodeCommit を使用して安全な Git ベースのリポジトリをホストできます。CodeCommit は、完全マネージド型のソース管理サービスです。ただし、CodeCommit リポジトリを誤って削除すると、その内容も削除され、復元できません。

このパターンは、リポジトリを変更した後に CodeCommit リポジトリを Amazon Simple Storage Service (Amazon S3) バケットに自動的にバックアップする方法を説明しています。CodeCommit リポジトリを後で削除すると、このバックアップ戦略では特定の時点で回復オプションが提供されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 要件に応じて、ユーザーアクセスが設定された既存の CodeCommit リポジトリ。詳細については、CodeCommit ドキュメントの AWS CodeCommit のセットアップ を参照してください。
- CodeCommit バックアップをアップロードする S3 バケット。

制限

- このパターンは、すべての CodeCommit リポジトリを自動的にバックアップします。各 CodeCommit リポジトリをバックアップする場合は、Amazon CloudWatch Events ルールを変更する必要があります。

アーキテクチャ

次の図は、このパターンのワークフローを図解したものです。

ワークフローは、以下の手順で構成されます。

1. コードは CodeCommit リポジトリにプッシュされます。
2. CodeCommit リポジトリは、リポジトリの変更 (`git push` コマンドなど) を CloudWatch イベントに通知します。
3. CloudWatch イベントは AWS CodeBuild を呼び出し、それに CodeCommit リポジトリ情報を送信します。
4. CodeBuild は CodeCommit リポジトリ全体をクローン作成し、.zip ファイルにパッケージ化します。
5. CodeBuild は S3 バケットに .zip ファイルをアップロードします。

テクノロジースタック

- CloudWatch Events
- CodeBuild
- CodeCommit
- Amazon S3

ツール

- Amazon CloudWatch Events – CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- AWS CodeBuild - AWS CodeBuild は、ソースコードをコンパイルし、テストを実行し、デプロイ準備ができたソフトウェアパッケージを作成するフルマネージド型の継続的統合サービスです。

- AWS CodeCommit — CodeCommit は、安全な Git ベースのリポジトリをホストする完全マネージド型のソース管理サービスです。
- AWS Identity and Access Management (IAM) は、IMS は AWS リソースへのアクセスのセキュアな制御に役立つ Web サービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。

エピック

CodeBuild プロジェクトを作成する

タスク	説明	必要なスキル
CodeBuild サービスロールを作成します。	AWS マネジメントコンソールにサインインし、IAM コンソールを開きます。[Roles (ロール)]、[Create role (ロールの作成)] の順に選択します。CodeBuild のサービスロールを作成して、CodeCommit リポジトリをクローン作成し、S3 バケットにファイルをアップロードし、Amazon CloudWatch にログを送信します。詳細については、CodeBuild ドキュメントの CodeBuild サービスロールの作成を参照してください。	クラウド管理者
CodeBuild プロジェクトを作成します。	CodeBuild コンソールで、[Create CodeBuild project (CodeBuild プロジェクトを作成)] を選択します。追加情報セクションの `buildspec.yml` テンプレートを使用して CodeBuild プロジェクトを作成します。このストーリーのヘルプについては、CodeBuild ドキュメントのビルドプロジェクトの作成を参照してください。	クラウド管理者

CloudWatch Events ルールを作成および設定する

タスク	説明	必要なスキル
CloudWatch Events の IAM ロール を作成します。	IAM コンソールで [Roles (ロール)] を選択し、CloudWatch Events 用に IAM ロールを作成します。詳細については、IAM ドキュメントの CloudWatch イベント IAM ロールを参照してください。 **Important** CloudWatch Events の IAM ロールにアクセス`codebuild:StartBuild` 許可を追加する必要があります。	クラウド管理者
CloudWatch Events ルールを作成します。	1. CloudWatch コンソールで [Events (イベント)] を選択してから、[Rules (ルール)] を選択します。[Create rule (ルールを作成)] を選択し、追加情報セクションの CloudWatch イベントルールを使用します。これで、CodeCommit リポジトリ内のイベントの変更 (`git push` または `git commit` コマンドなど) を監視するルールが作成されます。詳細については、AWS CodePipeline ドキュメントの CodeCommit ソースの	クラウド管理者

タスク	説明	必要なスキル
	CloudWatch Events ルールを作成するを参照してください。 2. [Targets (ターゲット)] 、[Topic (トピック)] を選択してから、[Configure input (入力の設定)] を選択します。[Input transformer (入力トランスフォーマー)] を選択し、追加情報セクションの入力パスと入力テンプレートを使用します。これで、CodeCommit リポジトリの詳細が構文解析され、環境変数として CodeBuild プロジェクトに送信されます。詳細については、CloudWatch ドキュメントの入力トランスフォーマーチュートリアルを参照してください。 3. [Configure details (詳細の設定)] を選択し、ルールの名前と説明を入力します。ルールの作成を選択します。 **Important** この CloudWatch Events ルールは、すべての CodeCommit リポジトリの変更を記述します。各	

タスク	説明	必要なスキル
	CodeCommit リポジトリをバックアップ、または異なるリポジトリのバックアップに別の S3 バケットを使用する場合は、CloudWatch Events ルールを変更する必要があります。	

関連リソース

CodeBuild プロジェクトの作成

- CodeBuild サービスロールを作成する
- CodeBuild プロジェクトを作成する
- Git クライアントのコマンドに必要な権限

CloudWatch Events ルールの作成と設定

- CodeCommit ソースの CloudWatch Events ルールを作成する
- 入力トランスフォーマーを使用してイベントターゲットに渡されるものをカスタマイズする
- イベントで開始する CloudWatch Events ルールを作成する
- CloudWatch Events の IAM ロールを作成する

追加情報

CodeBuild buildspec.yml テンプレート

```
version: 0.2
phases:
  install:
    commands:
      - pip install git-remote-codecommit
```

```
  build:
    commands:
       - env
       - git clone -b $REFERENCE_NAME codecommit::$REPO_REGION://$REPOSITORY_NAME
       - dt=$(date '+%d-%m-%Y-%H:%M:%S');
       - echo "$dt"
       - zip -yr $dt-$REPOSITORY_NAME-backup.zip ./
       - aws s3 cp $dt-$REPOSITORY_NAME-backup.zip s3:// #substitute a valid S3 Bucket
 Name here
```

CloudWatch Events ルール

```
{
  "source": [
    "aws.codecommit"
  ],
  "detail-type": [
    "CodeCommit Repository State Change"
  ],
  "detail": {
    "event": [
      "referenceCreated",
      "referenceUpdated"
    ]
  }
}
```

CloudWatch Events ルールターゲットのサンプル入力トランスフォーマー

入力パス:

```
{"referenceType":"$.detail.referenceType","region":"$.region","repositoryName":"$.detail.reposi
```

入力テンプレート (必要に応じて値を入力してください):

```
{
    "environmentVariablesOverride": [
        {
            "name": "REFERENCE_NAME",
            "value": ""
        },
        {
```

```
            "name": "REFERENCE_TYPE",
            "value": ""
        },
        {
            "name": "REPOSITORY_NAME",
            "value": ""
        },
        {
            "name": "REPO_REGION",
            "value": ""
        },
        {
            "name": "ACCOUNT_ID",
            "value": ""
        }
    ]
}
```

AWS CloudFormation スタックと関連リソースの削除を自動化する

作成者: SANDEEP SINGH (AWS) と James Jacob (AWS)

概要

AWS CloudFormation は、クラウドInfrastructure as Code (IaC) を管理するために広く使用されているサービスです。CloudFormation を使用する場合、関連するリソースをスタックと呼ばれる単一のユニットとして管理します。スタックを作成、更新、削除することで、リソースのコレクションを作成、更新、削除します。

CloudFormation スタック内のリソースが不要になる場合があります。リソースとその設定によっては、スタックとその関連リソースの削除が複雑になる場合があります。実際の本番稼働システムでは、CloudFormation が上書きできない競合する条件や制限により、削除が失敗したり、長時間かかることがあります。すべてのリソースが効率的かつ一貫した方法で適切に削除されるようにするには、慎重な計画と実行が必要になる場合があります。このパターンでは、以下の複雑さを伴う CloudFormation スタックの削除を管理するのに役立つフレームワークを設定する方法について説明します。

- 削除保護を持つリソース – 一部のリソースでは、削除保護が有効になっている場合があります。一般的な例としては、Amazon DynamoDB テーブルと Amazon Simple Storage Service (Amazon S3) バケットがあります。削除保護は、CloudFormation による削除など、自動削除を防止します。これらのリソースを削除する場合は、手動で、またはプログラムで削除保護を上書きまたは一時的に無効にする必要があります。先に進む前に、これらのリソースを削除する意味を慎重に検討する必要があります。
- 保持ポリシーを持つリソース – AWS Key Management Service （ AWS KMS) キーや Amazon S3 バケットなどの特定のリソースには、削除がリクエストされた後に保持する期間を指定する保持ポリシーがある場合があります。組織のポリシーと規制要件への準拠を維持するために、クリーンアップ戦略でこれらのポリシーを考慮する必要があります。
- VPC にアタッチされている Lambda 関数の遅延削除 – 仮想プライベートクラウド (VPC) にアタッチされているAWS Lambda関数の削除には、プロセスに関連する相互接続された複数の依存関係に応じて、5～40 分かかる場合があります。スタックを削除する前に VPC から関数をデタッチすると、この遅延を 1 分未満に短縮できます。
- CloudFormation によって直接作成されていないリソース – 特定のアプリケーション設計では、リソースは、アプリケーション自体またはスタックを介してプロビジョニングされたリソースによって、元の CloudFormation スタックの外部に作成される場合があります。以下に 2 つの例を示します。

- CloudFormation は、ユーザーデータスクリプトを実行する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをプロビジョニングする場合があります。その後、このスクリプトはアプリケーション関連のデータを保存するための AWS Systems Managerパラメータを作成する場合があります。このパラメータは CloudFormation では管理されません。
- CloudFormation は、ログを保存するための Amazon CloudWatch Logs グループを自動的に生成する Lambda 関数をプロビジョニングする場合があります。このロググループは CloudFormation では管理されません。

これらのリソースは CloudFormation によって直接管理されませんが、多くの場合、スタックが削除されたときにクリーンアップする必要があります。管理対象外のままにすると、孤立し、不要なリソース消費につながる可能性があります。

これらのガードレールは複雑さを引き起こす可能性がありますが、意図的で重要です。CloudFormation がすべての制約を上書きし、リソースを無差別に削除することを許可すると、多くのシナリオで有害で予期しない結果が生じる可能性があります。ただし、環境の管理を担当する DevOps またはクラウドエンジニアとして、特に開発、テスト、ステージング環境では、これらの制約を上書きする必要がある場合があります。

ターゲットを絞ったビジネス成果

このフレームワークを実装することで、次の利点を実現できます。

- コスト管理 – end-to-endのテスト環境やユーザー受け入れテスト環境など、一時的な環境を定期的に効率的にクリーンアップすることで、リソースが必要以上に長く実行されないようにします。これにより、コストを大幅に削減できます。
- セキュリティ — 古いリソースや未使用のリソースの自動クリーンアップは、攻撃対象領域を減らし、安全な AWS 環境を維持するのに役立ちます。
- 運用効率 – 定期クリーンアップと自動クリーンアップには、次の運用上の利点があります。
  - 古いロググループまたは空の Amazon S3 バケットを削除する自動スクリプトは、環境をクリーンで管理しやすいようにすることで、運用効率を向上させることができます。
  - スタックをすばやく削除して再作成すると、設計と実装の迅速な反復がサポートされるため、より堅牢で回復力のあるアーキテクチャになる可能性があります。
  - 環境を定期的に削除して再構築すると、潜在的な問題を特定して修正するのに役立ちます。これにより、インフラストラクチャが実際のシナリオに確実に耐えられるようになります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Python バージョン 3.6 以降、インストール済み
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み

制約事項

- 命名規則は、削除するリソースを識別するために使用されます。このパターンのサンプルコードでは、リソース名のプレフィックスを使用しますが、独自の命名規則を定義できます。この命名規則を使用しないリソースは識別されず、その後も削除されません。

アーキテクチャ

次の図は、このフレームワークがターゲット CloudFormation スタックとそれに関連付けられた追加のリソースを識別する方法を示しています。

この図表は、次のワークフローを示しています：

1. リソースの収集 – オートメーションフレームワークは命名規則を使用して、関連するすべての CloudFormation スタック、Amazon Elastic Container Registry (Amazon ECR) リポジトリ、DynamoDB テーブル、Amazon S3 バケットを返します。

> **Note**
>
> このステージの関数は、切り捨てられた API 結果セットを反復するプロセスを抽象化する Boto3 の機能であるページネーターを使用します。これにより、すべてのリソースが確実に処理されます。パフォーマンスをさらに最適化するには、サーバー側のフィルタリングを適用するか、JMESPath を使用してクライアント側のフィルタリングを実行することを検討してください。

2. 前処理 — 自動化フレームワークは、CloudFormation がリソースを削除できるようにするために上書きする必要があるサービス制約を識別して対処します。たとえば、DynamoDB テーブルの

`DeletionProtectionEnabled`設定を に変更します`False`。コマンドラインインターフェイスでは、リソースごとに、制約を上書きするかどうかを尋ねるプロンプトが表示されます。

3. スタックの削除 – 自動化フレームワークは CloudFormation スタックを削除します。コマンドラインインターフェイスで、スタックを削除するかどうかを確認するプロンプトが表示されます。
4. 後処理 – 自動化フレームワークは、スタックの一部として CloudFormation から直接プロビジョニングされなかった関連リソースを削除します。これらのリソースタイプの例には、Systems Manager パラメータと CloudWatch ロググループが含まれます。個別の関数は、これらのリソースを収集し、前処理してから削除します。コマンドラインインターフェイスでは、リソースごとに、リソースを削除するかどうかを確認するプロンプトが表示されます。

> **Note**
>
> このステージの関数は、切り捨てられた API 結果セットを反復するプロセスを抽象化する Boto3 の機能であるページネーターを使用します。これにより、すべてのリソースが確実に処理されます。パフォーマンスをさらに最適化するには、サーバー側のフィルタリングを適用するか、JMESPath を使用してクライアント側のフィルタリングを実行することを検討してください。

自動化とスケール

CloudFormation スタックにサンプルコードに含まれていない他のリソースが含まれている場合、またはスタックにこのパターンで対処されていない制約がある場合は、ユースケースに合わせて自動化フレームワークを適応させることができます。リソースの収集、前処理、スタックの削除、後処理と同じ方法に従います。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体にわたる管理に役立ちます AWS リージョン。
- CloudFormation コマンドラインインターフェイス (CFN-CLI) は、 AWS およびサードパーティーの拡張機能を開発およびテストし、CloudFormation で使用するために登録するのに役立つオープンソースツールです。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。

その他のツール

- Click は、コマンドラインインターフェイスの作成に役立つ Python ツールです。
- Poetry は、Python での依存関係の管理とパッケージングのためのツールです。
- Pyenv は、Python のバージョンを管理および切り替えるのに役立つツールです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub cloudformation-stack-cleanup リポジトリで入手できます。

ベストプラクティス

- 識別しやすいようにリソースにタグを付ける – タグ付け戦略を実装して、さまざまな環境や目的のために作成されたリソースを特定します。タグを使用すると、タグに基づいてリソースをフィルタリングできるため、クリーンアッププロセスを簡素化できます。
- リソースライフサイクルの設定 – リソースライフサイクルを定義して、一定期間後にリソースを自動的に削除します。この手法は、一時的な環境が永続的なコスト負担にならないようにするのに役立ちます。

エピック

ツールをインストールする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	1. 仮想環境にフォルダを作成します。プロジェクト名で名前を付けます。 2. ローカルマシンでターミナルを開き、このフォルダに移動します。 3. 次のコマンドを入力して、cloudformation-stack-cleanup リポジトリをプロ	DevOps エンジニア

タスク	説明	必要なスキル
	ジェクトディレクトリにクローンします。 `git clone https://github.com/aws-samples/cloudformation-stack-cleanup.git`	
Poetry をインストールします。	Poetry をターゲット仮想環境にインストールするには、https://python-poetry.org/docs/「Poetry ドキュメント」の手順 (Poetry ドキュメント) に従います。	DevOps エンジニア

依存関係をインストールします。	1. プロジェクトディレクトリに移動するには、次のコマンドを入力します。
	
	`cd cloudformation-stack-cleanup`
	
	2. 次のコマンドを入力します。
	
	`poetry install`
	
	これにより、Boto3、クリック、CloudFormation CLI のソースコードなど、必要な依存関係がすべてインストールされます。
		DevOps エンジニア

(オプション) Pyenv をインストールします。	指示 (GitHub) に従って Pyenv をインストールします。	DevOps エンジニア

(オプション) フレームワークをカスタマイズする

タスク	説明	必要なスキル
ターゲットリソースを収集、前処理、削除する関数を作成します。	1. クローンされたリポジトリで、次のコマンドを入力して cli ディレクトリに移動します。 `cd cfncli/cli` 2. cleanup_enviornment.py ファイルを開きます。 3. 変更するリソースのタイプを収集する新しい Python 関数を作成します。例については、このファイルの gather_ddb_tables 関数を参照してください。 4. ターゲットリソースのサービス制約を上書きする新しい Python 関数を作成します。例については、このファイルの remove_ddb_deletion_protection 関数を参照してください。 5. アンマネージドターゲットリソースを収集する新しい Python 関数を作成します。例については、このファイルの gather_log_groups 関数を参照してください。 6. アンマネージドターゲットリソースを削除する新	DevOps エンジニア、Python

タスク	説明	必要なスキル
	しい Python 関数を作成します。例については、このファイルの `delete_log_group` 関数を参照してください。 7. cleanup_enviornment.py ファイルを保存して閉じます。	

サンプルリソースを作成する

タスク	説明	必要なスキル
CloudFormation スタックを作成します。	1. プロジェクトディレクトリに移動します。 2. 次のコマンドを入力して、DynamoDB テーブルとセキュリティグループをプロビジョニングする CloudFormation スタックを作成します。の値を更新します`<VPCID>`。 `aws cloudformation create-stack \ --stack-name sampleforcleanup-Stack \ --template-body file://samples/sample-cfn-stack.yaml \ --parameters ParameterKey=VpcId,ParameterValue=<VPCID> \`	AWS DevOps

タスク	説明	必要なスキル
	`--region us-east-1`	
Systems Manager パラメータを作成します。	次のコマンドを入力して、CloudFormation を介してプロビジョニングされない Systems Manager パラメータを作成します。 `aws ssm put-parameter \ --name "/samplefor cleanup/database/password" \ --value "your_db_password" \ --type "SecureString" \ --description "Database password for my app" \ --tier "Standard" \ --region "us-east-1"`	AWS DevOps
Amazon S3 バケットを作成する。	次のコマンドを入力して、CloudFormation を介してプロビジョニングされない Amazon S3 バケットを作成します。 `aws s3api create-bucket \ --bucket samplesorcleanup-unmanagedbucket-<UniqueIdentifier> \ --region us-east-1 \ --create-bucket-configuration LocationConstraint=us-east-1`	AWS DevOps

サンプルリソースを削除する

タスク	説明	必要なスキル
CloudFormation スタックを削除します。	1. 次のコマンドを入力して、作成したサンプルの CloudFormation スタック、Systems Manager パラメータ、および Amazon S3 バケットを削除します。 `cfncli --region us-east-1 \ dev cleanup-env \ --prefix-list sampleforcleanup` 2. プロンプトが表示されたら、「」と入力Yして続行します。	AWS DevOps
リソースの削除を検証します。	出力で、すべてのサンプルリソースが削除されていることを確認します。サンプル出力については、このパターンの「追加のリソース」セクションを参照してください。	AWS DevOps

関連リソース

- スタックを削除する (CloudFormation ドキュメント)
- CloudFormation のトラブルシューティング (CloudFormation ドキュメント)
- Lambda 関数に Amazon VPC のリソースへのアクセスを許可する (Lambda ドキュメント)
- DELETE_FAILED ステータスのままになっている AWS CloudFormation スタックを削除するにはどうすればよいですか？ (AWS ナレッジセンター)

追加情報

以下は、 cfncli コマンドからの出力例です。

```
cfncli --region aus-east-1  dev cleanup-env --prefix-list sampleforcleanup


https://sts.us-east-1.amazonaws.com
Cleaning up: ['sampleforcleanup'] in xxxxxxxxxx:us-east-1
Do you want to proceed? [Y/n]: Y
No S3 buckets
No ECR repositories
No Lambda functions in VPC
The following DynamoDB tables will have their deletion protection removed:
sampleforcleanup-MyDynamoDBTable
Do you want to proceed with removing deletion protection from these tables? [Y/n]: Y
Deletion protection disabled for DynamoDB table 'sampleforcleanup-MyDynamoDBTable'.
The following CloudFormation stacks will be deleted:
sampleforcleanup-Stack
Do you want to proceed with deleting these CloudFormation stacks? [Y/n]: Y
Initiated deletion of CloudFormation stack: `sampleforcleanup-Stack`
Waiting for stack `sampleforcleanup-Stack` to be deleted...
CloudFormation stack `sampleforcleanup-Stack` deleted successfully.
The following ssm_params will be deleted:
/sampleforcleanup/database/password
Do you want to proceed with deleting these ssm_params? [Y/n]: Y
Deleted SSM Parameter: /sampleforcleanup/database/password
Cleaned up: ['sampleforcleanup']
```

AWS Service Catalog と を使用して、Gitflow 環境にホットフィックスソリューションをデプロイするための動的パイプライン管理を自動化する AWS CodePipeline

作成者: Balaji Vedagiri (AWS)、Faisal Shahdad (AWS)、Shanmugam Shanker (AWS)、Vivek Thangamuthu (AWS)

概要

> **Note**
>
> AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、ホットフィックスソリューションを本番環境に安全にデプロイすることのみに特化した動的ホットフィックスパイプラインを管理するシナリオに対処します。このソリューションは、 AWS Service Catalog ポートフォリオと製品を使用して実装および管理されます。Amazon EventBridge ルールは、イベントの自動化に使用されます。制限は、Service Catalog ポートフォリオの制約と AWS Identity and Access Management 開発者向けの (IAM) ロールを使用して適用されます。EventBridge ルールによってトリガーされる Service Catalog 製品の起動は、 AWS Lambda 関数のみが許可されます。このパターンは、「追加情報」で説明されている、特定の Gitflow セットアップを使用する環境向けに設計されています。

通常、修正プログラムは、本番稼働環境などのライブ環境で報告された重大な問題やセキュリティ問題に対処するためにデプロイされます。ホットフィックスはステージング環境と本番稼働環境にのみ直接デプロイする必要があります。ステージングパイプラインと本番パイプラインは、通常の開発リクエストに広く使用されます。これらのパイプラインは、本稼働環境に昇格できない品質保証に継続的な機能があるため、ホットフィックスのデプロイには使用できません。修正をリリースするために、このパターンでは、以下のセキュリティ機能を持つ動的で存続期間の短いパイプラインについて説明します。

- 自動作成 – ホットフィックスパイプラインは、 AWS CodeCommit リポジトリにホットフィックスブランチが作成されるたびに自動的に作成されます。
- アクセス制限 – 開発者は、ホットフィックスプロセス以外でこのパイプラインを作成するためのアクセス権限がありません。

- 制御されたステージ – パイプラインには、特別なアクセストークンを持つ制御されたステージがあり、プルリクエスト (PR) を 1 回のみ作成できます。
- 承認ステージ – 承認ステージは、関連するステークホルダーから必要な承認を得るためにパイプラインに含まれます。
- 自動削除 — `hotfix`ブランチが PR とマージされた後に CodeCommit リポジトリで削除されるたびに、ホットフィックスパイプラインが自動的に削除されます。

前提条件と制限

前提条件

- 次のように 3 つのアクティブな AWS アカウント が必要です。
  - ツールアカウント - 継続的インテグレーションと継続的デリバリー (CI/CD) のセットアップ用。
  - ステージアカウント - ユーザー承認テスト用。
  - 本番稼働用アカウント - ビジネスエンドユーザー向け。
  - (オプション) QA アカウント AWS アカウント として機能する を追加します。このアカウントは、QA を含むメインパイプラインのセットアップとテスト用のホットフィックスパイプラインソリューションの両方が必要な場合に必要です。
- 必要に応じて、メインパイプラインを使用して QA アカウントにデプロイするオプションの条件を持つ AWS CloudFormation スタック。パターンは、`hotfix`ブランチを作成および削除することで、メインパイプラインを設定せずにテストできます。
- Service Catalog 製品の作成に使用される CloudFormation テンプレートを保存する Amazon Simple Storage Service (Amazon S3) バケット。
- コンプライアンス要件 (リポジトリの作成後) に従って、CodeCommit リポジトリの PR 承認ルールを作成します。
- 開発者とチームの IAM アクセス許可を制限すると、prcreation-lambda Lambda 関数はパイプラインからのみ呼び出されるため、実行が拒否されます。

制約事項

- CloudFormation プロバイダーはデプロイステージで使用され、アプリケーションは CloudFormation 変更セットを使用してデプロイされます。別のデプロイオプションを使用する場合は、必要に応じて CodePipeline スタックを変更します。

- このパターンでは、 AWS CodeBuild およびその他の設定ファイルを使用してサンプルマイクロサービスをデプロイします。別のワークロードタイプ (サーバーレスワークロードなど) がある場合は、関連するすべての設定を更新する必要があります。
- このパターンでは、アプリケーションを単一の AWS リージョン（ たとえば、米国東部 (バージニア北部) us-east-1) にデプロイします AWS アカウント。複数のリージョンにデプロイするには、コマンドとスタックでリージョンリファレンスを変更します。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

このセクションの図は、ライフサイクル作成イベントとライフサイクル削除イベントのワークフローを示しています。

ライフサイクルイベントを作成するための前述の図は、以下を示しています。

1. 開発者は CodeCommit リポジトリに`hotfix-*`ブランチを作成して、ホットフィックス関連のソリューションを開発します。
2. `hotfix-*` ブランチ作成イベントは、EventBridge ルールを通じてキャプチャされます。イベントの詳細には、リポジトリ名とブランチ名が含まれます。
3. EventBridge ルールは AWS Lambda 関数 を呼び出します`hotfix-lambda-function`。EventBridge ルールは、イベント情報を入力として Lambda 関数に渡します。
4. Lambda 関数は入力を処理してリポジトリ名とブランチ名を取得します。処理された入力から取得した値を使用して Service Catalog 製品を起動します。
5. Service Catalog 製品には、ソリューションをステージ環境と本番環境にデプロイするパイプライン設定が含まれています。パイプラインブロックには、ソース、ビルド、デプロイの各ステージが含まれます。また、本番環境のデプロイを昇格させる手動承認ステージもあります。
6. ソースステージは、最初のステップで作成されたリポジトリと`hotfix-*`ブランチからコードを取得します。コードは、アーティファクトの Amazon S3 バケットを介してビルドステージに渡されます。ビルドステージでは、 `hotfix-*`ブランチで開発され、Amazon Elastic Container Registry (Amazon ECR) にプッシュされるホットフィックスを含むコンテナイメージが作成されます。

7. ステージ環境へのデプロイステージは、Amazon Elastic Container Service (Amazon ECS) を、 ホットフィックスを含む最新のコンテナイメージで更新します。ホットフィックスは、CloudFormation 変更セットを作成して実行することでデプロイされます。
8. `prcreation-lambda` Lambda 関数は、ステージ環境で正常にデプロイされた後に呼び出されます。この Lambda 関数は、`hotfix-*`ブランチからリポジトリの `develop` および `main`ブランチに PR を作成します。Lambda 関数は、`hotfix-*`ブランチで開発された修正がバックマージされ、後続のデプロイに含まれるようにします。
9. 手動承認ステージは、必要な利害関係者が修正を確認し、本番環境にデプロイするための承認を与えるのに役立ちます。
10. 本番環境へのデプロイステージでは、ホットフィックスを含む最新のコンテナイメージで Amazon ECS が更新されます。ホットフィックスは、CloudFormation 変更セットを作成して実行することでデプロイされます。

ライフサイクルイベントを削除するための前述の図は、次のとおりです。

1. 開発者は、ホットフィックスを本番環境に正常にデプロイした後、`hotfix-*`ブランチを削除します。
2. `hotfix-*` ブランチ削除イベントは、EventBridge ルールを通じてキャプチャされます。イベントの詳細には、リポジトリ名とブランチ名が含まれます。
3. EventBridge ルールは Lambda 関数を呼び出します。EventBridge ルールは、イベント情報を入力として Lambda 関数に渡します。
4. Lambda 関数は入力を処理してリポジトリ名とブランチ名を取得します。Lambda 関数は、渡された入力からそれぞれの Service Catalog 製品を決定し、製品を削除します。
5. Service Catalog でプロビジョニングされた製品の終了により、その製品で以前に作成されたパイプラインおよび関連リソースが削除されます。

自動化とスケール

- このパターンには、EventBridge ルールと Lambda 関数が含まれており、複数の修正ブランチ作成リクエストを並行して処理できます。Lambda 関数は、一致するイベントルールのために Service Catalog 製品をプロビジョニングします。

- パイプラインのセットアップは、バージョン管理機能を提供する Service Catalog 製品を使用して処理されます。また、このソリューションは、同じアプリケーションの複数のホットフィックス開発を並行して処理するように自動的にスケーリングされます。
- prcreation-lambda 関数は、自動プルリクエストの作成を通じて、これらのホットフィックスの変更が `main`と`develop`ブランチにもマージされるようにします。このアプローチは、 `main`と `develop` ブランチをすべての修正で最新の状態に保ち、潜在的なコードのリグレッションを回避するために不可欠です。このプロセスは、存続期間の長いすべてのブランチに最新の修正を適用することで、ブランチ間の一貫性を維持し、コードのリグレッションを防ぐのに役立ちます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体にわたる管理に役立ちます AWS リージョン。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。 AWS CodeCommit は、新規のお客様に利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細については、AWS CodeCommit 「リポジトリを別の Git プロバイダーに移行する方法」を参照してください。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- 「Amazon Elastic Container Registry (Amazon ECR)」 は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Service Catalog は、承認された IT サービスのカタログを一元管理するために役立ちます AWS。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- AWS CloudFormation Linter (cfn-lint) は、CloudFormation YAML テンプレートまたは JSON テンプレートを CloudFormation リソース仕様と照合する linter です。また、リソースプロパティの有効な値の確認やベストプラクティスの遵守など、他のチェックも実行します。
- cfn-nag は、パターンを検索して CloudFormation テンプレートで潜在的なセキュリティ問題を識別するオープンソースツールです。
- 「Docker」は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するPlatform as a Service (PaaS) 製品のセットです。このパターンでは、Docker でコンテナイメージをローカルでビルドしてテストします。
- Git はオープンソースの分散バージョンの管理システムです。

コードリポジトリ

このパターンのコードは、GitHub dynamic_hotfix_codepipeline リポジトリで入手できます。

ベストプラクティス

環境内の IAM ロールとサービスコントロールポリシー (SCP) を確認して調整し、アクセスが適切に制限されていることを確認します。これは、このパターンに含まれるセキュリティ対策を上書きする可能性のあるアクションを防ぐために重要です。最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

作業環境のセットアップ

タスク	説明	必要なスキル
リポジトリをクローン作成します。	サンプルリポジトリを作業場所の新しいディレクトリにクローンするには、次のコマンドを実行します。 `git clone git@githu b.com:aws-samples/`	AWS DevOps

タスク	説明	必要なスキル
	`dynamic_hotfix_cod epipeline.git`	

タスク	説明	必要なスキル

CloudFormation スタックデプロイ用の環境変数を出力します。	このパターンの後半で CloudFormation スタックへの入力として使用される以下の環境変数を定義します。
	
	• ApplicationName – この変数は、アプリケーション用に作成されたリソースに名前を付けるために使用され、追跡が容易になります。次のコマンドを使用して、 を実際のアプリケーション名Applicationname に置き換えます。
	
	`export ApplicationName=<Applicationname>`
	
	• BucketStartName – この変数は、Amazon S3 バケットに名前を付けるためのものです。S3 バケット名は、すべての でグローバルに一意である必要があります AWS アカウント。を S3 バケットの一意の名前BucketName に置き換えて、次のコマンドを使用します。
	
	`export BucketStartName=<BucketName>`
		AWS DevOps

タスク	説明	必要なスキル

	• アカウント番号とリージョン – これらの変数は、さまざまな環境とデプロイリージョンの AWS アカウント番号を保存します。次のコマンドを使用して、プレースホルダー (prodaccountnumber や など region) を実際の AWS アカウント数字と AWS リージョン 使用している に置き換えます。
	
	Note
	QAAccount は省略可能です。を使用する場合はQAAccount 、メインパイプラインスタックのパラメータを使用してセットアップします。
	
	`export ProdAccount=<prodaccountnumber>`
	`export StageAccount=<stage/preprodaccountnumber>`
	`export QAAccount=<qaccountnumber>`
	`export ToolsAccount=<toolsaccountnumber>`	

タスク	説明	必要なスキル
	`export DepRegion =<region>`	

で必要な前提条件を設定する AWS アカウント

タスク	説明	必要なスキル
ツールアカウントで CI/CD に必要なリソースを作成します。	ツールアカウントに CloudFormation スタックをデプロイするには、次のコマンドを使用します。(セットアップに QA アカウントを使用していない場合は、QAAccount パラメータを削除します。) `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/pre-reqs.yaml \ --stack-name prereqs \ --parameter-overrides BucketStartName=${BucketStartName} \ ApplicationName=${ApplicationName} ProdAccount=${ProdAccount} \ StageAccount=${StageAccount} ToolsAccount=${ToolsAccount} \ QAAccount=${QAAccount} \ --capabilities CAPABILITY_IAM`	AWS DevOps

タスク	説明	必要なスキル
	`CAPABILITY_NAMED_IAM --region ${DepRegion}` CodeCommit リポジトリと Amazon ECR が前のスタックから作成したリソースを書き留めます。これらのパラメータは、次のステップでパイプラインのmainブランチをセットアップするために必要です。	

タスク	説明	必要なスキル

ワークロードアカウントで CI/CD に必要なリソースを作成します。	1. 各ワークロードアカウント (ステージ、本番稼働用、およびオプションの QA) に CloudFormation テンプレートをパッケージ化するには、次のコマンドを使用します。次のコマンドで、 をパッケージ化に使用する Amazon S3 バケット名S3bucketpackage に置き換えます。
	
	`#InStageAccount`
	`aws cloudformation package \`
	`--template-file pre-requisites/infrastack.yaml \`
	`--s3-bucket <S3bucketpackage> \`
	`--s3-prefix infraStack \`
	`--region ${DepRegion} \`
	`--output-template-file pre-requisites/infrastructure_stage.template`
	
	`#InProdAccount`
	`aws cloudformation package \`
	`--template-file pre-requisites/infrastack.yaml \`
	`--s3-bucket <S3bucketpackage> \`
		AWS DevOps

タスク	説明	必要なスキル

	<pre>--s3-prefix infraStack \
	--region ${DepRegion} \
	--output-template-file pre-requisites/infrastructure_prod.template</pre>
	2. 各ワークロードアカウントに CloudFormation テンプレートをデプロイするには、次のコマンドを使用します。
	<pre>#InStageAccount
	aws cloudformation deploy --stack-name inframainstack \
	--parameter-overrides Applicationname=${Applicationname} ToolsAccount=${ToolsAccount} DepRegion=${DepRegion} \
	--template-file pre-requisites/infrastructure_stage.template --region ${DepRegion} --capabilities CAPABILITY_NAMED_IAM
	
	#InProdAccount
	aws cloudformation deploy --stack-name inframainstack \</pre>	

タスク	説明	必要なスキル
	--parameter-overrides ApplicationName=${ApplicationName} ToolsAccount=${ToolsAccount} DepRegion=${DepRegion} \ --template-file pre-requisites/infrastructure_prod.template --region ${DepRegion} --capabilities CAPABILITY_NAMED_IAM	

タスク	説明	必要なスキル

S3 アーティファクトバケットポリシーを更新して、ワークロードアカウントへのアクセスを許可します。	ツールアカウントの CloudFormation スタックの前提条件を更新するには、次のコマンドを使用して、ステージワークロードアカウントと本番稼働ワークロードアカウントにすべての必要なアクセス許可を追加します。（セットアップに パラメータを使用していない場合は、QAAccount パラメータを削除します。）
	`#InToolsAccount`
	`aws cloudformation deploy \`
	`--template-file pre-requisites/pre-reqs.yaml \`
	`--stack-name prereqs \`
	`--parameter-overrides BucketStartName=${BucketStartName} \`
	`ApplicationName=${ApplicationName} ProdAccount=${ProdAccount} \`
	`StageAccount=${StageAccount} ToolsAccount=${ToolsAccount} \`
	`QAAccount=${QAAccount} PutPolicy=true \`
	`--capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}`
		AWS DevOps

ツールアカウントで Lambda 関数と Service Catalog リソースを設定する

タスク	説明	必要なスキル
Service Catalog ポートフォリオと製品をセットアップします。	Service Catalog ポートフォリオと製品をセットアップするには、以下を実行します。 1. テンプレート pipeline-main.yaml と pipeline-hotfix.yaml を CodePipeline ディレクトリのリポジトリから、(Bucketname) にデプロイするリージョンの既存の Amazon S3 バケット () にアップロードしますDepRegion 。 ``` #InToolsAccount aws s3 cp ./codepip eline/pipeline-mai n.yaml s3://<Buc ketname>/pipeline- main.yaml aws s3 cp ./codepip eline/pipeline-hot fix.yaml s3://<Buc ketname>/pipeline- hotfix.yaml ``` 2. および hotfixブランチのパイプラインを管理する Service Catalog ポートフォリオmainと製品を設定します。次のスタックの Outputsセクションを参照してください。MainProductId MainProductArtifactId と セク	AWS DevOps

タスク	説明	必要なスキル
	ションの詳細を書き留めます。この情報は、mainブランチのパイプライン設定時の後のステップで必要です。 `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/servicecatalogsetup.yaml \ --stack-name servicecatalogsetup \ --parameter-overrides TemplateBucket=<Bucketname> \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}` 3. ツールアカウントのリソースを Service Catalog ポートフォリオのメインパイプラインポートフォリオにデプロイする IAM ロールへのアクセスを提供します。このポートフォリオを使用して、 mainブランチを使用してアプリケーションをデプロイします。アクセスを提供する方法の詳細については、Service Catalog ドキュメントの「ユーザーへ	

タスク	説明	必要なスキル
	のアクセスの許可」を参照してください。	

タスク	説明	必要なスキル
Lambda 関数を設定します。	このソリューションでは、次の Lambda 関数を使用してホットフィックスワークフローを管理します。 • `hotfix-lambda-function` は、hotfixブランチの作成時に Service Catalog 製品のプロビジョニングを処理します。 • `hotfix-cleanup-lambda-function` hotfixブランチが削除されると、 は製品の終了を管理します。 • `prcreation-lambda` は、hotfixブランチから develop および mainブランチへのプルリクエストを作成します。 hotfix ブランチが関連する EventBridge ルールを通じて作成または削除されたときに、Lambda 関数が Service Catalog 製品をプロビジョニングおよび終了できるようにするには、次の手順を実行します。 1. Lambda 関数の IAM ロールとアクセス許可を作成するには、次のコマンドを使	AWS DevOps

タスク	説明	必要なスキル
	用して CloudFormation スタックをデプロイします。 `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/lambdasetup.yaml \ --stack-name prsclambdasetup \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}` 2. スタックのデプロイ後、 を使用して Service Catalog ポートフォリオホットフィックスパイプラインポートフォリオ`hotfix-lambda-execution-role` へのアクセスを許可しますAWS Management Console。このアクセスにより、Lambda 関数はホットフィックスブランチの Service Catalog 製品を起動または終了できます。	

メインブランチのパイプラインを作成し、ワークロードアカウントにアプリケーションをデプロイする

タスク	説明	必要なスキル
`main` ブランチのパイプラインを設定します。	メインブランチのパイプラインを設定するに	AWS DevOps

タスク	説明	必要なスキル

	は、ツールアカウントで次のコマンドを実行します。MainProductId および のパラメータをservicecatalogsetup スタック出力の値MainProductArtifactId に置き換えます。
	
	`#InToolsAccount`
	`aws servicecatalog provision-product \`
	`--product-id <MainProductId> \`
	`--provisioning-artifact-id <MainProductArtifactId> \`
	`--provisioned-product-name "${ApplicationName}-main-pipeline" \`
	`--provisioning-parameters Key=CodeCommitRepoName,Value="${ApplicationName}-repository" Key=ECRRepository,Value="${ApplicationName}-app" \`
	`--region=${DepRegion}`	

タスク	説明	必要なスキル
main ブランチを使用してアプリケーションをデプロイします。	1. 事前リクエストで作成された CodeCommit リポジトリのクローンを作成するには、 git clone コマンドを使用します。 詳細については、CodeCommit ドキュメント」の説明に従ってリポジトリをクローンして CodeCommit リポジトリに接続する」を参照してください。 CodeCommit 2. リポジトリで使用可能なrepotemplates ディレクトリからすべてのアプリケーションファイルをローカルリポジトリクローン () にコピーします${ApplicationName}-repository 。ID を更新するには、次のファイルを変更しますToolsAccount 。各ファイルで、RegistryAccountid パラメータを見つけ、その値を ToolsAccount ID に設定します。CodeCommit リポジトリに変更をコミットし、 mainと develop ブランチの両方にファイルをプッシュします。 • ecs-configuration-prod.yaml • ecs-configuration-qa.yaml	AWS DevOps

タスク	説明	必要なスキル
	• ecs-configuration-stage.yaml 3. アプリケーションのデプロイを確認するには、 を使用して CodePipeline の実行をモニタリングします AWS Management Console。デプロイが完了したら、ステージ環境で Application Load Balancer FQDN を使用してアプリケーションにアクセスします。アプリケーションが期待どおりに動作することを確認します。 4. 本番環境へのデプロイを承認するには、CodePipeline コンソールを使用してアプリケーションのパイプラインを見つけます。`ApprovalToStart` ステージを見つけます。変更を確認し、満足できる場合は、本番環境のデプロイを続行するため、手動で承認します。	

hotfix-* ブランチのパイプラインを作成し、ホットフィックスをデプロイする

タスク	説明	必要なスキル
`hotfix-*` ブランチを作成し、変更をコミットします。	`hotfix-*` ブランチのパイプラインを作成し、ホットフィックスをワークロードア	AWS DevOps

タスク	説明	必要なスキル
	カウントにデプロイするには、次の手順を実行します。 1. キーワード で始まる名前を使用してブランチを作成しますhotfix。たとえば、このパターンではCodeCommit アプリケーションリポジトリ () の `hotfix-check1` ブランチを使用します`${ApplicationName}-repository` 。詳細については、CodeCommit ドキュメントの「AWS CodeCommit リポジトリに接続する」および「基本的な Git コマンド」を参照してください。CodeCommit 2. Service Catalog 製品が`hotfix-check1` ブランチに対して動的に`Hotfix CICD Pipeline`正常にプロビジョニングされていることを確認します。プロビジョニングされた製品名は、この修正ブランチ名とアプリケーションのCodeCommit リポジトリ名にちなんで命名されます。 3. index.html ファイルにいくつかの小さな変更をコミットし、CodeCommit リポジトリにプッシュします。	

タスク	説明	必要なスキル
	4. ステージ環境で CodePipeline の実行が成功したことを確認します。本番環境にデプロイするには、CodePipeline で手動承認を行います。 5. Application Load Balancer 完全修飾ドメイン名 (FQDN) を使用して、変更がアプリケーションのホームページに表示されることを確認します。FQDN は、の `Outputs`セクションで利用できます`inframainstack-ALBStack-*` 。	
`hotfix-check1` ブランチを削除します。	前に作成した`hotfix-check1` ブランチを削除するには、次の手順を実行します。 1. CodeCommit アプリケーションリポジトリにある`hotfix-check1` ブランチを削除します。 2. `hotfix-check1` ブランチ用にプロビジョニングされた Service Catalog 製品が正常に終了していることを確認します。	AWS DevOps

リソースをクリーンアップする

タスク	説明	必要なスキル

デプロイされたリソースをクリーンアップします。	以前にデプロイされたリソースをクリーンアップするには、次の手順を実行します。
	1. Amazon ECS サービスをワークロードアカウントのゼロレプリカにスケールダウンするには、次のコマンドを使用します。
	`aws ecs update-service --cluster ${ApplicationName}-Cluster --service ${ApplicationName}-Service-stage --desired-count 0 --region ${DepRegion}`
	`aws ecs update-service --cluster ${ApplicationName}-Cluster --service ${ApplicationName}-Service-prod --desired-count 0 --region ${DepRegion}`
	2. `main` ブランチ用にプロビジョニングされた Service Catalog 製品を削除します。
	3. ツールアカウントの Amazon S3 バケットで作成されたオブジェクトをクリーンアップします。レジストリ自体を削除する前
		AWS DevOps

タスク	説明	必要なスキル

	に、Amazon ECR 内のすべての Docker イメージを削除します。
	4. Service Catalog ポートフォリオを削除する前に、Service Catalog ポートフォリオのアクセス許可セクションで IAM ロールを削除します。
	5. ツールアカウントとワークロードアカウントにデプロイされた CloudFormation スタックを削除します。
	
	`##In Tools Account##`
	`aws cloudformation delete-stack --stack-name servicecatalogsetup --region ${DepRegion}`
	`aws cloudformation delete-stack --stack-name prlambdasetup --region ${DepRegion}`
	`aws cloudformation delete-stack --stack-name prereqs --region ${DepRegion}`
	
	`##In Workload Accounts##`
	`aws cloudformation delete-stack --stack-name inframainstack --region ${DepRegion}`	

タスク	説明	必要なスキル
	詳細については、Service Catalog ドキュメントの「プロビジョニング済み製品の削除」を参照してください。	

トラブルシューティング

問題	ソリューション
CodeCommit リポジトリにコミットした変更はデプロイされません。	CodeBuild ログに Docker ビルド操作のエラーがないか確認します。CodeBuild の詳細については、「のドキュメント」を参照してください。
Service Catalog 製品はプロビジョニングされていません。	失敗したイベントについて、関連する CloudFormation スタックを確認します。詳細については、CloudFormation ドキュメントを参照してください。

関連リソース

- 基本的な Git コマンド
- ブランチへのプッシュとマージを制限するように IAM ポリシーを設定する
- AWS CodeCommit リポジトリに接続する
- ユーザーにアクセス権を付与する
- Docker イメージを Amazon ECR プライベートリポジトリにプッシュする
- トラブルシューティング AWS CodeBuild
- とは AWS CodePipeline

追加情報

このパターンは、CI/CD プロセスの開発ワークフローに採用された Gitflow 設定の環境向けに設計されています。パイプラインは、開発から始まり、品質保証 (QA)、ステージ、本番環境を通過するデ

プロイサイクルに従います。CI/CD セットアップには、次のように環境へのプロモーションデプロイを含む 2 つの git ブランチが含まれています。

- `develop` ブランチは開発環境にデプロイされます。
- `main` ブランチは、QA、ステージ、本番環境にデプロイされます。

この設定では、新機能のアクティブな開発が進行中の間、通常のデプロイサイクルよりも速くホットフィックスまたはセキュリティパッチを適用するのが課題です。修正プログラムやセキュリティリクエストに対処し、ライブ環境が適切に機能し、安全に保つには、専用のプロセスが必要です。

ただし、以下の場合、専用のデプロイプロセスを必要としない他の使用可能なオプションを使用できます。

- CI/CD プロセスには、機能テストやend-to-endテストなどの自動テストが備わっているため、手動テストが不要になり、本番環境へのデプロイの遅延を防ぐことができます。ただし、自動テストが CI/CD プロセスに十分に統合されていない場合、本番環境に小さな修正をプッシュすると、開発者にとって複雑で面倒になる可能性があります。これは、QA 環境で承認とサインオフを待っている新機能がある可能性があるためです。修正プログラムまたはセキュリティ修正プログラムを同時に簡単に本番環境にプッシュすることはできません。
- 開発チームは、新しい機能を本番環境に継続的にデプロイし、各新機能のスケジュールされたデプロイにホットフィックスまたはセキュリティパッチを統合します。つまり、本番環境への次の機能更新は、新しい機能の追加と、修正プログラムまたはセキュリティパッチの追加の 2 つのコンポーネントで構成されます。ただし、デプロイサイクルが連続していない場合は、QA 環境で承認を待っている複数の新機能が既に存在する可能性があります。さまざまなバージョンを管理し、正しい変更が再適用されるようにすると、複雑になり、エラーが発生しやすくなります。

> **Note**
>
> バージョン 2 を使用して`hotfix`ブランチに AWS CodePipeline 適切なトリガーを設定している場合は、予定外のリクエストに対応するための専用のプロセスが必要です。バージョン 2 では、プッシュリクエストまたはプルリクエストのトリガーを設定できます。実行は、パイプラインの以前の状態に応じて、キューに入れられるか、すぐに実行されます。ただし、専用パイプラインでは、修正が本番環境にすぐに適用され、緊急の問題が遅滞なく解決されます。

Terraform を使用して Amazon Managed Grafana での Amazon MWAA カスタムメトリクスの取り込みと視覚化を自動化する

作成者: Faisal Abdullah (AWS) と Satya Vajrapu (AWS)

概要

このパターンでは、Amazon Managed Grafana を使用して、Amazon Managed Workflows for Apache Airflow (Amazon MWAA) によって取り込まれるカスタムメトリクスを作成およびモニタリングする方法について説明します。Amazon MWAA はワークフローのオーケストレーターとして機能し、Python でスクリプト化された有向非巡回グラフ (DAGs) を使用します。このパターンは、過去 1 時間以内に実行DAGs の合計数、1 時間あたりの合格および不合格の DAGs の数、これらのプロセスの平均期間など、カスタムメトリクスのモニタリングに焦点を当てています。この分析は、Amazon Managed Grafana が Amazon MWAA と統合して、この環境内のワークフローのオーケストレーションに関する包括的なモニタリングとインサイトを実現する方法を示しています。

前提条件と制限

前提条件

- 以下を作成および管理するために必要なユーザーアクセス許可 AWS アカウント を持つアクティブな AWS のサービス。
  - AWS Identity and Access Management (IAM) ロールとポリシー
  - AWS Lambda
  - Amazon Managed Grafana
  - Apache Airflow 用の Amazon マネージドワークフロー (Amazon MWAA)
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Timestream
- ローカルマシンまたは のターミナルであるシェル環境へのアクセスAWS CloudShell。
- Git がインストールされ、最新バージョンの AWS Command Line Interface （ AWS CLI) がインストールされ、設定されているシェル環境。詳細については、 AWS CLI ドキュメントの「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。
- 次の Terraform `required_version = ">= 1.6.1, < 2.0.0"` バージョンがインストールされました。tfswitch を使用して、Terraform の異なるバージョンを切り替えることができます。
- AWS IAM Identity Center の で ID ソースを設定しました AWS アカウント。詳細については、IAM Identity Center ドキュメントの「IAM Identity Center で ID ソースを確認する」を参照してくださ

い。デフォルト IAM アイデンティティセンターディレクトリ、Active Directory、または Okta などの外部 ID プロバイダー (IdP) から選択できます。詳細については、「関連リソース」を参照してください。

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Terraform `required_version = ">= 1.6.1, < 2.0.0"`
- Amazon Managed Grafana バージョン 9.4 以降。このパターンはバージョン 9.4 でテストされました。

アーキテクチャ

次のアーキテクチャ図は、ソリューション AWS のサービス で使用される を示しています。

前述の図は、以下のワークフローを順を追って示しています。

1. Amazon MWAA 内のカスタムメトリクスは、環境内で実行されている DAGs から取得されます。メトリクスは、CSV ファイル形式で Amazon S3 バケットにアップロードされます。次の DAGs、Amazon MWAA のデータベースクエリ機能を使用します。
   - `run-example-dag` – この DAG には、1 つ以上のタスクを定義するサンプル Python コードが含まれています。7 分ごとに実行され、日付が出力されます。日付の印刷後、DAG には、特定の期間に実行をスリープまたは一時停止するタスクが含まれます。
   - `other-sample-dag` – この DAG は 10 分ごとに実行され、日付を出力します。日付の印刷後、DAG には、特定の期間に実行をスリープまたは一時停止するタスクが含まれます。
   - `data-extract` – この DAG は 1 時間ごとに実行され、Amazon MWAA データベースをクエリしてメトリクスを収集します。メトリクスが収集されると、この DAG はそれらを Amazon S3 バケットに書き込み、さらなる処理と分析を行います。

2. データ処理を効率化するために、Lambda 関数は Amazon S3 イベントによってトリガーされたときに実行されるため、Timestream へのメトリクスのロードが容易になります。
3. Timestream は Amazon Managed Grafana 内のデータソースとして統合され、Amazon MWAA のすべてのカスタムメトリクスが保存されます。
4. ユーザーはデータをクエリし、カスタムダッシュボードを構築して、主要なパフォーマンス指標を視覚化し、Amazon MWAA 内のワークフローのオーケストレーションに関するインサイトを得ることができます。

ツール

AWS のサービス

- AWS IAM Identity Center を使用すると、すべての AWS アカウント およびクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。このパターンでは、 は Amazon S3 イベントに応答して Python コード AWS Lambda を実行し、コンピューティングリソースを自動的に管理します。
- Amazon Managed Grafana は、メトリクス、ログ、トレースのクエリ、相関付け、視覚化、アラートに使用できるフルマネージド型のデータ可視化サービスです。このパターンでは、Amazon Managed Grafana を使用して、メトリクスの視覚化とアラートのためのダッシュボードを作成します。
- Amazon Managed Workflows for Apache Airflow (Amazon MWAA) は、Apache Airflow 用のマネージドオーケストレーションサービスで、クラウド内のデータパイプラインを大規模にセットアップおよび運用するために使用できます。Apache Airflow は、ワークフローと呼ばれる一連のプロセスとタスクをプログラムで作成、スケジュール、モニタリングするために使用されるオープンソースツールです。このパターンでは、サンプル DAGs とメトリクスエクストラクタ DAG が Amazon MWAA にデプロイされます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。このパターンでは、Amazon S3 を使用して DAGs、スクリプト、カスタムメトリクスを CSV 形式で保存します。
- Amazon Timestream for LiveAnalytics は、1 日あたり何兆もの時系列データポイントを簡単に保存および分析できる、高速でスケーラブルなフルマネージド型の専用時系列データベースです。ま

た、LiveAnalytics の Timestream は、データ収集、視覚化、機械学習に一般的に使用されるサービスと統合されます。このパターンでは、生成された Amazon MWAA カスタムメトリクスを取り込むために使用されます。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。このパターンでは、Terraform モジュールを使用してインフラストラクチャのプロビジョニングを自動化しますAWS。

コードリポジトリ

このパターンのコードは、GitHub の visualize-amazon-mwaa-custom-metrics-grafana リポジトリで入手できます。`stacks/Infra` フォルダには以下が含まれています。

- すべての AWS リソースの Terraform 設定ファイル
- `grafana` フォルダ内の Grafana ダッシュボード .json ファイル
- `mwaa/dags` フォルダ内の Amazon Managed Workflows for Apache Airflow DAGs
- .csv ファイルを解析し、メトリクスを `src`フォルダの Timestream データベースに保存するための Lambda コード
- `templates` フォルダ内の IAM ポリシー .json ファイル

ベストプラクティス

Terraform は、実際のリソースを設定にマッピングできるように、マネージドインフラストラクチャと設定に関する状態を保存する必要があります。デフォルトでは、Terraform は という名前のファイルに状態をローカルに保存しますterraform.tfstate。インフラストラクチャの現在の状態が維持されるため、Terraform 状態ファイルの安全性と整合性を確保することが重要です。詳細については、Terraform ドキュメントの「Remote State」を参照してください。

エピック

Terraform を使用してインフラストラクチャをデプロイする

タスク	説明	必要なスキル

インフラストラクチャを準備します。	ソリューションインフラストラクチャをデプロイするには、以下を実行します。
	1. ローカルコンピュータまたは を使用して、ターミナルまたはコマンドプロンプトを開きます AWS CloudShell。
	2. リポジトリのクローンを作成するディレクトリに移動します。
	3. リポジトリのクローンを作成するには、次のコマンドを実行します。
	`git clone https://github.com/aws-samples/visualize-amazon-mwaa-custom-metrics-grafana`
	4. クローン作成プロセスが完了したら、次のコマンドを実行して、クローンされたリポジトリディレクトリに移動します。
	`cd visualize-amazon-mwaa-custom-metrics-grafana/stacks/infra`
		AWS DevOps

タスク	説明	必要なスキル

	5. 必要なプロバイダーをダウンロードして初期化するには、次のコマンドを実行します。
	
	`terraform init`
	
	6. Terraform が作成するすべてのリソースの包括的なビューを取得するには、次のコマンドを実行します。
	
	`terraform plan`
	
	Terraform は次のリソースをプロビジョニングします。
	• Amazon Virtual Private Cloud (Amazon VPC) と関連するネットワークコンポーネント
	• Amazon S3 のリソース
	• AWS Lambda 関数
	• Amazon Managed Grafana リソース (ワークスペース、ダッシュボード、データソース)
	• IAM リソースのサポート (ロールとポリシー)
	7. プラン出力から AWS リソースを作成するには、次のコマンドを実行します。	

タスク	説明	必要なスキル

	`terraform apply -auto-approve`
	
	インフラストラクチャのプロビジョニングは約 20 分で完了します。
	8. Terraform ファイルで定義された設定に従って指定された AWS リソースを作成するには、次のコマンドを実行します。
	
	`terraform apply`	

デプロイされたインフラストラクチャリソースを検証する

タスク	説明	必要なスキル
Amazon MWAA 環境を検証します。	Amazon MWAA 環境を検証するには、以下を実行します。 1. にサインインし AWS Management Console、Amazon MWAA ダッシュボードコンソールに移動し、Open Airflow UI を選択します。 2. Active ステータスの次の 3 つの DAGsが表示されます。 • データ抽出 • run-example-dag	AWS DevOps、データエンジニア

タスク	説明	必要なスキル
	• other-sample-dag 3. DAG がアクティブでない場合は、DAG 名の横にあるトグルスイッチを有効にすることでアクティブ化できます。	
DAG スケジュールを確認します。	各 DAG スケジュールを表示するには、Airflow UI のスケジュールタブに移動します。 以下の各 DAGs には、Amazon MWAA 環境で実行され、カスタムメトリクスを生成する事前設定されたスケジュールがあります。 • run-example-dag - 7 分ごとに実行されます • other-sample-dag - 10 分ごとに実行されます • data-extract - 1 時間ごとに実行されます 各 DAG が正常に実行されたことを Runs 列で確認することもできます。	データエンジニア、AWS DevOps

Amazon Managed Grafana 環境を設定する

タスク	説明	必要なスキル
Amazon Managed Grafana ワークスペースへのアクセスを設定します。	Terraform スクリプトは、必要な Amazon Managed Grafana ワークスペース、ダッシュボード、メトリクスページを作成しました。アクセスを表示できるように設定するには、以下を実行します。 1. Amazon Managed Grafana コンソールを開きます。 2. WorkSpaces でワークスペース を選択しgrafana-ws-dev 、下のペインの認証タブに移動します。 3. 新しいユーザーまたはグループの割り当てボタンを選択します。 4. グループタブにグループを追加するか、ユーザータブにユーザーを追加し、ユーザーとグループの割り当てボタンを選択します。 5. ユーザー (またはグループ) を追加したら、このユーザー (またはグループ) を管理者にします。「割り当てられたユーザー」タブの「割り当てられたユーザーまたはグループ」でユーザーを選択し、ドロップダウンメニューから「管理者にする」を選択します。	AWS DevOps

タスク	説明	必要なスキル
	詳細については、Amazon Managed Grafana ドキュメントの「Amazon Managed Grafana ワークスペース AWS IAM Identity Center で使用する」を参照してください。 6. WorkSpaces に移動し、Grafana ワークスペース URL を選択します。管理者として Amazon Managed Grafana にサインインするには、「 でサインイン AWS IAM Identity Center」を選択します。	

タスク	説明	必要なスキル
Amazon Timestream プラグインをインストールします。	Amazon MWAA カスタムメトリクスは Timestream データベースにロードされます。Timestream プラグインを使用して、Amazon Managed Grafana ダッシュボードでメトリクスを視覚化します。 Timestream プラグインをインストールするには、次の手順を実行します。 1. Amazon Managed Grafana コンソールで、左側のナビゲーションペインのメニューを展開し、管理プラグインに移動します。 2. Amazon Timestream プラグインの最新バージョンを検索してインストールします。 3. プラグインをインストールしたら、管理データソースに移動して Timestream データソースを確認します。データソースが表示されない場合は、ページを更新します。 詳細については、Amazon Managed Grafana ドキュメントの「プラグインを使用してワークスペースを拡張する」を参照してください。	AWS DevOps、DevOps エンジニア

Amazon Managed Grafana ダッシュボードでカスタムメトリクスを視覚化する

タスク	説明	必要なスキル
Amazon Managed Grafana ダッシュボードを表示します。	Amazon Managed Grafana ワークスペースに取り込まれたメトリクスを表示するには、次の手順を実行します。 1. Amazon Managed Grafana コンソールで、左側のナビゲーションペインのダッシュボードを選択します。 2. メトリクスを表示するには、MWAA イベントダッシュボードを選択し、mwaa_metrics を選択します。 ダッシュボードのメトリクスページには、次の情報が表示されます。 • 過去 1 時間の合計 DAG 実行数 • 過去 1 時間以内に成功、失敗、実行された DAG 実行の合計数 • すべての DAG 実行、成功した DAG 実行、失敗した DAG 実行の平均期間	AWS DevOps
Amazon Managed Grafana ダッシュボードをカスタマイズします。	今後の機能強化のためにダッシュボードをカスタマイズするには、以下を実行します。	AWS DevOps

タスク	説明	必要なスキル
	1. Amazon Managed Grafana ダッシュボード`mwaa_metrics` ページで、ダッシュボード設定アイコンを選択します。 2. ダッシュボードを定義するデータ構造を表示するには、JSON モデルを選択します。ダッシュボードをカスタマイズするには、この JSON モデルをコンソールで直接編集します。 または、このダッシュボードのソースコードは、GitHub リポジトリの `stacks/infra/grafana` フォルダの `dashboard.json` ファイルで使用できます。	

AWS リソースをクリーンアップする

タスク	説明	必要なスキル
Amazon MWAA DAG の実行を一時停止します。	DAG 実行を一時停止するには、以下を実行します。 1. Amazon MWAA コンソールで、Airflow 環境に移動し、Open Airflow UI を選択します。 2. DAG を一時停止するには、各 DAG の横にあるトグルスイッチを使用します。	AWS DevOps、データエンジニア

タスク	説明	必要なスキル
	3. Airflow UI ページを更新し、一時停止セクションに 3 つの DAGs を一覧表示します。	
Amazon S3 バケット内のオブジェクトを削除します。	Amazon S3 バケット mwaa-events-bucket-* および mwaa-metrics-bucket-* を削除するには、Amazon S3 ドキュメントの「バケットの削除」で Amazon S3 コンソールを使用する手順に従ってください。	AWS DevOps

タスク	説明	必要なスキル
Terraform によって作成されたリソースを破棄します。	Terraform によって作成されたリソースと関連するローカル Terraform 状態ファイルを破棄するには、次の手順を実行します。 1. （オプション) リソースを削除する前に、Terraform が行う変更をプレビューできます。プランを生成するには、次のコマンドを実行します。 `terraform plan -destroy` コマンド出力は、destroyコマンドが以前に作成されたすべての AWS リソースを削除することを示しています。 2. `terraform destroy -auto-approve` このコマンドは、インフラストラクチャを破棄するのに約 20 分かかります。 **Note** Terraform によって管理されるすべてのリソースを破棄するには、次のコマンドを実行します。:	AWS DevOps

タスク	説明	必要なスキル

	`-auto-approve` タグは、ユーザーの確認がリソースの破棄を開始するまで待機しません。
	3. ローカル Terraform 状態ファイルを削除するには、次のコマンドを実行します。
	`rm .terraform.lock.hcl`
	`rm -rf .terraform`
	`rm terraform.tfstate*`	

トラブルシューティング

問題	ソリューション
`null_resource.plugin_mgmt (local-exec): aws: error: argument operation: Invalid choice, valid choices are:`	AWS CLI を最新バージョンにアップグレードします。
データソースのロードエラー - `Fetch error: 404 Not Found Instantiating…`	エラーは断続的です。数分待ってからデータソースを更新し、リストされた Timestream データソースを表示します。

関連リソース

AWS ドキュメント

- ダッシュボードと視覚化のための Amazon Managed Grafana
- Okta を使用するように Amazon Managed Grafana を設定する
- Amazon Managed Grafana ワークスペース AWS IAM Identity Center で を使用する
- Amazon MWAA での DAGs の使用

AWS ビデオ

- 次のビデオに示すように、認証用に Amazon Managed Grafana を使用して IAM Identity Center を設定します。

https://www.youtube-nocookie.com/embed/XX2Xcz-Ps9U?controls=0

- IAM Identity Center が利用できない場合は、次の動画に示すように、Okta などの外部 ID プロバイダー (IdP) を使用して Amazon Managed Grafana 認証を統合することもできます。

https://www.youtube-nocookie.com/embed/Z4JHxl2xpOg?controls=0

追加情報

Amazon MWAA 環境の包括的なモニタリングおよびアラートソリューションを作成し、プロアクティブな管理と潜在的な問題や異常への迅速な対応を可能にします。Amazon Managed Grafana には次の機能が含まれています。

アラート – 事前定義されたしきい値または条件に基づいて、Amazon Managed Grafana でアラートを設定できます。特定のメトリクスが指定されたしきい値を上回ったり下回ったりした場合に、関連するステークホルダーに警告する E メール通知を設定します。詳細については、Amazon Managed Grafana ドキュメントの「Grafana アラート」を参照してください。

統合 – Amazon Managed Grafana を OpsGenie、PagerDuty、Slack などのさまざまなサードパーティーツールと統合して、通知機能を強化できます。例えば、Amazon Managed Grafana で生成されたアラートに基づいて、ウェブフックを設定したりAPIs と統合したりして、これらのプラットフォームでインシデントや通知をトリガーしたりできます。さらに、このパターンはリソースを作成するための GitHub リポジトリを提供します AWS 。このコードをインフラストラクチャデプロイワークフローとさらに統合できます。

自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する

作成者: Balaji Panneerselvam (AWS)、Anand Jumnani (AWS)、Attila Dancso (AWS)、James O'Hara (AWS)、Pavan Dusanapudi (AWS)

概要

Amazon Lex 会話ボットの開発とデプロイは、複数の機能、デベロッパー、環境を管理しようとすると難しい場合があります。Infrastructure as Code (IaC) の原則を使用した自動ワークフローは、プロセスを合理化するのに役立ちます。このパターンは、Amazon Lex 開発者の生産性を向上させ、次の方法で効率的なボットライフサイクル管理を可能にするのに役立ちます。

- 複数の機能の同時開発を有効にする - 自動ワークフローを使用すると、デベロッパーは異なる機能を別々のブランチで並行して作業できます。その後、他の作業をブロックすることなく、変更をマージしてデプロイできます。
- Amazon Lex コンソール UI を使用する - 開発者は、ユーザーフレンドリーな Amazon Lex コンソールを使用してボットを構築およびテストできます。その後、ボットはデプロイ用のインフラストラクチャコードで記述されます。
- 環境間でボットを昇格させる - ワークフローは、開発やテストなどの下位環境から本番環境へのボットバージョンの昇格を自動化します。このアプローチにより、手動プロモーションのリスクとオーバーヘッドが軽減されます。
- バージョン管理の維持 - Amazon Lex サービスを通じてのみではなく、Git でボット定義を管理することで、バージョン管理と監査証跡が提供されます。または AWS Management Console APIs のみを使用して に保存されているボットを変更する場合とは異なり、変更は個々のデベロッパーに対して追跡されます AWS。

Amazon Lex ボットのリリースプロセスを自動化することで、チームはリスクと労力を軽減して、より迅速に機能を提供できます。ボットは、Amazon Lex コンソールで分離されるのではなく、バージョン管理の対象のままです。

前提条件と制限

前提条件

- ワークフローには、異なる環境 (開発、本番稼働、DevOps) AWS アカウント に対して複数の が含まれます。そのためには、アカウント管理とクロスアカウントアクセス設定が必要です。

- Python 3.9 は、デプロイ環境またはパイプラインで使用できます。
- Git は、ソース管理のためにローカルワークステーションにインストールおよび設定されています。
- AWS Command Line Interface （ AWS CLI) がインストールされ、コマンドラインまたは Python を使用して認証されるように設定されています。

機能制限

- リポジトリアクセス – ワークフローは、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインに、ソースコードリポジトリに変更をコミットするために必要なアクセス許可があることを前提としています。
- 初期ボットバージョン – ツールでは、 AWS CloudFormation テンプレートを使用してボットの初期バージョンをデプロイする必要があります。自動ワークフローを引き継ぐ前に、ボットの最初の反復を作成し、リポジトリにコミットする必要があります。
- マージ競合 – ワークフローは同時開発を有効にすることを目的としていますが、異なるブランチからの変更を統合するとマージ競合が発生する可能性があります。ボット設定の競合を解決するには、手動による介入が必要になる場合があります。

製品バージョン

- Python 3.9 以降
- AWS CDK v2 2.124.0 以降
- AWS SDK for Python (Boto3)1.28 以降

アーキテクチャ

次の図は、ソリューションの高レベルのアーキテクチャと主要コンポーネントを示しています。

主なコンポーネントは次のとおりです。

- Lex ボットリポジトリ – Amazon Lex ボットの IaC 定義を保存する Git リポジトリ。 Amazon Lex
- DevOps – 開発およびデプロイプロセス用の CI/CD パイプラインおよび関連リソースを格納する AWS アカウント 専用の です。

- パイプライン – ボットの開発とデプロイのライフサイクルのさまざまな段階を自動化する AWS CodePipeline インスタンス。新しいボットの作成、ボットの定義のエクスポート、ボット定義のインポート、ボットの削除などです。
- チケットボットとメインボット – Amazon Lex ボットリソース。チケットボットは個々のチームまたは開発者によって開発された機能固有のボットであり、メインボットはすべての機能を統合するベースラインボットです。

アーキテクチャ図は、次のワークフローを示しています。

1. ベースラインメインボット – ワークフローの開始点は、開発 (Dev) 環境でメインボットのベースラインを設定することです。メインボットは、将来の開発と機能の追加の基盤として機能します。
2. チケットボットの作成 – 新機能または変更が必要な場合、チケットボットが作成されます。チケットボットは基本的に、デベロッパーがメインバージョンに影響を与えることなく操作できるメインボットのコピーまたはブランチです。
3. チケットボットのエクスポート - チケットボットでの作業が完了すると、Amazon Lex サービスからエクスポートされます。次に、チケットボットを含むブランチは、メインブランチから再ベースされます。このステップにより、チケットボットの開発中にメインボットに加えられた変更が確実に組み込まれ、潜在的な競合が軽減されます。
4. 再ベースのチケットボットをインポートして検証する – 再ベースのチケットボットは開発環境にインポートされ、メインブランチからの最新の変更で正しく機能することを確認します。検証が成功すると、プルリクエスト (PR) が作成され、チケットボットの変更がメインブランチにマージされます。
5. チケットボットの削除 – 変更がメインブランチに正常にマージされると、チケットボットは必要ありません。チケットボットを削除して、環境をクリーンで管理しやすいようにすることができます。
6. メインボットを開発環境にデプロイしてテストする – 新機能や変更を含む更新されたメインボットが開発環境にデプロイされます。ここでは、すべての機能が期待どおりに動作することを確認するために、徹底的なテストが行われます。
7. メインボットを本番環境にデプロイする – 開発環境でのテストが完了し、成功すると、メインボットが本番環境にデプロイされます。このステップは、ワークフローの最終段階であり、エンドユーザーが新機能を利用できるようになります。

自動化とスケール

自動ワークフローにより、デベロッパーは異なる機能を別々のブランチで並行して作業できます。これにより、同時開発が容易になり、チームは効果的にコラボレーションし、機能をより迅速に提供できます。ブランチが互いに分離されている場合、他の進行中の作業をブロックしたり妨害したりすることなく、変更をマージしてデプロイできます。

ワークフローは、開発、テスト、本番稼働など、さまざまな環境にわたるボットバージョンのデプロイと昇格を自動化します。

Git などのバージョン管理システムにボット定義を保存すると、包括的な監査証跡が提供され、効率的なコラボレーションが可能になります。変更は個々のデベロッパーに対して追跡され、開発ライフサイクル全体で透明性と説明責任が確保されます。また、このアプローチによりコードレビューが容易になり、チームは本番環境にデプロイする前に問題を特定して対処できます。

AWS CodePipeline などを使用することで AWS のサービス、自動化ワークフローはワークロードとチームサイズの増大に合わせて拡張できます。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、使い慣れたプログラミング言語を使用してコード内の AWS クラウド インフラストラクチャを定義し、それをプロビジョニングするためのオープンソースのソフトウェア開発フレームワークです AWS CloudFormation。このパターンのサンプル実装では、Python を使用します。
- AWS CDK コマンドラインインターフェイス (AWS CDK CLI) - Toolkit AWS CDK は、 AWS CDK アプリを操作するための主要なツールです。アプリケーションを実行し、定義したアプリケーションモデルを調査し、CDK によって生成された AWS CloudFormation テンプレートを生成してデプロイします。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。このパターンでは、CloudFormation を使用して、Infrastructure as Code を使用して Amazon Lex ボット設定と関連リソースをデプロイします。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。このパターンでは、CodeBuild を使用してデプロイアーティファクトを構築およびパッケージ化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。このパターンでは、CodePipeline を使用して継続的デリバリーパイプラインを調整します。

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドAWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Lex V2 は、音声とテキストを使用してアプリケーションの会話インターフェイス (ボット) を構築 AWS のサービス するための です。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。

その他のツール

- Git はオープンソースの分散バージョン管理システムです。

コードリポジトリ

このパターンのコードは、GitHub management-framework-sample-for-amazon-lex リポジトリで入手できます。コードリポジトリには、次のフォルダとファイルが含まれています。

- `prerequisite` フォルダ – 必要なリソースと環境を設定するための CloudFormation スタック定義 (を使用 AWS CDK) が含まれています。
- `prerequisite/lexmgmtworkflow` folder – スタック定義や Python コードなど、Lex 管理ワークフロープロジェクトのメインディレクトリ。
- `prerequisite/tests` – ユニットテストが含まれます。
- `src` フォルダ – Amazon Lex ボット管理ラッパーとユーティリティを含むソースコードディレクトリ。
- `src/dialogue_lambda` – Amazon Lex ボットとの会話中にユーザー入力を傍受して処理するダイアログフック Lambda 関数のソースコードディレクトリ。

ベストプラクティス

- 懸念事項の分離
  - DevOps、開発、本番環境間の責任を明確に分離します。

  - 環境 AWS アカウント ごとに個別の を使用して、適切な分離とセキュリティの境界を適用します。
  - クロスアカウントロールと最小特権のアクセス原則を使用して、環境間の制御されたアクセスを確保します。
- コードとしてのインフラストラクチャ
  - ベストプラクティスと進化する要件に合わせて、インフラストラクチャコードを定期的に見直して更新します。
  - ソースコードリポジトリの明確な分岐とマージ戦略を確立する
- テストと検証
  - パイプラインのさまざまな段階で自動テストを実装して、開発サイクルの早い段階で問題を検出します。
  - Amazon Lex コンソールまたは自動テストフレームワークを使用して、ボットの設定と機能を検証してから、より高い環境に昇格します。
  - 本番環境または重要な環境へのデプロイには、手動承認ゲートの実装を検討してください。
- モニタリングとログ記録
  - パイプライン、デプロイ、ボットインタラクションのモニタリングとログ記録のメカニズムを設定します。
  - パイプラインイベント、デプロイステータス、ボットパフォーマンスメトリクスをモニタリングして、問題を迅速に特定して対処します。
  - Amazon CloudWatch などの AWS のサービス、および を使用して AWS CloudTrail、一元的なログ記録とモニタリング AWS X-Ray を行います。
  - 自動ワークフローのパフォーマンス、効率、有効性を定期的に確認して分析します。
- セキュリティとコンプライアンス
  - 安全なコーディングプラクティスを実装し、Amazon Lex ボットの開発とデプロイ AWS のセキュリティのベストプラクティスに従います。
  - 最小特権の原則に従って、IAM ロール、ポリシー、アクセス許可を定期的に確認および更新します。
  - セキュリティスキャンとコンプライアンスチェックをパイプラインに統合することを検討してください。

エピック

Amazon Lex ボット管理用の IaC を設定する Amazon Lex

タスク	説明	必要なスキル

ローカル CDK 環境を設定します。	1. このパターンのリポジトリのクローンを作成し、`prerequisite` ディレクトリに移動するには、次のコマンドを実行します。
	`git clone https://github.com/aws-samples/management-framework-sample-for-amazon-lex.git`
	`cd management-framework-sample-for-amazon-lex`
	2. Python 仮想環境をインストールしてアクティブ化するには、次のコマンドを実行して、CDK の依存関係をグローバルではなくプロジェクトフォルダにローカルにインストールします。
	`pip install virtualenv`
	`python<version> -m venv .venv`
	`source .venv/bin/activate`
	`python -m pip install -r requirements.txt`
		AWS DevOps

タスク	説明	必要なスキル

devops 環境でクロスアカウントロールを作成します。	devops アカウントは、CI/CD パイプラインのホスティングと管理を担当します。CI/CD パイプラインが devおよび prod環境とやり取りできるようにするには、次のコマンドを実行してdevops、アカウントにクロスアカウントロールを作成します。
	`cdk bootstrap --profile =devops`
	
	`cdk deploy LexMgmtDe vopsRoleStack -c dev-account-id=222 2222222222 -c prod- account-id=33333333333 3 --profile=devops`
		AWS DevOps

タスク	説明	必要なスキル

dev 環境でクロスアカウントロールを作成します。	dev アカウントがこのロールを引き受けることができるようにするために必要なアクセス許可を持つ IAM ロールをdevopsアカウントに作成します。CI/CD パイプラインは、このロールを使用して、Amazon Lex ボットリソースのデプロイや管理など、devアカウントでアクションを実行します。
	
	IAM ロールを作成するには、次のコマンドを実行します。
	
	`cdk bootstrap --profile=dev`
	
	`cdk deploy LexMgmtCrossaccountRoleStack -c devops-account-id=1111111111111 --profile=dev`
		AWS DevOps

タスク	説明	必要なスキル

prod 環境でクロスアカウントロールを作成します。	prod アカウントがこのロールを引き受けることができるようにするために必要なアクセス許可を持つ IAM ロールをdevopsアカウントに作成します。CI/CD パイプラインは、このロールを使用して、Amazon Lex ボットリソースのデプロイや管理など、prodアカウントでアクションを実行します。
	`cdk bootstrap --profile =prod`
	`cdk deploy LexMgmtCr ossaccountRoleStac k -c devops-account- id=1111111111111 -- profile=prod`
		AWS DevOps

devops 環境にパイプラインを作成します。	Amazon Lex ボットの開発ワークフローを管理するには、次のコマンドを実行してdevops環境 でパイプラインを設定します。 `cdk deploy LexMgmtWo rkflowStack -c devops- account-id=111111111 1111 -c dev-account- id=2222222222222 -c prod-account-id=33 3333333333 --profile =devops`	AWS DevOps

メインボットのベースラインを確立する

タスク	説明	必要なスキル
メインボットの初期バージョンを定義します。	メインボットの初期バージョンを定義するには、`BaselineBotPipeline` パイプラインをトリガーします。 パイプラインは、CloudFormation テンプレートで定義されている基本的なボット定義をデプロイし、メインボット定義を .json ファイルとしてエクスポートします。 は、メインボットコードをバージョン管理システムに保存します。	AWS DevOps

機能開発ワークフローを実装する

タスク	説明	必要なスキル
チケットボットを作成して、機能を開発およびテストします。	`TicketBot` は、機能ブランチの既存のメインボット定義からインポートされる新しいボットインスタンスです。このアプローチにより、新しいボットにはメインボットからの現在の機能と設定がすべて含まれます。 チケットボットの初期バージョンを定義するには、`CreateTicketBotPipeline` パイプラインをトリガーします。	Lex ボット開発者

タスク	説明	必要なスキル
	パイプラインは、バージョン管理システムに新しい機能ブランチを作成し、メインボットに基づいて新しいチケットボットインスタンスを作成します。	
チケットボット機能を開発してテストします。	機能を開発してテストするには、 にサインイン AWS Management Console し、Amazon Lex コンソールを https://console.aws.amazon.com/lex/:://www..com で開きます。詳細については、Amazon Lex ドキュメントの「コンソールを使用したボットのテスト」を参照してください。 `TicketBot` インスタンスで、ボットの機能を追加、変更、または拡張して、新機能を実装できるようになりました。たとえば、インテント、発話、スロット、ダイアログフローを作成または変更できます。詳細については、Amazon Lex ドキュメントのインテントの追加を参照してください。	Lex ボット開発者

タスク	説明	必要なスキル
チケットボット定義をエクスポートします。	エクスポートされたボット定義は、基本的にボットの設定と機能を JSON 形式で表現したものです。 チケットボット定義をエクスポートするには、`ExportTicketBotPipeline` パイプラインをトリガーします。 パイプラインは、チケットボット定義を .json ファイルとしてエクスポートし、バージョン管理システムの機能ブランチにチケットボットコードを保存します。	Lex ボット開発者
最新のメインブランチから機能ブランチを再ベースします。	新機能の開発中に、メインブランチはさまざまな開発者やチームから他の変更を受け取った可能性があります。 これらの変更を機能ブランチに組み込むには、Git `rebase`オペレーションを実行します。このオペレーションでは、基本的に、メインブランチからの最新のコミットに加えて特徴量ブランチからのコミットをリプレイし、特徴量ブランチに最新の変更がすべて含まれるようにします。	Lex ボット開発者

タスク	説明	必要なスキル
再ベースのチケットボットをインポートして検証します。	機能ブランチを再ベースしたら、それをチケットボットインスタンスにインポートする必要があります。このインポートにより、既存のチケットボットが再ベースブランチからの最新の変更で更新されます。 再ベースのチケットボットをインポートするには、`ImportTicketBotPipeline` パイプラインをトリガーします。 パイプラインは、バージョン管理システムの機能ブランチのチケットボット定義 .json ファイルを`TicketBot` インスタンスにインポートします。	Lex ボット開発者

タスク	説明	必要なスキル
再ベースのボット定義を検証します。	再ベースのボット定義をインポートしたら、その機能を検証することが重要です。新機能が期待どおりに動作し、既存の機能と競合しないようにする必要があります。 この検証には、通常、さまざまな入力シナリオでボットをテストし、レスポンスをチェックし、ボットが意図したとおりに動作することを確認します。検証は、次のいずれかの方法で実行できます。 • Amazon Lex コンソールを使用してボットを手動でテストします。 • ユーザーとのやり取りをシミュレートし、予想されるレスポンスをアサートできるテストフレームワークとツールを使用して、自動化されたアプローチを使用します。	Lex ボット開発者

タスク	説明	必要なスキル
機能ブランチをメインブランチにマージします。	分離されたTicketBot インスタンスで新機能を開発してテストしたら、次の操作を行います。 1. バージョン管理システムの対応する機能ブランチに変更をコミットします。 2. 機能ブランチをメインブランチにマージするには、プルリクエスト (PR) を作成します。この PR は、変更を確認してメインコードベースに組み込むリクエストとして機能します。	Lex Bot Developer、リポジトリ管理者

タスク	説明	必要なスキル
機能ブランチとチケットボットを削除します。	機能ブランチがメインブランチに正常にマージされたら、ソースコードリポジトリから機能ブランチとチケットボットを削除します。 機能ブランチとチケットボットを削除するには、`DeleteTicketBotPipeline` パイプラインをトリガーします。 パイプラインは、開発プロセス中に作成された一時的なボットリソース (チケットボットなど) を削除します。このアクションは、クリーンなリポジトリを維持し、将来の機能ブランチとの混同や競合を防ぐのに役立ちます。	Lex ボット開発者

メインボットのメンテナンス

タスク	説明	必要なスキル
最新のメインボット定義をdev環境にインポートします。	メインブランチの最新のメインボット定義をdev環境にインポートするには、`DeployBotDevPipeline` パイプラインをトリガーします。 パイプラインは、承認時に git タグも作成します。	AWS DevOps

タスク	説明	必要なスキル
最新のメインボット定義をprod環境にインポートします。	メインブランチの最新のボット定義をprod環境にインポートするには、前のタスクのタグリファレンスをパラメータとして指定し、DeployBotProdPipeline パイプラインをトリガーします。 パイプラインは、特定のタグからprod環境に最新のボット定義をインポートします。	AWS DevOps

トラブルシューティング

問題	ソリューション
Amazon Lex ボットを別の にデプロイする場合 AWS アカウント、ツールサービスには、それらのアカウントのリソースにアクセスするために必要なアクセス許可が必要です。	クロスアカウントアクセスを許可するには、IAM ロールとポリシーを使用します。ターゲットアカウントに IAM ロールを作成し、必要なアクセス許可を付与するロールにポリシーをアタッチします。次に、Amazon Lex ボットがデプロイされているアカウントからこれらのロールを引き受けます。 詳細については、Amazon Lex ドキュメントの「インポートに必要な IAM アクセス許可」および「Lex V2 でボットをエクスポートするために必要な IAM アクセス許可」を参照してください。 Amazon Lex

関連リソース

- Amazon Lex V2 でのボットのインポート V2
- CodePipeline でパイプラインを開始する

- Amazon Lex V2 ボットの使用

AWS CodePipeline および AWS CodeBuild を使用して、スタックセットデプロイを自動化

作成者：Thiyagarajan Mani (AWS), Mihir Borkar (AWS), と Raghu Gowda (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

継続的統合と継続的デリバリ (CI/CD) のプロセスでは、既存のすべてのAWSアカウントと、AWS Organizationsで組織に追加する新しいアカウントにアプリケーションを自動的にデプロイする場合があります。この要件に対応する CI/CD ソリューションを設計する場合、AWS CloudFormation の委任スタックセット管理者 キャパシティが役立ちます。管理アカウントへのアクセスを制限することでセキュリティレイヤーを実現できるためです。ただし、AWS CodePipeline はサービス管理型のアクセス権限モデルを使用して、アプリケーションを複数のアカウントとリージョンにデプロイします。AWS CodePipeline に委任スタックセット管理者機能が適用されないため、 AWS Organizations 管理アカウントを使用して、スタックセットでデプロイします。

このパターンでは、この制限を回避する方法を示しています。このパターンでは、AWS CodeBuild とカスタムスクリプトを使用して、AWS CodePipeline 付けのスタックセットのデプロイを自動化します。以下のアプリケーションデプロイアクティビティを自動化します。

- アプリケーションをスタックセットとして既存の組織単位 (OU) にデプロイ
- アプリケーションのデプロイをその他の OU やリージョンにまで拡張
- デプロイされたアプリケーションをすべてまたは特定の OU またはリージョンから削除

前提条件と制限

前提条件

このパターンのステップに従う前に、

- AWS Organizations の管理アカウントで組織を作成します。手順については、AWS Organizations ドキュメントを参照してください。
- AWS Organizations と CloudFormation 間の信頼できるアクセスを有効にして、サービス管理の権限を使用します。手順について、CloudFormation ドキュメントのAWS Organizations との信頼できるアクセスを有効にする を参照してください。

機能制限

このパターンで提供されるコードには以下の制限があります。

- 1 つのアプリケーションにデプロイする CloudFormation テンプレートは 1 つだけです。複数のテンプレートのデプロイが現在サポートされていません。
- 現在の実装をカスタマイズするには、DevOpsの専門知識が必要です。
- このパターンでは AWS キー管理システム (AWS KMS) キーは使用されません。ただし、このパターンに含まれている CloudFormation テンプレートを再設定することで、この機能を有効にできます。

アーキテクチャ

この CI/CD デプロイパイプラインのアーキテクチャは、以下を処理します。

- スタックセットデプロイの責任を、アプリケーションデプロイのスタックセット管理者として専用 CI/CD アカウントに委任することで、管理アカウントへの直接アクセスを制限します。
- サービス管理型の権限モデルを使用して、OU に新しいアカウントが作成されマッピングされるたびに、アプリケーションを自動的にデプロイします。
- 環境レベルですべてのアカウントでアプリケーションバージョンの一貫性を確保します。
- リポジトリレベルとパイプラインレベルで複数の承認ステージを使用して、デプロイされたアプリケーションのセキュリティとガバナンスをさらに強化します。
- CodeBuild のカスタムビルドのデプロイスクリプトを使用して、スタックセットとスタックインスタンスを自動的にデプロイまたは削除することで、CodePipeline の現在の制限を克服します。カスタムスクリプトによって実装される API 呼び出しのフロー制御と階層の図について、追加情報セクションを参照してください。
- 開発環境、テスト環境、本番環境用に個別のスタックセットを作成します。さらに、各段階で複数の OU とリージョンを組み合わせたスタックセットを作成できます。たとえば、開発デプロイメント段階でサンドボックスと開発 OU を組み合わせることができます。
- アカウントのサブセットや OU リストへのアプリケーションのデプロイや除外が適用されます。

自動化とスケール

このパターンで提供されるコードを使用して、AWS CodeCommit リポジトリとアプリケーション用のコードパイプラインを作成できます。その後、これらをスタックセットとして OU レベルで複数

のアカウントにデプロイできます。このコードでは、承認者に通知する Amazon Simple Notification Service (Amazon SNS) トピック、必要な AWS Identity and Access Management (IAM) ロール、管理アカウントに適用するサービスコントロールポリシー (SCP) などのコンポーネントも自動化されます。

ツール

AWS サービス

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- AWS CodeBuild は完全マネージド型の構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS Organizations は、作成して一元管理している複数の AWS アカウントを組織に統合するためのアカウント管理サービスです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

コードリポジトリ

このパターンのコードは、GitHub 内の「自動コードパイプラインスタックセットデプロイ」リポジトリで利用できます。フォルダ構造やその他の詳細については、リポジトリの readme ファイルを参照してください。

ベストプラクティス

このパターンでは、OU レベルでアプリケーションをデプロイしている間、管理アカウントへの直接アクセスが制限されます。パイプラインとリポジトリのプロセスに複数の承認ステージを追加するこ

とで、このアプローチを使用してデプロイするアプリケーションとコンポーネントのセキュリティとガバナンスを強化できます。

エピック

AWS Organizations アカウントを設定

タスク	説明	必要なスキル
管理アカウントですべての特徴量を有効にします。	AWS Organizations ドキュメント の指示に従って、組織の管理アカウントのすべての機能を有効にします。	AWS 管理者、プラットフォーム管理者
CI/CD アカウントを作成します。	AWS Organizations では、組織内で専用の CI/CD アカウントを作成し、そのアカウントへのアクセスを所有および管理するチームを割り当てます。	AWS 管理者
委任管理者を追加する	管理アカウントに、前のステップで作成した CI/CD アカウントを委任スタックセット管理者として登録します。手順については、AWS CloudFormation のドキュメント を参照してください。	AWS 管理者、プラットフォーム管理者

アプリケーションリポジトリと CI/CD パイプラインを作成

タスク	説明	必要なスキル
コードリポジトリを複製します。	1. このパターンで提供されるコードリポジトリをコンピューターに複製します。	AWS DevOps

タスク	説明	必要なスキル
	``` git clone https://github.com/aws-samples/automated-code-pipeline-stackset-deployment.git ``` 2. readme ファイル をレビューして、ディレクトリー構造やその他の詳細を理解します。	

タスク	説明	必要なスキル
SNS トピックを作成します。	GitHub リポジトリに用意されている `sns-template.yaml` テンプレートを使用して SNS トピックを作成し、承認リクエストのサブスクリプションを設定できます。 1. AWS コンソールで、CI/CD アカウントにサインインします。 2. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation/) を開きます。 3. 新しいリソースを使用して、スタックを作成します (標準)。 4. テンプレートを指定では、テンプレートファイルをアップロー、ファイルを選択の順に選択し、クローンした GitHub リポジトリの `templates` フォルダから `sns-template.yaml` ファイルを選択します。次へ をクリックします。 5. わかりやすいアプリケーションスタック名を提供します。 6. リソースのプレフィックスを指定します。	AWS DevOps

タスク	説明	必要なスキル
	7. 次へ 、次へ、送信を選択します。 8. スタックが正常に作成されたら、出力タブを選択し、プルリクエスト、テスト環境、および本番環境の SNS トピックの Amazon リソースネーム (ARN) を書き留めます。この情報は次のステップで使用します。	

タスク	説明	必要なスキル
CI/CD コンポーネントの IAM ロールを作成します。	GitHub リポジトリに用意されている `cicd-role-template.yaml` テンプレートを使用して、CI/CD コンポーネントに必要な IAM ロールとポリシーを作成できます。 1. AWS コンソールで、CI/CD アカウントにサインインします。 2. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation/) を開きます。 3. 新しいリソースを使用して、スタックを作成します (標準)。 4. テンプレートを指定では、テンプレートファイルをアップロー、ファイルを選択の順に選択し、クローンした GitHub リポジトリの `templates` フォルダから `cicd-role-template.yaml` ファイルを選択します。次へ をクリックします。 5. わかりやすいアプリケーションスタック名を指定します。 6. パラメータで以下の値を使用します。	AWS DevOps

タスク	説明	必要なスキル
	• アクセス許可の境界として使用するポリシーを選択します。この ARN は、IAM コンソールのアクセス権限境界ポリシーのポリシー詳細セクションから取得できます。 • 以前にメモした SNS 本稼働の承認トピックの ARN。 • 以前にメモした テスト承認トピックの ARN。 • テンプレートによって作成されたリソース 7. 次へ 、次へ、送信を選択します。 8. スタックが正常に作成された後、出力タブを選択し、作成された IAM ロールの ARN を書き留めます。この情報は次のステップで使用します。	

タスク	説明	必要なスキル
アプリケーションの CodeCommit リポジトリとコードパイプラインを作成します。	GitHub リポジトリに提供されている `cicd-pipeline-template.yaml` テンプレートを使用して、アプリケーションの CodeCommit リポジトリとコードパイプラインを作成できます。 1. AWS コンソールで、CI/CD アカウントにサインインします。 2. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation/) を開きます。 3. 新しいリソースを使用して、スタックを作成します (標準)。 4. テンプレートを指定では、テンプレートファイルをアップロー、ファイルを選択の順に選択し、クローンした GitHub リポジトリの `templates` フォルダから `cicd-pipeline-template.yaml` ファイルを選択します。次へ をクリックします。 5. わかりやすいアプリケーションスタック名を指定します。 6. パラメータで以下の値を使用します。	AWS DevOps

タスク	説明	必要なスキル
	• AppRepositoryName — アプリケーション用に作成される CodeCommit リポジトリの名前。 • AppRepositoryDescription — アプリケーション用に作成される CodeCommit リポジトリの簡単な説明。 • ApplicationName – アプリケーションの名前を指定します。この文字列は、CodeCommit リポジトリの名前と CI/CD パイプラインのプレフィックスとして使用されます。 • CloudWatchEventRoleARN — 前のタスクの CloudWatch イベントロールの ARN。 • CodeBuildProjectRoleARN — 前のタスクの CodeBuild プロジェクトロールの ARN。 • CodePipelineRoleARN — 前のタスクの CodePipeline ロールの ARN。 • DeploymentConfigBucket — デプロイ設定ファイルとスクリプト.zip ファイルが保存される Amazon Simple Simple Storage	

タスク	説明	必要なスキル
	Service (Amazon S3) バケット名。 • DeploymentConfigKey — パスと.zip ファイル名 (Amazon S3 キー)。 • PRApprovalSNSARN — プルリクエスト通知の SNS トピックの ARN。 • ProdApprovalSNSARN — プロダクション承認用の SNS トピックの ARN。 • TESTApprovalSNSARN — テスト承認用の SNS トピックの ARN。 • TemplateBucket — CI/CD パイプライン作成テンプレートが保存される CI/CD アカウントの S3 バケットの名前。 7. 次へ 、次へ、送信を選択します。 8. スタックが正常に完了する時、指定された名前とデフォルトのディレクトリ構造、デプロイ設定ファイル、スクリプト、およびリポジトリのコードパイプラインを持つ CodeCommit リポジトリが作成されます。	

スタックセットをデプロイします。

タスク	説明	必要なスキル

アプリケーションリポジトリをクローニングします。	以前に使用した CI/CD パイプラインテンプレートは、サンプルアプリケーションリポジトリとコードパイプラインを作成します。リポジトリを複製して検証するには：
	1. ソースアカウントにサインインします。
	2. 前のエピックで作成したアプリケーションリポジトリと CI/CD パイプラインを探します。
	3. リポジトリの URL をコピーし、git clone コマンドを使用してローカルマシン上のリポジトリを複製します。
	4. ディレクトリ構造とファイルが以下のとおりであることを確認します。
	`root`
	`|- deploy_configs`
	`|  |- deployment_config.json`
	`|- parameters`
	`|  |- template-parameter-dev.json`
	`|  |- template-parameter-test.json`
	`|  |- template-parameter-prod.json`
	`|- templates`
		アプリ開発者、データエンジニア

タスク	説明	必要なスキル

	<pre>| |- template.yml
	|- buildspec.yml</pre>
	deploy_configs フォルダにデプロイ設定ファイルが含まれ、templates と parameters フォルダに独自の CloudFormation テンプレートとパラメータファイルに置き換えるデフォルトファイルが含まれます。
	
	Important
	フォルダ構造をカスタマイズしないでください。
	
	5. フィーチャーブランチを作成します。	

タスク	説明	必要なスキル
アプリケーションアーティファクトの追加。	CloudFormation テンプレートを使用してアプリケーションリポジトリを更新します。 **Note** このソリューションは、単一の CloudFormation テンプレートのデプロイのみをサポートします。 1. アプリケーションコードの変更をデプロイするための CloudFormation テンプレートを作成し、`<application-name>.yaml` という名前を付けます。 2. アプリケーションリポジトリの `templates` フォルダーにある `template.yml` ファイルを、ステップ 1 で作成した CloudFormation テンプレートに置き換えます。 3. 環境 (開発、テスト、実稼働) ごとにパラメータファイルを準備します。 4. パラメータファイルには `<cloudformation-template-name>-parameter-<environment-`	アプリ開発者、データエンジニア

タスク	説明	必要なスキル
	name>.json 形式を使用して名前を付けます。 5. parameters フォルダーのデフォルトのパラメーターファイルを、ステップ 4 で作成したファイルに置き換えます。	

タスク	説明	必要なスキル

デプロイ設定ファイルを更新します。	ファイルを更新します。
	1. アプリケーションリポジトリで、deploy_configs フォルダに移動します。
	2. deployment_config.json ファイルを開きます。
	
	{
	"deployment_action": "<deploy/delete>",
	"stack_set_name": "<stack set name>",
	"stack_set_desciption": "<stack set description>",
	"deployment_targets": {
	"dev": {
	"org_units": ["list of OUs"],
	"regions": ["list of regions"],
	"filter_accounts": ["list of accounts"],
	"filter_type":
		アプリ開発者、データエンジニア

タスク	説明	必要なスキル

	"<DIFFERENCE/INTERSECTION/UNION>"
	},
	"test": {
	"org_units": ["list of OUs"],
	"regions": ["list of regions"],
	"filter_accounts": ["list of accounts"],
	"filter_type": "<DIFFERENCE/INTERSECTION/UNION>"
	},
	"prod": {
	"org_units": ["list of OUs"],
	"regions": ["list of regions"],
	"filter_accounts": ["list of accounts"],	

タスク	説明	必要なスキル

	<pre>"filter_type": "<DIFFERENCE/INTERSECTION/UNION>"
	}
	},
	"cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
	"auto_deployement": "<True/False>",
	"retain_stacks_on_account_removal": "<True/False>",
	"region_deployment_concurrency": "<SEQUENTIAL/PARALLEL>"
	}</pre>	

	3. デプロイアクション、スタックセット名、スタックセットの説明、およびデプロイターゲットの値を更新します。 たとえば、deployment_action を delete に設定すると、スタックセット全体と関連するスタックインスタンスを削除できます。新しいスタックセットを作成したり、既存のス	

タスク	説明	必要なスキル
	タックセットを更新したり、その他の OU やリージョンのスタックインスタンスを追加または削除したりする場合、deploy を使用します。詳細と例については、追加情報 の セクションを参照してください。 このパターンでは、デプロイ設定ファイルに指定したスタックセット名に環境名を追加して、環境ごとに個別のスタックセットを作成します。	

タスク	説明	必要なスキル
変更内容をコミットし、スタックセットをデプロイします。	アプリケーションテンプレートで指定した変更をコミットし、スタックセットを複数の環境に段階的にマージしてデプロイします。 1. すべてのファイルを保存し、変更をローカルアプリケーションリポジトリの機能ブランチにコミットします。 2. フィーチャーブランチをリモートリポジトリにプッシュします。 3. プルリクエストを作成して、変更をメインブランチにマージします。 プルリクエストが承認され、変更がメインブランチにマージされる場合、CI/CD パイプラインが開始されます。 4. 開発デプロイステージが正常に完了した場合、CloudFormation コンソール、StackSets、Service-Managedタブを確認します。 dev サフィックスが付いた新しいスタックセットが表示されます。	アプリ開発者、データエンジニア

タスク	説明	必要なスキル
	5. 開発デプロイステージの CodeBuild ログに問題がないかを確認します。 6. スタックセットをテスト環境と本番環境にデプロイするには、承認者に各ステージのデプロイを承認して、ステップ 5 と 6 を繰り返します。テスト環境と本番環境のスタックセットには、`test` と `prod` サフィックスとが付いています。	

トラブルシューティング

問題	ソリューション
例外のせいでデプロイが失敗しました。 テンプレートパラメータファイルの名前を<アプリケーション名>-パラメータ-<evn>.json に変更します。デフォルトの名前は使用できません	CloudFormation テンプレートのパラメータファイルは、指定された命名規則に従う必要があります。パラメータファイル名を更新し、再試行します。
例外のせいでデプロイが失敗しました。 CloudFormation テンプレートの名前を <アプリケーション名>.yml 、デフォルトのテンプレート.yml 、または template.yaml は使用できませんに変更します	CloudFormation テンプレートのパラメータファイルは、指定された命名規則に従う必要があります。ファイル名を更新して、再試行します。
例外のせいでデプロイが失敗しました。	CloudFormation テンプレートのファイル命名規則と、指定された環境のパラメータファイルを確認します。

問題	ソリューション
{環境名} 環境の有効な CloudFormation テンプレートとそのパラメータファイルが見つかりません	
例外のせいでデプロイが失敗しました。 デプロ設定ファイルに無効なデプロイアクションが提供されています。有効なオプションは「デプロイ」と「削除」です。	デプロイ設定ファイルの `deployment_action` パラメータに無効な値を指定しました。このパラメータには、`deploy` と `delete` の 2 つの有効な値があります。`deploy` を使用して、スタックセットとそれに関連するスタックインスタンスを作成、更新します。スタックセット全体と関連するスタックインスタンスを削除する場合のみ、`delete` を使用します。

関連リソース

- GitHub 自動コードパイプラインスタックセットデプロイ リポジトリ
- 組織内のすべての特徴量の有効化 (AWS Organizations ドキュメント)
- 委任管理者を登録 (AWS CloudFormation ドキュメント)
- サービスマネージド型スタックセットのアカウントレベルのターゲット (AWS CloudFormation ドキュメント)

追加情報

フローチャート

次のフローチャートは、スタックセットのデプロイを自動化するためにカスタムスクリプトによって実装される API 呼び出しのフロー制御と階層を示しています。

サンプルのデプロイ設定ファイル

新しいスタックセットの作成t

次のデプロイ設定ファイルは、3 つの OU の AWS リージョン us-east-1 の sample-stack-set と呼ばれる新しいスタックセットを作成します。

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                             "dev": {
                                        "org_units": ["dev-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                             "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                             "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                           },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

既存のスタックセットを別の OU にデプロイ

前の例で示した設定をデプロイし、dev-org-unit-2 開発環境で呼び出される別の OU にスタックセットをデプロイする場合、デプロイ設定ファイルは次のようになります。

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
```

```
                            "dev": {
                                    "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                           },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

既存のスタックセットを別の AWS リージョンにデプロイ

前の例で示した設定をデプロイし、2 つの OU(dev-org-unit-1 と dev-org-unit-2) の開発環境内の別の AWS リージョン (us-east-2) にスタックセットをデプロイする場合、デプロイ設定ファイルは次のようになります。

> **Note**
>
> CloudFormation テンプレートのリソースは有効で、リージョン固有である必要があります。

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
```

```
                            "dev": {
                                    "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                    "regions": ["us-east-1", "us-east-2"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                           },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

OU または AWS リージョンからのスタックインスタンスを削除

前の例で示したデプロイ設定がデプロイされたとみなされます。次の設定ファイルは OU dev-org-unit-2 の両方のリージョンからスタックインスタンスを削除します。

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                    "org_units": ["dev-org-unit-1"],
                                    "regions": ["us-east-1", "us-east-2"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "test": {
```

```
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                        },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

次の設定ファイルは、開発環境内の両方の OU の AWS リージョン us-east-1 からスタックインスタンスを削除します。

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                    "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                    "regions": ["us-east-2"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
```

```
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                            },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
 }
```

スタックセット全体を削除します。

次のデプロイ設定ファイルは、スタックセット全体と関連するスタックインスタンスを削除します。

```
{
     "deployment_action": “delete”,
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                    "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                    "regions": ["us-east-2"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                            "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                            },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
```

```
}
```

デプロイメントからアカウントを除外します。

次のデプロイ設定ファイルでは、OU dev-org-unit-1 の一部であるアカウント 111122223333 がデプロイから除外されます。

```
{
    "deployment_action": "deploy",
    "stack_set_name": "sample-stack-set",
    "stack_set_desciption": "this is a sample stack set",
   "deployment_targets": {
                            "dev": {
                                        "org_units": ["dev-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": ["111122223333"],
                                        "filter_type": "DIFFERENCE"
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                          },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

OU のアカウントのサブセットにアプリケーションをデプロイ

次のデプロイ設定ファイルでは、OU dev-org-unit-1 の 3 つのアカウント (111122223333、444455556666、および 777788889999)にのみアプリケーションをデプロイします。

```
{
    "deployment_action": "deploy",
    "stack_set_name": "sample-stack-set",
    "stack_set_desciption": "this is a sample stack set",
   "deployment_targets": {
                            "dev": {
                                        "org_units": ["dev-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": ["111122223333",
"444455556666", "777788889999"],
                                        "filter_type": "INTERSECTION"
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                          },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

Cloud Custodian と AWS CDK を使用して、Systems Manager の AWS マネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする

作成者: Ali Asfour (AWS)、Aaron Lennon (AWS)

概要

運用タスクを自動化し、より多くの可視性と制御を提供する AWS Systems Manager に、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを統合することができます。Systems Manager と統合するには、EC2 インスタンスに AWS Systems Manager Agent(SSM Agent) をインストールし、`AmazonSSMManagedInstanceCore` AWS Identity and Access Management (IAM) ポリシーをプロファイルに追加する必要があります。

ただし、すべての EC2 インスタンスプロファイルに `AmazonSSMManagedInstanceCore` ポリシーを確実にアタッチする場合、インスタンスプロファイルを持たない新規 EC2 インスタンスや、インスタンスプロファイルを持つが `AmazonSSMManagedInstanceCore` ポリシーを持たない EC2 インスタンスを更新する際に問題が発生する可能性があります。また、このポリシーを複数の Amazon Web Services (AWS) アカウントや AWS リージョンに追加することが難しい場合もあります。

このパターンは、AWS アカウントに次の 3 つの Cloud Custodian ポリシーをデプロイすることで、これらの課題を解決するのに役立ちます。

- 最初の Cloud Custodian ポリシーは、インスタンスプロファイルを持つが、`AmazonSSMManagedInstanceCore` ポリシーを持たない既存の EC2 インスタンスをチェックします。その後、`AmazonSSMManagedInstanceCore` ポリシーがアタッチされます。
- 2 つ目の Cloud Custodian ポリシーは、インスタンスプロファイルのない既存の EC2 インスタンスをチェックし、`AmazonSSMManagedInstanceCore` ポリシーがアタッチされたデフォルトのインスタンスプロファイルを追加します。
- 3 つ目の Cloud Custodian ポリシーでは、EC2 インスタンスとインスタンスプロファイルの作成を監視するための AWS Lambda 関数をアカウント内に作成します。これにより、EC2 インスタンスの作成時に自動的に `AmazonSSMManagedInstanceCore` ポリシーがアタッチされます。

このパターンは、AWS DevOps ツールを使用することで、個別のコンピューティング環境をプロビジョニングすることなく、マルチアカウント環境で Cloud Custodian ポリシーの継続的かつ大規模なデプロイを実現します。

前提条件と制限

前提条件

- 2 つ以上の AWS アカウントがアクティブである。一方のアカウントはセキュリティアカウントで、他方はメンバーアカウントである。
- セキュリティアカウントで AWS リソースをプロビジョニングする権限がある。このパターンでは、管理者権限を使用しますが、組織の要件とポリシーに従って権限を付与する必要があります。
- セキュリティアカウントから IAM ロールをメンバーアカウントに引き継ぎ、必要な IAM ロールを作成できます。詳細については、IAM ドキュメントの「IAM ロールを使用して AWS アカウント間でアクセスを委任する」を参照してください。
-

> **Important**
>
> AWS コマンドラインインターフェイス (AWS CLI) をインストールして設定済み。テスト目的で、`aws configure` コマンドを使用するか、環境変数を設定することで、AWS CLI を設定できます。: これは本番環境には推奨されず、このアカウントには最小特権アクセスのみを付与することをお勧めします。詳細については、IAM ドキュメントの「最小特権を付与する」を参照してください。

- `devops-cdk-cloudcustodian.zip` ファイル (添付) は、ローカルコンピュータにダウンロードされます。
- Python に精通していること。
- 必要なツール (Node.js、AWS Cloud Development Kit (AWS CDK)、および Git) をインストールして設定済み。`devops-cdk-cloudcustodian.zip` ファイル内の `install-prerequisites.sh` ファイルを使用して、これらのツールをインストールできます。このファイルを root 権限で実行していることを確認します。

機能制限

- このパターンは実稼働環境でも使用できますが、すべての IAM ロールとポリシーが組織の要件とポリシーを満たしていることを確認してください。

パッケージバージョン

- Cloud Custodian バージョン 0.9 以降
- TypeScript バージョン 3.9.7 以降

- Node.js バージョン 14.15.4 以降
- npm バージョン 7.6.1 以降
- AWS CDK バージョン 1.96.0 またはそれ以降

アーキテクチャ

この図表は、次のワークフローを示しています：

1. Cloud Custodian ポリシーが、セキュリティ アカウントに AWS CodeCommit リポジトリにプッシュされます。Amazon CloudWatch Events ルールは、AWS CodePipeline パイプラインを自動的に起動します。
2. パイプラインは CodeCommit から最新のコードを取得し、AWS CodeBuild で処理される CI/CD (継続統合および継続配信) パイプラインの継続的な統合部分に送信します。
3. CodeBuild は、Cloud Custodian ポリシーのポリシー構文検証を含む完全な DevSecOps アクションを実行し、`--dryrun` モードでこれらのポリシーを実行して、どのリソースが認識されているかをチェックします。
4. エラーがなければ、次のタスクで変更を確認し、メンバーアカウントへのデプロイを承認するよう管理者にアラートが送信されます。

テクノロジースタック

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- IAM
- Cloud Custodian

自動化とスケール

AWS CDK Pipelines モジュールは、AWS CloudFormation スタックを使用した AWS リソースのデプロイに加え、CodePipeline を使用する CI/CD コンジットをプロビジョニングして、CodeBuild を介したソースコードのビルドとテストをオーケストレーションします。このパターンは、組織内のす

べてのメンバーアカウントとリージョンで使用できます。Roles creation スタックを拡張して、メンバーアカウントに他の IAM ロールをデプロイすることもできます。

ツール

- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、AWS CloudFormation を通じてプロビジョニングするための、ソフトウェア開発フレームワーク
- AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- AWS CodeBuild はクラウドで動作する、完全マネージド型のビルドサービスです。
- AWS CodeCommit は、プライベートな保存と管理を有効にするバージョン管理サービスです。
- AWS CodePipelineは、ソフトウェアをリリースするために必要な手順のモデル化、視覚化、および自動化に使用できる継続的な配信サービスです。
- AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスをセキュアに制御するためのウェブサービスです。
- Cloud Custodian は、多くの組織がパブリック クラウドアカウントの管理に使用しているツールとスクリプトを 1 つのオープンソースツールに統合します。
- Node.js は Google Chrome の V8 JavaScript エンジンをベースに構築された JavaScript ランタイムです。

コード

このパターンで使用されるモジュール、アカウント関数、ファイル、およびデプロイコマンドの詳細なリストについては、devops-cdk-cloudcustodian.zip ファイル (添付) README 内のファイルを参照してください。

エピック

AWS CDK でパイプラインをセットアップする

タスク	説明	必要なスキル
CodeCommit リポジトリを設定します。	1. devops-cdk-cloudcustodian.zip ファイル (添付) を、ローカルコンピ	開発者

タスク	説明	必要なスキル
	ユータの作業ディレクトリで解凍します。 2. AWS マネジメントコンソールにセキュアなアカウントでログインし、CodeCommit コンソールを開いて、新しい `devops-cdk-cloudcustodian` リポジトリを作成します。 3. プロジェクトディレクトリに移動し、CodeCommit リポジトリをオリジンに設定し、変更をコミットしてから、以下のコマンドを実行して、それらをオリジンブランチにプッシュします。 • `cd devops-cdk-cloudcustodian` • `git init --initial-branch=main` • `git add . git commit -m 'initial commit'` • `git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/devops-cdk-cloudcustodian` • `git push origin main`	

タスク	説明	必要なスキル
	詳細については、AWS CodeCommit ドキュメントの「CodeCommit リポジトリの作成」を参照してください。	
必要なツールをインストールします。	`install-prerequisites.sh` ファイルを使用して Amazon Linux に必要なすべてのツールをインストールします。AWS CLI は事前にインストールされているため、含まれていません。 詳細については、AWS CDK ドキュメントの「AWS CDK の使用開始」にある「前提条件」セクションを参照してください。	開発者

タスク	説明	必要なスキル
必要な AWS CDK パッケージをインストールします。	1. AWS CLI で、`$ python3 -m venv .env` コマンドを実行して仮想環境を設定します。 2. AWS CLIで、`$ source .env/bin/activate` コマンドを実行して仮想環境をアクティブにします。 3. 仮想環境がアクティブになったら、`$ pip install -r requirements.txt` コマンドを実行して必要な依存関係をインストールします。 4. その他の依存関係 (他の AWS CDK ライブラリなど) を追加するには、`requirements.txt` ファイルに追加し、`pip install -r requirements.txt` コマンドを実行します。 以下のパッケージは AWS CDK に必要であり、`requirements.txt` ファイルに含まれています。 • `aws-cdk.aws-cloudwatch` • `aws-cdk.aws-codebuild`	開発者

タスク	説明	必要なスキル
	• aws-cdk.aws-codecommit • aws-cdk.aws-codedeploy • aws-cdk.aws-codepipeline • aws-cdk.aws-codepipeline-actions • aws-cdk.aws-events • aws-cdk.aws-events-targets • aws-cdk.aws-iam • aws-cdk.aws-logs • aws-cdk.aws-s3 • aws-cdk.aws-sns • aws-cdk.aws-sns-subscriptions • aws-cdk.aws-sqs • aws-cdk.core	

環境の設定

タスク	説明	必要なスキル
必要な変数を更新してください。	CodeCommit vars.py リポジトリのルートフォルダにあるファイルを開き、次の変数を更新します。 • パイプラインをデプロイする AWS リージョンで var_deploy_region =	開発者

タスク	説明	必要なスキル
	‘us-east-1’ を更新します。 • CodeCommit リポジトリの名前で var_codecommit_repo_name = “cdk-cloudcustodian” を置き換えます。 • CodeCommit ブランチの名前で var_codecommit_branch_name = “main” を更新します。 • 変更を承認する管理者のメールアドレスで var_adminEmail=notifyadmin@email.com’ を更新します。 • 変更があった場合に Cloud Custodian 通知を送信する Slack webhook を使用して var_slackWebHookUrl = https://hooks.slack.com/services/T000000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX を更新します。 • 組織 ID で var_orgId = ‘o-yyyyyyyyyy’ を更新します。 • パイプラインがデプロイされているアカウントの AWS アカウント ID で security_account =	

タスク	説明	必要なスキル
	‘123456789011’ を更新します。 • AWS CDK スタックをブートストラップし、必要な IAMロールをデプロイするメンバーアカウントを使用して member_accounts = [‘111111111111’,’111111111112’,’111111111113’] を更新します。 • パイプラインで AWS CDK をメンバーアカウントに自動的にブートストラップする場合は、cdk_bootstrap_member_accounts = True を True に設定します。 True に設定した場合は、メンバーアカウント内の既存の IAMロールの名前も必要になります。このロールは、セキュリティアカウントから引き受けることができます。この IAM ロールには、AWS CDK をブートストラップするためのアクセス権限も必要です。 • **Note** メンバーアカウント内の既存の IAM ロールの名前を使用して cdk_boots	

タスク	説明	必要なスキル
	trap_role = 'AWSControlTowerExecution' を更新します。このロールは、セキュリティアカウントから引き受けることができます。このロールには、AWS CDK をブートストラップするためのアクセス権限も必要です。: これは、 cdk_bootstrap_member_accounts がに設定されている場合にのみ適用されますTrue。	

タスク	説明	必要なスキル

account.yml ファイルをメンバーアカウント情報で更新します。	c7n-org Cloud Custodian ツールを複数のアカウントに対して実行するには、accounts.yml 設定ファイルをリポジトリのルートに配置する必要があります。以下は、AWS の Cloud Custodian 設定のサンプルファイルです。
	`accounts:`
	`- account_id: '123123123123'`
	`  name: account-1`
	`  regions:`
	`  - us-east-1`
	`  - us-west-2`
	`  role: arn:aws:iam::123123123123:role/CloudCustodian`
	`  vars:`
	`  charge_code: xyz`
	`  tags:`
	`  - type:prod`
	`  - division:some division`
	`  - partition:us`
	`  - scope:pci`
		開発者

AWS アカウントをブートストラップする

タスク	説明	必要なスキル
セキュリティアカウントをブートストラップします。	以下のコマンドを実行して、cloudcustodian_stack アプリケーションで	開発者

タスク	説明	必要なスキル
	deploy_account をブートストラップします。 `cdk bootstrap -a 'python3 cloudcustodian/cloudcustodian_stack.py'`	
オプション 1 - メンバーアカウントを自動的にブートストラップします。	vars.py ファイルで cdk_bootstrap_member_accounts 変数が True に設定されている場合、member_accounts 変数で指定されたアカウントは パイプラインによって自動的にブートストラップされます。 必要に応じて、IAM ロールを使用して *cdk_bootstrap_role* を更新できます。このロールは、セキュリティアカウントから引き受けることができ、AWS CDK をブートストラップするために必要な権限を持っています。 member_accounts 変数に追加された新規アカウントは、パイプラインによって自動的にブートストラップされ、必要なロールをデプロイできるようになります。	開発者

タスク	説明	必要なスキル

オプション 2 - メンバーアカウントを手動でブートストラップします。	この方法はお勧めしませんが、cdk_bootstrap_member_accounts の値を False に設定し、次のコマンドを使用して手動で実行できます：
	
	`$ cdk bootstrap -a 'python3 cloudcustodian/member_account_roles_stack.py' \`
	`--trust {security_account_id} \`
	`--context assume-role-credentials:writeIamRoleName={role_name} \`
	`--context assume-role-credentials:readIamRoleName={role_name} \`
	`--mode=ForWriting \`
	`--context bootstrap=true \`
	`--cloudformation-execution-policies arn:aws:iam::aws:policy/AdministratorAccess`
		開発者

タスク	説明	必要なスキル
	Important セキュリティアカウントから引き受けることができ、AWS CDK をブートストラップするために必要なアクセス許可を持つ IAM ロールの名前で `{security_account_id}` と `{role_name}` の値を更新してください。 AWS CloudFormation など、他の方法を使用してメンバーアカウントをブートストラップすることもできます。詳細については、AWS CDK ドキュメントの「ブートストラップ」を参照してください。	

AWS CDK スタックをデプロイする

タスク	説明	必要なスキル
メンバーアカウントで IAM ロールを作成します。	次のコマンドを実行して、`member_account_roles_stack` スタックをデプロイし、メンバーアカウントに IAMロールを作成します：	開発者

タスク	説明	必要なスキル
	`cdk deploy --all -a 'python3 cloudcustodian/member_account_roles_stack.py' --require-approval never`	
Cloud Custodian パイプラインスタックをデプロイします。	次のコマンドを実行して、セキュリティアカウントにデプロイされる Cloud Custodian `cloudcustodian_stack.py` パイプラインを作成します。 `cdk deploy -a 'python3 cloudcustodian/cloudcustodian_stack.py'`	開発者

関連リソース

- AWS CDK の使用開始

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CDK を使用してマイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する

作成者: Varsha Raju (AWS)

概要

このパターンは、Amazon Elastic Container Service (Amazon ECS) でマイクロサービスを構築およびデプロイするための、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインと基盤となるインフラストラクチャを自動的に作成する方法を示しています。概念実証の CI/CD パイプラインを設定して、組織に CI/CD、マイクロサービス、DevOps の利点を説明する場合は、このアプローチを使用できます。また、このアプローチを使用して最初の CI/CD パイプラインを作成し、組織の要件に応じてカスタマイズまたは変更もできます。

このパターンのアプローチでは、本番環境と非本番環境を作成し、それぞれに仮想プライベートクラウド (VPC) と 2 つのアベイラビリティゾーンで実行するように設定された Amazon ECS クラスターがあります。これらの環境はすべてのマイクロサービスとユーザーで共有され、各マイクロサービスに CI/CD パイプラインを作成します。これらの CI/CD パイプラインは、AWS CodeCommit のソースリポジトリから変更を取得し、変更を自動的にビルドして、本番環境と非本番環境にデプロイします。パイプラインがすべてのステージを正常に完了すると、URL を使用して本番環境と非本番環境のマイクロサービスにアクセスできます。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS)アカウント。
- `starter-code.zip` ファイル (添付) を含む既存の Amazon Simple Storage Service (Amazon S3) バケット。
- AWS Cloud Development Kit (AWS CDK) はお使いのアカウントにインストールおよび設定済みです。詳細については、AWS CDK ドキュメントの「Getting started with the AWS CDK」を参照してください。
- Python 3 と `pip` をインストールおよび設定済みです。詳細については、Python のドキュメントを参照してください。
- AWS CDK、AWS CodePipeline、AWS CodeBuild、CodeCommit、Amazon Elastic Container Registry (Amazon ECR)、Amazon ECS、AWS Fargate に精通していること。
- Docker に精通していること。

- CI/CD と DevOps について理解していること。

制限

- AWS アカウントの全般的な制限が適用されます。この詳細については、AWS 全般のリファレンスドキュメントの「AWS Service Quotas」を参照してください。

製品バージョン

- このコードは、 Node.js バージョン 16.13.0 および AWS CDK バージョン 1.132.0 を使用してテストされました。

アーキテクチャ

この図表は、次のワークフローを示しています：

1. アプリ開発者は CodeCommit リポジトリにコードをコミットします。
2. パイプラインが開始されます。
3. CodeBuild は、Docker イメージをビルドして Amazon ECR リポジトリにプッシュします。
4. CodePipeline は、非本番環境の Amazon ECS クラスター内の既存の Fargate サービスに新しいイメージをデプロイします。
5. Amazon ECS は Amazon ECR リポジトリから非本番環境の Fargate サービスにイメージを引き出します。
6. テストは非本番環境の URL を使用して実行されます。
7. リリースマネージャーは本番環境へのデプロイを承認します。
8. CodePipeline は、新しいイメージを本番環境の Amazon ECS クラスター内の既存の Fargate サービスにデプロイします。
9. Amazon ECS は Amazon ECR リポジトリから本番環境の Fargate サービスにイメージを引き出します。
10. 本番環境ユーザーは本番環境の URL を使用して機能にアクセスします。

テクノロジースタック

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- Amazon ECR
- Amazon ECS
- Amazon VPC

自動化とスケール

このパターンのアプローチを使用して、共有の AWS CloudFormation スタックにデプロイされたマイクロサービスのパイプラインを作成できます。自動化により、各 VPC に複数の Amazon ECS クラスターを作成できるほか、共有 Amazon ECS クラスターにデプロイされたマイクロサービスのパイプラインも作成できます。ただし、そのためには、新しいリソース情報をパイプラインスタックへの入力として提供する必要があります。

ツール

- AWS CDK— AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、AWS CloudFormation でプロビジョニングするソフトウェア開発フレームワークです。
- AWS CodeBuild - AWS CodeBuild はクラウドの完全マネージド型のビルドサービスです。CodeBuild はソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイできるアーティファクトを作成します。
- AWS CodeCommit – AWS CodeCommit は、 AWS クラウドでの Git リポジトリのプライベートな保存と管理を可能にするバージョン制御サービスです。CodeCommit により、お客様が独自のソース制御システムを管理、またはインフラストラクチャのスケーリングに関して心配する必要はなくなります。
- AWS CodePipeline – AWS CodePipeline は、ソフトウェアのリリースに必要な手順のモデル化、可視化、および自動化に使用できる継続的な配信サービスです。ソフトウェアリリースプロセスのさまざまなステージを素早くモデル化して設定できます。CodePipeline はソフトウェア変更の継続的リリースに必要なステップを自動化します。
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理に使用される、高度にスケーラブルで高速のコンテナ管理サービスです。タス

クとサービスは、 AWS Fargate で管理されているサーバーレスインフラストラクチャで実行できます。または、インフラストラクチャをより詳細に制御するために、管理する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのクラスターでタスクとサービスを実行できます。

- Docker — Dockerを使用すると、開発者は任意のアプリケーションを軽量、ポータブル、自給自足のコンテナとして梱包、出荷、実行する上で役立ちます。

コード

このパターンのコードは、`cicdstarter.zip` および `starter-code.zip` ファイル (添付) にあります。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
AWS CDK の作業ディレクトリを設定します。	1. ローカルマシンで `cicdproject` という名前のディレクトリを作成します。 2. `cicdstarter.zip` ファイル (添付)を `cicdproject` ディレクトリにダウンロードし、解凍します。これで、`cicdstarter` という名前のフォルダが作成されます。 3. `cd <user-home>/cicdproject/cicdstarter` コマンドを実行します。 4. `python3 -m venv .venv` コマンドを実行して Python 仮想環境を設定します。	AWS DevOps、クラウドインフラストラクチャ

タスク	説明	必要なスキル
	5. `source ./.venv/bin/activate` コマンドを実行します。 6. `aws configure` コマンドを実行、または以下の環境変数を使用して AWS 環境を設定します。 • `AWS_ACCESS_KEY_ID` • `AWS_SECRET_ACCESS_KEY` • `AWS_DEFAULT_REGION`	

共有インフラストラクチャの作成

タスク	説明	必要なスキル
共有インフラストラクチャを作成します。	1. 作業ディレクトリで、`cd cicdvpcecs` コマンドを実行します。 2. `pip3 install -r requirements.txt` コマンドを実行して、必要な Python 依存関係をすべてインストールします 3. `cdk bootstrap` command を実行して、AWS CDK の AWS 環境を設定します。 4. `cdk synth --context aws_account=<aws_account_ID> --context aws_regio`	AWS DevOps、クラウドインフラストラクチャ

タスク	説明	必要なスキル
	n=<aws-region> コマンドを実行します。 5. cdk deploy --context aws_account=<aws_account_ID> --context aws_region=<aws-region> コマンドを実行します。 6. AWS CloudFormation スタックは、以下のインフラストラクチャを作成します。 • cicd-vpc-ecs/cicd-vpc-nonprod という名前の非本番環境 VPC • cicd-vpc-ecs/cicd-vpc-prod という名前の本番環境 VPC • cicd-ecs-nonprod という名前の非本番環境用 Amazon ECS クラスター • cicd-ecs-prod という名前の本番環境用 Amazon ECS クラスター	

タスク	説明	必要なスキル
AWS CloudFormation スタックを監視します。	1. AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開き、リストから `cicd-vpc-ecs` スタックを選択します。 2. スタックの詳細ペインで、[Events] (イベント) タブを選択し、スタックの作成の進行状況をモニタリングします。	AWS DevOps、クラウドインフラストラクチャ

タスク	説明	必要なスキル
AWS CloudFormation スタックをテストします。	1. `cicd-vpc-ecs` AWS CloudFormation スタックが作成されたら、`cicd-vpc-ecs/cicd-vpc-nonprod` および `cicd-vpc-ecs/cicd-vpc-prod` VPC が作成されていることを確認します。 2. `cicd-ecs-nonprod` と `cicd-ecs-prod` Amazon ECS クラスターが作成されていることを確認します。 **Important** 2 つの VPCs IDs と、両方の VPC のデフォルトのセキュリティグループのセキュリティグループ IDs を必ず記録してください。VPCs	AWS DevOps、クラウドインフラストラクチャ

マイクロサービスの CI/CD パイプラインの作成

タスク	説明	必要なスキル
マイクロサービスのインフラストラクチャーを作成します。	1. マイクロサービスに名前を付けます。たとえば、このパターンはマイクロサービスの名前として `myservice1` を使用します。	AWS DevOps、クラウドインフラストラクチャ

タスク	説明	必要なスキル
	2. 作業ディレクトリで `cd <working-directory>/cdkpipeline` コマンドを実行します。 3. `pip3 install -r requirements.txt` コマンドを実行します。 4. このパターンの追加情報セクションにある `cdk synth` コマンドをすべて実行します。 5. このパターンの追加情報セクションにある `cdk deploy` コマンドをすべて実行します。 **Note** ディレクトリの `cdk.json` ファイルを使用して、両方のコマンドの値を指定することもできます。	
AWS CloudFormation スタックを監視します。	AWS CloudFormation コンソールを開き、`myservice1-cicd-stack` スタックの進捗状況をモニタリングします。最終的に、ステータスは `CREATE_COMPLETE` に変わります。	AWS DevOps、クラウドインフラストラクチャ

タスク	説明	必要なスキル
AWS CloudFormation スタックをテストします。	1. AWS CodeCommit コンソールで、`myservice1` という名前のリポジトリが存在し、スターターコードが含まれていることを確認します。 2. AWS CodeBuild コンソールで、`myservice1` という名前のビルドプロジェクトが存在することを確認します。 3. Amazon ECR コンソールで、`myservice1` という名前のAmazon ECR リポジトリが存在することを確認します。 4. Amazon ECS コンソールで、`myservice1` という名前の Fargate サービスが非本番環境と本番環境の Amazon ECS クラスターの両方に存在することを確認します。 5. Amazon Elastic Compute Cloud (Amazon EC2) コンソールで、非本番環境と本番環境の Application Load Balancer が作成されていることを確認します。ALB の DNS 名を記録します。 6. AWS CodePipeline コンソールで、`myservice1` という名前のパイプライン	

タスク	説明	必要なスキル
	が存在することを確認します。Source、Build、Deplo NonProd および Deploy-Prod ステージが必要です。パイプラインにも in progress ステータスが必要です。 7. すべてのステージが完了するまでパイプラインを監視します。 8. 本番環境用に手動で承認します。 9. ブラウザウィンドウに、ALB の DNS 名を入力します。 10.アプリケーションは非本番環境 URL と本番環境 URL に Hello World が表示されるはずです。	

タスク	説明	必要なスキル
パイプラインを使用します。	1. 前に作成した CodeCommit リポジトリを開き、`index.js` ファイルを開きます。 2. `Hello World` を `Hello CI/CD` に置き換えます。 3. 変更を保存してメインブランチにコミットします。 4. パイプラインが開始され、変更が`Build`、`Deploy-NonProd` および `Deploy-Prod` ステージを通過することを確認します。 5. 本番環境用に手動で承認します。 6. これで、本番環境 URL と非本番環境 URL の両方に `Hello CICD` が表示されるはずです。	AWS DevOps、クラウドインフラストラクチャ
各マイクロサービスにこのエピックを繰り返します。	このエピックのタスクを繰り返して、各マイクロサービスの CI/CD パイプラインを作成します。	AWS DevOps、クラウドインフラストラクチャ

関連リソース

- Python と AWS CDK の併用
- AWS CDK Python リファレンス
- AWS CDK を使用して AWS Fargate サービスを作成する

追加情報

cdk synthコマンド

```
cdk synth --context aws_account=<aws_account_number> --context
 aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production
 VPC> --context vpc_prod_id=<id_of_production_VPC> --context
 ecssg_nonprod_id=< default_security_group_id_of_non-production_VPC>
 --context ecssg_prod_id=<default_security_group_id_of_production_VPC>
 --context code_commit_s3_bucket_for_code=<S3 bucket name> --context
 code_commit_s3_object_key_for_code=<Object_key_of_starter_code> --context
 microservice_name=<name_of_microservice>
```

cdk deploy コマンド

```
cdk deploy --context aws_account=<aws_account_number> --context
 aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production_VPC>
 --context vpc_prod_id=<id_of_production_VPC> --context ecssg_nonprod_id=<
 default_security_group_id_of_non-production_VPC> --context
 ecssg_prod_id=<default_security_group_id_of_production_VPC> --
context code_commit_s3_bucket_for_code=<S3 bucket name> --context
 code_commit_s3_object_key_for_code=<Object_key_of_starter_code> --context
 microservice_name=<name_of_microservice>
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

DevOps プラクティスと AWS Cloud9 を使用して、マイクロサービスで疎結合アーキテクチャを構築する

作成者: Alexandre Nardi (AWS)

概要

注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、Amazon Web Services (AWS) で DevOps プラクティスをテストし始めたデベロッパーや開発リーダー向けに、サーバーレスアーキテクチャで一般的なウェブアプリケーションを開発する方法を示しています。書籍を閲覧したり購入したりするためのストアフロントとバックエンドを作成するサンプルアプリケーションを構築し、独自に開発できるマイクロサービスを提供します。このパターンでは、開発環境として AWS Cloud9、データストアとして Amazon DynamoDB データベースを使用し、継続的インテグレーションと継続的デプロイ (CI/CD) 機能には AWS CodePipeline や AWS CodeBuild などの AWS サービスを使用しています。

このパターンは、以下の開発アクティビティをガイドします。

- 標準の AWS Cloud9 開発環境を作成する
- AWS CloudFormation テンプレートを使用して書籍用のウェブアプリケーションとマイクロサービスを作成
- AWS Cloud9 を使用してフロントエンドの変更、変更のコミット、変更のテストを行う
- マイクロサービスへの CI/CD パイプラインの作成とテスト
- ユニットテストの自動化

このパターンのコードは、GitHub 内の「AWS DevOps End-to-End Workshop」リポジトリで利用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント

- AWS DevOps End-to-End Workshop からコンピュータにダウンロードされたファイル

> **Important**
>
> このデモアプリケーションを AWS アカウントで構築すると、AWS リソースが作成および消費されます。アプリケーションの作成と実行に使用した AWS サービスまたはリソースのコストは、お客様が負担します。作業が終わったら、継続的な請求を避けるため、必ずすべてのリソースを削除してください。クリーンアップの手順については、「エピック」セクションを参照してください。

制限

このチュートリアルは、デモンストレーションと開発のみを目的としています。本稼働環境で使用するには、AWS Identity and Access Management (IAM) ドキュメントの「IAM でのセキュリティのベストプラクティス」を参照し、IAM ロール、Amazon DynamoDB、および使用するその他のサービスに必要な変更を加えます。ウェブアプリケーションは AWS Bookstore Demo App から派生したものです。その他の考慮事項については、README ファイルの「Known limitations」セクションを参照してください。

アーキテクチャ

Bookstore アプリケーションのアーキテクチャは、AWS Bookstore Demo App の README ファイルの「Architecture」セクションで説明されています。

デプロイの観点から見ると、Bookstore デモアプリは 1 つの CloudFormation テンプレートを使用して、すべてのサービスとオブジェクトを 1 つのスタックにデプロイします。このパターンでは、特定のデベロッパーまたはチームが特定の製品 (Books) でどのように作業し、アプリケーションの他の部分とは独立して更新できるかを示すために、いくつかの変更を加えています。このため、このパターンのコードでは、Books マイクロサービスの AWS Lambda 関数と関連オブジェクトを 2 つ目の CloudFormation テンプレートに分離して、Books スタックを作成します。これにより、CI/CD の手法を使用してマイクロサービスが更新されていることを確認できます。以下の図では、破線の枠線が Books マイクロサービスを表しています。

ツール

ツール

- JavaScript テスト用の Jest フレームワーク
- Python 3.9

コード

このパターンのソースコードとテンプレートは、GitHub の AWS DevOps End-to-End Workshop リポジトリにあります。「エピック」セクションの手順を実行する前に、リポジトリからすべてのファイルをコンピュータにダウンロードしてください。

> **Note**
>
> エピックセクションでは、プロセスに関する一般的な情報を提供するために、このウォークスルーの大まかな手順について説明します。各ステップを完了するには、AWS DevOps End-to-End Workshop リポジトリの README ファイルを参照して詳細な手順を確認してください。

AWS DevOps End-to-End Workshop リポジトリは、AWS Bookstore Demo App リポジトリを拡張し、AWS Cloud9 起動コードの修正バージョンを使用して AWS Cloud9 IDE を作成します。

ベストプラクティス

Bookstore アプリケーションの使い方は簡単です。推奨ベストプラクティスを以下に示します。

- アプリケーションをインストールするときは、任意のプロジェクト名を使用することも、便利なデフォルト名 (`demobookstore`) を使用することもできます。
- アプリケーションを稼働させた後、もう 1 日テストを続けたい場合は Amazon Neptune データベースをシャットダウンすることをお勧めします。データベースインスタンスでは追加料金が発生する可能性があるためです。ただし、データベースは 7 日後に自動的に起動されることに注意してください。
- コードの詳細については、AWS Bookstore Demo App リポジトリのドキュメントを参照してください。各マイクロサービスとテーブルについて説明しています。
- その他のベストプラクティスについては、AWS DevOps End-to-End Workshop リポジトリの README ファイルにある「Some challenges if you have time...」のセクションを参照してください。この情報を確認して、セキュリティに関する追加機能を深く掘り下げ、サービスのデカップリングを実践することをお勧めします。

エピック

ソースコードのダウンロード

タスク	説明	必要なスキル
GitHub からソースコードをダウンロードします。	このパターンのソースコードとテンプレートは、GitHub の AWS DevOps End-to-End Workshop リポジトリにあります。「エピック」セクションの次のステップに進む前に、リポジトリからすべてのファイルをコンピュータにダウンロードしてください。 **Note** エピックセクションでは、プロセスに関する一般的な情報を提供するために、このウォークスルーの大まかな手順について説明します。各ステップを完了するには、AWS DevOps End-to-End Workshop リポジトリの README ファイルを参照して詳細な手順を確認してください。 AWS DevOps End-to-End Workshop リポジトリは、AWS Bookstore Demo	アプリ開発者

タスク	説明	必要なスキル
	App リポジトリを拡張し、AWS Cloud9 起動コードの修正バージョンを使用して AWS Cloud9 IDE を作成します。	

Bookstore ウェブアプリケーションと Books マイクロサービスを構築します。

タスク	説明	必要なスキル
Bookstore アプリケーションのフロントエンド関数と Lambda 関数を作成します。	1. CloudFormation コンソールにログインし、`DemoBookstoreMainTemplate.yml` テンプレートをデプロイして DemoBookStoreStack スタックを作成します。これにより、Books マイクロサービスの外部にあるフロントエンド関数と Lambda 関数が作成されます。 2. スタックの [出力] タブで、[WebApplication] ラベルの下にあるウェブサイト URL を書きとめておきます。	開発者
Books マイクロサービスを作成します。	CloudFormation コンソールで、`DemoBookstoreBooksServiceTemplate.yml` テンプレートをデプロイして DemoBooks	開発者

タスク	説明	必要なスキル
	ServiceStack スタックを作成します。	
アプリケーションをテストします。	DemoBookStoreStack スタックのウェブサイト URL から Bookstore アプリケーションにアクセスします。	開発者

Cloud9 環境を使用してアプリケーションを保守する

タスク	説明	必要なスキル
AWS Cloud9 IDE を作成します。	CloudFormation コンソールで、`C9EnvironmentTemplate.yml` テンプレートをデプロイして AWS Cloud9 環境を作成します。	デベロッパー、デベロッパーリーダー
CodeCommit リポジトリを作成します。	1. AWS CodeCommit コンソールにログインし、フロントエンドアプリケーションのコードを含む `demobookstore-WebAssets` リポジトリがあることを確認します。 2. `demobookstore-BooksService` という Books マイクロサービス用のリポジトリを作成します。 3. `git clone` コマンドを使用して、AWS Cloud9 (`demobookstore-WebAssets` と `demobookstore-BooksService`)	開発者

タスク	説明	必要なスキル
	の 2 つのリポジトリを複製します。	
フロントエンドのコードを変更し、パイプラインを確認します。	1. AWS Cloud9 を使用して Web ページのコードを変更します。これにより `demobookstore-WebAssets` リポジトリが更新されます。 2. AWS CodePipeline コンソールで、DemoBookStore-Assets-Pipeline が実行されていることを確認します。 3. ブラウザ (Firefox では Ctrl+F5) からウェブアプリケーションを更新して、ウェブアプリケーションをテストします。	開発者

Books マイクロサービス用の CI/CD パイプラインを実装します。

タスク	説明	必要なスキル
ビルドとサービス更新用の YAML ファイルを追加します。	1. AWS Cloud9 で、`buildspec.yml` および `DemoBooksToreBooksServiceUpdateTemplate.yml` ファイルをアップロードします。 • `buildspec.yml` には構築手順の他、自動テストのテスト手順も含まれ	開発者

タスク	説明	必要なスキル
	ています。この時点でコメントが付けられており、後で使用します。 • DemoBookstoreBooksServiceUpdateTemplate.yml は DemoBookstoreBooksServiceTemplate.yml の更新バージョンで、パイプラインのデプロイ段階で使用されます。 2. ファイルをコミットしてプッシュします。	
ビルドパイプライン用の S3 バケットを作成します。	S3 バケットを作成するには、Amazon S3 ドキュメントの指示に従ってください。 • バケット名はグローバルに一意である必要があります。 例えば、demobookstore-books-service-pipeline-bucket-<YYYYMMDDHHMM> 。 • [すべてのパブリックアクセスをブロックする] チェックボックスをオフにし、[確認する...] チェックボックスを選択します。	開発者

タスク	説明	必要なスキル
IAM を使用して CloudFormation デプロイメント用のロールを作成します。	`demobookstore-CloudFormation-role` ロールを作成し、`AdministratorAccess` ポリシーをアタッチします。次のエピックでは、このロールを最低限の権限に再設定できます。	開発者
Books マイクロサービスの構築とデプロイを自動化する新しいパイプラインを作成してください。	README ファイルの説明に従って、コミット、ビルド、デプロイの各ステージを含むパイプライン (DemoBookStore-BooksService-Pipeline など) を作成します。	開発者
AWS Cloud9 でマイクロサービスをテストします。	ListBooks 関数に変更を加え、パイプラインが機能することを確認します。	開発者
ListBooks Lambda 関数のユニットテストを自動化します。	AWS Cloud9 IDE で、ビルドを有効にしてユニットテストを実行し、テスト結果を確認します。手順については、README ファイルを参照してください。	開発者

(オプション) 追加機能を実装します。

タスク	説明	必要なスキル
解決策を安全にしましょう。	最小限の権限 `demobookstore-CloudFormation-role` を持つように設定し	開発者

タスク	説明	必要なスキル
	、他のユーザーロールも確認してください。	
CloudFormation テンプレート内の依存関係を排除します。	`DemoBookstoreMainTemplate.yml` テンプレートと `DemoBookstoreBooksServiceTemplate.yml` テンプレートの間で情報を交換する方法は、出力とインポートに基づいています。これら 2 つのテンプレート間で値を渡すことで、依存関係が増えます。依存関係をなくすには、AWS Systems Manager Parameter Store の使用を検討してください。	開発者
カートマイクロサービスを作成してください。	Books マイクロサービスを例に挙げると、ショッピングカート関連の機能を `DemoBookstoreMainTemplate.yml` テンプレートから取り除き、Cart マイクロサービスを作成できます。	開発者

クリーンアップ

タスク	説明	必要なスキル
S3 バケットを削除します。	Amazon S3 コンソールで、サンプルウェブアプリケーションに関連付けられている以下のバケットを削除します。	開発者

タスク	説明	必要なスキル
	• AWS Bookstore デモアプリケーション用に 2 つのバケットが作成されました。バケット名は、フロントエンドを作成したときに AWS CloudFormation に提供したスタック名 (例: DemoBookStoreStack) で始まります。 • ビルドパイプライン用に 1 つのバケット (例: demobookstore-books-service-pipeline-bucket-<YYYYMMDDHHMM>)。	
スタックを削除します。	CloudFormation コンソールで、サンプルウェブアプリケーションに関連付けられているスタックを削除します。 • DemoBooksServiceStack • DemoBookStoreStack 削除には 90 分以上かかる場合があります。削除に失敗した場合は、それらを再度削除し、通知に基づいて手動リソース (VPC やネットワークインターフェイスなど) もすべて削除します。	開発者

タスク	説明	必要なスキル
IAM ロールを削除します。	IAM コンソールで、以下のロールを削除します。 • `demobookstore-Cloudformation-role` • `demobookstore-BooksService-BuildProject-service-role` 詳しい手順については、IAM のドキュメントを参照してください。	開発者

関連リソース

- AWS Bookstore Demo App
- AWS Cloud9 起動の例
- AWS CloudFormation コンソールでのスタックの作成 (AWS CloudFormation ドキュメント)
- バケットの作成 (Amazon S3 ドキュメント)

追加情報

詳細な手順については、GitHub AWS DevOps End-to-End Workshop リポジトリの README ファイルを参照してください。

2023 年 5 月の更新について: このパターンは、新しいバージョンの Node と Python を使用するように更新されました。ソースコード内の多くのパッケージを更新し、Glyphicon はフリーではなくなったため削除しました。また、AWS Bookstore Demo App リポジトリへの依存関係もすべて削除したため、2 つのリポジトリを独立して進化させることができるようになりました。

GitHub Actions と Terraform を使用して Docker イメージを構築して Amazon ECR にプッシュする

作成者: R"ka Modi (AWS)

概要

このパターンでは、再利用可能な GitHub ワークフローを作成して Dockerfile を構築し、結果のイメージを Amazon Elastic Container Registry (Amazon ECR) にプッシュする方法について説明します。このパターンは、Terraform と GitHub Actions を使用して Dockerfiles のビルドプロセスを自動化します。これにより、人為的ミスの可能性が最小限に抑えられ、デプロイ時間が大幅に短縮されます。

GitHub リポジトリのメインブランチへの GitHub プッシュアクションにより、リソースのデプロイが開始されます。ワークフローは、GitHub 組織とリポジトリ名の組み合わせに基づいて一意の Amazon ECR リポジトリを作成します。次に、Dockerfile イメージを Amazon ECR リポジトリにプッシュします。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- アクティブな GitHub アカウント。
- GitHub リポジトリ。
- Terraform バージョン 1 以降がインストールされ、設定されています。
- Terraform バックエンド用の Amazon Simple Storage Service (Amazon S3) バケット。
- Terraform 状態ロックと整合性のための Amazon DynamoDB テーブル。テーブルには、タイプが `LockID`の という名前のパーティションキーが必要です`String`。これが設定されていない場合、状態ロックは無効になります。
- Terraform の Amazon S3 バックエンドを設定するアクセス許可を持つ AWS Identity and Access Management (IAM) ロール。設定手順については、Terraform のドキュメントを参照してください。

機能制限

この再利用可能なコードは、GitHub Actions でのみテストされています。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon ECR リポジトリ
- GitHub Actions
- Terraform

ターゲットアーキテクチャ

この図表は、以下を示すものです：

1. ユーザーは、Dockerfile テンプレートと Terraform テンプレートを GitHub リポジトリに追加します。

2. これらの追加により、GitHub Actions ワークフローが開始されます。

3. ワークフローは、Amazon ECR リポジトリが存在するかどうかをチェックします。そうでない場合、GitHub の組織とリポジトリ名に基づいてリポジトリが作成されます。

4. ワークフローは Dockerfile を構築し、Amazon ECR リポジトリにイメージをプッシュします。

ツール

Amazon サービス

- Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブル、信頼性の高いマネージドコンテナレジストリサービスです。

その他のツール

- GitHub Actions は GitHub プラットフォームに統合されており、GitHub リポジトリ内のワークフローの作成、共有、実行に役立ちます。GitHub Actions を使用して、コードの構築、テスト、デプロイなどのタスクを自動化できます。
- Terraform は、クラウドおよびオンプレミスインフラストラクチャの作成と管理に役立つ HashiCorp のInfrastructure as Code (IaC) ツールです。

コードリポジトリ

このパターンのコードは、GitHub Docker ECR Actions Workflow リポジトリで入手できます。

- GitHub Actions を作成すると、Docker ワークフローファイルはこのリポジトリの `/.github/workflows/` フォルダに保存されます。このソリューションのワークフローは workflow.yaml ファイルにあります。
- `e2e-test` フォルダには、リファレンスとテスト用のサンプル Dockerfile が用意されています。

ベストプラクティス

- Dockerfiles を記述するためのベストプラクティスについては、Docker ドキュメントを参照してください。
- Amazon ECR の VPC エンドポイントを使用します。VPC エンドポイントは、プライベート IP アドレスを介して Amazon ECR APIs にプライベートにアクセスできるテクノロジーである AWS PrivateLink を利用しています。Fargate 起動タイプを使用する Amazon ECS タスクの場合、VPC エンドポイントを使用すると、タスクにパブリック IP アドレスを割り当てることなく、タスクは Amazon ECR からプライベートイメージをプルできます。

エピック

OIDC プロバイダーと GitHub リポジトリを設定する

タスク	説明	必要なスキル
OpenID Connect を設定します。	OpenID Connect (OIDC) プロバイダーを作成します。このアクションで使用される IAM ロールの信頼ポリシーでプロバイダーを使用します。手順については、GitHub ドキュメントの「Amazon Web Services での OpenID Connect の設定」を参照してください。	AWS 管理者、AWS DevOps、AWS 全般

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	GitHub Docker ECR Actions Workflow リポジトリをローカルフォルダにクローンします。 `$git clone https://github.com/aws-samples/docker-ecr-actions-workflow`	DevOps エンジニア

GitHub の再利用可能なワークフローをカスタマイズし、Docker イメージをデプロイする

タスク	説明	必要なスキル
Docker ワークフローを開始するイベントをカスタマイズします。	このソリューションのワークフローは workflow.yaml にあります。このスクリプトは現在、`workflow_dispatch` イベントを受信したときにリソースをデプロイするように設定されています。イベントを `workflow_call` に変更し、別の親ワークフローからワークフローを呼び出すことで、この設定をカスタマイズできます。	DevOps エンジニア
ワークフローをカスタマイズします。	Workflow.yaml ファイルは、動的で再利用可能な GitHub ワークフローを作成するように設定されています。このファイルを編集してデフォルト設定をカスタマイズ	DevOps エンジニア

タスク	説明	必要なスキル

	することも、workflow_dispatch イベントを使用してデプロイを手動で開始する場合は、GitHub Actions コンソールから入力値を渡すこともできます。
	
	• 正しい AWS アカウント ID とターゲットリージョンを指定してください。
	• Amazon ECR ライフサイクルポリシーを作成し (サンプルポリシーを参照)、それに応じてデフォルトパス (e2e-test/policy.json) を更新します。
	• ワークフローファイルには、入力として 2 つの IAM ロールが必要です。
	  • Terraform の Amazon S3 バックエンドを設定するアクセス許可を持つ IAM ロール (「前提条件」セクションを参照)。デフォルトのロール名は、それに応じて .yaml ファイルworkload-assumable-role で更新できます。
	  • GitHub へのアクセス許可を持つ IAM ロール。このロールは Amazon ECR ポリシーでも使用され、Amazon ECR	

タスク	説明	必要なスキル
	オペレーションを制限します。詳細については、data.tf ファイルを参照してください。	
Terraform テンプレートをデプロイします。	ワークフローは、設定した GitHub イベントに基づいて、Amazon ECR リポジトリを作成する Terraform テンプレートを自動的にデプロイします。これらのテンプレートは、Github リポジトリのルートにある `.tf`ファイルとして使用できます。	AWS DevOps、DevOps エンジニア

トラブルシューティング

問題	ソリューション
Amazon S3 と DynamoDB を Terraform リモートバックエンドとして設定する際の問題またはエラー。	Terraform ドキュメントの指示に従って、リモートバックエンド設定の Amazon S3 および DynamoDB リソースに必要なアクセス許可を設定します。
`workflow_dispatch` イベントでワークフローを実行または開始できません。	`workflow_dispatch` イベントからデプロイするように設定されたワークフローは、ワークフローがメインブランチでも設定されている場合のみ機能します。

関連リソース

- ワークフローの再利用 (GitHub ドキュメント)
- ワークフローのトリガー (GitHub ドキュメント)

AWS CodeCommit、AWS CodePipeline、AWS Device Farm を使用して iOS アプリを構築してテストする

作成者:Abdullahi Olaoye (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンでは、AWS CodePipeline を使用して AWS 上の実際のデバイスで iOS アプリケーションを構築してテストする、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを作成する手順の概要を示しています。このパターンでは、AWS CodeCommit を使用してアプリケーションコードを保存し、Jenkins のオープンソースツールを使用して iOS アプリケーションを構築し、AWS Device Farm を使用して構築したアプリケーションを実際のデバイス上でテストします。これらの 3 つのフェーズは、AWS CodePipeline を使用してパイプラインにまとめてオーケストレーションされます。

このパターンは、AWS DevOps ブログの「AWS DevOps とモバイルサービスを使用した iOS および iPadOS アプリの構築とテスト」の投稿に基づいています。詳細な手順については、このブログ記事を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Apple 開発者アカウント
- ビルドサーバー (macOS)
- Xcode バージョン 11.3 (ビルドサーバーにインストールしてセットアップ済み)
- AWS コマンドラインインターフェイス (AWS CLI) をインストールして設定済み
- Git の基本的な知識

機能制限

- アプリケーションビルドサーバーは macOS を実行していることが必要です。
- CodePipeline が構築を開始するためにリモートで接続できるように、ビルドサーバーにはパブリック IP アドレスが必要です。

アーキテクチャ

ソーステクノロジースタック

- シミュレータを使用するか、物理デバイス上で手動テストを行うオンプレミスの iOS アプリケーション構築プロセス

ターゲットテクノロジースタック

- アプリケーションのソースコードを保存するための AWS CodeCommit リポジトリ
- Xcode を使用してアプリケーションを構築するための Jenkins サーバー
- 実際のデバイスでアプリケーションをテストするための AWS Device Farm デバイスプール

ターゲット アーキテクチャ

ユーザーがソースリポジトリに変更をコミットすると、パイプライン (AWS CodePipeline) はソースリポジトリからコードを取得し、Jenkins ビルドを開始して、アプリケーションコードを Jenkins に渡します。ビルドの完了後、パイプラインはビルドアーティファクトを取得し、AWS Device Farm ジョブを開始してデバイスプールに対してアプリケーションをテストします。

ツール

- AWS CodePipeline は、完全マネージド型の継続的デリバリーサービスであり、リリースパイプラインを自動化して、アプリケーションとインフラストラクチャを迅速かつ確実に更新するのに役立ちます。CodePipeline はお客様が定義したリリースモデルに基づき、コードチェンジがあった場合のフェーズの構築、テスト、およびデプロイを自動化します。
- AWS CodeCommit は、セキュアな Git ベースのリポジトリをホストするフルマネージド型のソースコントロールサービスです。これにより、セキュアで拡張性の高い エコシステムで、チームによるコードの共同作業が簡単になります。CodeCommit を使用することにより、独自のソース管理システムを運用したり、そのインフラストラクチャをスケールしたりする必要がなくなります。
- AWS Device Farm は、テストインフラストラクチャをプロビジョニングおよび管理することなく、幅広いデスクトップブラウザと実際のモバイルデバイスでテストすることで、ウェブおよびモバイルアプリケーションの品質を向上させるアプリケーションテストサービスです。
- Jenkins は、デベロッパーがソフトウェアを確実に構築、テスト、デプロイできるようにするオープンソースのオートメーションサーバーです。

エピック

ビルド環境をセットアップします。

タスク	説明	必要なスキル
macOS を実行しているビルドサーバーに Jenkins をインストールします。	Jenkins を使用してアプリケーションを構築するので、最初に Jenkins をビルドサーバーにインストールする必要があります。このタスクとそれ以降のタスクの詳細な手順については、このパターンの最後にある関連リソースセクションの「AWS DevOps およびモバイルサービスおよびその他のリソースを使用した iOS および iPadOS アプリの構築とテスト」のブログ記事を参照してください。	DevOps
Jenkins を設定します。	画面上の指示に従って Jenkins を設定します。	DevOps
Jenkins 用の AWS CodePipeline プラグインをインストールします。	Jenkins が AWS CodePipeline サービスとやり取りできるようにするには、このプラグインを Jenkins サーバーにインストールする必要があります。	DevOps
自由形式の Jenkins プロジェクトを作成します。	Jenkins で、自由形式のプロジェクトを作成します。トリガーやその他のビルド設定オプションを指定するようにプロジェクトを設定します。	DevOps

AWS Device Farm を設定する

タスク	説明	必要なスキル
Device Farm プロジェクトを作成します。	AWS Device Farm コンソールを開きます。テスト用のプロジェクトとデバイスプールを作成します。手順については、ブログ記事を参照してください。	開発者

ソースリポジトリを設定する

タスク	説明	必要なスキル
CodeCommit リポジトリを作成します。	ソースコードを保存するリポジトリを作成します。	DevOps
アプリケーションコードをリポジトリにコミットします。	作成した CodeCommit リポジトにリコネクトします。 コードをローカルマシンからリポジトリにプッシュします。	DevOps

パイプラインを設定する

タスク	説明	必要なスキル
AWS CodePipeline でパイプラインを作成します。	AWS CodePipeline コンソールを使用してパイプラインを作成します。パイプラインは CI/CD プロセスのすべてのフェーズをオーケストレーションします。手順については、AWS ブログ記事「AWS DevOps およびモバイルサー	DevOps

タスク	説明	必要なスキル
	ビスを使用した iOS および iPadOS アプリの構築とテスト」を参照してください。	
パイプラインにテストステージを追加します。	テストステージを追加して AWS Device Farm と統合するには、パイプラインを編集します。	DevOps
パイプラインを開始します。	パイプラインと CI/CD プロセスを開始するには、「変更のリリース」を選択します。	DevOps

アプリケーションテストの結果を表示します。

タスク	説明	必要なスキル
テスト結果を確認します。	AWS Device Farm コンソールで、作成したプロジェクトを選択し、テストの結果を確認します。コンソールには、各テストの詳細が表示されます。	開発者

関連リソース

このパターンのステップバイステップの説明

- 「AWS DevOps とモバイルサービスを使用した iOS および iPadOS アプリの構築とテスト」(AWS DevOps ブログ記事)

AWS Device Farm を設定する

- AWS Device Farm コンソール

ソースリポジトリを設定する

- 「新しい AWS CodeCommit リポジトリの作成」
- 「AWS CodeCommit リポジトリに接続する」

「パイプラインを設定する」

- 「AWS CodePipeline コンソール」

追加リソース

- 「AWS CodePipeline ドキュメント」
- 「AWS CodeCommit ドキュメント」
- 「AWS Device Farm ドキュメント」
- 「Jenkins ドキュメント」
- 「macOS での Jenkins のインストール」
- 「Jenkins 用の AWS CodePipeline プラグイン」
- 「Xcode のインストール」
- AWS CLI の インストールと設定
- 「Git ドキュメント」

AWS CDK アプリケーションまたは CloudFormation テンプレートのベストプラクティスを cdk-nag ルールパックを使用して確認する

作成者: Arun Donti

概要

このパターンでは、cdk-nag ユーティリティを使用して、ルールパックの組み合わせにより AWS Cloud Development Kit (AWS CDK) アプリケーションのベストプラクティスを確認する方法を説明しています。cdk-nag は cfn_nag からヒントを得たオープンソースプロジェクトです。CDK Aspects を使用して、AWS ソリューションライブラリ、医療保険の携行性と責任に関する法律 (HIPAA)、米国立標準技術研究所 (NIST) 800-53 などの評価パックにルールを実装しています。これらのパックに含まれるルールを使用して AWS CDK アプリケーションのベストプラクティスを確認したり、ベストプラクティスに基づいてコードを検出して修正したり、評価に使用したくないルールを抑制したりできます。

また、cdk-nag を使用することで、cloudformation-include モジュールを使って AWS CloudFormation テンプレートを確認することもできます。

利用可能なすべてのパックについては、cdk-nag リポジトリの「ルール」セクションを参照してください。評価パックは以下に対するものが利用可能です。

- ソリューションライブラリ
- HIPAA security
- NIST 800-53 rev 4
- NIST 800-53 rev 5
- Payment Card Industry Data Security Standard (PCI DSS) 3.2.1

前提条件と制限

前提条件

- CDK を使用するアプリケーション

ツール

- CDK — Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、AWS CloudFormation を通じてプロビジョニングするためのソフトウェア開発フレームワークです。
- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。テンプレートを使用してリソースとその依存関係を記述し、リソースを個別に管理する代わりに、それらをスタックとしてまとめて起動して構成できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理し、プロビジョニングすることが可能です。

エピック

cdk-nag を AWS CDK アプリケーションに統合する

タスク	説明	必要なスキル
cdk-nag について学びます。	cdk-nag GitHub リポジトリに移動して、ドキュメントに目を通します。	アプリ開発者
cdk-nag パッケージを AWS CDK アプリケーションにインストールします。	AWS CDK アプリケーションで cdk-nag を使用するには、最初にこれをインストールする必要があります。cdk-nag は PyPI、npm、NuGet、および Apache Maven からダウンロードできます。入手可能なバージョンとダウンロード場所に関する最新情報については、リポジトリの Readme ファイルを参照してください。	アプリ開発者
NagPacks を選択します。	cdk-nag には、NagPacks と呼ばれるさまざまなルールパッ	アプリ開発者

タスク	説明	必要なスキル
	クがあります。各 NAGPack には、特定の標準に準拠するルールが含まれています。たとえば、AWS ソリューション NagPack には一般的なベストプラクティスが含まれており、NIST 800-53 rev 5 NagPack はコンプライアンスに役立ちます。アプリケーションには複数の NAGPack を適用でき、必要に応じてパックを追加したり削除したりできます。利用可能なパックのリストについては、GitHub リポジトリの Readme ファイルを参照してください。各パックの個々のルールについては、GitHub リポジトリの Rules セクションを参照してください。	

タスク	説明	必要なスキル

cdk-nag を AWS CDK アプリケーションに統合します。	cdk-nag はアプリケーション全体に統合することも、アプリケーションの個々のステージやスタックに統合することもできます。たとえば、AWS Solutionsと HIPAA security NagPacks をアプリケーション全体レベルで AWS CDK v2 TypeScript アプリケーションに統合するには、次のコードを使用できます。
	
	`import { App, Aspects } from 'aws-cdk-lib';`
	`import { CdkTestStack } from '../lib/cdk-test-stack';`
	`import { AwsSolutionsChecks, HIPAASecurityChecks } from 'cdk-nag';`
	
	`const app = new App();`
	`new CdkTestStack(app, 'CdkNagDemo');`
	`// Simple rule informational messages`
	`Aspects.of(app).add(new AwsSolutionsChecks());`
	`// Additional explanations on the purpose of triggered rules`
	`Aspects.of(app).add(new HIPAASecurityChecks({ verbose: true }));`
		アプリ開発者

関連リソース

- cdk-nag コードリポジトリ
- Construct Hub の cdk-nag

Amazon EKS で実行されているアプリケーションの相互 TLS 認証を設定する

作成者：マヘンドラ・シッダッパ (AWS)

概要

証明書ベースの相互 Transport Layer Security (TLS) は、サーバーとクライアント間の双方向ピア認証を提供するオプションの TLS コンポーネントです。相互 TLS では、クライアントはセッションネゴシエーションプロセス中に X.509 証明書を提供する必要があります。サーバーは、この証明書を使用してクライアントを識別し、認証します。

相互 TLS は、モノのインターネット (IoT) アプリケーションの一般的な要件であり、ビジネストゥビジネスアプリケーションやオープンバンキングなどの標準に使用できます。

このパターンでは、NGINX Ingress Controller を使用して Amazon Elastic Kubernetes Service (Amazon EKS) クラスターで実行されているアプリケーションの相互 TLS を設定する方法を説明します。イングレスリソースに注釈を付けることで、NGINX イングレスコントローラーのビルトイン相互 TLS 機能を有効にできます。NGINX コントローラーの相互 TLS アノテーションについて詳しくは、Kubernetes ドキュメントの「クライアント証明書認証」を参照してください。

> **Important**
>
> このパターンでは、自己署名証明書を使用します。このパターンはテストクラスターでのみ使用し、本番環境では使用しないことをお勧めします。このパターンを本番環境で使用する場合は、「AWS プライベート認証局 (AWS プライベート CA)」または既存のパブリックキーインフラストラクチャ (PKI) 標準を使用してプライベート証明書を発行できます。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS)アカウント。
- 既存の アマゾン EKS クラスター。
- AWS コマンドラインインターフェイス(AWS CLI) バージョン 1.7 以降。macOS、Linux、または Windows にインストールされ、設定されている。

- Amazon EKS クラスターにアクセスするためにインストールして設定した kubectl コマンドラインユーティリティ。詳細については、Amazon EKS ドキュメントの「kubectl のインストール」を参照してください。
- アプリケーションをテストするための既存のドメインネームシステム (DNS) 名。

機能制限

- このパターンでは、自己署名証明書を使用します。このパターンはテストクラスターでのみ使用し、本番環境では使用しないことをお勧めします。

アーキテクチャ

テクノロジースタック

- Amazon EKS
- Amazon Route 53
- kubectl

ツール

- 「Amazon Elastic Kubernetes Service (Amazon EKS)」は、AWS で Kubernetes を実行する際に役立ち、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持する必要はありません。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- 「Kubectl」は、Amazon EKS クラスターを操作するために使用するコマンドラインユーティリティです。

エピック

自己署名証明書を生成します

タスク	説明	必要なスキル
CA キーと証明書を生成します。	次のコマンドを実行して、証明機関 (CA) キーと証明書を生成します。 `openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Test Cert Authority'`	DevOps エンジニア
サーバーキーと証明書を生成し、CA 証明書で署名します。	サーバーキーと証明書を生成し、次のコマンドを実行して CA 証明書で署名します。 `openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN= <your_domain_name> ' && openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt` **Important** を既存のドメイン名<your_dom	DevOps エンジニア

タスク	説明	必要なスキル
	ain_name> に置き換えてください。	
クライアントキーと証明書を生成し、CA 証明書で署名します。	クライアントキーと証明書を生成し、次のコマンドを実行して CA 証明書で署名します。 `openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Test' && openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt`	DevOps エンジニア

NGINX イングレスコントローラーをデプロイします。

タスク	説明	必要なスキル
Amazon EKS クラスターに NGINX イングレスコントローラーをデプロイします。	次のコマンドを使用して、NGINX イングレスコントローラをデプロイします。 `kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.0/deploy/static/provider/aws/deploy.yaml`	DevOps エンジニア

タスク	説明	必要なスキル
NGINX Ingress Controller サービスが実行中であることを確認します。	以下のコマンドを使用して、NGINX イングレスコントローラサービスが実行されていることをを確認します。 `kubectl get svc -n ingress-nginx` **⚠ Important** サービスアドレスのフィールドに Network Load Balancer のドメイン名が含まれていることを確認します。	DevOps エンジニア

Amazon EKS クラスターにネームスペースを作成して、相互 TLS をテストします。

タスク	説明	必要なスキル
Amazon EKS クラスターにネームスペースを作成します。	次のコマンドを実行して、Amazon EKS クラスターに `mtls` という名前の名前空間を作成します。 `kubectl create ns mtls` これにより、相互 TLS をテストするためのサンプルアプリケーションがデプロイされます。	DevOps エンジニア

サンプルアプリケーションのデプロイとサービスを作成します。

タスク	説明	必要なスキル

Kubernetes デプロイメントとサービスを mtls 名前空間に作成します。	`mtls.yaml` という名前のファイルを作成します。ファイルに次のコードを貼り付けます。
	
	`kind: Deployment`
	`apiVersion: apps/v1`
	`metadata:`
	`  name: mtls-app`
	`  labels:`
	`  app: mtls`
	`spec:`
	`  replicas: 1`
	`  selector:`
	`  matchLabels:`
	`  app: mtls`
	`  template:`
	`  metadata:`
	`  labels:`
	`  app: mtls`
	`  spec:`
	`  containers:`
	`  - name: mtls-app`
	`  image: hashicorp/http-echo`
	`  args:`
	`  - "-text=mTLS is working"`
	
	`---`
	
	`kind: Service`
	`apiVersion: v1`
	`metadata:`
	`  name: mtls-service`
	`spec:`
	`  selector:`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre>app: mtls
	ports:
	- port: 5678 #
	Default port for image</pre>
	次のコマンドを実行して、mtls 名前空間に Kubernetes デプロイとサービスを作成します。
	<pre>kubectl create -f
	mtls.yaml -n mtls</pre>	

Kubernetes デプロイが作成されていることを確認します。	デプロイが作成され、1 つのポッドが使用可能になっていることを確認するには、次のコマンドを実行します。
	<pre>kubectl get deploy -n
	mtls</pre>
		DevOps エンジニア

Kubernetes サービスが作成されていることを確認します。	次のコマンドを実行して、Kubernetes サービスが作成されたことを確認します。
	<pre>kubectl get service -n
	mtls</pre>
		DevOps エンジニア

mtls 名前空間にシークレットを作成します。

タスク	説明	必要なスキル
イングレスリソースにシークレットを作成します。	以下のコマンドを実行して、前に作成した証明書を使用して NGINX Ingress コントロー	DevOps エンジニア

タスク	説明	必要なスキル
	ラーのシークレットを作成します。 `kubectl create secret generic mtls-certs --from-file=tls.crt=server.crt --from-file=tls.key=server.key --from-file=ca.crt=ca.crt -n mtls` シークレットには、サーバーを識別するためのクライアント用のサーバー証明書と、クライアント証明書を検証するためのサーバー用の CA 証明書があります。	

mtls 名前空間に Ingress リソースを作成します。

タスク	説明	必要なスキル

mtls 名前空間にイングレスリソースを作成します。	`ingress.yaml` という名前のファイルを作成します。ファイルに次のコードを貼り付けます (`<your_domain_name>` を既存のドメイン名に置き換えます)。
	
	`apiVersion: networking.k8s.io/v1`
	`kind: Ingress`
	`metadata:`
	`  annotations:`
	`  nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre> nginx.ingress.kube
	rnetes.io/auth-tls-
	secret: mtls/mtls-certs
	  name: mtls-ingress
	spec:
	  ingressClassName:
	 nginx
	  rules:
	  - host: "*.<your_
	domain_name>"
	    http:
	      paths:
	      - path: /
	        pathType: Prefix
	        backend:
	          service:
	            name: mtls-
	service
	            port:
	              number:
	 5678
	  tls:
	  - hosts:
	    - "*.<your_
	domain_name>"
	    secretName: mtls-
	certs</pre>
	次のコマンドを実行して、mtls ネームスペースに Ingress リソースを作成します。
	<pre>kubectl create -f
	 ingress.yaml -n mtls</pre>
	つまり、NGINX Ingress コントローラーはトラフィックを	

タスク	説明	必要なスキル
	サンプルアプリケーションにルーティングできます。	
Ingress リソースが作成されていることを確認します。	次のコマンドを実行して、Ingress リソースが作成されたことを確認します。 `kubectl get ing -n mtls` **⚠ Important** Ingress リソースのアドレスに、NGINX Ingress Controller 用に作成されたロードバランサーが表示されていることを確認します。	DevOps エンジニア

ホスト名がロードバランサーを指すように DNS を設定します。

タスク	説明	必要なスキル
NGINX Ingress コントローラーのロードバランサーを指す CNAME レコードを作成します。	AWS マネジメントコンソールにサインインし、Amazon Route 53 コンソールを開いて、NGINX イングレスコントローラーのロードバランサーに `mtls.<your_domain_name>` を指す正規名 (CNAME) レコードを作成します。 詳細については、Route 53 ドキュメントの「Route 53 コン	DevOps エンジニア

タスク	説明	必要なスキル
	ソールを使用したレコードの作成」を参照してください。	

アプリケーションをテストする

タスク	説明	必要なスキル
証明書なしで相互 TLS セットアップをテストする。	以下のコマンドを実行してください。 `curl -k https://mtls.<your_domain_name>` 「400 必要な SSL 証明書は送信されませんでした」というエラー応答が表示されるはずです。	DevOps エンジニア
証明書を使用して相互 TLS セットアップをテストします。	以下のコマンドを実行してください。 `curl -k https://mtls.<your_domain_name> --cert client.crt --key client.key` 「mTLS は動作しています」という応答が返されるはずです。	DevOps エンジニア

関連リソース

- 「Amazon Route 53 コンソールを使用したレコードの作成」
- 「Amazon EKS の NGINX 入力コントローラーでのNetwork Load Balancer の使用」

- 「クライアント証明書認証」

AWS CloudFormation を使用して AppStream 2.0 リソースの作成を自動化する

作成者: Ram Kandaswamy (AWS)

概要

このパターンは、AWS CloudFormation テンプレートを使用してAmazon Web Services (AWS) クラウドでの Amazon AppStream 2.0 リソースの作成を自動化するためのコードサンプルとステップを提供します。このパターンは、AWS CloudFormation スタックを使用して、イメージビルダー、イメージ、フリートインスタンス、スタックなどの AppStream 2.0 アプリケーションリソースの作成を自動化する方法を示しています。AppStream 2.0 アプリケーションは、デスクトップ配信モードまたはアプリケーション配信モードのいずれかを使用して HTML5 準拠のブラウザでエンドユーザーにストリーミングできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AppStream 2.0 利用規約への同意
- スタック、フリート、イメージビルダーなどの AppStream リソースに関する基本的な知識

制限

- AppStream 2.0 インスタンスに関連付けられている AWS Identity and Access Management (IAM) ロールは、インスタンスの作成後に変更することはできません。
- イメージビルダーの作成後に AppStream 2.0 イメージビルダーインスタンスのプロパティ (サブネットやセキュリティグループなど) を変更することはできません。

アーキテクチャ

次の図は、AWS CloudFormation テンプレートを使用して AppStream 2.0 リソースの作成を自動化する方法を示しています。

この図表は、次のワークフローを示しています：

1. このパターンの「追加情報」セクションにある YAML コードに基づいて AWS CloudFormation テンプレートを作成します。
2. AWS CloudFormation テンプレートは AWS CloudFormation テストスタックを作成します。
   a. (オプション) AppStream 2.0 を使用して Image Builder インスタンスを作成します。
   b. (オプション) カスタムソフトウェアを使用して Windows イメージを作成します。
3. AWS CloudFormation スタックは AppStream 2.0 フリートインスタンスとスタックを作成します。
4. AppStream 2.0 リソースを HTML5 準拠のブラウザでエンドユーザーにデプロイします。

テクノロジースタック

- Amazon AppStream 2.0
- AWS CloudFormation

ツール

- Amazon AppStream 2.0 は、どこからでもデスクトップアプリケーションに瞬時にアクセスできるフルマネージドアプリケーションストリーミングサービスです。AppStream 2.0 は、アプリケーションをホストして実行するために必要な AWS リソースを管理します。また、自動的にスケールし、ユーザーにオンデマンドでアクセスを提供します。
- 「AWS CloudFormation」は、AWS リソースのモデル化と設定、迅速かつ一貫したプロビジョニング、ライフサイクル全体にわたる管理を支援します。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにわたってスタックを管理およびプロビジョニングできます。

エピック

(オプション) AppStream 2.0 イメージを作成する

タスク	説明	必要なスキル
カスタムソフトウェアをインストールしてイメージを作成します。	1. ユーザーにデプロイする AppStream 2.0 アプリケー	AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル
	ションをインストールします。 2. Photon のイメージ作成エージェントまたは PowerShell スクリプトを使用して、カスタムソフトウェア用の新しい Windows イメージを作成します。 **Note** Windows AppLocker 機能を使用してイメージをさらにロックダウンすることを検討してください。	

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation テンプレートを更新します。	1. このパターンの「追加情報」セクションのコードを YAML ファイルとして保存します。 2. YAML ファイルを、環境内のパラメータに必要な値で更新します。	AWS システム管理者、クラウド管理者、クラウドアーキテクト、AWS 全般、AWS 管理者
テンプレートを使用して AWS CloudFormation スタックを作成します。	1. AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開きます。	アプリ所有者、AWS システム管理者、Windows エンジニア

タスク	説明	必要なスキル
	2. ナビゲーションペインで、[スタック] を選択します。 3. [スタックの作成] を選択し、[With new resources (standard) 新しいリソースを使用 (標準)] を選択します。 4. [前提条件 - テンプレートの準備] セクションで、 [テンプレートの準備完了] を選択します。 5. [テンプレートの指定] セクションで、[テンプレートファイルのアップロード] を選択します。 6. [ファイルを選択] を選択し、更新した AWS CloudFormation テンプレートを選択します。 7. ウィザードの残りの手順を完了してスタックを作成します。	

関連リソース

リファレンス

- Amazon AppStream 2.0 の使用を開始する: サンプルアプリケーションを使用してセットアップする
- AppStream 2.0 フリートとスタックを作成する

チュートリアルと動画

- Amazon AppStream 2.0 User Workflow

- How to Migrate a Legacy Windows Forms App to Amazon AppStream 2.0
- AWS re:Invent 2018: Securely Deliver Desktop Applications with Amazon AppStream 2.0 (BAP201)

追加情報

次のコードは、AppStream 2.0 リソースを自動的に作成できる AWS CloudFormation テンプレートの例です。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  SubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
  testSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup::Id'
  ImageName:
    Type: String
Resources:

  AppStreamFleet:
    Type: 'AWS::AppStream::Fleet'
    Properties:
      ComputeCapacity:
        DesiredInstances: 5
      InstanceType: stream.standard.medium
      Name: appstream-test-fleet
      DisconnectTimeoutInSeconds: 1200
      FleetType: ON_DEMAND
      IdleDisconnectTimeoutInSeconds: 1200
      ImageName: !Ref ImageName
      MaxUserDurationInSeconds: 345600
      VpcConfig:
        SecurityGroupIds:
          - !Ref testSecurityGroup
        SubnetIds: !Ref SubnetIds
  AppStreamStack:
    Type: 'AWS::AppStream::Stack'
    Properties:
      Description: AppStream stack for test
      DisplayName: AppStream test Stack
      Name: appstream-test-stack
      StorageConnectors:
        - ConnectorType: HOMEFOLDERS
```

```
      UserSettings:
        - Action: CLIPBOARD_COPY_FROM_LOCAL_DEVICE
          Permission: ENABLED
        - Action: CLIPBOARD_COPY_TO_LOCAL_DEVICE
          Permission: ENABLED
        - Action: FILE_DOWNLOAD
          Permission: ENABLED
        - Action: PRINTING_TO_LOCAL_DEVICE
          Permission: ENABLED
  AppStreamFleetAssociation:
    Type: 'AWS::AppStream::StackFleetAssociation'
    Properties:
      FleetName: appstream-test-fleet
      StackName: appstream-test-stack
    DependsOn:
      - AppStreamFleet
      - AppStreamStack
```

Firelens ログルーターを使用して Amazon ECS 用のカスタムログパーサーを作成する

作成者: Varun Sharma (AWS)

概要

Firelensは、Amazon Elastic Container Service (Amazon ECS) と AWS Fargate 用のログルーターです。Firelens を使用して、Amazon ECS から Amazon CloudWatch やその他の転送先 (Splunk や Sumo Logic など) にコンテナログをルーティングできます。Firelens は Fluentd または Fluent Bit をロギングエージェントとして使用し、これにより Amazon ECS タスク定義パラメータを使用してログをルーティングできます。

ログをソースレベルで解析することを選択することで、ロギングデータを分析し、クエリを実行して、運用上の問題に効率的かつ効果的に対応できます。アプリケーションが異なればロギングパターンも異なるため、ログを構造化して最終転送先での検索を容易にするカスタムパーサーを使用する必要があります。

このパターンでは、カスタムパーサーを備えた Firelens ログルーターを使用して、Amazon ECS で実行されているサンプル Spring Boot アプリケーションから CloudWatch にログをプッシュします。その後、Amazon CloudWatch Logs Insights を使用して、カスタムパーサーによって生成されたカスタムフィールドに基づいてログをフィルタリングできます。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS)アカウント。
- ユーザーのローカルマシンにインストールされ、構成された AWS コマンドラインインターフェイス (AWS CLI)。
- コンピュータにインストールされて構成されている Docker。
- Amazon Elastic Container Registry (Amazon ECR) にある既存の Spring Boot ベースのコンテナ化アプリケーション。

アーキテクチャ

テクノロジースタック

- CloudWatch
- Amazon ECR
- Amazon ECS
- Fargate
- Docker
- Fluent Bit

ツール

- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理を簡単に行うことのできる、高度にスケーラブルで高速なコンテナ管理サービスです。
- Identity and Access Management (IAM) - IAM は、AWS サービスへのアクセスをセキュアに制御するためのウェブサービスです。
- CLI – AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールであり、コマンドラインシェルのコマンドを使って AWS サービスと対話することができます。
- Docker — Docker は、アプリケーションの開発、出荷、実行のためのオープンプラットフォームです。

コード

このパターンには以下のファイルが添付されています。

- `customFluentBit.zip` – カスタムの解析と構成を追加するためのファイルが含まれています。
- `firelens_policy.json` – IAM ポリシーを作成するためのポリシードキュメントが含まれます。
- `Task.json` – Amazon ECS のサンプルタスク定義が含まれています。

エピック

カスタム Fluent Bit イメージの作成

タスク	説明	必要なスキル
Amazon ECR リポジトリを作成します。	AWS マネジメントコンソールにサインインし、Amazon ECR コンソールを開いて、`fluentbit_custom` というリポジトリを作成します。 これに関する詳細については、Amazon ECR のドキュメントの「リポジトリを作成する」を参照してください。	システム管理者、開発者
customFluentBit.zip パッケージを解凍する。	1. `customFluentBit.zip` パッケージ (添付) をローカルマシンにダウンロードします。 2. `customFluentBit` ディレクトリに移動して、次のコマンドを実行します: `unzip -d customFluentBit.zip` 3. このディレクトリには、カスタムの解析と構成を追加するのに必要な以下のファイルが含まれています。 • `parsers/springboot_parser.conf` – パーサーディレクティブが含まれ、カスタムパーサーの正規表現 (regex)	

タスク	説明	必要なスキル
	パターンを定義します。特定のパーサーに正規表現パターンを追加できます。 • `conf/parse_springboot.conf` – フィルターとサービスディレクティブが含まれます。 • Dockerkerfile	
カスタム Docker イメージを作成します。	1. ディレクトリを `customFluentBit` に変更します。 2. Amazon ECR コンソールを開き、`fluentbit_custom` リポジトリを選択してから [プッシュコマンドを表示] を選択します。 3. プロジェクトをアップロードします。 4. アップロードが完了したら、ビルドの URL をコピーします。この URL は Amazon ECS でコンテナを作成する際に必要です。 詳細については、Amazon ECR のドキュメントの「Docker イメージの作成」を参照してください。	システム管理者、開発者

Amazon ECS クラスターの設定

タスク	説明	必要なスキル
Amazon ECS クラスターを作成します。	Amazon ECS クラスターを作成するには、Amazon ECS ドキュメントの「クラスターの作成」の「ネットワーク専用テンプレート」セクションの指示に従います。 **Note** Amazon ECS クラスターの新しい Virtual Private Cloud (VPC) を作成するには、VPC の作成 を必ず選択してください。	システム管理者、開発者

Amazon ECS タスクを設定する

タスク	説明	必要なスキル
Amazon ECS タスク実行 IAM ロールを設定します。	`AmazonECSTaskExecutionRolePolicy` マネージドポリシーを使用して Amazon ECS タスク実行 IAM ロールを作成します。これに関する詳細については、Amazon ECS デベロッパーガイドの 「Amazon ECS タスク実行 IAM ロール」 を参照してください。	システム管理者、開発者

タスク	説明	必要なスキル
	Note IAM ロールの Amazon リソースネーム (ARN) を必ず記録してください。	
IAM ポリシーを Amazon ECS タスク実行 IAM ロールに添付する。	1. `firelens_policy.json` (添付された) ポリシードキュメントを使用して IAM ポリシーを作成します。これに関する詳細については、IAM ドキュメントの「JSON タブでのポリシーの作成」を参照してください。 2. このポリシーを、先に作成した Amazon ECS タスク実行 IAM ロールに添付します。詳細については、IAM ドキュメントの「IAM ポリシーの追加 (CLI)」を参照してください。	システム管理者、開発者

タスク	説明	必要なスキル
Amazon ECS タスク定義のセットアップ	1. `Task.json` サンプルタスク定義 (添付) の以下のセクションを更新してください。 • タスク実行 IAM ロールの ARN を使用して、`executionRoleArn` と `taskRoleArn` を更新します。 • 先に作成したカスタム Fluent Bit Docker イメージで、`containerDefinitions` のイメージを更新します。 • `containerDefinitions` のイメージを、アプリケーションイメージの名前で更新します。 2. Amazon ECS コンソールを開き、[タスク定義] を選択し、[新しいタスク定義の作成] を選択してから、[互換性の選択] ページで [Fargate] を選択します。 3. [Json による構成] を選択し、更新した `Task.json` ファイルをテキスト領域に貼り付けて、[保存] を選択します。 4. タスク定義を作成します。	システム管理者、開発者

タスク	説明	必要なスキル
	これに関する詳細については、Amazon ECS ドキュメントの「タスク定義の作成」」を参照してください。	

Amazon ECS タスクを実行する

タスク	説明	必要なスキル
Amazon ECS タスクを実行します。	Amazon ECS コンソールで [クラスター] を選択し、先に作成したクラスターを選択して、スタンドアロンタスクを実行します。 これに関する詳細については、Amazon ECS ドキュメントの「スタンドアロンタスクの実行」を参照してください。	システム管理者、開発者

CloudWatch のログを検証する

タスク	説明	必要なスキル
ログを検証します。	1. CloudWatch コンソールを開き、[ロググループ] を選択して、`/aws/ecs/containerinsights/{{cluster_ARN}}/firelens/application` を選択します。 2. ログ、特にカスタムパーサーによって追加された力	システム管理者、開発者

タスク	説明	必要なスキル
	スタムフィールドを検証します。 3. CloudWatch を使用して、カスタムフィールドに基づいてログをフィルタリングします。	

関連リソース

- Amazon ECS のドッカーの基本
- AWS Fargate 上の Amazon ECS
- 基本的なサービスパラメーターの構成

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

CodePipeline と HashiCorp Packer を使用してパイプラインと AMI を作成

作成者：Akash Kumar (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンでは、AWS CodePipeline を使用してAmazon Web Services (AWS) クラウドにパイプラインを作成し、HashiCorp Packer を使用してAmazon マシンイメージ (AMI) にパイプラインを作成するコードサンプルとステップを提供します。このパターンでは、Git ベースのバージョン管理システムでコードの構築とテストを自動化する「継続的インテグレーション」 手法に基づいています。このパターンでは、AWS CodeCommit を使用してコードリポジトリを作成して複製します。次に、プロジェクトを作成し、AWS CodeBuild を使用してソースコードを設定します。最後に、リポジトリにコミットされる AMI を作成します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起動するための Amazon Linux AMI
- 「HashiCorp パッカー」 0.12.3 以降
- Amazon CloudWatch Events (オプション)
- Amazon CloudWatch Logs (オプション)

アーキテクチャ

次の図表には、このパターンのアーキテクチャを使用して AMI の作成を自動化するアプリケーションコードの例を示しています。

この図表は、次のワークフローを示しています：

1. 開発者はコードの変更をプライベート CodeCommit Git リポジトリにコミットします。次に、CodePipeline は CodeBuild を使用して、ビルドを開始し、Amazon Simple Storage Service

(Amazon S3) バケットに、デプロイする準備ができている新しい「アーティファクト」を追加します。

2. CodeBuild は Packer を使用して JSON テンプレートに基づいて AMI をバンドルしてパッケージ化します。有効にする場合、CloudWatch Events はソースコードに変更が生じたときにパイプラインを自動的に開始できます。

テクノロジースタック

- CodeBuild
- CodeCommit
- CodePipeline
- CloudWatch Events (オプション)

ツール

- AWS CodeBuild はクラウドで動作する、完全マネージド型のビルドサービスです。CodeBuild はソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイできるアーティファクトを作成します。
- AWS CodeCommit – AWS CodeCommit は、 AWS クラウドでの Git リポジトリのプライベートな保存と管理を可能にするバージョン制御サービスです。CodeCommit により、お客様が独自のソース制御システムを管理、またはインフラストラクチャのスケーリングに関して心配する必要はなくなります。
- AWS CodePipeline は、コードをリリースするために必要な手順のモデル化、視覚化、および自動化ができる継続的なデリバリサービスです。
- 「HashiCorp パッカー」 – HashiCorp パッカーは、単一のソース構成から同一のマシンイメージの作成を自動化するためのオープンソースツールです。パッカーは軽量で、すべての主要オペレーティングシステムで動作し、複数のプラットフォーム用のマシンイメージを並行で作成します。

コード

このパターンには以下の添付ファイルが含まれます。

- `buildspec.yml` – このファイルは、CodeBuild を使用してデプロイ用のアーティファクトをビルドおよび作成します。

- `amazon-linux_packer-template.json` – このファイルは、パッカーを使用して Amazon Linux AMI を作成します。

エピック

コードリポジトリをセットアップ

タスク	説明	必要なスキル
リポジトリを作成します。	CodeCommit リポジトリを作成します。	AWS システム管理者
リポジトリをクローン作成します。	CodeCommit リポジトリのクローンを作成してリポジトリに接続する。	アプリ開発者
ソースコードをリモートリポジトリにプッシュします。	1. コミットを作成して、`buildspec.yml` と `amazon-linux_packer-template.json` ファイルをローカルリポジトリに追加します。 2. ローカルリポジトリからリモート CodeCommit リポジトリにコミットをプッシュします。	アプリ開発者

アプリケーションの CodeBuild プロジェクトを作成

タスク	説明	必要なスキル
ビルドプロジェクトを作成します。	1. AWS マネジメントコンソールにサインインし、AWS CodeBuild コンソールを開いて、ビルドプロジェクトを作成]を選択します。	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
	2. プロジェクト名 にプロジェクトの名前を入力します。 3. [ソースプロバイダ] で、AWS CodeCommitを選択します。 4. リポジトリで、コードパイプラインを構築するレポジトリを選択します。 5. [環境イメージ] で、[マネージド型イメージ] または [カスタムイメージ] を選択します。 6. [Operating system] で、[Ubuntu] を選択します。 7. [ランタイム] で、標準を選択します。 8. [イメージ] で、[aws/codebuild/standard:4.0] を選択します。 9. [イメージのバージョン] で、[このランタイムバージョンには常に最新のイメージを使用] を選択します。 10[環境タイプ] は、[Linux] を選択します。 11.特権チェックボックスを選択します。 12.サービスロールには、新規サービスロール、または既存のサービスロールを選択します。	

タスク	説明	必要なスキル
	13.[ビルド仕様] で、ビルド仕様ファイルの使用またはビルドコマンドの挿入を選択します。 14.(オプション)アーティファクトセクションのタイプでは、アーティファクトなしを選択します。 15.(推奨) ビルド出力ログを CloudWatch ログにアップロードするには、CloudWatch ログを選択します。 16.(オプション) ビルド出力ログを Amazon S3 にアップロードするには、S3 logs チェックボックスを選択します。 17.Create build project (ビルドプロジェクトの作成)を選択します。	

パイプラインのセットアップ

タスク	説明	必要なスキル
パイプライン名	1. AWS マネジメントコンソールにサインインし、「AWS CodePipeline コンソール」を開いて、パイプラインの作成を選択します。	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
	2. パイプライン名 に、パイプラインの名前を入力します。 3. サービスロール で、新規サービスロール、または既存のサービスロールを選択します。 4. ロール名 に、ロールの名前を入力します。 5. Amazon S3 がバケットを作成し、そのバケットにアーティファクトを保存するようにする場合、詳細設定セクションでアーティファクトストア にデフォルトの位置 を選択します。 カスタムの場所で、既存の S3 バケットを使用を選択します。次へをクリックします。 6. ソースプロバイダ で、AWS CodeCommit を選択します。 7. リポジトリ名では、以前にクローンされたリポジトリを選択します。ブランチ名には、ソースコードブランチを選択します。 8. 変更検出オプションでは、Amazon CloudWatch Events (推奨)を選択してパイプラインを開始するか、AWS CodePipelineを選択して定期的に変更を確認	

タスク	説明	必要なスキル
	します。[Next (次へ)] を選択します。 9. [ビルドプロバイダ] で、[AWS CodeBuild] を選択します。 10.プロジェクト名には、アプリケーションの CodeBuild プロジェクトの作成エピックで作成したビルドプロジェクトを選択します。 11.を選択し、次にビルドオプションを選択します。 12.デプロイステージをスキップを選択します。 13.パイプラインの作成 を選択します。	

関連リソース

- AWS CodeCommit リポジトリ内のファイルの操作
- ビルドプロジェクトを操作する
- CodePipeline でのパイプラインの使用

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

CodePipeline を使用してパイプラインを作成し、アーティファクトの更新をオンプレミスの EC2 インスタンスにデプロイ

作成者: Akash Kumar (AWS) と Sandeep Reddy Jogammagari (AWS)

概要

このパターンでは、コードサンプルとステップを提供して、Amazon Web Services (AWS) クラウドでパイプラインを作成し、更新されたアーティファクト をAWS CodePipeline のオンプレミスの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにデプロイします。このパターンでは、継続的インテグレーションの手法に基づいています。この手法では、Git ベースのバージョン管理システムを使用してコードの構築とテストを自動化します。このパターンでは、AWS CodeCommit を使用してコードリポジトリを作成して複製します。次に、プロジェクトを作成し、AWS CodeBuild を使用してソースコードを設定します。最後に、アプリケーションを作成し、AWS CodeDeploy を使用してオンプレミス EC2 インスタンスのターゲット環境を設定します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- デプロイ時に EC2 インスタンスを識別するためのユーザー定義タグ
- EC2 インスタンスにインストールされた、CodeDeploy エージェント
- EC2 インスタンスにインストールされた、必要なランタイムソフトウェア
- Java 開発キット用のAmazon Corretto 8
- インストールされたApache Tomcat ウェブサーバー
- Amazon CloudWatch Events (オプション)
- ウェブサーバーにログインするキーペア (オプション)
- ウェブアプリケーションの Apache Maven アプリケーションプロジェクト

アーキテクチャ

次の図表は、このパターンのアーキテクチャを使用してオンプレミスの EC2 インスタンスにデプロイされる Java ウェブアプリケーションの例を示しています。

この図表は、次のワークフローを示しています：

1. 開発者はコードの変更をプライベート CodeCommit Git リポジトリにコミットします。
2. CodePipeline は CodeBuild を使用してビルドを開始し、Amazon Simple Storage Service (Amazon S3) バケットでデプロイする準備ができている新しいアーティファクトを追加します。
3. CodePipeline は CodeDeploy エージェントを使用して、デプロイアーティファクトの変更に必要な依存関係をプレインストールします。
4. CodePipeline は CodeDeploy エージェントを使用して、 S3 バケットのアーティファクトをターゲットの EC2 インスタンスにデプロイします。有効にする場合、ソースコードに変更が生じたときに、CloudWatch イベントはパイプラインを自動的に開始できます。

テクノロジースタック

- CodeBuild
- CodeCommit
- CodeDeploy
- CodePipeline
- CloudWatch Events (オプション)

ツール

- AWS CodeBuildは完全マネージド型のビルドサービスで、ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成をサポートします。CodeBuild はソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイできるアーティファクトを作成します。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。

コード

このパターンには、以下の添付ファイルが含まれます。

- `buildspec.yml` – このファイルには、デプロイ用のアーティファクトをビルドおよび作成するために CodeBuild が必要とするアクションを指定します。
- `appspec.yml` – このファイルには、CodeDeploy がアプリケーションを作成し、オンプレミス EC2 インスタンスのターゲット環境を設定するために必要なアクションを指定します。
- `install_dependencies.sh` – このファイルは Apache Tomcat ウェブサーバーの依存関係をインストールします。
- `start_server.sh` – このファイルにより Apache Tomcat ウェブサーバーが起動します。
- `stop_server.sh` – このファイルは Apache Tomcat ウェブサーバーを停止します。

エピック

コードリポジトリをセットアップ

タスク	説明	必要なスキル
リポジトリを作成します。	CodeCommit リポジトリを作成します。	AWS システム管理者
リポジトリをクローン作成します。	CodeCommit リポジトリのクローンを作成してリポジトリに接続する。	アプリ開発者
ソースコードをリモートリポジトリにプッシュします。	1. コミットを作成して、`buildspec.yml` と `appspec.yml` ファイルをローカルリポジトリに追加します。 2. ローカルリポジトリからリモート CodeCommit リポジトリにコミットをプッシュします。	アプリ開発者

アプリケーションの CodeBuild プロジェクトを作成

タスク	説明	必要なスキル
ビルドプロジェクトを作成します。	1. AWS マネジメントコンソールにサインインし、AWS CodeBuild コンソールを開き、ビルドプロジェクトの作成を選択します。 2. プロジェクト名 にプロジェクトの名前を入力します。 3. [ソースプロバイダ] で、AWS CodeCommitを選択します。 4. リポジトリで、コードパイプラインを構築するレポジトリを選択します。 5. 環境イメージで、マネージド型イメージまたはカスタムイメージを選択します。 6. オペレーティングシステムで、Amazon Linux 2を選択します。 **Note** Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。	AWS 管理者、アプリ開発者

タスク	説明	必要なスキル
	7. ランタイム] で、標準を選択します。 8. イメージで、aws/codebuild/amazonlinux2-aarch64-standard:2.0 を選択します。 9. イメージのバージョンで、このランタイムバージョンには常に最新のイメージを使用 を選択します。 10.サービスロールには、新規サービスロール、または既存のサービスロールを選択します。 = 11.ビルド仕様 で、ビルド仕様ファイルの使用またはビルドコマンドの挿入を選択します。 12.(オプション) アーティファクトの追加.を選択して、アーティファクトを設定します。 13.(オプション) ビルド出力ログを Amazon CloudWatch にアップロードするには、CloudWatch ログを選択します。 14.Create build project (ビルドプロジェクトの作成)を選択します。	

オンプレミス EC2 インスタンスのアーティファクトデプロイを設定

タスク	説明	必要なスキル
アプリケーションの作成	1. AWS マネジメントコンソールにサインインし、AWS CodeDeploy コンソールを開き、アプリケーションの作成を選択します。 2. アプリケーション名 に、アプリケーションの名前を入力します。 3. コンピューティングプラットフォーム で EC2/オンプレミス を選択します。 4. アプリケーションの作成 を選択し、デプロイグループの作成を選択します。 5. デプロイグループ名 に名前を入力します。 6. **Note** CodeDeployのサービスロールを作成する：サービスロールには、CodeDeployにターゲット環境へのアクセスを許可するアクセス許可が必要です。 7. サービスロール には、ステップ 6 で作成したサービスロールを選択します。 8. [デプロイタイプ] には、ビジネス要件に基づいてイン	AWS システム管理者、アプリ開発者

タスク	説明	必要なスキル
	プレースまたはブルー/グリーンを選択します。 9. 環境設定では、ビジネス要件を満たすオプションを選択します。 10.(オプション) Amazon EC2 コンソールでロードバランサーのターゲットグループを個別に作成し 、AWS CodeDeploy コンソールのデプロイグループの作成ページに戻って、ロードバランサーとターゲットグループを選択します。 11.デプロイグループの作成 を選択します。	

パイプラインのセットアップ

タスク	説明	必要なスキル
パイプラインを作成します。	1. AWS マネジメントコンソールにサインインし、AWS CodePipeline コンソールを開き、パイプラインの作成を選択します。 2. パイプライン名 に、パイプラインの名前を入力します。 3. サービスロール で、新規サービスロール、または既存のサービスロールを選択します。	AWS システム管理者、アプリ開発者

タスク	説明	必要なスキル
	4. ロール名 に、ロールの名前を入力します。 5. Amazon S3 がバケットを作成し、そのバケットにアーティファクトを保存するようにする場合、詳細設定セクションでアーティファクトストア にデフォルトの位置 を選択します。 カスタムの場所で、既存の S3 バケットを使用を選択します。次へをクリックします。 6. ソースプロバイダ で、AWS CodeCommit を選択します。 7. リポジトリ名では、以前にクローンされたリポジトリを選択します。ブランチ名では、ソースコードブランチを選択します。 8. 変更検出オプション で、Amazon CloudWatch Events (推奨) または AWS CodePipeline を選択します。次へ をクリックします。 9. ビルドプロバイダ で、AWS CodeBuildを選択します。 10.プロジェクト名では、このパターンのアプリケーション用の CodeBuild プロジェクトの作成セクションで作	

タスク	説明	必要なスキル
	成したビルドプロジェクトを選択します。 11次へを選択し、次にビルドオプションを選択します。 12.プロバイダをデプロイするで、AWS CodeDeploy を選択します。 13.アプリケーション名とデプロイグループを選択し、次へを選択します。 14パイプラインの作成 を選択します。	

関連リソース

- AWS CodeCommit リポジトリ内のファイルの操作
- ビルドプロジェクトを操作する
- CodeDeploy 内でアプリケーションを用いて作業をする
- CodePipeline でのパイプラインの使用

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Java および Python プロジェクト用の動的 CI パイプラインを自動的に作成

アロマル・ラージ・ジャヤラジャン（AWS）、アマルナス・レディ（AWS）、マヘシュ・ラグナンダナン（AWS）、ビジェッシュ・ビジャヤクマラン・ナイール（AWS）によって作成された

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、AWS 開発者ツールを使用して Java および Python プロジェクトの動的継続的インテグレーション (CI) パイプラインを自動的に作成する方法を示しています。

テクノロジースタックが多様化し、開発活動が増えるにつれて、組織全体で一貫性のある CI パイプラインを作成して維持することが難しくなる可能性があります。AWS Step Functions のプロセスを自動化することで、CI パイプラインの使用法とアプローチに一貫性を持たせることができます。

動的 CI パイプラインの作成を自動化するために、このパターンでは以下の変数入力を使用します。

- プログラミング言語 (Java または Python のみ)
- パイプライン名
- 必要なパイプラインステージ

> **Note**
>
> Step Functions は、複数の AWS サービスを使用してパイプラインの作成を調整します。このソリューションで使用されている AWS サービスの詳細については、このパターンの「ツール」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- このソリューションがデプロイされている同じ AWS リージョンにある Amazon S3 バケット
- このソリューションに必要なリソースを作成するために必要な AWS CloudFormation アクセス許可を持つ AWS Identity and Access Management (IAM)「プリンシパル」

制約事項

- このパターンは Java プロジェクトと Python プロジェクトのみをサポートします。
- このパターンでプロビジョニングされる IAM ロールは、最小特権の原則に従います。IAM ロールの権限は、CI パイプラインが作成する必要のある特定のリソースに基づいて更新する必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- IAM
- Amazon Simple Storage Service (Amazon S3)
- AWS Systems Manager
- AWS Step Functions
- AWS Lambda
- Amazon DynamoDB

ターゲットアーキテクチャ

次の図は、AWS 開発者ツールを使用して Java および Python プロジェクトの動的 CI パイプラインを自動的に作成するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. AWS ユーザーは CI パイプラインを作成するための入力パラメータを JSON 形式で提供します。この入力により、AWS 開発者ツールを使用して CI パイプラインを作成する Step Functions ワークフロー (ステートマシン) が開始されます。
2. Lambda 関数は Amazon S3 バケットに保存されている「インプット-レファレンス」という名前のフォルダを読み取り、「buildspec.yml」ファイルを生成します。この生成されたファイルは CI

パイプラインのステージを定義し、パラメータ参照を格納しているのと同じ Amazon S3 バケットに保存されます。

3. Step Functions は CI パイプライン作成ワークフローの依存関係に変更がないか確認し、必要に応じて依存関係スタックを更新します。
4. Step Functions は、CodeCommit リポジトリ、CodeBuild プロジェクト、CodePipeline パイプラインを含む CI パイプラインリソースを CloudFormation スタックに作成します。
5. CloudFormation スタックは、選択したテクノロジースタック (Java または Python) のサンプルソースコードと 「buildspec.yml」ファイルを CodeCommit リポジトリにコピーします。
6. CI パイプラインランタイムの詳細は DynamoDB テーブルに保存されます。

自動化とスケール

- このパターンは 1 つの開発環境でのみ使用できます。複数の開発環境で使用するには、設定の変更が必要です。
- 複数の CloudFormation スタックのサポートを追加するには、追加の CloudFormation テンプレートを作成できます。詳細については、CloudFormation ドキュメントの「AWS CloudFormation 入門」を参照してください。

ツール

ツール

- AWS Step Functionsは、AWS Lambda関数と他のAWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS CodeBuild はフルマネージドの構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Systems Manager Parameter Store は、設定データ管理とシークレット管理用の安全な階層型ストレージを提供します。

コード

このパターンのコードは、GitHub 内の「automated-ci-pipeline-creation」リポジトリで利用できます。リポジトリには、このパターンで概説されているターゲットアーキテクチャの作成に必要な CloudFormation テンプレートが含まれています。

ベストプラクティス

- トークンやパスワードなどの認証情報 (シークレット) を CloudFormation テンプレートや Step Functions アクション設定に直接入力しないでください。その場合、情報は DynamoDB ログに表示されます。代わりに、AWS Secrets Manager を使用してシークレットを設定し、保存してください。次に、必要に応じて CloudFormation テンプレートとStep Functions アクション設定内の Secrets Manager に保存されているシークレットを参照します。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager とは」を参照してください。
- Amazon S3 に保存した CodePipeline アーティファクトのサーバー側の暗号化を設定する 詳細については、CodePipeline ドキュメントの「CodePipeline 用に Amazon S3 に保存したアーティファクトのサーバー側の暗号化を設定する」を参照してください。
- IAM ロールを設定する際には、最小特権アクセス許可を適用します。詳細については、IAM ドキュメントの最小特権アクセス許可を適用するを参照してください。
- Amazon S3 バケットがパブリックにアクセスできないことを確認します。詳細については、Amazon S3 ドキュメントの「S3 バケットのブロックパブリックアクセス設定を行う」を参照してください。
- Amazon S3 バケットのバージョニングを必ず有効にしてください。詳細については、「Amazon S3 バケットでのバージョニングの使用」を参照してください。

- IAM ポリシーを設定するときは IAM アクセスアナライザーを使用してください。このツールは安全で機能的なIAMポリシーを作成するのに役立つ実用的な推奨事項を提供します。詳細については、IAM ドキュメントの「AWS アイデンティティとアクセスマネジメントアクセスアナライザの使用」を参照してください。
- 可能な場合は、IAM ポリシーを設定する際に特定のアクセス条件を定義してください。
- モニタリングと監査の目的で Amazon CloudWatch のロギングを有効化します。詳細については、Cloudwatch ドキュメントの「Amazon CloudWatch Logsとは」を参照してください。

エピック

AWS の前提条件の設定

タスク	説明	必要なスキル
Amazon S3 バケットを作成する。	Amazon S3 バケットを作成して (または既存のバケットを使用して)、ソリューションに必要な CloudFormation テンプレート、ソースコード、および入力ファイルを保存します。 詳細については、Amazon S3 ドキュメントの「ステップ 1：最初の S3 バケットの作成」を参照してください。 **Note** Amazon S3 バケットは、ソリューションをデプロイするのと同じ AWS リージョンに存在する必要があります。	AWS DevOps

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	ターミナルウィンドウで次のコマンドを実行して、GitHub「自動 CI パイプライン作成」リポジトリをクローンします。 `git clone https://github.com/aws-samples/automated-ci-pipeline-creation.git` 詳細については、Github ドキュメントの「リポジトリのクローン」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
クローンした GitHub リポジトリから Amazon S3 バケットにソリューションテンプレートフォルダをアップロードします。	クローンした「ソリューション-テンプレート」フォルダから内容をコピーし、作成した Amazon S3 バケットにアップロードします。 詳細については、Amazon S3 ドキュメントの「オブジェクトのアップロード」を参照してください。 **Note** 必ず Solution-Templates フォルダの内容のみをアップロードしてください。Amazon S3 バケットのルートレベルでのみファイルをアップロードできます。	AWS DevOps

解決策をデプロイする

タスク	説明	必要なスキル
クローンされた GitHub リポジトリ内の template.yml ファイルを使用して、ソリューションをデプロイするための CloudFormation スタックを作成します。	1. AWS マネジメントコンソールにサインインしてから、AWS CloudFormation コンソールを開きます。 2. [スタックの作成] を選択してください。ドロップダウンリストが表示されます。	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	3. ドロップダウンリストで「新規リソース追加 (標準)」を選択します。[スタックの作成] ページを開きます。 4. [テンプレートの指定] セクションで、[テンプレートファイルのアップロード] のチェックマークを選択します。 5. [Choose file] を選択します。次に、クローンした GitHub リポジトリのルートフォルダに移動し、「template.yml」ファイルを選択します。次に、[開く] を選択します。 6. [Next (次へ)] を選択します。[スタック詳細の指定] ページが表示されます。 7. [パラメータ] セクションで、以下のパラメータを指定します。 • 「s3TemplateBucketName」には、前に作成しておいた Amazon S3 バケットの名前を入力します。このバケットには、このソリューションのソースコードとリファレンスが格納されています。バケット名のパラメーターが小文字であることを確認します。	

タスク	説明	必要なスキル
	• 「DynamoDBTable」には、CloudFormation スタックが作成する DynamoDB テーブルの名前を入力します。 • 「StateMachineName」には、CloudFormation スタックが作成するStep Functions ステートマシンの名前を入力します。 8. [Next (次へ)] を選択します。スタックオプションの設定 ページが開きます。 9. [スタックオプションの設定] ページで、[次へ] を選択します。デフォルト値はいずれも変更しないでください。「レビュー」ページが開きます。 10スタックの作成設定を確認します。次に、「スタックの作成] を選択してスタックを起動します。 Note スタックの作成中に、スタックページにステータスが CREATE_IN_PROGRESS と表示されます。このパターンの残りのステップを	

タスク	説明	必要なスキル
	完了する前に、スタックのステータスが「作成_完了」に変わるのを必ず待ってください。	

セットアップをテストする

タスク	説明	必要なスキル
作成した関数を実行します。	1. AWS マネジメントコンソールにサインインし、「手順関数コンソール」を開きます。 2. 作成した関数が表示されます。 3. [実行のスタート] を選択します。次に、ワークフローの入力値を JSON 形式で入力します (以下の入力例を参照)。 4. [実行のスタート] を選択します。 (JSON 形式) を使用してエクスポートおよびダウンロードするには： `{ "details": { "tech_stack": "Name of the Tech Stack (python/java)",`	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	<pre>"project_name": "Name of the Project that you want to create with", "pre_build": "Choose the step if it required in the buildspec.yml file i.e., yes/no", "build": "Choose the step if it required in the buildspec.yml file i.e., yes/no", "post_build": "Choose the step if it required in the buildspec.yml file i.e., yes/no", "reports": "Choose the step if it required in the buildspec.yml file i.e., yes/no", } }</pre> Java JSON 入力の例 <pre>{ "details": { "tech_stack": "java", "project_name": "pipeline-java-pjt", "pre_build": "yes", "build": "yes", "post_build": "yes", "reports": "yes" } }</pre>	

タスク	説明	必要なスキル
	Python JSON 入力の例 `{ "details": { "tech_stack": "python", "project_name": "pipeline-python-pjt", "pre_build": "yes", "build": "yes", "post_build": "yes", "reports": "yes" } }`	
CI パイプラインの CodeCommit リポジトリが作成されたことを確認します。	1. AWS マネジメントコンソールにサインインし、CodeCommit コンソールを開きます。 2. 「リポジトリ」ページで、作成した CodeCommit リポジトリの名前がリポジトリのリストに表示されていることを確認します。x リポジトリの名前には、「Pipeline-java-PJT-Repo」という名前が付加されます。 3. CodeCommit リポジトリを開き、サンプルソースコードと「buildspec.yml」ファイルがメインブランチにプッシュされていることを確認します。	AWS DevOps

タスク	説明	必要なスキル
CodeBuild プロジェクトリソースを確認してください。	1. AWS マネジメントコンソールにサインインし、CodeBuild コンソールを開きます。 2. 「ビルドプロジェクト」ページで、作成した CodeBuild プロジェクトの名前がプロジェクトのリストに表示されていることを確認します。プロジェクトの名前には「Pipeline-java-PJT-Build」を追加します。 3. CodeBuild プロジェクトの名前を選択してプロジェクトを開きます。次に、以下の設定を確認して検証します。 • プロジェクトの設定 • ソース • 環境 • Buildspec • Batch 構成 • アーティファクト	AWS DevOps

タスク	説明	必要なスキル
CodePipeline のステージを検証します。	1. AWS マネジメントコンソールにサインインし、CodePPipeline コンソールを開きます。 2. 「パイプライン」ページで、作成したパイプラインの名前がパイプラインのリストに表示されていることを確認します。パイプラインの名前には「Pipeline-java-Pjt-Pipeline」を追加します。 3. パイプラインの名前を選択してパイプラインを開きます。次に、「コミット」や「デプロイ」など、パイプラインの各段階を確認して検証します。	AWS DevOps
CI パイプラインが正常に実行されたことを確認します。	1. 「CodePipeline コンソール」の「パイプライン」ページで、パイプラインの名前を選択してパイプラインのステータスを表示します。 2. パイプラインの各ステージのステータスが「Succeeded」であることを確認します。	AWS DevOps

リソースのクリーンアップ

タスク	説明	必要なスキル
CloudFormation のリソーススタックを削除します。	CloudFormation 内の CI パイプラインのリソーススタックを削除します。 詳細については、CloudFormationドキュメントの「AWS CloudFormation でのスタックの削除」を参照してください。 **Note** <project_name>-stack という名前のスタックを削除してください。	AWS DevOps
Amazon S3 とCloudFormation の CI パイプラインの依存関係を削除します。	1. 「DeplomentArtifactBucket」という名前の Amazon S3 バケットを空にします。詳細については、Amazon S3 ドキュメントの「バケットを空にする」を参照してください。 2. CloudFormation 内の CI パイプラインの依存関係スタックを削除します。詳細については、CloudFormationドキュメントの「AWS CloudFormation でのスタックの削除」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
	Note pipelinepipeline-creation-dependencies-stackという名前のスタックを削除してください。	
Amazon S3 バケットを削除します。	このパターンの「前提条件の設定」セクションで作成した Amazon S3 バケットを削除します。このバケットには、このソリューションのテンプレートが保存されています。 詳細については、Amazon S3 ドキュメントの「バケットの削除」を参照してください。	AWS DevOps

関連リソース

- Lambda を使用する Step Functions ステートマシンの作成 (AWS Step Functions ドキュメント)
- AWS Step Functions WorkFlow Studio (AWS Step Functions ドキュメント)
- DevOps と AWS
- AWS CloudFormation の仕組み (AWS CloudFormation ドキュメント)
- AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy、AWS CodePipeline を使用して CI/CD を完了する (AWS ブログ記事)
- IAM および AWS STS のクォータ、名前の要件、文字数の制限 (IAM ドキュメント)

Terraform を使用して CloudWatch Synthetics Canary をデプロイする

作成者: Dhrubajyoti Mukherjee (AWS)、Jean-Francois Landreau (AWS)

概要

顧客の観点からシステムの状態を検証し、顧客が接続できることを確認することは重要です。顧客がエンドポイントを頻繁に呼び出さない場合、これはさらに困難になります。Amazon CloudWatch Synthetics は、パブリックエンドポイントとプライベートエンドポイントの両方をテストできる Canary の作成をサポートしています。Canary を使用すれば、使用中でなくてもシステムの状態を知ることができます。これらの Canary は Node.js Puppeteer スクリプトまたは Python Selenium スクリプトのどちらかです。

このパターンでは、HashiCorp Terraform を使用してプライベートエンドポイントをテストする Canary をデプロイする方法を説明します。URL が `200-OK` に返されるかどうかをテストする Puppeteer スクリプトが組み込まれています。その後、Terraform スクリプトを、プライベートエンドポイントをデプロイするスクリプトと統合できます。パブリックエンドポイントを監視するようにソリューションを変更することもできます。

前提条件と制限

前提条件

- 仮想プライベートクラウド (VPC) とプライベートサブネットを持つアクティブなAmazon Web Services (AWS) アカウント
- プライベートサブネットからアクセスできるエンドポイントの URL
- Terraform をインストールしたデプロイメント環境

制限

現在のソリューションは、次の CloudWatch Synthetics ランタイムバージョンで動作します。

- syn-nodejs-puppeteer-3.4
- syn-nodejs-puppeteer-3.5
- syn-nodejs-puppeteer-3.6
- syn-nodejs-puppeteer-3.7

新しいランタイムバージョンがリリースされると、現在のソリューションの更新が必要な場合があります。また、セキュリティアップデートに対応できるようにソリューションを変更する必要もあります。

製品バージョン

- Terraform 1.3.0

アーキテクチャ

Amazon CloudWatch Synthetics は、CloudWatch、Lambda、および Amazon Simple Storage Service (Amazon S3) をベースにしています。Amazon CloudWatch には、Canary を作成するウィザードと、Canary の実行状況を表示するダッシュボードが用意されています。Lambda 関数はスクリプトを実行します。Amazon S3 は、Canary 実行のログとスクリーンショットを保存します。

このパターンは、ターゲットサブネットにデプロイされた Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを介してプライベートエンドポイントをシミュレートします。Lambda 関数には、プライベートエンドポイントがデプロイされている VPC 内の伸縮自在なネットワークインターフェイスが必要です。

図に示す内容は以下のとおりです。

1. Synthetics canary が Canary Lambda 関数を開始します。
2. Canary Lambda 関数が Elastic Network Interface に接続します。
3. Canary Lambda 関数はエンドポイントのステータスを監視します。
4. Synthetics Canary は、実行データを S3 バケットと CloudWatch メトリックスにプッシュします。
5. メトリックスに基づいて CloudWatch アラームが開始されます。
6. CloudWatch アラームが Amazon Simple Notification Service (Amazon SNS) トピックを開始します。

ツール

AWS サービス

- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。このパターンは VPC エンドポイントとエラスティックネットワークインターフェイスを使用します。

その他のサービス

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するのに役立つオープンソースの Infrastructure as Code (IaC) ツールです。このパターンでは、Terraform を使用してインフラストラクチャをデプロイします。
- Puppeteer は Node.js ライブラリです。CloudWatch Synthetics ランタイムは Puppeteer フレームワークを使用しています。

コード

このソリューションは、GitHub の cloud watch-synthetics-canary-terraform リポジトリで利用できます。詳細については、「追加情報」セクションを参照してください。

エピック

プライベート URL を監視するソリューションを実装してください。

タスク	説明	必要なスキル
プライベート URL を監視するための要件を収集する。	ドメイン、パラメータ、ヘッダーなどの URL 定義をすべ	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル
	て収集します。Amazon S3 や Amazon CloudWatch とプライベートに通信するには、VPC エンドポイントを使用してください。VPC とサブネットがエンドポイントにどのようにアクセスできるかに注意してください。Canary 実行の頻度を考えてみましょう。	

タスク	説明	必要なスキル
プライベート URL を監視するように既存のソリューションを変更する。	`terraform.tfvars` ファイルの変更: • `name` – Canary の名前。 • `runtime_version` – Canary のランタイムバージョン。syn-nodejs-puppeteer-3.7 の使用をお勧めします。 • `take_screenshot` – スクリーンショットを撮るべきかどうか。 • `api_hostname` – 監視対象のエンドポイントのホスト名。 • `api_path` – 監視対象のエンドポイントのパス。 • `vpc_id` – Canary Lambda 関数が使用する VPC ID。 • `subnet_ids` – Canary Lambda 関数によって使用されるサブネット ID。 • `frequency` – Canary の実行頻度 (分単位)。 • `alert_sns_topic` – CloudWatch アラーム通知の送信先として指定する SNS トピック。	クラウドアーキテクト

タスク	説明	必要なスキル

ソリューションをデプロイして運用する。	ソリューションを展開するには、次を実行します。
	1. 開発環境の `cloudwatch-synthetics-canary-terraform` ディレクトリから Terraform を初期化します。
	`terraform init`
	2. 変更を計画し、レビューします。
	`terraform plan`
	3. ソリューションをデプロイします。
	`terraform apply`
		クラウドアーキテクト、DevOps エンジニア

トラブルシューティング

問題	ソリューション
プロビジョニングされたリソースの削除が停止する。	Canary Lambda 関数、対応する elastic network interface、セキュリティグループをこの順序で手動で削除します。

関連リソース

- 模擬モニタリングの使用
- Monitor API Gateway endpoints with Amazon CloudWatch Synthetics (ブログ記事)

追加情報

リポジトリのアーティファクト

リポジトリのアーティファクトは、次の構造を持っています。

```
.
### README.md
### main.tf
### modules
#   ### canary
#   ### canary-infra
### terraform.tfvars
### tf.plan
### variable.tf
```

main.tf ファイルにはコアモジュールが含まれ、2 つのサブモジュールがデプロイされます。

- canary-infra は、Canary に必要なインフラストラクチャをデプロイします。
- canary は、Canary をデプロイします。

ソリューションの入力パラメータは terraform.tfvars ファイルにあります。次のコード例を使用して 1 つの Canary を作成できます。

```
module "canary" {
    source = "./modules/canary"
    name   = var.name
    runtime_version = var.runtime_version
    take_screenshot = var.take_screenshot
    api_hostname = var.api_hostname
    api_path = var.api_path
    reports-bucket = module.canary_infra.reports-bucket
    role = module.canary_infra.role
    security_group_id = module.canary_infra.security_group_id
    subnet_ids = var.subnet_ids
    frequency = var.frequency
    alert_sns_topic = var.alert_sns_topic
}
```

対応する.var ファイルは次のとおりです。

```
name   = "my-canary"
runtime_version = "syn-nodejs-puppeteer-3.7"
take_screenshot = false
api_hostname = "mydomain.internal"
api_path = "/path?param=value"
vpc_id = "vpc_id"
subnet_ids = ["subnet_id1"]
frequency = 5
alert_sns_topic = "arn:aws:sns:eu-central-1:111111111111:yyyyy"
```

ソリューションのクリーンアップ

開発環境でこれをテストする場合は、コストの発生を避けるためにソリューションをクリーンアップできます。

1. AWS マネジメントコンソールで、Amazon S3 コンソールに移動します。ソリューションが作成した Amazon S3 バケットを空にします。必要な場合は、必ずデータのバックアップを取ってください。
2. 開発環境の `cloudwatch-synthetics-canary-terraform` ディレクトリから `destroy` コマンドを実行します。

```
terraform destroy
```

Amazon ECS に Java マイクロサービス用の CI/CD パイプラインをデプロイ

作成者:Vijay Thompson (AWS) と Sankar Sangubotla (AWS)

概要

このパターンでは、AWS CodeBuild を使用して既存の Amazon Elastic Container Service (Amazon ECS) クラスターに Java マイクロサービスの継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインをデプロイする手順を説明します。開発者が変更をコミットすると、CI/CD パイプラインが開始され、CodeBuild でビルドプロセスが開始されます。ビルドが完了すると、アーティファクトは Amazon Elastic Container Registry (Amazon ECR) にプッシュされ、Amazon ECR からの最新のビルドが取得されて Amazon ECS サービスにプッシュされます。

前提条件と制限

前提条件

- Amazon ECS で実行されている既存の Java マイクロサービスアプリケーション
- AWS CodeBuild と AWS CodePipeline に精通していること

アーキテクチャ

ソーステクノロジースタック

- Amazon ECS で実行されている Java マイクロサービス
- Amazon ECR のコードリポジトリ
- AWS Fargate

ソースアーキテクチャ

ターゲットテクノロジースタック

- Amazon ECR
- Amazon ECS
- AWS Fargate

- CodePipeline
- CodeBuild

ターゲットアーキテクチャ

自動化とスケール

CodeBuild `buildspec.yml` ファイル:

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}')
  build:
    commands:
      - echo Build started on `date`
      - echo building the Jar file
      - mvn clean install
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:$BUILD_TAG .
      - docker tag $REPOSITORY_URI:$BUILD_TAG $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:$BUILD_TAG
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"%s","imageUri":"%s"}]' $DOCKER_CONTAINER_NAME
 $REPOSITORY_URI:$IMAGE_TAG > imagedefinitions.json
      - cat imagedefinitions.json
artifacts:
    files:
```

```
    - imagedefinitions.json
    - target/DockerDemo.jar
```

ツール

AWS サービス

- AWS CodeBuild はフルマネージドの構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。AWS CodeBuild では、スケーリングが継続的に行われ、複数のビルドが同時に実行されますので、ビルドがキューに残されません。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。AWS CodePipeline を GitHub などのサードパーティーサービスと統合することも、Amazon ECR などの AWS サービスを使用することもできます。
- 「Amazon Elastic Container Registry (Amazon ECR)」は、フルマネージド型のレジストリで、開発者は Docker コンテナイメージを簡単に保存、管理、デプロイできます。Amazon ECR は Amazon ECS と統合されているため、開発から本番環境までのワークフローを簡素化できます。Amazon ECR は、可用性が高くスケーラブルなアーキテクチャでイメージがホストされるため、アプリケーション用のコンテナを確実にデプロイできます。AWS Identity and Access Management (IAM) との統合により、各リポジトリのリソースレベルでの制御が可能になります。
- 「Amazon Elastic Container Service (Amazon ECS)」 は非常にスケーラブルかつ高性能なコンテナオーケストレーションサービスで、Docker コンテナに対応しており、コンテナ化されたアプリケーションを AWS で簡単に実行してスケーリングできるようになります。Amazon ECS では、独自のコンテナオーケストレーションソフトウェアをインストールして運用したり、仮想マシンのクラスターを管理およびスケーリングしたり、それらの仮想マシン上でコンテナをスケジュールしたりする必要がなくなります。
- 「AWS Fargate」 は Amazon ECS のコンピューティングエンジンで、これを使用するとサーバーまたはクラスターを管理する必要なくコンテナを実行できるようになります。AWS Fargate を使用すると、コンテナを実行するために仮想マシンのクラスターをプロビジョニング、設定、スケールする必要がありません。これにより、サーバータイプの選択、クラスターをスケールするタイミングの決定、クラスターのパッキングの最適化を行う必要がなくなります。

その他のツール

- 「Docker」は、コンテナ と呼び出すパッケージでアプリケーションをビルド、テスト、配信できるプラットフォームです。

- 「Git」は、ソフトウェア開発中のソースコードの変更を追跡するための分散型バージョン管理システムです。プログラマー間の作業を調整するために設計されていますが、どのファイルセットの変更も追跡できます。その目標には、スピード、データインテグリティ、分散型非線形ワークフローのサポートなどがあります。

エピック

AWS CodeBuild でのビルドプロジェクトのセットアップ

タスク	説明	必要なスキル
CodeBuild ビルドプロジェクトを作成します。	「AWS CodeBuild コンソールで」、ビルドプロジェクトを作成し、その名前を指定します。	アプリ開発者、AWS システム管理者
ソースを選択します。	このパターンはコードリポジトリに Git を使用するため、使用可能なオプションのリストから GitHub を選択してください。パブリックのリポジトリかGitHub アカウントかを選択します。	アプリ開発者、AWS システム管理者
リポジトリを選択します。	コードをビルドするリポジトリを選択します。	アプリ開発者、AWS システム管理者
環境を選択します。	管理対象イメージのリストから選択するか、Docker を使用してカスタムイメージを選択できます。このパターンでは以下のマネージドイメージを使用します。 • **Note** Amazon Linux 2 (: Amazon Linux 2 の	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
	サポートは終了間近です。 詳細については、「Amazon Linux 2 のFAQs」を参照してください。 ） • ランタイム標準 • イメージバージョン 1.0	
サービスロールを選択します。	サービスロールを作成するか、既存のロールのリストから選択できます。	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
環境変数を追加します。	追加設定セクションで、以下の環境変数を設定します。 • デフォルトの AWS リージョンの AWS_DEFAULT_REGION • ユーザーアカウント番号の AWS_ACCOUNT_ID • Amazon ECR プライベートリポジトリの IMAGE_REPO • ビルドのバージョンの BUILD_TAG (この変数の値は最新のビルドです) • DOCKER_CONTAINER_NAME はタスク内のコンテナの名前です。 これらの変数は `buildspec.yml` ファイル内のプレースホルダーであり、それぞれの値に置き換えられます。	アプリ開発者、AWS システム管理者
buildspec ファイルを作成します。	このパターンで提供される設定と同じ `pom.xml` 場所に `buildspec.yml` ファイルを作成して追加することも、オンラインの buildspec エディタを使用して設定を追加することもできます。表示される手順に従って、環境変数を適切な値に設定します。	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
アーティファクト用にプロジェクトを設定します。	(オプション) 必要に応じて、ビルドプロジェクトにアーティファクトを設定します。	アプリ開発者、AWS システム管理者
Amazon CloudWatch Logs を設定します。	(オプション) 必要に応じて、ビルドプロジェクト用に Amazon CloudWatch Logs を設定します。このステップはオプションですが推奨されます。	アプリ開発者、AWS システム管理者
Amazon S3 ログを設定します。	(オプション) ログを保存する場合は、ビルドプロジェクト用に Amazon Simple Storage Service (Amazon S3) ログを設定します。	アプリ開発者、AWS システム管理者

AWS CodePipeline でパイプラインを設定します。

タスク	説明	必要なスキル
パイプラインを作成する	「AWS CodePipeline コンソールで」、パイプラインを作成し、その名前を指定します。パイプラインの作成に関する詳細については、「AWS CodePipeline のドキュメント」を参照してください。	アプリ開発者、AWS システム管理者
サービスロールを選択します。	サービスロールを作成するか、既存のサービスロールのリストから選択します。サービスロールを作成する場合は、ロールの名前を指定し、CodePipeline のオプション	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
	を選択してロールを作成します。	
アーティファクトストアを選択します。	詳細設定で、Amazon S3 にバケットを作成してアーティファクトを保存させたい場合は、アーティファクトストアのデフォルトの場所を使用します。または、カスタムの場所を選択し、既存のバケットを指定します。暗号化キーを使用してアーティファクトを暗号化することもできます。	アプリ開発者、AWS システム管理者
ソースプロバイダを指定します。	ソースプロバイダーに対して、GitHub (バージョン 2)を選択します。	アプリ開発者、AWS システム管理者
コードのリポジトリとブランチを選択します。	ログインしていない場合は、GitHub に接続するための接続の詳細を入力し、リポジトリ名とブランチ名を選択します。	アプリ開発者、AWS システム管理者
変更検出オプション。	ソースコードの変更時にパイプラインを開始するを選択し、次のページに移動します。	アプリ開発者、AWS システム管理者

タスク	説明	必要なスキル
ビルドプロバイダを選択します。	ビルドプロバイダーで AWS CodeBuild を選択し、ビルドプロジェクトの AWS リージョンとプロジェクト名の詳細を指定します。 ビルドタイプにはシングルビルドを選択します。	アプリ開発者、AWS システム管理者
デプロイプロバイダを選択します。	デプロイプロバイダに対して、[Amazon ECS] を選択します。クラスター名、サービス名、イメージ定義ファイル (ある場合)、デプロイのタイムアウト値 (必要な場合) を選択します。パイプラインの作成 を選択します。	アプリ開発者、AWS システム管理者

関連リソース

- 「AWS ECS のドキュメント」
- 「AWS ECR のドキュメント」
- 「AWS CodeBuild ドキュメント」
- 「AWS CodePipeline ドキュメント」
- 「Amazon ECR をソースとして使用してコンテナイメージの継続的デリバリーパイプラインを構築」 (ブログ投稿)

ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation

作成者: Anand Bukkapatnam Tirumala (AWS)

概要

このパターンは、チャットアプリケーションで Amazon Q Developer を使用して、SonarQube を通じて報告された静的アプリケーションセキュリティテスト (SAST) スキャン障害の管理を合理化する包括的なソリューションを示しています。この革新的なアプローチは、カスタムアクションと通知を会話型インターフェイスに統合し、開発チーム内での効率的なコラボレーションと意思決定プロセスを可能にします。

今日のペースの速いソフトウェア開発環境では、SAST スキャン結果を効率的に管理することは、コードの品質とセキュリティを維持する上で不可欠です。ただし、多くの組織が次のような大きな課題に直面しています。

- 非効率的な通知システムにより、重大な脆弱性の認識が遅れる
- 承認ワークフローの切断による意思決定プロセスの遅延
- SAST スキャンの失敗に対する即時かつ実用的な対応の欠如
- セキュリティ検出結果に関する断片化されたコミュニケーションとコラボレーション
- セキュリティツールの時間がかかり、エラーが発生しやすい手動インフラストラクチャ設定

これらの問題は、多くの場合、セキュリティリスクの増加、リリースの遅延、チームの生産性の低下につながります。これらの課題に効果的に対処するには、SAST の結果管理を合理化し、チームのコラボレーションを強化し、インフラストラクチャのプロビジョニングを自動化できるソリューションが必要です。

ソリューションの主な機能は次のとおりです。

- カスタマイズされた通知 – リアルタイムのアラートと通知はチームチャットチャネルに直接配信されるため、SAST スキャンの脆弱性や障害に対する迅速な認識とアクションが保証されます。
- 会話による承認 – ステークホルダーは、チャットインターフェイス内でシームレスに SAST スキャン結果の承認ワークフローを開始および完了できるため、意思決定プロセスが高速化されます。

- カスタムアクション – チームは、品質ゲートの障害に対して E メールメッセージを自動的にトリガーし、セキュリティ問題への応答性を高めるなど、SAST スキャンの結果に基づいてカスタムアクションを定義して実行できます。
- 一元化されたコラボレーション – SAST スキャン関連のすべての議論、決定、アクションは、統合されたチャット環境内に保持され、チームメンバー間のコラボレーションと知識共有の向上を促進します。
- Infrastructure as Code (IaC) – ソリューション全体が AWS CloudFormation テンプレートでラップされるため、手動セットアップエラーを減らしながら、インフラストラクチャのプロビジョニングを迅速かつ確実に行うことができます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 「 ツール」に AWS のサービス リストされている に関連付けられたリソースを作成および管理するためのアクセス許可を持つ AWS Identity and Access Management (IAM) ロール。
- Slack ワークスペース。
- 必要な Slack ワークスペースにプラグインとして追加されたチャットアプリケーションの Amazon Q Developer。詳細については、Slack ドキュメントの「Slack ワークスペースにアプリケーションを追加する」を参照してください。登録が成功 AWS Management Console したら、「」に示すように Slack ワークスペース ID を書き留めます。
- チャットアプリケーションクライアントで設定された Amazon Q Developer で、ワークスペース ID が AWS CloudFormation コンソールで入力可能。手順については、「Amazon Q Developer in chat applications 管理者ガイド」の「Configure a Slack client」を参照してください。
- 承認 E メールメッセージを送信するために Amazon Simple Email Service (Amazon SES) で作成および検証されるソース E メールアカウント。セットアップ手順については、「Amazon Simple Email Service デベロッパーガイド」の「E メール ID の作成と検証」を参照してください。
- 承認通知を受信するための送信先 E メールアドレス。このアドレスは、共有受信トレイまたは特定のチームディストリビューションリストにすることができます。
- からアクセスできる運用上の SonarQube インスタンス AWS アカウント。詳細については、SonarQube のインストール手順」を参照してください。
- パイプラインを介してプロジェクトをトリガーおよび作成するアクセス許可を持つ SonarQubeuser トークン。

制約事項

- カスタムアクションボタンの作成は、このソリューションの手動プロセスです。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス リージョン別の「」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

この図は、自動化されたコード品質保証ワークフローを示しています。

1. コードの準備とアップロード：
   - 開発者はコードベースを .zip ファイルに圧縮します。
   - 開発者は、.zip ファイルを指定された Amazon Simple Storage Service (Amazon S3) バケットに手動でアップロードします。
2. Amazon S3 イベントトリガーと AWS Step Functions オーケストレーション：
   - Amazon S3 アップロードイベントは Step Functions ワークフローをトリガーします。
   - Step Functions は、SonarQube を使用して SAST スキャンを調整します。
   - ワークフローは AWS CodeBuild ジョブのステータスをモニタリングして、次のアクションを決定します。CodeBuild が成功すると (品質ゲートパス)、ワークフローは終了します。CodeBuild が失敗すると、診断のために AWS Lambda 関数が呼び出されます。詳細については、このセクションの後半にあるAWS Step Functions ロジックを参照してください。
3. AWS CodeBuild 実行：
   - CodeBuild ジョブは、アップロードされたコードベースで SonarQube スキャンを実行します。
   - スキャンアーティファクトは、監査と分析のために別の Amazon S3 バケットに保存されます。
4. 障害分析 (Lambda 関数):
   - CodeBuild が失敗すると、`CheckBuildStatus`Lambda 関数がトリガーされます。
   - CodeBuild が成功すると、プロセスは終了し、それ以上のアクションは必要ありません。
5. Lambda 関数が障害の原因を分析する (品質ゲートの障害やその他の問題)

- CheckBuildStatus 関数は、詳細な障害情報を含むカスタムペイロードを作成します。
- CheckBuildStatus 関数は、カスタムペイロードを Amazon Simple Notification Service (Amazon SNS) トピックに発行します。

6. 通知システム：
   - Amazon SNS は、Slack 統合用のチャットアプリケーションでペイロードを Amazon Q Developer に転送します。
7. Slack 統合：
   - チャットアプリケーションの Amazon Q Developer は、指定された Slack チャネルに通知を投稿します。
8. 承認プロセス：
   - 承認者は、Slack 通知で障害の詳細を確認します。
   - 承認者は、Slack の承認ボタンを使用して承認を開始できます。
9. 承認ハンドラー：
   - 承認 Lambda 関数は、Slack からの承認アクションを処理します。
   - 承認関数は、カスタムメッセージを Amazon SES に発行します。
10. 生成されたメッセージ：
   - 承認関数は、デベロッパー通知用のカスタムメッセージを生成します。
11. 開発者への通知：
   - Amazon SES は、次のステップまたは必要なアクションを含む E メールメッセージを開発者に送信します。

このワークフローは、手動コードアップロードと自動品質チェックを組み合わせて、Slack を通じてすぐにフィードバックを提供し、必要に応じて人間の介入を可能にし、堅牢で柔軟なコードレビュープロセスを確保します。

AWS Step Functions ロジック

前のアーキテクチャ図に示すように、SonarQube の品質ゲートパスが失敗した場合、ワークフローは CheckBuildStatus Lambda 関数に送られます。このCheckBuildStatus関数は、Slack チャネルで通知をトリガーします。各通知には、推奨される次のステップに関する情報が含まれています。通知のタイプは次のとおりです。

- アプリケーションがコードセキュリティスキャンで失敗した – アップロードされたコードが SonarQube セキュリティスキャンに合格しなかった場合、ユーザーはこの通知を受け取ります。

ユーザーは、ビルドを受け入れる承認を選択できます。ただし、この通知では、潜在的なコード品質の低下とセキュリティリスクに注意するようユーザーにアドバイスしています。通知には、次の詳細が含まれます。

  - 次のステップ: エラー: 品質ゲートのステータス: FAILED – 指定された URL で詳細を表示します。
  - 提供された URL のドキュメントに記載されているように、脆弱性をトリアージします。
  - CodeBuild の詳細は、指定された URL の場所にあります。
- アプリケーションスキャンパイプラインが他の理由で失敗した – ユーザーは、コードセキュリティスキャンの失敗以外の理由でパイプラインが失敗したときにこの通知を受け取ります。通知には、次の詳細が含まれます。
  - 次のステップについては、詳細なトラブルシューティングのために提供されているリンクを参照してください。

Slack チャネルに表示される通知のスクリーンショットを表示するには、GitHub chatops-slack リポジトリのアセットフォルダに移動します。

次の図は、品質ゲートパスが失敗した後の Step Functions ステップステータスの例を示しています。

ツール

AWS のサービス

- チャットアプリケーションの Amazon Q Developer を使用すると、Amazon Chime、Microsoft Teams、および Slack チャットチャネルを使用して、 AWS アプリケーションの運用イベントをモニタリングして対応できます。サポート終了通知： 2026 年 2 月 20 日に、 AWS は Amazon Chime サービスのサポートを終了します。2026 年 2 月 20 日以降、Amazon Chime コンソールまたは Amazon Chime アプリケーションリソースにアクセスできなくなります。詳細については、ブログ記事を参照してください。これは Amazon Chime SDK サービスの可用性には影響しません。
- AWS CloudFormation は、 AWS リソースのセットアップ、迅速かつ一貫したプロビジョニング、および AWS アカウント と 全体のライフサイクル全体の管理に役立ちます AWS リージョン。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。
- Amazon Simple Email Service (Amazon SES) は、独自の E メールアドレスとドメインを使用して E メールメッセージを送受信するのに役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータの保存、保護、取得に役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functions はサーバーレスオーケストレーションサービスで、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立ちます。

その他のツール

- Salesforce が提供する Slack は、チャットとビデオのコラボレーションを提供し、コードなしでプロセスを自動化し、情報共有をサポートする AI を活用した会話型プラットフォームです。
- SonarQube は、30 を超える言語、フレームワーク、IaC プラットフォームでコーディングの問題を検出するように設計されたオンプレミスの分析ツールです。

コードリポジトリ

このパターンのコードは、GitHub Chatops-slack リポジトリで入手できます。

ベストプラクティス

- CloudFormation スタック管理 – CloudFormation スタックの実行中に障害が発生した場合は、障害が発生したスタックを削除することをお勧めします。次に、正しいパラメータ値を使用して再作成

します。このアプローチは、クリーンなデプロイをサポートし、潜在的な競合や部分的な実装を回避するのに役立ちます。

- 共有受信トレイ E メール設定 – SharedInboxEmailパラメータを設定するときは、関連するすべてのデベロッパーがアクセスできる共通のディストリビューションリストを使用します。このアプローチは透明性を高め、重要な通知が関連するチームメンバーに到達するのに役立ちます。
- 本番承認ワークフロー – 本番稼働環境では、ビルド承認に使用される Slack チャネルへのアクセスを制限します。指定された承認者のみがこのチャネルのメンバーである必要があります。このプラクティスでは、明確な責任チェーンを維持し、重要な変更を承認できるユーザーを制限することでセキュリティを強化します。
- IAM アクセス許可 – 最小権限の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

初期セットアップを実行する

タスク	説明	必要なスキル
リポジトリをクローン作成します。	このパターンの Chatops-slack リポジトリのクローンを作成するには、次のコマンドを使用します。 `git clone “git@github.com:aws-samples/chatops-slack.git“`	AWS DevOps、ビルドリード、DevOps エンジニア、クラウド管理者
Lambda コードを含む .zip ファイルを作成します。	`CheckBuildStatus` および `ApprovalEmail` 機能の AWS Lambda 関数コードの .zip ファイルを作成します。`notification.zip` とを作成するにはapproval.zip 、次のコマンドを使用します。	AWS DevOps、ビルドリード、DevOps エンジニア、クラウド管理者

タスク	説明	必要なスキル

	`cd chatops-slack/src`
	
	`chmod -R 775 *`
	
	`zip -r approval.zip approval`
	
	`zip -r notification.zip notification`	

pre-requisite.yml スタックファイルをデプロイする

タスク	説明	必要なスキル
`pre-requisite.yml` スタックファイルを実行します。	`pre-requisite.yml` CloudFormation スタックファイルは、`app-security.yml` スタックファイルを実行する前に必要な初期リソースをデプロイします。`pre-requisite.yml` ファイルを実行するには、以下を実行します。 1. にサインインし AWS Management Console、AWS CloudFormationコンソールを開きます。スタックの作成を選択し、ドロップダウンリストから新しいリソース (標準) を選択します。	AWS 管理者、AWS DevOps、ビルドリード、DevOps エンジニア

タスク	説明	必要なスキル
	2. スタックの作成ページで、既存のテンプレートの選択とテンプレートファイルのアップロードを選択します。次に、「ファイルの選択」を選択し、「pre-requisite.yml」を選択します。[次へ] を選択します。 3. スタックの詳細を指定ページで、「追加情報」の説明に従ってパラメータの値を入力します。[次へ] を選択します。 4. 「スタックオプションの設定」ページで、「前提条件」の説明に従って、リソースを作成するための IAM ロールを選択します。[次へ] を選択します。 5. 確認と作成ページで、送信を選択します。 6. スタックの詳細ページで、リソースタブと出力タブを選択します。次のステップで使用される `S3Lambda`、`CKMSKeyArn` 、および `CKMSKeyId` パラメータの値を書き留めます。	

タスク	説明	必要なスキル
.zip ファイルを Amazon S3 バケットにアップロードします。	前に作成した `notification.zip` および `approval.zip` ファイルを、という名前の Amazon S3 バケットにアップロードします`S3LambdaBucket`。`app-security.yml` CloudFormation スタックファイルは、`S3LambdaBucket` を使用して Lambda 関数をプロビジョニングします。	AWS DevOps、ビルドリード、DevOps エンジニア、AWS システム管理者

app-security.yml スタックファイルを実行する

タスク	説明	必要なスキル
`app-security.yml` スタックファイルを実行します。	`app-security.yml` スタックファイルは、通知と承認システムの残りのインフラストラクチャをデプロイします。`app-security.yml` ファイルを実行するには、以下を実行します。 1. にサインインし AWS Management Console、AWS CloudFormationコンソールを開きます。スタックの作成を選択し、ドロップダウンリストから新しいリソース (標準) を選択します。 2. スタックの作成ページで、既存のテンプレートの	AWS DevOps、AWS システム管理者、DevOps エンジニア、ビルドリード

タスク	説明	必要なスキル
	選択とテンプレートファイルのアップロードを選択します。次に、「ファイルの選択」を選択し、「app-security.yml」を選択します。[次へ] を選択します。 3. スタックの詳細を指定ページで、「追加情報」の説明に従ってパラメータの値を入力します。[次へ] を選択します。 4. 「スタックオプションの設定」ページで、「前提条件」の説明に従って、リソースを作成するための IAM ロールを選択します。[次へ] を選択します。 5. 確認と作成ページで、送信を選択します。	

タスク	説明	必要なスキル
通知設定をテストします。	通知設定をテストするには、以下を実行します。 1. [Amazon SNS コンソール] を開きます。左のナビゲーションペインで、[トピック] を選択します。 2. LambdaToAWSSlackChatbot で終わるトピック名を選択します。 3. トピックの詳細ページで、メッセージを発行を選択します。 4. トピックへのメッセージの発行ページで、メッセージ本文がエンドポイントに送信するように、次のように入力します。 `{ "version": "1.0", "source": "custom", "content": { "description": ":warning: This is a test notification" } }` 5. [メッセージの発行] を選択します。 テストメッセージが正常に配信されると、Slack チャネ	AWS DevOps、AWS システム管理者、DevOps エンジニア、ビルドリード

タスク	説明	必要なスキル
	ルに通知が表示されます。詳細については、「Amazon Q Developer in chat applications 管理者ガイド」の「Test notifications from AWS のサービス to Slack」を参照してください。	

承認フローを設定する

タスク	説明	必要なスキル
カスタム Lambda アクションを設定します。	カスタム AWS Lambda アクションを設定するには、以下を実行します。 1. Slack チャネルの配信通知の下部にある縦の省略記号ボタンを選択します。 2. 「アクションの管理」で、「作成」を選択します。 3. カスタムアクション名を入力します。例えば、Approve です。この名前は、カスタムアクションの一意の識別子です。 4. カスタムアクションボタンの名前を入力します。例えば、Approve です。この名前は、通知のボタンに表示されます。この名前は 20 文字以下で、絵文字を組み込むことができます。	AWS 管理者、AWS DevOps、ビルドリード、DevOps エンジニア、Slack 管理者

タスク	説明	必要なスキル
	5. カスタムアクションタイプで、Lambda アクションを選択します。 6. [次へ] を選択します。 7. このアクションをデプロイする AWS アカウント と AWS リージョン を選択します。 8. Lambdas のロードを選択します。 9. Lambda 関数の定義で、ApprovalEmailLambda で終わる Lambda 関数を選択します。 [次へ] を選択します。 10承認ボタンを作成するには、表示条件ページで保存を選択します。	
承認フローを検証します。	承認フローが期待どおりに機能することを検証するには、Slack の「承認」ボタンを選択します。 Slackbot は、確認文字列の承認 E メールが正常に送信されたことを示す通知をメッセージスレッドに送信する必要があります。	AWS 管理者、AWS DevOps、DevOps エンジニア、Slack 管理者

トラブルシューティング

問題	ソリューション
Slack の設定ミス	Slack の設定ミスに関連する問題のトラブルシューティングについては、「Amazon Q Developer in chat applications 管理者ガイド」の「Amazon Q Developer のトラブルシューティング」を参照してください。
その他の理由でスキャンが失敗しました	このエラーは、コードビルドタスクが失敗したことを意味します。問題をトラブルシューティングするには、メッセージ内のリンクに移動します。コードビルドタスクの失敗には、次の原因が考えられます。 • アプリケーションが正しくパッケージ化されていません。`sonar-scanner` コマンドで `sonar.project.env.properties` ファイルが見つかりません。 • `SonarFileName` 、`SonarFile Directory` 、または `SonarToken` パラメータの値が正しくありません。値を確認してから、スタックファイルを再度実行します。 • Sonar ホストに到達できません。 • ログを使用してトラブルシューティングできるその他の問題。

関連リソース

AWS ドキュメント

- Slack クライアントを設定する
- カスタムアクションの作成

- E メールアドレス ID 手順の作成 https://docs.aws.amazon.com/ses/latest/dg/creating-identities.html#verify-email-addresses-procedure
- チュートリアル: Slack の使用を開始する

その他のリソース

- Slack ワークスペースにアプリケーションを追加する (Slack ドキュメント)
- トークンの生成と使用 (SonarQube ドキュメント)
- サーバーのインストールの概要 (SonarQube ドキュメント)

追加情報

このソリューションは、リリース管理の目的で、チャットアプリケーションのカスタムアクションで Amazon Q Developer を強調しています。ただし、特定のユースケースの Lambda コードを変更し、その上に構築することで、ソリューションを再利用できます。

CloudFormation スタックファイルのパラメータ

次の表は、CloudFormation スタックファイル のパラメータとその説明を示していますpre-requisite.yml。

キー	説明
StackName	CloudFormation スタックの名前。
S3LambdaBucket	Lambda コードをアップロードする Amazon S3 バケットの名前。名前はグローバルに一意である必要があります。
SonarToken	「前提条件」で説明されている SonarQube ユーザートークン。

次の表は、CloudFormation スタックファイル のパラメータとその説明を示していますapp-security.yml。

キー	説明

CKMSKeyArn	このスタックで作成された IAM ロールと Lambda 関数で使用される AWS KMS key Amazon リソースネーム (ARN)。
CKMSKeyId	このスタックで作成された Amazon SNS トピックで使用される AWS KMS key ID。
EnvironmentType	アプリケーションスキャンパイプラインをデプロイするクライアント環境の名前。許可された値のドロップダウンリストから環境名を選択します。
S3LambdaBucket	approval.zip および notification.zip ファイルを含む Amazon S3 バケットの名前。
SESEmail	「前提条件」で説明されている Amazon SES に登録された E メール ID の名前。この ID は送信元の E メールアドレスです。
SharedInboxMail	スキャン通知の送信先 E メールアドレス。
SlackChannelId	通知を送信する Slack チャネルのチャネル ID。チャネル ID を検索するには、Slack アプリのチャネル詳細でチャネル名を右クリックします。チャネル ID は下部にあります。
SlackWorkspaceId	「前提条件」で説明されている Slack ワークスペース ID。Slack ワークスペース ID を検索するには、 にサインインし AWS Management Console、チャットアプリケーションコンソールで Amazon Q Developer を開き、設定済みクライアント、Slack、WorkspaceID を選択します。
StackName	CloudFormation スタックの名前。
SonarFileDirectory	sonar.project.<env>.properties ファイルを含むディレクトリ。

SonarFileName	sonar.project.<env>properties ファイルの名前。
SourceCodeZip	ファイルとソースコードを含む .zip sonar.project.<env>properties ファイルの名前。

AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイする

Shrikant Patil (AWS) によって作成

概要

このパターンは、AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイ方法を表示しています。ネットワークファイアウォールリソースは、AWS CloudFormation テンプレートを使用してデプロイされます。ネットワークファイアウォールは、ネットワークトラフィックに合わせて自動的にスケーリングし、数十万の接続をサポートできるため、独自のネットワークセキュリティインフラストラクチャの構築と維持について心配する必要はありません。Transit Gateway は、仮想プライベートクラウド (VPC) とオンプレミスネットワークを相互接続するために使用できるネットワークの中継ハブです。

このパターンでは、ネットワーキングアーキテクチャにインスペクション VPC を組み込む方法も学習します。最後に、このパターンでは、Amazon CloudWatch を使用してファイアウォールのアクティビティをリアルタイムでモニタリングする方法について説明します。

> **Tip**
>
> Network Firewall サブネットを使用して他の AWS サービスをデプロイしないようにするのがベストプラクティスです。これは、Network Firewall がファイアウォールサブネット内の送信元または発信先からのトラフィックを検査できないためです。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Identity and Access Management (IAM) ロールとポリシーの権限
- CloudFormation テンプレートの権限

制約事項

ドメインのフィルタリングに問題があり、別の種類の設定が必要になる可能性があります。詳細については、Network Firewall ドキュメントの「AWS Network Firewallのステートフルドメインリストのルールグループ」を参照してください。

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch Logs
- Amazon VPC
- AWS Network Firewall
- AWS Transit Gateway

ターゲットアーキテクチャ

次のダイアグラムは、ネットワークファイアウォールと Transit Gateway を使用してトラフィックを検査する方法を示しています。

アーキテクチャには、以下のコンポーネントが含まれます。

- アプリケーションは 2 つのスポーク VPC でホストされます。VPC はネットワークファイアウォールによって監視されます。
- Egress VPC はインターネットゲートウェイに直接アクセスできますが、ネットワークファイアウォールによって保護されていません。
- インスペクション VPC は、ネットワークファイアウォールがデプロイされる場所です。

自動化とスケール

「CloudFormation」を使用すると、「インフラストラクチャをコード」として使用してこのパターンを作成できます。

ツール

AWS サービス

- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、 からの口グを一元化するのに役立ちます。一元化により、口グを監視して安全にアーカイブできます。

- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。
- 「AWS Network Firewall」 は、 AWS クラウドの VPC に対して、ステートフルでマネージド型のネットワークファイアウォールならびに侵入検知および防止サービスです。
- AWS Transit Gateway は VPC とオンプレミスネットワークを接続する一元的ハブです。

コード

このパターンのコードは、GitHub 内の「Transit Gateway を使用した AWS Network Firewall デプロイメント」リポジトリで利用できます。このリポジトリの CloudFormation テンプレートを使用して、ネットワークファイアウォールを使用する単一のインスペクション VPC をデプロイできます。

エピック

スポーク VPC とインスペクション VPC の作成

タスク	説明	必要なスキル
CloudFormation テンプレートを準備、デプロイします。	1. GitHub 「リポジトリ」から `cloudformation/aws_nw_fw.yml` テンプレートをダウンロードします。 2. 指定した値を使用してテンプレートを更新します。 3. テンプレートをデプロイします。	AWS DevOps

トランジットゲートウェイとルートを作成する

タスク	説明	必要なスキル
Transit Gateway を作成します。	1. AWS マネジメントコンソールにサインインし、	AWS DevOps

タスク	説明	必要なスキル
	「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで Transit Gatewaysを選択します。 3. [Transit Gateway の作成] を選択します。 4. 名前タグに対して、Transit Gateway の名前を入力します。 5. 説明に対して、Transit Gateway の説明を入力します。 6. Amazon 側の自律システム番号 (ASN) については、デフォルトの ASN 値のままにしておきます。 7. DNS Support オプションを選択します。 8. VPN ECMP サポートオプションを選択します。 9. デフォルトルートテーブルの関連付けオプションを選択します。このオプションは、Transit Gateway アタッチメントを Transit Gateway のデフォルトルートテーブルに自動的に関連付けます。 10.デフォルトルートテーブルの伝播オプションを選択します。このオプションは、Transit Gateway アタッチ	

タスク	説明	必要なスキル
	メントを Transit Gateway のデフォルトルートテーブルに自動的に伝播します。 11[Transit Gateway の作成] を選択します。	
Transit Gateway アタッチメントを作成します。	以下については、「Transit Gateway アタッチメントを作成」します。 • インスペクション VPC および Transit Gateway サブネット内のインスペクションアタッチメント • スポーク VPCA とプライベートサブネットの SpokeVPCA アタッチメント • スポーク VPC とプライベートサブネットの SpokeVPCB アタッチメント • エグレス VPC とプライベートサブネットの EgressVPC アタッチメント	AWS DevOps

タスク	説明	必要なスキル
Transit Gateway ルートテーブルを作成します。	1. スポーク VPC に対して、「トランジットゲートウェイのルートテーブルを作成」します。このルートテーブルは、インスペクション VPC 以外のすべての VPC に関連付ける必要があります。 2. ファイアウォールに対して、「トランジットゲートウェイのルートテーブルを作成」します。このルートテーブルはインスペクション VPC にのみ関連付ける必要があります。 3. ファイアウォールに対して、Transit Gateway のルートテーブルにルートを追加します。 • `0.0.0/0` には EgresSVPC アタッチメントを使用してください。 • SpokeVPCA CIDR ブロックには、SpokeVPC1 アタッチメントを使用してください。 • SpokeVPCA CIDR ブロックに対して、SpokeVPC2 アタッチメントを使用します。 4. スポーク VPC に対して、Transit Gateway の	AWS DevOps

タスク	説明	必要なスキル
	ルートテーブルにルートを追加します。`0.0.0/0` には、インスペクション VPC アタッチメントを使用してください。	

ファイアウォールとルートを作成する

タスク	説明	必要なスキル
インスペクション VPC にファイアウォールを作成します。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」を開きます。 2. ナビゲーションペインで、ネットワークファイアウォールの下にあるファイアウォールを選択します。 3. [ファイアウォールを作成] を選択します。 4. 名前に対して、このファイアウォールの識別に使用する名前を入力します。ファイアウォールの作成後は、ファイアウォールの名前を変更することはできません。 5. VPC の場合は、インスペクション VPC を選択します。 6. アベイラビリティーゾーンとサブネットでは、特定したゾーンとファイア	AWS DevOps

タスク	説明	必要なスキル
	ウォールサブネットを選択します。 7. [関連する関連付けられたファイアウォールポリシー] セクションで、[既存のファイアウォールポリシーを関連付ける] を選択し、先ほど作成したファイアウォールポリシーを選択します。 8. [ファイアウォールを作成] を選択します。	

タスク	説明	必要なスキル
ファイアウォールポリシーを作成します。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインの [Network Firewall] (ネットワークファイアウォール) で [Firewall policies] (ファイアウォールポリシー) を選択します。 3. 「ファイアウォールポリシーの説明」ページで、[ファイアウォールポリシーの作成] を選択します。 4. [名前] に、ファイアウォールポリシーに使用する名前を入力します。このパターンの後半でポリシーをファイアウォールに関連付けるときに、この名前を使用してポリシーを識別します。一度作成したトラフィックポリシーの名前は変更できません。 5. [Next (次へ)] を選択します。 6. ルールグループを追加ページのステートレスルールグループセクションで、ステートレスルールグループを追加するを選択します。	AWS DevOps

タスク	説明	必要なスキル
	7. 既存のルールグループから追加ダイアログボックスで、先ほど作成したステートレスルールグループのチェックボックスを選択します。ルールグループを追加 を選択します。注:ページの下部にあるファイアウォールポリシーの容量カウンターには、ファイアウォールポリシーで許可されている最大キャパシティの横に、このルールグループを追加して消費された容量が表示されます。 8. ステートレスのデフォルトアクションをステートフルルールに転送するに設定します。 9. ステートフルルールグループセクションでステートフルルールグループを追加を選択し、先ほど作成したステートフルルールグループのチェックボックスを選択します。ルールグループを追加 を選択します。 10次へを選択してセットアップウィザードの残りの手順を実行し、ファイアウォールポリシーを作成を選択します。	

タスク	説明	必要なスキル
VPC ルートテーブルを更新します	インスペクション VPC ルートテーブル 1. ANF サブネットルートテーブル (`Inspection-ANFRT`) で、Transit Gateway ID `0.0.0/0` に追加します。 2. トランジットゲートウェイのサブネットルートテーブル (`Inspection-TGWRT`) で、 EgressVPC に `0.0.0/0` を追加します。 SpokeVPCA ルートテーブル プライベートルートテーブルで、 `0.0.0.0/0` をトランジットゲートウェイ ID に追加します。 スポーク VPCB ルートテーブル プライベートルートテーブルで、 `0.0.0.0/0` をトランジットゲートウェイ ID に追加します。 egress VPC ルートテーブル egress パブリックルートテーブルで、SpokeVPCA ブロックとスポーク VPCB CIDR ブロックをTransit Gateway ID に追加します。プライベート	AWS DevOps

タスク	説明	必要なスキル
	サブネットについても同じ手順を繰り返します。	

リアルタイムのネットワーク検査を実行するように CloudWatch をセットアップする

タスク	説明	必要なスキル
ファイアウォールのロギング設定を更新します。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで、ネットワークファイアウォールの下にあるファイアウォールを選択します。 3. ファイアウォールページで、編集するファイアウォールの名前を選択します。 4. ファイアウォールの詳細タブを選択します。ログ記録セクションで、編集を選択します。 5. 必要に応じてログタイプの選択を調整します。アラートログとフローログのロギングを設定できます。 • アラート — アクションが Alert または Drop に設定されているステートフルルールに一致するトラフィックのログを送信します。ステートフルの	AWS DevOps

タスク	説明	必要なスキル
	ルールとルールグループについては、「AWS Network Firewall のルールグループ」を参照してください。 • Flow — ステートレスエンジンがステートフルルールエンジンに転送するすべてのネットワークトラフィックのログを送信します。 6. 選択した各ログタイプについて、送信先タイプを選択し、ロギング先の情報を入力します。詳細については、ネットワークファイアウォールのドキュメントの「AWS Network Firewall ネットワークファイアウォールロギング先」を参照してください。 7. [Save] を選択します。	

セットアップを確認します。

タスク	説明	必要なスキル
EC2 インスタンスを起動してセットアップをテストします。	スポーク VPC 内に 2 つの「Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起動します」。1 つはジャンプボックス用、もう 1 つはテスト接続用です。	AWS DevOps

タスク	説明	必要なスキル
メトリクスを確認します。	メトリクスはまずサービスの名前空間ごとにグループ化され、次に各名前空間内のさまざまなディメンションの組み合わせごとにグループ化されます。ネットワークファイアウォールの CloudWatch 名前空間は `AWS/NetworkFirewall` です。 1. AWS マネジメントコンソールにサインインし、「CloudWatch コンソール」 を開きます。 2. ナビゲーションペインで Metrics (メトリクス) を選択します。 3. すべてのメトリクスタブでリージョンを選択し、AWS/ネットワークファイアウォールを選択します。	AWS DevOps

関連リソース

- 「インターネットゲートウェイを備えたシンプルなシングルゾーンアーキテクチャ」
- 「インターネットゲートウェイを備えたマルチのゾーンアーキテクチャ」
- 「インターネットゲートウェイと NAT ゲートウェイを使用するアーキテクチャ」

AWS CodePipeline CI/CD パイプラインで AWS Glue ジョブをデプロイする

作成者: Bruno Klein (AWS) および Luis Henrique Massao Yamada (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、AWS CodeCommit と AWS CodePipeline を AWS Glue と統合し、開発者が変更をリモート AWS CodeCommit リポジトリにプッシュするとすぐに AWS Lambda を使用してジョブを起動する方法を示しています。 AWS CodeCommit

開発者が抽出、変換、ロード (ETL) リポジトリに変更を送信し、その変更を AWS CodeCommit にプッシュすると、新しいパイプラインが呼び出されます。パイプラインは、これらの変更を含む AWS Glue ジョブを起動する Lambda 関数を開始します。AWS Glue ジョブは ETL タスクを実行します。

このソリューションは、企業、開発者、データエンジニアが変更をコミットしてターゲットリポジトリにプッシュしたらすぐにジョブを開始する場合に役立ちます。より高いレベルの自動化と再現性を実現できるため、ジョブの起動時とライフサイクル中のエラーを回避できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Git がローカルマシンにインストールされている
- Amazon Cloud 開発キット (Amazon CDK) がローカルマシンにインストールされている
- Python がローカルマシンにインストールされている
- 添付ファイルセクションのコード

制限

- パイプラインは、AWS Glue ジョブが正常に起動されるとすぐに終了します。ジョブの完了を待たずに行われます。

- 添付のコードはデモ専用です。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Glue
- AWS Lambda
- AWS CodePipeline
- AWS CodeCommit

ターゲット アーキテクチャ

このプロセスは次の 4 つの手順で構成されます。

1. 開発者またはデータエンジニアは ETL コードを変更してコミットし、その変更を AWS CodeCommit にプッシュします。
2. プッシュによってパイプラインが開始されます。
3. パイプラインは Lambda 関数を開始します。Lambda 関数はリポジトリで `codecommit:GetFile` を呼び出し、ファイルを Amazon Simple Storage Service (Amazon S3) にアップロードします。
4. Lambda 関数は ETL コードで新しい AWS Glue ジョブを起動します。
5. Lambda 関数はパイプラインを終了します。

自動化とスケール

添付ファイルのサンプルは、AWS Glue を AWS CodePipeline と統合する方法を示しています。ユーザー自身で使用するためにカスタマイズまたは拡張できるベースラインサンプルを提供します。詳細については、エピックセクションを参照ください。

ツール

- AWS CodePipeline – AWS CodePipeline はフルマネージド型の継続的デリバリーサービスで、アプリケーションとインフラストラクチャの更新を迅速かつ信頼できるものにするパイプラインのリリース自動化に役立ちます。

- AWS CodeCommit — AWS CodeCommit は、安全な Git ベースのリポジトリをホストするフルマネージド型のソース管理サービスです。
- AWS Lambda – AWS Lambda はサーバーをプロビジョニングまたは管理しなくてもコードを実行できるサーバーレスコンピュートサービスです。
- AWS Glue – AWS Glue は、分析、機械学習、アプリケーション開発用のデータの検出、準備、結合を容易にするサーバーレスデータ統合サービスです。
- Git クライアント — Git は GUI ツールを提供し、コマンドラインまたはデスクトップツールを使用して GitHub から必要なアーティファクトをチェックアウトできます。
- AWS CDK — AWS CDK は、使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースの定義に役立つオープンソースのソフトウェア開発フレームワークです。

エピック

サンプルコードをデプロイする

タスク	説明	必要なスキル
AWS CLI を設定します。	AWS コマンドラインインターフェイス (AWS CLI) を設定し、現在の AWS アカウントをターゲットにして認証します。手順については、AWS CLI ドキュメントを参照してください。	開発者、DevOpsエンジニア
サンプルプロジェクトファイルを抽出します。	添付ファイルからファイルを抽出し、サンプルプロジェクトファイルを含めるフォルダを作成します。	開発者、DevOpsエンジニア

サンプルコードをデプロイします。	ファイルを抽出したら、抽出場所から以下のコマンドを実行してベースラインサンプルを作成します。
	`cdk bootstrap`
	`cdk deploy`
		開発者、DevOpsエンジニア

タスク	説明	必要なスキル
	``` git init git remote add origin <code-commit-repos itory-url> git stage . git commit -m "adds sample code" git push --set-upstream origin main ``` 最後のコマンドの後、パイプラインと AWS Glue ジョブのステータスを監視できます。	
コードをカスタマイズします。	etl.py ファイルのコードをビジネス要件に合わせてカスタマイズします。ETL コードを改訂、パイプラインステージを変更、またはソリューションを拡張できます。	データエンジニア

関連リソース

- AWS CDK の使用開始
- AWS Glue にジョブを追加する
- CodePipeline のソースアクション統合
- CodePipeline の パイプラインで AWS Lambda 関数を呼び出す
- AWS Glue プログラミング
- AWS CodeCommit GetFile API

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

EC2 インスタンスプロファイルを使用して AWS Cloud9 から Amazon EKS クラスターをデプロイする

作成者: Sagar Panigrahi (AWS)

概要

注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら

このパターンでは、AWS Cloud9 と AWS CloudFormation を使用して Amazon Elastic Kubernetes Service (Amazon EKS) クラスターを作成する方法を説明します。このクラスターは、Amazon Web Services (AWS) アカウントのユーザーにプログラムによるアクセスを有効にしなくても操作できます。

AWS Cloud9 は、コードの記述、実行、デバッグに使用するクラウドベースの統合開発環境 (IDE) です。AWS Cloud9 は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイルと AWS CloudFormation テンプレートを使用して Amazon EKS クラスターをプロビジョニングするコントロールセンターとして使用されます。

AWS Identity and Access Management (IAM) ユーザーを作成せず、代わりに IAM ロールを使用する場合は、このパターンを使用できます。ロールベースのアクセスコントロール (RBAC) は、各ユーザーのロールに基づいてリソースへのアクセスを規制します。このパターンは、Amazon EKS クラスター内の RBAC を更新し、特定の IAM ロールへのアクセスを許可する方法を示しています。

このパターンの設定は、DevOps チームが AWS Cloud9 の機能を使用して Amazon EKS インフラストラクチャを作成する Infrastructure as Code (IaC)リソースを維持および開発する上でも役立ちます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- アカウントの IAM ロールとポリシーを作成する権限。ユーザーの IAM ロールには `AWSCloud9Administrator` ポリシーが含まれている必要があります。Amazon EKS クラスターの作成に必要であるため、`AWSServiceRoleForAmazonEKS` と `eksNodeRoles` ロールも作成する必要があります。
- Kubernetes のコンセプトに関する知識。

制限

- このパターンでは、基本的な Amazon EKS クラスターの作成方法を説明します。本稼働環境のクラスターの場合は、AWS CloudFormation テンプレートを更新する必要があります。
- このパターンでは、追加 Kubernetes コンポーネント (Fluentd、Ingress コントローラーまたはストレージコントローラーなど) はデプロイされません。

アーキテクチャ

テクノロジースタック

- AWS Cloud9
- AWS CloudFormation
- Amazon EKS
- IAM

自動化とスケール

このパターンを拡張して、継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインに組み込んで、Amazon EKS のプロビジョニング全体を自動化することもできます。

ツール

- AWS CloudFormation – AWS CloudFormation は AWS リソースのモデル化とセットアップに役立ち、リソースの管理時間を減らし、アプリケーションによりフォーカスできます。
- AWS Cloud9 – AWS Cloud9 は、リッチなコード編集エクスペリエンスを実現しており、複数のプログラミング言語、ランタイムデバッガ、組み込みターミナルがサポートされています。
- AWS CLI – AWS コマンドラインインターフェイス (AWS CLI) はオープンソースツールで、コマンドラインシェルのコマンドを使用して AWS サービスとインタラクトできます。
- Kubectl — `kubectl` は、Amazon EKS クラスターとインタラクトできるコマンドラインユーティリティです。

エピック

EC2 インスタンスプロファイルの IAM ロールを作成する

タスク	説明	必要なスキル
IAM ポリシーを作成します。	AWS マネジメントコンソールにサインインし、IAM コンソールを開き、[Policies (ポリシー)] を選択してから、[Create policy (ポリシーの作成)] を選択します。[JSON] タブを選択し、policy-role-eks-instance-profile-for-cloud9.json ファイル (添付) のコンテンツを貼り付けます。 ポリシー検証中に生成されたセキュリティ警告、エラー、または一般的な警告を解決してから、[Review policy (ポリシーの確認)] を選択します。ポリシーの [Name(名前)] を入力します。ポリシー名に `eks-instance-profile-for-cloud9` を使用することをお勧めします。 ポリシーの [Summary (概要)] を確認して、ポリシーによって付与された権限を確認します。次に、[Create policy (ポリシーの作成)] を選択します。	クラウド管理者
ポリシーを使用して IAM ロールを作成します。	IAM コンソールで、[Roles (ロール)] を選択してから、[Create role (ロー	クラウド管理者

タスク	説明	必要なスキル
	ルの作成)] を選択します。[AWS Service (AWS サービス)] を選択してから、リストから [EC2] を選択します。 [Next: Permissions (次へ:アクセス許可) 」を選択し、前に作成した IAM ポリシーを検索します。要件に適切なタグを選択します。 確認セクションに、ロールの名前を入力します。ロール名には `role-eks-instance-profile-for-cloud9` を使用することをお勧めします。次に、[Create role (ロールの作成)] を選択します。	

Amazon EKS RBAC の IAM ポリシーとロールを作成する

タスク	説明	必要なスキル
IAM ポリシーを作成します。	IAM コンソールで、[Policies (ポリシー)] を選択してから、[Create policy (ポリシーの作成)] を選択します。[JSON] タブを選択し、policy-for-eks-rbac.json ファイル (添付) のコンテンツを貼り付けます。 ポリシー検証中に生成されたセキュリティ警告、エラー、または一般的な警告を解決し	クラウド管理者

タスク	説明	必要なスキル
	てから、[Review policy (ポリシーの確認)] を選択します。ポリシーの [Name(名前)] を入力します。ポリシー名に `policy-for-eks-rbac` を使用することをお勧めします。ポリシーの [Summary (概要)] を確認して、ポリシーによって付与された権限を確認します。次に、[Create policy (ポリシーの作成)] を選択します。	
ポリシーを使用して IAM ロールを作成します。	IAM コンソールで、[Roles (ロール)] を選択してから、[Create role (ロールの作成)] を選択します。[AWS Service (AWS サービス)] を選択してから、リストから [EC2] を選択します。[Next: Permissions (次へ:アクセス許可) 」を選択し、前に作成した IAM ポリシーを検索します。要件に適切なタグを選択します。 確認セクションに、ロールの名前を入力します。ロール名には `role-eks-admin-for-rbac` を使用することをお勧めします。次に、[Create role (ロールの作成)] を選択します。	クラウド管理者

AWS Cloud9 環境を作成する

タスク	説明	必要なスキル
AWS Cloud9 環境を作成します。	AWS Cloud9 コンソールを開いて、[Create environment (環境を作成)] を選択します。 [Name environment (名前環境)] ページで、環境の名前を入力します。環境名には `eks-management-env` を使用することをお勧めします。要件に応じて、残りの設定をしてから、「Next step (次の手順)] を選択します。 [Review (確認)] ページで、[Create environment (環境の作成)] を選択します。AWS Cloud9 が環境を作成するのを待機します。これには数分間かかる場合があります。 使用可能な設定オプションの詳細については、AWS Cloud9 ドキュメントの EC2 環境の作成を参照してください。	クラウド管理者
AWS Cloud9 の一時的な IAM 認証情報を削除します。	AWS Cloud9 環境がプロビジョニングされたら、歯車アイコンの [Settings (設定)] を選択します。[Preferences (設定)] で、[AWS settings (AWS 設定)] を選択してから、[Credentia	クラウド管理者

タスク	説明	必要なスキル
	ls (認証情報)] を選択します。 AWS マネージドー時認証情報をオフにしてタブを閉じます。	
EC2 インスタンスプロファイルを基盤となる EC2 インスタンスにアタッチします。	Amazon EC2 コンソールを開き、AWS Cloud9 の環境と一致する EC2 インスタンスを選択します。推奨した名前を使用した場合は、EC2 インスタンスは `aws-cloud9-eks-management-env` と呼ばれます。 EC2 インスタンスを選択し、[Actions (アクション)] を選択してから、[Instance settings (インスタンス設定)] を選択します。[Attach/replace IAM role (IAM ロールをアタッチ/置換)] を選択します。`role-eks-instance-profile-for-cloud9` または前に作成した IAM ロールの名前を検索してから、[Apply (適用)] を選択します。	クラウド管理者

Amazon EKS クラスターを作成する

タスク	説明	必要なスキル
Amazon EKS クラスターを作成します。	AWS CloudFormation 用の eks-cfn.yaml (添付) テンプレートをダウンロードして開きます。要件に従って、テンプレートを編集します。 AWS Cloud9 環境を開き、[New file (新規ファイル)] を選択します。先に作成した AWS CloudFormation テンプレートをフィールドに貼り付けます。テンプレート名には eks-cfn.yaml を使用することをお勧めします。 AWS Cloud9 ターミナルで、次のコマンドを実行して Amazon EKS クラスターを作成します。 `aws cloudformation create-stack --stack-name eks-cluster --template-body file://eks-cfn.yaml --region <your_AWS_Region>` AWS CloudFormation の呼び出しが成功すると、出力に AWS CloudFormation スタックの Amazon リソースネーム (ARN) が表示されます。ス	クラウド管理者

タスク	説明	必要なスキル
	タックの作成には、10～20 分かかる場合があります。	
Amazon EKS クラスターのステータスを検証します。	AWS CloudFormation コンソールで、[スタック] ページを開いてから、スタック名を選択します。 スタックはスタックステータスコードが CREATE_COMPLETE を示すと作成されます。詳細については、AWS CloudFormation ドキュメントの AWS CloudFormation スタックデータとリソースを表示するを参照してください。	クラウド管理者

Amazon EKS クラスターの Kubernetes リソースにアクセスする

タスク	説明	必要なスキル
AWS Cloud9 環境に kubectl をインストールします。	Amazon EKS ドキュメントの kubectl をインストールするの指示に従って、AWS Cloud9 環境に kubectl をインストールします。	クラウド管理者
AWS Cloud9 の新しい Amazon EKS 設定を更新します。	AWS Cloud9 ターミナルで次のコマンドを実行して、kubeconfig を Amazon EKS クラスターから AWS Cloud9 環境に更新します。 `aws eks update-kubeconfig --name`	クラウド管理者

タスク	説明	必要なスキル
	`EKS-DEV2 --region <your_AWS_Region>` ⚠ Important EKS-DEV2 は、クラスターの作成に使用した AWS CloudFormation テンプレート内の Amazon EKS クラスターの名前です。 `kubectl get all -A` コマンドを実行し、すべての Kubernetes リソースを表示します。	

タスク	説明	必要なスキル

管理者の IAM ロールを Kubernetes RBAC に追加します。	AWS Cloud9 ターミナルで次のコマンドを実行して、Amazon EKS の RBAC 設定マップを編集モードで開きます。
	
	`kubectl edit cm/aws-auth -n kube-system`
	
	`mapRoles` セクションに、次の行を追加します。
	
	`- groups:`
	`- system:masters`
	`rolearn: <ARN_of_IAM_role _from_second_epic>`
	`username: eksadmin`
	
	構文エラーを避けるため、YAML 形式のファイルをリントします。`vi` コマンドを使用してファイルを保存してから、ファイルを終了します。
	
	Note
	このセクションを追加することで、Amazon EKS クラスターで完全な管理者アクセスを受け取ることを `<ARN_of_IAM_role _from_second_epic>`
		クラウド管理者

タスク	説明	必要なスキル
	Kubernetes RBAC に通知します。つまり、特定された IAM ロールは Kubernetes クラスターで管理アクションを実行できるということです。Amazon EKS クラスターがプロビジョニングされている間、AWS は既存のセクションを `mapRoles` に追加します。	

関連リソース

リファレンス

- モジュール式でスケーラブルな Amazon EKS アーキテクチャ(クイックスタート)
- Amazon EKS クラスターのユーザーまたは IAM ロールを管理する
- 新しい Amazon EKS コントロールプレーンを作成する AWS CloudFormation テンプレート

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CodePipeline、AWS CodeCommit、AWS CodeBuild を使用して、複数の AWS リージョンにコードをデプロイする

作成者：Anand Krishna Varanasi (AWS)

概要

このパターンは、AWS AWS CloudFormation を使用して複数のAmazon Web Services (AWS) リージョンにまたがるインフラストラクチャまたはアーキテクチャを構築する方法を示しています。これには、複数の AWS リージョンにわたる継続的インテグレーション (CI) / 継続的デプロイ (CD) が含まれており、デプロイを迅速に行うことができます。 このパターンのステップは、例として 3 つの AWS リージョンにデプロイする AWS CodePipeline ジョブの作成でテストされています。ユースケースに基づいてリージョン数を変更できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- • Amazon S3 FullAccess ポリシーと CloudWatch FullAccess ポリシーを含む CodeBuild ロール。これらのポリシーは、CodeBuild に Amazon CloudWatch を通じて AWS CodeCommit のイベントをモニタリングしたり、Amazon Simple Storage Service (Amazon S3) をアーティファクトストアとして使用したりするためのアクセス権を付与します。
  - 以下のポリシーを持つ AWS CloudFormation ロール。これにより、AWS CloudFormation は、ビルドの最終段階で AWS Lambda 関数を作成または更新したり、Amazon CloudWatch Logs をプッシュまたはモニタリングしたり、変更セットを作成および更新したりできるようになります。
    - AWSLambdaFullAccess
    - AWSCodeDeployFullAccess
    - CloudWatchFullAccess
    - AWSCloudFormationFullAccess
    - AWSCodePipelineFullAccess

> **Note**
>
> AWS CodeBuild と AWS CloudFormation の 2 つの AWS Identity and Access Management (IAM) ロールには、アーティファクトのテスト、バンドル、パッケージ化、複数の AWS

リージョンへのデプロイという CI タスクを並列して実行するための CodeBuild 用の適切なポリシーがあります。 CodePipeline によって作成されたポリシーをクロスチェックして、CodeBuild と AWS CloudFormation が CI および CD フェーズで適切なアクセス許可を持っていることを確認します。

アーキテクチャ

このパターンの複数リージョンのアーキテクチャとワークフローは以下のステップで構成されています。

1. CodeCommit リポジトリにコールを送信します。
2. コードの更新またはコミットを受信すると、CodeCommit は CloudWatch イベントを呼び出し、そのイベントによって CodePipeline ジョブが開始されます。
3. CodePipeline は CodeBuild によって処理される CI を利用します。以下のタスクが実行されます。
   - AWS CloudFormation テンプレートのテスト (オプション)
   - デプロイに含まれる各リージョンの AWS CloudFormation テンプレートをパッケージ化します。例えば、このパターンは 3 つの AWS リージョンに並列にデプロイされるため、CodeBuild は AWS CloudFormation テンプレートを、指定された各リージョンに 1 つずつ、合計 3 つの S3 バケットにパッケージ化します。S3 バケットは CodeBuild によってアーティファクトリポジトリとしてのみ使用されます。
4. CodeBuild は、3 つの AWS リージョンで並列に実行される次のデプロイフェーズの入力としてアーティファクトをパッケージ化します。 異なる数のリージョンを指定すると、CodePipeline はそれらのリージョンにデプロイされます。

ツール

ツール

- AWS CodePipeline – AWS CodePipelineは、ソフトウェアを継続的に変更するために必要な手順のモデル化、視覚化、および自動化に使用できる継続的な配信サービスです。
- AWS CodeBuild – CodeBuild は、ソースコードをコンパイルし、ユニットテストを実行し、すぐにデプロイできるアーティファクトを生成するフルマネージドビルドサービスです。

- AWS CodeCommit — CodeCommit は、Amazon Web Services がホスティングするバージョン管理サービスで、アセット (ソースコード、バイナリファイルなど) をクラウド内で非公開で保存して管理するために使用できます。
- AWS CloudFormation – AWS CloudFormation は、Amazon Web Services リソースのモデル化とセットアップを支援するサービスであるため、これらのリソースの管理に費やす時間を減らし、AWS で実行されるアプリケーションに集中する時間を増やすことができます。
- AWS Identity and Access Management (IAM) – AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスをセキュアに制御するのに役立つウェブサービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。Web スケールのコンピューティングを開発者が容易にできるように設計されています。

コード

次のサンプルコードは BuildSpec.yaml ファイル (ビルドフェーズ) 用です。

```
---
artifacts:
discard-paths: true
files:
- packaged-first-region.yaml
- packaged-second-region.yaml
- packaged-third-region.yaml
phases:
build:
commands:
- echo "********BUILD PHASE - CF PACKAGING**********"
- "aws cloudformation package --template-file sam-template.yaml --s3-bucket
 $S3_FIRST_REGION --output-template-file packaged-first-region.yaml --region
 $FIRST_REGION"
- "aws cloudformation package --template-file sam-template.yaml --s3-bucket
 $S3_SECOND_REGION --output-template-file packaged-second-region.yaml --region
 $SECOND_REGION"
- "aws cloudformation package --template-file sam-template-anand.yaml --s3-bucket
 $S3_THIRD_REGION --output-template-file packaged-third-region.yaml --region
 $THIRD_REGION"
install:
commands:
- echo "********BUILD PHASE - PYTHON SETUP**********"
runtime-versions:
python: 3.8
```

```
post_build:
commands:
- echo "********BUILD PHASE - PACKAGING COMPLETION**********"
pre_build:
commands:
- echo "********BUILD PHASE - DEPENDENCY SETUP**********"
- "npm install --silent --no-progress"
- echo "********BUILD PHASE - DEPENDENCY SETUP DONE**********"
version: 0.2
```

エピック

コードと CodeCommit リポジトリを準備します。

タスク	説明	必要なスキル
デプロイするプライマリ AWS リージョンを選択します。	AWS アカウントにサインインして、デプロイするプライマリリージョンを選択します。CodeCommit リポジトリはプライマリ リージョンにあります。	DevOps
CodeCommit リポジトリを作成します。	CodeCommit リポジトリを作成し、必要なコードをプッシュします。コードには通常、AWS CloudFormation または AWS SAM テンプレート、Lambda コード (ある場合)、および AWS CodePipeline への入力としての CodeBuild `buildspec.yaml` ファイルが含まれます。	DevOps
コードを CodeCommit リポジトリにプッシュします。	「添付ファイル」セクションで、この例のコードをダウンロードし、必要なコードをそのコードにプッシュします。	DevOps

タスク	説明	必要なスキル
	通常、コードには、パイプラインへの入力として AWS CloudFormation または AWS SAM テンプレート、Lambda コード、および CodeBuild `buildspec.yaml` ファイルを含めることができます。	

ソースフェーズ: パイプラインを作成する

タスク	説明	必要なスキル
CodePipeline ジョブを作成します。	CodePipeline コンソールで、[パイプラインの作成] を選択します。	DevOps
CodePipeline ジョブに名前を付け、サービスロール設定を選択します。	ジョブの名前を入力し、デフォルトのサービスロール設定のままにして、CodePipeline が必要なポリシーをアタッチしたロールを作成します。	DevOps
アーティファクトストアの場所を指定します。	[詳細設定] では、デフォルトのオプションをそのまま使用して、CodePipeline がコードアーティファクトストレージに使用する S3 バケットを作成します。代わりに既存の S3 バケットを使用する場合、そのバケットは最初の工ピックで指定したプライマリリージョンにある必要があります。	DevOps

タスク	説明	必要なスキル
暗号化キーを指定します。	デフォルトのオプションであるデフォルトの AWS マネージドキーをそのまま使用するか、AWS Key Management Service (AWS KMS) のカスタマーマネージドキーを使用します。	DevOps
ソースプロバイダを指定します。	[ソースプロバイダ] には、AWS CodeCommit を選択します。	DevOps
リポジトリを指定します。	最初のエピックで作成した CodeCommit リポジトリを選択します。ブランチにコードを配置した場合は、ブランチを選択します。	DevOps
コード変更の検出方法を指定します。	CodeCommit が CodePipeline ジョブを開始するための変更トリガーとして、デフォルトの Amazon CloudWatch Events ままにします。	DevOps

ビルドフェーズ:パイプラインを設定する

タスク	説明	必要なスキル
ビルドプロバイダを指定します。	[ビルドプロバイダ] には、[AWS CodeBuild] を選択します。	DevOps
AWS リージョンを指定します。	最初のエピックで指定したプライマリ リージョンを選択します。	DevOps

ビルドフェーズ:プロジェクトを作成して設定する

タスク	説明	必要なスキル
プロジェクトの作成	プロジェクトの名前を入力し、[プロジェクトの作成] を選択します。	DevOps
環境イメージを指定します。	このパターンのデモンストレーションでは、デフォルトの CodeBuild マネージドイメージを使用します。また、カスタムの Docker イメージがある場合は、使用することもできます。	DevOps
オペレーティングシステムを指定します。	Amazon Linux 2 または Ubuntu のいずれかを選択します。 **Note** Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。	DevOps
サービスロールを指定します。	CodePipeline ジョブの作成を開始する前に CodeBuild 用に作成したロールを選択します。 （「前提条件」セクションを参照してください。）	DevOps
追加のオプションを設定します。	[タイムアウト] と [キュータイムアウト] は、デフォルト値の	DevOps

タスク	説明	必要なスキル
	ままにします。証明書については、使用したいカスタム証明書がない限り、デフォルト設定のままにします。	
環境変数	デプロイする AWS リージョンごとに、S3 バケット名とリージョン名 (us-east-1 など) を指定して環境変数を作成します。	DevOps
buildspec.yml でない場合は、buildspec ファイル名を指定します。	ファイル名がデフォルトの `buildspec.yaml` の場合は、このフィールドを空白のままにします。buildspec ファイルの名前を変更した場合は、ここに名前を入力します。CodeCommit リポジトリ内のファイルの名前と一致することを確認します。	DevOps
ログ記録を指定します。	Amazon CloudWatch Events のログを表示するには、デフォルト設定のままにします。または、特定のグループ名やロガー名を定義することもできます。	DevOps

デプロイフェーズをスキップします。

タスク	説明	必要なスキル
デプロイフェーズをスキップして、パイプラインの作成を完了します。	パイプラインを設定すると、CodePipeline ではデプロイフェーズに 1 つのステー	DevOps

タスク	説明	必要なスキル
	ジのみ作成できます。複数の AWS リージョンにデプロイするには、このフェーズをスキップしてください。パイプラインを作成した後、複数のデプロイ フェーズ段階を追加できます。	

デプロイフェーズ: パイプラインを最初のリージョンにデプロイするように設定します。

タスク	説明	必要なスキル
デプロイ フェーズにステージを追加します。	パイプラインを編集し、デプロイ フェーズでステージを追加を選択します。この第 1 ステージはプライマリ リージョン用です。	DevOps
ステージのアクション名を指定します。	最初の (プライマリ) ステージおよびリージョンを反映する一意の名前を入力します。たとえば、 primary_<リージョン>_deploy と入力します。	DevOps
アクションプロバイダを指定します。	[アクションプロバイダ] で、AWS CloudFormation を選択します。	DevOps
第 1 ステージのリージョンを設定します。	第 1 (プライマリ) リージョンは、CodePipeline と CodeBuild が設定されているのと同じリージョンを選択します。これは、スタックをデプロイするプライマリ リージョンです。	DevOps

タスク	説明	必要なスキル
入力アーティファクトを指定します。	BuildArtifact を選択します。これはビルドフェーズの出力です。	DevOps
必要なアクションを指定します。	[アクションモード] で、[スタックを作成または更新する] をクリックします。	DevOps
CloudFormation スタックの名前を入力します。		DevOps
第 1 リージョンのテンプレートを指定します。	CodeBuild によってパッケージ化され、第 1 (プライマリ) リージョンの S3 バケットにダンプされたリージョン固有のパッケージ名を選択します。	DevOps
機能を指定します。	スタックテンプレートに IAM リソースがある場合やマクロを含むテンプレートから直接スタックを作成する場合に、機能が必要です。このパターンでは、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_AUTO_EXPAND を使用します。	DevOps

デプロイフェーズ:パイプラインを第 2 リージョンにデプロイするように設定します。

タスク	説明	必要なスキル
デプロイ フェーズに第 2 ステージを追加します。	第 2 リージョンにステージを追加するには、パイプライン	DevOps

タスク	説明	必要なスキル
	を編集し、デプロイ フェーズで [ステージを追加] を選択します。重要: 第 2 リージョンの作成プロセスは、次の値を除いて第 1 リージョンの作成プロセスと同じです。	
第 2 ステージのアクション名を指定します。	第 2 ステージと第 2 リージョンを表す一意の名前を入力します。	DevOps
第 2 ステージのリージョンを設定します。	スタックをデプロイする第 2 リージョンを選択します。	DevOps
第 2 リージョンのテンプレートを指定します。	CodeBuild によってパッケージ化され、第 2 リージョンの S3 バケットにダンプされたリージョン固有のパッケージ名を選択します。	DevOps

デプロイフェーズ:パイプラインを第 3 リージョンにデプロイするように設定します。

タスク	説明	必要なスキル
デプロイ フェーズにステージを追加します。	第 3 リージョンにステージを追加するには、パイプラインを編集し、デプロイフェーズで [ステージを追加] を選択します。重要: 第 2 リージョンの作成プロセスは、次の値を除いて、これまでの 2 つのリージョンの作成プロセスと同じです。	DevOps

タスク	説明	必要なスキル
第 3 ステージのアクション名を指定します。	第 3 ステージと第 3 リージョンを表す一意の名前を入力します。	DevOps
第 3 ステージのリージョンを設定します。	スタックをデプロイする第 3 リージョンを選択します。	DevOps
第 3 リージョンのテンプレートを指定します。	CodeBuild によってパッケージ化され、第 3 リージョンの S3 バケットにダンプされたリージョン固有のパッケージ名を選択します。	DevOps

デプロイをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	デプロイをクリーンアップするには、各リージョンの CloudFormation スタックを削除します。次に、CodeCommit、CodeBuild、および CodePipeline リソースをプライマリリージョンから削除します。	DevOps

関連リソース

- 「AWS CodePipeline とは何ですか?」
- 「AWS サーバーレスアプリケーションモデル」
- 「AWS CloudFormation」
- 「AWS CodePipeline 用の AWS CloudFormation アーキテクチャ構造リファレンス」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Terraform を使用して Amazon Redshift SQL クエリを実行する

作成者: Sylvia Qi (AWS) と Aditya Ambati (AWS)

概要

Amazon Redshift のデプロイと管理に Infrastructure as Code (IaC) を使用することは、DevOps 内で一般的なプラクティスです。IaC は、クラスター、スナップショット、パラメータグループなど、さまざまな Amazon Redshift リソースのデプロイと設定を容易にします。ただし、IaC は、テーブル、スキーマ、ビュー、ストアドプロシージャなどのデータベースリソースの管理には拡張されません。これらのデータベース要素は SQL クエリによって管理され、IaC ツールでは直接サポートされていません。これらのリソースを管理するためのソリューションとツールがありますが、テクノロジースタックに追加のツールを導入したくはないかもしれません。

このパターンは、Terraform を使用して、テーブル、スキーマ、ビュー、ストアドプロシージャなどの Amazon Redshift データベースリソースをデプロイする方法の概要を示しています。このパターンは、2 種類の SQL クエリを区別します。

- 再現不可能なクエリ – これらのクエリは、Amazon Redshift の初回デプロイ時に 1 回実行され、重要なデータベースコンポーネントを確立します。
- 繰り返し可能なクエリ – これらのクエリはイミュータブルであり、データベースに影響を与えることなく再実行できます。このソリューションでは、Terraform を使用して繰り返し可能なクエリの変更をモニタリングし、それに応じて適用します。

詳細については、「追加情報」の「ソリューションのチュートリアル」を参照してください。

前提条件と制限

前提条件

をアクティブに AWS アカウント し、デプロイマシンに以下をインストールする必要があります。

- AWS Command Line Interface (AWS CLI)
- Amazon Redshift の読み取り/書き込みアクセス許可で設定されたAWS CLI プロファイル
- Terraform バージョン 1.6.2 以降
- Python3
- Boto3

制約事項

- このソリューションは、Terraform ではクラスターの作成中に 1 つのデータベースしか作成できないため、1 つの Amazon Redshift データベースをサポートします。
- このパターンには、繰り返し可能なクエリへの変更を適用する前に検証するテストは含まれません。信頼性を高めるために、このようなテストを組み込むことをお勧めします。
- このソリューションを説明するために、このパターンはローカル Terraform 状態`redshift.tf`ファイルを使用するサンプルファイルを提供します。ただし、本番環境では、安定性とコラボレーションを強化するために、ロックメカニズムを備えたリモート状態ファイルを使用することを強くお勧めします。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

このソリューションは、Amazon Redshift パッチ 179 で開発およびテストされています。

コードリポジトリ

このパターンのコードは、GitHub amazon-redshift-sql-deploy-terraform リポジトリで入手できます。

アーキテクチャ

次の図は、Terraform が再現不可能な SQL クエリと反復可能な SQL クエリの両方を処理することで Amazon Redshift データベースリソースを管理する方法を示しています。

図表に示す内容は以下のステップです。

1. Terraform は、Amazon Redshift クラスターの初回デプロイ時に、再現不可能な SQL クエリを適用します。
2. 開発者は、反復可能な SQL クエリに変更をコミットします。
3. Terraform は、反復可能な SQL クエリの変更をモニタリングします。
4. Terraform は、Amazon Redshift データベースに繰り返し可能な SQL クエリを適用します。

このパターンが提供するソリューションは、Amazon Redshift の Terraform モジュールに基づいて構築されています。Terraform モジュールは Amazon Redshift クラスターとデータベースをプロビジョニングします。モジュールを強化するために、Amazon Redshift ExecuteStatement API オペレーションを使用して SQL クエリを実行するカスタム Python スクリプトを呼び出す `terraform_data`リソースを使用しました。その結果、モジュールは以下を実行できます。

- データベースのプロビジョニング後に SQL クエリを使用して、任意の数のデータベースリソースをデプロイします。
- 繰り返し可能な SQL クエリの変更を継続的にモニタリングし、Terraform を使用してそれらの変更を適用します。

詳細については、「追加情報」の「ソリューションのチュートリアル」を参照してください。

ツール

AWS のサービス

- Amazon Redshift は、 でフルマネージド型のペタバイト規模のデータウェアハウスサービスです AWS クラウド。

その他のツール

- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。
- Python は、SQL クエリを実行するためにこのパターンで使用される汎用プログラミング言語です。

ベストプラクティス

- Amazon Redshift のベストプラクティス
- Amazon Redshift Data API を使用して Amazon Redshift クラスターを操作する

エピック

Terraform を使用してソリューションをデプロイする

タスク	説明	必要なスキル
リポジトリのクローンを作成します。	Amazon Redshift クラスターをプロビジョニングするための Terraform コードを含む Git リポジトリのクローンを作成するには、次のコマンドを使用します。 `git clone https://github.com/aws-samples/amazon-redshift-sql-deploy-terraform.git`	DevOps エンジニア

Terraform 変数を更新します。	特定の要件に従って Amazon Redshift クラスターのデプロイをカスタマイズするには、`terraform.tfvars` ファイルで次のパラメータを更新します。
	`region = "<AWS_REGION>"`
	`cluster_identifier = "<REDSHIFT_CLUSTER_IDENTIFIER>"`
	`node_type = "<REDSHIFT_NODE_TYPE>"`
	`number_of_nodes = "<REDSHIFT_NODE_COUNT>"`
		DevOps エンジニア

タスク	説明	必要なスキル
	database_name = "<REDSHIFT_DB_NAME>" subnet_ids = "<REDSHIFT_SUBNET_IDS>" vpc_security_group_ids = "<REDSHIFT_SECURITY_GROUP_IDS>" run_nonrepeatable_queries = true run_repeatable_queries = true sql_path_bootstrap = "<BOOTSTRAP_SQLS_PATH>" sql_path_nonrepeatable = "<NON-REPEATABLE_SQLS_PATH>" sql_path_repeatable = "<REPEATABLE_SQLS_PATH>" sql_path_finalize = "<FINALIZE_SQLS_PATH>" create_random_password = false master_username = "<REDSHIFT_MASTER_USERNAME>"	

タスク	説明	必要なスキル

Terraform を使用してリソースをデプロイします。	1. デプロイプロセスを準備するには、次のコマンドを使用して、クローンされたリポジトリ内で Terraform を初期化します。
	`terraform init`
	2. Terraform がインフラストラクチャに適用する変更をプレビューするには、次のコマンドを使用して実行プランを作成します。
	`terraform plan -var-file terraform.tfvars`
	3. Amazon Redshift クラスターと関連リソースをプロビジョニングするには、次のコマンドを使用して Terraform 実行プランを適用します。
	`terraform apply -var-file terraform.tfvars`
		DevOps エンジニア

タスク	説明	必要なスキル

(オプション) 追加の SQL クエリを実行します。	サンプルリポジトリには、デモ用の SQL クエリがいくつか用意されています。独自の SQL クエリを実行するには、それらを次のフォルダに追加します。
	`/bootstrap`
	`/nonrepeatable`
	`/repeatable`
	`/finalize`	

SQL ステートメントの実行をモニタリングする

タスク	説明	必要なスキル
SQL ステートメントのデプロイをモニタリングします。	Amazon Redshift クラスターへの SQL 実行の結果をモニタリングできます。失敗した SQL 実行と成功した SQL 実行を示す出力の例については、「追加情報」の「SQL ステートメントの例」を参照してください。	DBA、DevOps エンジニア
リソースをクリーンアップします。	Terraform によってデプロイされたすべてのリソースを削除するには、次のコマンドを実行します。 `terraform destroy`	DevOps エンジニア

結果を検証する

タスク	説明	必要なスキル
Amazon Redshift クラスター内のデータを検証します。	1. にサインインし AWS Management Console、Amazon Redshift コンソールを開きます。 2. ナビゲーションメニューで [クラスター] を選択します。リストで関連するクラスター名を選択します。 3. Amazon Redshift ドキュメントの「Amazon Redshift クエリエディタ v2 を使用したデータベースのクエリ」の手順に従います。	DBA、AWS DevOps

関連リソース

AWS ドキュメント

- Amazon Redshift でプロビジョニングされたクラスター
- Amazon Redshift Data API の問題のトラブルシューティング

その他のリソース

- コマンド: apply (Terraform ドキュメント)

追加情報

ソリューションのチュートリアル

このソリューションを使用するには、Amazon Redshift SQL クエリを特定の方法で整理する必要があります。すべての SQL クエリは、 `.sql` 拡張子を持つファイルに保存する必要があります。

このパターンで提供されているコード例では、SQL クエリは次のフォルダ構造に整理されています。コード (sql-queries.tf と sql-queries.py) を変更して、独自のユースケースに合った任意の構造を操作できます。

```
/bootstrap
     |- Any # of files
     |- Any # of sub-folders
/nonrepeatable
     |- Any # of files
     |- Any # of sub-folders
/repeatable
     /udf
          |- Any # of files
          |- Any # of sub-folders
     /table
          |- Any # of files
          |- Any # of sub-folders
     /view
          |- Any # of files
          |- Any # of sub-folders
     /stored-procedure
          |- Any # of files
          |- Any # of sub-folders
/finalize
     |- Any # of files
     |- Any # of sub-folders
```

上記のフォルダ構造を考慮すると、Amazon Redshift クラスターのデプロイ中に、Terraform は次の順序でクエリを実行します。

1. /bootstrap
2. /nonrepeatable
3. /repeatable
4. /finalize

/repeatable フォルダには、、/udf、/table、 /viewの 4 つのサブフォルダが含まれています/stored-procedure。これらのサブフォルダは、Terraform が SQL クエリを実行する順序を示します。

SQL クエリを実行する Python スクリプトは ですsql-queries.py。まず、スクリプトは、sql_path_bootstrapパラメータなど、特定のソースディレクトリのすべてのファイルとサブフォルダを読み取ります。次に、スクリプトは Amazon Redshift ExecuteStatement API オペレーションを呼び出してクエリを実行します。ファイル内に 1 つ以上の SQL クエリがある場合があります。次のコードスニペットは、ファイルに保存されている SQL ステートメントを Amazon Redshift クラスターに対して実行する Python 関数を示しています。

```
def execute_sql_statement(filename, cluster_id, db_name, secret_arn, aws_region):
    """Execute SQL statements in a file"""
    redshift_client = boto3.client(
        'redshift-data', region_name=aws_region)
    contents = get_contents_from_file(filename),
    response = redshift_client.execute_statement(
        Sql=contents[0],
        ClusterIdentifier=cluster_id,
        Database=db_name,
        WithEvent=True,
        StatementName=filename,
        SecretArn=secret_arn
    )
    ...
```

Terraform スクリプトは、sql-queries.pyスクリプトを呼び出す terraform_data リソースsql-queries.tfを作成します。、/bootstrap、、 /nonrepeatable /repeatableの 4 つのフォルダのそれぞれにterraform_dataリソースがあります/finalize。次のコードスニペットは、 /bootstrapフォルダで SQL クエリを実行するterraform_dataリソースを示しています。

```
locals {
  program               = "${path.module}/sql-queries.py"
  redshift_cluster_name = try(aws_redshift_cluster.this[0].id, null)
}

resource "terraform_data" "run_bootstrap_queries" {
  count      = var.create && var.run_nonrepeatable_queries && (var.sql_path_bootstrap !
= "") && (var.snapshot_identifier == null) ? 1 : 0
  depends_on = [aws_redshift_cluster.this[0]]

  provisioner "local-exec" {
    command = "python3 ${local.program} ${var.sql_path_bootstrap}
 ${local.redshift_cluster_name} ${var.database_name} ${var.redshift_secret_arn}
 ${local.aws_region}"
```

```
  }
}
```

これらのクエリを実行するかどうかは、次の変数を使用して制御できます。sql_path_bootstrap、、sql_path_nonrepeatable、sql_path_repeatableまたは でクエリを実行しない場合はsql_path_finalize、それらの値を に設定します""。

```
  run_nonrepeatable_queries = true
  run_repeatable_queries    = true
  sql_path_bootstrap        = "src/redshift/bootstrap"
  sql_path_nonrepeatable    = "src/redshift/nonrepeatable"
  sql_path_repeatable       = "src/redshift/repeatable"
  sql_path_finalize         = "src/redshift/finalize"
```

を実行するとterraform apply、Terraform はスクリプトの結果に関係なく、スクリプトの完了後に追加されたterraform_dataリソースを考慮します。一部の SQL クエリが失敗し、再実行する場合は、Terraform 状態からリソースを手動で削除して、terraform apply再度実行できます。たとえば、次のコマンドは Terraform 状態からrun_bootstrap_queriesリソースを削除します。

```
terraform state rm module.redshift.terraform_data.run_bootstrap_queries[0]
```

次のコード例は、run_repeatable_queriesリソースが sha256 ハッシュを使用してrepeatableフォルダの変更をモニタリングする方法を示しています。フォルダ内のファイルが更新されると、Terraform はディレクトリ全体を更新対象としてマークします。次に、Terraform は次の 中にディレクトリでクエリを再度実行しますterraform apply。

```
resource "terraform_data" "run_repeatable_queries" {
  count      = var.create_redshift && var.run_repeatable_queries &&
 (var.sql_path_repeatable != "") ? 1 : 0
  depends_on = [terraform_data.run_nonrepeatable_queries]

  # Continuously monitor and apply changes in the repeatable folder
  triggers_replace = {
    dir_sha256 = sha256(join("", [for f in fileset("${var.sql_path_repeatable}",
 "**") : filesha256("${var.sql_path_repeatable}/${f}")]))
  }

  provisioner "local-exec" {
    command = "python3 ${local.sql_queries} ${var.sql_path_repeatable}
 ${local.redshift_cluster_name} ${var.database_name} ${var.redshift_secret_arn}"
  }
```

```
}
```

コードを絞り込むには、すべてのファイルに変更を無差別に適用するのではなく、repeatableフォルダ内で更新されたファイルのみに変更を検出して適用するメカニズムを実装できます。

SQL ステートメントの例

次の出力は、失敗した SQL 実行とエラーメッセージを示しています。

```
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Executing:
 ["/bin/sh" "-c" "python3 modules/redshift/sql-queries.py src/redshift/nonrepeatable
 testcluster-1 db1 arn:aws:secretsmanager:us-east-1:XXXXXXXXXXXX:secret:/redshift/
master_user/password-8RapGH us-east-1"]
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):
 ----------------------------------------------------------------
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): src/redshift/
nonrepeatable/table/admin/admin.application_family.sql
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):
 ----------------------------------------------------------------
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Status:
 FAILED
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): SQL execution
 failed.
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Error
 message: ERROR: syntax error at or near ")"
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):   Position:
 244
module.redshift.terraform_data.run_nonrepeatable_queries[0]: Creation complete after 3s
 [id=ee50ba6c-11ae-5b64-7e2f-86fd8caa8b76]
```

次の出力は、成功した SQL 実行を示しています。

```
module.redshift.terraform_data.run_bootstrap_queries[0]: Provisioning with 'local-
exec'...
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): Executing:
 ["/bin/sh" "-c" "python3 modules/redshift/sql-queries.py src/redshift/bootstrap
 testcluster-1 db1 arn:aws:secretsmanager:us-east-1:XXXXXXXXXXXX:secret:/redshift/
master_user/password-8RapGH us-east-1"]
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec):
 ----------------------------------------------------------------
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): src/redshift/
bootstrap/db.sql
```

```
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec):
 ------------------------------------------------------------------
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): Status: FINISHED
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): SQL execution
 successful.
module.redshift.terraform_data.run_bootstrap_queries[0]: Creation complete after 2s
 [id=d565ef6d-be86-8afd-8e90-111e5ea4a1be]
```

AWS Organizations 内の組織全体の AWS Backup レポートを CSV ファイルとしてエクスポートする

作成者: Aromal Raj Jayarajan (AWS) と Purushotham G K (AWS)

概要

このパターンは、AWS Organizations 内のある組織全体の AWS Backup ジョブレポートを CSV ファイルとしてエクスポートする方法を示しています。このソリューションでは、AWS Lambda と Amazon EventBridge を使用して AWS Backup ジョブレポートをステータスに基づいて分類します。これは、ステータスベースの自動化を構成するときに役立ちます。

AWS Backup は、AWS のサービス、クラウド内、およびオンプレミス間の一元化およびデータ保護の自動化を支援します。ただし、AWS Organizations 内で構成された AWS Backup ジョブの場合、統合レポートは各組織の管理アカウントの AWS マネジメントコンソールでのみ使用できます。このレポートを管理アカウントの外部に置くことで、監査に必要な労力を減らし、自動化、通知、アラートの範囲を広げることができます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 少なくとも管理アカウントとメンバーアカウントを含む、AWS Organizations 内のアクティブな組織
- AWS Organizations の組織レベルで構成された AWS Backup (詳細については、AWS ブログの「Backup を使用して AWS のサービス全体にわたる大規模集中バックアップを自動化する」を参照してください)
- ローカルマシンにインストールされて構成されている Git

機能制限

このパターンで提供されるソリューションは、AWS Backup ジョブ専用に構成された AWS リソースを識別します。このレポートでは、AWS Backup によるバックアップ用に構成されていない AWS Backup リソースは特定できません。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Backup
- AWS CloudFormation
- Amazon EventBridge
- AWS Lambda
- AWS Security Token Service (AWS STS)
- Amazon Simple Storage Service (Amazon S3)
- AWS Identity and Access Management (IAM)

ターゲットアーキテクチャ

次の図は、AWS Organizations 内のある組織全体から AWS Backup ジョブレポートを CSV ファイルとしてエクスポートするワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. スケジュールされた EventBridge イベントルールは、メンバー (レポーティング) AWS アカウントの Lambda 関数を呼び出します。
2. 次に、Lambda 関数は AWS STS を使用して、管理アカウントへの接続に必要な権限を持つ IAM ロールを引き受けます。
3. Lambda 関数は以下を実行します。
   - AWS Backup サービスに統合された AWS Backup ジョブレポートをリクエストする
   - AWS Backup ジョブのステータスに基づいて結果を分類する
   - レスポンスを CSV ファイルに変換する
   - 作成日に基づいてラベル付けされたフォルダ内のレポーティングアカウントの Amazon S3 バケットに、結果をアップロードする

ツール

ツール

- AWS Backup は、フルマネージド型のサービスで、 AWS サービス、クラウド、オンプレミスにおけるデータ保護の一元化と自動化に役立ちます。

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。たとえば、AWS Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Storage Service (Amazon S3) 」は、どのようなデータの量であっても、保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

Code

このパターンのコードは、GitHub のaws-backup-report-generator リポジトリにあります。

ベストプラクティス

- Amazon S3 のセキュリティベストプラクティス (Amazon S3 ユーザーガイド)
- AWS Lambda 関数を操作するためのベストプラクティス (AWS Lambda デベロッパーガイド)
- 管理アカウントのベストプラクティス (AWS Organizations ユーザーガイド)

エピック

ソリューションコンポーネントをデプロイする

タスク	説明	必要なスキル
GitHub リポジトリのクローン作成	ターミナルウィンドウで次のコマンドを実行して、GitHub の aws-backup-report-generator リポジトリをクローン作成します。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	`git clone https://github.com/aws-samples/aws-backup-report-generator.git` 詳細については、GitHub Docs の「リポジトリのクローン作成」を参照してください。	

タスク	説明	必要なスキル
メンバー (レポーティング) AWS アカウントにソリューションコンポーネントをデプロイします。	1. メンバー (レポーティング) アカウントで AWS マネジメントコンソールにサインインし、CloudFormation コンソールを開きます。 2. [スタックの作成] を選択し、[With new resources (standard) 新しいリソースを使用 (標準)] を選択します。 3. [スタックの作成] ページの [テンプレートを指定] セクションで、[テンプレートファイルをアップロード] を選択します。 4. [Choose file] を選択します。次に、ローカルワークステーションにある、クローン作成された GitHub リポジトリのルートフォルダに移動し、template-reporting.yaml を選択します。 5. [開く] を選択し、[次へ] を選択します。 6. 「スタックの詳細を指定」ページの [スタック名] に、CloudFormation スタックの名前を入力します。 7. ManagementAccountID には、AWS Organizations 内の組織の管理アカウントの	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	AWS アカウント ID を入力します。 8. [Next (次へ)] を選択します。 9. [Configure stack options] (スタックオプションの構成) ページで、[Next] (次へ) を選択します。 10.[確認] ページで、構成を確認したことを示すチェックボックスを選択します。 11.[スタックの作成] を選択してください。ソリューションコンポーネントがメンバー (レポーティング) アカウントにデプロイされると、スタックには CREATE_COMPLETE ステータスが表示されます。	

ソリューションをテストする

タスク	説明	必要なスキル
テスト前に EventBridge ルールが実行されていることを確認する。	少なくとも 24 時間待つか、CloudFormation テンプレートの template-reporting.yml ファイル内のレポート頻度を増やして、EventBridge ルールが実行されることを確認してください。 レポート頻度を増やすには	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	1. クローン作成したリポジトリにある template-reporting.yml ファイルを開きます。 2. 論理 ID 'LambdaSchedule' を持つイベントルールで、'ScheduleExpression'を探します。 3. 有効な cron 式が含まれるように 'ScheduleExpression' キーを編集してください。たとえば、次の cron 式では、5 分ごとにイベントルールが実行されるようスケジュールされます: “`cron(* /5 * * * *)`”	

タスク	説明	必要なスキル
生成されたレポートを Amazon S3 バケットで確認します。	1. メンバー (レポーティング) アカウントで AWS マネジメントコンソールにサインインし、CloudFormation コンソールを開きます。 2. [スタック] ペインで、作成したスタックの名前を選択します。次に、[リソース] タブを選択します。 3. [リソース] ペインの [ロジカル ID] 列で、BackupReports3Bucket を探します。次に、その論理 ID の横にある [物理 ID] 列のリンクを選択して、関連する Amazon S3 バケットを新しいタブで開きます。 4. バケットに次の形式で生成されたレポートが含まれていることを確認してください: BackupReports/<yyyy>/<mm>/<dd>/BackupReport-<BACKUP JOB STATUS>-<dd>-<Mon>-<yyyy>.csv	AWS DevOps、DevOps エンジニア

リソースのクリーンアップ

タスク	説明	必要なスキル
メンバー (レポーティング) アカウントからソリューションコンポーネントを削除します。	1. メンバー (レポーティング) アカウントで、ソリューションの Amazon S3 バケットを開きます。手順	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	については、このパターンの「ソリューションのテスト」セクションの「生成されたレポートを S3 バケットで確認する」のステップ 2 ~ 4 を参照してください。 2. バケットの内容を削除し、バケットを空にします。手順については、Amazon S3 ユーザーガイド の「バケットを空にする」を参照してください。 3. メンバー (レポーティング) アカウントで AWS マネジメントコンソールにサインインし、CloudFormation コンソールを開きます。 4. [スタック] ペインで、作成したスタックの名前の横にあるチェックボックスをオンにします。その後、[削除] をクリックします。	
管理アカウントからソリューションコンポーネントを削除します。	1. 管理アカウントで AWS マネジメントコンソールにサインインし、CloudFormation コンソールを開きます。 2. [スタック] ペインで、作成したスタックの名前の横にあるチェックボックスをオンにします。その後、[削除] をクリックします。	AWS DevOps、DevOps エンジニア

関連リソース

- チュートリアル: スケジュールされたイベントで AWS Lambda を使用する (AWS Lambda ドキュメント)
- AWS Lambda 関数を実行するスケジュールされたイベントの作成 (AWS SDK for JavaScript ドキュメント)
- IAM チュートリアル: IAM ロールを使用して AWS アカウント間でアクセスを委任する (IAM ドキュメント)
- AWS Organizations 用語と概念 (AWS Organizations ドキュメント)
- AWS Backup コンソールを使用したレポートプランの作成 (AWS Backup ドキュメント)AWS Backup
- 監査レポートを作成する (AWS Backup ドキュメント)
- オンデマンドレポートの作成 (AWS Backup ドキュメント)
- AWS Backup とは? (AWS Backup ドキュメント)
- AWS Backup を使用して AWS のサービス全体で大規模に一元化されたバックアップを自動化する (AWS ブログ記事)

Amazon EC2 インスタンスのリストのタグを CSV ファイルにエクスポートする

作成者：Sida Ju (AWS) と Pac Joonhyun (AWS)

概要

このパターンは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのリストのタグをプログラムで CSV ファイルにエクスポートする方法を示しています。

以下の Python スクリプトの例を使用すると、Amazon EC2 インスタンスを確認して特定のタグで分類するのにかかる時間を短縮できます。たとえば、このスクリプトを使用して、セキュリティチームがソフトウェアアップデートのフラグを付けたインスタンスのリストをすばやく特定して分類できます。

前提条件と制限

前提条件

- Python 3 がインストールされ、構成されている
- AWS コマンドラインインターフェイス (AWS CLI) がインストールされ、構成されている

制約事項

このパターンで指定される Python スクリプトの例では、以下の属性のみに基づいて Amazon EC2 インスタンスを検索できます。

- インスタンス ID
- プライベート IPv4 アドレス
- パブリック IPv4 アドレス

ツール

- 「Python」は汎用のコンピュータープログラミング言語です。
- virtualenv は、隔離された Python 環境を作成するのに役立ちます。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

コードリポジトリ

このパターンの Python スクリプトの例は、GitHub の search-ec2-instances-export-tags リポジトリにあります。

エピック

前提条件をインストールして設定する

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	**Note** AWS CLI コマンドの実行中にエラーが発生した場合は、最新の AWS CLI バージョンを使用していることを確認してください。 ターミナルウィンドウで次の Git コマンドを実行して、GitHub の search-ec2-instances-export-tags リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/search-ec2-instances-export-tags.git`	DevOps エンジニア
virtualenv をインストールして有効にします。	1. 次のコマンドを実行して virtualenv をインストールします。	DevOps エンジニア

タスク	説明	必要なスキル

	`python3 -m pip install virtualenv`
	2. 次のコマンドを実行して、新しい仮想化環境を作成します。
	`python3 -m venv env`
	3. 次のコマンドを実行して、新しい仮想化環境を有効にします。
	`source env/bin/activate`
	詳細については、「virtualenv ユーザーガイド」を参照してください。	

依存関係をインストールします。	1. ターミナルで次のコマンドを実行して、コードディレクトリを開きます。
	`cd search-ec2-instances-export-tags`
	2. 次の pip コマンドを実行して `requirements.txt` ファイルをインストールします。
	`pip3 install -r requirements.txt`
		DevOps エンジニア

タスク	説明	必要なスキル
AWS の名が付いたプロファイルを構成します。	まだ構成していない場合は、スクリプトを実行するために必要な認証情報を含む AWS の名前付きプロファイルを構成します。名前付きプロファイルを作成するには、 configure コマンドを実行します。 詳細については、AWS CLI ドキュメントの「名前付きプロファイルを使用する」を参照してください。	DevOps エンジニア

Python スクリプトを構成して実行する

タスク	説明	必要なスキル
入力ファイルを作成します。	スクリプトでタグを検索してエクスポートしたい Amazon EC2 インスタンスのリストを含む入力ファイルを作成します。インスタンス ID、プライベート IPv4 アドレス、またはパブリック IPv4 アドレスを一覧表示できます。 **Important** 各 Amazon EC2 インスタンスが入力ファイルの独自の行にリストされていることを確認します。	DevOps エンジニア

タスク	説明	必要なスキル
	入力データの例 ``` 1 i-0547c351bdfe85b9 f 2 54.157.194.156 3 172.31.85.33 4 54.165.198.144 5 i-0b6223b5914111a4 b 6 172.31.85.44 7 54.165.198.145 8 172.31.80.219 9 172.31.94.199 ```	

タスク	説明	必要なスキル
Python スクリプトを実行する。	ターミナルで次のコマンドを実行して、スクリプトを実行します。 `python search_instances.py -i INPUTFILE -o OUTPUTFILE -r REGION [-p PROFILE]` **Note** を入力ファイルの名前INPUTFILE に置き換えます。OUTPUTFILE を、CSV 出力ファイルの名前に置き換えます。REGION を Amazon EC2 リソースが存在する AWS リージョンに置き換えます。AWS の名前付きプロファイルを使用している場合は、PROFILE を使用している名前付きプロファイルに置き換えます。 サポートされているパラメータとその説明のリストを取得するには、次のコマンドを実行します。	DevOps エンジニア

タスク	説明	必要なスキル
	`python search_instances.py -h` 詳細および出力ファイルの例については、GitHub の search-ec2-instances-export-tags リポジトリにある README.md ファイルを参照してください。	

関連リソース

- AWS CLI の構成 (AWS CLI ユーザーガイド)

Troposphere を使用して AWS Config マネージドルールを含む AWS CloudFormation テンプレートを生成します

作成者:ルーカス・ネイション (AWS) とフレディ・ウィルソン (AWS)

概要

多くの組織は、「AWS Config マネージド」ルールを使用して、Amazon Web Services (AWS) リソースのコンプライアンスを一般的なベストプラクティスに照らして評価しています。ただし、これらのルールは保守に時間がかかる場合があり、このパターンは Python ライブラリである「Troposphere」を活用して AWS Config マネージドルールを生成および管理するのに役立ちます。

このパターンは、Python スクリプトを使用して AWS マネージドルールを含む Microsoft Excel スプレッドシートを AWS CloudFormation テンプレートに変換することで、AWS Config マネージドルールを管理するのに役立ちます。Troposphere は infrastructure as code (IaC) として機能します。つまり、JSON や YAML 形式のファイルを使用する代わりに、マネージドルールで Excel スプレッドシートを更新できるということです。次に、このテンプレートを使用して AWS CloudFormation スタックを起動し、AWS アカウントのマネージドルールを作成および更新します。

AWS CloudFormation テンプレートは Excel スプレッドシートを使用して各 AWS Config マネージドルールを定義し、AWS マネジメントコンソールで個々のルールを手動で作成する手間を省くのに役立ちます。このスクリプトでは、各マネージドルールのパラメータは空の辞書にデフォルト設定され、`ComplianceResourceTypes` スコープのデフォルトは `THE_RULE_IDENTIFIER.template file` からになります。ルール識別子の詳細については、AWS Config ドキュメントの「AWS CloudFormation テンプレートを使用した AWS Config マネージドルールの作成」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS CloudFormation テンプレートを使用して AWS Config マネージドルールを作成することに精通していること。これに関する詳細については、AWS Config ドキュメントの「AWS CloudFormation テンプレートを使用した AWS Config マネージドルールの作成」を参照してください。
- Python 3 がインストールされ、設定されています。詳細については、「Python ドキュメント」を参照してください。

- 既存の統合開発環境 (IDE)。
- サンプル `excel_config_rules.xlsx` Excel スプレッドシート (添付) の列で組織単位 (OU) を特定します。

エピック

AWS Config マネージドルールのカスタマイズと設定

タスク	説明	必要なスキル
サンプル Excel スプレッドシートを更新してください。	サンプル `excel_config_rules.xlsx` Excel スプレッドシート (添付) をダウンロードし、使用したい AWS Config マネージドルールを `Implemented` としてラベルを付けます。 `Implemented` とマークされたルールは AWS CloudFormation テンプレートに追加されます。	開発者
(オプション) config_rules_params.json ファイルを AWS Config ルールパラメータで更新します。	一部の AWS Config マネージドルールはパラメータを必要とするため、`--param-file` オプションを使用して JSON ファイルとして Python スクリプトに渡す必要があります。たとえば、`access-keys-rotated` マネージドルールは次の `maxAccessKeyAge` パラメータを使用します。 `{`	開発者

タスク	説明	必要なスキル
	"access-keys-rotated": { "InputParameters": { "maxAccessKeyAge": 90 } } } このサンプルパラメータでは、maxAccessKeyAge を 90 日に設定されています。スクリプトはパラメータファイルを読み取り、見つかった InputParameters をすべて追加します。	

タスク	説明	必要なスキル
(オプション) config_rules_params.json ファイルを AWS Config コンプライアンスリソースタイプで更新します。	デフォルトでは、Python スクリプトは AWS 定義のテンプレートから `ComplianceResourceTypes` を取得します。特定の AWS Config マネージドルールの範囲をオーバーライドする場合は、`--param-file` オプションを使用して JSON ファイルとして Python スクリプトに渡す必要があります。 たとえば、次のサンプルコードは、`ec2-volume-inuse-check` の `ComplianceResourceTypes` を `["AWS::EC2::Volume"]` リストに設定する方法を示しています。 `{ "ec2-volume-inuse-check": { "Scope": { "ComplianceResourceTypes": [ "AWS::EC2::Volume" ] } } }`	開発者

Python スクリプトを実行する。

タスク	説明	必要なスキル
requirements.txt ファイルから pip パッケージをインストールします。	`requirements.txt` ファイル (添付) をダウンロードし、IDE で次のコマンドを実行して Python パッケージをインストールします。 `pip3 install -r requirements.txt`	開発者
Python スクリプトを実行する。	1. 添付 `aws_config_rules.py` ファイルをローカルマシンにダウンロードします。 2. **Note** - `python3 aws_config_rules.py --ou <OU_NAME>` コマンドを実行します。: Excel スプレッドシートで選択する OU 列`--ou`を定義します。 また、次のオプションパラメータを含めることができます。 • `--config-rule-option` — Excel スプレッドシートから選択するルー	開発者

タスク	説明	必要なスキル
	ルを定義します。デフォルトは Implemented パラメータ。 • --excel-file — Excel スプレッドシートのパス。デフォルト: aws_config_rules.xlsx 。 • --param-file — パラメータ JSON ファイルのパス。デフォルト: config_rules_params.json 。 • --max-execution-frequency — AWS Config マネージドルールを評価する頻度を定義します。選択肢は One_Hour、Three_Hours 、Six_Hours 、Twelve_Hours 、または TwentyFour_Hours です。デフォルト: TwentyFour_Hours 。	

AWS Config マネージドルールをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation スタックを起動します。	1. AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開いて [スタックの作成] を選択します。	開発者

タスク	説明	必要なスキル
	2. [テンプレートの指定] ページで [テンプレートファイルをアップロード] を選択し、AWS CloudFormation テンプレートをアップロードします。 3. スタック名を指定し、[次へ] を選択します。 4. タグを指定し、[次へ] を選択します。 5. [スタックの作成] を選択してください。	

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

SageMaker ノートブックインスタンスに、別の AWS アカウントの CodeCommit リポジトリへの一時的なアクセスを許可する

作成者: Helge Aufderheide (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、Amazon SageMaker ノートブックインスタンスとユーザーに、別の AWS アカウントにある AWS CodeCommit リポジトリへの一時的なアクセスを許可する方法を示しています。このパターンはまた、各エンティティが各リポジトリで実行できる特定のアクションに対してきめ細かいアクセス権限を付与する方法も示しています。

組織は多くの場合、開発環境をホストするアカウントとは別の AWS アカウントに CodeCommit リポジトリを保存します。このマルチアカウント設定は、リポジトリへのアクセスを制御し、リポジトリを誤って削除するリスクを軽減するのに役立ちます。クロスアカウントアクセス権の付与には、AWS Identity and Access Management (IAM) ロールを使用するのがベストプラクティスです。これにより、各 AWS アカウントの事前定義済みの IAM アイデンティティが一時的にロールを引き受け、アカウント間で制御された信頼の連鎖を構築できます。

> **Note**
>
> 同様の手順を適用して、他の IAM ID に CodeCommit リポジトリへのクロスアカウントアクセスを許可できます。詳細については、「 CodeCommit ユーザーガイド」の「ロールを使用して CodeCommit リポジトリへのクロスアカウントアクセスを構成する」を参照してください。

前提条件と制限

前提条件

- CodeCommit リポジトリを持つアクティブな AWS アカウント (アカウント A)
- SageMaker ノートブックインスタンスを持つ 2 つ目のアクティブな AWS アカウント (アカウント B)
- アカウント A の IAM ロールを作成および変更するための十分な権限を持つ AWS ユーザー

- アカウント B の IAM ロールを作成および変更するための十分な権限を持つ 2 人目の AWS ユーザー

アーキテクチャ

以下の図は、SageMaker ノートブックインスタンスと 1 つの AWS アカウントのユーザーに CodeCommit リポジトリへのクロスアカウントアクセスを許可するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. アカウント B の AWS ユーザーロールと SageMaker ノートブックインスタンスロールは、名前付きプロファイルを前提としています。
2. 名前付きプロファイルのアクセス権限ポリシーは、プロファイルが引き受けるアカウント A の CodeCommit アクセスロールを指定します。
3. アカウント A の CodeCommit アクセスロールの信頼ポリシーにより、アカウント B の名前付きプロファイルが CodeCommit アクセスロールを引き受けます。
4. アカウント A の CodeCommit リポジトリの IAM アクセス権限ポリシーにより、CodeCommit アクセスロールは CodeCommit リポジトリにアクセスできます。

テクノロジースタック

- CodeCommit
- Git
- IAM
- pip
- SageMaker

ツール

- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

- Git は、ソフトウェア開発中のソースコードの変更を追跡するための分散型バージョン管理システムです。
- git-remote-codecommit は、Git を拡張することにより、CodeCommit リポジトリからコードをプッシュおよびプルするための簡単な方法を提供するユーティリティです。
- pip は Python のパッケージインストーラです。pip を使用して Python Package インデックスやその他のインデックスからパッケージをインストールできます。

ベストプラクティス

IAM ポリシーでアクセス許可を設定するときは、タスクの実行に必要なアクセス許可のみを付与するようにします。詳細については、IAM ドキュメントの最小特権アクセス許可を適用するを参照してください。

このパターンを実装する場合は、必ず以下を実行してください。

- IAM プリンシパルには、各リポジトリ内で必要な特定のアクションを実行するための権限のみが含まれていることを確認してください。たとえば、承認された IAM プリンシパルは変更を特定のリポジトリブランチにプッシュしてマージすることを許可しますが、保護されたブランチへのマージのみを要求することが推奨されます。
- IAM プリンシパルには、各プロジェクトのそれぞれのロールと責任に基づいて異なる IAM ロールが割り当てられていることを確認してください。たとえば、開発者はリリースマネージャーや AWS 管理者とは異なるアクセス権限を持ちます。

エピック

IAM ロールを構成する

タスク	説明	必要なスキル
CodeCommit のアクセスロールと権限ポリシーを構成します。	**Note** このエピックで説明されている手動セットアッププロセスを自動化するには、AWS CloudFormation テン	AWS 全般、AWS DevOps

タスク	説明	必要なスキル
	プレートを使用できます。 CodeCommit リポジトリを含むアカウント (アカウント A) で、次の操作を行います。 1. アカウント B の SageMaker ノートブックインスタンスロールが引き受けることができる IAM ロールを作成します。 2. リポジトリへのアクセスを許可する IAM ポリシーを作成し、そのポリシーをロールにアタッチします。テスト目的でのみ、AWSCodeCommitPowerUser AWS マネージドポリシーを選択してください。このポリシーは、リソースを削除する権限を除くすべての CodeCommit 権限を付与します。 3. ロールの信頼ポリシーを変更し、信頼できるエンティティとしてアカウント B がリストされるようにします。	

タスク	説明	必要なスキル
	Important この設定を本番環境に移行する前に、最小特権のアクセス許可を適用する独自の IAM ポリシーを作成することがベストプラクティスです。詳細については、このパターンの「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル
アカウント B の SageMaker ノートブックインスタンスのロールに、アカウント A の CodeCommit アクセスロールを引き受ける権限を付与します。	SageMaker ノートブックインスタンスの IAM ロールを含むアカウント (アカウント B) で、次の操作を行います。 1. IAM ロールまたはユーザーがアカウント A の CodeCommit アクセスロールを引き受けることを許可する IAM ポリシーを作成します。 IAM ロールまたはユーザーがクロスアカウントロールを引き受けることを許可する IAM アクセス権限ポリシーの例 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "VisualEditor0", "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "arn:aws:iam:::accountA_ID:role/accountArole_ID" } }` 2. アカウント B の SageMaker ノートブックイ	AWS 全般、AWS DevOps

タスク	説明	必要なスキル
	ンスタンスのロールにポリシーをアタッチします。 3. アカウント B の SageMaker ノートブックインスタンスのロールが、アカウント A の CodeCommit アクセスロールを引き受けるようにします。 **Note** リポジトリの Amazon リソースネーム (ARN) を表示するには、AWS CodeCommit ユーザーガイドの「CodeCommit リポジトリの詳細を表示する」を参照してください。	AWS CodeCommit

アカウント B に SageMaker ノートブックインスタンスを設定する

タスク	説明	必要なスキル
AWS SageMaker ノートブックインスタンスにユーザープロファイルを設定して、アカウント A のロールを引き受けます。	**Important** 最新バージョンの AWS コマンドラインインターフェイス (AWS CLI) がインス	AWS 全般、AWS DevOps

タスク	説明	必要なスキル
	トールされていること を確認します。 SageMaker ノートブックインスタンスを含むアカウント (アカウント B) で、次の操作を行います。 1. AWS マネジメントコンソールにサインインし、SageMaker コンソールを開きます。 2. SageMaker ノートブックインスタンスにアクセスします。Jupyter インターフェースが開きます。 3. [新規] を選択し、次に [ターミナル] を選択します。Jupyter 環境に新しいターミナルウィンドウが開きます。 4. SageMaker ノートブックインスタンスの ~/.aws/config ファイルに移動します。次に、以下のステートメントを入力してファイルにユーザープロファイルを追加します。 `------.aws/config-------------[profile remoterepouser]`	

タスク	説明	必要なスキル

	`role_arn = arn:aws:iam::<ID of Account A>:role/<rolename>`
	`role_session_name = remoteaccesssession`
	`region = eu-west-1`
	`credential_source = Ec2InstanceMetadata`
	`----------`
	`----------`	

git-remote-codecommit ユーティリティをインストールします。	「 CodeCommit ユーザーガイド」の「ステップ 2: git-remote-codecommit をインストールする」の指示に従ってください。	データサイエンティスト

リポジトリにアクセスする

タスク	説明	必要なスキル
Git コマンドまたは SageMaker を使用して CodeCommit リポジトリにアクセスする。	Git を使用するには これで、アカウント B で SageMaker ノートブックインスタンスのロールを引き受ける IAM プリンシパルは、Git コマンドを実行してアカウント A の CodeCommit リポジトリにアクセスできるようになりました。たとえば、ユーザーは、`git clone`、`git pull`、`git push` などのコマンドを実行できます。	Git、バッシュコンソール

タスク	説明	必要なスキル
	手順については、「AWS CodeCommit ユーザーガイド」の「AWS CodeCommit リポジトリに接続する」 を参照してください。 CodeCommit で Git を使用する方法については、「AWS CodeCommit ユーザーガイド」の「AWS CodeCommit 入門」を参照してください。 SageMaker を使用するには SageMaker コンソールから Git を使用するには、Git が CodeCommit リポジトリから認証情報を取得できるよう許可する必要があります。手順については、「SageMaker ドキュメント」の「別の アカウントの CodeCommit リポジトリをノートブックインスタンスに関連付ける」を参照してください。	

関連リソース

- ロールを使用して AWS CodeCommit リポジトリへのクロスアカウントアクセスを設定する (AWS CodeCommit ドキュメント)
- IAM チュートリアル: IAM ロールを使用して AWS アカウント間でアクセスを委任する (IAM ドキュメント)

追加情報

CodeCommit 権限を特定のアクションに制限する

IAM プリンシパルが CodeCommit リポジトリで実行できるアクションを制限するには、CodeCommit アクセスポリシーで許可されているアクションを変更します。

CodeCommit API の操作の詳細については、「AWS CodeCommit ユーザーガイド」の「CodeCommit アクセス権限リファレンス」を参照してください。

> **Note**
>
> ユースケースに合わせて AWSCodeCommitPowerUser AWS 管理ポリシーを編集することもできます。

CodeCommit 権限を特定のリポジトリに制限する

特定のユーザーだけが複数のコードリポジトリにアクセスできるマルチテナント環境を作成するには、次の操作を行います。

1. アカウント A に複数の CodeCommit アクセスロールを作成します。次に、アカウント B の特定のユーザーがロールを引き受けることができるように、各アクセスロールの信頼ポリシーを構成します。
2. 各 CodeCommit アクセスロールのポリシーに 「リソース」 条件を追加して、各ロールが引き受けることができるコードリポジトリを制限します。

特定の CodeCommit リポジトリに対する IAM プリンシパルのアクセスを制限する「リソース」条件の例

```
"Resource" : [<REPOSITORY_ARN>,<REPOSITORY_ARN> ]
```

> **Note**
>
> 同じ AWS アカウント内の複数のコードリポジトリを識別して区別するために、リポジトリの名前に異なるプレフィックスを割り当てることができます。たとえば、myproject-subproject1-repo1 や myproject-subproject2-repo1 など、異なる開発者グループに対応するプレフィックスを付けた名前をコードリポジトリに付けます。その後、割り当てられた

プレフィックスに基づいて、開発者グループごとに IAM ロールを作成できます。たとえば、myproject-subproject1-repoaccess という名前のロールを作成し、myproject-subproject1 というプレフィックスを含むすべてのコードリポジトリへのアクセスをそのロールに付与することができます。

特定のプレフィックスを含むコードリポジトリ ARN を参照する「リソース」条件の例

```
"Resource" : arn:aws:codecommit:<region>:<account-id>:myproject-subproject1-*
```

マルチアカウント DevOps 環境に GitHub Flow 分岐戦略を実装する

作成者: Mike Stephens (AWS) と Abhilash Vinod (AWS)

概要

ソースコードリポジトリを管理する場合、さまざまな分岐戦略が、開発チームが使用するソフトウェア開発およびリリースプロセスに影響します。一般的な分岐戦略の例としては、Trunk、GitHub Flow、Gitflow などがあります。これらの戦略では異なるブランチが使用され、各環境で実行されるアクティビティは異なります。DevOps プロセスを実装している組織は、ビジュアルガイドを活用して、これらの分岐戦略の違いを理解できます。このビジュアルを組織で使用すると、開発チームが作業を調整し、組織の標準に従うのに役立ちます。このパターンでは、このビジュアルを提供し、組織で GitHub Flow 分岐戦略を実装するプロセスについて説明します。

このパターンは、複数の を持つ組織の DevOps ブランチ戦略の選択と実装に関するドキュメントシリーズの一部です AWS アカウント。このシリーズは、クラウドでのエクスペリエンスを合理化するために、最初から正しい戦略とベストプラクティスを適用するのに役立つように設計されています。GitHub Flow は、組織が使用できる分岐戦略の 1 つにすぎません。このドキュメントシリーズでは、Trunk および Gitflow 分岐モデルについても説明します。まだ行っていない場合は、このパターンのガイダンスを実装する前に、マルチアカウント DevOps 環境の Git 分岐戦略の選択を確認することをお勧めします。デューデリジェンスを使用して、組織に適した分岐戦略を選択してください。

このガイドでは、組織が GitHub フロー戦略を実装する方法を示す図を示します。AWS Well-Architected DevOps ガイダンスを確認して、ベストプラクティスを確認することをお勧めします。このパターンには、DevOps プロセスの各ステップで推奨されるタスク、ステップ、制限が含まれます。

前提条件と制限

前提条件

- Git、インストール済み。これはソースコードリポジトリツールとして使用されます。
- Draw.io済み」。このアプリケーションは、図を表示および編集するために使用されます。

アーキテクチャ

ターゲット アーキテクチャ

次の図は、Punnett square (Wikipedia) のように使用できます。垂直軸のブランチを水平軸の AWS 環境と並べて、各シナリオで実行するアクションを決定します。数字は、ワークフロー内のアクショ

ンのシーケンスを示します。この例では、featureブランチから本番環境へのデプロイまでを案内します。

GitHub Flow アプローチの AWS アカウント、環境、ブランチの詳細については、「マルチアカウント DevOps 環境の Git ブランチ戦略の選択」を参照してください。

自動化とスケール

継続的インテグレーションと継続的デリバリー (CI/CD) は、ソフトウェアリリースのライフサイクルを自動化するプロセスです。これにより、本番環境への最初のコミットから新しいコードを取得するために従来必要とされていた手動プロセスの多くまたはすべてを自動化できます。CI/CD パイプラインには、サンドボックス、開発、テスト、ステージング、本番環境が含まれます。各環境では、CI/CD パイプラインはコードをデプロイまたはテストするために必要なインフラストラクチャをプロビジョニングします。CI/CD を使用することで、開発チームはコードを変更し、自動的にテストおよびデプロイできます。CI/CD パイプラインは、機能の受け入れとデプロイに一貫性、標準、ベストプラクティス、最小限の承認レベルを適用することで、開発チームのガバナンスとガードレールも提供します。詳細については、「継続的インテグレーションと継続的デリバリーの実践 AWS」を参照してください。

AWS は、CI/CD パイプラインの構築に役立つように設計された一連の開発者サービスを提供します。たとえば、 AWS CodePipelineは、リリースパイプラインを自動化して、迅速かつ信頼性の高いアプリケーションとインフラストラクチャの更新を行うのに役立つフルマネージドの継続的デリバリーサービスです。 は、ソースコードをAWS CodeBuildコンパイルし、テストを実行し、ready-to-deployソフトウェアパッケージを生成します。詳細については、「Developer Tools on AWS」を参照してください。

ツール

AWS サービスとツール

AWS には、このパターンの実装に使用できる一連の開発者サービスが用意されています。

- AWS CodeArtifact は、アプリケーション開発用のソフトウェアパッケージの保存と共有に役立つ、高度にスケーラブルなマネージドアーティファクトリポジトリサービスです。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。

- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、 AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。

その他のツール

- Draw.io 「https://」デスクトップは、フローチャートと図を作成するためのアプリケーションです。コードリポジトリには、Draw.iodrawio 形式の テンプレートが含まれています。
- Figma は、コラボレーション用に設計されたオンライン設計ツールです。コードリポジトリには、 Figma の .fig 形式のテンプレートが含まれています。

コードリポジトリ

このパターンの図のソースファイルは、GitHub Flow リポジトリの GitHub Git Branching Strategy で入手できます。これには、PNG、draw.io、および Figma 形式のファイルが含まれます。これらの図は、組織のプロセスをサポートするために変更できます。

ベストプラクティス

AWS Well-Architected DevOps ガイダンスのベストプラクティスと推奨事項に従い、マルチアカウント DevOps 環境の Git 分岐戦略を選択します。これらは、GitHub フローベースの開発を効果的に実装し、コラボレーションを促進し、コード品質を向上させ、開発プロセスを合理化するのに役立ちます。

エピック

GitHub フローワークフローの確認

タスク	説明	必要なスキル
標準の GitHub フロープロセスを確認します。	1. サンドボックス環境では、開発者は`feature`ブランチから`main`ブランチを作成し、命名パターン を使用します`feature/<ticket>_<initials>`	DevOps エンジニア

タスク	説明	必要なスキル
	_<short description> 。 2. 開発者はfeatureブランチに 1 つ以上のコミットを追加します。それぞれが個別の変更または改善を表します。 3. 開発者はマージリクエスト (MR) を開き、変更をmainブランチにマージします。これにより、レビュープロセスが開始されます。 4. レビュープロセス中に、開発者はコードの変更について話し合い、フィードバックを提供します。目標は、変更が高品質で、プロジェクトの標準を確実に満たすことです。 5. 開発者がマージリクエストを作成すると、自動ビルドプロセスが開始され、featureブランチの変更が開発環境にデプロイされます。 6. 自動テストは、マージリクエストにカプセル化された変更の整合性と品質を検証します。マージリクエストを完了するには、ビルドの成功、デプロイの成功、テストの成功が必要です。	

タスク	説明	必要なスキル
	7. レビュープロセスが完了すると、変更はmainブランチにマージされます。 8. 承認者は、テスト環境へのリリースアーティファクトのデプロイを手動で承認します。 9. 承認者は、リリースアーティファクトのステージング環境へのデプロイを手動で承認します。 10承認者は、リリースアーティファクトの本番環境へのデプロイを手動で承認します。	

タスク	説明	必要なスキル
バグ修正 GitHub フロープロセスを確認します。	1. 開発者はbugfixブランチからmainブランチを作成し、命名パターン を使用しますbugfix/<ticket number>_<developer initials>_<descriptor> 。 2. 開発者は問題を修正し、修正をコミットして、bugfixブランチを構築します。 3. 開発者はマージリクエストを開き、bugfixブランチをmainブランチにマージします。これにより、レビュープロセスが開始されます。 4. レビュープロセス中に、開発者はコードの変更について話し合い、フィードバックを提供します。 5. レビューが完了し、承認されると、デベロッパーはbugfixブランチのマージリクエストをmainブランチに完了します。 6. 承認者は、リリースアーティファクトの上位環境へのデプロイを手動で承認します。	DevOps エンジニア

タスク	説明	必要なスキル
ホットフィックス GitHub フロープロセスを確認します。	GitHub Flow は、コードの変更が頻繁に、確実に上位の環境にデプロイされる継続的な配信を可能にするように設計されています。重要なのは、すべてのfeatureブランチがいつでもデプロイ可能であることです。 Hotfix ブランチは、featureまたは bugfixブランチに似ていますが、これらの他のブランチと同じプロセスに従うことができます。ただし、緊急性を考慮すると、通常、ホットフィックスの優先度は高くなります。チームのポリシーと状況の即時性によっては、プロセスの特定のステップが迅速化される可能性があります。例えば、ホットフィックスのコードレビューは高速追跡される場合があります。したがって、ホットフィックスプロセスは機能またはバグフィックスプロセスと並行しますが、ホットフィックスを取り巻く緊急性により、手続きの遵守に変更が必要な場合があります。ホットフィックスが効率的かつ安全に処理されるように、ホットフィックスの管理に関	DevOps エンジニア

タスク	説明	必要なスキル
	するガイドラインを確立することが重要です。	

トラブルシューティング

問題	ソリューション
ブランチの競合	GitHub Flow モデルで発生する可能性がある一般的な問題は、本稼働環境でホットフィックスが発生する必要があるが、同じリソースが変更されている `feature`、`bugfix`、または `hotfix`ブランチで対応する変更が発生する必要があることです。へのマージ時に大きな競合を避けるため、 から変更`main`を下位ブランチに頻繁にマージすることをお勧めします`main`。
チーム成熟度	GitHub Flow は、真の継続的インテグレーションと継続的デリバリー (CI/CD) を採用して、より高い環境への毎日のデプロイを奨励します。チームには、機能を構築し、自動化テストを作成するためのエンジニアリング成熟度が必要です。チームは、変更が承認される前に、マージリクエストのレビューをすべて実行する必要があります。これにより、開発プロセスの品質、説明責任、効率性を促進する堅牢なエンジニアリング文化が育まれます。

関連リソース

このガイドには Git のトレーニングは含まれていませんが、このトレーニングが必要な場合は、インターネット上で利用できる質の高いリソースが多数あります。Git ドキュメントサイトから始めることをお勧めします。

以下のリソースは、 の GitHub Flow 分岐ジャーニーに役立ちます AWS クラウド。

AWS DevOps ガイダンス

- AWS DevOps ガイダンス
- AWS デプロイパイプラインリファレンスアーキテクチャ
- DevOps とは
- DevOps リソース

GitHub フローガイダンス

- GitHub フロークイックスタートチュートリアル (GitHub)
- GitHub Flow を使用する理由

その他のリソース

- 12 要素のアプリ方法 (12factor.net://www.com)

マルチアカウント DevOps 環境に Gitflow 分岐戦略を実装する

作成者: Mike Stephens (AWS)、Stephen DiCato (AWS)、Tim Wondergem (AWS)、Abhilash Vinod (AWS)

概要

ソースコードリポジトリを管理する場合、さまざまな分岐戦略が、開発チームが使用するソフトウェア開発およびリリースプロセスに影響します。一般的な分岐戦略の例としては、Trunk、Gitflow、GitHub Flow などがあります。これらの戦略では異なるブランチが使用され、各環境で実行されるアクティビティは異なります。DevOps プロセスを実装している組織は、ビジュアルガイドを活用して、これらの分岐戦略の違いを理解できます。このビジュアルを組織で使用すると、開発チームが作業を調整し、組織の標準に従うのに役立ちます。このパターンは、このビジュアルを提供し、組織内で Gitflow 分岐戦略を実装するプロセスについて説明します。

このパターンは、複数の を持つ組織の DevOps ブランチ戦略の選択と実装に関するドキュメントシリーズの一部です AWS アカウント。このシリーズは、クラウドでのエクスペリエンスを合理化するために、最初から正しい戦略とベストプラクティスを適用するのに役立つように設計されています。Gitflow は、組織が使用できる分岐戦略の 1 つにすぎません。このドキュメントシリーズでは、Trunk および GitHub Flow 分岐モデルについても説明します。まだ行っていない場合は、このパターンのガイダンスを実装する前に、マルチアカウント DevOps 環境の Git 分岐戦略の選択を確認することをお勧めします。デューデリジェンスを使用して、組織に適した分岐戦略を選択してください。

このガイドでは、組織が Gitflow 戦略をどのように実装するかを示す図を提供します。AWS Well-Architected DevOps ガイダンスを確認して、ベストプラクティスを確認することをお勧めします。このパターンには、DevOps プロセスの各ステップで推奨されるタスク、ステップ、制限が含まれます。

前提条件と制限

前提条件

- Git、インストール済み。これはソースコードリポジトリツールとして使用されます。
- Draw.ioインストール済みhttps://github.com/jgraph/drawio-desktop/releases」。このアプリケーションは、図を表示および編集するために使用されます。
- （オプション) Gitflow プラグインがインストールされています。

アーキテクチャ

ターゲット アーキテクチャ

次の図は、Punnett 二乗 (Wikipedia) のように使用できます。垂直軸のブランチを水平軸の AWS 環境と並べて、各シナリオで実行するアクションを決定します。数字は、ワークフロー内のアクションのシーケンスを示します。この例では、機能ブランチから本番環境へのデプロイまでを案内します。

Gitflow アプローチの AWS アカウント、環境、ブランチの詳細については、「マルチアカウント DevOps 環境の Git ブランチ戦略の選択」を参照してください。

自動化とスケール

継続的インテグレーションと継続的デリバリー (CI/CD) は、ソフトウェアリリースのライフサイクルを自動化するプロセスです。これにより、本番環境への最初のコミットから新しいコードを取得するために従来必要とされていた手動プロセスの多くまたはすべてを自動化できます。CI/CD パイプラインには、サンドボックス、開発、テスト、ステージング、本番環境が含まれます。各環境では、CI/CD パイプラインはコードをデプロイまたはテストするために必要なインフラストラクチャをプロビジョニングします。CI/CD を使用することで、開発チームはコードを変更し、自動的にテストおよびデプロイできます。CI/CD パイプラインは、機能の受け入れとデプロイに一貫性、標準、ベストプラクティス、最小限の承認レベルを適用することで、開発チームのガバナンスとガードレールも提供します。詳細については、「継続的インテグレーションと継続的デリバリーの実践 AWS」を参照してください。

AWS は、CI/CD パイプラインの構築に役立つように設計された一連の開発者サービスを提供します。たとえば、 AWS CodePipelineは、リリースパイプラインを自動化して、迅速かつ信頼性の高いアプリケーションとインフラストラクチャの更新を行うのに役立つフルマネージドの継続的デリバリーサービスです。 は、ソースコードをAWS CodeBuildコンパイルし、テストを実行し、ready-to-deployソフトウェアパッケージを生成します。詳細については、「Developer Tools on AWS」を参照してください。

ツール

AWS サービスとツール

AWS は、このパターンを実装するために使用できる一連の開発者サービスを提供します。

- AWS CodeArtifact は、アプリケーション開発用のソフトウェアパッケージの保存と共有に役立つ、高度にスケーラブルなマネージドアーティファクトリポジトリサービスです。

- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、 AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。

その他のツール

- Draw.io 「https://」デスクトップは、フローチャートと図を作成するためのアプリケーションです。コードリポジトリには、Draw.iodrawio 形式の テンプレートが含まれています。
- Figma は、コラボレーション用に設計されたオンライン設計ツールです。コードリポジトリには、 Figma の .fig 形式のテンプレートが含まれています。
- （オプション) Gitflow プラグインは、Gitflow 分岐モデルに高レベルのリポジトリオペレーションを提供する Git 拡張機能のコレクションです。

コードリポジトリ

このパターンの図のこのソースファイルは、GitFlow リポジトリの GitHub Git Branching Strategy GitFlowで入手できます。これには、PNG、draw.io、および Figma 形式のファイルが含まれます。これらの図は、組織のプロセスをサポートするために変更できます。

ベストプラクティス

AWS Well-Architected DevOps Guidance のベストプラクティスと推奨事項に従い、マルチアカウント DevOps 環境の Git 分岐戦略を選択します。これらは、Gitflow ベースの開発を効果的に実装し、コラボレーションを促進し、コード品質を向上させ、開発プロセスを合理化するのに役立ちます。

エピック

Gitflow ワークフローの確認

タスク	説明	必要なスキル
標準の Gitflow プロセスを確認します。	1. サンドボックス環境では、開発者はfeatureブランチからdevelopブランチを作成し、命名パターンを使用しますfeature/<ticket>_<initials>_<short description> 。 2. デベロッパーは、チケットを完了するためにコードを開発し、サンドボックス環境にコードを反復的にデプロイします。 **Note** 開発者は、オプションでsandboxブランチを作成して、サンドボックス環境で自動ビルドまたはデプロイパイプラインを実行できます。 3. デベロッパーは、スカッシュマージを使用してfeature、ブランチからdevelopブランチへのマージリクエストを作成します。	DevOps エンジニア

タスク	説明	必要なスキル
	4. 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインは、developブランチを自動的に構築し、開発環境にデプロイします。 5. (オプション) デベロッパーは、リリースアクティビティを続行する前に、追加のfeatureブランチを開発ブランチに統合します。 6. develop ブランチの機能をリリースする準備ができたら、開発者はreleaseブランチrelease/v <number> から という名前のdevelopブランチを作成します。 7. 開発者はリリースブランチを構築し、アーティファクトを発行して他の環境間で再利用します。 8. 承認者は、テスト環境へのリリースアーティファクトのデプロイを手動で承認します。 9. 承認者は、リリースアーティファクトのステージング環境へのデプロイを手動で承認します。 10承認者は、リリースアーティファクトの本番環境へのデプロイを手動で承認します。	

タスク	説明	必要なスキル
	11.デベロッパーはreleaseブランチをmainブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。 12.デベロッパーはreleaseブランチをdevelopブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。	

タスク	説明	必要なスキル
修正プログラム Gitflow プロセスを確認します。	1. 開発者はhotfixブランチからmainブランチを作成し、命名パターン を使用しますhotfix/<ticket>_<initials>_<short description> 。 2. 開発者はreleaseブランチからmainブランチを作成し、 という名前を付けますrelease/v<number> 。 3. 開発者は問題を修正し、修正をコミットして、hotfixブランチを構築します。 4. デベロッパーは、スカッシュマージを使用してhotfix、ブランチからrelease/v<number> ブランチへのマージリクエストを作成します。 5. デベロッパーはreleaseブランチを構築し、アーティファクトを発行して他の環境間で再利用します。 6. 承認者は、テスト環境へのリリースアーティファクトのデプロイを手動で承認します。 7. 承認者は、リリースアーティファクトのステージン	DevOps エンジニア

タスク	説明	必要なスキル
	グ環境へのデプロイを手動で承認します。 8. 承認者は、リリースアーティファクトの本番環境へのデプロイを手動で承認します。 9. デベロッパーは`release`ブランチを`main`ブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。 10.デベロッパーは`release`ブランチを`develop`ブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。 11競合が検出されると、デベロッパーはアラートを受け取り、マージリクエストとの競合を解決します。	

タスク	説明	必要なスキル
バグ修正 Gitflow プロセスを確認します。	1. 開発者は現在のbugfixブランチからrelease/v<number> ブランチを作成し、命名パターン を使用しますbugfix/<ticket number>_<developer initials>_<descriptor> 。 2. 開発者は問題を修正し、修正をコミットして、bugfixブランチを構築します。 3. デベロッパーは、スカッシュマージを使用してbugfix、ブランチからrelease/v<number> ブランチへのマージリクエストを作成します。 4. デベロッパーはreleaseブランチを構築し、アーティファクトを発行して他の環境間で再利用します。 5. 承認者は、テスト環境へのリリースアーティファクトのデプロイを手動で承認します。 6. 承認者は、ステージ環境へのリリースアーティファクトのデプロイを手動で承認します。 7. 承認者は、リリースアーティファクトの本番環境へ	DevOps エンジニア

タスク	説明	必要なスキル
	のデプロイを手動で承認します。 8. デベロッパーはreleaseブランチをmainブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。 9. デベロッパーはreleaseブランチをdevelopブランチにマージします。開発者は自動スクリプトを使用して早送りマージを実行するのが理想的です。スカッシュマージを使用しないでください。 10競合が検出されると、デベロッパーはアラートを受け取り、マージリクエストとの競合を解決します。	

トラブルシューティング

問題	ソリューション
ブランチの競合	Gitflow モデルで発生する可能性がある一般的な問題は、本稼働環境でホットフィックスが発生する必要があるが、対応する変更が、別のブランチが同じリソースを変更しているより低い環境で発生する必要があることです。一度にアクティブなリリースブランチは 1 つだけにす

問題	ソリューション
	ることをお勧めします。一度に複数のアクティブな がある場合、環境の変更が衝突し、ブランチを本番環境に移行できない可能性があります。
マージ	リリースはメインにマージし直して、できるだけ早く開発して、作業をプライマリブランチに統合する必要があります。
スカッシュマージ	ブランチからfeatureブランチにマージする場合にのみ、スカッシュマージを使用しますdevelop。上位ブランチでスカッシュマージを使用すると、変更を下位ブランチにマージするときに問題が発生します。

関連リソース

このガイドには Git のトレーニングは含まれていませんが、このトレーニングが必要な場合は、インターネット上で利用できる質の高いリソースが多数あります。Git ドキュメントサイトから始めることをお勧めします。

以下のリソースは、 での Gitflow 分岐ジャーニーに役立ちます AWS クラウド。

AWS DevOps ガイダンス

- AWS DevOps ガイダンス
- AWS デプロイパイプラインリファレンスアーキテクチャ
- DevOps とは
- DevOps リソース

Gitflow ガイダンス

- 元の Gitflow ブログ (Vincent Driessen ブログ記事)
- Gitflow ワークフロー (アトラシアン)

- GitHub での Gitflow: GitHub ベースのリポジトリで GitHub Flow ワークフローを使用する方法 (YouTube 動画)
- Git Flow Init の例 (YouTube 動画)
- Gitflow リリースブランチの開始から終了まで (YouTube 動画)

その他のリソース

12 要素のアプリ方法 (12factor.net://www.com)

マルチアカウント DevOps 環境のトランク分岐戦略を実装する

作成者: Mike Stephens (AWS) と Rayjan Wilson (AWS)

概要

ソースコードリポジトリを管理する場合、さまざまな分岐戦略が、開発チームが使用するソフトウェア開発およびリリースプロセスに影響します。一般的な分岐戦略の例としては、Trunk、GitHub Flow、Gitflow などがあります。これらの戦略では異なるブランチが使用され、各環境で実行されるアクティビティは異なります。DevOps プロセスを実装している組織は、ビジュアルガイドを活用して、これらの分岐戦略の違いを理解できます。このビジュアルを組織で使用すると、開発チームが作業を調整し、組織の標準に従うのに役立ちます。このパターンでは、このビジュアルを提供し、組織でトランク分岐戦略を実装するプロセスについて説明します。

このパターンは、複数の を持つ組織の DevOps ブランチ戦略の選択と実装に関するドキュメントシリーズの一部です AWS アカウント。このシリーズは、クラウドでのエクスペリエンスを合理化するために、最初から正しい戦略とベストプラクティスを適用するのに役立つように設計されています。トランクは、組織が使用できる分岐戦略の 1 つにすぎません。このドキュメントシリーズでは、GitHub Flow および Gitflow 分岐モデルについても説明します。まだ行っていない場合は、このパターンのガイダンスを実装する前に、マルチアカウント DevOps 環境の Git 分岐戦略の選択を確認することをお勧めします。デューデリジェンスを使用して、組織に適した分岐戦略を選択してください。

このガイドでは、組織がトランク戦略を実装する方法を示す図を示します。WellAWS -Architected DevOps の公式ガイダンスを確認して、ベストプラクティスを確認することをお勧めします。このパターンには、DevOps プロセスの各ステップで推奨されるタスク、ステップ、制限が含まれます。

前提条件と制限

前提条件

- Git、インストール済み。これはソースコードリポジトリツールとして使用されます。
- Draw.io済み」。このアプリケーションは、図を表示および編集するために使用されます。

アーキテクチャ

ターゲット アーキテクチャ

次の図は、Punnett square (Wikipedia) のように使用できます。垂直軸のブランチを水平軸の AWS 環境と並べて、各シナリオで実行するアクションを決定します。数字は、ワークフロー内のアクショ

ンのシーケンスを示します。この例では、featureブランチから本番環境へのデプロイまでを案内します。

トランクアプローチの AWS アカウント、環境、ブランチの詳細については、「マルチアカウント DevOps 環境の Git ブランチ戦略の選択」を参照してください。

自動化とスケール

継続的インテグレーションと継続的デリバリー (CI/CD) は、ソフトウェアリリースのライフサイクルを自動化するプロセスです。これにより、本番環境への最初のコミットから新しいコードを取得するために従来必要とされていた手動プロセスの多くまたはすべてを自動化できます。CI/CD パイプラインには、サンドボックス、開発、テスト、ステージング、本番環境が含まれます。各環境では、CI/CD パイプラインはコードをデプロイまたはテストするために必要なインフラストラクチャをプロビジョニングします。CI/CD を使用することで、開発チームはコードを変更し、自動的にテストおよびデプロイできます。CI/CD パイプラインは、機能の受け入れとデプロイに一貫性、標準、ベストプラクティス、最小限の承認レベルを適用することで、開発チームのガバナンスとガードレールも提供します。詳細については、「継続的インテグレーションと継続的デリバリーの実践 AWS」を参照してください。

AWS は、CI/CD パイプラインの構築に役立つように設計された一連の開発者サービスを提供します。たとえば、 AWS CodePipelineは、リリースパイプラインを自動化して、迅速かつ信頼性の高いアプリケーションとインフラストラクチャの更新を行うのに役立つフルマネージドの継続的デリバリーサービスです。 は、ソースコードをAWS CodeBuildコンパイルし、テストを実行し、ready-to-deployソフトウェアパッケージを生成します。詳細については、「Developer Tools on AWS」を参照してください。

ツール

AWS サービスとツール

AWS には、このパターンの実装に使用できる一連の開発者サービスが用意されています。

- AWS CodeArtifact は、アプリケーション開発用のソフトウェアパッケージの保存と共有に役立つ、高度にスケーラブルなマネージドアーティファクトリポジトリサービスです。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。

- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、 AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。

その他のツール

- Draw.io 「https://」デスクトップ – フローチャートと図を作成するためのアプリケーション。
- Figma は、コラボレーション用に設計されたオンライン設計ツールです。コードリポジトリには、 Figma の .fig 形式のテンプレートが含まれています。

コードリポジトリ

このパターンの図のソースファイルは、GitHub Git Branching Strategy for Trunk リポジトリで入手できます。これには、PNG、draw.io、および Figma 形式のファイルが含まれます。これらの図は、組織のプロセスをサポートするために変更できます。

ベストプラクティス

AWS Well-Architected DevOps ガイダンスのベストプラクティスと推奨事項に従い、マルチアカウント DevOps 環境の Git 分岐戦略を選択します。これらは、トランクベースの開発の効果的な実装、コラボレーションの促進、コード品質の向上、開発プロセスの合理化に役立ちます。

エピック

トランクワークフローの確認

タスク	説明	必要なスキル
標準の Trunk プロセスを確認します。	1. サンドボックス環境では、開発者は`feature`ブランチから`main`ブランチを作成し、命名パターン を使用します`feature/<ticket>_<initials>_<short description>` 。	DevOps エンジニア

タスク	説明	必要なスキル
	2. デベロッパーは、チケットを完了するためにコードを開発し、サンドボックス環境にコードを反復的にデプロイします。 **Note** 開発者は、オプションでsandboxブランチを作成して、サンドボックス環境で自動ビルドまたはデプロイパイプラインを実行できます。 3. デベロッパーは、スカッシュマージを使用してfeature、ブランチからmainブランチへのマージリクエストを作成します。 4. 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインは、mainブランチから開発環境にアーティファクトを自動的に構築して公開します。 5. 承認者は、リリースアーティファクトの開発環境へのデプロイを手動で承認します。 6. 承認者は、テスト環境へのリリースアーティファクト	

タスク	説明	必要なスキル
	のデプロイを手動で承認します。 7. 承認者は、リリースアーティファクトのステージング環境へのデプロイを手動で承認します。 8. 承認者は、リリースアーティファクトの本番環境へのデプロイを手動で承認します。	

トラブルシューティング

問題	ソリューション
ブランチの競合	トランクモデルで発生する可能性がある一般的な問題は、本番稼働環境でホットフィックスが発生する必要があるが、同じリソースが変更されているfeatureブランチで対応する変更が発生する必要があることです。へのマージに伴う大きな競合を避けるため、 から変更mainを下位ブランチに頻繁にマージすることをお勧めしますmain。

関連リソース

このガイドには Git のトレーニングは含まれていませんが、このトレーニングが必要な場合は、インターネット上で利用できる質の高いリソースが多数あります。Git ドキュメントサイトから始めることをお勧めします。

以下のリソースは、 でのトランク分岐ジャーニーに役立ちます AWS クラウド。

AWS DevOps ガイダンス

- AWS DevOps ガイダンス

- AWS Deployment Pipeline Reference Architecture
- DevOps とは
- DevOps リソース

トランクガイダンス

- トランクベースの開発

その他のリソース

- 12 要素のアプリ方法 (12factor.net://www.com)

AWS インフラストラクチャをデプロイする前に、一元化されたカスタム Checkov スキャンを実装してポリシーを適用する

作成者: Benjamin Morris (AWS)

概要

このパターンは、GitHub 組織全体で再利用できる 1 つのリポジトリにカスタム Checkov ポリシーを記述するための GitHub Actions フレームワークを提供します。このパターンに従うことで、情報セキュリティチームは会社の要件に基づいてカスタムポリシーを記述、追加、維持できます。カスタムポリシーは、GitHub 組織内のすべてのパイプラインに自動的にプルできます。このアプローチを使用して、リソースがデプロイされる前に、リソースに会社の標準を適用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- GitHub Actions を使用する GitHub 組織
- AWS HashiCorp Terraform または AWS CloudFormation

機能制限

- このパターンは GitHub Actions 用に記述されています。ただし、GitLab などの同様の継続的インテグレーションおよび継続的デリバリー (CI/CD) フレームワークに適応できます。GitHub の特定の有料バージョンは必要ありません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、 AWS ドキュメントの「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

このパターンは、GitHub 再利用可能なワークフローとカスタム Checkov ポリシーを含む GitHub リポジトリとしてデプロイされるように設計されています。再利用可能なワークフローでは、Terraform と CloudFormation の両方の infrastructure as code (IaC) リポジトリをスキャンできます。

次の図は、Reusable GitHub ワークフローリポジトリとカスタム Checkov ポリシーリポジトリを個別のアイコンとして示しています。ただし、これらのリポジトリは個別のリポジトリまたは単一のリポジトリとして実装できます。このサンプルコードでは、ワークフロー用の ファイル (`.github/workflows`) とカスタムポリシー用の ファイル (`custom_policies` フォルダと `.checkov.yml` 設定ファイル) を含む単一のリポジトリを同じリポジトリで使用します。

この図表は、次のワークフローを示しています：

1. ユーザーは GitHub リポジトリにプルリクエストを作成します。
2. パイプラインワークフローは、Checkov 再利用可能なワークフローへの参照を含め、GitHub Actions で開始されます。
3. パイプラインワークフローは、外部リポジトリから参照された Checkov 再利用可能なワークフローをダウンロードし、GitHub Actions を使用してその Checkov ワークフローを実行します。
4. Checkov の再利用可能なワークフローは、外部リポジトリからカスタムポリシーをダウンロードします。
5. Checkov の再利用可能なワークフローは、組み込みの Checkov ポリシーとカスタム Checkov ポリシーの両方に対して GitHub リポジトリの IaC を評価します。Checkov の再利用可能なワークフローは、セキュリティの問題が見つかったかどうかに基づいて合格または不合格になります。

自動化とスケール

このパターンにより、Checkov 設定を一元管理できるため、ポリシーの更新を 1 か所に適用できます。ただし、このパターンでは、各リポジトリが中央の再利用可能なワークフローへの参照を含むワークフローを使用する必要があります。このリファレンスを手動で追加することも、スクリプトを使用して各リポジトリの `.github/workflows`フォルダにファイルをプッシュすることもできます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。Checkov は CloudFormation をスキャンできます。

その他のツール

- Checkov は、IaC のセキュリティとコンプライアンスの設定ミスをチェックする静的コード分析ツールです。
- GitHub Actions は GitHub プラットフォームに統合されており、GitHub リポジトリ内のワークフローの作成、共有、実行に役立ちます。GitHub Actions を使用して、コードの構築、テスト、デプロイなどのタスクを自動化できます。
- Terraform は HashiCorp の IaC ツールで、クラウドおよびオンプレミスのリソースの作成と管理に役立ちます。Checkov は Terraform をスキャンできます。

コードリポジトリ

このパターンのコードは、GitHub centralized-custom-checkov-sast リポジトリで入手できます。

ベストプラクティス

- 一貫したセキュリティ体制を維持するには、会社のセキュリティポリシーを Checkov ポリシーと整合させます。
- Checkov カスタムポリシーの実装の初期段階では、Checkov スキャンでソフトフェイルオプションを使用して、セキュリティ問題のある IaC をマージできます。プロセスが成熟したら、ソフトフェイルオプションからハードフェイルオプションに切り替えます。

エピック

カスタムポリシー用の中央 Checkov リポジトリを作成する

タスク	説明	必要なスキル
中央 Checkov リポジトリを作成します。	組織内で使用されるカスタム Checkov ポリシーを保存するリポジトリを作成します。 クイックスタートとして、このパターンの GitHub centralized-custom-checkov-sastリポジトリの内容を中央の Checkov リポジトリにコピーできます。	DevOps エンジニア

タスク	説明	必要なスキル
再利用可能なワークフロー用のリポジトリを作成します。	再利用可能なワークフローのリポジトリが既に存在する場合、またはカスタム Checkov ポリシーと同じリポジトリに再利用可能なワークフローファイルを含める場合は、このステップをスキップできます。 再利用可能なワークフローを保持する GitHub リポジトリを作成します。他のリポジトリのパイプラインはこのリポジトリを参照します。	DevOps エンジニア

再利用可能な Checkov ワークフローとサンプル Checkov ワークフローを作成する

タスク	説明	必要なスキル
再利用可能な Checkov ワークフローを追加します。	再利用可能なワークフローリポジトリに再利用可能な Checkov GitHub Actions ワークフロー (YAML ファイル) を作成します。このパターンで提供されているワークフローファイルから、この再利用可能なワークフローを適応させることができます。 変更の例は、再利用可能なワークフローを変更してソフトフェイルオプションを使用することです。`soft-fail` を に設定すると、Checkov スキャンが失敗した	DevOps エンジニア

タスク	説明	必要なスキル
	場合でもジョブを正常に完了trueできます。手順については、Checkov ドキュメントの「ハードフェイルとソフトフェイル」を参照してください。	
ワークフローの例を追加します。	ワークフローを参照する Checkov reusable ワークフローの例を追加します。これにより、reusableワークフローを再利用するためのテンプレートが提供されます。サンプルリポジトリでは、 checkov-source.yaml が再利用可能なワークフローであり、 が を消費する例checkov-scan.yaml ですcheckov-source 。 Checkov ワークフローの例の記述の詳細については、「追加情報」を参照してください。	DevOps エンジニア

会社ポリシーを Checkov カスタムポリシーに関連付ける

タスク	説明	必要なスキル
Checkov で適用できるポリシーを決定します。	1. インフラストラクチャのセキュリティに関連する企業ポリシーと、実施すべき要件を確認します。	セキュリティおよびコンプライアンス

タスク	説明	必要なスキル
	2. Checkov カスタムポリシーを使用して、実装できる要件を決定します。 3. ポリシーコントロールを Checkov カスタムポリシーにマッピングする命名規則を作成します。通常、Checkov カスタムポリシーには、Checkov 名、ポリシーソース (カスタム) 、およびポリシー番号 (例:) を持つ識別子がありますCKV2_CUSTOM_123 。 Checkov カスタムポリシーの作成の詳細については、Checkov ドキュメントの「カスタムポリシーの概要」を参照してください。	
Checkov カスタムポリシーを追加します。	特定された企業ポリシーを中央リポジトリのカスタム Checkov ポリシーに変換します。シンプルな Checkov ポリシーは、Python または YAML で記述できます。	セキュリティ

一元化された Checkov カスタムポリシーを実装する

タスク	説明	必要なスキル
Checkov 再利用可能なワークフローをすべてのリポジトリに追加します。	この時点で、再利用可能なワークフローを参照する Checkov ワークフローの例が	DevOps エンジニア

タスク	説明	必要なスキル
	必要です。再利用可能なワークフローを参照するサンプル Checkov ワークフローを、それを必要とする各リポジトリにコピーします。	
マージ前に Checkov が実行されるようにメカニズムを作成します。	プルリクエストごとに Checkov ワークフローが実行されるようにするには、プルリクエストをマージする前に成功した Checkov ワークフローを必要とするステータスチェックを作成します。GitHub では、プルリクエストをマージする前に、特定のワークフローの実行を要求できます。	DevOps エンジニア

タスク	説明	必要なスキル
組織全体の PAT を作成し、シークレットとして共有します。	GitHub 組織が公開されている場合は、このステップをスキップできます。 このパターンでは、Checkov ワークフローが GitHub 組織のカスタムポリシーリポジトリからカスタムポリシーをダウンロードできることが必要です。Checkov ワークフローがそれらのリポジトリにアクセスできるようにアクセス許可を提供する必要があります。 これを行うには、組織リポジトリを読み取るアクセス許可を持つ個人用アクセストークン (PAT) を作成します。この PAT を、組織全体のシークレット (有料プランの場合) または各リポジトリのシークレット (無料バージョン) としてリポジトリと共有します。サンプルコードでは、シークレットのデフォルト名は ですORG_PAT。	DevOps エンジニア

タスク	説明	必要なスキル
(オプション) Checkov ワークフローファイルを変更から保護します。	Checkov ワークフローファイルを不要な変更から保護するには、 CODEOWNERS ファイルを使用できます。CODEOWNERS ファイルは、通常、 ディレクトリのルートにデプロイされます。 たとえば、checkov-scan.yaml ファイルの変更時に GitHub 組織のsecEngグループからの承認を要求するには、リポジトリのCODEOWNERS ファイルに以下を追加します。 `[Checkov] .github/workflows/checkov-scan.yaml @myOrg/secEng` CODEOWNERS ファイルは、保存先のリポジトリに固有です。リポジトリで使用される Checkov ワークフローを保護するには、各リポジトリにCODEOWNERS ファイルを追加 (または更新) する必要があります。 Checkov ワークフローファイルの保護の詳細については、「追加情報」を参照してください。CODEOWNERS ファイルの詳細については、CI/CD プロバイダー	DevOps エンジニア

タスク	説明	必要なスキル
	(GitHub など) の公式ドキュメントを参照してください。	

関連リソース

- Checkov カスタムポリシーの概要
- CloudFormation 設定スキャン
- GitHub Actions 再利用可能なワークフロー

追加情報

Checkov ワークフローファイルの書き込み

を記述するときはcheckov-scan.yaml、いつ実行するかを検討してください。最上位onキーは、ワークフローの実行時期を決定します。サンプルリポジトリでは、mainブランチをターゲットとするプルリクエストがある場合 (およびプルリクエストのソースブランチが変更された場合) 、ワークフローが実行されます。ワークフローは、 workflow_dispatchキーのために必要に応じて実行することもできます。

ワークフローを実行する頻度に基づいて、ワークフロートリガー条件を変更できます。たとえば、を pull_requestに置き換えてbranchesキーpushを削除することで、コードがブランチにプッシュされるたびにワークフローを実行するように変更できます。

個々のリポジトリ内で作成したワークフローファイルの例を変更できます。たとえば、リポジトリがブランチを中心に構造化productionされている場合、ターゲットproductionブランチの名前をから mainに調整できます。

Checkov ワークフローファイルの保護

Checkov スキャンは、潜在的なセキュリティ設定ミスに関する有用な情報を提供します。ただし、一部のデベロッパーは、生産性の障壁であると認識し、スキャンワークフローを削除または無効にしようとする場合があります。

この問題に対処するには、セキュリティスキャンの長期的な価値に関するメッセージの改善や、安全なインフラストラクチャのデプロイ方法に関する明確なドキュメントの作成など、いくつかの方法があります。これらは、DevSecOps コラボレーションに対する重要な「ソフト」アプローチであり、

この問題の根本原因の解決策と見なすことができます。ただし、CODEOWNERSファイルなどの技術的なコントロールをガードレールとして使用して、開発者を正しい道に導くこともできます。

サンドボックスでのパターンのテスト

サンドボックス環境でこのパターンをテストするには、次の手順に従います。

1. 新しい GitHub 組織を作成します。組織内のすべてのリポジトリへの読み取り専用アクセス権を持つトークンを作成します。このトークンは有料環境ではなくサンドボックス環境用であるため、このトークンを組織全体のシークレットに保存することはできません。
2. Checkov 設定を保持するcheckovリポジトリと、再利用可能なワークフロー設定を保持するgithub-workflowsリポジトリを作成します。リポジトリにサンプルリポジトリの内容を入力します。
3. アプリケーションリポジトリを作成し、checkov-scan.yamlワークフローをコピーしてその.github/workflowsフォルダに貼り付けます。組織の読み取り専用アクセス用に作成した PAT を含むシークレットをリポジトリに追加します。デフォルトのシークレットは ですORG_PAT。
4. アプリケーションリポジトリに Terraform または CloudFormation コードを追加するプルリクエストを作成します。Checkov はスキャンして結果を返す必要があります。

CodeCommit 内のモノリポジトリの変更を自動的に検出し、さまざまな CodePipeline パイプラインを開始する

作成者: Helton ribeiro (AWS)、Petrus Batalha (AWS)、Ricardo Morais (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら

このパターンは、 でモノレポベースのアプリケーションのソースコードの変更を自動的に検出し AWS CodeCommit 、マイクロサービスごとに継続的インテグレーションと継続的デリバリー (CI/CD) の自動化 AWS CodePipeline を実行するパイプラインを で開始するのに役立ちます。このアプローチは、モノレポベースのアプリケーション内の各マイクロサービスに専用の CI/CD パイプラインを持たせることができることを意味します。これにより、可視性が向上し、コードの共有が容易になり、コラボレーション、標準化、発見が容易になります。

このパターンで説明されているソリューションは、モノレポ内のマイクロサービス間で依存関係分析を実行しません。ソースコードの変更のみを検出し、一致する CI/CD パイプラインを開始します。

このパターンでは、 を統合開発環境 (IDE) AWS Cloud9 として使用し AWS Cloud Development Kit (AWS CDK) 、 `MonoRepoStack`と の 2 つの AWS CloudFormation スタックを使用してインフラストラクチャを定義します`PipelinesStack`。`MonoRepoStack` スタックは、 にモノリポジトリ AWS CodeCommit を作成し、CI/CD パイプラインを開始する AWS Lambda 関数を作成します。`PipelinesStack` スタックはパイプラインインフラストラクチャを定義します。

> **Important**
>
> このパターンのワークフローは概念実証 (PoC) です。テスト環境でのみ使用することをお勧めします。このパターンのアプローチを本番環境で使用する場合は、 AWS Identity and Access Management （ IAM) ドキュメントの「IAM のセキュリティのベストプラクティス」を参照して、IAM ロールと に必要な変更を加えます AWS のサービス。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み。詳細については、AWS CLI AWS CLIドキュメントの「 のインストール、更新、アンインストール」を参照してください。
- ローカルマシンにインストール済みの Python 3 と `pip`。詳細については、Python のドキュメントを参照してください。
- AWS CDK、インストールおよび設定済み。詳細については、 AWS CDK ドキュメントの「 の開始 AWS CDK方法」を参照してください。
- IDE AWS Cloud9 がインストールされ、設定されている。詳細については、 AWS Cloud9 ドキュメントの「セットアップ AWS Cloud9」を参照してください。
- GitHub AWS CodeCommit モノレポマルチパイプライントリガーリポジトリは、ローカルマシンでクローンされます。
- CodePipeline でビルドしてデプロイするアプリケーションコードを含む既存のディレクトリ。
- での DevOps のベストプラクティスに関する知識と経験 AWS クラウド。DevOps の知識を高めるために、DevOps プラクティスを使用してマイクロサービスと疎結合アーキテクチャを構築する」パターンと AWS Cloud9、 AWS 「規範ガイダンス」ウェブサイトを使用できます。

アーキテクチャ

次の図は、 を使用して、 `MonoRepoStack`と の 2 つの AWS CloudFormation スタックを持つインフラストラクチャ AWS CDK を定義する方法を示していますPipelinesStack。

この図表は、次のワークフローを示しています：

1. ブートストラッププロセスでは、 AWS CDK を使用して AWS CloudFormation スタックMonoRepoStackと を作成しますPipelinesStack。
2. `MonoRepoStack` スタックは、アプリケーション用の CodeCommit リポジトリと、各コミットの後に開始される `monorepo-event-handler` Lambda 関数を作成します。
3. `PipelinesStack` スタックは、Lambda 関数によって開始されるパイプラインを CodePipeline に作成します。各マイクロサービスにはインフラストラクチャパイプラインが定義されている必要があります。
4. `microservice-n` のパイプラインは Lambda 関数によって開始され、CodeCommit のソースコードに基づいて分離された CI/CD ステージを開始します。

5. microservice-1 のパイプラインは Lambda 関数によって開始され、CodeCommit のソースコードに基づいて分離された CI/CD ステージを開始します。

次の図は、 アカウントPipelinesStackでの AWS CloudFormation スタックMonoRepoStackと のデプロイを示しています。

1. ユーザーがアプリケーションのマイクロサービスの 1 つでコードを変更します。
2. ユーザーは、ローカルリポジトリから CodeCommit リポジトリに変更をプッシュします。
3. プッシュアクティビティは、CodeCommit リポジトリへのすべてのプッシュを受け取る Lambda 関数を開始します。
4. Lambda 関数は、 の一機能である Parameter Store でパラメータを読み取り AWS Systems Manager、最新のコミット ID を取得します。パラメータの命名形式は です/MonoRepoTrigger/{repository}/{branch_name}/LastCommit。パラメータが見つからない場合、Lambda 関数は CodeCommit リポジトリから最後のコミット ID を読み取り、返された値を Parameter Store に保存します。
5. コミット ID と変更されたファイルを特定すると、Lambda 関数は各マイクロサービスディレクトリのパイプラインを識別し、必要な CodePipeline パイプラインを開始します。

ツール

- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それを通じてプロビジョニングするためのソフトウェア開発フレームワークです AWS CloudFormation。
- Python は、迅速に作業し、システムをより効果的に統合できるプログラミング言語です。

コード

このパターンのソースコードとテンプレートは、GitHub AWS CodeCommit monorepo マルチパイプライントリガーリポジトリで利用できます。

ベストプラクティス

- このサンプルアーキテクチャには、デプロイされたインフラストラクチャのモニタリングソリューションは含まれていません。このソリューションを本番環境にデプロイする場合は、モニタリングを有効にすることをお勧めします。詳細については、 AWS Serverless Application Model （ AWS

SAM) ドキュメントのCloudWatch Application Insights を使用してサーバーレスアプリケーションをモニタリングする」を参照してください。

- このパターンで提供されるサンプルコードを編集するときは、 AWS CDK ドキュメントのクラウドインフラストラクチャの開発とデプロイに関するベストプラクティスに従ってください。
- マイクロサービスパイプラインを定義するときは、 AWS CodePipeline ドキュメントのセキュリティのベストプラクティスを確認してください。
- cdk-nag ユーティリティを使用して AWS CDK 、コードのベストプラクティスを確認することもできます。このツールは、パックごとにグループ化された一連のルールを使用してコードを評価します。使用可能なパックは次のとおりです。
  - AWS ソリューションライブラリ
  - 医療保険の相互運用性と説明責任に関する法律 (HIPAA) セキュリティ
  - 米国国立標準技術研究所 (NIST) 800-53 rev 4
  - NIST 800-53 rev 5
  - Payment Card Industry Data Security Standard (PCI DSS) 3.2.1

エピック

環境をセットアップする

タスク	説明	必要なスキル
仮想 Python 環境を作成します。	AWS Cloud9 IDE で、仮想 Python 環境を作成し、次のコマンドを実行して必要な依存関係をインストールします。 `make install`	開発者
AWS リージョン の AWS アカウント と をブートストラップします AWS CDK。	次のコマンドを実行して、必要な AWS アカウント とリージョンをブートストラップします。 `make bootstrap account-id=<your-AWS-account-ID>`	開発者

タスク	説明	必要なスキル
	region=<required-region>	

マイクロサービス用の新しいパイプラインの追加

タスク	説明	必要なスキル
サンプルコードをアプリケーションディレクトリに追加します。	サンプルアプリケーションコードを含むディレクトリを、クローンされた GitHub AWS CodeCommit モノレポマルチパイプライントリガーリポジトリのmonorepo-sample ディレクトリに追加します。	開発者
monorepo-main.json ファイルを編集します。	アプリケーションのコードのディレクトリ名とパイプラインの名前を、クローンされたリポジトリ の monorepo-main.json ファイルに追加します。	開発者
パイプラインを作成します。	リポジトリの Pipelines ディレクトリに、classアプリケーションのパイプラインを追加します。ディレクトリには、 pipeline_hotsite.py と の 2 つのサンプルファイルが含まれていますpipeline_demo.py 。各ファイルには、ソース、ビルド、デプロイの 3 つのステージがあります。	開発者

タスク	説明	必要なスキル
	ファイルの 1 つをコピーして、アプリケーションの要件に応じて変更を加えることができます。	

タスク	説明	必要なスキル

monorepo_config.py ファイルを編集します。	service_map に、アプリケーションのディレクトリ名と、パイプライン用に作成したクラスを追加します。
	
	例えば、次のコードは、MySamplePipeline クラスで pipeline_mysample.py と名前が付けられたファイルを使用する Pipelines ディレクトリ内のパイプライン定義を示しています。
	
	`...`
	`# Pipeline definition imports`
	`from pipelines.pipeline_demo import DemoPipeline`
	`from pipelines.pipeline_hotsite import HotsitePipeline`
	`from pipelines.pipeline_mysample import MySamplePipeline`
	
	`### Add your pipeline configuration here`
	`service_map: Dict[str, ServicePipeline] = {`
	`# folder-name -> pipeline-class`
	`'demo': DemoPipeline(),`
	`'hotsite': HotsitePipeline(),`
		開発者

タスク	説明	必要なスキル
	`'mysample': MySamplePipeline() }`	

MonorePostack スタックをデプロイします。

タスク	説明	必要なスキル
AWS CloudFormation スタックをデプロイします。	`make deploy-core` コマンドを実行して、 AWS CloudFormation MonoRepoStack デフォルトのパラメータ値を持つスタックをクローンされたリポジトリのルートディレクトリにデプロイします。 リポジトリの名前は、`make deploy-core monorepo-name=<repo_name>` コマンドを実行して変更できます。 **Note** `make deploy monorepo-name=<repo_name>` コマンドを使用して、両方のパイプラインを同時にデプロイできます。	開発者

タスク	説明	必要なスキル
CodeCommit リポジトリを検証します。	`aws codecommit get-repository --repository-name <repo_name>` コマンドを実行して、リソースが作成されたことを確認します。 **Important** AWS CloudFormation スタックはモノレポが保存されている CodeCommit リポジトリを作成するため、変更のプッシュを開始した場合は`cdk destroy MonoRepoStack` コマンドを実行しないでください。	開発者
AWS CloudFormation スタックの結果を検証します。	次のコマンドを実行して、 AWS CloudFormation `MonoRepoStack` スタックが正しく作成および設定されていることを確認します。 `aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE --query 'StackSummaries[?StackName == 'MonoRepoStack']'`	開発者

PipelinesStack スタックをデプロイします。

タスク	説明	必要なスキル
AWS CloudFormation スタックをデプロイします。	AWS CloudFormation `PipelinesStack` スタックは、スタックをデプロイした後にデプロイする必要がありますMonoRepoStack 。monorepo のコードベースに新しいマイクロサービスが追加されるとスタックのサイズが大きくなり、新しいマイクロサービスが導入されるとスタックが再デプロイされます。 `make deploy-pipelines` コマンドを実行して Pipelines Stack スタックをデプロイします。 **Note** `make deploy monorepo-name=<repo_name>` コマンドを実行して、両方のパイプラインを同時にデプロイすることもできます。 以下の出力例は、実装の最後に `PipelinesStacks` デプロイメントによってマイクロ	開発者

タスク	説明	必要なスキル
	サービスの URL がどのように出力されるかを示しています。 `Outputs: PipelinesStack.dem ourl = .cloudfront.net PipelinesStack.hotsi teurl = .cloudfro nt.net`	
AWS CloudFormation スタックの結果を検証します。	次のコマンドを実行して、 AWS CloudFormation `PipelinesStacks` スタックが正しく作成および設定されていることを確認します。 `aws cloudformation list-stacks --stack-s tatus-filter CREATE_CO MPLETE UPDATE_COMPLETE --query 'StackSum maries[?StackName == 'PipelinesStack']'`	開発者

リソースをクリーンアップする

タスク	説明	必要なスキル
AWS CloudFormation スタックを削除します。	`make destroy` コマンドを実行します。	開発者
パイプラインの S3 バケットを削除します。	1. にサインインAWS Management Consoleし、Amazon Simple Storage Service	開発者

タスク	説明	必要なスキル
	(Amazon S3) コンソールを開きます。 2. パイプラインに関連付けられている S3 バケットを削除し、次の名前を使用します。 `pipelinesstack-codepipeline*`	

トラブルシューティング

問題	ソリューション
AWS CDK 問題が発生しました。	AWS CDK ドキュメントの「一般的なAWS CDK 問題のトラブルシューティング」を参照してください。
マイクロサービスコードをプッシュしましたが、マイクロサービスパイプラインが実行されませんでした。	セットアップの検証 ブランチ設定を確認します。 • コードを正しいブランチにプッシュしていることを確認してください。このパイプラインは、`main`ブランチに変更が行われた場合にのみ実行されるように設定されています。他のブランチにプッシュしても、特に設定されていない限り、パイプラインは開始されません。 • コードをプッシュしたら、コミットが に表示されているかどうかをチェック AWS CodeCommit して、プッシュが成功し、ローカル環境とリポジトリ間の接続が損なわれていないことを確認します。コードのプッシュに問題がある場合は、認証情報を更新します。

問題	ソリューション
	設定ファイルを検証します。 • の service_map 変数がマイクロサービスの現在のディレクトリ構造monorepo_config.py を正確に反映していることを確認します。この変数は、コードプッシュをそれぞれのパイプラインにマッピングする上で重要な役割を果たします。 • monorepo-main.json マイクロサービスの新しいマッピングを含めるように が更新されていることを確認します。このファイルは、パイプラインがマイクロサービスの変更を認識して正しく処理するために不可欠です。 コンソールでのトラブルシューティング AWS CodePipeline は以下をチェックします。 • でAWS Management Console、パイプラインがホスト AWS リージョン されている にいることを確認します。CodePipeline コンソールを開き、マイクロサービスに対応するパイプラインが開始されたかどうかを確認します。 エラー分析: パイプラインが開始されたが失敗した場合は、CodePipeline から提供されたエラーメッセージまたはログを確認して、問題の原因を理解します。 AWS Lambda トラブルシューティング : • AWS Lambda コンソールで、monorepo-event-handler Lambda 関数を開きま

問題	ソリューション
	す。コードプッシュに応答して関数が開始されたことを確認します。 ログ分析: Lambda 関数のログに問題がないか調べます。ログは、関数が実行されたときに何が起こったかに関する詳細なインサイトを提供し、関数がイベントを期待どおりに処理したかどうかを特定するのに役立ちます。

問題	ソリューション
すべてのマイクロサービスを再デプロイする必要があります。	すべてのマイクロサービスの再デプロイを強制するには、2 つのアプローチがあります。要件に合ったオプションを選択します。 アプローチ 1: Parameter Store でパラメータを削除する このメソッドでは、デプロイに使用された最後のコミット ID を追跡する Systems Manager パラメータストア内の特定のパラメータを削除します。このパラメータを削除すると、システムはすべてのマイクロサービスを新しい状態として認識するため、次のトリガーで再デプロイするように強制されます。 ステップ: 1. モノレポのコミット ID または関連するデプロイマーカーを保持する特定の Parameter Store エントリを見つけます。パラメータ名は次の形式に従います。 `"/MonoRepoTrigger/{repository}/{branch_name}/LastCommit"` 2. パラメータ値が重要である場合、またはリセットする前にデプロイ状態のレコードを維持する場合は、パラメータ値をバックアップすることを検討してください。 3. AWS Management Console、 AWS CLI、または SDKs を使用して、識別されたパラメータを削除します。このアクションにより、デプロイマーカーがリセットされます。 4. 削除後、リポジトリへの次のプッシュにより、デプロイを検討する最新のコミットが検索されるため、システムはすべてのマイクロサービスをデプロイします。

問題	ソリューション
	メリット： • 最小限のステップで簡単かつ迅速に実装できます。 • デプロイを開始するために任意のコード変更を行う必要はありません。 デメリット： • デプロイプロセスをより細かく制御できます。 • Parameter Store が他の重要な設定の管理に使用されると、潜在的にリスクがあります。 アプローチ 2: 各モノレポサブフォルダにコミットをプッシュする この方法では、マイナーな変更を行い、モノレポ内の各マイクロサービスサブフォルダにプッシュして、個々のパイプラインを開始します。 ステップ: 1. 再デプロイが必要なモノレポ内のすべてのマイクロサービスを一覧表示します。 2. マイクロサービスごとに、そのサブフォルダで影響のない最小限の変更を行います。これは、README ファイルの更新、設定ファイルへのコメントの追加、またはサービスの機能に影響を与えない変更である可能性があります。 3. これらの変更を明確なメッセージ (「マイクロサービスの再デプロイの開始」など) でコミットし、リポジトリにプッシュします。デ

問題	ソリューション
	プロイを開始するブランチに変更を必ずプッシュしてください。 4. 各マイクロサービスのパイプラインをモニタリングして、パイプラインが正常に開始および完了していることを確認します。 メリット： • 再デプロイされるマイクロサービスをきめ細かく制御します。 • 他の目的で使用される可能性のある設定パラメータの削除は伴わないため、より安全です。 デメリット： • 特に多数のマイクロサービスでは、より時間がかかります。 • コミット履歴を整理する可能性のある不要なコード変更を行う必要があります。

関連リソース

- CDK Pipelines を使用した継続的な統合と配信 (CI/CD) (AWS CDK ドキュメント)
- aws-cdk/pipelines モジュール (AWS CDK API リファレンス)

AWS CloudFormation を使用して Bitbucket リポジトリを AWS Amplify と統合する

作成者：アルウィン・エイブラハム (AWS)

概要

AWS Amplify を使用すると、通常必要なインフラストラクチャをセットアップしなくても、静的ウェブサイトを迅速にデプロイしてテストできます。既存のアプリケーションコードを移行する場合でも、新しいアプリケーションを構築する場合でも、組織が Bitbucket をソース管理に使用したい場合は、このパターンのアプローチを導入できます。AWS CloudFormation を使用して Amplify を自動的にセットアップすることで、使用している設定を可視化できます。

AWS CloudFormation を使用して Bitbucket リポジトリを AWS Amplify と統合することで、フロントエンド CI/CD パイプラインとデプロイ環境を作成します。このパターンのアプローチは、反復可能なデプロイメントのためのAmplify フロントエンドパイプラインを構築できることを意味します。

前提条件と制限

前提条件

- Amazon Web Services (AWS) (AWS) アカウント。
- 管理者アクセス権を持つアクティブな Bitbucket アカウント
- 「cURL」または「Postman」アプリケーションを使用する端末へのアクセス
- Amplify Gurify の使用方法
- AWS CloudFormation に精通している
- YAML 形式のファイルに精通していること

アーキテクチャ

テクノロジースタック

- Amplify
- AWS CloudFormation
- Bitbucket

ツール

- AWS Amplify — Amplify は、開発者がクラウドを利用したモバイルアプリやウェブアプリを開発およびデプロイするのに役立ちます。
- AWS CloudFormation – AWS CloudFormation は AWS リソースのモデル化とセットアップに役立つサービスです。リソース管理に割く時間を減らし、AWS で実行するアプリケーションにより注力できるようになります。
- Bitbucket — Bitbucket は、プロフェッショナルチーム向けに設計された Git リポジトリ管理ソリューションです。Git リポジトリの管理、ソースコードの共同編集、開発フローのガイドなどを一元的に行えます。

コード

`bitbucket-amplify.yml`ファイル (添付) には、このパターンの AWS CloudFormation テンプレートが含まれています。

エピック

Bitbucket リポジトリを設定します。

タスク	説明	必要なスキル
(オプション) Bitbucket リポジトリを作成します。	1. Bitbucket アカウントにサインインし、新しいリポジトリを作成します。詳細については、Bitbucket ドキュメントの「Git リポジトリの作成」を参照してください。 2. ワークスペースの名前を記録します。	DevOps エンジニア

タスク	説明	必要なスキル
	Note 既存の Bitbucket リポジトリを使用することもできます。	
ワークスペース設定を開きます。	1. ワークスペースを開き、「リポジトリ」タブを選択します。 2. Amplify と使用する DB リポジトリを選択します。 3. リポジトリ名の上にあるワークスペースの名前を選択します。 4. サイドバーで「設定」を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
OAuth コンシューマーを作成します。	1. 「アプリと機能」セクションで「OAuth コンシューマー」を選択し、「コンシューマーを追加」を選択します。 2. ストリームの名前 (例 : `Amplify Integration`) を入力します。 3. [コールバック URL] を入力します。このフィールドは必須入力ですが、統合を完了するために使用されるわけではないので、値は`http://localhost:3000` でもかまわない 4. 「これは個人消費者です」のボックスにチェックを入れます。 5. アクセス許可を使用する場合、以下を選択します。 • プロジェクト – `Read` • リポジトリ – `Admin` • プルリクエスト – `Read` • ウェブフック —`Read`と`Write` 6. その他のフィールドは、デフォルト設定のままにしておき、「提出」を選択します。 7. 生成されたキーとシークレットを記録します。	DevOps エンジニア

タスク	説明	必要なスキル
OAuth アクセストークンを取得します。	1. ターミナルウィンドウを開いて、次のコマンドを実行します。 `curl -X POST -u "KEY:SECRET" https://bitbucket.org/site/oauth2/access_token -d grant_type=client_credentials` **Important** KEY と を、前に記録したキーとシークレットSECRETに置き換えます。 2. 引用符を使わずにアクセストークンを記録します。トークンの有効期間は限られており、デフォルトは 2 時間です。この期間内に AWS CloudFormation テンプレートを実行する必要があります。	DevOps エンジニア

AWS CloudFormation スタックの設定とデプロイ

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをダウンロードします。	`bitbucket-amplify.yml` AWS CloudFormation のテンプレートをダウンロード	

タスク	説明	必要なスキル
	します。このテンプレートは、Amplify プロジェクトとブランチに加えて、AAmplify でCI/CDパイプラインを作成します。	

タスク	説明	必要なスキル
AWS CloudFormation スタックの設定とデプロイ	1. デプロイする AWS リージョンの AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開きます。 2. [スタックの作成 (新しいリソースを使用)] を選択し、[テンプレートファイルを更新]を選択します。 3. `bitbucket-amplify.yml` ファイルをアップロードします。 4. 「次へ」を選択し、スタック名を入力して、次のパラメータを入力します。 • アクセストークン：前に作成した OAuth アクセストークンを貼り付けます。 • リポジトリ URL：Bitbucket プロジェクトリポジトリの URL を追加します。URL の形式は`https://bitbucket.org/<WORKSPACE_NAME>/<REPO_NAME>` です。 • ブランチ名：Bitbucket リポジトリ内のブランチの名前と一致する必要があります。このブランチは AWS CloudFormation スタックの実行時には存	DevOps エンジニア

タスク	説明	必要なスキル
	在する必要はありませんが、コードを環境にデプロイするためには必要です。 • プロジェクト名：これは Amplify プロジェクトに関連付ける名前です。 5. [次へ] を選択してから、[スタックの作成] を選択します。	

CI/CD パイプラインをテストする

タスク	説明	必要なスキル
コードをリポジトリのブランチにデプロイします。	1. `git clone https://bitbucket.org/<WORKSPACE_NAME>/<REPO_NAME>` コマンドを実行して Bitbucket リポジトリのクローンを作成します。 2. AWS CloudFormation スクリプトを実行したときに使用されたブランチ名を確認してください。新しいブランチを作成してチェックアウトするには、`git checkout -b <BRANCH_NAME>` コマンドを実行します。既存のブランチをチェックアウトするには、`git checkout`	アプリ開発者

タスク	説明	必要なスキル
	<BRANCH_NAME> コマンドを実行します。 3. コードをブランチにコミットし、git commitおよびgit pushコマンドを実行してリモートブランチにプッシュします。 4. 次に、Amplify はアプリケーションをビルドしてデプロイします。 このコマンドの詳細については、Bitbucket ドキュメントの「基本の Git コマンド」を参照してください。	

関連リソース

認証方法 (アトラシアン製品ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Step Functions と Lambda プロキシ関数を使用して AWS アカウント間で CodeBuild プロジェクトを起動する

作成者: Richard Milner-Watts (AWS)、Amit Anjarlekar (AWS)

概要

このパターンは、AWS Step Functions と AWS Lambda プロキシ関数を使用して、複数の AWS アカウントで AWS CodeBuild プロジェクトを非同期的に起動する方法を示しています。 AWS Step Functions AWS Lambda パターンのサンプル Step Functions ステートマシンを使用して、CodeBuild プロジェクトの成功をテストできます。

CodeBuild は、フルマネージドランタイム環境から AWS コマンドラインインターフェイス (AWS CLI) を使用して運用タスクを起動するのに役立ちます。環境変数をオーバーライドすることで、ランタイムにおける CodeBuild プロジェクトの動作を変更できます。また、CodeBuild を使用してワークフローを管理できます。詳細については、AWS ワークショップウェブサイトの「サービスカタログツール」と、AWS データベースブログの「AWS CodeBuild と Amazon EventBridge を使用して Amazon RDS for PostgreSQL のジョブをスケジュールする」を参照してください。

前提条件と制限

前提条件

- 2 つのアクティブな AWS アカウント。1 つはStep Functions で Lambda プロキシ関数を呼び出すためのソースアカウントで、もう 1 つはリモート CodeBuild サンプルプロジェクトを構築するためのターゲットアカウント

制約事項

- このパターンを使用して、アーティファクトをアカウント間でコピーすることはできません。

アーキテクチャ

このパターンが構築するアーキテクチャを次の図に示します。

この図表は、次のワークフローを示しています：

1. Step Functions ステートマシンは、指定された入力マップを解析し、定義したアカウント、リージョン、プロジェクトごとに Lambda プロキシ関数 (`codebuild-proxy-lambda`) を呼び出します。
2. Lambda プロキシ関数は、AWS Security Token Service (AWS STS) を使用して、ターゲットアカウントの IAM ポリシー (`codebuild-proxy-role`) に関連付けられている IAM プロキシロール (`codebuild-proxy-policy`) を引き受けます。
3. 引き受けたロールを使用して、Lambda 関数は CodeBuild プロジェクトを起動し、CodeBuild ジョブ ID を返します。Step Functions ステートマシンは、成功または失敗のステータスを受け取るまでループし、CodeBuild ジョブをポーリングします。

ステートマシンのロジックを次の図に示します。

テクノロジースタック

- AWS CloudFormation
- CodeBuild
- IAM
- Lambda
- Step Functions
- X-Ray

ツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- AWS CloudFormation Designer には、CloudFormation テンプレートの表示と編集に役立つ JSON と YAML の統合エディタが用意されています。
- AWS CodeBuild はフルマネージドの構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Step Functionsは、AWS Lambda関数と他のAWS サービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。
- AWS X-Ray は、アプリケーションで処理するリクエストに関するデータを収集するのに役立ち、さらにデータの表示、フィルタリング、インサイトによって問題や機会を特定して最適化するために使用できるツールを提供します。

コード

このパターンのサンプルコードは、GitHubCross Account CodeBuild Proxy リポジトリで入手できます。このパターンでは、AWS Lambda Powertools for Python ライブラリを使用してロギング機能とトレース機能を提供しています。このライブラリとそのユーティリティの詳細については、「Powertools for AWS Lambda (Python)」を参照してください。

ベストプラクティス

1. Step Function ステートマシンの待機時間値を調整して、ジョブステータスのポーリングリクエストを最小限に抑えます。CodeBuild プロジェクトに予想される実行時間を使用します。
2. Step Functions でマップの `MaxConcurrency`プロパティを調整して、並行して実行できる CodeBuild プロジェクトの数を制御します。
3. 必要に応じて、本番稼働の準備状況のサンプルコードを確認します。ソリューションによってログに記録される可能性のあるデータと、デフォルトの Amazon CloudWatch 暗号化で十分かどうかを検討します。

エピック

ソースアカウントで Lambda プロキシ関数と関連する IAM ロールを作成する

タスク	説明	必要なスキル
AWS アカウント ID を記録する。	アカウント間のアクセスを設定するには、AWS アカウント ID が必要です。	AWS DevOps

タスク	説明	必要なスキル
	ソースアカウントとターゲットアカウントの AWS アカウント ID を記録します。詳細については、IAM ドキュメントの「アカウント ID の検索」を参照してください。	

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをダウンロードする。	1. このパターンの `sample_target_codebuild_template.yaml` AWS CloudFormation テンプレートを GitHub リポジトリからダウンロードします。 2. このパターンの `codebuild_lambda_proxy_template.yaml` AWS CloudFormation テンプレートを GitHub リポジトリからダウンロードします。 **Note** AWS CloudFormation テンプレートでは、 `<SourceAccountId>` はソースアカウントの AWS アカウント ID で、 `<TargetAccountId>` はターゲットアカウントの AWS アカウント ID です。	AWS DevOps

タスク	説明	必要なスキル
AWS CloudFormation スタックを設定しデプロイします。	1. ソースアカウントの AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開いて、[スタック] を選択します。 2. [スタックの作成] を選択し、[With new resources (standard) 新しいリソースを使用 (標準)] を選択します。 3. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。 4. [テンプレートファイルのアップロード] で [ファイル] を選択し、ダウンロードしたファイル`codebuild_lambda_proxy_template.yaml` を選択します。[Next (次へ)] を選択します。 5. [スタック名] にスタックの名前 (例: `codebuild-lambda-proxy`) を入力します。 6. **Note** `crossAccountTargetRoleArn` パラメータをお使い	AWS DevOps

タスク	説明	必要なスキル
	の <TargetAccountId> (たとえば <arn:aws:iam::123456789012:role/proxy-lambda-codebuild-role>) に置き換えます。: パラメータのデフォルト値targetCodeBuildProject を更新する必要はありません。 7. [次へ] を選択し、デフォルトのスタック作成オプションを許可してから、[次へ] を選択します。 8. [AWS CloudFormation によって、カスタム名の IAM リソースが作成される場合があることを承認します] チェックボックスを選択した後、[スタックの作成] を選択します。 **Note** ターゲットアカウントにリソースを作成する前に、プロキシ Lambda 関数の AWS	

タスク	説明	必要なスキル
	CloudFormation スタックを作成する必要があります。ターゲットアカウントで信頼ポリシーを作成すると、IAM ロールはロール名から内部識別子に変換されます。これが、IAM ロールがすでに存在している必要がある理由です。	
プロキシ関数とステートマシンが作成されていることを確認します。	1. AWS CloudFormation スタックが CREATE_COMPLETE ステータスになるまで待ちます。この所要時間は 1 分以内となります。 2. AWS Lambda コンソールを開き、関数を選択し、次に`lambda-proxy-Proxy Lambda-<GUID>` 関数を選択します。 3. AWS Step Functions コンソールを開き、ステートマシンを選択し、`sample-crossaccount-codebuild-state-machine` ステートマシンを検索します。	AWS DevOps

ターゲットアカウントに IAM ロールを作成し、サンプル CodeBuild プロジェクトを起動する

タスク	説明	必要なスキル
AWS CloudFormation スタックを設定しデプロイします。	1. ターゲットアカウントの AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開いて、[スタック] を選択します。 2. [スタックの作成] を選択し、[With new resources (standard) 新しいリソースを使用 (標準)] を選択します。 3. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。 4. [テンプレートファイルのアップロード] で [ファイルの選択] を選択し、**sample_target_codebuild_template.yaml** ファイルを選択します。[Next (次へ)] を選択します。 5. [スタック名] にスタックの名前 (例: `sample-codebuild-stack`) を入力します。 6. `crossAccountSourceRoleArn` パラメータをお使いの <SourceAccountId> (たとえば <arn:aws:iam::1234	AWS DevOps

タスク	説明	必要なスキル
	56789012:role/codebuild-proxy-lambda-role>) に置き換えます。 7. [次へ] を選択し、デフォルトのスタック作成オプションを許可してから、[次へ] を選択します。 8. [AWS CloudFormation によって、カスタム名の IAM リソースが作成される場合があることを承認します] チェックボックスを選択した後、[スタックの作成] を選択します。	
サンプル CodeBuild プロジェクトが作成されていることを確認します。	1. AWS CloudFormation スタックが CREATE_COMPLETE ステータスになるまで待ちます。この所要時間は 1 分以内となります。 2. AWS CodeBuild コンソールを開き、sample-codebuild-project プロジェクトを検索します。	AWS DevOps

クロスアカウントの Lambda プロキシ関数のテスト

タスク	説明	必要なスキル
ステートマシンを起動します。	1. ソースアカウントの AWS マネジメントコンソールにサインインし、AWS Step Functions コンソールを開	AWS DevOps

タスク	説明	必要なスキル
	いて、[ステートマシン] を選択します。 2. sample-crossaccount-codebuild-state-machine ステートマシンを選択し、次に[実行開始]を選択します。 3. 入力エディタで、次の JSON を入力し、 を CodeBuild プロジェクトを含むアカウントの AWS アカウント ID <TargetAccountID> に置き換えます。 `{ "crossAccountTargetRoleArns": [ { "arn": "arn:aws:iam::<TargetAccountID>:role/proxy-lambda-codebuild-role", "region": "eu-west-1", "codeBuildProject": "sample-codebuild-project", "SampleValue1": "Value1", "SampleValue2": "Value2" } ] }`	

タスク	説明	必要なスキル
	Note キーと値のペアは、ソースアカウントの 関数からターゲットアカウントの CodeBuild プロジェクトに環境変数として渡されます。 4. [実行のスタート] を選択します。 5. ステートマシンページの [詳細] タブで、[実行ステータス] が [成功] に設定されているかどうかを確認します。これで、ステートマシンの実行が確認できました。注: ステートマシンが [成功] ステータスになるまで約 30 秒かかることがあります。 6. ステートマシンのステップの出力と入力を確認するには、[実行イベント履歴] セクションでそのステップを展開します。たとえば、「Lambda - CodeBuild プロキシ – 開始」ステップを展開します。出力には、オーバーライドされた環境変数、元のペイロード、およ	

タスク	説明	必要なスキル
	び CodeBuild ジョブ ID の詳細が含まれます。	
環境変数を検証します。	1. ターゲットアカウントの AWS マネジメントコンソールにサインインします。 2. AWS CodeBuild コンソールを開き、[ビルド] を展開して [プロジェクトのビルド] を選択します。 3. `sample-codebuild-project` プロジェクトを選択し、詳細の表示を選択します。 4. ビルド履歴タブで、プロジェクトの最新のビルドを選択し、ログの表示を選択します。 5. ログ出力で、STDOUT に出力される環境変数が Step Functions サンプルステートマシンの環境変数と一致することを確認します。	AWS DevOps

トラブルシューティング

問題	ソリューション
Step Functions の実行に予想よりも時間がかかります。	Step Function ステートマシンでマップの `MaxConcurrency` プロパティを調整して、並列で実行できる CodeBuild プロジェクトの数を制御します。

問題	ソリューション
CodeBuild ジョブの実行に予想以上に時間がかかります。	1. Step Functions ステートマシンの待機時間値を調整して、ジョブステータスのポーリングリクエストを最小限に抑えます。CodeBuild プロジェクトに予想される実行時間を使用します。 2. CodeBuild が使用する適切なツールであるかどうかを検討します。例えば、CodeBuild ジョブの初期化に必要な時間は、AWS Lambda よりも大幅に長くなる可能性があります。高スループットと高速な完了時間が必要な場合は、ビジネスロジックを AWS Lambda に移行し、ファンアウトアーキテクチャを使用することを検討してください。

Application Recovery Controller を使用して EMR クラスターのマルチ AZ フェイルオーバーを管理する

作成者: Aarti Rajput (AWS)、Ashish Bhatt (AWS)、Neeti Mishra (AWS)、Nidhi Sharma (AWS)

概要

このパターンは、Amazon EMR ワークロードの効率的なディザスタリカバリ戦略を提供し、単一の 内の複数のアベイラビリティーゾーン間で高可用性とデータ整合性を確保します AWS リージョン。この設計では、Amazon Application Recovery Controller と Application Load Balancer を使用して、Apache Spark ベースの EMR クラスターのフェイルオーバーオペレーションとトラフィック分散を管理します。

標準条件下では、プライマリアベイラビリティーゾーンは、完全な読み取り/書き込み機能を備えたアクティブな EMR クラスターとアプリケーションをホストします。アベイラビリティーゾーンが予期せず失敗した場合、トラフィックは自動的にセカンダリアベイラビリティーゾーンにリダイレクトされ、そこで新しい EMR クラスターが起動されます。両方のアベイラビリティーゾーンは、専用のゲートウェイエンドポイントを介して共有 Amazon Simple Storage Service (Amazon S3) バケットにアクセスし、一貫したデータ管理を実現します。このアプローチにより、ダウンタイムを最小限に抑え、アベイラビリティーゾーンの障害発生時に重要なビッグデータワークロードを迅速に復旧できます。このソリューションは、リアルタイム分析が重要な金融や小売などの業界で役立ちます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon Elastic Compute Cloud での Amazon EMR (Amazon EC2)
- EMR クラスターのマスターノードから Amazon S3 へのアクセス。
- AWS マルチ AZ インフラストラクチャ

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

製品バージョン

- Amazon EMR 6.x 以降のリリース

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EMR クラスター
- Amazon Application Recovery Controller
- Application Load Balancer
- Amazon S3 バケット
- Amazon S3 のゲートウェイエンドポイント

ターゲット アーキテクチャ

このアーキテクチャは、複数のアベイラビリティーゾーンを使用し、Application Recovery Controller を通じて自動復旧メカニズムを実装することで、アプリケーションの耐障害性を提供します。

1. Application Load Balancer は、トラフィックをアクティブな Amazon EMR 環境にルーティングします。これは通常、プライマリアベイラビリティーゾーンのプライマリ EMR クラスターです。
2. アクティブな EMR クラスターはアプリケーションリクエストを処理し、読み取りおよび書き込みオペレーションのために専用の Amazon S3 ゲートウェイエンドポイントを介して Amazon S3 に接続します。
3. Amazon S3 は中央データリポジトリとして機能し、チェックポイントまたは EMR クラスター間の共有ストレージとして使用される可能性があります。

   EMR クラスターは、`s3://`プロトコルと EMR ファイルシステム (EMRFS) を介して Amazon S3 に直接書き込むときにデータの一貫性を維持します。データの整合性を確保するために、このパターンのソリューションは Amazon S3 への先行書き込みログ記録 (WAL) を実装し、Amazon S3 バージョニング機能を使用してデータバージョンを追跡し、必要に応じてロールバックを有効にします。読み取りオペレーションの場合、クラスターは Amazon S3 Select を使用して共有 Amazon S3 ストレージレイヤーにアクセスし、パフォーマンスを最適化します。さらに、Spark キャッシュメカニズムによって補完され、Amazon S3 アクセスの繰り返しを最小限に抑えます。Amazon S3 は、複数のアベイラビリティーゾーンにわたって 99.999999999「」% の耐久性

を実現するように設計されており、ネイティブの Amazon EMR 統合を提供し、信頼性の高いクラスター間データ整合性ソリューションを提供します。

4. Application Recovery Controller は、プライマリアベイラビリティーゾーンの状態を継続的にモニタリングし、必要に応じてフェイルオーバーオペレーションを自動的に管理します。
5. Application Recovery Controller がプライマリ EMR クラスターで障害を検出すると、次のアクションを実行します。
   - アベイラビリティーゾーン 2 のセカンダリ EMR クラスターへのフェイルオーバープロセスを開始します。
   - ルーティング設定を更新して、トラフィックをセカンダリクラスターに転送します。

ツール

AWS サービス

- Amazon Application Recovery Controller は、 とアベイラビリティーゾーン間のアプリケーションの復旧を管理 AWS リージョン および調整するのに役立ちます。このサービスは、従来のツールやプロセスに必要な手動ステップを減らすことで、プロセスを簡素化し、アプリケーション復旧の信頼性を向上させます。
- Application Load Balancer は、Open Systems Interconnection (OSI) モデルの 7 番目のレイヤーであるアプリケーションレイヤーで動作します。受信アプリケーショントラフィックを EC2 インスタンスなどの複数のターゲットに分散し、複数のアベイラビリティーゾーンに分散します。これにより、アプリケーションの可用性が向上します。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- Amazon EMR は、Apache Spark、Apache Hive、Presto などのオープンソースフレームワークのデータ処理、インタラクティブ分析、機械学習を提供するビッグデータプラットフォームです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon S3 は、いつでもどこからでも、任意の量のデータを保存および取得するために使用できるシンプルなウェブサービスインターフェイスを提供します。このサービスを使用すると、クラウドネイティブストレージを利用するアプリケーションを簡単に構築できます。
- Amazon S3 のゲートウェイエンドポイントは、 AWS ネットワーク経由で仮想プライベートクラウド (VPC) から Amazon S3 にアクセスするためにルートテーブルで指定するゲートウェイです。

ベストプラクティス

- AWS セキュリティ、アイデンティティ、コンプライアンスのベストプラクティスに従って、堅牢で安全なアーキテクチャを確保します。
- アーキテクチャを AWS Well-Architected フレームワークに合わせます。
- Amazon S3 Access Grants を使用して、Spark ベースの EMR クラスターから Amazon S3 へのアクセスを管理します。詳細については、ブログ記事「Use Amazon EMR with S3 Access Grants to Scale Spark access to Amazon S3」を参照してください。
- Amazon S3 で Spark のパフォーマンスを向上させます。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
AWS Management Consoleにサインインします。	AWS Management Console に IAM ユーザーとしてサインインします。手順については、AWS ドキュメントを参照してください。	AWS DevOps
を設定します AWS CLI。	をインストールする AWS CLI か、最新バージョンに更新して、 AWS のサービス で を操作できるようにします AWS Management Console。手順については、 AWS CLI ドキュメントを参照してください。	AWS DevOps

EMR クラスターに Spark アプリケーションをデプロイする

タスク	説明	必要なスキル
S3 バケットを作成する。	1. S3 バケットを作成して、入力データセット、ログ、ア	AWS DevOps

タスク	説明	必要なスキル
	プリケーション、および出力データを保存します。手順については、Amazon S3 のドキュメントを参照してください。 2. バケットを入力データ (`dataset`) 、ログ ()、Spark アプリケーション (`logs`) 、および出力データ (`spark-app`) の別々のフォルダに整理しますoutput。	

タスク	説明	必要なスキル

EMR クラスターを作成します。	1. 次の AWS CLI コマンドを使用して、高可用性のために 2 つのアベイラビリティーゾーン (us-east-1a や など) にまたがるインスタンスを持つ EMR クラスター (バージョン 6.12 以降などus-east-1b) を作成します。コマンドは、例としてm4.largeインスタンスタイプを指定します。
	
	`aws emr create-cluster \ --ec2-attributes AvailabilityZone=<AZ-name-1> \ --release-label emr-6.12.0 \ --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large`
	
	`aws emr create-cluster \ --ec2-attributes AvailabilityZone=<AZ-name-2> \ --release-label emr-6.12.0 \ --instance-groups InstanceGroupType=MASTER,InstanceCou`
		AWS DevOps

タスク	説明	必要なスキル
	nt=1,InstanceType=m4.large InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large 詳細については、create-cluster コマンドと Amazon EMR ドキュメントを参照してください。 2. 必要に応じて、キーペア、サービスロール、インスタンスプロファイルに必要なアクセス許可を付与します。	

タスク	説明	必要なスキル
EMR クラスターのセキュリティ設定を行います。	1. AWS CLI describe-cluster コマンドを使用して、EMR クラスターのマスターノードに関連付けられているセキュリティグループを特定します。 `aws emr describe-cluster --cluster-id j-XXXXXXXX` 2. セキュリティを強化するには、セキュリティグループ設定を変更して、Secure Shell (SSH) のマスターノードへのアクセス (TCP ポート 22) を許可しますが、特定の IP アドレスに制限します。 詳細については、Amazon EMR ドキュメントを参照してください。	AWS DevOps

タスク	説明	必要なスキル

EMR クラスターに接続します。	提供されたキーペアを使用して、SSH 経由で EMR クラスターのマスターノードに接続します。
	
	キーペアファイルがアプリケーションと同じディレクトリに存在することを確認します。
	
	次のコマンドを実行して、キーペアに正しいアクセス許可を設定し、SSH 接続を確立します。
	
	`chmod 400 <key-pair-name>`
	`ssh -i ./<key-pair-name> hadoop@<master-node-public-dns>`
		AWS DevOps

タスク	説明	必要なスキル

Spark アプリケーションをデプロイします。	SSH 接続を確立すると、Hadoop コンソールに表示されます。
	
	1. vim などのテキストエディタを使用して、Spark アプリケーションファイル (`main.py`) を作成または編集します。
	
	`vim main.py`
	
	Spark アプリケーションの作成と変更の詳細については、Amazon EMR ドキュメントを参照してください。
	2. Spark アプリケーションを EMR クラスターに送信し、S3 バケットの入力データと出力データの場所を指定します。
	
	`spark-submit main.py –data_source <input-data-folder-in-s3> –output_uri <output-folder-in-s3>`
	
	例えば (前に設定したフォルダに基づく) 、
	
	`spark-submit main.py –data_source dataset –output_uri output`
		AWS DevOps

タスク	説明	必要なスキル
	3. アプリケーションログを確認して、アプリケーションの進行状況をモニタリングします。 `yarn logs -applicationId <application-id>`	
Spark アプリケーションをモニタリングします。	1. 別のターミナルウィンドウを開き、EMR クラスターのリソースマネージャーウェブ UI への SSH トンネルを確立します。 `ssh -i <key-pair-name> -N -L 8157:<resource-manager-public-dns>:8088 hadoop@<resource-manager-public-dns>` 2. アプリケーションをモニタリングするには、ウェブブラウザで に移動して、リソースマネージャーのウェブ UI `http://localhost:8157` にアクセスします。	AWS DevOps

トラフィックを別のアベイラビリティーゾーンに移動する

タスク	説明	必要なスキル
Application Load Balancer を作成します。	内の 2 つのアベイラビリティーゾーンにデプロイされて	AWS DevOps

タスク	説明	必要なスキル
	いる Amazon EMR マスターノード間でトラフィックをルーティングするターゲットグループを設定します AWS リージョン。 手順については、Elastic Load Balancing ドキュメントの「Application Load Balancer のターゲットグループを作成する」を参照してください。	

タスク	説明	必要なスキル
Application Recovery Controller でゾーンシフトを設定します。	このステップでは、Application Recovery Controller のゾーンシフト機能を使用して、トラフィックを別のアベイラビリティーゾーンにシフトします。 1. Application Recovery Controller コンソールを開きます。 2. 「開始方法」で、「ゾーンシフトの開始」を選択します。 3. トラフィックを遠ざけるアベイラビリティーゾーンを選択します。 4. リソーステーブルから、ゾーンシフトでサポートされているリソース (Application Load Balancer など) を選択します。 5. [ゾーンシフトの有効期限を設定] で、ゾーンシフトの有効期限を選択または入力します。期間は 1 分から 3 日 (72 時間) の間で設定できます。 すべてのゾーンシフトは一時的なものです。有効期限は必ず設定しますが、アクティブなシフトは、後から新しい有効期限 (最大 3 日後) に更新できます。	AWS DevOps

タスク	説明	必要なスキル
	6. このゾーンシフトに関するコメントを入力します。 7. このチェックボックスをオンにすると、ゾーンシフトを開始した際、トラフィックがアベイラビリティーゾーンからシフトし、アプリケーションの容量が減ることを了承します。 8. [開始] を選択します。 を使用するには AWS CLI、Application Recovery Controller ドキュメントの「ゾーンシフト AWS CLI で を使用する例」を参照してください。	

タスク	説明	必要なスキル

ゾーンシフトの設定と進行状況を確認します。	1. ゾーンシフトに登録されているリソースを確認します。
	`aws arc-zonal-shift list-managed-resources --region <AWS-region-name>`
	たとえば、次の出力は、リソースが両方のアベイラビリティーゾーンで稼働していることを確認します。
	`"appliedWeights": { "use1-az1": 1.0, "use1-az2": 1.0 },`
	2. ゾーンシフトを視覚化するには、次の AWS CLI コマンドを使用してゾーンシフトを開始します。
	`aws arc-zonal-shift start-zonal-shift \ --resource-identifier <application-load-balancer-arn> \ --away-from <source-AZ> \ --expires-in 10m --comment "testing" \ --region <AWS-region-name>`
		AWS DevOps

タスク	説明	必要なスキル

	ここで、 <source-AZ> はトラフィックを遠ざけるアベイラビリティーゾーンの識別子であり、 <application-load-balancer-arn> は Application Load Balancer の Amazon リソースネーム (ARN) です。
	3. トラフィックが別のアベイラビリティーゾーンに移行したことを確認します。
	`aws arc-zonal-shift get-managed-resource \ --resource-identifier <application-load-balancer-arn> \ --region <AWS-region-name>`
	ゾーンシフトは、次の重みで確認できます。
	`"appliedWeights": { "use1-az1": 0.0, "use1-az2": 1.0 },`	

関連リソース

- AWS CLI コマンド：

  - クラスター作成
  - describe-cluster
  - arc-zonal-shift
- スポットインスタンスの Amazon EMR クラスターインスタンスタイプとベストプラクティスの設定 (Amazon EMR ドキュメント)
- IAM でのセキュリティのベストプラクティス (IAM ドキュメント)
- インスタンスプロファイルを使用する (IAM ドキュメント)
- ゾーンシフトとゾーンオートシフトを使用して ARC でアプリケーションを復旧する (Application Recovery Controller ドキュメント)

AWS コードサービスと AWS KMS マルチリージョンキーを使用して、複数のアカウントとリージョンへのマイクロサービスのブルー/グリーンデプロイを管理

作成者：Balaji Vedagiri (AWS)、Ashish Kumar (AWS)、Faisal Shahdad (AWS)、Anand Krishna Varanasi (AWS)、Vanitha Dontireddy (AWS) と Vivek Thangamuthu (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、ブルー/グリーンデプロイ戦略に従い、中央のAWS アカウントから複数のワークロードアカウントとリージョンにグローバルマイクロサービスアプリケーションをデプロイする方法を説明しています。このパターンは以下をサポートします。

- ソフトウェアは中央アカウントで開発されますが、ワークロードとアプリケーションは複数のアカウントと AWS リージョンに分散されます。
- ディザスタリカバリとして、単一の AWS キー管理システム (AWS KMS) マルチリージョンキーが暗号化と復号に使用されます。
- KMS キーはリージョン固有であり、パイプラインアーティファクト用に 3 つの異なるリージョンで管理しまたは作成する必要があります。KMS マルチリージョンキーは、リージョン間で同じキー ID を保持することに役立ちます。
- Git ワークフローの分岐モデルは 2 つのブランチ (開発とメイン) で実装され、コードはプルリクエスト (PR) でマージされます。このスタックからデプロイされる AWS Lambda 関数は、開発ブランチからメインブランチへの PR を作成します。メインブランチにマージするPR は、AWS CodePipeline パイプラインを開始します。これにより、継続的インテグレーションと継続的デリバリー (CI/CD) フローをオーケストレーションし、アカウントにスタックをデプロイします。

このパターンは、AWS CloudFormation スタックによるコードとしての Infrastructure as Code (IaC) 設定のサンプルを提供され、このユースケースを説明します。マイクロサービスのブルー/グリーンデプロイは、AWS CodeDeploy を使用して実装されます。

前提条件と制限

前提条件

- 4 つのアクティブな AWS アカウント:

  - コードパイプラインを管理し、AWS CodeCommit リポジトリを維持するためのツールアカウント。
  - マイクロサービスワークロードをデプロイするための 3 つのワークロード (テスト) アカウント。
- このパターンでは次のリージョンを使用します。他のリージョンを使用する場合は、AWS CodeDeploy と AWS KMS マルチリージョンスタックに適切な変更を加える必要があります。
  - ツール (AWS CodeCommit) アカウント： `ap-south-1`
  - ワークロード (テスト) アカウント 1： `ap-south-1`
  - ワークロード (テスト) アカウント 2： `eu-central-1`
  - ワークロード (テスト) アカウント 3： `us-east-1`
- 各ワークロードアカウントのデプロイ用の 3 つの Amazon Simple Storage Service (Amazon S3) バケット。(これらはこのパターンで、`S3BUCKETNAMETESTACCOUNT1`、`S3BUCKETNAMETESTACCOUNT2`、`S3BUCKETNAMETESTACCOUNT3` と呼ばれています。)

  たとえば、これらのバケットは、特定のアカウントとリージョンに次のように固有のバケット名で作成できます (xxxx をランダムなナンバーに置き換える)。

```
##In Test Account 1
aws s3 mb s3://ecs-codepipeline-xxxx-ap-south-1 --region ap-south-1
##In Test Account 2
aws s3 mb s3://ecs-codepipeline-xxxx-eu-central-1 --region eu-central-1
##In Test Account 3
aws s3 mb s3://ecs-codepipeline-xxxx-us-east-1 --region us-east-1

#Example
##In Test Account 1
aws s3 mb s3://ecs-codepipeline-18903-ap-south-1 --region ap-south-1
##In Test Account 2
aws s3 mb s3://ecs-codepipeline-18903-eu-central-1 --region eu-central-1
##In Test Account 3
aws s3 mb s3://ecs-codepipeline-18903-us-east-1 --region us-east-1
```

制約事項

このパターンは、AWS CodeBuild とその他の設定ファイルでサンプルマイクロサービスをデプロイします。別のワークロードタイプ (サーバーレスなど) を使用している場合は、関連する設定をすべて更新する必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation
- AWS CodeCommit
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

ターゲットアーキテクチャ

自動化とスケール

設定は、AWS CloudFormation スタックテンプレート (IaC) を使用して自動化されます。複数の環境やアカウントに合わせて簡単にスケールすることができます。

ツール

AWS サービス

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- AWS CodeBuild は完全マネージド型の構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodeDeploy は、Amazon Elastic Compute Cloud (Amazon EC2) またはオンプレミスインスタンス、AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- 「Amazon Elastic Container Registry (Amazon ECR)」は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。

- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- 「Git」は、AWS CodeCommit リポジトリと連携するオープンソースの分散型バージョン管理システムです。
- 「Docker」は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するPlatform as a Service (PaaS) 製品のセットです。このパターンでは、Docker でコンテナイメージをローカルでビルドしてテストします。
- 「cfn-lint」と「cfn-nag」は、CloudFormation スタックにエラーやセキュリティ上の問題がないかを確認することに役立つオープンソースツールです。

コードリポジトリ

このパターンのコードは、GitHub 内の「複数のリージョンとアカウントのグローバルブルー/グリーンデプロイ」リポジトリで利用できます。

エピック

環境変数のセットアップ

タスク	説明	必要なスキル
CloudFormation スタックデプロイ用の環境変数を出力します。	このパターンの後半で CloudFormation スタックへの入力として使用する環境変数を定義します。 1. 前述の「前提条件」セクションで説明したように、3 つのアカウントとリ	AWS DevOps

タスク	説明	必要なスキル

	ージョンで作成したバケット名を更新します。
	
	`export S3BUCKETNAMETESTACCOUNT1=<S3BUCKETACCOUNT1>`
	`export S3BUCKETNAMETESTACCOUNT2=<S3BUCKETACCOUNT2>`
	`export S3BUCKETNAMETESTACCOUNT3=<S3BUCKETACCOUNT3>`
	
	2. バケット名はグローバルに一意である必要があるため、ランダムな文字列を定義してアーティファクトバケットを作成します。
	
	`export BUCKETSTARTNAME=ecs-codepipeline-artifacts-19992`
	
	3. アカウント ID とリージョンを定義して出力します。
	
	`export TOOLSACCOUNT=<TOOLSACCOUNT>`
	`export CODECOMMITACCOUNT=<CODECOMMITACCOUNT>`
	`export CODECOMMITREGION=ap-south-1`
	`export CODECOMMITREPONAME=Poc`
	`export TESTACCOUNT1=<TESTACCOUNT1>`
	`export TESTACCOUNT2=<TESTACCOUNT2>`	

タスク	説明	必要なスキル
	`export TESTACCOU NT3=<TESTACCOUNT3> export TESTACCOU NT1REGION=ap-south -1 export TESTACCOU NT2REGION=eu-centr al-1 export TESTACCOU NT3REGION=us-east-1 export TOOLSACCO UNTREGION=ap-south -1 export ECRREPOSI TORYNAME=web`	

インフラストラクチャ用にCloudFormation スタックをパッケージしてデプロイ

タスク	説明	必要なスキル
リポジトリをクローン作成します。	「サンプルリポジトリ」を作業場所の新しいリポジトリにクローンを作成します。 `##In work location git clone https://g ithub.com/aws-samp les/ecs-blue-green -global-deployment- with-multiregion-cmk- codepipeline.git`	AWS DevOps
Cloudformation リソースをパッケージします。	このステップでは、CloudFormation テンプレートが参照するローカルアーティファクトをパッケージして、Amazon Virtual Private	AWS DevOps

タスク	説明	必要なスキル

	Cloud (Amazon VPC) や Application Load Balancer などのサービスに必要なインフラストラクチャリソースを作成します。
	
	このテンプレートはコードリポジトリの `Infra` フォルダで利用できます。
	
	`##In TestAccount1##`
	`aws cloudformation package \`
	`--template-file mainInfraStack.yaml \`
	`--s3-bucket $S3BUCKETNAMETESTACCOUNT1 \`
	`--s3-prefix infraStack \`
	`--region $TESTACCOUNT1REGION \`
	`--output-template-file infrastructure_${TESTACCOUNT1}.template`
	
	`##In TestAccount2##`
	`aws cloudformation package \`
	`--template-file mainInfraStack.yaml \`
	`--s3-bucket $S3BUCKETNAMETESTACCOUNT2 \`
	`--s3-prefix infraStack \`
	`--region $TESTACCOUNT2REGION \`	

タスク	説明	必要なスキル
	--output-template-file infrastructure_${TESTACCOUNT2}.template ##In TestAccount3## aws cloudformation package \ --template-file mainInfraStack.yaml \ --s3-bucket $S3BUCKETNAMETESTACCOUNT3 \ --s3-prefix infraStack \ --region $TESTACCOUNT3REGION \ --output-template-file infrastructure_${TESTACCOUNT3}.template	

タスク	説明	必要なスキル

パッケージテンプレートを検証します。	パッケージテンプレートを検証します。
	
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT1}.template`
	
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT2}.template`
	
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT3}.template`
		AWS DevOps

タスク	説明	必要なスキル

パッケージファイルをワークロードアカウントにデプロイ	1. 設定に基づき、infraParameters.json スクリプトのプレースホルダー値とアカウント名を更新します。
	2. パッケージテンプレートを 3 つのワークロードアカウントにデプロイします。
	
	`##In TestAccount1##`
	`aws cloudformation deploy \`
	`--template-file infrastructure_${TESTACCOUNT1}.template \`
	`--stack-name mainInfrastack \`
	`--parameter-overrides file://infraParameters.json \`
	`--region $TESTACCOUNT1REGION \`
	`--capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM`
	
	`##In TestAccount2##`
	`aws cloudformation deploy \`
	`--template-file infrastructure_${TESTACCOUNT2}.template \`
	`--stack-name mainInfrastack \`
		AWS DevOps

タスク	説明	必要なスキル
	--parameter-overrides file://infraParameters.json \ --region $TESTACCOUNT2REGION \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM ##In TestAccount3## aws cloudformation deploy \ --template-file infrastructure_${TESTACCOUNT3}.template \ --stack-name mainInfrastack \ --parameter-overrides file://infraParameters.json \ --region $TESTACCOUNT3REGION \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM	

サンプルイメージをプッシュして Amazon ECS をスケールします。

タスク	説明	必要なスキル
Amazon ECR リポジトリにサンプルイメージをプッシュします	(パラメータで設定されているように) web という Amazon Elastic Container Registry (Amazon ECR) リポジトリに、サンプル (NGINX) イメー	AWS DevOps

タスク	説明	必要なスキル

	ジをプッシュします。このイメージは必要に応じてカスタマイズできます。
	
	ログインして、Amazon ECR にイメージをプッシュするための認証情報を設定するには、「Amazon ECR のドキュメント」の指示に従ってください。
	
	コマンドは以下のとおりです。
	
	`docker pull nginx`
	`docker images`
	`docker tag <imageid> aws_account_id.dkr.ecr.region.amazonaws.com/<web>:latest`
	`docker push <aws_account_id>.dkr.ecr.<region>.amazonaws.com/<web>:tag`	

タスク	説明	必要なスキル
Amazon ECS をスケールしてアクセスを検証します。	1. Amazon ECS をスケールして 2 つのレプリカを作成します。 `aws ecs update-service --cluster QA-Cluster --service Poc-Service --desired-count 2` `Poc-Service` はサンプルアプリケーションを指します。 2. ブラウザから完全修飾ドメイン名 (FQDN) または DNS を使用するか、curl コマンドを使用して、Application Load Balancer からサービスにアクセスできることを確認します。	AWS DevOps

コードサービスとリソースを設定

タスク	説明	必要なスキル
ツール アカウントに CodeCommit リポジトリを作成します。	GitHub リポジトリの `code` フォルダにある `codecommit.yaml` テンプレートを使用して、ツールアカウントに CodeCommit リポジトリを作成します。このリポジトリは、コードを開発する予定の 1 つのリージョンにのみ作成する必要があります。	AWS DevOps

タスク	説明	必要なスキル
	`aws cloudformation deploy --stack-name codecommitrepoStack --parameter-overrides CodeCommitReponame=$CODECOMMITREPONAME \ ToolsAccount=$TOOLSACCOUNT --template-file codecommit.yaml --region $TOOLSACCOUNTREGION \ --capabilities CAPABILITY_NAMED_IAM`	

タスク	説明	必要なスキル
CodePipeline によって生成されたアーティファクトを管理するための S3 バケットを作成します。	GitHub リポジトリの code フォルダにある pre-reqs-bucket.yaml テンプレートを使用して、CodePipeline によって生成されたアーティファクトを管理するための S3 バケットを作成します。スタックは 3 つのワークロード (テスト) とツールのアカウントとリージョンにすべてデプロイする必要があります。 `aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TESTACCOUNT1REGION --capabilities CAPABILITY_NAMED_IAM`  `aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketSta`	AWS DevOps

タスク	説明	必要なスキル
	rtName=$BUCKETSTAR TNAME \ TestAccount1=$TE STACCOUNT1 TestAccou nt2=$TESTACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ TestAccount1=$TE STACCOUNT1 TestAccou nt2=$TESTACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT3REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts	

タスク	説明	必要なスキル
	-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TOOLSACCOUNTREGION --capabilities CAPABILITY_NAMED_IAM	

タスク	説明	必要なスキル
マルチリージョン KMS キーを設定します。	1. CodePipeline が使用するプライマリキーとレプリカキーを持つマルチリージョン KMS キーを作成します。この例では、`ToolsAccount1region - ap-south-1` がプライマリリージョンになります。 `aws cloudformation deploy --stack-name ecs-codepipeline-pre-reqs-KMS \ --template-file pre-reqs_KMS.yaml --parameter-overrides \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT --region $TOOLSACCOUNTREGION` 2. CMKARN 変数を設定して CodeBuild プロジェクトに渡します。この値は ecs-codepipeline-pre-reqs-KMS テンプレートスタックの出力で利用できます (キー ID はすべてのリージョンで同じで、`mrk-` で始まります)。または、ツールアカウントから CMKARN 値を取	AWS DevOps

タスク	説明	必要なスキル

	得することもできます。すべてのアカウントセッションでそれらを出力します。
	
	`export CMKARN1=arn:aws:kms:ap-south-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMKARN2=arn:aws:kms:eu-central-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMKARN3=arn:aws:kms:us-east-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMARNTOOLS=arn:aws:kms:ap-south-1:<TOOLSACCOUNTID>:key/mrk-xxx`	

タスク	説明	必要なスキル

ツールアカウントで CodeBuild プロジェクトを設定します。	1. GitHub リポジトリの code フォルダにある codebuild_IAM.yaml テンプレートを使用して、ツールアカウントの単一リージョンに AWS CodeBuild 用の AWS アイデンティティとアクセス管理 (IAM) を設定します。
	`#In ToolsAccount`
	`aws cloudformation deploy --stack-name ecs-codebuild-iam \`
	`--template-file codebuild_IAM.yaml --region $TOOLSACCOUNTREGION \`
	`--capabilities CAPABILITY_NAMED_IAM`
	2. codebuild.yaml テンプレートを使用して、ビルドプロジェクト用に CodeBuild を設定します。このテンプレートを 3 つのリージョンに次のようにすべてデプロイします。
	`aws cloudformation deploy --stack-name ecscodebuildstack --parameter-overrides ToolsAccount=$TOOLSACCOUNT \`
	`CodeCommitRepoName=$CODECOMMITREPONAME`
		AWS DevOps

タスク	説明	必要なスキル
	``` ECRRepositoryName= $ECRREPOSITORYNAME APPACCOUNTID=$TEST ACCOUNT1 \ TestAccount3=$TE STACCOUNT3 CodeCommi tRegion=$CODECOMMI TREGION CMKARN=$C MKARN1 \ --template-file codebuild.yaml --region $TESTACCO UNT1REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name ecscodebuildstack -- parameter-overrides ToolsAccount=$TOOL SACCOUNT \ CodeCommitRepoName= $CODECOMMITREPONAME ECRRepositoryName= $ECRREPOSITORYNAME APPACCOUNTID=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tRegion=$CODECOMMI TREGION CMKARN=$C MKARN2 \ --template-file codebuild.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name ```	

タスク	説明	必要なスキル
	ecscodebuildstack --parameter-overrides ToolsAccount=$TOOLSACCOUNT \ CodeCommitRepoName=$CODECOMMITREPONAME ECRRepositoryName=$ECRREPOSITORYNAME APPACCOUNTID=$TESTACCOUNT3 \ CodeCommitRegion=$CODECOMMITREGION CMKARN=$CMKARN3 \ --template-file codebuild.yaml --region $TESTACCOUNT3REGION --capabilities CAPABILITY_NAMED_IAM	

タスク	説明	必要なスキル

ワークロードアカウントに CodeDeploy を設定します。	GitHub リポジトリの `code` フォルダにある `codedeploy.yaml` テンプレートを使用して、3 つのワークロードアカウントにすべて CodeDeploy を設定します。`mainInfraStack` の出力には、Amazon ECS クラスターの Amazon リソースネーム (ARN) と Application Load Balancer リスナーが含まれます。
	
	Note
	インフラストラクチャスタックの値はエクスポート済みであるため、CodeDeploy スタックテンプレートによってインポートされます。
	
	`##WorkloadAccount1##`
	`aws cloudformation deploy --stack-name ecscodedeploystack \`
	`--parameter-overrides ToolsAccount=$TOOLSACCOUNT mainInfrastackname=mainInfrastack \`
	`--template-file codedeploy.yaml --region $TESTACCOUNT1REGION --capabil`
		AWS DevOps

タスク	説明	必要なスキル
	ities CAPABILIT Y_NAMED_IAM ##WorkloadAccount2## aws cloudformation deploy --stack-name ecscodedeploystack \ --parameter-overrides ToolsAccount=$TOOL SACCOUNT mainInfra stackname=mainInfr astack \ --template-file codedeploy.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM ##WorkloadAccount3## aws cloudformation deploy --stack-name ecscodedeploystack \ --parameter-overrides ToolsAccount=$TOOL SACCOUNT mainInfra stackname=mainInfr astack \ --template-file codedeploy.yaml --region $TESTACCO UNT3REGION --capabil ities CAPABILIT Y_NAMED_IAM	

ツールアカウントで CodeBuild プロジェクトを設定

タスク	説明	必要なスキル
ツールアカウントにコードパイプラインを作成します。	ツールアカウントで、以下のコマンドを実行します。 `aws cloudformation deploy --stack-name ecscodepipelinestack --parameter-overrides \ TestAccount1=$TESTACCOUNT1 TestAccount1Region=$TESTACCOUNT1REGION \ TestAccount2=$TESTACCOUNT2 TestAccount2Region=$TESTACCOUNT2REGION \ TestAccount3=$TESTACCOUNT3 TestAccount3Region=$TESTACCOUNT3REGION \ CMKARNTools=$CMKTROOLSARN CMKARN1=$CMKARN1 CMKARN2=$CMKARN2 CMKARN3=$CMKARN3 \ CodeCommitRepoName=$CODECOMMITREPONAME BucketStartName=$BUCKETSTARTNAME \ --template-file codepipeline.yaml --capabilities CAPABILITY_NAMED_IAM`	AWS DevOps
AWS KMS キーポリシーと S3 バケットポリシーの CodePipeline と CodeBuild	1. AWS KMS キーポリシーの CodePipeline と CodeBuild	AWS DevOps

タスク	説明	必要なスキル
ロールへのアクセスを提供します。	ロールへのアクセスを提供します。 ``` aws cloudformation deploy --stack-name ecs-codepipeline-p re-reqs-KMS \ --template-file pre- reqs_KMS.yaml -- parameter-overrides \ CodeBuildConditi on=true TestAccou nt1=$TESTACCOUNT1 TestAccount2=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT --region $TOOLSACC OUNTREGION ``` 2. CodePipeline と CodeDeploy ロールへのアクセスを許可するように S3 バケットポリシーを更新します。 ``` aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ PutS3BucketPolic y=true TestAccou nt1=$TESTACCOUNT1	

タスク	説明	必要なスキル
	TestAccount2=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT1REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ PutS3BucketPolic y=true TestAccou nt1=$TESTACCOUNT1 TestAccount2=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter-	

タスク	説明	必要なスキル
	overrides BucketStartName=$BUCKETSTARTNAME \ PutS3BucketPolicy=true TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TESTACCOUNT3REGION --capabilities CAPABILITY_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ PutS3BucketPolicy=true TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TOOLSACCOUNTREGION --capabil	

タスク	説明	必要なスキル
	`ities CAPABILIT Y_NAMED_IAM`	

パイプラインを呼び出してテスト

タスク	説明	必要なスキル
変更を CodeCommit リポジトリにプッシュします。	1. 「AWS CodeCommit のドキュメント」の説明に従い、`codecommi trepoStack` で git clone コマンドを使用して作成された CodeCommit リポジトリの子ローンを作成します。 2. 入力アーティファクトを必要な詳細で更新します。 • JSON ファイル：このファイルの3 か所にあるファイルの `AccountID` を更新します。アカウント ID を含む 3 つのファイル名を変更します。 • YAML ファイル：タスク定義 ARN とバージョンを更新します。アカウント ID を含む 3 つのファイル名を変更します。 3. `index.html` ファイルを変更して、ホームページにいくつかの変更を加えます。	

タスク	説明	必要なスキル

	4. 以下のファイルをリポジトリにコピーしてコミットします。
	
	`index.html`
	`Dockerfile`
	`buildspec.yaml`
	`appspec_<accountid>.yaml (3 files - one per account )`
	`taskdef<accountid>.json (3 files - one per account)`
	
	5. パイプラインを起動または再起動し、結果を確認します。
	6. FQDN または DNS を使用してApplication Load Balancer からサービスにアクセスし、更新がデプロイされていることを確認します。	

クリーンアップ

タスク	説明	必要なスキル
デプロイされたすべてのリソースをクリーンアップします。	1. Amazon ECS をゼロインスタンスにスケールダウンします。 `aws ecs update-service --cluster QA-Cluster --service`	

タスク	説明	必要なスキル
	Poc-Service -- desired-count 0 2. 各アカウントとリージョンのCloudFormation スタックを削除します。 ##In Tools Account## aws cloudformation delete-stack -- stack-name ecscodepi pelinestack --region $TOOLSACCOUNTREGION aws cloudformation delete-stack -- stack-name ecscodebu ildstack --region $TESTACCOUNT1REGION aws cloudformation delete-stack -- stack-name ecscodebu ildstack --region $TESTACCOUNT2REGION aws cloudformation delete-stack -- stack-name ecscodebu ildstack --region $TESTACCOUNT3REGION aws cloudformation delete-stack -- stack-name ecs-codep ipeline-pre-reqs-K MS --region $TOOLSACC OUNTREGION aws cloudformation delete-stack -- stack-name codecommi trepoStack --region $TOOLSACCOUNTREGION aws cloudformation delete-stack --	

タスク	説明	必要なスキル
	stack-name pre-reqs-artifacts-bucket --region $TESTACCOUNT1REGION aws cloudformation delete-stack --stack-name pre-reqs-artifacts-bucket --region $TESTACCOUNT2REGION aws cloudformation delete-stack --stack-name pre-reqs-artifacts-bucket --region $TESTACCOUNT3REGION aws cloudformation delete-stack --stack-name pre-reqs-artifacts-bucket --region $TOOLSACCOUNTREGION aws cloudformation delete-stack --stack-name ecs-codebuild-iam --region $TOOLSACCOUNTREGION ##NOTE: Artifact buckets will not get deleted if there are artifacts so it has to be emptied manually before deleting.## ##In Workload / Test Accounts## ##Account:1## aws cloudformation delete-stack --	

タスク	説明	必要なスキル
	stack-name ecscodedeployストック --region $TESTACCOUNT1REGION aws cloudformation delete-stack --stack-name mainInfrastack --region $TESTACCOUNT1REGION ##Account:2## aws cloudformation delete-stack --stack-name ecscodedeployストック --region $TESTACCOUNT2REGION aws cloudformation delete-stack --stack-name mainInfrastack --region $TESTACCOUNT2REGION ##Account:3## aws cloudformation delete-stack --stack-name ecscodedeployストック --region $TESTACCOUNT3REGION aws cloudformation delete-stack --stack-name mainInfrastack --region $TESTACCOUNT3REGION ##NOTE: Amazon ECR (web) will not get deleted if the registry still includes images. It can be manually cleaned up if not required.	

トラブルシューティング

問題	ソリューション
リポジトリにコミットした変更はデプロイされません。	• CodeBuild ログに Docker ビルド操作のエラーがないか確認します。CodeBuild の詳細については、「のドキュメント」を参照してください。 • CodeDeploy のデプロイをチェックして、Amazon ECS のデプロイに問題がないか確認します。

関連リソース

- 「Docker イメージをプッシュする」(Amazon ECR のドキュメント)
- 「AWS CodeCommit リポジトリに接続する」(AWS CodeCommit のドキュメント)
- 「AWS CodeBuild のトラブルシューティング」(AWS CodeBuild のドキュメント)

AWS CloudFormation と AWS Config を使用して Amazon ECR リポジトリのワイルドカード権限をモニタリングする

作成者：Vikrant Telkar (AWS)、Sajid Momin (AWS), と Wassim Benhallam (AWS)

概要

Amazon Web Services (AWS) クラウド上のAmazon Elastic Container Registry (Amazon ECR) は AWS Identity and Access Management (IAM) を使用したリソースベースのアクセス権限によるプライベートリポジトリをサポートする マネージドコンテナイメージレジストリサービスです。

IAM はリソース属性とアクション属性の両方で「*」ワイルドカードをサポートしているため、一致する複数の項目を簡単に自動的に選択できます。テスト環境では、「リポジトリポリシーステートメント」のプリンシパル要素にある `ecr:*` 「ワイルドカードアクセス権限」を使用して、認証されたすべての AWS ユーザーに Amazon ECR リポジトリへのアクセスを許可できます。`ecr:*` ワイルドカードアクセス権限は、本稼働データにアクセスできない開発アカウントで開発やテストを行う場合に役立ちます。

ただし、`ecr:*` ワイルドカード権限は重大なセキュリティ上の脆弱性を引き起こす可能性があるため、本番環境では使用しないようにする必要があります。このパターンのアプローチは、`ecr:*` リポジトリポリシーステートメントにワイルドカード権限が含まれている Amazon ECR リポジトリを特定することに役立ちます。 このパターンには、AWS Config でカスタムルールを作成するためのステップと AWS CloudFormation テンプレートが含まれています。次に、AWS Lambda 関数が Amazon ECR リポジトリのポリシーステートメントを監視して、`ecr:*` ワイルドカードアクセス権限を確認します。準拠していないリポジトリポリシーステートメントが見つかると、Lambda は Amazon EventBridge にイベントを送信するよう AWS Config に通知し、EventBridge は Amazon Simple Notification Service (Amazon SNS) トピックを開始します。SNS トピックは、非準拠のリポジトリポリシーステートメントについてメールで通知します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI) がインストール済みおよび設定済み。詳細については、AWS CLI ドキュメントの「AWS CLI バージョン 2 のインストール、更新、およびアンインストール」を参照してください。
- ポリシーステートメントが添付された既存の Amazon ECR リポジトリがテスト環境にインストールし、設定されています。詳細については、「Amazon ECR のドキュメント」の「プライベート

リポジトリを作成する」と「リポジトリポリシーステートメントを設定する」を参照してください。

- お好みの AWS リージョンで設定された AWS Config。詳細については、「AWS Config のドキュメント」の「AWS Config の開始方法」を参照してください。
- `aws-config-cloudformation.template` ファイル (添付) は、ローカルマシンにダウンロードされます。

機能制限

- このパターンのソリューションはリージョナルで、リソースは、同じリージョンに作成されます。

アーキテクチャ

次の図は、AWS Config が Amazon ECR リポジトリポリシーステートメントを評価する方法を示しています。

この図表は、次のワークフローを示しています：

1. AWS Config はカスタムルールを開始します。
2. カスタムルールは Lambda 関数を呼び出して Amazon ECR リポジトリポリシーステートメントのコンプライアンスを評価します。次に、Lambda 関数は非準拠のリポジトリポリシーステートメントを識別します。
3. Lambda 関数はコンプライアンス違反ステータスを AWS Config に送信します。
4. AWS Config は EventBridge にイベントを送信します。
5. EventBridge はコンプライアンス違反通知を SNS トピックに公開します。
6. Amazon SNS は、ユーザーや承認されたユーザーにメールアラートを送信します。

自動化とスケール

このパターンのソリューションでは、Amazon ECR リポジトリのポリシーステートメントをいくつでもモニタリングできますが、評価するリソースはすべて同じリージョンで作成されている必要があります。

ツール

- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理およびプロビジョニングすることが可能です。
- 「AWS Config」— AWS Config は、AWS アカウント内の AWS リソースの設定の詳細なビューを提供します。これには、リソース間の関係と設定の履歴が含まれるため、時間の経過と共に設定と関係がどのように変わるかを確認できます。
- 「Amazon ECR」— Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。Amazon ECR は、 IAM を使用するリソースベースの許可を持つプライベートリポジトリをサポートします。
- 「Amazon EventBridge」— Amazon EventBridgeは、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。EventBridge は、アプリケーション、software as a service (SaaS) アプリケーション、および AWS サービスからのリアルタイムデータのストリームを、AWS Lambda 関数、API デスティネーションを使用した HTTP 呼び出しエンドポイント、または他のアカウントのイベントバスなどのターゲットに配信します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージ配信や送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンのコードは `aws-config-cloudformation.template` ファイル (添付) にあります。

エピック

AWS CloudFormation スタックを作成する

タスク	説明	必要なスキル
AWS CloudFormation スタックを作成する	AWS CLI で次のコマンドを実行して AWS CloudFormation スタックを作成します。 `$ aws cloudformation create-stack --stack-name=AWSConfigECR \ --template-body file://aws-config-cloudformation.template \ --parameters ParameterKey=<email>,ParameterValue=<myemail@example.com> \ --capabilities CAPABILITY_NAMED_IAM`	AWS DevOps

AWS Config カスタムルールをテスト

タスク	説明	必要なスキル
AWS Config カスタムルールをテストします。	1. AWS マネジメントコンソールにサインインし、AWS Config コンソールを開いて、[リソース] を選択します。 2. [リソース インベントリ] ページでは、リソース カテゴリ、リソース タイプ、およびコンプライアンス ス	AWS DevOps

タスク	説明	必要なスキル
	テータスでフィルタリングできます。 3. `ecr:*` を含む Amazon ECR リポジトリが `NON-COMPLIANT` であり、`ecr:*` を含まない Amazon ECR リポジトリが `COMPLIANT` です。 4. Amazon ECR リポジトリに非準拠のポリシーステートメントが含まれている場合、SNS トピックに登録されているメールアドレスに通知が届きます。	

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する

作成者: Sarat Chandra Pothula (AWS) と VAMSI KRISHNA SUNKAVALLI (AWS)

概要

サーバーレスインフラストラクチャを複数の AWS アカウント および環境にデプロイしている組織では、コードの重複、手動プロセス、一貫性のないプラクティスなどの課題に直面することがよくあります。このパターンのソリューションは、Go および GitHub Actions の再利用可能なワークフロー AWS Cloud Development Kit (AWS CDK) で を使用して、マルチアカウントのサーバーレスインフラストラクチャ管理を合理化する方法を示しています。このソリューションは、クラウドリソースをコードとして定義し、標準化された継続的インテグレーション/継続的デプロイ (CI/CD) プロセスを実装し、モジュール式の再利用可能なコンポーネントを作成する方法を示しています。

これらのツールを使用することで、組織はクロスアカウントリソースを効率的に管理し、一貫したデプロイパイプラインを実装し、複雑なサーバーレスアーキテクチャを簡素化できます。このアプローチは、 で使用するための標準化されたプラクティスを適用することでセキュリティとコンプライアンスを強化し AWS アカウント、最終的には生産性を向上させ、サーバーレスアプリケーションの開発とデプロイのエラーを減らします。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Identity and Access Management (IAM) ロールとアクセス許可がデプロイプロセスで導入されています。これには、Amazon Elastic Container Registry (Amazon ECR) リポジトリへのアクセス、 AWS Lambda 関数の作成、およびターゲット全体のその他の必要なリソースへのアクセス許可が含まれます AWS アカウント。
- AWS Command Line Interface （ AWS CLI) バージョン 2.9.11 以降、インストールおよび設定済み。
- AWS Cloud Development Kit (AWS CDK) バージョン 2.114.1 以降、インストールおよびブートストラップ。
- Go 1.22 以降がインストールされています。
- Docker 24.0.6 以降がインストールされている。

制約事項

- 言語の互換性 – Go はサーバーレスアプリケーションに人気のある言語です。ただし、Go に加えて、 は C#、Java、Python、TypeScript などの他のプログラミング言語 AWS CDK をサポートしています。組織に他の言語の既存のコードベースや専門知識がある場合は、パターンで説明されているソリューションを完全に使用するために Go を適応させるか、学習する必要がある場合があります。
- 学習曲線 – Go (組織を初めて使用する場合 ） AWS CDK、GitHub の再利用可能なワークフローを採用すると、デベロッパーや DevOps チーム向けの学習曲線が必要になる場合があります。これらのテクノロジーをスムーズに導入し、効果的に使用するために、トレーニングとドキュメントが必要になる場合があります。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

このソリューションでは、次のステップを実行します。

1. 開発者はリポジトリのクローンを作成し、新しいブランチを作成し、ローカル環境でアプリケーションコードを変更します。
2. 開発者はこれらの変更をコミットし、新しいブランチを GitHub リポジトリにプッシュします。
3. 開発者は GitHub リポジトリにプルリクエストを作成し、機能または新機能ブランチをメインブランチにマージすることを提案します。
4. このプルリクエストは、継続的インテグレーション (CI) の GitHub Actions ワークフローをトリガーします。このパターンの CI および継続的デプロイ (CD) ワークフローは、再利用可能なワークフローを使用します。再利用可能なワークフローは、さまざまなプロジェクトまたはリポジトリ間で共有および実行できる、事前定義されたモジュラーテンプレートです。再利用可能なワークフローは、CI/CD プロセスの標準化と効率を向上させます。
5. CI ワークフローは、必要な環境を設定し、イメージの Docker タグを生成し、アプリケーションコードを使用して Docker イメージを構築します。
6. CI ワークフローは central AWS アカウント GitHub OIDC ロールを使用して AWS で認証します。CI ワークフローの場合、 central AWS アカウント GitHub OIDC ロールは AWS Security Token Service （ AWS STS) を使用して一時的な認証情報を取得します。これらの認証情報によ

り、ロールは Docker イメージを構築して中央の Amazon ECR リポジトリにプッシュできます AWS アカウント。

7. CI ワークフローは、構築された Docker イメージを Amazon ECR にプッシュします。
8. CI ワークフローは、イメージタグを Systems Manager パラメータストアに保存します。
9. CI ワークフローが正常に完了すると、Docker イメージタグが出力されます。
10. CD ワークフローをトリガーすると、デベロッパーはデプロイする Docker イメージのイメージタグを手動で入力します。このイメージタグは、CI ワークフロー中に生成され、Amazon ECR にプッシュされたタグに対応します。
11. 開発者は CD ワークフローを手動でトリガーします。CD ワークフローは CD 再利用可能なワークフローを使用します。
12. CD ワークフローは central AWS アカウント GitHub OIDC ロール AWS を使用して で認証します。CD ワークフローの場合、 AWS STS 最初に central AWS アカウント GitHub OIDC ロールを引き受けるために使用されます。次に、このロールはターゲットアカウントのデプロイの CDK ブートストラップロールを引き受けます。
13. CD ワークフローは、 を使用して AWS CloudFormation テンプレート AWS CDK を合成します。
14. CD ワークフローは、Lambda 関数に手動で指定されたイメージタグ AWS アカウント を使用して、CDK デプロイを使用してアプリケーションをターゲットにデプロイします。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。CloudFormation は、 AWS CDK デプロイプロセスの不可欠な部分です。CDK は CloudFormation テンプレートを合成し、CloudFormation を使用して AWS 環境内のリソースを作成または更新します。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Systems Manager Parameter Store は、設定データ管理とシークレット管理のための安全な階層型ストレージを提供します。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- GitHub Actions は、GitHub リポジトリと緊密に統合された継続的インテグレーションおよび継続的デリバリー (CI/CD) プラットフォームです。GitHub Actions を使用して、ビルド、テスト、デプロイパイプラインを自動化できます。
- Go は、Google がサポートするオープンソースのプログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub aws-cdk-golang-serverless-cicd-github-actions リポジトリで入手できます。

ベストプラクティス

- モジュラー設計 – AWS CDK コードをモジュール式で再利用可能なコンストラクトまたはスタックに整理し、複数のアカウントやプロジェクトでコードの再利用と保守性を促進します。
- 懸念事項の分離 – インフラストラクチャコードをアプリケーションコードから分離し、各コンポーネントの独立したデプロイと管理を可能にします。
- バージョニングとイミュータビリティ – インフラストラクチャをコードとして扱い (IaC)、バージョン管理に Git を使用します。既存のリソースを変更するのではなく、新しいリソースを作成して、イミュータブルなインフラストラクチャの原則を採用します。
- テストと検証 – コードと AWS CDK デプロイの正確性と信頼性をサポートするために、ユニットテスト、統合テスト、end-to-endテストを含む包括的なテスト戦略を実装します。
- セキュリティとコンプライアンス – 最小特権アクセス、安全な通信、データ暗号化などの AWS セキュリティのベストプラクティスに従います。コンプライアンスチェックと監査メカニズムを実装して、組織のポリシーと規制要件に準拠していることを確認します。脆弱性のスキャン、イメージ署名の強制、組織のコンプライアンス要件の遵守など、コンテナイメージのセキュリティのベストプラクティスを実装します。

- モニタリングとログ記録 — サーバーレスアプリケーションとインフラストラクチャのヘルスとパフォーマンスを追跡するためのモニタリングとログ記録メカニズムを設定します。Amazon CloudWatch AWS のサービス と同様に を使用し AWS CloudTrail、モニタリングと監査 AWS X-Ray の目的で使用します。
- 自動化と CI/CD – GitHub の再利用可能なワークフローやその他の CI/CD ツールを使用して、ビルド、テスト、デプロイのプロセスを自動化します。これにより、複数のアカウント間で一貫性のある反復可能なデプロイをサポートできます。
- 環境管理 – 個別の環境 (開発、ステージング、本番稼働など) を維持します。環境間の変更を促進するための戦略を実装し、本番デプロイの前に適切なテストと検証を行います。
- ドキュメントとコラボレーション – インフラストラクチャコード、デプロイプロセス、ベストプラクティスを文書化して、チーム内での知識の共有とコラボレーションを容易にします。
- コスト最適化 — リソースの適正化、自動スケーリングの使用、 や などのコスト最適化サービスの利用など、 AWS コストのモニタリングと最適化戦略を実装 AWS Budgets します AWS Cost Explorer。
- ディザスタリカバリとバックアップ – サーバーレスアプリケーションとインフラストラクチャリソースのバックアップと復元メカニズムを実装して、ディザスタリカバリシナリオを計画します。
- 継続的な改善 – 定期的に見直し、プラクティス、ツール、プロセスを更新して、サーバーレスエコシステムにおける最新のベストプラクティス、セキュリティの推奨事項、技術的進歩に合わせます。
- セキュリティ体制の改善 – Amazon ECR のインターフェイス VPC エンドポイント、および Parameter Store を設定することで AWS Lambda、仮想プライベートクラウド (VPC) AWS Systems Manager のセキュリティ体制AWS PrivateLinkを改善するために使用します。

エピック

環境のセットアップ

タスク	説明	必要なスキル
中央で Amazon ECR リポジトリを作成します AWS アカウント。	複数の 間でコンテナイメージを共有するには AWS アカウント、Amazon ECR のクロスアカウントアクセスを設定する必要があります。まず、中央で Amazon ECR リポジトリ	AWS DevOps

タスク	説明	必要なスキル
	を作成します AWS アカウント。 Amazon ECR リポジトリを作成するには、次のコマンドを実行します。 `aws ecr create-repository --repository-name sample-repo` 後のタスクでは、コンテナイメージを使用する AWS アカウント 必要がある他の へのプルアクセスを許可します。	

タスク	説明	必要なスキル
Amazon ECR リポジトリにクロスアカウントアクセス許可を追加します。	中央の Amazon ECR リポジトリにクロスアカウントアクセス許可を追加するには AWS アカウント、次のコードを実行します。 `{ "Version": "2008-10-17", "Statement": [{ "Sid": "LambdaECRImageRetrievalPolicy", "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": ["ecr:BatchGetImage", "ecr:GetDownloadUrlForLayer",], "Condition": { "StringLike": { "aws:sourceArn": "arn:aws:lambda:<Target_Region>:<Target_Account_ID>:function:*" } } }, { "Sid": "new statement", "Effect": "Allow",`	AWS DevOps

タスク	説明	必要なスキル

	<pre>"Principal": {
	"AWS": "arn:aws:iam::<Target_Account_ID>:root"
	},
	"Action": [
	"ecr:BatchGetImage",
	"ecr:GetDownloadUrlForLayer",
	],
	}
	]
	}</pre>	

タスク	説明	必要なスキル
中央で GitHub OIDC ロールのロールを設定します AWS アカウント。	1. GitHub の OIDC をフェデレーティッドアイデンティティとして信頼 AWS するように を設定します。これには、GitHub OIDC プロバイダーを に追加 AWS し、IAM でロールと信頼ポリシーを設定します。これを行うには、GitHub ドキュメントの「Amazon Web Services での OpenID Connect の設定」の手順に従います。 2. ロールを作成したら、ロールに必要なアクセス許可を追加します。例えば、Amazon ECR と Parameter Store AWS Systems Manager のアクセス許可を追加します。詳細については、IAM ドキュメントのGitHub OIDC ID プロバイダーのロールの設定」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
ターゲットで AWS 環境をブートストラップします AWS アカウント。	特定の AWS アカウント および で CDK 環境をセットアップ AWS リージョン し、中央アカウントからのクロスアカウントデプロイを有効にして、最小特権の原則を CloudFormation 実行ロールに適用します。 AWS 環境をブートストラップするには、次のコマンドを実行します。 `cdk bootstrap aws://<Target_Account_ID>/<Target_Region> --trust <Central_Account_ID> --cloudformation-execution-policies arn:aws:iam::aws:policy/<Least_Privilege_Policy>`	AWS DevOps

タスク	説明	必要なスキル
ターゲット AWS アカウントブートストラップロールへのアクセスを中央 AWS アカウント OIDC ロールに付与します。	CDK ブートストラップは、CDK デプロイプロセスの AWS アカウント さまざまな段階で中央 が引き受けるように設計された次の IAM ロールを作成します。 • ファイル発行ロール • イメージ発行ロール • ルックアップロール • デプロイロール 各ロールには、最小特権の原則に従って、目的に合わせた特定のアクセス許可があります。Target_Region 各ロール名の Target_Account_ID と は、これらのロールが異なる AWS アカウント およびリージョン間で一意であることを示すのに役立ちます。このアプローチは、マルチアカウント、マルチリージョンのセットアップで明確な識別と管理をサポートします。 `Target Account CDK Bootstrap Roles arn:aws:iam::<Target_Account_ID>:role/cdk-deploy-role-<Target_Account_ID>-<Target_Region>`	AWS DevOps

タスク	説明	必要なスキル
	arn:aws:iam::<Target_Account_ID>:role/cdk-file-publishing-role-<Target_Account_ID>-<Target_Region> arn:aws:iam::<Target_Account_ID>:role/cdk-image-publishing-role-<Target_Account_ID>-<Target_Region> arn:aws:iam::<Target_Account_ID>:role/cdk-lookup-role-<Target_Account_ID>-<Target_Region> • 中央アカウントの OIDC ロールのアクセス許可ポリシーを更新して、ターゲットアカウントのロールを引き受ける権限を付与します。この設定により、さまざまな に CDK スタックをデプロイできます AWS アカウント。中央アカウントの OIDC ロールがターゲットアカウントから必要なアクセス許可を採用できるようにすることで、クロスアカウント CDK デプロイ用の安全なブリッジを作成します。このアプローチは、シームレスなマルチアカウントインフラストラクチャ管理を容易にしながら、適切なアクセスコントロールを維持します。	

タスク	説明	必要なスキル
	中央の OIDC ロールのアクセス許可ポリシーを更新するには AWS アカウント、次のコードを使用します。 `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": ["arn:aws:iam::<Target_Account_ID>:role/cdk-deploy-role-<Target_Account_ID>-<Target_Region>”, “arn:aws:iam::<Target_Account_ID>:role/cdk-file-publishing-role-<Target_Account_ID>-<Target_Region>”, “arn:aws:iam::<Target_Account_ID>:role/cdk-image-publishing-role-<Target_Account_ID>-<Target_Region>”, “arn:aws:iam::<Target_Account_ID>:role/cdk-lookup-role-<Target_Account_ID>-<Target_Region>”]`	

タスク	説明	必要なスキル
	<pre> }] }</pre>	

Docker イメージの構築

タスク	説明	必要なスキル
プロジェクトリポジトリのクローンを作成します。	このパターンの GitHub リポジトリのクローンを作成するには、次のコマンドを実行します。 <pre>git clone https://github.com/aws-samples/aws-cdk-golang-serverless-cicd-github-actions.git</pre>	AWS DevOps
Dockerfile パスに移動します。	Dockerfile パスに移動するには、次のコマンドを実行します。 <pre>cd lambda</pre>	AWS DevOps
Amazon ECR で Docker を認証します。	Amazon ECR には、プライベートコンテナリポジトリへの安全なアクセスが必要です。この方法でサインインすると、ローカルマシンまたは CI/CD 環境で Docker が Amazon ECR と安全にやり取りできるようになります。	AWS DevOps

タスク	説明	必要なスキル
	Amazon ECR で Docker を認証するには、次のコマンドを実行します。 `aws ecr get-login -password --region <AWS_REGION> | docker login --username AWS --password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com` プレースホルダーAWS_REGION と を自分の情報AWS_Account_ID で修正します。	
Docker イメージを作成します。	Docker イメージを構築するには、次のコマンドを実行します。 `docker build --platform linux/arm64 -t sample-app .`	AWS DevOps

タスク	説明	必要なスキル

Docker イメージにタグを付けてプッシュします。	Docker イメージにタグを付けて Amazon ECR リポジトリにプッシュするには、次のコマンドを実行します。
	`docker tag sample-app:latest <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/<ECR_REPOSITORY>:<DOCKER_TAG>`
	`docker push <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/<ECR_REPOSITORY>:<DOCKER_TAG>`
	プレースホルダー `AWS_Account_ID` 、`AWS_REGION` 、`ECR_REPOSITORY` 、および `DOCKER_TAG` を自分の情報で修正します。
		AWS DevOps

AWS CDK アプリをデプロイする

タスク	説明	必要なスキル
CDK スタックを環境固有の変数で合成します。	CDK コードで定義されているインフラストラクチャの CloudFormation テンプレートを生成するには、次のコマンドを実行します。	AWS DevOps

タスク	説明	必要なスキル
	`ENV=<environment> IMAGETAG=<image_tag> ECR_ARN=<ecr_repo_arn> cdk synth` 以下のプレースホルダーをお客様の情報で修正します。 • `environment` – 、dev、stagingなどの特定の環境名に置き換えられますprod。 • `image_tag` – を、v1.0.0や などの Docker イメージの特定のタグに置き換えますlatest。 • `ecr_repo_arn` – を Amazon ECR リポジトリの Amazon リソースネーム (ARN) に置き換えます。	

タスク	説明	必要なスキル
CDK スタックをデプロイします。	CDK スタックを にデプロイするには AWS アカウント、次のコマンドを実行します。`--require-approval never` フラグは、CDK がすべての変更を自動的に承認して実行することを意味します。これには、CDK が通常手動レビューを必要とするものとしてフラグ付けする変更 (IAM ポリシーの変更やリソースの削除など) が含まれます。本番環境で `--require-approval never`フラグを使用する前に、CDK コードと CI/CD パイプラインが十分にテストされ、安全であることを確認します。 `ENV=<environment> IMAGETAG=<image_tag> ECR_ARN=<ecr_repo_arn> cdk deploy --require-approval never`	AWS DevOps

GitHub Actions ワークフローを使用して CI/CD を自動化する

タスク	説明	必要なスキル
機能ブランチを作成し、変更を追加します。	前に作成したクローンされたリポジトリを使用し、機能ブランチを作成してから、アプリケーションコードに変更を	AWS DevOps

タスク	説明	必要なスキル

	追加します。次のコマンドを使用します。
	
	`git checkout -b <feature_branch>`
	`git add .`
	`git commit -m “add your changes”`
	`git push origin <feature_branch>`
	
	変更の例を次に示します。
	
	• Lambda 関数ロジックの変更
	• Lambda コードへの新機能の追加
	• Lambda 関数内のバグの修正または既存のコードの最適化
	
	GitHub Actions は再利用可能なワークフローを使用し、CI/CD パイプラインをトリガーします。	

変更をマージします。	プルリクエストを作成し、変更を main にマージします。	AWS DevOps

トラブルシューティング

問題	ソリューション
`AccessDenied` リソースを にデプロイする際のエラー。例 AWS アカウント: `AccessDen`	この問題を解決するには、以下を実行してクロスアカウントのアクセス許可を確認します。

問題	ソリューション
`ied: User not authorized to perform: “sts:AssumeRole" 。`	• クロスアカウントデプロイに必要な IAM ロールとポリシーが設定されていることを確認します。 • assume ロールのアクセス許可が正しく設定されていることを確認します。
古い CDK バージョンの`undefined: awscdkStack` エラーなど、バージョンの不一致による互換性の問題。	この問題を解決するには、以下を実行して、必要なバージョンの AWS CDK と Go を使用していることを確認します。 • AWS CDK および Go の互換性のあるバージョンを使用していることを確認してください。 • 最新バージョンで既知の問題や重大な変更がないか確認します。
例えば、YAML 設定が正しくない、または保護されたブランチ`Error: No such file or directory` が原因で CI/CD パイプライン`Permission denied` が失敗します。	GitHub Actions 設定の問題を解決するには、再利用可能なワークフローが正しく参照および設定されていることを確認します。

関連リソース

「AWS リソース」

- AWS セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス
- AWS CDK ワークショップ
- AWS クラウド開発キットライブラリ
- コンテナイメージを使用した Lambda 関数の作成
- Amazon Elastic Container Registry の Identity and Access Management
- Go での AWS CDK の使用

その他のリソース

- Amazon Web Services での OpenID Connect の設定 (GitHub ドキュメント)
- Golang ドキュメント
- GitHub Actions のクイックスタート (GitHub ドキュメント)
- ワークフローの再利用 (GitHub ドキュメント)

AWS CodeCommit イベントからカスタムアクションを実行する

作成者:Abdullahi Olaoye (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

AWS CodeCommit リポジトリを使用してコードを保存する場合、リポジトリを監視し、特定のイベントが発生したときに、アクションのワークフローを開始したい場合があります。たとえば、ユーザーがコミット内のコード行にコメントしたときに、メール通知を送信したり、コミット後にリポジトリコンテンツのセキュリティスキャンを実行する AWS Lambda 関数を開始したりできます。このパターンは、カスタムアクション用の CodeCommit リポジトリを設定する手順の概要を示しています。このパターンは、AWS CodeCommit の通知ルールを使用して対象のイベントをキャプチャし、これらのイベントを設定済みのターゲットに送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Git コマンドに精通します。
- AWS CodeCommit のセットアップ 手順については、AWS CodeCommit のセットアップ を参照してください。
- AWS コマンドラインインターフェイス (AWS CLI)、インストールおよび設定 手順については、AWS DataSync の使用開始 を参照してください。

アーキテクチャ

ツール

AWS サービス

- AWS CodeCommit — CodeCommit は、安全な Git ベースのリポジトリをホストする完全マネージド型のソース管理サービスです。これにより、安全で拡張性の高いエコシステムで、チームによ

るコードの共同作業が簡単になります。CodeCommit は、独自のソース管理システムを運用したり、そのインフラストラクチャをスケールしたりする必要がなくなります。

- Amazon Simple Notification Service (Amazon SNS) は、アプリケーション、エンドユーザー、およびデバイスでクラウドから通知を瞬時に送受信できるようにするウェブサービスです。Amazon SNS は、高スループットのプッシュベースの多対多メッセージング用のトピック (通信チャネル) を提供します。Amazon SNS トピックを使用すると、パブリッシャーはメッセージを多数のサブスクライバーに配信して、Amazon Simple Queue Service (Amazon SQS) キュー、AWS Lambda 関数、HTTP/S ウェブフックなどのparallel 処理を行うことができます。Amazon SNS を使用して、モバイルプッシュ、SMS、E メールを使用してエンドユーザーに通知を送信することもできます。

エピック

CodeCommit リポジトリ設定する

タスク	説明	必要なスキル
CodeCommit リポジトリを作成します。	AWS CLI または CodeCommit コンソールを使用して CodeCommit リポジトリを作成します。手順については、CodeCommit リポジトリの作成 を参照してください。	DevOps エンジニア
コンテンツを CodeCommit にプッシュします。	リポジトリを作成したら、Git コマンドを使用してリポジトリにコンテンツを追加します。既存の Git リポジトリのコンテンツ、またはローカルのバージョン管理されていないコンテンツをコンピューターから移行できます。手順については、リポジトリへのファイルの追加 または AWS CodeCommit への移行 を参照してください。	DevOps エンジニア

Amazon SNS をセットアップする

タスク	説明	必要なスキル
SNS トピックを作成します。	この SNS トピックは CodeCommit からイベントを受け取ります。手順については、Amazon SNS トピックの作成 を参照してください。	クラウドアーキテクト、DevOps エンジニア
カスタムアクション用のリソースを作成します。	カスタムアクションを実行するには、対応するリソースを作成する必要があります。たとえば、カスタムアクションが Lambda コードを実行して SQS キューにメッセージを送信することである場合、Lambda 関数と SQS キューを作成する必要があります。E メールや SMS 通知などのアクションには、リソースは必要ありません。詳細については、作成するリソースのタイプに関する AWS ドキュメント を参照してください。	クラウドアーキテクト、DevOps エンジニア
カスタムアクションリソースを SNS トピックにサブスクライブします。	カスタムアクションに応じて、適切なプロトコルのサブスクリプションを作成します。たとえば、E メール通知用の E メールアドレス、カスタムコードを実行する Lambda 関数、Amazon SQS にイベントを送信するための SQS キューを登録します。E メールや SMS などのサブス	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	クリプションプロトコルの場合は、それぞれ E メールまたは電話番号に送信されたリンクから、サブスクリプションを確認する必要があります。手順については、Amazon SNS トピックへサブスクライブする を参照してください。	

通知ルールの設定

タスク	説明	必要なスキル
CodeCommit リポジトリの通知ルールを作成します。	通知ルールを作成するときは、通知を開始する Git イベントを選択し、ターゲットタイプとして SNS トピックを選択してから、先に作成した SNS トピックを選択します。複数のデータリポジトリへのリンクを設定することもできます。手順については、通知ルールの作成 を参照してください。	DevOps エンジニア
カスタムアクションをテストします。	通知を開始するように設定されたイベントのいずれかを実行します。たとえば、プルリクエストをトリガーとして選択した場合は、プルリクエストを作成します。カスタムアクションが実行されているはずです。たとえば、E メールで使用量の通知を受け取る場合は、トピックに E メールア	DevOps エンジニア

タスク	説明	必要なスキル
	ドレスをサブスクライブします。	

関連リソース

- AWS CodeCommit ドキュメント
- Amazon SNS のドキュメント
- Git ドキュメンテーション

GitHub Actions を使用して AWS CloudFormation テンプレートに基づいて AWS Service Catalog 製品をプロビジョニングする

作成者: Ashish Bhatt (AWS) と R"ka Modi (AWS)

概要

このパターンは、AWS Service Catalog製品とポートフォリオを使用して標準化され、チーム AWS のサービス 間で準拠したプロビジョニングを行う効率的なアプローチを組織に提供します。 は、基本ネットワークインフラストラクチャをプロビジョニングするために、Service Catalog 製品とポートフォリオに不可欠なコンポーネントを組み合わせるAWS CloudFormationのに役立ちます AWS クラウド。このパターンは、GitHub Actions を使用して、Infrastructure as Code (IaC) を自動開発ワークフローに統合することで、DevOps プラクティスも促進します。

AWS Service Catalog を使用すると、組織は で承認された IT サービスを作成および管理でき、標準化 AWS、一元管理、セルフサービスプロビジョニング、コスト管理などのメリットが得られます。GitHub Actions を通じて Service Catalog ポートフォリオと製品のデプロイを自動化することで、企業は以下を実行できます。

- 一貫性のある反復可能なデプロイを実現します。
- IaC のバージョン管理を使用します。
- クラウドリソース管理を既存の開発ワークフローと統合します。

この組み合わせにより、手動エラーを減らし、全体的な効率を向上させながら、クラウド運用を合理化し、コンプライアンスを強化し、承認された サービスの配信を加速できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- GitHub リポジトリへのアクセス
- AWS CloudFormation と の基本的な理解 AWS Service Catalog
- CloudFormation テンプレートをホストするための Amazon Simple Storage Service (Amazon S3) バケット
- GitHub と 間の接続に使用される という名前`github-actions`の AWS Identity and Access Management (IAM) ロール AWS

機能制限

- このパターンの再利用可能なコードは、GitHub Actions でのみテストされています。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

このパターンのソリューションは、次の GitHub Marketplace アクションとそれぞれのバージョンを使用して作成されました。

- `actions/checkout@v4`
- `aws-actions/configure-aws-credentials@v2`
- `aws-actions/aws-cloudformation-github-deploy@v1.2.0`

アーキテクチャ

次の図は、このソリューションのアーキテクチャを示しています。

1. 管理者またはプラットフォームエンジニアは、標準化された CloudFormation テンプレートを GitHub リポジトリにプッシュし、そこでテンプレートが維持されます。GitHub リポジトリには、GitHub Actions AWS Service Catalog を使用した のプロビジョニングを自動化するワークフローも含まれています。
2. GitHub Actions は、OpenID Connect (OIDC) プロバイダー AWS クラウド を使用して に接続し、Service Catalog をプロビジョニングするワークフローをトリガーします。
3. Service Catalog には、開発者が標準化された AWS リソースのプロビジョニングに直接使用できるポートフォリオと製品が含まれています。このパターンは、仮想プライベートクラウド (VPCs)、サブネット、NAT およびインターネットゲートウェイ、ルートテーブルなどの AWS リソースをバンドルします。
4. デベロッパーが Service Catalog 製品を作成すると、Service Catalog はそれを事前設定済みの標準化された AWS リソースに変換します。その結果、開発者は個々のリソースをプロビジョニングして手動で設定する必要がなくなるため、時間を節約できます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。これは、 製品タイプの 1 つとして簡単に使用できる Infrastructure as Code (IaC) サービスです AWS Service Catalog。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Service Catalog は、承認された IT サービスのカタログを一元管理するために役立ちます AWS。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他

- GitHub Actions は、GitHub リポジトリと緊密に統合された継続的インテグレーションおよび継続的デリバリー (CI/CD) プラットフォームです。GitHub Actions を使用して、ビルド、テスト、デプロイパイプラインを自動化できます。

コードリポジトリ

このパターンのコードは、GitHub service-catalog-with-github-actions リポジトリで入手できます。リポジトリには、以下の目的のファイルが含まれています。

- `github/workflows`:
  - `e2e-test.yaml` – このファイルは`workflow.yaml`、再利用可能なワークフローである を呼び出します。このワークフローは、ブランチにコミットとプッシュがあるとすぐにトリガーされます。
  - `workflow.yaml` – このファイルには、このソリューションの再利用可能なワークフローが含まれており、トリガー`workflow_call`として が設定されています。再利用可能なワークフローとして、 は他のワークフローから呼び出す`workflow.yaml`ことができます。
- `templates`:

- `servicecatalog-portfolio.yaml` – この CloudFormation テンプレートには、Service Catalog ポートフォリオと Service Catalog 製品をプロビジョニングするリソースが含まれています。テンプレートには、Service Catalog ポートフォリオと製品のプロビジョニング中に使用される一連のパラメータが含まれています。1 つのパラメータは、テンプレートが`vpc.yaml`アップロードされる Amazon S3 ファイル URL を受け入れます。このパターンには、 AWS リソースをプロビジョニングする `vpc.yaml` ファイルが含まれていますが、設定にパラメータ S3 ファイル URL を使用することもできます。
- `vpc.yaml` – この CloudFormation テンプレートには、Service Catalog product に追加される AWS リソースが含まれています。 AWS リソースには、VPCs、サブネット、インターネットゲートウェイ、NAT ゲートウェイ、ルートテーブルが含まれます。`vpc.yaml` テンプレートは、Service Catalog 製品およびポートフォリオテンプレートで CloudFormation テンプレートを使用する方法の例です。

ベストプラクティス

- AWS Service Catalog ドキュメントの「 のセキュリティのベストプラクティス AWS Service Catalog」を参照してください。
- GitHub ドキュメントの「GitHub Actions のセキュリティ強化」を参照してください。 GitHub

エピック

ローカルワークステーションのセットアップ

タスク	説明	必要なスキル
ローカルワークステーションに Git をセットアップします。	ローカルワークステーションに Git をインストールして設定するには、Git ドキュメントの「開始方法 – Git のインストール」の手順を使用します。	アプリ開発者
GitHub プロジェクトリポジトリのクローンを作成します。	GitHub プロジェクトリポジトリのクローンを作成するには、次の手順を実行します。	DevOps エンジニア

タスク	説明	必要なスキル

	1. このパターンの GitHub リポジトリを開く
	2. コードを選択してクローンオプションを表示し、HTTPS タブで提供されている URL をコピーします。
	3. ワークステーションにプロジェクトのフォルダを作成します。
	4. ターミナルを開き、このフォルダに移動します。
	5. GitHub リポジトリのクローンを作成するには、ステップ 2 でコピーした URL を使用して次のコマンドを実行します。
	`git clone https://github.com/aws-samples/service-catalog-with-github-actions.git`
	6. クローン作成が完了したら、プロジェクトフォルダのクローンリポジトリに変更するには、次のコマンドを実行します。
	`cd <folder-name>/service-catalog-with-github-actions`	

タスク	説明	必要なスキル
	7. 選択した統合開発環境 (IDE) でプロジェクトを開きます。	

OIDC プロバイダーを設定する

タスク	説明	必要なスキル
OIDC プロバイダーを設定します。	AWS 認証情報を存続期間の長い GitHub シークレットとして保存することなく AWS、GitHub Actions ワークフローが のリソースにアクセスできるようにする OpenID Connect (OIDC) GitHub プロバイダーを作成します。手順については、GitHub ドキュメントの「Amazon Web Services での OpenID Connect の設定」を参照してください。 OIDC プロバイダーが設定されると、前提条件で前`github-actions` 述した IAM ロール の信頼ポリシーが更新されます。	AWS 管理者、AWS DevOps、AWS 全般

GitHub Actions パイプラインをトリガーして Service Catalog ポートフォリオと製品をデプロイする

タスク	説明	必要なスキル
`e2e-test.yaml` を更新する。	`e2e-test.yaml` ファイルは、 で再利用可能なワー	DevOps エンジニア

タスク	説明	必要なスキル

	クフローをトリガーしますworkflow.yaml 。で次の入力パラメータの値を更新して検証しますe2e-test.yaml 。
	• aws_account_id – 正しい を指定します AWS アカウント。
	• aws_region – 正しい を指定します AWS リージョン。
	• s3BucketName – CloudFormation テンプレートを保持する Amazon S3 バケットを指定します。
	• ワークフローファイルには、入力として 2 つの IAM ロールが必要です。
	  • LaunchConstraintRole - エンドユーザーが製品を起動、更新、または終了するときに が AWS Service Catalog 引き受ける IAM ロール。
	  • PrincipalArn - Service Catalog ポートフォリオに関連付けられるプリンシパル (IAM ユーザー、ロール、またはグループ) の Amazon リソースネーム (ARN)。PrincipalType が の場合IAM、	

タスク	説明	必要なスキル
	サポートされている値は完全に定義された IAM Amazon リソースネーム (ARN) です。`PrincipalType` が の場合`IAM_PATTERN` 、サポートされている値は、次の形式の のない IAM ARN `AccountID` です。`arn:partition:iam::resource-type/resource-id`	

デプロイの検証

タスク	説明	必要なスキル
Service Catalog リソースを検証します。	Service Catalog リソースを検証するには、以下を実行します。 1. AWS Management Console の にサインインし AWS アカウント、 AWS リージョン が正しいことを確認します。 2. に移動AWS Service Catalogし、ポートフォリオが管理ポートフォリオに存在することを確認します。 3. ポートフォリオを選択し、製品、制約、アクセ	AWS DevOps

タスク	説明	必要なスキル
	スタブの情報を検証します。	

リソースをクリーンアップする

タスク	説明	必要なスキル
CloudFormation スタックを削除します。	CloudFormation スタックを削除するには、次の手順を実行します。 1. AWS CloudFormation コンソールを https://console.aws.amazon.com/cloudformation://www.com で開きます。 2. 画面上部のナビゲーションバーで、スタック AWS リージョン がある を選択します。 3. [スタック] ページで、削除するスタックを選択します。スタックは現在実行中である必要があります。 4. [スタックの詳細] ペインで、[削除] を選択します。 5. プロンプトが表示されたら、[スタックの削除] を選択します。 詳細については、CloudFormation ドキュメントの「CloudFormation コンソー	DevOps エンジニア、AWS 管理者

タスク	説明	必要なスキル
	ルからスタックを削除するCloudFormation 」を参照してください。	

トラブルシューティング

問題	ソリューション

`e2e-test`
 	
`Can't find 'action.yml', 'action.yaml' or 'Dockerfile' under '*/home/runner/work/service-catalog-with-github-actions/service-catalog-with-github-actions`
 	
`Did you forget to run actions/checkout before running your local action?`
	正しいリポジトリ設定が有効になっていることを確認するには、次の手順を実行します。
	
	1. Github リポジトリの設定タブに移動します。
	2. 左側のメニューから Actions, General を選択します。
	3. アクセスセクションに移動し、「XXX」組織のリポジトリからアクセス可能なオプションを選択します。

関連リソース

AWS ドキュメント

- Service Catalog の概要

その他のリソース

- ワークフローをトリガーするイベントについて (GitHub ドキュメント)
- ワークフローの再利用 (GitHub ドキュメント)

追加情報

エピックに関連するスクリーンショットを表示するには、このパターンの GitHub リポジトリの Images フォルダに移動します。次のスクリーンショットを使用できます。

- AWS Service Catalog ポートフォリオ、管理セクション
- AWS Service Catalog product、Administration セクション
- AWS Service Catalog product、ユーザー/プロビジョニングセクション

ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする

作成者: Benjamin Morris (AWS)、Aman Kaur Gandhi (AWS)、Chad Moon (AWS)、Nima Fotouhi (AWS)

概要

パイプラインのスコープ AWS Identity and Access Management 超過 (IAM) ロールのアクセス許可は、組織に不要なリスクをもたらす可能性があります。開発者は、開発中に広範なアクセス許可を付与するが、コードのトラブルシューティング後にアクセス許可の範囲を絞り込むことは無視することがあります。これにより、強力なロールがビジネスニーズなしに存在し、セキュリティエンジニアによってレビューされたことがない可能性があるという問題が発生します。

このパターンは、ロール自動販売機 (RVM) というこの問題の解決策を提供します。RVM は、安全で一元化されたデプロイモデルを使用して、開発者の労力を最小限に抑えながら、個々の GitHub リポジトリのパイプラインに最小特権の IAM ロールをプロビジョニングする方法を示します。RVM は中央ソリューションであるため、変更を承認するために必要なレビューワーとしてセキュリティチームを設定できます。このアプローチにより、セキュリティは過剰に許可されたパイプラインロールリクエストを拒否できます。

RVM は Terraform コードを入力として受け取り、パイプライン対応 IAM ロールを出力として生成します。必要な入力は、 AWS アカウント ID、GitHub リポジトリ名、およびアクセス許可ポリシーです。RVM はこれらの入力を使用して、ロールの信頼ポリシーとアクセス許可ポリシーを作成します。結果の信頼ポリシーにより、指定された GitHub リポジトリがロールを引き受け、パイプラインオペレーションに使用できます。

RVM は IAM ロール (ブートストラップ中に設定) を使用します。このロールには、組織内の各アカウントでrole-provisioning-role受けるアクセス許可があります。ロールは、 AWS Control Tower Account Factory for Terraform (AFT) または AWS CloudFormation StackSets を介して設定されます。role-provisioning-roles は、開発者向けのパイプラインロールを実際に作成するロールです。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- GitHub Actions を通じてコードとしてのインフラストラクチャ (IaC) をデプロイするために使用される GitHub 組織。(GitHub Enterprise/Premium/Ultimate は必要ありません) 。

- マルチアカウント AWS 環境。この環境は の一部である必要はありません AWS Organizations。
- all AWS アカウント (AFT や CloudFormation StackSets など) に IAM ロールをデプロイするメカニズム。
- Terraform バージョン 1.3 以降がインストールされ、設定されています。
- Terraform AWS Provider バージョン 4 以降がインストールされ、設定されています。

制約事項

- このパターンのコードは GitHub Actions と Terraform に固有です。ただし、パターンの一般的な概念は、他の継続的インテグレーションおよびデリバリー (CI/CD) フレームワークで再利用できます。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、このパターンのワークフローを図解したものです。

ロール自動販売機の一般的な使用のワークフローは、次のステップで構成されます。

1. 開発者は、新しくリクエストされた IAM ロールの Terraform コードを含むコードを RVM GitHub リポジトリにプッシュします。このアクションは、RVM GitHub Actions パイプラインをトリガーします。
2. パイプラインは、OpenID Connect (OIDC) 信頼ポリシーを使用して RVM ロール引き受けロールを引き受けます。
3. RVM パイプラインが実行されると、開発者の新しい IAM ロールをプロビジョニングするアカウントの RVM ワークフローロールを引き受けます。(RVM ワークフローロールは、AFT または CloudFormation StackSets.)
4. RVM は、適切なアクセス許可と信頼を持つ開発者の IAM ロールを作成し、そのロールを他のアプリケーションパイプラインで引き受けることができます。
5. アプリ開発者は、この RVM でプロビジョニングされたロールを引き受けるようにアプリパイプラインを設定できます。

作成されたロールには、開発者がリクエストしたアクセス許可とReadOnlyAccessポリシーが含まれます。ロールは、開発者が指定したリポジトリのmainブランチに対して実行されるパイプラインでのみ引き受けることができます。このアプローチは、ブランチの保護とレビューがロールを使用するために必要なことを確認するのに役立ちます。

自動化とスケール

最小特権のアクセス許可では、プロビジョニングされる各ロールの詳細に注意する必要があります。このモデルにより、これらのロールの作成に必要な複雑さが軽減され、開発者は追加の学習や労力を必要とせずに必要なロールを作成できます。

ツール

AWS のサービス

- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用を認可するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。

その他のツール

- Git はオープンソースの分散バージョン管理システムです。これには、組織アカウントを作成する機能が含まれます。
- GitHub Actions は、GitHub リポジトリと緊密に統合された継続的インテグレーションおよび継続的デリバリー (CI/CD) プラットフォームです。GitHub Actions を使用して、ビルド、テスト、デプロイパイプラインを自動化できます。
- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンのコードは、GitHub role-vending-machine リポジトリで入手できます。

ベストプラクティス

- 正しい方法を簡単にし、間違った方法を簡単にする – 正しいことを容易にします。デベロッパーが RVM プロビジョニングプロセスに苦労している場合、他の方法でロールを作成しようとする可

能性があります。これにより、RVM の中心的な性質が損なわれます。セキュリティチームが RVM を安全かつ効果的に使用する方法について明確なガイダンスを提供していることを確認します。

また、デベロッパーが間違ったことを行うのを難しくする必要があります。サービスコントロールポリシー (SCPs) またはアクセス許可の境界を使用して、他のロールを作成できるロールを制限します。このアプローチは、ロールの作成を RVM やその他の信頼できるソースのみに制限するのに役立ちます。

- 良い例を挙げる – 必然的に、一部の開発者は新しいロールにアクセス許可を付与するための非公式テンプレートとして RVM リポジトリ内の既存のロールを適応させます。コピーできる最小アクセス許可の例がある場合、開発者が広範でワイルドカードの多いアクセス許可をリクエストするリスクを軽減できます。多数のワイルドカードを持つアクセス許可の高いロールから始めると、その問題は時間の経過とともに増える可能性があります。
- 命名規則と条件を使用する – 開発者がアプリケーションが作成するすべてのリソース名を知らなくても、命名規則を使用してロールのアクセス許可を制限する必要があります。例えば、Amazon S3 バケットを作成する場合、リソースキーの値は のようになります`arn:aws:s3:::myorg-myapp-dev-*`。これにより、ロールには、その名前に一致するバケット以外のアクセス許可がない可能性があります。IAM ポリシーを使用して命名規則を適用すると、命名規則への準拠を向上させるという追加の利点があります。この改善は、一致しないリソースの作成が許可されないために発生します。
- プルリクエスト (PR) レビューを要求する – RVM ソリューションの価値は、新しいパイプラインロールをレビューできる一元的な場所を作成することです。ただし、この設計は、安全で高品質のコードが RVM にコミットされることを保証するガードレールがある場合にのみ役立ちます。コード (など`main`) をデプロイするために使用されるブランチをダイレクトプッシュから保護し、それらをターゲットとするマージリクエストの承認が必要です。
- 読み取り専用ロールの設定 – デフォルトでは、RVM はリクエストされた各ロール`readonly`のバージョンをプロビジョニングします。このロールは、パイプラインワークフローなど、データを書き込まない CI/CD `terraform plan`パイプラインで使用できます。このアプローチは、読み取り専用ワークフローの動作が間違っている場合に不要な変更を防ぐのに役立ちます。

  デフォルトでは、 AWS 管理`ReadOnlyAccess`ポリシーは読み取り専用ロールと読み取り/書き込みロールの両方にアタッチされます。このポリシーは、必要なアクセス許可を決定する際の反復の必要性を減らしますが、一部の組織では過度に許容されている場合があります。必要に応じて、Terraform コードからポリシーを削除できます。
- 最小アクセス許可の付与 – 最小権限の原則に従い、タスクの実行に必要な最小アクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

環境を準備する

タスク	説明	必要なスキル

サンプルリポジトリを GitHub 組織にコピーします。	このパターンのリポジトリをクローンするか、このリポジトリを GitHub 組織にフォークして、ニーズに合わせて調整できるようにします。
	• このリポジトリのクローンを作成する場合は、次のコマンドを使用できます。
	`git clone https://github.com/aws-samples/role-vending-machine`
	• リポジトリを GitHub 組織にフォークすることを選択した場合は、次のコマンドを使用できます。
	`gh repo fork aws-samples/role-vending-machine --org YOUR_ORGANIZATION_NAME`
		DevOps エンジニア

RVM AWS アカウント の を決定します。	RVM AWS アカウント に使用するインフラストラクチャのデプロイを決定します。管理アカウントまたはルートアカウントを使用しないでください。	クラウドアーキテクト

タスク	説明	必要なスキル
(オプション) 組織のパイプラインに PRsの作成を許可します。	**Note** このステップは、`generate_providers_and_account_vars` ワークフローに PRsの作成を許可する場合にのみ必要です。 組織のパイプラインが PRsを作成できるようにするには、次の手順を実行します。 1. `https://github.com/organizations/YOUR_ORG/settings/actions` に移動します。 2. GitHub Actions にプルリクエストの作成と承認を許可するを選択します。 詳細については、GitHub ドキュメントのリポジトリの GitHub Actions 設定の管理を参照してください。 GitHub	DevOps エンジニア

タスク	説明	必要なスキル
RVM アカウントに読み取り専用アクセス許可を付与します。	RVM アカウントに読み取り専用アクセス許可を付与する委任ポリシーを管理アカウントに作成します。これにより、RVM GitHub ワークフローは、`generate_providers_and_account_vars.py` スクリプトの実行時に AWS 組織のアカウントのリストを動的にプルできます。 次のコードを使用して、 をステップ 2 で選択した AWS アカウント ID <YOUR RVM Account ID>に置き換えます。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "Statement", "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam::<YOUR RVM Account ID>:root" }, "Action": [ "organizations:ListAccounts", "organizations:DescribeOrganization",`	クラウド管理者

タスク	説明	必要なスキル
	"organizations:DescribeOrganizationalUnit", "organizations:ListRoots", "organizations:ListAWSServiceAccessForOrganization", "organizations:ListDelegatedAdministrators"], "Resource": "*" }] }	
サンプルリポジトリからデフォルト値を更新します。	特定の環境で動作するように RVM を設定するには AWS リージョン、以下を実行します。 1. `scripts/generate_providers_and_account_vars.py` およびその他のファイル (`bootstrap` フォルダなど) を、運用している適切なリージョンで更新します。 2. AWS アカウント ID、リージョン、および指定するその他の変数を使用して `.github/workflows/.env` ファイルを更新します。	DevOps エンジニア

インフラストラクチャの初期化

タスク	説明	必要なスキル
RVM リポジトリをブートストラップします。	このステップは、RVM パイプライン自体で使用される OIDC 信頼ロールと IAM ロールを作成して、他のロールの運用と販売を開始できるようにするために必要です。 RVM アカウントのコンテキストで、 `scripts/bootstrap` ディレクトリから`terraform apply`コマンドを手動で実行します。変数ドキュメントに基づいて、必要な値を指定します。	DevOps エンジニア

オペレーションの設定

タスク	説明	必要なスキル
`github-workflow-rvm` および `github-workflow-rvm-readonly` ロールをすべてのアカウントにデプロイします。	AFT や StackSets など、組織のプラクティスに沿ったデプロイ方法を選択します。このメソッドを使用して、`scripts/assumed_role/main.tf` ファイル内の 2 つの IAM ロール (デフォルト名 `github-workflow-rvm` と `github-workflow-rvm-readonly`) を、RVM がパイプラインロールを作成できるようにする各アカウントにデプロイします。	AWS 管理者

タスク	説明	必要なスキル
	これらの IAM ロールには、RVM アカウントのロール引き受けロール (または`readonly`同等のロール) が引き受けることを許可する信頼ポリシーがあります。ロールには、 に一致するロールの読み取りと書き込み (`readonly`ロールを使用する場合を除く) を許可する IAM アクセス許可ポリシーもあります`github-workflow-role-*` 。	

タスク	説明	必要なスキル
`generate_providers_and_account_vars` ワークフローを実行します。	パイプラインロールを作成する準備が整うように RVM を設定するには、次の手順を実行します。 1. `generate_providers_and_account_vars` workflow_dispatch を使用してワークフローを実行します。 2. ワークフローが作成する PR をマージします。 ワークフローが完了すると、RVM は次の準備が整います。 • 新しいパイプラインロールのリクエストを受け入れます。 • 必要に応じて、読み取り専用ロールまたは読み取り/書き込みロールを作成します。 • 指定された にロールをデプロイします AWS アカウント。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
RVM を使用してロールを作成しましたが、GitHub はロールを引き受けることができません。	GitHub リポジトリの名前が`github_workflow_roles` モジュールに提供された名前と一致していることを確認します。ロールは、1 つのリポジトリのみが引き受けることができるようにスコープされます。 同様に、GitHub パイプラインで使用されるブランチが、`github_workflow_roles` モジュールに提供されたブランチの名前と一致することを確認します。通常、書き込みアクセス許可を持つ RVM 作成ロールは、`main`ブランチ (つまり、 から取得されたデプロイ) を対象とするワークフローでのみ使用できます`main`。
特定のリソースを読み取るアクセス許可がないため、読み取り専用ロールはパイプラインの実行に失敗しています。	`ReadOnlyAccess` ポリシーには広範な読み取り専用アクセス許可がありますが、一部の読み取りアクション (特定の AWS Security Hub アクションなど) はありません。 `github-workflow-roles` モジュールの `inline_policy_readonly` パラメータを使用して、特定のアクションのアクセス許可を追加できます。

関連リソース

- Stack AWS CloudFormation StackSets を使用するためのベストプラクティス
- 複数のアカウントを使用した AWS 環境の整理
- AWS Control Tower Account Factory for Terraform (AFT) の概要
- ポリシーのベストプラクティス

追加情報

GitHub 環境の使用

GitHub 環境は、ロールアクセスのブランチベースの制限に代わるアプローチです。GitHub 環境を使用する場合は、IAM 信頼ポリシーの追加条件の構文の例を次に示します。この構文は、GitHub アクションがProduction環境で実行されている場合にのみロールを使用できることを指定します。

```
"StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:environment:Production"
}
```

構文例では、次のプレースホルダー値を使用します。

- octo-org は GitHub 組織名です。
- octo-repo はリポジトリ名です。
- Production は、特定の GitHub 環境名です。

Amazon CloudWatch メトリクスを CSV ファイルに公開

作成者:Abdullahi Olaoye (AWS)

概要

このパターンでは、Python スクリプトを使用して Amazon CloudWatch メトリクスを取得し、メトリクス情報をカンマ区切り値 (CSV) ファイルに変換して読みやすくします。このスクリプトは、メトリックスを取得する必要がある AWS サービスを必須の引数として取ります。AWS リージョンと AWS 認証情報プロファイルをオプションの引数として指定できます。これらの引数を指定しない場合、スクリプトはスクリプトが実行されるワークステーションに設定されているデフォルトのリージョンとプロファイルを使用します。スクリプトが実行されると、CSV ファイルが生成され、同じディレクトリに保存されます。

このパターンで提供されるスクリプトと関連ファイルについては、添付ファイルセクションを参照してください。

前提条件と制限

前提条件

- Python 3.x
- AWS コマンドラインインターフェイス (AWS CLI)

制限

このスクリプトでは、現在以下の AWS サービスをサポートします。

- AWS Lambda
- Amazon Elastic Compute Cloud (Amazon EC2)
  - デフォルトでは、スクリプトは Amazon Elastic Block Store (Amazon EBS) ボリュームメトリクスを収集しません。Amazon EBS メトリクスを収集するには、添付 `metrics.yaml` ファイルを変更する必要があります。
- Amazon Relational Database Service (Amazon RDS)
  - ただし、このスクリプトは Amazon Aurora をサポートしていません。
- Application Load Balancer
- Network Load Balancer
- Amazon API Gateway

ツール

- Amazon CloudWatch は、DevOps エンジニア、開発者、サイト信頼性エンジニア (SRE)、ITマネージャー向けに構築された、モニタリングサービスです。CloudWatch では、データと実用的なインサイトを提供して、アプリケーションのモニタリング、システム全体のパフォーマンスの変化に関する対応、リソース使用率の最適化、および運用状態の統合的な確認を行うことができます。CloudWatch は、ログ、メトリックス、イベントの形式でモニタリングデータと運用データを収集し、AWS とオンプレミスサーバーで実行される AWS のリソース、アプリケーション、サービスを一元的に表示します。

エピック

前提条件をインストールして設定する

タスク	説明	必要なスキル
前提条件をインストールします。	次のコマンドを実行してください。 `$ pip3 install -r requirements.txt`	開発者
AWS CLI を設定します。	次のコマンドを実行してください。 `$ aws configure`	開発者

Python スクリプトを設定します

タスク	説明	必要なスキル
スクリプトを開きます。	スクリプトのデフォルト設定を変更するには、`metrics.yaml` を開きます。	開発者

タスク	説明	必要なスキル
スクリプトの期間を設定します。	これは取得する期間です。デフォルト期間は 5 分 (300 秒) です。期間を変更できますが、以下の制限に注意します。 • 指定する時間値が 3 時間から 15 日前までの場合は、60 秒 (1 分) の倍数を使用します。 • 指定する時間値が 15 時間から 63 日前までの場合、期間に 300 秒 (5 分) `の倍数を使用します。 • 指定する時間の値が 63 日前よりも大きい場合、期間に 3,600 秒 (1 時間) の倍数を使用します。 そうしないと、API オペレーションはデータポイントを返しません。	開発者
スクリプトの時間を設定します。	この値には、取得するメトリックスの時間数を指定します。デフォルトは 1 時間です。複数日分のメトリクスを取得するには、値を時間単位で指定します。たとえば、2 日間の場合は 48 と指定します。	開発者

タスク	説明	必要なスキル
スクリプトの統計値を変更します。	(オプション) グローバル統計値は Average で、特定の統計値が割り当てられていないメトリクスを取得するときに使用されます。このスクリプトは統計値 Maximum、SampleCount 、Sum をサポートします。	開発者

Python スクリプトを実行する。

タスク	説明	必要なスキル

スクリプトを実行します。	以下のコマンドを使用します。
	`$ python3 cwreport.py <service>`
	サービス値、オプション region 、profile パラメータのリストを表示するには、以下のコマンドを実行します。
	`$ python3 cwreport.py -h`
	オプションパラメータの詳細については、追加情報セクションを参照してください。
		開発者

関連リソース

- 「AWS CLI の設定」
- Amazon CloudWatch メトリクスを使用する
- 「Amazon CloudWatch ドキュメント」
- 「EC2: CloudWatch - メトリクス」
- AWS ラムダメトリックス
- 「Amazon RDS メトリクス」
- 「Application Load Balancer のメトリクス」
- 「Network Load Balancer のメトリクス」
- 「Amazon API Gateway のメトリクス」

追加情報

スクリプトの使用法

```
$ python3 cwreport.py -h
```

シンタックス例

```
python3 cwreport.py <service> <--region=Optional Region> <--profile=Optional credential
 profile>
```

パラメータ

- service (必須) – スクリプトを実行したいサービス。スクリプトは現在、AWS Lambda、Amazon EC2、Amazon RDS、Application Load Balancer、Network Load Balancer、API ゲートウェイといったサービスをサポートします。
- region (オプション) – メトリックスを取得する AWS リージョン。デフォルトのリージョンは `ap-southeast-1` です。
- プロファイル (オプション) – 使用する AWS CLI の名前付きプロファイル。このパラメータを指定しない場合、デフォルトで設定されている認証情報プロファイルが使用されます。

例

- デフォルトのリージョンと ap-southeast-1 デフォルトで設定された認証情報を使用して Amazon EC2 メトリックスを取得するには:`$ python3 cwreport.py ec2`
- リージョンを指定して API ゲートウェイメトリクスを取得するには:`$ python3 cwreport.py apigateway --region us-east-1`
- AWS プロファイルを指定して Amazon EC2 メトリックスを取得するには: `$ python3 cwreport.py ec2 --profile testprofile`
- リージョンとプロファイルの両方を指定して、 Amazon EC2 メトリックスを取得するには: `$ python3 cwreport.py ec2 --region us-east-1 --profile testprofile`

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Lambda オートメーションを使用して 全体の Amazon EC2 エントリを AWS アカウント から削除 AWS Managed Microsoft AD する

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Tamilselvan P (AWS)

概要

Active Directory (AD) は、ドメイン情報とネットワークサービスとのユーザーインタラクションを管理する Microsoft スクリプトツールです。これは、従業員の認証情報とアクセス許可を管理するために、マネージドサービスプロバイダー (MSPsの間で広く使用されています。AD 攻撃者は非アクティブなアカウントを使用して組織をハッキングできるため、非アクティブなアカウントを見つけて定期的なメンテナンススケジュールで無効にすることが重要です。を使用すると AWS Directory Service for Microsoft Active Directory、Microsoft Active Directory をマネージドサービスとして実行できます。このパターンは、非アクティブなアカウントをすばやく見つけて削除するように AWS Lambda 自動化を設定するのに役立ちます。

次のシナリオが組織に適用される場合、このパターンが役立ちます。

- 一元化された AD 管理 – 組織に複数の があり AWS アカウント、それぞれに独自の AD デプロイがある場合、すべてのアカウントでユーザーアカウントとアクセス許可を一貫して管理するのは難しい場合があります。アカウント間の AD クリーンアップソリューションを使用すると、すべての AD インスタンスから非アクティブなアカウントを一元的に無効化または削除できます。
- AD の再構築または移行 – 組織が AD デプロイを再構築または移行する場合、アカウント間の AD クリーンアップソリューションが環境の準備に役立ちます。このソリューションは、不要なアカウントや非アクティブなアカウントを削除し、移行プロセスを簡素化し、潜在的な競合や問題を減らすのに役立ちます。

このパターンを使用すると、次の利点が得られます。

- データベースとサーバーのパフォーマンスを向上させ、非アクティブなアカウントからのセキュリティの脆弱性を修正します。
- AD サーバーがクラウドでホストされている場合、非アクティブなアカウントを削除すると、パフォーマンスを向上させながらストレージコストを削減することもできます。帯域幅料金とコンピューティングリソースの両方が低下する可能性があるため、月額請求額が減少する可能性があります。
- クリーンな Active Directory を使用して、潜在的な攻撃者を阻止します。

前提条件と制限

前提条件

- アクティブな親アカウント AWS アカウント と 1 つ以上の子アカウント。このパターンでは、親アカウントは Active Directory が作成される場所です。子アカウントは Windows サーバーをホストし、親アカウントの Active Directory を介して結合されます。
- Git をローカルワークステーションにインストールして設定します。
- Terraform がローカルワークステーションにインストールおよび設定されている。
- AWS Managed Microsoft AD 親アカウントで設定され、すべての子アカウントと共有される ディレクトリ。詳細については、 AWS Directory Service 管理ガイドの「チュートリアル: シームレスな EC2 ドメイン参加のための AWS Managed Microsoft AD ディレクトリの共有」を参照してください。
- 仮想プライベートクラウド (VPC) ピアリング接続、または AWS Directory Service （ 親アカウント) の VPC と Amazon Elastic Compute Cloud (Amazon EC2) インスタンス (子アカウント) の VPC の間で利用可能な AWS Transit Gateway 接続。詳細については、「 管理ガイド」の「ディレクトリ所有者とディレクトリコンシューマーアカウント間の VPC ピアリング接続を設定する」を参照してください。 AWS Directory Service
- すべての親アカウントと子アカウントの`EC2WindowsUserdata`スクリプトで設定された Windows マシン。スクリプトファイルは、このパターンのコードリポジトリのルートで使用できます。
- 親アカウントからの AWS Lambda 関数の使用を許可する信頼ポリシーで設定された、各子アカウントで使用可能なクロスアカウント AWS Identity and Access Management (IAM) ロール。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge AWS アカウントでの 間のイベントの送受信」を参照してください。 EventBridge
- 親アカウントの Parameter Store AWS Systems Manager で使用できる次のシークレット値。
  - `domainJoinUser` – ディレクトリサービスのユーザー名
  - `domainJoinPassword` – ディレクトリサービスのパスワード

  シークレットの詳細については、「 AWS Secrets Manager ユーザーガイド」の「 AWS Secrets Manager シークレットの作成」を参照してください。

制約事項

- 子アカウントでのリソースの作成は、Terraform では自動化されません。を使用して、次のリソースを手動で作成する必要があります AWS Management Console。

  - Amazon EC2 終了イベントを親アカウントに送信する Amazon EventBridge ルール
  - 信頼ポリシーを持つ子アカウントでの Amazon EC2 クロスアカウントロールの作成
  - VPC ピアリングまたは Transit Gateway 接続
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Terraform バージョン 1.1.9 以降
- Terraform AWS プロバイダーバージョン 3.0 以降

アーキテクチャ

次の図は、ソリューションの高レベルのアーキテクチャを示しています。

アーキテクチャ図は、次のプロセスを示しています。

1. 子アカウントでは、EventBridge ルールはすべての Amazon EC2 終了イベントを収集します。ルールは、親アカウントに存在する EventBridge にこれらのイベントを送信します。
2. 親アカウントから、EventBridge はすべてのイベントを収集し、Lambda 関数 をトリガーするルールを含めますADcleanup-Lambda。
3. 親アカウントは、親アカウントまたは子アカウントから終了イベントを受け取り、Lambda 関数をトリガーします。
4. Lambda 関数は、Python boto モジュールを使用して Amazon EC2 Auto Scaling グループを呼び出し、ランダムなインスタンス ID を取得します。インスタンス ID は、Systems Manager コマンドを実行するために使用されます。
5. Lambda 関数は、boto モジュールを使用して Amazon EC2 に別の呼び出しを行います。Lambda 関数は、実行中の Windows サーバーのプライベート IP アドレスを取得し、そのアドレスを一時変数に保存します。ステップ 5.1 および 5.2 では、実行中の Windows EC2 インスタンスが子アカウントから収集されます。
6. Lambda 関数は、Systems Manager をもう一度呼び出して、接続されているコンピュータ情報を取得します AWS Directory Service。

7. AWS Systems Manager ドキュメントは、Amazon EC2 Windows サーバーで PowerShell コマンドを実行して、AD に接続されているコンピュータのプライベート IP アドレスを取得するのに役立ちます。(Systems Manager ドキュメントは、ステップ 4 で取得したインスタンス ID を使用します)。
8. AD ドメインのユーザー名とパスワードは Parameter Store AWS Systems Manager に保存されます。 AWS Lambda Systems Manager は Parameter Store を呼び出し、AD への接続に使用するユーザー名とパスワードの値を取得します。
9. Systems Manager ドキュメントを使用して、ステップ 4 で前に取得したインスタンス ID を使用してPowerShell スクリプトが Amazon EC2 Windows サーバーで実行されます。
10. Amazon EC2 は PowerShell コマンド AWS Directory Service を使用して に接続し、使用されていないコンピュータや非アクティブのコンピュータを削除します。

ツール

AWS サービス

- AWS Directory Service は、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Relational Database Service (Amazon RDS) for SQL Server、Amazon FSx for Windows File Server AWS のサービス など、他の で Microsoft Active Directory (AD) を使用する複数の方法を提供します。
- AWS Directory Service for Microsoft Active Directory では、ディレクトリ対応のワークロードと AWS リソースが で Microsoft Active Directory を使用できます AWS クラウド。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon EventBridge は、アプリケーションをさまざまなソースからのリアルタイムデータに接続するのに役立つサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。IAM を使用すると、 のサービスとリソースにアクセスできるユーザーまたはユーザーを指定し AWS、きめ細かなアクセス許可を一元管理し、アクセスを分析してアクセス許可を絞り込むことができます AWS。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間が短縮され、 AWS リソースを大規模に安全に管理できます。
- AWS Systems Manager ドキュメントは、Systems Manager がマネージドインスタンスで実行するアクションを定義します。Systems Manager には、ランタイムでパラメータを指定して使用できる事前設定済みのドキュメントが 100 件以上含まれています。
- AWS Systems Manager Parameter Store は の機能であり、設定データ管理 AWS Systems Manager とシークレット管理のための安全な階層型ストレージを提供します。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。
- PowerShell は Windows、Linux、および macOS で動作するMicrosoft の自動化および構成管理プログラムです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub aws-lambda-ad-cleanup-terraform-samples リポジトリで入手できます。

ベストプラクティス

- ドメインを自動的に結合します。 AWS Directory Service ドメインの一部となる Windows インスタンスを起動するときは、後でインスタンスを手動で追加するのではなく、インスタンスの作成プロセス中にドメインに参加します。ドメインを自動的に結合するには、新しいインスタンスを起動するときに、ドメイン結合ディレクトリのドロップダウンリストから正しいディレクトリを選択します。詳細については、 AWS Directory Service 管理ガイドのAmazon EC2 Windows インスタンスを AWS Managed Microsoft AD Active Directory にシームレスに結合する」を参照してください。
- 未使用のアカウントを削除します。 AD で使用されたことがないアカウントを見つけるのが一般的です。システム内に残っている無効または非アクティブなアカウントと同様に、未使用のアカウントを無視すると、AD システムが遅くなったり、組織がデータ侵害に対して脆弱になる可能性があります。

- Active Directory のクリーンアップを自動化します。 セキュリティリスクを軽減し、古いアカウントが AD のパフォーマンスに影響を与えないようにするには、AD クリーンアップを定期的に実行する必要があります。スクリプトを記述することで、ほとんどの AD 管理タスクとクリーンアップタスクを実行できます。タスクの例としては、無効化されたアカウントと非アクティブなアカウントの削除、空のグループと非アクティブなグループの削除、期限切れのユーザーアカウントとパスワードの検索などがあります。

エピック

子アカウントのセットアップ

タスク	説明	必要なスキル
子アカウントにクロスアカウントロールを作成します。	子アカウントにクロスアカウントロールを作成するには、次の手順を実行します。 1. 子アカウントごとに、 という名前の管理ポリシー`ec2crossaccountrole` を使用して という名前のロールを作成します`AmazonEC2ReadOnlyAccess` 。（詳細については、IAM ドキュメントの「カスタム信頼ポリシーを使用してロールを作成する」を参照してください）。 2. カスタム信頼ポリシーセクションで、次のコードを追加します。 `{ "Version": "2012-10-17", "Statement": [ {`	DevOps エンジニア

タスク	説明	必要なスキル
	"Effect": "Allow", "Principal": { "Service": "ec2.amazonaws.com" }, "Action": "sts:AssumeRole" }, { "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam::${Parentaccountid}:role/ADcleanuprole" }, "Action": "sts:AssumeRole" }] }	

タスク	説明	必要なスキル
子アカウントにイベントルールを作成します。	子アカウントごとに EventBridge ルールを作成するには、次の手順を実行します。 1. 子にサインインし AWS アカウント、https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。 2. ナビゲーションペインで [ルール] を選択します。 3. [ルールを作成] を選択します。 4. 名前を入力し、オプションでルールの説明を入力します。 5. イベントバスで、AWS のデフォルトイベントバスを選択します。 6. ルールタイプで、イベントパターンを持つルール を選択します。 7. [次へ] を選択します。 8. イベントパターンの場合は、次のコードを貼り付けます。 `{ "source": ["aws.ec2"], "detail-type": ["EC2 Instance State-change Notification"],`	DevOps エンジニア

タスク	説明	必要なスキル

	<pre>"detail": {
	"state": ["terminated"]
	}
	}</pre>
	9. [次へ] を選択します。
	10.ターゲットタイプで、別のアカウントまたはリージョンのイベントバスを選択します。 イベントバスをターゲットとして、親アカウントのイベントバス Amazon リソースネーム (ARN) を入力します。
	11実行ロール で、この特定のリソースの新しいロールを作成する を選択します。
	12Next を選択して新しいルールの詳細を確認し、Create を選択します。
	
	詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge のイベントに対応するルールの作成EventBridge」を参照してください。	

タスク	説明	必要なスキル
EC2 インスタンスを作成し、AD に結合します。	Windows 用の EC2 インスタンスを作成するには、次の手順を実行します。 1. このパターンのコードリポジトリで使用可能なEC2WindowsUserdata スクリプトを使用します。 2. ユーザーデータスクリプトで、親アカウントDirectory service addresses の値を使用するように次のコードを修正します。 `set-DnsClientServerAddress -InterfaceIndex 6 -ServerAddresses $(Directory service addresses)`	DevOps エンジニア

ローカルワークステーションをセットアップする

タスク	説明	必要なスキル
プロジェクトフォルダを作成し、ファイルを追加します。	リポジトリのクローンを作成し、プロジェクトフォルダを作成するには、次の手順を実行します。 1. このパターンの GitHub リポジトリを開きます。	DevOps エンジニア

タスク	説明	必要なスキル

	2. コードボタンを選択すると、クローンのドロップダウンにクローンを作成するオプションが表示されます。
	3. HTTPS タブで、ウェブ URL を使用してクローンで提供された URL をコピーします。
	4. マシンにフォルダを作成し、プロジェクト名で名前を付けます。
	5. ローカルマシンでターミナルを開き、このフォルダに移動します。
	6. git リポジトリのクローンを作成するには、次のコマンドを使用します。
	`git clone <repository-URL>.git`
	7. リポジトリのクローンを作成したら、次のコマンドを使用してクローンされたディレクトリに移動します。
	`cd <directory name>/terraform-aws-lambda-ad-cleanup/multiple-account-cleanup`
	8. クローンされたリポジトリで、選択した統合開発環境 (IDE) でこのプロジェクトを開きます。	

タスク	説明	必要なスキル
adcleanup.zip ファイルを構築します。	lambda_function.py ファイルを圧縮するには、次のコマンドを実行します。 zip -r adcleanup.zip lambda_function.py	DevOps エンジニア

Terraform 設定を使用してターゲットアーキテクチャをプロビジョニングする

タスク	説明	必要なスキル
Terraform 変数の値を指定します。	子アカウントの場合、terraform.tfvars ファイルに文字列タイプとして次のarn変数の値を指定します。 • lambda_env_cross_role_arn • child_account_cross_role_arn	DevOps エンジニア
Terraform 構成を初期化します。	Terraform ファイルを含む作業ディレクトリを初期化するには、次のコマンドを実行します。 terraform init	DevOps エンジニア
変更をプレビューします。	インフラストラクチャがデプロイされる前に Terraform がインフラストラクチャに加える変更をプレビューできます。Terraform が必要に応じて変更を行うことを確認するに	DevOps エンジニア

タスク	説明	必要なスキル
	は、次のコマンドを実行します。 `terraform plan –-var-file=examples/terraform.tfvars`	
提案されたアクションを実行します。	`terraform plan` コマンドの結果が期待どおりであることを確認するには、以下を実行します。 1. 次のコマンドを実行します。`terraform apply` 2. にサインインし AWS Management Console、リソースが存在することを確認します。	DevOps エンジニア

デプロイメントを確認する

タスク	説明	必要なスキル
Lambda 関数を実行してテストします。	デプロイが正常に行われたことを確認するには、以下を実行します。 1. にサインインし AWS Management Console、Lambda コンソールを開きます。Functions ページを開き、ADcleanup-Lambda-* で始まる関数名を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	2. 関数の概要ページで、コードソースセクションのコードタブでテストを選択します。 3. テストイベントを保存するには、イベントの名前を指定し、保存を選択します。イベントをテストするには、もう一度テストを選択します。 実行結果には、関数の出力が表示されます。	

タスク	説明	必要なスキル
親アカウントからの EventBridge ルール実行の結果を表示します。	親アカウントの Amazon EC2 終了イベントに基づく EventBridge ルールの結果を表示するには、次の手順を実行します。 1. 親アカウントから EC2 インスタンスを終了します。 2. 親アカウントの Lambda コンソールを開きます。Functions ページを開き、ADcleanup-Lambda-* で始まる関数名を選択します。 3. Monitor タブを選択し、CloudWatch ログの表示を選択します。 CloudWatch コンソールのロググループページには、Lambda 関数の結果が表示されます。	DevOps エンジニア

タスク	説明	必要なスキル
子アカウントからの EventBridge ルール実行の結果を表示します。	子アカウントの Amazon EC2 終了イベントに基づく EventBridge ルールの結果を表示するには、次の手順を実行します。 1. 子アカウントから EC2 インスタンスを終了します。 2. 親アカウントの Lambda コンソールを開きます。関数ページを開き、ADcleanup-Lambda-* で始まる関数名を選択します。 3. Monitor タブを選択し、View CloudWatch logs を選択します。 CloudWatch コンソールのロググループページには、Lambda 関数の結果が表示されます。	DevOps エンジニア

使用後のインフラストラクチャのクリーンアップ

タスク	説明	必要なスキル
インフラストラクチャをクリーンアップします。	作成したインフラストラクチャをクリーンアップするには、次のコマンドを使用します。 `terraform destroy`	DevOps エンジニア

タスク	説明	必要なスキル
	`destroy` コマンドを確認するには、「」と入力しますyes。	
クリーンアップ後に を確認します。	リソースが正常に削除されたことを確認します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
AWS Directory Service （ 親アカウント) と Amazon EC2 インスタンス (子アカウント) 間の接続の問題 – VPC ピアリングが使用可能であっても、子アカウントのコンピュータを AD に参加させることはできません。	VPCs にルーティングを追加します。手順については、 AWS Directory Service ドキュメントの「ディレクトリ所有者とディレクトリコンシューマーアカウント間の VPC ピアリング接続を設定する」を参照してください。

関連リソース

AWS ドキュメント

- Amazon EventBridge と AWS Identity and Access Management
- Systems Manager に必要なインスタンスのアクセス許可を設定する
- の ID とアクセスの管理 AWS Directory Service
- Lambda のアイデンティティベースの IAM ポリシー
- Amazon EC2 Windows インスタンスを AWS Managed Microsoft AD Active Directory に手動で結合する
- AWS Lambda オートメーションを使用して同じ の Amazon EC2 エントリを AWS アカウント から削除 AWS Managed Microsoft AD する

その他のリソース

- AWS プロバイダー (Terraform ドキュメント)

- バックエンド設定 (Terraform ドキュメント)
- Terraform のインストール (Terraform ドキュメント)
- Python boto モジュール (Python Package Index リポジトリ)
- Terraform バイナリダウンロード (Terraform ドキュメント)

AWS Lambda オートメーションを使用して AWS アカウント から同じ の Amazon EC2 AWS Managed Microsoft AD エントリを削除する

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Tamilselvan P (AWS)

概要

Active Directory (AD) は、ドメイン情報とネットワークサービスとのユーザーインタラクションを管理する Microsoft スクリプトツールです。これは、従業員の認証情報とアクセス許可を管理するために、マネージドサービスプロバイダー (MSPsの間で広く使用されています。AD 攻撃者は非アクティブなアカウントを使用して組織へのハッキングを試みることができるため、非アクティブなアカウントを見つけて定期的なメンテナンススケジュールで無効にすることが重要です。を使用すると AWS Directory Service for Microsoft Active Directory、Microsoft Active Directory をマネージドサービスとして実行できます。

このパターンは、非アクティブなアカウントをすばやく検索して削除するように AWS Lambda 自動化を設定するのに役立ちます。このパターンを使用すると、次の利点が得られます。

- データベースとサーバーのパフォーマンスを向上させ、非アクティブなアカウントからのセキュリティの脆弱性を修正します。
- AD サーバーがクラウドでホストされている場合、非アクティブなアカウントを削除すると、パフォーマンスを向上させながらストレージコストを削減することもできます。帯域幅の料金とコンピューティングリソースの両方が減少するため、月額請求額が減少する可能性があります。
- 潜在的な攻撃者をクリーンな Active Directory で阻止します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Git がローカルワークステーションにインストールおよび設定されている。
- ローカルワークステーションに Terraform がインストールされ、設定されている。
- Active Directory モジュール (`ActiveDirectory`) を搭載した Windows コンピュータ。
- のディレクトリ AWS Managed Microsoft AD と、パラメータストアの AWS Systems Manager パラメータに保存されている認証情報。

- AWS Identity and Access Management 「ツール」に AWS のサービス リストされている への アクセス許可を持つ (IAM) ロール。IAM の詳細については、「関連リソース」を参照してください。

制約事項

- このパターンは、クロスアカウント設定をサポートしていません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Terraform バージョン 1.1.9 以降
- Terraform AWS プロバイダーバージョン 3.0 以降

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

この図表は、次のワークフローを示しています：

1. Amazon EventBridge は、cron 式に基づいて AWS Lambda 関数をトリガーします。（このパターンでは、cron 式スケジュールは 1 日に 1 回です）。
2. 必要な IAM ロールとポリシーが作成され、Terraform AWS Lambda を介して にアタッチされます。
3. AWS Lambda 関数が実行され、Python boto モジュールを使用して Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling Groups を呼び出します。Lambda 関数はランダムなインスタンス ID を取得します。インスタンス ID は AWS Systems Manager コマンドの実行に使用されます。
4. AWS Lambda は、boto モジュールを使用して Amazon EC2 をもう一度呼び出し、実行中の Windows サーバーのプライベート IP アドレスを取得し、そのアドレスを一時変数に保存します。

5. AWS Lambda は、Systems Manager をもう一度呼び出して、接続されているコンピュータ情報を取得します AWS Directory Service。
6. AWS Systems Manager ドキュメントは、Amazon EC2 Windows サーバーで PowerShell スクリプトを実行して、AD に接続されているコンピュータのプライベート IP アドレスを取得するのに役立ちます。
7. AD ドメインのユーザー名とパスワードは Parameter Store AWS Systems Manager に保存されます。 AWS Lambda Systems Manager は Parameter Store を呼び出し、AD の接続に使用するユーザー名とパスワードの値を取得します。
8. Systems Manager ドキュメントを使用して、ステップ 3 で前に取得したインスタンス ID を使用してAmazon EC2 Windows サーバーで PowerShell スクリプトを実行します。
9. Amazon EC2 AWS Directory Service は PowerShell コマンドを使用して接続し、使用されていないコンピュータや非アクティブのコンピュータを削除します。

ツール

AWS サービス

- AWS Directory Service には、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Relational Database Service (Amazon RDS) for SQL Server、Amazon FSx for Windows File Server AWS のサービス など、他の で Microsoft Active Directory (AD) を使用する複数の方法が用意されています。
- AWS Directory Service for Microsoft Active Directory では、ディレクトリ対応のワークロードと AWS リソースが で Microsoft Active Directory を使用できるようになります AWS クラウド。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon EventBridge は、アプリケーションをさまざまなソースからのリアルタイムデータに接続するのに役立つサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWSリソースへのアクセスを安全に管理できます。IAM を使用すると、 のサービスとリソースにアクセスできるユーザーまたはユーザーを指定し AWS、きめ細かなアクセス許可を一元管理し、アクセスを分析してアクセス許可を絞り込むことができます AWS。

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理を簡素化し、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できるようにします。
- AWS Systems Manager ドキュメントは、Systems Manager がマネージドインスタンスで実行するアクションを定義します。Systems Manager には、ランタイムでパラメータを指定して使用できる事前設定済みのドキュメントが 100 件以上含まれています。
- AWS Systems Manager Parameter Store は の機能であり、設定データ管理 AWS Systems Manager とシークレット管理のための安全な階層型ストレージを提供します。

その他のツール

- HashiCorp Terraform は、オープンソースのInfrastructure as Code (IaC) ツールで、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理できます。
- PowerShell は Windows、Linux、および macOS で動作するMicrosoft の自動化および構成管理プログラムです。
- 「Python」は汎用のコンピュータープログラミング言語です。

Code リポジトリ

このパターンのコードは、GitHub Custom AD Cleanup Automation ソリューションリポジトリにあります。

ベストプラクティス

- ドメインを自動的に結合します。 AWS Directory Service ドメインの一部となる Windows インスタンスを起動する場合は、後でインスタンスを手動で追加するのではなく、インスタンスの作成プロセス中にドメインに参加します。ドメインを自動的に結合するには、新しいインスタンスを起動するときに、ドメイン結合ディレクトリのドロップダウンリストから正しいディレクトリを選択します。詳細については、 AWS Directory Service 管理ガイドのAmazon EC2 Windows インスタンスを AWS Managed Microsoft AD Active Directory にシームレスに結合する」を参照してください。
- 未使用のアカウントを削除します。 使用したことのないアカウントを AD で検索するのが一般的です。システム内に残っている無効または非アクティブなアカウントと同様に、未使用のアカウン

トを無視すると、AD システムが遅くなったり、組織がデータ侵害に対して脆弱になる可能性があります。

- Active Directory のクリーンアップを自動化します。 セキュリティリスクを軽減し、古いアカウントが AD のパフォーマンスに影響を与えないようにするには、AD クリーンアップを定期的に実行する必要があります。スクリプトを記述することで、ほとんどの AD 管理およびクリーンアップタスクを実行できます。タスクの例には、無効化されたアカウントと非アクティブなアカウントの削除、空のグループと非アクティブなグループの削除、期限切れのユーザーアカウントとパスワードの検索などがあります。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
プロジェクトフォルダを作成し、ファイルを追加します。	リポジトリのクローンを作成し、プロジェクトフォルダを作成するには、次の手順を実行します。 1. このパターンの GitHub リポジトリを開きます。 2. クローンドロップダウンでクローンを作成するオプションを表示するには、コードボタンを選択します。 3. HTTPS タブで、ウェブ URL を使用して Clone で提供された URL をコピーします。 4. マシンにフォルダを作成し、プロジェクト名で名前を付けます。	DevOps エンジニア

タスク	説明	必要なスキル

	5. ローカルマシンでターミナルを開き、このフォルダに移動します。
	6. git リポジトリのクローンを作成するには、次のコマンドを使用します。
	`git clone <repository-URL>.git`
	7. リポジトリのクローンを作成したら、次のコマンドを使用してクローンされたディレクトリに移動します。
	`cd <directory name>`
	8. クローンされたリポジトリで、選択した統合開発環境 (IDE) でこのプロジェクトを開きます。	

Terraform 設定を使用してターゲットアーキテクチャをプロビジョニングする

タスク	説明	必要なスキル
Terraform の設定を初期化します。	Terraform ファイルを含む作業ディレクトリを初期化するには、次のコマンドを実行します。 `terraform init`	DevOps エンジニア
変更をプレビューします。	インフラストラクチャがデプロイされる前に、Terraform がインフラストラクチャに加える変更をプレビューできます	DevOps エンジニア

タスク	説明	必要なスキル
	。Terraform が必要に応じて変更を行うことを確認するには、次のコマンドを実行します。 `terraform plan`	
提案されたアクションを実行します。	`terraform plan` コマンドの結果が期待どおりに動作することを確認するには、次の手順を実行します。 1. 以下のコマンドを実行してください。 `terraform apply` 2. にサインインし AWS Management Console、リソースが存在することを確認します。	DevOps エンジニア
インフラストラクチャをクリーンアップします。	作成したインフラストラクチャをクリーンアップするには、次のコマンドを使用します。 `terraform destroy` destroy コマンドを確認するには、「」と入力しますyes。	DevOps エンジニア

デプロイメントを確認する

タスク	説明	必要なスキル
Lambda 関数を実行してテストします。	デプロイが正常に行われたことを確認するには、次の手順を実行します。 1. にサインイン AWS Management Console し、 コンソールを開きます。関数ページを開き、ADcleanup-Lambda-* で始まる関数名を選択します。 2. 関数の概要ページで、コードソースセクションのコードタブでテストを選択します。 3. テストイベントを保存するには、イベントの名前を指定し、保存を選択します。次に、イベントをテストするには、もう一度テストを選択します。 実行結果には、関数の出力が表示されます。	DevOps エンジニア
Lambda 関数の結果を表示します。	このパターンでは、EventBridge ルールは Lambda 関数を 1 日に 1 回実行します。Lambda 関数の結果を表示するには、次の手順を実行します。	DevOps エンジニア

タスク	説明	必要なスキル
	1. にサインイン AWS Management Console し、AWS Lambda コンソールを開きます。関数ページを開き、ADcleanup-Lambda-* で始まる関数名を選択します。 2. Monitor タブを選択し、View CloudWatch logs を選択します。 CloudWatch コンソールのロググループページには、Lambda 関数の結果が表示されます。	

使用後のインフラストラクチャのクリーンアップ

タスク	説明	必要なスキル
インフラストラクチャをクリーンアップします。	作成したインフラストラクチャをクリーンアップするには、次のコマンドを使用します。 `terraform destroy` destroy コマンドを確認するには、「」と入力しますyes。	DevOps エンジニア
クリーンアップ後に を確認します。	リソースが正常に削除されたことを確認します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
AD コンピュータを削除しようとすると、「アクセスが拒否されました」というメッセージが表示されます。デフォルトでは、アクションは AD サービスの一部として接続されている 2 つのプライベート IP アドレスを削除しようとするため、AD コンピュータを削除することはできません。	このエラーを回避するには、AD コンピュータ出力と Windows を実行しているマシンの出力の違いを一覧表示するときに、次の Python オペレーションを使用して最初の 2 つのコンピュータを無視します。 `Difference = Difference[2:]`
Lambda が Windows サーバーで PowerShell スクリプトを実行すると、Active Directory モジュールがデフォルトで使用可能であることが期待されます。モジュールが使用できない場合、Lambda 関数はGet-AdComputer is not installed on instance」というエラーを作成します。	このエラーを回避するには、EC2 インスタンスのユーザーデータを使用して必要なモジュールをインストールします。このパターンの GitHub リポジトリにある EC2WindowsUserdata スクリプトを使用します。

関連リソース

AWS ドキュメント

- Amazon EventBridge と AWS Identity and Access Management
- Systems Manager に必要なインスタンスのアクセス許可を設定する
- の ID とアクセスの管理 AWS Directory Service
- Amazon EC2 Windows インスタンスを AWS Managed Microsoft AD Active Directory に手動で結合する
- でのアイデンティティベースの IAM ポリシーの使用 AWS Lambda

その他のリソース

- AWS プロバイダー (Terraform ドキュメント)
- バックエンド設定 (Terraform ドキュメント)

- Terraform のインストール (Terraform ドキュメント)
- Python boto モジュール (Python Package Index リポジトリ)
- Terraform バイナリダウンロード (Terraform ドキュメント)

pytest フレームワーク AWS Glue を使用して で Python ETL ジョブのユニットテストを実行する

作成者: Praveen Kumar Jeyarajan (AWS)、Vaidy Sankaran (AWS)

概要

ローカル開発環境で の Python 抽出、変換、ロード (ETL) ジョブ AWS Glue のユニットテストを実行できますが、DevOps パイプラインでこれらのテストをレプリケートするのは困難で時間がかかる場合があります。 AWS テクノロジースタックでメインフレーム ETL プロセスをモダナイズする場合、ユニットテストは特に難しい場合があります。このパターンは、既存の機能をそのまま維持し、新機能をリリースしたときに主要なアプリケーション機能の中断を回避し、高品質のソフトウェアを維持しながら、ユニットテストを簡素化する方法を示しています。このパターンのステップとコードサンプルを使用して、 の Python ETL ジョブのユニットテストを実行するには、 の pytest フレームワーク AWS Glue を使用します AWS CodePipeline。このパターンを使用して、複数の AWS Glue ジョブをテストおよびデプロイすることもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon ECR Public Gallery からダウンロードされた、 AWS Glue ライブラリの Amazon Elastic Container Registry (Amazon ECR) イメージ URI https://gallery.ecr.aws/glue/aws-glue-libs
- ターゲット AWS アカウント と のプロファイルを持つ Bash ターミナル (任意のオペレーティングシステム上) AWS リージョン
- Python 3.10 以降
- Pytest
- テスト用の Moto Python ライブラリ AWS のサービス

アーキテクチャ

次の図は、Python に基づく AWS Glue ETL プロセスのユニットテストを一般的なエンタープライズスケールの AWS DevOps パイプラインに組み込む方法を示しています。

この図表は、次のワークフローを示しています：

1. ソースステージでは、 AWS CodePipeline はバージョニングされた Amazon Simple Storage Service (Amazon S3) バケットを使用して、ソースコードアセットを保存および管理します。これらのアセットには、サンプル Python ETL ジョブ (`sample.py`) 、ユニットテストファイル (`test_sample.py`) 、 AWS CloudFormation テンプレートが含まれます。次に、CodePipeline は、最新のコードをメインブランチから AWS CodeBuild プロジェクトに転送して、さらに処理します。
2. ビルドおよび公開ステージでは、前のソースステージの最新のコードは、 AWS Glue パブリック Amazon ECR イメージを使用してユニットテストされます。次に、テストレポートが CodeBuild レポートグループに公開されます。 AWS Glue ライブラリのパブリック Amazon ECR リポジトリのコンテナイメージには、 で PySpark ベースの ETL タスクを AWS Glue ローカルで実行およびユニットテストするために必要なすべてのバイナリが含まれています。パブリックコンテナリポジトリには、 でサポートされているバージョンごとに 1 つずつ、3 つのイメージタグがあります AWS Glue。デモンストレーションの目的から、このパターンでは `glue_libs_4.0.0_image_01` image タグを使用しています。このコンテナイメージを CodeBuild のランタイムイメージとして使用するには、使用する予定のイメージタグに対応するイメージ URI をコピーし、`TestBuild` リソースの GitHub リポジトリ内の `pipeline.yml` ファイルを更新します。
3. デプロイステージでは、CodeBuild プロジェクトが起動し、すべてのテストに合格すると Amazon S3 バケットにコードを発行します。
4. ユーザーは、 `deploy`フォルダの CloudFormation テンプレートを使用して AWS Glue タスクをデプロイします。

ツール

AWS のサービス

- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- AWS Glue はフルマネージド ETL サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。

- Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。

その他のツール

- Python は、高水準のインタープリター型汎用プログラミング言語です。
- Moto はテスト用の Python ライブラリです AWS のサービス。
- Pytest は、アプリケーションやライブラリの複雑な機能テストをサポートするように拡張できる小規模なユニットテストを作成するためのフレームワークです。
- の Python ETL ライブラリ AWS Glue は、PySpark バッチジョブのローカル開発で使用される Python ライブラリのリポジトリです AWS Glue。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-glue-jobs-unit-testing」リポジトリで利用できます。リポジトリには次のリソースが含まれています。

- `src` フォルダ内のサンプル Python ベースの AWS Glue ジョブ
- `tests` フォルダ内の関連ユニットテストケース (pytest フレームワークを使用して構築)
- `deploy` フォルダ内の CloudFormation テンプレート (YAML で記述)

ベストプラクティス

CodePipeline リソースのセキュリティ

CodePipeline のパイプラインに接続するソースリポジトリに暗号化と認証を使用することがベストプラクティスです。詳細については、CodePipeline ドキュメントの「Security best practices」を参照してください。

CodePipeline リソースで使用するモニタリングとログ記録

AWS ログ記録機能を使用して、ユーザーがアカウントで実行するアクションと使用するリソースを決定するのがベストプラクティスです。ログファイルには次の内容が表示されます。

- アクションが実行された日時
- アクション送信元 IP アドレス
- 不適切なアクセス権限が理由で失敗したアクション

ログ記録機能は、 AWS CloudTrail および Amazon CloudWatch Events で使用できます。CloudTrail を使用して、 によって行われた、または に代わって行われた AWS API コールおよび関連イベントを記録できます AWS アカウント。詳細については、CodePipeline ドキュメントの「 を使用した CodePipeline API コールのログ記録 AWS CloudTrail」を参照してください。 CodePipeline

CloudWatch Events を使用して、 で実行されている AWS クラウド リソースとアプリケーションをモニタリングできます AWS。CloudWatch イベントでアラートを作成することもできます。詳細については、CodePipeline ドキュメントの「CodePipeline イベントのモニタリング」を参照してください。

エピック

ソースコードをデプロイするには

タスク	説明	必要なスキル

コードアーカイブをデプロイ用に準備する。	1. GitHub aws-glue-jobs-unit-testing リポジトリから code.zip をダウンロードするか、コマンドラインツールを使用して.zip ファイルを自分で作成してください。例えば、Linux または Mac では、ターミナルで以下のコマンドを実行して.zip ファイルを作成できます。
	`git clone https://github.com/aws-samples/aws-glue-jobs-unit-testing.git`
	`cd aws-glue-jobs-unit-testing`
	`git checkout master`
	`zip -r code.zip src/ tests/ deploy/`
	2. にAWS Management Consoleサインインし、
		DevOps エンジニア

タスク	説明	必要なスキル
	AWS リージョン 任意の を選択します。 3. Amazon S3 バケットを作成し、作成した Amazon S3 バケットに .zip パッケージとcode.zipファイル (以前にダウンロード) をアップロードします。	

タスク	説明	必要なスキル
CloudFormation スタックを更新する。	1. にサインインし AWS Management Console 、CloudFormation コンソールを開きます。 2. [スタックの作成] を選択し、[既存のリソースを使用する (リソースのインポート)] を選択します。 3. [スタックの作成] ページの [テンプレートを指定] セクションで、[テンプレートファイルをアップロード] を選択し、[pipeline.yml] テンプレート (GitHub リポジトリからダウンロード) を選択します。[次へ] を選択します。 4. [スタック名] には [glue-unit-testing-pipeline] と入力するか、任意のスタック名を選択します。 5. [ApplicationStackName] には、あらかじめ入力されている [glue-codepipeline-app] 名を使用してください。これは、パイプラインによって作成される CloudFormation スタックの名前です。 6. [BucketName] には、あらかじめ入力されている aws-glue-artifacts-us-east-1 バケット名を使用してくださ	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	い。これは、.zip ファイルを含む Amazon S3 バケットの名前で、コードアーティファクトを保存するためにパイプラインによって使用されます。 7. [CodeZipFile] には、あらかじめ入力されている code.zip 値を使用してください。これは、サンプルコード Amazon S3 オブジェクトのキー名です。オブジェクトは .zip ファイルである必要があります。 8. [TestReportGroupName] には、あらかじめ入力されている glue-unittest-report という名前を使用してください。これは、ユニットテストレポートを保存するために作成された CodeBuild テストレポートグループの名前です。 9. [Next] (次へ) を選択してから [Configure stack options] (スタックオプションの設定) ページでもう一度 [Next] (次へ) をクリックします。 10. [確認] ページの [機能] で、[CloudFormation がカスタム名の IAM リソースを作成する可能性があることを承認する] オプションを選択します。	

タスク	説明	必要なスキル
	11[送信] を選択します。スタックの作成が完了すると、作成されたリソースが [リソース] タブに表示されます。スタックの作成には約 5～7 分かかります。 スタックは、Amazon S3 をソースとして使用して CodePipeline ビューを作成します。 Amazon S3 上記のステップでは、パイプラインは aws-glue-unit-test-pipeline です。	

ユニットテストを実行する

タスク	説明	必要なスキル
パイプラインでユニットテストを実行する。	1. デプロイされたパイプラインをテストするには、 にサインインし AWS Management Console、CodePipeline コンソールを開きます。 2. CloudFormation スタックによって作成されたパイプラインを選択し、[変更をリリース] を選択します。パイプラインの実行が開始されます (Amazon S3 バケット内の最新のコードを使用)。	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	3. Test_and_Build フェーズが終了したら、[詳細] タブを選択し、ログを調べます。 4. [レポート] タブを選択し、[レポート履歴] からテストレポートを選択すると、ユニットテストの結果が表示されます。 5. デプロイステージが完了したら、 AWS Glue コンソールでデプロイされた AWS Glue ジョブを実行してモニタリングします。詳細については、 AWS Glue ドキュメントの「モニタリング AWS Glue」を参照してください。	

すべての AWS リソースをクリーンアップする

タスク	説明	必要なスキル
環境内のリソースをクリーンアップする。	追加のインフラストラクチャコストを避けるため、このパターンで提供されている例を試した後は、必ずスタックを削除してください。 1. CloudFormation コンソールを開いて、作成したスタックを選択します。 2. [削除] を選択します。これにより、 AWS Identity and Access Managemen	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	t (IAM) ロール、IAM ポリシー、CodeBuild プロジェクトなど、スタックが作成したすべてのリソースが削除されます。	

トラブルシューティング

問題	ソリューション
CodePipeline サービスロールは Amazon S3 バケットにアクセスできません。	• CodePipeline サービスロールにアタッチされたポリシーについて、`s3:ListBucket` をポリシー内のアクションのリストに追加します。サービスロールポリシーを表示する手順については、「View the pipeline ARN and service role ARN (console)」を参照してください。CodePipeline サービスロールにアクセス許可を追加する」で説明されているように、サービスロールのポリシーステートメントを編集します。 • パイプラインの Amazon S3 アーティファクトバケットにアタッチされているリソースベースのポリシーについては、アーティファクトバケットポリシーとも呼ばれ、CodePipeline サービスロールに アクセス`s3:ListBucket` 許可の使用を許可するステートメントを追加します。
CodePipeline は、Amazon S3 バケットがバージョニングされていないというエラーを返します。	CodePipeline では、ソース Amazon S3 バケットをバージョニングする必要があります。Amazon S3 バケットでバージョニングを有効にします。手順については、「バケットでのバージョニングの有効化」を参照してください。

関連リソース

- AWS Glue
- ローカルでの AWS Glue ジョブの開発とテスト
- の AWS CloudFormation AWS Glue

追加情報

さらに、 AWS Command Line Interface () を使用して AWS CloudFormation テンプレートをデプロイできますAWS CLI。詳細については、CloudFormation ドキュメントの「変換を使用したテンプレートの迅速なデプロイ」を参照してください。

Amazon S3 に Helm v3 チャートリポジトリを設定する

作成者: Abhishek Sharma (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

このパターンは、Helm v3 レポジトリを Amazon Web Services (AWS) クラウド上の Amazon Simple Storage Service (Amazon S3) に統合することで、Helm v3 チャートの効率的な管理をサポートしています。このパターンを使用するには、Kubernetes に加え、Kubernetes のパッケージマネージャーである Helm に精通している必要があります。Helm リポジトリでチャートを保存し、チャートのバージョンを管理することで、システム停止時の平均復旧時間 (MTTR) を改善できます。

このパターンでは、AWS CodeCommit で Helm リポジトリを作成し、S3 バケットを Helm チャートリポジトリとして使用することで、組織全体の開発者がチャートを一元管理し、アクセスできるようにしています。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Python バージョン 2.7.12 以降
- pip
- サブネットと Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを備えた仮想プライベートクラウド (VPC)
- EC2 インスタンスに Git をインストールします。
- S3 バケットを作成するための AWS Identity and Access Management (IAM) アクセス
- クライアントマシンから Amazon S3 への IAM （プログラミングまたはロール）アクセス
- AWS CodeCommit リポジトリ
- AWS コマンドラインインターフェイス (AWS CLI)

製品バージョン

- Helm v3

- Python バージョン 2.7.12 以降

アーキテクチャ

ターゲットテクノロジースタック

- Amazon S3
- AWS CodeCommit
- Helm
- Kubectl
- Python と pip
- Git
- helm-s3 プラグイン

ターゲット アーキテクチャ

自動化とスケール

- Helm を既存の継続的な統合/継続的な配信 (CI/CD) 自動化ツールに組み込み、Helm チャートのパッケージングとバージョン管理を自動化できます (このパターンの対象外です)。
- GitVersion または Jenkins のビルド番号を使用して、チャートのバージョン管理を自動化できます。

ツール

- Helm – Helm は、Kubernetes クラスター上でアプリケーションをインストールおよび管理するのに役立つ Kubernetes のパッケージマネージャです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- helm-s3 プラグイン — helm-s3 プラグインは Amazon S3 とのやり取りをサポートします。Helm v2 と Helm v3 を併用できます。

エピック

Helm v3 のインストールと検証

タスク	説明	必要なスキル
Helm v3 クライアントをインストールします。	Helm クライアントをローカルシステムにダウンロードしてインストールするには、`sudo curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash` コマンドを実行します。	クラウド管理者、DevOps エンジニア
インストールを検証します。	`helm version --short` コマンドを実行して Helm クライアントを検証します。	クラウド管理者、DevOps エンジニア

S3 バケットを Helm リポジトリに初期化します。

タスク	説明	必要なスキル
Helm チャート用の S3 バケットを作成します。	独自の S3 バケットを作成します。バケットに `stable/myapp` という名前のフォルダを作成します。このパターンの例では、`s3://my-helm-charts/stable/myapp` をターゲットチャートリポジトリとして使用しています。	クラウド管理者、DevOps エンジニア
Amazon S3 用の helm-s3 プラグインをインストールします。	クライアントマシンに helm-s3 プラグインをインストールするには、`helm plugin install https://g`	クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	ithub.com/hypnoglow/helm-s3.git コマンドを実行します。	
Amazon S3 Helm リポジトリを初期化します。	ターゲットフォルダを Helm レポジトリとして初期化するには、helm s3 init s3://my-helm-charts/stable/myapp コマンドを使用します。 このコマンドは、index.yaml ターゲット内にファイルを作成し、保存されているすべてのチャート情報を追跡します。	クラウド管理者、DevOps エンジニア
新しく作成した Helm リポジトリを検証します。	index.yaml ファイルが正常に作成されたことを確認するには、aws s3 ls s3://my-helm-charts/stable/myapp/ コマンドを実行します。	クラウド管理者、DevOps エンジニア
Amazon S3 リポジトリをクライアントマシンの Helm に追加します。	ターゲットリポジトリのエイリアスを Helm クライアントマシンに追加するには、helm repo add stable-myapp s3://my-helm-charts/stable/myapp/ コマンドを使用します。	クラウド管理者、DevOps エンジニア

Amazon S3 Helm リポジトリにチャートをパッケージして公開します

タスク	説明	必要なスキル
Helm チャートのクローンを作成します。	CodeCommit リポジトリにローカル Helm チャートがない場合は、`git clone <url_of_your_helm_source_code>.git` コマンドを実行して GitHub リポジトリからクローンを作成します。	クラウド管理者、DevOps エンジニア
Helm チャートをパッケージ化します。	作成またはクローンしたチャートをパッケージ化するには、`helm package ./my-app` コマンドを使用します。 例として、このパターンでは `my-app` チャートを使用しています。このコマンドは、`my-app` チャートフォルダのすべてのコンテンツを、`Chart.yaml` ファイルに記載されているバージョン番号で名前が付けられたアーカイブファイルにパッケージ化します。	クラウド管理者、DevOps エンジニア
ローカルパッケージを Amazon S3 Helm リポジトリに保存します。	ローカルパッケージを Amazon S3 の Helm レポジトリにアップロードするには、`helm s3 push ./my-app-0.1.0.tgz stable-myapp` コマンドを実行します。	クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	コマンドでは、my-app はグラフのフォルダ名、0.1.0 は Chart.yaml に記載されているチャートのバージョン、stable-myapp はターゲットリポジトリのエイリアスです。	
Helm チャートを検索します。	チャートがローカルと Amazon S3 Helm リポジトリの両方に表示されることを確認するには、helm search repo stable-myapp コマンドを実行します。	クラウド管理者、DevOps エンジニア

Helm リポジトリのアップグレード

タスク	説明	必要なスキル
チャートを変更してパッケージ化します。	values.yaml で、replicaCount の値を 1 に設定し、次にチャートをパッケージ化し、今度は Chart.yaml のバージョンを 0.1.1 に変更します。CI/CD パイプラインで GitVersion などのツールまたは Jenkins ビルド番号を使用することで、自動化によるバージョン管理を実現するのが最適です。バージョン番号の自動化は、このパターンの範囲外です。helm package ./my-app/ コマンドを実行して、	クラウド管理者、DevOps エンジニア

タスク	説明	必要なスキル
	このパッケージをインストールします。	
新しいバージョンを Amazon S3 の Helm リポジトリにプッシュします。	新しいパッケージ、バージョン 0.1.1 を Amazon S3 の my-helm-charts Helm リポジトリにプッシュするには、`helm s3 push ./my-app-0.1.1.tgz stable-myapp` コマンドを実行します。	クラウド管理者、DevOps エンジニア

Helm チャートを検証します。	チャートがローカルと Amazon S3 Helm リポジトリの両方に表示されることを確認するには、以下のコマンドを実行します。
	`helm repo update`
	`helm search repo stable-myapp`
		クラウド管理者、DevOps エンジニア

Amazon S3 Helm リポジトリからチャートを検索してインストールします。

タスク	説明	必要なスキル
my-app チャートのすべてのバージョンを検索します。	使用可能なすべてのバージョンのチャートを表示するには、`--versions` フラグを使用して `helm search repo my-app --versions` コマンドを実行します。 フラグがない場合、Helm はデフォルトでアップロードされ	DevOps エンジニア

タスク	説明	必要なスキル
	た最新バージョンのチャートを表示します。	
Amazon S3 Helm リポジトリからチャートをインストールします。	自動インストールはこのパターンの対象外ですが、手動でインストールできます。前のタスクの検索結果には、my-app チャートの複数のバージョンが表示されます。Amazon S3 Helm リポジトリから新しいバージョン (0.1.1) をインストールするには、`helm upgrade --install my-app-release stable-myapp/my-app --version 0.1.1 --namespace dev` コマンドを使用します	DevOps エンジニア

Helm を使用して以前のバージョンにロールバックします。

タスク	説明	必要なスキル
特定のリビジョンの詳細を確認します。	自動ロールバックはこのパターンの対象外ですが、手動でロールバックできます。動作中のバージョンに切り替えたり、ロールバックしたりする前や、リビジョンをインストールする前に追加の検証を行う場合は、`helm get values --revision=2 my-app-release` コマンドを使用して各リビジョンに	DevOps エンジニア

タスク	説明	必要なスキル
	渡された値を確認してください。	
以前のバージョンにロールバックします。	自動ロールバックはこのパターンの対象外です。以前のリビジョンに手動でロールバックするには、`helm rollback my-app-release 1` コマンドを使用します。 この例では、リビジョン番号 1 にロールバックしています。	DevOps エンジニア

関連リソース

- 「psql ドキュメント」
- 「helm-s3 プラグイン (MIT ライセンス)」
- Amazon S3

AWS CodePipeline と AWS CDK を使用して CI/CD パイプラインを設定する

作成者: Konstantin Zarudaev (AWS)、Cizer Pereira (AWS)、Lars Kinder (AWS)、Yasha Dabas (AWS)

ホーム

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細については、継続的インテグレーションと継続的デリバリー (CI/CD) によるソフトウェアビルドとリリースプロセスの自動化は、反復可能なビルドとユーザーへの新機能の迅速な配信をサポートします。各コード変更をすばやく簡単にテストでき、ソフトウェアをリリースする前にバグを見つけて修正できます。各変更をステージングとリリースのプロセスで実行することで、アプリケーションやインフラストラクチャコードの品質を検証できます。CI/CD は、アプリケーション開発チームがコード変更をより頻繁かつ確実に行うのに役立つ文化、一連の運用原則、一連のプラクティスを体現しています。この実装は CI/CD パイプラインとも呼ばれます。

このパターンは、AWS CodeCommit リポジトリを使用して、アマゾン ウェブ サービス (AWS) で再利用可能な継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを定義します。AWS CodePipeline パイプラインは、AWS Cloud Development Kit (AWS CDK) v2 を使用して作成されています。

CodePipeline を使用すると、AWS マネジメントコンソールインターフェイス、AWS コマンドラインインターフェイス (AWS CLI)、AWS CloudFormation、または AWS SDK を使用して、ソフトウェアリリースプロセスのさまざまな段階をモデル化することができます。このパターンは、AWS CDK を使用して CodePipeline とそのコンポーネントを実装する方法を示しています。ライブラリを構築することに加えて、AWS CDK には AWS CDK アプリケーションを操作するための主要なツールであるツールキット (CLI コマンド cdk) が含まれています。他の機能の中でも、このツールキットには、1 つ以上のスタックを CloudFormation テンプレートに変換して AWS アカウントにデプロイする機能があります。

パイプラインにはサードパーティライブラリのセキュリティを検証するテストが含まれており、指定された環境で迅速かつ自動的なリリースを保証するのに役立ちます。アプリケーションを検証プロセスにかけることで、アプリケーション全体のセキュリティを強化できます。

このパターンの目的は、デプロイするリソースが DevOps のベストプラクティスに従っていることを確認しながら、CI/CD パイプラインを使用してコードをデプロイする時間を短縮することです。サンプルコードを実装すると、リンティング、テスト、セキュリティチェック、デプロイ、およびデ

プロイ後のプロセスを含む AWS CodePipeline が完成します。このパターンには Makefile のステップも含まれています。Makefile を使用すると、デベロッパーは CI/CD のステップをローカルで再現し、開発プロセスのスピードを上げることができます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 基本的な理解は以下のとおりです。
  - AWS CDK
  - AWS CloudFormation
  - AWS CodePipeline
  - TypeScript

制限

このパターンでは、TypeScript のみに AWS CDK を使用します。AWS CDK でサポートされている他の言語は対象外です。

製品バージョン

次のツールの最新バージョンを使用します。

- AWS コマンドラインインターフェイス (AWS CLI)
- cfn_nag
- git-remote-codecommit
- Node.js

アーキテクチャ

ターゲットテクノロジースタック

- AWS CDK
- AWS CloudFormation
- AWS CodeCommit
- AWS CodePipeline

ターゲットアーキテクチャ

パイプラインは、AWS CodeCommit リポジトリ (`SampleRepository`) の変更によってトリガーされます。最初に、CodePipeline はアーティファクトをビルドし、それ自体を更新し、デプロイプロセスを開始します。作成されたパイプラインは、次の 3 つの独立した環境に解決策をデプロイします。

- 開発 – アクティブな開発環境での 3 段階のコードチェック
- テスト – 統合/リグレッションテスト環境
- 本番 – 本番環境

開発段階には、リンティング、セキュリティ、ユニットテストの 3 つのステップが含まれます。これらの手順はプロセスを高速化するために並行して実行されます。パイプラインが動作するアーティファクトのみを提供するようにするため、プロセス内のステップが失敗するたびにパイプラインの実行が停止されます。開発段階のデプロイ後、パイプラインは検証テストを実行して結果を検証します。成功すると、パイプラインはデプロイ後の検証を含むテスト環境にアーティファクトをデプロイします。最後のステップは、アーティファクトを Prod 環境にデプロイすることです。

次の図は、CodeCommit リポジトリから CodePipeline によって実行されるビルドおよび更新プロセスまでのワークフロー、3 つの開発環境ステップ、および 3 つの環境のそれぞれにおけるその後のデプロイと検証を示しています。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。このパターンでは、CloudFormation テンプレートを使用して CodeCommit リポジトリと CodePipeline CI/CD パイプラインを作成できます。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

その他のツール

- cfn_nag は、CloudFormation テンプレートのパターンを探して潜在的なセキュリティ問題を特定するオープンソースのツールです。
- git-remote-codecommit は、Git を拡張して CodeCommit リポジトリからコードをプッシュおよびプルするためのユーティリティです。
- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。

コード

このパターンのコードは、GitHub 内の「AWS CodePipeline with CI/CD practices」リポジトリで利用できます。

ベストプラクティス

AWS Identity and Access Management (IAM) ポリシーなどのリソースを確認して、組織のベストプラクティスに沿っていることを確認します。

エピック

ツールをインストールする

タスク	説明	必要なスキル

macOS または Linux にツールをインストールします。	macOS または Linux を使用している場合は、任意のターミナルで次のコマンドを実行するか、Homebrew for Linux を使用してツールをインストールできます。
	`brew install`
	`brew install git-remote-codecommit`
		DevOps エンジニア

タスク	説明	必要なスキル
	`brew install ruby brew-gem` `brew-gem install cfn-nag`	
AWS CLI をセットアップします。	AWS CLI を設定するには、お使いのオペレーティングシステムの手順に従ってください。 • Windows: AWS CodeCommit 認証情報ヘルパーを使用して Windows で AWS CLI リポジトリへの HTTPS 接続をセットアップする手順 • Linux、macOS、Unix: AWS CLI 認証情報ヘルパーを使用した、Linux、macOS、または Unix での AWS CodeCommit リポジトリへの HTTPS 接続のセットアップ手順	DevOps エンジニア

初期のデプロイをセットアップする

タスク	説明	必要なスキル
コードをダウンロードまたはクローンします。	このパターンで使用されるコードを取得するには、以下のいずれかを実行します。 • GitHub リポジトリ内のリリースから最新のソースコードをダウンロードし、ダ	DevOps エンジニア

タスク	説明	必要なスキル

	ウンロードしたファイルをフォルダに解凍します。
	• 次のコマンドを実行して、プロジェクトのクローンを作成します。
	`git clone --depth 1 https://github.com/aws-samples/aws-codepipeline-cicd.git`
	クローニングしたリポジトリから .git ディレクトリを削除します。
	`cd ./aws-codepipeline-cicd`
	`rm -rf ./.git`
	後で、新しく作成した AWS CodeCommit リポジトリをリモートオリジンとして使用します。	

タスク	説明	必要なスキル

AWS アカウントに接続します。	一時的なセキュリティトークンまたはランディングゾーン認証を使用して接続できます。正しいアカウントと AWS リージョンを使用していることを確認するには、以下のコマンドを実行します。
	
	`AWS_REGION="eu-west-1"`
	`ACCOUNT_NUMBER=$(aws sts get-caller-identity --query Account --output text)`
	`echo "${ACCOUNT_NUMBER}"`
		DevOps エンジニア

タスク	説明	必要なスキル

環境を起動します。	AWS CDK 環境を起動するには、以下のコマンドを実行します。
	`npm install`
	`npm run cdk bootstrap "aws://${ACCOUNT_NUMBER}/${AWS_REGION}"`
	環境を正常に起動すると、次の出力が表示されるはずです。
	`# Bootstrapping environment aws://{account}/{region}...`
	`# Environment aws://{account}/{region} bootstrapped`
	AWS CDK 起動の詳細については、AWS CDK のドキュメントを参照してください。
		DevOps エンジニア

タスク	説明	必要なスキル

テンプレートを合成します。	AWS CDK アプリケーションを合成するには、cdk synth コマンドを使用します。
	`npm run cdk synth`
	次のような出力が表示されます。
	`Successfully synthesized to <path-to-directory>/aws-codepipeline-cicd/cdk.out`
	`Supply a stack id (CodePipeline, Dev-MainStack) to display its template.`
		DevOps エンジニア

タスク	説明	必要なスキル

CodePipeline スタックをデプロイします。	CloudFormation テンプレートを起動して合成したので、デプロイできます。デプロイすると、CodePipeline パイプラインと CodeCommit リポジトリが作成されます。これらは、パイプラインのソースとトリガーになります。
	`npm run cdk -- deploy CodePipeline --require-approval never`
	コマンドを実行すると、CodePipeline スタックが正常にデプロイされ、情報が出力されるはずです。`CodePipeline.RepositoryName` により、AWS アカウントの CodeCommit リポジトリの名前が表示されます。
	`CodePipeline: deploying...`
	`CodePipeline: creating CloudFormation changeset...`
	`# CodePipeline Outputs: CodePipeline.RepositoryName = SampleRepository`
	`Stack ARN: arn:aws:cloudformation:REGION:ACCOUNT-ID`
		DevOps エンジニア

タスク	説明	必要なスキル
	`:stack/CodePipeline/STACK-ID`	

タスク	説明	必要なスキル

リモート CodeCommit リポジトリとブランチを設定します。	デプロイが成功すると、CodePipeline はパイプラインの最初の実行を開始します。この実行は AWS CodePipeline コンソールで確認できます。AWS CDK と CodeCommit はデフォルトブランチを開始しないため、この最初のパイプライン実行は失敗し、次のエラーメッセージが返されます。
	`The action failed because no branch named main was found in the selected AWS CodeCommit repository SampleRepository. Make sure you are using the correct branch name, and then try again. Error: null`
	このエラーを修正するには、リモートオリジンを SampleRepository として設定し、必要な main ブランチを作成します。
	`RepoName=$(aws cloudformation describe-stacks --stack-name CodePipeline --query "Stacks[0].Outputs[?OutputKey=='RepositoryName'].OutputValue" --output text)`
		DevOps エンジニア

タスク	説明	必要なスキル

	`echo "${RepoName}"`
	`#`
	`git init`
	`git branch -m master main`
	`git remote add origin codecommit://${RepoName}`
	`git add .`
	`git commit -m "Initial commit"`
	`git push -u origin main`	

デプロイされた CodePipeline パイプラインをテストする

タスク	説明	必要なスキル
変更をコミットしてパイプラインを有効にします。	初期デプロイメントが成功すると、SampleRepository の main ブランチをソースブランチとして持つ完全な CI/CD パイプラインが完成するはずです。main ブランチに変更をコミットするとすぐに、パイプラインは次の一連のアクションを開始して実行します。 1. CodeCommit リポジトリからコードを取得します。 2. コードをビルドします。 3. パイプライン自体を更新します (UpdatePipeline)。	DevOps エンジニア

タスク	説明	必要なスキル
	4. リンティング、セキュリティ、ユニットテストのチェック用に 3 つのジョブをparallel して実行します。 5. 成功した場合、パイプラインは Main スタックを `./lib/main-stack.ts` から開発環境にデプロイします。 6. デプロイされたリソースのデプロイ後チェックを実行します。CodePipeline コンソールでは、CodePipeline のすべてのステップと結果を確認できます。 7. 成功すると、パイプラインはテスト環境と Prod 環境のデプロイと検証を繰り返します。	

Makefile を使用してローカルでテストする

タスク	説明	必要なスキル
Makefile を使用して開発プロセスを実行します。	`make` コマンドを使用してパイプライン全体をローカルで実行することも、個別のステップ (`make linting` など) を実行することもできます。 `make` を使用してテストするには、次のアクションを実行します。	アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル
	• ローカルパイプラインを実装: `make` • ユニットテストのみを実行: `make unittest` • 現在のアカウントにデプロイ: `make deploy` • 環境をクリーンアップ: `make clean`	

リソースをクリーンアップする

タスク	説明	必要なスキル
AWS CDK アプリケーションリソースを削除します。	AWS CDK アプリをクリーンアップするには、次のコマンドを実行します。 `cdk destroy --all` 起動中に作成された Amazon Simple Storage Service (Amazon S3) バケットは、自動的に削除されないことに注意してください。削除を許可する保持ポリシーが必要か、AWS アカウントで手動で削除する必要があります。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
テンプレートが期待どおりに動作しません。	何か問題が発生し、テンプレートが機能しない場合は、次があることを確認してください。 • 適切なバージョンのツール。 • ターゲット AWS アカウントへのアクセス (ネットワーク接続)。 • ターゲット AWS アカウントへの十分なアクセス許可。

関連リソース

- IAM アイデンティティセンターで一般的なタスクを開始する
- AWS CodePipeline ドキュメント
- AWS CDK

cert-managerと Let's Encrypt を使用して Amazon EKS のアプリケーションのエンドツーエンド暗号化を設定

作成者: Mahendra Revanasiddappa (AWS) と Vasanth Jeyaraj (AWS)

概要

エンドツーエンド暗号化の実装は複雑で、マイクロサービスアーキテクチャ内の各アセットの証明書を管理する必要があります。Amazon Web Services (AWS) ネットワークの端にあるTransport Layer Security (TLS) 接続は、Network Load Balancer または Amazon API Gateway を使用して終了できますが、一部の組織ではエンドツーエンドの暗号化が必要です。

このパターンでは、入力に NGINX Ingress コントローラーを使用します。これは、Kubernetes Ingress を作成する場合、イングレスリソースがNetwork Load Balancer を使用するためです。Network Load Balancer は、クライアント証明書のアップロードを許可しません。したがって、Kubernetes Ingress では相互TLSを実現できません。

このパターンは、アプリケーション内のすべてのマイクロサービス間の相互認証を必要とする組織を対象としています。相互 TLS は、ユーザー名やパスワードを管理する負担を軽減し、すぐに使えるセキュリティフレームワークを使用することもできます。このパターンのアプローチは、接続されているデバイスが多数ある組織や、厳格なセキュリティガイドラインに準拠する必要がある場合にも適合します。

このパターンは、Amazon Elastic Kubernetes Service (Amazon EKS) 上で実行されるアプリケーションにエンドツーエンドの暗号化を実装することで、組織のセキュリティ体制強化に役立ちます。このパターンでは、「Amazon EKS リポジトリの GitHub エンドツーエンド暗号化」 のサンプルアプリケーションとコードを提供し、マイクロサービスが Amazon EKS でエンドツーエンド暗号化を使用してどのように実行されるかを示します。このパターンのアプローチでは、Kubernetes のアドオンである「cert-manager」 を使用し、認証局 (CA) として 「Let's Encrypt」 を使用します。Let's Encrypt は費用対効果の高い証明書管理ソリューションで、90 日間有効な証明書を無料で提供しています。Cert-manager は、新しいマイクロサービスが Amazon EKS にデプロイされたときに、証明書のオンデマンドプロビジョニングとローテーションを自動化します。

対象者

このパターンは、Kubernetes、TLS、Amazon Route 53、およびドメインネームシステム (DNS) の使用経験があるユーザーに推奨されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 既存の アマゾン EKS クラスター。
- macOS、Linux、または Windows にインストールされ、設定された AWS コマンドラインインターフェイス (AWS CLI) バージョン 1.7 以降。
- Amazon EKS クラスターにアクセスするために、インストールおよび設定された `kubectl` コマンドラインユーティリティ。これについての詳細は、Amazon EKS ドキュメントの「kubectl のインストール」 を参照してください。
- アプリケーションをテストするための既存のアプリケーション名です。これについての詳細は、Amazon Route 53 デベロッパーガイドの「 Amazon Route 53 を使用したドメイン名の登録」を参照してください。
- ローカルマシンにインストールされている最新の 「Helm」 バージョン。これについての詳細は、Amazon EKS ドキュメントの「Amazon EKS での Helm の使用」 と GitHub 「Helm」 リポジトリを参照してください。
- ローカルマシンに複製された GitHub 「Amazon EKS リポジトリの エンドツーエンド暗号化」。
- 複製された GitHub 「Amazon EKS リポジトリのエンドツーエンド暗号化」から `policy.json` と `trustpolicy.json` のファイルの以下の値を置き換えます:
  - `<account number>` — ソリューションをデプロイするアカウントの AWS アカウント ID と置き換えます。
  - `<zone id>` — ドメイン名の Route 53 ゾーン ID に置き換えます。
  - `<node_group_role>` — Amazon EKS ノードに関連付けられている AWS 識別とアクセス管理 (IAM) ロールの名前に置き換えます。
  - `<namespace>` — NGINX Ingress コントローラーとサンプルアプリケーションをデプロイする Kubernetes 名前空間に置き換えます。
  - `<application-domain-name>` — Route 53 の DNS ドメイン名に置き換えます。

機能制限

- このパターンでは、証明のローテーション方法を説明するものではなく、Amazon EKS のマイクロサービスで証明書を使用する方法のみを示しています。

アーキテクチャ

次の図表は、このパターンのアプリケーションのワークフローとアーキテクチャコンポーネントを示しています。

この図表は、次のワークフローを示しています：

1. クライアントは DNS 名へのアプリケーションへのアクセスリクエストを送信します。
2. Route 53 レコードは Network Load Balancer の CNAMEです。
3. Network Load Balancer は、TLS リスナーで設定された NGINX Ingress コントローラーにリクエストを転送します。NGINX Ingress コントローラーと Network Load Balancer 間の通信は HTTPS プロトコルに従います。
4. NGINX Ingress Controller は、アプリケーションサービスへのクライアントのリクエストに基づいてパスベースのルーティングを実行します。
5. アプリケーションサービスはリクエストをアプリケーションポッドに転送します。このアプリケーションは、シークレットを呼び出すことで。同じ証明書を使用するように設計されています。
6. ポッドは cert-managerの証明書を使用してサンプルアプリケーションを実行します。NGINX Ingress コントローラーとポッド間の通信には HTTPS が使用されます。

> **Note**
>
> Cert-manager は独自の名前空間で実行されます。Kubernetes クラスターロールを使用して、証明書を特定の名前空間のシークレットとしてプロビジョニングします。これらの名前空間はアプリケーションポッドと NGINX Ingress コントローラーにアタッチできます。

ツール

AWS サービス

- Amazon Elastic Kubernetes Service (Amazon EKS) は、AWS で Kubernetes を簡単に実行できるようにするマネージド型サービスです。ユーザー独自の Kubernetes コントロールプレーンまたはノードをインストール、操作、維持する必要はありません。
- Elastic Load Balancing は、受信したトラフィックを複数のターゲット、コンテナ、IPアドレスに自動的に分散します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

その他のツール

- 「cert-manager」 は Kubernetes のアドオンで、証明書をリクエストして Kubernetes コンテナに配布し、証明書の更新を自動化します。
- 「NGINX Ingress Controller 」は、Kubernetes およびコンテナ化された環境におけるクラウドネイティブアプリケーション向けのトラフィック管理ソリューションです。

エピック

Route 53 でパブリックホストゾーンを作成して設定

タスク	説明	必要なスキル
Route 53 にドメインのパブリックホストゾーンを作成します。	AWS マネジメントコンソールにサインインし、Amazon Route 53 コンソールを開き、[ホストゾーン] を選択し、[ホストゾーンの作成] を選択します。パブリックホストゾーンを作成し、ゾーン ID を記録します。これについての詳細は、Amazon Route 53 ドキュメントの「公開ホストゾーンの作成」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
	Note ACME DNS01 は DNS プロバイダーを使用して、cert-manager が証明書を発行するためのチャレンジを投稿します。このチャレンジでは、ドメイン名の TXT レコードに特定の値を入力して、そのドメイン名の DNS を管理していることを証明するよう求められます。Let's Encrypt が ACME クライアントにトークンを渡した後、クライアントはそのトークンとアカウントキーから得た TXT レコードを作成し、そのレコードを `_acme-challenge.<YOURDOMAIN>` に保存します。次に、Let's Encrypt は DNS にそのレコードを問い合わせます。一致するものが見つかったら、証明書の発行に進むことができます。	

証明書マネージャーがパブリックホストゾーンにアクセスできるように IAM ロールを設定する

タスク	説明	必要なスキル
cert-manager 用の IAM ポリシーを作成します。	ユーザーが Route 53 ドメインを所有していることを検証する権限を cert-manager に提供するには、IAM ポリシーが必要です。`policy.json` のサンプル IAM ポリシーが、複製されたGitHub「Amazon EKS リポジトリのエンドツーエンドの暗号化」リポジトリの `1-IAMRole` ディレクトリにあります。 次の AWS CLI コマンドを使用して、IAM ポリシーを作成します。 `aws iam create-policy \ --policy-name PolicyForCertManager \ --policy-document file://policy.json`	AWS DevOps
cert-managerの IAM ロールを作成します。	IAM ポリシーを作成したら、IAM ロールを作成する必要があります。`trustpolicy.json` サンプル IAM ロールが `1-IAMRole` ディレクトリに提供されます。 IAM ロールを作成するには、次のコマンドをAWS CLIに入力します。	AWS DevOps

タスク	説明	必要なスキル
	`aws iam create-role \ --role-name RoleForCertManager \ --assume-role-policy-document file://trustpolicy.json`	
ロールへのポリシーの付与	AWS CLI に以下のコマンドを入力して、 IAM ロールに IAMポリシーをアタッチします。`AWS_ACCOUNT_ID` を自分の AWS アカウントのID に置き換えます。 `aws iam attach-role-policy \ --policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/PolicyForCertManager \ --role-name RoleForCertManager`	AWS DevOps

Amazon EKS で NGINX Ingress コントローラーをセットアップする

タスク	説明	必要なスキル
NGINX Ingress コントローラーをデプロイします。	Helm を使用する `nginx-ingress` の最新バージョンをインストールします。デプロイする前に、要件に応じて `nginx-ingress` の設定を変更できます。このパターンでは、`5-Nginx-Ingress-Controller` ディレクトリの使	AWS DevOps

タスク	説明	必要なスキル
	用可能の注釈付き、内部向けのNetwork Load Balancer を使用します。 `5-Nginx-Ingress-Controller` ディレクトリから次の Helm コマンドを実行して、 NGINX Ingress コントローラーをインストールします。 `helm install test-nginx nginx-stable/nginx-ingress -f 5-Nginx-Ingress-Controller/values_internal_nlb.yaml`	
コントローラーがインストールされていることを確認します。	`helm list` コマンドを入力します。出力では NGINX Ingress コントローラーがインストールされていることが表示されるはずです。	AWS DevOps

タスク	説明	必要なスキル
Route 53 A レコードを作成します。	A レコードは NGINX Ingress コントローラーによって作成された Network Load Balancer を指します。 1. Network Load BalancerのDNS 名を取得します。手順については、「ELB ロードバランサーの DNS 名の取得」 を参照してください。 2. Amazon Route 53 コンソールで、ホストゾーンを選択します。 3. レコードを作成するパブリックホストゾーンを選択し、レコードの作成 を選択します。 4. レポートの [Name](名前) を入力します。 5. [レコードタイプ] で、[A — IPv4 アドレスと一部のAWS リソースにトラフィックをルーティングする] を選択します。 6. Alias を有効にします。 7. [トラフィックのルーティング先] で、次の操作を行います： a. [Network Load Balancer へのエイリアス] を選択します。 b. Network Load Balancer がデプロイされている	AWS DevOps

タスク	説明	必要なスキル
	AWS リージョンを選択します。 c. Network Load Balancer の DNS 名を入力します。 8. [レコードを作成] を選択します。	

Amazon EKS に NGINX 仮想サーバーをセットアップ

タスク	説明	必要なスキル
NGINX 仮想サーバーをデプロイします。	NGINX VirtualServer リソースは、イングレスリソースの代替となるロードバランシング設定です。NGINX VirtualServer リソースを作成するための設定は、`6-Nginx-Virtual-Server` ディレクトリの `nginx_virtualserver.yaml` のファイルで利用可能です。NGINX VirtualServer リソースを作成するには、`kubectl` に以下のコマンドを入力します。 `kubectl apply -f nginx_virtualserver.yaml` **⚠ Important** ファイル内のアプリケーションドメイン	AWS DevOps

タスク	説明	必要なスキル
	名、証明書シークレット、およびアプリケーションサービス名を必ず更新してくださいnginx_virtualserver.yaml 。	
NGINX 仮想サーバーが作成されていることを確認します。	kubectl にある以下のコマンドを入力して、NGINX VirtualServer リソースが正常に作成されたことを確認します。 `kubectl get virtualserver` **Note** Host 列がアプリケーションのドメイン名と一致することを確認します。	AWS DevOps

タスク	説明	必要なスキル
TLS を有効にして NGINX ウェブサーバーをデプロイします。	このパターンでは、TLS が有効になっている NGINX Web サーバーを、エンドツーエンドの暗号化をテストするためのアプリケーションとして使用します。テストアプリケーションのデプロイに必要な設定ファイルは、`demo-webserver` のディレクトリに使用可能です。 アプリケーションのテストを行うには、次のコマンドを`kubectl` に入力します。 `kubectl apply -f nginx-tls-ap.yaml`	AWS DevOps

タスク	説明	必要なスキル
テストアプリケーションリソースが作成されたことを確認します。	次のコマンドを kubectl に入力して、テストアプリケーションに必要なリソースが作成されていることを確認します • `kubectl get deployments` **Note** Ready 列と Available 列を検証します。 • `kubectl get pods | grep -i example-deploy` **Note** ポッドは running状態である必要があります。 • `kubectl get configmap` • `kubectl get svc`	AWS DevOps

タスク	説明	必要なスキル
アプリケーションを検証します。	1. <application-domain-name> を 先ほど作成したRoute53 DNS 名に置き換えて、次のコマンドを入力します。 `curl --verbose https://<application-domain-name>` 2. アプリケーションにアクセスできることを確認します。	AWS DevOps

関連リソース

「AWS リソース」

- 「Amazon Route 53 コンソールを使用したレコードの作成」 (Amazon Route 53 ドキュメント)
- 「Amazon EKS の NGINX イングレスコントローラーでのNetwork Load Balancer の使用」 (AWS ブログ投稿)

その他のリソース

- 「Route 53」 (cert-managerドキュメント)
- 「DNS01 チャレンジプロバイダーの設定」 (cert-managerドキュメント)
- 「Let's Encrypt DNS チャレンジ」(Let's Encrypt ドキュメント)

Flux を使用して Amazon EKS マルチテナントアプリケーションのデプロイを簡素化する

作成者: Nadeem Rahaman (AWS)、Aditya Ambati (AWS)、Aniket Dekate (AWS)、および Shrikant Patil (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

製品やサービスを提供する多くの企業は、内部ビジネス機能間のデータ障壁を維持するために必要なデータ規制業界です。このパターンでは、Amazon Elastic Kubernetes Service (Amazon EKS) のマルチテナンシー機能を使用して、単一の Amazon EKS クラスターを共有するテナントまたはユーザー間で論理的および物理的な分離を実現するデータプラットフォームを構築する方法について説明します。このパターンは、以下のアプローチを通じて分離を提供します。

- Kubernetes 名前空間の分離
- ロールベースのアクセスコントロール (RBAC)
- ネットワークポリシー
- リソースクォータ
- AWS Identity and Access Management サービスアカウント (IRSA) の (IAM) ロール

さらに、このソリューションは Flux を使用して、アプリケーションをデプロイするときにテナント設定をイミュータブルに保ちます。設定に Flux `kustomization.yaml` ファイルを含むテナントリポジトリを指定することで、テナントアプリケーションをデプロイできます。

このパターンでは、以下を実装します。

- AWS CodeCommit Terraform スクリプトを手動でデプロイすることで作成されるリポジトリ、AWS CodeBuild プロジェクト、 AWS CodePipeline パイプライン。
- テナントをホストするために必要なネットワークコンポーネントとコンピューティングコンポーネント。これらは、Terraform を使用して CodePipeline と CodeBuild を介して作成されます。
- Helm チャートで設定されるテナント名前空間、ネットワークポリシー、リソースクォータ。
- Flux を使用してデプロイされた、異なるテナントに属するアプリケーション。

独自の要件とセキュリティ上の考慮事項に基づいて、マルチテナンシー用の独自のアーキテクチャを慎重に計画および構築することをお勧めします。このパターンは、実装の開始点を提供します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Command Line Interface （ AWS CLI) バージョン 2.11.4 以降、インストールおよび設定済み
- ローカルマシンにインストールされた Terraform バージョン 0.12 以降
- Terraform AWS プロバイダーバージョン 3.0.0 以降
- Kubernetes プロバイダーバージョン 2.10 以降
- Helm プロバイダーバージョン 2.8.0 以降
- Kubectl プロバイダーバージョン 1.14 以降

制約事項

- Terraform 手動デプロイの依存関係: CodeCommit リポジトリ、CodeBuild プロジェクト、CodePipeline パイプラインの作成を含むワークフローの初期セットアップは、手動 Terraform デプロイに依存します。これにより、インフラストラクチャの変更には手動による介入が必要になるため、自動化とスケーラビリティの点で潜在的な制限が生じます。
- CodeCommit リポジトリの依存関係: ワークフローは、ソースコード管理ソリューションとして CodeCommit リポジトリに依存し、密接に結合されています AWS のサービス。

アーキテクチャ

ターゲットアーキテクチャ

このパターンでは、次の図に示すように、3 つのモジュールをデプロイして、データプラットフォームのパイプライン、ネットワーク、コンピューティングインフラストラクチャを構築します。

パイプラインアーキテクチャ：

ネットワークアーキテクチャ：

コンピューティングアーキテクチャ：

ツール

AWS のサービス

- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストールまたは維持 AWS することなく、 で Kubernetes を実行するのに役立ちます。
- AWS Transit Gateway は、仮想プライベートクラウド (VPC) とオンプレミスネットワークを接続する中央ハブです。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

その他のツール

- Cilium ネットワークポリシーは、Kubernetes L3 および L4 ネットワークポリシーをサポートしています。これらは L7 ポリシーで拡張して、HTTP、Kafka、gRPC、およびその他の同様のプロトコルの API レベルのセキュリティを提供できます。
- Flux は、Kubernetes へのアプリケーションのデプロイを自動化する Git ベースの継続的デリバリー (CD) ツールです。
- Helm は、Kubernetes クラスターへのアプリケーションのインストールと管理に役立つ Kubernetes のオープンソースパッケージマネージャーです。
- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンのコードは、GitHub EKS Multi-Tenancy Terraform Solution リポジトリで入手できます。

ベストプラクティス

この実装を使用するためのガイドラインとベストプラクティスについては、以下を参照してください。

- Amazon EKS マルチテナンシーのベストプラクティス
- Flux ドキュメント

エピック

Terraform のビルド、テスト、デプロイの各ステージのパイプラインを作成する

タスク	説明	必要なスキル
プロジェクトリポジトリのクローンを作成します。	ターミナルウィンドウで次のコマンドを実行して、GitHub EKS マルチテナンシー Terraform Solution リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/aws-eks-multitenancy-deployment.git`	AWS DevOps
Terraform S3 バケットと Amazon DynamoDB をブートストラップします。	1. `bootstrap` フォルダで `bootstrap.sh` ファイルを開き、S3 バケット名、DynamoDB テーブル名、および の変数値を更新します AWS リージョン。 `S3_BUCKET_NAME="<S3_BUCKET_NAME>"`	AWS DevOps

タスク	説明	必要なスキル
	DYNAMODB_TABLE_NAME="<DYNAMODB_NAME>" REGION="<AWS_REGION>" 2. bootstrap.sh スクリプトを実行します。このスクリプトには AWS CLI、前提条件の一部としてインストールした が必要です。 cd bootstrap ./bootstrap.sh	

タスク	説明	必要なスキル

run.sh および locals.tf ファイルを更新します。	1. ブートストラッププロセスが正常に完了したら、bootstrap.sh スクリプトの variables セクションから S3 バケットと DynamoDB テーブル名をコピーします。
	`# Variables S3_BUCKET_NAME="<S3_BUCKET_NAME>" DYNAMODB_TABLE_NAME="<DYNAMODB_NAME"`
	2. これらの値をプロジェクトのルートディレクトリにあるrun.shスクリプトに貼り付けます。
	`BACKEND_BUCKET_ID="<SAME_NAME_AS_S3_BUCKET_NAME>" DYNAMODB_ID="<SAME_NAME_AS_DYNAMODB_NAME>"`
	3. プロジェクトコードを CodeCommit リポジトリにアップロードします。次の変数を demo/pipeline/locals.tf ファイルtrueで に設定することで、Terraform を使用してこのリポジトリを自動的に作成できます。
		AWS DevOps

タスク	説明	必要なスキル
	`create_new_repo = true` 4. 要件に従って `locals.tf` ファイルを更新し、パイプラインリソースを作成します。	

パイプラインモジュールをデプロイします。	パイプラインリソースを作成するには、次の Terraform コマンドを手動で実行します。これらのコマンドを自動的に実行するためのオーケストレーションはありません。
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t init`
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t plan`
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t apply`
		AWS DevOps

ネットワークインフラストラクチャを作成する

タスク	説明	必要なスキル
パイプラインを開始します。	1. `templates` フォルダで、`buildspec` ファイルに次の変数が に設定されていることを確認しますnetwork。	AWS DevOps

タスク	説明	必要なスキル

	<pre>TF_MODULE_TO_BUILD:
	 "network"</pre>
	2. CodePipeline コンソールのパイプラインの詳細ページで、変更のリリースを選択してパイプラインを開始します。
	
	この初回実行後、CodeCommit リポジトリのメインブランチに変更をコミットするたびに、パイプラインが自動的に開始されます。
	
	パイプラインには次のステージが含まれます。
	
	• `validate` は Terraform を初期化し、checkov ツールと tfsec ツールを使用して Terraform セキュリティスキャンを実行し、スキャンレポートを S3 バケットにアップロードします。
	• `plan` は Terraform プランを表示し、プランを S3 バケットにアップロードします。
	• `apply` は S3 バケットから Terraform プラン出力を適用し、 AWS リソースを作成します。	

タスク	説明	必要なスキル
	• destroy は、applyステージ中に作成された AWS リソースを削除します。このオプションのステージを有効にするには、 demo/pipeline/locals.tf ファイルtrueで次の変数を に設定します。 `enable_destroy_stage = true`	

タスク	説明	必要なスキル
ネットワークモジュールを介して作成されたリソースを検証します。	パイプラインが正常にデプロイされた後に、次の AWS リソースが作成されていることを確認します。 • 3 つのパブリックサブネットと 3 つのプライベートサブネット、インターネットゲートウェイ、NAT ゲートウェイを持つ Egress VPC。 • 3 つのプライベートサブネットを持つ Amazon EKS VPC。 • テナント 1 とテナント 2 VPCs とそれぞれ 3 つのプライベートサブネット。 • すべての VPC アタッチメントと各プライベートサブネットへのルートを持つトランジットゲートウェイ。 • 送信先 CIDR ブロックが の Amazon EKS 出力 VPC の静的トランジットゲートウェイルート`0.0.0.0/0` 。これは、すべての VPCs Amazon EKS 出力 VPC を介してアウトバウンドインターネットアクセスできるようにするために必要です。	AWS DevOps

コンピューティングインフラストラクチャを作成する

タスク	説明	必要なスキル

を更新`locals.tf` して、CodeBuild プロジェクトの VPC へのアクセスを有効にします。	Amazon EKS プライベートクラスターのアドオンをデプロイするには、CodeBuild プロジェクトを Amazon EKS VPC にアタッチする必要があります。
	1. `demo/pipeline` フォルダで、`locals.tf` ファイルを開き、`vpc_enabled` 変数を に設定します`true`。
	2. `run.sh` スクリプトを実行して、パイプラインモジュールに変更を適用します。
	`demo/pipeline/locals.tf`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd init`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd plan`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd apply`
		AWS DevOps

`buildspec` ファイルを更新してコンピューティングモジュールを構築します。	`templates` フォルダのすべての `buildspec` YAML ファイルで、`TF_MODULE`	AWS DevOps

タスク	説明	必要なスキル
	_TO_BUILD 変数の値をから network に設定しますcompute。 `TF_MODULE_TO_BUILD: "compute"`	

タスク	説明	必要なスキル

テナント管理 Helm チャートの values ファイルを更新します。	1. 次の場所で values.yaml ファイルを開きます。
	`cd cfg-terraform/demo/compute/cfg-tenant-mgmt`
	ファイルは次のようになります。
	`---`
	`global:`
	`  clusterRoles:`
	`  operator:`
	`  platform-tenant`
	`  flux: flux-tenant-applier`
	`  flux:`
	`  tenantCloneBaseUrl: ${TEANT_BASE_URL}`
	`  repoSecret:`
	`  ${TENANT_REPO_SECRET}`
	`tenants:`
	`  tenant-1:`
	`  quotas:`
	`  limits:`
	`  cpu: 1`
	`  memory: 1Gi`
	`  flux:`
	`  path: overlays/tenant-1`
	`  tenant-2:`
	`  quotas:`
	`  limits:`
	`  cpu: 1`
	`  memory: 2Gi`
	`  flux:`
		AWS DevOps

タスク	説明	必要なスキル

	<pre>path: overlays/
	tenant-2</pre>
	2. global および tenantsセクションで、要件に基づいて設定を更新します。
	• tenantCloneBaseUrl – すべてのテナントのコードをホストするリポジトリへのパス (すべてのテナントに同じ Git リポジトリを使用します)
	• repoSecret – グローバルテナント Git リポジトリに対して認証するための SSH キーと既知のホストを保持する Kubernetes シークレット
	• quotas – テナントごとに適用する Kubernetes リソースクォータ
	• flux path – グローバルテナントリポジトリ内のテナントアプリケーション YAML ファイルへのパス	

タスク	説明	必要なスキル
コンピューティングリソースを検証します。	前のステップでファイルを更新すると、CodePipeline が自動的に起動します。コンピューティングインフラストラクチャ用に次の AWS リソースが作成されていることを確認します。 • プライベートエンドポイントを持つ Amazon EKS クラスター • Amazon EKS ワーカーノード • Amazon EKS アドオン: 外部シークレット、`aws-loadbalancer-controller` 、および `metrics-server` • GitOps モジュール、Flux Helm チャート、Cilium Helm チャート、テナント管理 Helm チャート	AWS DevOps

テナント管理およびその他のリソースを確認する

タスク	説明	必要なスキル
Kubernetes のテナント管理リソースを検証します。	次のコマンドを実行して、Helm の助けを借りてテナント管理リソースが正常に作成されたことを確認します。 1. 「」で指定されているように、テナント名前空間が作	AWS DevOps

タスク	説明	必要なスキル

	成されましたvalues.yaml 。
	`kubectl get ns -A`
	2. クォータは、 で指定されているように、各テナント名前空間に割り当てられますvalues.yaml 。
	`kubectl get quota --namespace=<tenant_namespace>`
	3. 各テナント名前空間のクォータの詳細が正しい。
	`kubectl describe quota cpu-memory-resource-quota-limit -n <tenant_namespace>`
	4. Cilium ネットワークポリシーが各テナント名前空間に適用されました。
	`kubectl get CiliumNetworkPolicy -A`	

タスク	説明	必要なスキル

テナントアプリケーションのデプロイを確認します。	次のコマンドを実行して、テナントアプリケーションがデプロイされたことを確認します。
	
	1. Flux は、GitOps モジュールで指定された CodeCommit リポジトリに接続できます。
	
	`kubectl get gitrepositories -A`
	
	2. Flux kustomization コントローラーは、CodeCommit リポジトリに YAML ファイルをデプロイしました。
	
	`kubectl get kustomizations -A`
	
	3. すべてのアプリケーションリソースはテナント名前空間にデプロイされます。
	
	`kubectl get all -n <tenant_namespace>`
	
	4. イングレスはテナントごとに作成されています。
	
	`kubectl get ingress -n <tenant_namespace>`	

トラブルシューティング

問題	ソリューション

次のようなエラーメッセージが表示されます。
 	
`Failed to checkout and determine revision: unable to clone unknown error: You have successfully authenticated over SSH. You can use Git to interact with AWS CodeCommit.`
	問題のトラブルシューティングを行うには、次の手順に従います。
	
	1. テナントアプリケーションリポジトリを確認する: リポジトリが空または設定ミスのため、エラーが発生している可能性があります。テナントアプリケーションリポジトリに必要なコードが含まれていることを確認します。
	2. `tenant_mgmt` モジュールを再デプロイする: `tenant_mgmt` モジュール設定ファイルで appブロックを見つけ、 deployパラメータを に設定します`0`。
	
	`deploy = 0`
	
	Terraform apply コマンドを実行したら、deployパラメータ値を に戻します`1`。
	
	`deploy = 1`
	
	3. ステータスを再確認する: 前のステップを実行した後、次のコマンドを使用して問題が解決しないかどうかを確認します。
	
	`kubectl get gitrepositories -A`
	
	それでも問題が解決しない場合は、Flux ログを詳しく調べるか、Flux の一般的なトラブルシューティングガイドを参照してください。

関連リソース

- Terraform の Amazon EKS 設計図
- Amazon EKS ベストプラクティスガイド、マルチテナンシーセクション
- Flux ウェブサイト
- Helm ウェブサイト

追加情報

テナントアプリケーションをデプロイするためのリポジトリ構造の例を次に示します。

```
applications
sample_tenant_app
### README.md
### base
#   ### configmap.yaml
#   ### deployment.yaml
#   ### ingress.yaml
#   ### kustomization.yaml
#   ### service.yaml
### overlays
    ### tenant-1
    #   ### configmap.yaml
    #   ### deployment.yaml
    #   ### kustomization.yaml
    ### tenant-2
        ### configmap.yaml
        ### kustomization.yaml
```

カスタムリソースを使用して複数の E メールエンドポイントを SNS トピックにサブスクライブする

作成者: Ricardo Morais (AWS)

概要

注意、2022 年 8 月: AWS CloudFormation は、:: SNS:: Topic オブジェクトとそのサブスクリプション属性を介して複数のリソースのサブスクリプションをサポートするようになりました。

このパターンでは、Amazon Simple Notification Service (Amazon SNS) トピックから通知を受け取るための、複数の E メールアドレスをサブスクライブする方法について示しています。ここでは、AWS Lambda 関数を AWS CloudFormation テンプレートのカスタムリソースとして使用します。Lambda 関数は、SNS トピックの E メールエンドポイントを指定する入力パラメータに関連付けられています。

現在は、AWS CloudFormation テンプレートオブジェクト ::SNS::Topic と :: SNS:: Subscription を使用して、単一のエンドポイントを SNS トピックにサブスクライブが可能です。複数のエンドポイントをサブスクライブするには、オブジェクトを複数回呼び出す必要があります。Lambda 関数をカスタムリソースとして使用することで、入力パラメータを通じて複数のエンドポイントをサブスクライブできます。この Lambda 関数は、任意の AWS CloudFormation テンプレートのカスタムリソースとして使用できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- ローカル環境で、アクセスキーとシークレットキーを使用して構成された AWS プロファイル。
- 次に対するアクセス許可を付与します。
  - AWS Identity and Access Management (IAM) ロールとポリシー
  - AWS Lambda 関数
  - Lambda 関数をアップロードするための、Amazon Simple Storage Service (Amazon S3)
  - Amazon SNS トピックとポリシー
  - AWS CloudFormation スタック

制約事項

- このコードは Linux および macOS ワークステーションをサポートします。

製品バージョン

- AWS コマンドラインインターフェイス (AWS CLI)バージョン 2 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation
- Amazon SNS
- AWS Lambda

ツール

ツール

- AWS CLI バージョン 2

コード

添付ファイルには以下のファイルが含まれます。

- Lambda 関数: `lambda_function.py`
- AWS CloudFormation テンプレート: `template.yaml`
- 複数または単一の E メールエンドポイントサブスクリプションを処理するための 2 つのパラメータファイル: `parameters-multiple-values.json` (デフォルトとして使用) および `parameters-one-value.json`

スタックのデプロイには、どちらのパラメータファイルを使用してもかまいません。複数の E メールエンドポイントを指定するには:

```
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION>
```

1 つの E メールエンドポイントを指定するには:

```
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION> -f parameters-one-value.json
```

エピック

オプション 1 - 1 つの E メールサブスクリプションで SNS トピックをデプロイする

タスク	説明	必要なスキル
SNS トピックサブスクリプション用の E メールエンドポイントを構成する。	ファイル `parameters-one-value.json` (添付) を編集し、使用する E メールアドレス (`someone@example.com` など) を反映するように `pSNSNotificationsEmail` パラメータの値を変更します。	
リソースとサブスクリプションを作成する AWS CloudFormation スタックをデプロイします。	AWS プロファイル名、AWS リージョン、および `parameters-one-value.json` ファイルを使用して deploy.sh コマンドを実行します。 `./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION> -f parameters-one-value.json`	適切なアクセス許可を持つ IAM ロール

オプション 2 - 2 つ以上の E メールサブスクリプションを含む SNS トピックをデプロイする

タスク	説明	必要なスキル
SNS トピックサブスクリプション用の E メールエンドポイントを構成する。	ファイル `parameters-multiple-values.json` (添付) を編集し、使用するメールアドレスを反映するように `pSNSNotificationsEmail` パラメータの値を次のようにカンマで区切って変更します: `someone1@example.com,someone2@example.com`	
リソースとサブスクリプションを作成する AWS CloudFormation スタックをデプロイします。	AWS プロファイル名と AWS リージョンを使用して deploy.sh コマンドを実行します。`parameters-multiple-values.json` ファイルはデフォルトで使用されるため、指定する必要はありません。 `./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION>`	適切なアクセス許可を持つ IAM ロール

オプション 3 - AWS CloudFormation テンプレートを使用して SNS トピックをデプロイする

タスク	説明	必要なスキル
SNS トピックを作成します。	`AWS::SNS::Topic` テンプレートオブジェクトでサブスクリプションエンドポイントを指定せずに、AWS	適切なアクセス許可を持つ IAM ロール

タスク	説明	必要なスキル
	CloudFormation テンプレートを使用して SNS トピックを作成します。添付ファイルにある `template.yaml` を出発点として使用できます。	
SNS トピックポリシーを作成します。	AWS CloudFormation テンプレートに SNS トピックポリシーを作成します。	適切なアクセス許可を持つ IAM ロール
E メールエンドポイントリストを SNS トピックにサブスクライブする。	E メールエンドポイントのリスト (1 つ以上) に基づいて、作成した SNS トピックにエンドポイントをサブスクライブします。	適切なアクセス許可を持つ IAM ロール

関連リソース

リファレンス

- AWS CloudFormation カスタムリソース (AWS ドキュメント)
- Python、 Lambda、およびクレパーによる AWS CloudFormation カスタムリソース作成 (ブログ記事)

必要なツール

- AWS CLI バージョン 2

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Fargate WaitCondition フックコンストラクトを使用してリソースの依存関係とタスク実行を調整する

作成者: Stan Fan (AWS)

概要

このパターンでは、Amazon Elastic Container Service (Amazon ECS`waitcondition-hook-for-aws-fargate-task`) クラスターでAWS Fargateタスクをオーケストレーションするために設計されたクラウドネイティブソリューションである WaitCondition フック () npm パッケージについて説明します。

WaitCondition フックは、 との統合用に特別にカスタマイズされた AWS Cloud Development Kit (AWS CDK) コンストラクトです AWS CloudFormation。WaitCondition フックには、次の主要な機能があります。

- 待機条件メカニズムとして機能し、指定された Fargate タスクが完了するまで CloudFormation スタックの実行を一時停止します。これにより、正常なデプロイとリソースプロビジョニングに役立ちます。
- TypeScript と Python をサポートしているため、 AWS CDK プロジェクトに最適です。
- 開発者とアーキテクトは、コンテナ化されたアプリケーションのタスク完了とリソース管理を調整することで、デプロイをオーケストレーションできます AWS。
- CloudFormation ライフサイクルに 1 つ以上のコンテナが埋め込まれた Fargate タスクの実行を有効にします。 と は、タスクの失敗を処理し、タスクの失敗後に CloudFormation スタックをロールバックできます。
- リソースと Fargate タスク実行結果の間に依存関係を追加する柔軟性を提供し、カスタムタスクを有効にしたり、他のエンドポイントを呼び出したりします。例えば、CloudFormation スタックを一時停止し、データベース移行 (Fargate タスクによって実行) を待機し、データベース移行の成功に依存する可能性のある他のリソースをプロビジョニングできます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。

- AWS Cloud Development Kit (AWS CDK) ローカルワークステーションにインストールされたコマンドラインインターフェイス (CLI)。詳細については、 AWS CDK ドキュメントの AWS CDK CLI リファレンスを参照してください。
- ノードパッケージマネージャー (npm)。ローカルワークステーションにインストールされ、AWS CDK TypeScript の 用に設定されています。詳細については、npm ドキュメントの [Node.js と npm のダウンロードとインストール] を参照してください。
- ローカルワークステーションにインストールされた Yarn。詳細については、Yarn ドキュメントの「Installation」を参照してください。

制約事項

- このソリューションは 1 つの にデプロイされます AWS アカウント。
- コンテナの想定されるリターンコードは成功0です。その他のリターンコードは失敗を示し、CloudFormation スタックはロールバックされます。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

アーキテクチャ

次の図は、コンストラクトアーキテクチャを示しています。

この図は、 のワークフローを示していますwaitcondition-hook-for-aws-fargate-task。

1. WaitCondition および WaitConditionHandler は、 AWS Lambda 関数からのレスポンスをリッスンするようにプロビジョニングされます。
2. タスクの結果に応じて、 CallbackFunctionまたは のいずれかErrorHandlerFunctionが Fargate タスクの終了によってトリガーされます。
3. Lambda 関数は、SUCCEED または FAILURE シグナルを に送信しますWaitConditionHandler。
4. WaitConditionHandler Fargate タスクの実行結果が成功した場合は がリソースをプロビジョニングし続けます。タスクが失敗した場合はスタックをロールバックします。

次の図は、データベース移行を実行するワークフローの例を示しています。

このワークフロー例では、次のように `waitcondition-hook-for-aws-fargate-task`コンストラクトを使用してデータベース移行を実行します。

1. Amazon Relational Database Service (Amazon RDS) インスタンスがプロビジョニングされます。
2. `waitcondition-hook-for-aws-fargate-task` コンストラクトはデータベース移行タスクを実行し、スタックを Amazon Elastic Compute Cloud (Amazon EC2) インスタンスとして一時停止します。
3. 移行タスクが正常に終了すると、成功シグナルが CloudFormation に送信されます。それ以外の場合は、Fail シグナルを CloudFormation に送信し、スタックをロールバックします。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、それを通じてプロビジョニングするのに役立つソフトウェア開発フレームワークです AWS CloudFormation。
- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、 AWS アカウント および 全体のライフサイクル全体の管理に役立ちます AWS リージョン。
- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- AWS Fargate を使用すると、サーバーや Amazon EC2 インスタンスを管理することなくコンテナを実行できます。これは Amazon ECS と組み合わせて使用されます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- AWS Step Functions は、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、独自のデータセンターで運用する従来のネットワークに似ていますが、スケーラブルなインフラストラクチャを使用する利点がありますAWS。

その他のツール

- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。
- Yarn は、JavaScript プロジェクトの依存関係を管理するために使用できるオープンソースのパッケージマネージャーです。Yarn は、パッケージの依存関係のインストール、更新、設定、削除に役立ちます。

コードリポジトリ

このパターンのコードは、GitHub waitcondition-hook-for-aws-fargate-task リポジトリで入手できます。

ベストプラクティス

- AWS CDK アプリを構築するときは、v2 AWS CDK ドキュメントの「 を使用したクラウドインフラストラクチャの開発とデプロイのベストプラクティス AWS CDK」に従ってください。
- AWS Fargate タスクについては、Amazon ECS ドキュメントの「Amazon ECS コンテナイメージのベストプラクティス」に従ってください。

エピック

のセットアップ AWS CDK

タスク	説明	必要なスキル
をインストールします AWS CDK。	AWS CDK ローカルマシンまたは他の環境に をインストー	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	ルするには、次のコマンドを実行します。 `npm install -g aws-cdk@latest`	
をブートストラップします AWS CDK。	ブートストラップは、デプロイする環境を準備するプロセスです。ターゲット AWS アカウント と の AWS CDK ツールキットをブートストラップするには AWS リージョン、次のコマンドを実行します。 `cdk bootstrap aws://ACCOUNT-NUMBER-1/REGION-1` このコマンドは、 という名前の CloudFormation スタックを作成しますCDKToolkit 。	クラウドアーキテクト

AWS Fargate タスクコンストラクトの WaitCondition フックを実行する

タスク	説明	必要なスキル
CDK プロジェクトを作成します。	必要な言語を使用して CDK プロジェクトを作成します。このパターンでは TypeScript を使用します。TypeScript を使用して CDK プロジェクトを作成するには、次のコマンドを実行します。	クラウドアーキテクト

タスク	説明	必要なスキル
	`cdk init app –language typescript`	
パッケージをインストールします。	CDK プロジェクトのルートパスで `npm install` を実行します。CDK ライブラリをインストールしたら、次のコマンドを実行して `waitcondition-hook-for-aws-fargate-task` をインストールします。 `yarn add waitcondition-hook-for-aws-fargate-task`	クラウドアーキテクト

タスク	説明	必要なスキル

CDK アプリケーションと Amazon ECS コンポーネントを構築します。	CDK プロジェクトを構築します。Amazon ECS タスク定義リソースが必要です。タスク定義の作成については、Amazon ECS ドキュメントの「Amazon ECS タスク定義」を参照してください。
	
	次の例では、このコンストラクトを使用します。
	
	`import * as cdk from 'aws-cdk-lib';`
	`import { Vpc } from 'aws-cdk-lib/aws-ec2';`
	`import * as ecr from 'aws-cdk-lib/aws-ecr';`
	`import * as ecs from 'aws-cdk-lib/aws-ecs';`
	`import { Construct } from 'constructs';`
	`import { FargateRunner } from 'waitcondition-hook-for-aws-fargate-task';`
	`import { Queue } from 'aws-cdk-lib/aws-sqs';`
	
	`export class FargateRunnerStack extends cdk.Stack {`
	`constructor(scope: Construct, id: string, props?: cdk.StackProps) {`
		クラウドアーキテクト

タスク	説明	必要なスキル

	super(scope, id, props);
	// Define the VPC
	const vpc = new Vpc(this, 'MyVpc')
	// Define the Fargate Task
	const taskDefinition = new ecs.FargateTaskDefinition(this, 'MyTask', {});
	// Import exiting ecr repo
	const repo = ecr.Repository.fromRepositoryName(this, 'MyRepo', 'RepoName');
	// Add a container to the task
	taskDefinition.addContainer('MyContainer', {
	image: ecs.ContainerImage.fromEcrRepository(repo),
	});
	// Create the Fargate runner
	const myFargateRunner = new FargateRunner(this, 'MyRunner', {
	fargateTaskDef: taskDefinition,
	timeout: `${60 * 5}`,
	vpc: vpc,
	});
	// Create the SQS queue	

タスク	説明	必要なスキル
	`const myQueue = new Queue(this, 'MyQueue', {}); // Add dependency myQueue.node.addDependency(myFargateRunner); } }`	

CDK アプリケーションを同期して起動します。	1. アセットと CloudFormation テンプレートを生成するには、CDK ルートパスで次のコマンドを実行します。
	`cdk synth`
	2. synth コマンドが成功したら、次のコマンドを実行してリソースをデプロイします。
	`cdk deploy`
	
	`waitcondition-hook-for-aws-fargate-task` コンストラクトは Fargate タスクを実行します。
		クラウドアーキテクト

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップします。	前のステップからプロビジョニングされたリソースをク	クラウドアーキテクト

タスク	説明	必要なスキル
	リーンアップするには、次のコマンドを実行します。 `cdk destroy`	

トラブルシューティング

問題	ソリューション
CloudFormation スタックの一般的な障害	一般的な CloudFormation スタック障害のトラブルシューティングに役立つように、次の例に示すように `--no-rollback` フラグを追加します。 `cdk deploy --no-rollback` このコマンドは CloudFormation スタックのロールバックを一時停止し、トラブルシューティングするリソースを提供します。詳細については、 AWS CloudFormation ドキュメントの「リソースをプロビジョニングするときに障害を処理する方法を選択する」を参照してください。
AWS Step Functions 失敗	AWS Step Functions ステートマシンは、さまざまな理由で実行に失敗することがあります。`–disable-rollback` が設定されている場合は、次の手順を使用してトラブルシューティングを行います。 1. にサインインし AWS Management Console、検索フィールドに Step Functions と入力し、Step Functions サービスを選択します。

問題	ソリューション
	2. 左側のナビゲーションペインで、ステートマシンを選択し、CloudFormation スタックによってプロビジョニングされるステートマシンを選択します。 3. 実行 で、予期せず失敗した実行の名前を選択します。 4. イベントビューで、失敗したステップを選択します。 詳細については、 AWS Step Functions ドキュメントの「Step Functions の問題のトラブルシューティング」および「Step Functions コンソールでの実行の詳細の表示」を参照してください。

問題	ソリューション
AWS Lambda 関数の失敗	このコンストラクトは、 `CallbackFunction` と の 2 つの Lambda 関数をプロビジョニングします`ErrorhandlerFunction` 。処理されていない例外など、さまざまな理由で失敗する可能性があります。トラブルシューティングを行うには、次の手順を実行します。 1. にサインインし AWS Management Console、検索フィールドに CloudWatch と入力し、CloudWatch サービスを選択します。 2. 左側のナビゲーションペインで、[ロググループ] をクリックします。 3. 検索フィールドに、Lambda 関数の名前を入力します。 4. Lambda 関数に関連付けられているロググループ名を選択します。 5. Lambda 関数の実行結果に移動するには、最新のログストリームを選択します。 詳細については、 AWS Lambda ドキュメントの「Lambda の問題のトラブルシューティング」を参照してください。

関連リソース

AWS ドキュメント

- AWS CDK コンストラクト API リファレンス
- の開始方法 AWS CDK
- Amazon ECS リソースを作成して使用する方法について説明します。
- Step Functions の使用を開始する方法について説明します。

- とは AWS CDK

その他のリソース

- AWS Fargate タスクの待機条件フック (npm)
- waitcondition-hook-for-aws-fargate-task 1.0.6 (pypi.org://www.)

AWS CodePipeline のサードパーティの Git ソースリポジトリを使用する

作成者: Kirankumar Chandrashekar (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンでは、AWS CodePipeline をサードパーティーの Git ソースリポジトリで使用する方法を説明します。

AWS CodePipeline は、ソフトウェアの構築、テスト、製品へのデプロイを自動化する 継続的デリバリーサービス です。このサービスは現在、GitGitHub、「AWS CodeCommit」、およびアトラシアン Bitbucket によって管理されている Git リポジトリをサポートしています。ただし、一部の企業では、シングルサインオン (SSO) サービスとMicrosoft Active Directoryと統合されたサードパーティ製の Git リポジトリを認証に使用しています。カスタムアクションと Webhook を作成することで、これらのサードパーティの Git リポジトリを CodePipeline のソースとして使用できます。

ウェブフックとは、GitHub リポジトリなど、別のツールにおけるイベントを検出し、この外部イベントをパイプラインに接続する HTTP 通知です。CodePipeline でウェブフックを作成すると、サービスは Git リポジトリのウェブフックで使用できる URL を返します。Git リポジトリの特定のブランチにコードをプッシュすると、Git ウェブフックはこの URL から CodePipeline ウェブフックを開始し、パイプラインのソースステージを「進行中」に設定します。パイプラインがこの状態になると、ジョブワーカーは CodePipeline をポーリングしてカスタムジョブを探し、ジョブを実行し、成功または失敗のステータスを CodePipeline に送信します。この場合、パイプラインはソースステージにあるため、ジョブワーカーは Git リポジトリのコンテンツを取得して圧縮し、ポーリングされたジョブによって提供されたオブジェクトキーを使用して、パイプラインのアーティファクトが保存されている Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。カスタムアクションのトランジションを Amazon CloudWatch のイベントに関連付けて、そのイベントに基づいてジョブワーカーを起動することもできます。この設定により、サービスがネイティブにサポートしていないサードパーティの Git リポジトリを CodePipeline のソースとして使用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ウェブフックをサポートし、インターネット経由で CodePipeline ウェブフック URL に接続できる Git リポジトリ

- AWS コマンドラインインターフェイス (AWS CLI) が「インストールされ」、AWS アカウントと連携するように「設定されている」

アーキテクチャ

このパターンには以下のステップが含まれます。

1. ユーザーは Git リポジトリにコードをコミットします。
2. Git ウェブフックが呼び出されます。
3. CodePipeline ウェブフックが呼び出されます。
4. パイプラインは「進行中」に設定され、ソースステージは「進行中」状態に設定されます。
5. ソースステージアクションは、開始されたことを示す CloudWatch Events ルールを開始します。
6. CloudWatch イベントは Lambda 関数を開始します。
7. Lambda 関数はカスタムアクションジョブの詳細を取得します。
8. Lambda 関数は AWS CodeBuild を起動し、ジョブに関連するすべての情報を渡します。
9. CodeBuild は、HTTPS Git アクセス用の公開 SSH キーまたはユーザー認証情報をSecrets Manager から取得します。
10. CodeBuild は特定のブランチの Git リポジトリをクローンします。
11. CodeBuild はアーカイブを圧縮し、CodePipeline アーティファクトストアとして機能する S3 バケットにアップロードします。

ツール

- AWS CodePipeline – AWS CodePipeline はフルマネージド型の継続的デリバリーサービスで、アプリケーションとインフラストラクチャの更新を迅速かつ信頼できるものにするパイプラインのリリース自動化に役立ちます。CodePipeline はお客様が定義したリリースモデルに基づき、コードチェンジがあった場合のフェーズの構築、テスト、およびデプロイを自動化します。機能とアップデートをすばやく、信頼性の高い方法で配信できます。AWS CodePipeline は、GitHub などのサードパーティサービスや独自のカスタムプラグインと統合できます。
- AWS Lambda – AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。Lambda を使用すれば、実質どのようなタイプのアプリケーションやバックエンドサービスでも、管理をまったく必要とせずに実行できます。コードをアップロードするだ

けで、コードの実行とスケールに必要な処理はすべて Lambda により自動的に実行され、高い可用性が維持されます。コードは、他の AWS サービスから自動的に開始するよう設定することも、ウェブやモバイルアプリケーションから直接呼び出すよう設定することもできます。

- AWS CodeBuild – AWS CodeBuild はフルマネージド型の継続的統合サービスであり、このサービスにより、ソースコードをコンパイルし、テストを実行し、デプロイ可能なソフトウェアパッケージを作成できます。CodeBuild により、ビルドサーバーのプロビジョニング、管理、スケーリングが不要になります。CodeBuild では、継続的にスケーリングされ、複数のビルドが同時にプロセスされるため、ビルドの実行までキューで待機することはありません。パッケージ済みのビルド環境を使用、またはご自分のビルドツールを使用するカスタムビルド環境を作成できることですぐに開始できます。
- AWS Secrets Manager – AWS Secrets Manager は、アプリケーション、サービス、IT リソースへのアクセスに必要なシークレットの保護に役立ちます。このサービスを使用すると、データベースクレデンシャル、APIキー、およびその他のシークレットをライフサイクル全体で簡単にローテーション、管理、および取得できます。ユーザーとアプリケーションは、機密情報をプレーンテキストにハードコーディングしなくても、Secrets Manager API を呼び出すことでシークレットを取得します。Secrets Manager では、Amazon Relational Database Service (Amazon RDS)、Amazon Redshift、Amazon DocumentDB の統合が組み込まれています。このサービスは、API キーや OAuth トークンなど、他のタイプのシークレットをサポートするように拡張できます。さらに、Secrets Manager では、きめ細かい権限を使用してシークレットへのアクセスを制御したり、AWS クラウド、サードパーティサービス、オンプレミス環境内のリソースに対するシークレットのローテーションを一元的に監査したりできます。
- Amazon CloudWatch – Amazon CloudWatch は、DevOps エンジニア、開発者、サイト信頼性エンジニア (SRE)、ITマネージャー、アプリケーション所有者向けに構築された、モニタリングおよびオブザーバビリティサービスです。CloudWatch では、データと実用的なインサイトを提供して、アプリケーションのモニタリング、システム全体のパフォーマンスの変化に関する対応、リソース使用率の最適化、および運用状態の統合的な確認を行うことができます。CloudWatch は、ログ、メトリックス、イベントの形式でモニタリングデータと運用データを収集し、AWS とオンプレミスサーバーで実行される AWS のリソース、アプリケーション、サービスを一元的に把握できるようにします。CloudWatch を使用すると、環境内の異常な動作の検出、アラームの設定、ログとメトリクスの並列表示、自動アクションの実行、問題のトラブルシューティング、およびアプリケーションの円滑な稼働を維持するための洞察の発見を行うことができます。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、ウェブサイト、モバイルアプリケーション、バックアップおよび復元、アーカイブ、エンタープライズアプリケーション、IoT デバイス、ビッグデータ分析など、広範なユースケースのデータを容量にかかわらず、保存して保護することができます。Amazon S3 には、特定のビジネス、組織、コンプライアンスの要件を満たすために、データへのアクセスを最適化、整理、設定できる管理機能があります。

エピック

CodePipeline でカスタムアクションを作成および追加する

タスク	説明	必要なスキル
AWS CLI または AWS CloudFormation を使用してカスタムアクションを作成します。	このステップでは、特定のリージョンの AWS アカウントのパイプラインのソースステージで使用できるカスタムソースアクションを作成します。カスタムソースアクションを作成するには、AWS CLI または AWS CloudFormation (コンソールではない) を使用する必要があります。このエピックやその他のエピックで説明されているコマンドとステップの詳細については、このパターンの最後にある「関連リソース」セクションを参照してください。AWS CLI では、カスタムアクションタイプの作成コマンドを使用します。--configuration-properties を使用して、ジョブワーカーがジョブの CodePipeline をポーリングするときに処理する必要があるすべてのパラメーターを指定します。このカスタムソースステージでパイプラインを作成するときに同じ値を使用できるように、--provider オプションと--action-version オプションに指定されている値を必ず書き留めてください。リソースタイプ	AWS 全般

タスク	説明	必要なスキル
	AWS：：CodePipeline：：CustomActionType を使用して AWS CloudFormation でカスタムソースアクションを作成することもできます。	

認証を設定する。

タスク	説明	必要なスキル
SSH キーペア を作成します。	Secure Shell (SSH) key pair を作成します。手順については、Github のドキュメントを参照してください。	システム/DevOps エンジニア
AWS Secrets Manager でシークレットを作成します。	SSH キーkey pair からプライベートキーの内容をコピーし、AWS Secrets Manager でシークレットを作成します。このシークレットは、Git リポジトリにアクセスするときの認証に使用されます。	AWS 全般
パブリックキーを Git リポジトリに追加します。	SSH キーペアのパブリックキーを Git リポジトリアカウント設定に追加して、プライベートキーに対する認証を行います。	システム/DevOps エンジニア

パイプラインと Webhook を作成します。

タスク	説明	必要なスキル
カスタムソースアクションを含むパイプラインを作成します。	CodePipeline でパイプラインを作成します。ソースステージを設定するときは、以前に作成したカスタムソースアクションを選択します。AWS CodePipeline コンソールまたは AWS CLI でこれを行うことができます。CodePipeline は、カスタムアクションに設定した設定プロパティの入力を求めます。この情報は、ジョブワーカーがカスタムアクションのジョブを処理するために必要です。ウィザードに従い、パイプラインの次のステージを作成します。	AWS 全般
CodePipeline Web フックを作成します。	カスタムソースアクションで作成したパイプライン用の Webhook を作成します。ウェブフックを作成するには、AWS CLI または AWS CloudFormation (コンソールではない) を使用する必要があります。AWS CLI で put-webhook コマンドを実行し、ウェブフックオプションに適切な値を指定します。コマンドから返される Webhook URL を書き留めておきます。AWS CloudFormation を使用してウェブフックを作成する場合は、リソースタイプ	AWS 全般

タスク	説明	必要なスキル
	AWS：：CodePipeline：：Webhook を使用してください。作成したリソースからウェブフック URL を出力し、書き留めておいてください。	

タスク	説明	必要なスキル
Lambda 関数と CodeBuild プロジェクトを作成します。	カスタムアクションを作成する際、このカスタムアクションのジョブリクエストの CodePipeline をポーリングするジョブワーカーを作成し、ジョブを実行してステータス結果を CodePipeline に返す必要があります。 パイプラインのカスタムソースアクションステージが「進行中」に移行したときに Amazon CloudWatch Events ルールによって開始される Lambda 関数を作成します。Lambda 関数が開始されると、ジョブをポーリングしてカスタムアクションジョブの詳細を取得する必要があります。PollForJobs API を使用してこの情報を返すことができます。ポーリングされたジョブ情報を取得したら、Lambda 関数は確認を返し、カスタムアクションの設定プロパティから取得したデータを使用して情報を処理する必要があります。ワーカーが Git リポジトリと通信する準備ができたら、CodeBuild プロジェクトを開始します。SSH クライアントを使用して Git タスクを処理する方が便利なためです。	AWS 全般、コード開発者

CloudWatch でイベントを作成する

タスク	説明	必要なスキル
CloudWatch Events ルールを作成します。	パイプラインのカスタムアクションステージが「進行中」に移行するたびに、ターゲットとして Lambda 関数を開始する CloudWatch Events ルールを作成します。	AWS 全般

関連リソース

CodePipeline でカスタムアクションにタグ付けする

- CodePipeline でカスタムアクションを作成および追加する
- AWS：：CodePipeline：：CustomActionType

認証の設定

- AWS Secrets Manager でのシークレットの作成と管理

パイプラインと Web フックの作成

- CodePipeline でパイプラインの作成
- put-webhook コマンドリファレンス
- AWS：：CodePipeline：：Webhook
- PollForJobs API リファレンス
- CodePipeline でカスタムアクションを作成および追加する
- AWS CodeBuild でビルドプロジェクトを削除する

イベントの作成

- Amazon CloudWatch Eventsでパイプラインステートの変更を検知し反応する

その他の参考資料

- CodePipeline でのパイプラインの使用
- AWS Lambda 開発者ガイド

AWS CodePipeline を使用して Terraform の構成を検証するための CI/CD パイプラインを作成する

作成者: Aromal Raj Jayarajan (AWS) と Vijesh Vijayakumaran Nair (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、AWS CodePipeline がデプロイした継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインを使用して HashiCorp Terraform の構成をテストする方法を説明しています。

Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理する上で役立つコマンドラインインターフェイスアプリケーションです。このパターンで提供されるソリューションは、CodePipeline の 5 つのステージを実行して Terraform 構成の整合性を検証する際に役立つ CI/CD パイプラインを作成します。

1. “checkout” は、テスト中の Terraform 構成を AWS CodeCommit リポジトリから取得します。
2. “validate” は、tfsec、TFLint、checkov などの infrastructure as code (IaC) 検証ツールを実行します。このステージでは、次の Terraform IaC 検証コマンドも実行されます: `terraform validate` および `terraform fmt`
3. “plan” は、Terraform 構成が適用された場合にインフラストラクチャにどのような変更が適用されるかを示します。
4. “apply” は、生成された計画を使用して、必要なインフラストラクチャをテスト環境にプロビジョニングします。
5. “destroy” は、“apply” ステージ中に作成されたテストインフラストラクチャを削除します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- インストールされ、構成済みの AWS コマンドラインインターフェイス (AWS CLI)
- ローカルマシンにインストールされて構成されている Git
- ローカルマシンにインストールされ、構成済みの Terraform

制約事項

- このパターンのアプローチでは、AWS CodePipeline を 1 つの AWS アカウントと AWS リージョンのみにデプロイします。マルチアカウントおよびマルチリージョンデプロイには、構成の変更が必要です。
- このパターンでプロビジョニングされる AWS Identity and Access Management (IAM) ロール (codepipeline_iam_role) は、最小特権の原則に従います。IAM ロールの権限は、パイプラインが作成する必要のある特定のリソースに基づいて更新する必要があります。

製品バージョン

- AWS CLI バージョン 2.9.15 以降
- Terraform バージョン 1.3.7 以降:

アーキテクチャ

ターゲットテクノロジースタック

- CodePipeline
- CodeBuild
- AWS CodeCommit
- AWS IAM
- Amazon Simple Storage Service (Amazon S3)
- AWS Key Management Service (AWS KMS)
- Terraform

ターゲットアーキテクチャ

次の図は、CodePipeline で Terraform 構成をテストするための CI/CD パイプラインワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. CodePipeline では、AWS ユーザーは AWS CLI で `terraform apply` コマンドを実行して Terraform プランで提案されているアクションを開始します。

2. AWS CodePipeline は、CodeCommit、CodeBuild、AWS KMS、Amazon S3 へのアクセスに必要なポリシーを含む IAM サービスロールを引き受けます。
3. CodePipelineは、“checkout” パイプラインステージを実行して、テスト用に AWS CodeCommit リポジトリから Terraform 構成を取得します。
4. CodePipeline は、CodeBuild プロジェクトで IaC 検証ツールを実行し、Terraform IaC 検証コマンドを実行して Terraform 構成をテストする “validate” ステージを実行します。
5. CodePipeline は “plan” ステージを実行し、Terraform 構成に基づいて CodeBuild プロジェクトでプランを作成します。AWS ユーザーは、変更をテスト環境に適用する前にこのプランを確認できます。
6. Code Pipeline は、CodeBuild プロジェクトを使用してテスト環境に必要なインフラストラクチャをプロビジョニングすることで、プランを実装する “apply” ステージを実行します。
7. CodePipeline は、“destroy” ステージを実行することで、“apply” ステージ中に作成されたテストインフラストラクチャを CodeBuild で削除します。
8. Amazon S3 バケットには、AWS KMS カスタマーマネージドキーを使用して暗号化および復号化されたパイプラインアーティファクトが格納されます。

ツール

ツール

サービス

- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS CodeBuild はフルマネージドの構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意のデータ量を保存、保護、取得する際に役立つクラウドベースのオブジェクトストレージサービスです。

その他のサービス

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するのに役立つコマンドラインインターフェイスアプリケーションです。

コード

このパターンのコードは、GitHub 内の「aws-codepipeline-terraform-cicdsamples」リポジトリで利用できます。リポジトリには、このパターンで概説されているターゲットアーキテクチャの作成に必要な Terraform 構成が含まれています。

エピック

ソリューションコンポーネントのプロビジョニング

タスク	説明	必要なスキル
GitHub リポジトリのクローン作成	ターミナルウィンドウで以下のコマンドを実行して GitHub の aws-codepipeline-terraform-cicdsamples リポジトリをクローン作成します。 `git clone https://github.com/aws-samples/aws-codepipeline-terraform-cicd-samples.git` 詳細については、GitHub ドキュメントの「リポジトリのクローン作成」を参照してください。	DevOps エンジニア
Terraform 変数定義ファイルを作成する。	ユースケース要件に基づいて `terraform.tfvars` ファイルを作成します。クローン作成したリポジトリにある `examples/terraform`	DevOps エンジニア

タスク	説明	必要なスキル
	`.tfvars` ファイル内の変数を更新できます。 詳細については、Terraform ドキュメントの「ルートモジュール変数への値の割り当て」を参照してください。 **Note** リポジトリの `Readme.md` ファイルには、必要な変数に関する詳細情報が含まれています。	
AWS を Terraform プロバイダーとして構成する。	1. コードエディタで、クローン作成したリポジトリの `main.tf` ファイルを開きます。 2. ターゲット AWS アカウントへの接続を確立するために必要な構成を追加します。 詳細については、 のプロバイダーを参照してください。	DevOps エンジニア

タスク	説明	必要なスキル
Amazon S3 レプリケーションバケットを作成するための Terraform プロバイダー構成を更新する。	1. 次のコマンドを実行して、リポジトリの S3 ディレクトリを開きます。 `cd ./modules/s3` 2. `tf` ファイル内の `region` 値を更新して、Amazon S3 レプリケーションバケットを作成するための Terraform プロバイダー構成を更新します。Amazon S3 がオブジェクトをクローンするリージョンを必ず入力してください。 3. (オプション) デフォルトでは、Terraform はローカル状態のファイルを使用して状態管理を行います。Amazon S3 をリモートバックエンドとして追加する場合は、Terraform 構成を更新する必要があります。詳細については、Terraform ドキュメントの「バックエンドの構成」を参照してください。 **Note** レプリケーションは、Amazon S3 バケット間でオブジェクトの	DevOps エンジニア

タスク	説明	必要なスキル
	自動非同期コピーを有効にします。	
Terraform 構成を初期化します。	Terraform 構成ファイルを含む作業ディレクトリを初期化するには、クローン作成したリポジトリのルートフォルダで以下のコマンドを実行します。 `terraform init`	DevOps エンジニア
Terraform プランを作成します。	Terraform プランを作成するには、クローン作成したリポジトリのルートフォルダで以下のコマンドを実行します。 `terraform plan --var-file=terraform.tfvars -out=tfplan` **Note** Terraform は設定ファイルを評価して、宣言されたリソースのターゲット状態を判断します。次に、ターゲットの状態を現在の状態と比較し、プランを作成します。	DevOps エンジニア

タスク	説明	必要なスキル
Terraform プランを検証します。	Terraform プランを確認し、ターゲット AWS アカウントで必要なアーキテクチャが構成されていることを確認します。	DevOps エンジニア
ソリューションをデプロイします。	1. Terraform プランを適用するには、クローン作成したリポジトリのルートフォルダで以下のコマンドを実行します。 `terraform apply "tfplan"` 2. 「yes」と入力して、リソースをデプロイすることを確認します。 **Note** Terraform は、構成ファイルに宣言されている目標状態を達成するために、インフラストラクチャを作成、更新、または破棄します。	DevOps エンジニア

パイプラインを実行して Terraform の構成を検証します。

タスク	説明	必要なスキル
ソースコードリポジトリをセットアップします。	1. Terraform の出力から、検証したい Terraform 構成を含むリポジトリのソースリポジトリの詳細を取得します。 2. AWS マネジメントコンソールにサインインします。次に、CodeCommit コンソールを開きます。 3. `main` という名前のソースリポジトリに、新しいブランチを作成します。手順については、CodeCommit ドキュメントの「CodeCommit にブランチを作成する」を参照してください。 4. ソースリポジトリの `main` ブランチをローカルワークステーションにクローン作成します。方法については、CodeCommit ドキュメントの「CLI 認証情報ヘルパーを使用した、Windows 上の CodeCommit リポジトリへの HTTPS 接続のセットアップ手順」を参照してください。 5. 次のコマンドを実行して、GitHub の aws-codepipeline-terraform-cicdsamples リポジトリか	DevOps エンジニア

タスク	説明	必要なスキル
	らtemplates フォルダをコピーします。 `cp -r templates $YOUR_CODECOMMIT_REPO_ROOT` **Note** templates フォルダには、ソースリポジトリのルートディレクトリのビルド仕様ファイルと検証スクリプトが含まれています。 6. 必要な Terraform IaC 構成をソースリポジトリのルートフォルダに追加します。 7. リモートバックエンドの詳細をプロジェクトの Terraform 構成に追加します。詳細については、「Terraform ドキュメント」の S3 を参照してください。 8. (オプション) templates フォルダ内の変数を更新して、構成済みのスキャンやツール変更バージョンを有効または無効にしたり、カスタムスクリプトファイルでディレクトリを指定したりします。詳細について	

タスク	説明	必要なスキル
	は、このパターンの「追加情報」セクションを参照してください。 9. 変更をソースリポジトリの `main` ブランチにプッシュします。	

タスク	説明	必要なスキル
パイプラインのステージを検証します。	1. AWS マネジメントコンソールにサインインし、CodePipeline コンソールを開きます。 2. 前のエピックセクションの `terraform apply "tfplan"` コマンドから生成された出力で、生成された CodePipeline の名前を見つけてください。 3. CodePipeline コンソールでパイプラインを開き、[変更のリリース] を選択します。 4. パイプラインの各ステージを確認し、予測どおりに機能していることを確認します。 詳細については、AWS CodePipeline ユーザーガイドの「パイプラインの詳細と履歴を表示する (コンソール)」を参照してください。 **Important** ソースリポジトリのメインブランチに変更がコミットされると、テストパイプラインは自動的にアクティブ化されます。	DevOps エンジニア

タスク	説明	必要なスキル
レポート出力を確認します。	1. CodePipeline コンソールの左側のナビゲーションペインで、[ビルド] を選択します。次に、[レポート履歴] を選択します。 2. パイプラインが生成する tfsec と checkov のスキャンレポートを確認します。このレポートは、視覚化とグラフィックにより問題を特定するのに役立ちます。 **Note** `<project_name>-validate` CodeBuild プロジェクトは、“`validate`” ステージ中にコードの脆弱性レポートを生成します。	DevOps エンジニア

リソースのクリーンアップ

タスク	説明	必要なスキル
パイプラインと関連リソースをクリーンアップします。	AWS アカウントからテストリソースを削除するには、クローン作成したリポジトリのルートフォルダで次のコマンドを実行します。	DevOps エンジニア

タスク	説明	必要なスキル
	`terraform destroy --var-file=terraform.tfvars`	

トラブルシューティング

問題	ソリューション
“apply” ステージ中に AccessDenied エラーが表示される。	1. “apply” ステージに関連付けられた CodeBuild プロジェクトの実行ログを確認して、不足している IAM 権限がないか確認します。詳細については、AWS CodeBuild ユーザーガイドの「AWS CodeBuild のビルド詳細を表示する」を参照してください。 2. コードエディタで、クローン作成したリポジトリの`modules` フォルダを開きます。次に、`iam-role` フォルダに移動し、そのフォルダにある `main.tf` ファイルを開きます。 3. `codepipeline_policy` ステートメントに、AWS アカウントのリソースをプロビジョニングするために必要な IAM ポリシーを追加します。

関連リソース

- モジュールブロック (Terraform ドキュメント)
- CI/CD を使用して Terraform で AWS セキュリティサービスをデプロイおよび設定する方法 (AWS ブログ記事)
- サービスにリンクされたロールの使用 (IAM ドキュメント)
- create-pipeline (AWS CLI ドキュメント)
- Amazon S3 for CodePipeline に保存されているアーティファクトのサーバー側の暗号化を設定する (AWS CodePipeline ドキュメント)

- AWS CodeBuild のクォータ (AWS CodeBuild ドキュメント)
- AWS CodePipeline でのデータ保護 (AWS CodePipeline ドキュメント)

追加情報

カスタム Terraform モジュール

以下は、このパターンで使用されるカスタム Terraform モジュールのリストです。

- `codebuild_terraform` は、パイプラインの各ステージを形成する CodeBuild プロジェクトを作成します。
- `codecommit_infrastructure_source_repo` は、ソース CodeCommit リポジトリをキャプチャして作成します。
- `codepipeline_iam_role` は、パイプラインに必要な IAM ロールを作成します。
- `codepipeline_kms` は、Amazon S3 オブジェクトの暗号化と復号化に必要な AWS KMS キーを作成します。
- `codepipeline_terraform` は、ソース CodeCommit リポジトリのテストパイプラインを作成します。
- `s3_artifacts_bucket` は、Amazon S3 バケットを作成して、パイプラインアーティファクトを管理します。

ビルド仕様ファイル

以下は、このパターンが各パイプラインステージを実行するために使用するビルド仕様 (buildspec) ファイルのリストです。

- `buildspec_validate.yml` は、“validate” ステージを実行します。
- `buildspec_plan.yml` は、“plan” ステージを実行します。
- `buildspec_apply.yml` は、“apply” ステージを実行します。
- `buildspec_destroy.yml` は、“destroy” ステージを実行します。

ビルド仕様ファイル変数

各 buildspec ファイルは次の変数を使用して異なるビルド固有の設定を有効にします。

変数	デフォルト値	説明
CODE_SRC_DIR	"."	ソース CodeCommit ディレクトリを定義します。
TF_VERSION	「1.3.7」	ビルド環境の Terraform バージョンを定義します。

buildspec_validate.yml ファイルでは、さまざまなビルド固有の設定を有効にする以下の変数もサポートしています。

変数	デフォルト値	説明
SCRIPT_DIR	「./templates/scripts」	スクリプトディレクトリを定義します。
ENVIRONMENT	「dev」	環境名を定義します
SKIPVALIDATIONFAILURE	「Y」	障害発生時の検証をスキップします
ENABLE_TFVALIDATE	「Y」	Terraform 検証を有効にします
ENABLE_TFFORMAT	「Y」	Terraform フォーマットを有効にします
ENABLE_TFCHECKOV	「Y」	checkovスキャンを有効にします
ENABLE_TFSEC	「Y」	tfsec スキャンを有効にします
TFSEC_VERSION	「v1.28.1」	tfsec バージョンを定義します。

その他のパターン

- AWS PrivateLink と Network Load Balancer を使用して、Amazon EKS 上のコンテナアプリケーションにプライベートアクセスします。
- ある の AWS CodeCommit リポジトリ AWS アカウント を別のアカウントの Amazon SageMaker AI Studio Classic に関連付ける
- で Landing Zone Accelerator を使用してアカウントの作成を自動化する AWS
- AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する
- AWS Batch を使用して Amazon RDS for PostgreSQL DBインスタンスのバックアップを自動化します
- AWS SAM を使用してネストされたアプリケーションのデプロイを自動化
- CI/CD パイプラインを使用して Amazon EKS へのノード終了ハンドラーのデプロイを自動化する
- Amazon MQ で RabbitMQ 設定を自動化する
- 全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント
- CI/CD パイプラインを使用して Amazon EKS へ Java アプリケーションを自動的にビルドし、デプロイする
- Python アプリケーションを使用して Amazon DynamoDB の PynamoDB モデルと CRUD 関数を自動的に生成する DynamoDB
- CodePipeline、IAM アクセスアナライザー、AWS CloudFormation マクロを使用して、AWS アカウントの IAM ポリシーとロールを自動的に検証してデプロイする
- AWS クラウド上の Stromasys Charon-SSP エミュレーターで Sun SPARC サーバーをバックアップ
- AWS DataOps Development Kit を使用して Google Analytics データを取り込み、変換、分析するためのデータパイプラインを構築する
- Amazon EC2 Auto Scaling と Systems Manager を搭載した Micro Focus Enterprise Server PAC を構築する
- EC2 Image Builder と Terraform を使用して、強化されたコンテナイメージ用のパイプラインを構築する
- Amazon SageMaker AI と Azure DevOps を使用して MLOps ワークフローを構築する
- AWS Managed Microsoft AD およびオンプレミスの Microsoft Active Directory を使用して DNS 解決を一元化する
- サーバーレスアプローチを使用して AWS サービスを連結する
- NLog を使用して Amazon CloudWatch Logs 内の.NET アプリケーションのロギングを設定します

- AWS CodeCommit リポジトリから最新の AWS Amplify ウェブアプリケーションを継続的にデプロイする
- SageMaker 用のカスタム Docker コンテナイメージを作成し、AWS Step Functions のモデルトレーニングに使用する
- AWS CodePipeline に適用されない AWS リージョンにパイプラインを作成
- Amazon CloudWatch 異常検知により、カスタムメトリクスのアラームを作成
- AWS CDK アスペクトとエスケープハッチを使用してデフォルトのロール名をカスタマイズする
- 複数のコード成果物のセキュリティ問題を同時に検出するパイプラインをデプロイする
- インフラストラクチャをコードとして使用して、AWS クラウドにサーバーレスデータレイクをデプロイして管理する
- Amazon EKS と Amazon S3 の Helm チャートリポジトリを使用して Kubernetes のリソースとパッケージをデプロイする
- AWS CDK と TypeScript を使用してマルチスタックのアプリケーションをデプロイする
- Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする
- RAG および ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する
- AWS CloudFormation テンプレートを使用して Amazon GuardDuty を条件付きで有効にする
- Amazon EKS Pod Identity と KEDA を使用して Amazon EKS でイベント駆動型自動スケーリングを設定する
- Amazon Personalize を使用して、パーソナライズされ再ランク付けされたレコメンデーションを生成します
- AWS KMS キーのキーの状態が変更されたときに Amazon SNS 通知を受け取る
- Amazon ECR リポジトリに移行するときに、重複するコンテナイメージを自動的に特定する
- Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する
- AWS CDK を使用して複数の AWS リージョン、アカウント、OU にわたって Amazon DevOps Guru を有効にすることで、運用パフォーマンスを向上させます。
- Kubernetes DaemonSet を使用して Amazon EKS ワーカーノードに SSM エージェントをインストールします
- Stonebranch ユニバーサルコントローラーと AWS Mainframe Modernizationを統合
- メインフレームのモダナイゼーション: Rocket Software Enterprise Suite AWS を使用した での DevOps

- を使用してアクセス AWS IAM Identity Center 許可セットをコードとして管理する AWS CodePipeline
- AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。
- DNS レコードを Amazon Route 53 プライベートホストゾーンに一括で移行する
- AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する
- 複数の で共有 Amazon マシンイメージの使用をモニタリングする AWS アカウント
- App2Container が生成した Docker イメージを最適化する
- AWS Step Functions を使用して ETL パイプラインを検証、変換、パーティショニングでオーケストレーションします
- IaC 原則を使用して Amazon Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する
- 非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約
- コードリポジトリ AWS Service Catalog を使用して で Terraform 製品をプロビジョニングする
- フィルタリングされた Amazon ECR コンテナイメージをアカウントまたはリージョン間で複製する
- コンテナを再起動せずにデータベースの認証情報をローテーションする
- AWS Step Functions から AWS Systems Manager Automation タスクを同期的に実行する AWS Step Functions
- AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する
- Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する
- Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする
- AWS CloudFormation を使用して Amazon EC2 に UiPath RPA ボットを自動的にセットアップする
- Application Load Balancer を使用して Amazon ECS の相互 TLS でアプリケーション認証を簡素化する
- C# と AWS CDK を使用するサイロモデル用の SaaS アーキテクチャでのテナントオンボーディング
- Terraform を使用すると、組織の Amazon GuardDuty が自動的に有効になります。

- Amazon Bedrock エージェントを使用して、テキストベースのプロンプトによる Amazon EKS でのアクセスエントリコントロールの作成を自動化する
- Account Factory for Terraform (AFT) のコードをローカルで検証する
- フラスコと AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化

インフラストラクチャ

トピック

- Session Manager と Amazon EC2 Instance Connect により踏み台ホストにアクセス
- AWS Managed Microsoft AD およびオンプレミスの Microsoft Active Directory を使用して DNS 解決を一元化する
- Amazon CloudWatch オブザーバビリティアクセスマネージャーを使用してモニタリングを一元化
- 起動時に EC2 インスタンスに必須タグが欠けていないか確認する
- Session Manager を使用して Amazon EC2 インスタンスに接続
- AWS CodePipeline に適用されない AWS リージョンにパイプラインを作成
- AWS CDK アスペクトとエスケープハッチを使用してデフォルトのロール名をカスタマイズする
- Amazon EC2 にプライベート静的 IP を使用して Cassandra クラスターをデプロイしてリバランスを回避する
- AWS Transit Gateway Connect を使用して VRF を AWS に拡張する
- AWS KMS キーのキーの状態が変更されたときに Amazon SNS 通知を受け取る
- 非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約
- コードリポジトリ AWS Service Catalog を使用して で Terraform 製品をプロビジョニングする
- Amazon SES を使用して、単一メールアドレスで複数の AWS アカウントを登録する
- 単一アカウントの AWS 環境でハイブリッドネットワークの DNS 解決を設定
- AWS CloudFormation を使用して Amazon EC2 に UiPath RPA ボットを自動的にセットアップする
- AWS で可用性の高い PeopleSoft アーキテクチャを設定する
- AWS Elastic Disaster Recoveryで Oracle JD Edwards EnterpriseOne のディザスタリカバリをセットアップする
- マルチリージョン、マルチアカウント組織で AWS CloudFormation ドリフト検出を設定する
- S3 バケットを AWS CloudFormation スタックとして正常にインポートしました
- AWS DataSync を使用して、さまざまな AWS リージョンの Amazon EFS ファイルシステム間でデータを同期する
- LocalStack および Terraform Tests を使用して AWS インフラストラクチャをテストする
- SAP ペースメーカークラスターを ENSA1 から ENSA2 にアップグレード
- 異なる AWS アカウント間の VPC で一貫したアベイラビリティーゾーンを使用する

- アクセスコントロールと自動化に IAM ポリシーでユーザー IDs
- Account Factory for Terraform (AFT) のコードをローカルで検証する
- その他のパターン

Session Manager と Amazon EC2 Instance Connect により踏み台ホストにアクセス

作成者：Piotr Chotkowski (AWS) と Witold Kowalik (AWS)

概要

踏み台ホストは、ジャンプボックスと呼ばれることもありますが、外部ネットワークからプライベートネットワークにあるリソースへの単一アクセスポイントを提供するサーバーです。インターネットなどの外部のパブリックネットワークに公開されているサーバーは、不正アクセスによる潜在的なセキュリティリスクをもたらします。これらのサーバーへのアクセスを保護し、制御することは重要です。

このパターンでは、Session Manager と Amazon EC2 Instance Connect を使用して、 にデプロイされた Amazon Elastic Compute Cloud (Amazon EC2) 踏み台ホストに安全に接続する方法について説明します AWS アカウント。Session Manager は の一機能です AWS Systems Manager。このパターンには次のようなメリットがあります。

- デプロイされた踏み台ホストには、パブリックインターネットに公開されているオープンなインバウンドポートはありません。これにより、潜在的なアタックサーフェスが減少します。
- に長期 Secure Shell (SSH) キーを保存して維持する必要はありません AWS アカウント。代わりに、各ユーザーは踏み台ホストに接続するたびに新しい SSH キーペアを生成します。 AWS Identity and Access Management （ IAM) ポリシーは、ユーザーの AWS 認証情報にアタッチされ、踏み台ホストへのアクセスを制御します。

対象者

このパターンは、Amazon EC2、Amazon Virtual Private Cloud (Amazon VPC)、および Hashicorp Terraform の基本的な知識を持つ読者を対象としています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Command Line Interface （ AWS CLI) バージョン 2、インストールおよび設定済み
- 用の Session Manager プラグイン AWS CLI、インストール済み

- 「インストール済み」のテラフォーム CLI
- Amazon Simple Storage Service (Amazon S3) バケットやAmazon DynamoDB テーブルなど Terraform ステータスを格納するリモートバックエンドとしてのTerraform「ステータス」の格納 Terraform 状態のリモートバックエンドの使用の詳細については、Amazon S3 バックエンド (Terraform ドキュメント) 」を参照してください。Amazon S3 バックエンドでリモート状態管理を設定するコードサンプルについては、「remote-state-s3-backend (Terraform Registry)」を参照してください。次の要件に注意してください。
  - Amazon S3 バケットと DynamoDB テーブルは同じ にある必要があります AWS リージョン。
  - DynamoDB テーブルを作成する場合、パーティションキーは `LockID` (大文字と小文字を区別)、パーティションキータイプは `String` である必要があります。他の設定はすべてデフォルト値のままにしておきます。詳細については、「DynamoDB のドキュメント」の「プライマリキーについて」と「テーブルの作成」を参照してください。
- SSH クライアント、インストール済み

制約事項

- このパターンは、概念実証 (PoC) として、または今後の開発の基礎となることを目的としています。本稼働環境では、現行の形式を使用しないでください。デプロイする前に、要件とユースケースに合うようにリポジトリ内のサンプルコードを見直してください。
- このパターンは、ターゲットの踏み台ホストがオペレーティングシステムとして Amazon Linux 2 を使用していることを前提としています。他の Amazon マシンイメージ (AMI) を使用することもできますが、他のオペレーティングシステムはこのパターンの対象外です。

> **Note**
>
> Amazon Linux 2 のサポート終了が近づいています。詳細については、「Amazon Linux 2 のFAQs」を参照してください。

- このパターンでは、踏み台ホストは NAT ゲートウェイとインターネットゲートウェイのないプライベートサブネットに配置されます。この設計により、Amazon EC2 インスタンスはパブリックインターネットから分離されます。インターネットと通信できるようにする特定のネットワーク設定を追加できます。詳細については、「Amazon VPC のドキュメント」の「仮想プライベートクラウド (VPC) を他のネットワークに接続」を参照してください。同様に、最小特権の原則に従って、明示的にアクセス許可を付与 AWS アカウント しない限り、踏み台ホストは 内の他のリソースにアクセスできません。詳細については、IAM ドキュメントの「リソースベースのポリシー」を参照してください

製品バージョン

- AWS CLI バージョン 2
- Terraform バージョン 1.3.9

アーキテクチャ

ターゲットテクノロジースタック

- シングルパブリックサブネットを持つ VPC
- 以下の「インターフェイス VPC エンドポイント」：
  - `amazonaws.<region>.ssm` – AWS Systems Manager サービスのエンドポイント。
  - `amazonaws.<region>.ec2messages` — Systems Manager は、このエンドポイントを使用して、SSM Agent から Systems Manager サービスへの呼び出しを行います。
  - `amazonaws.<region>.ssmmessages` – Session Manager はこのエンドポイントを使用して、安全なデータチャネルを介して Amazon EC2 インスタンスに接続します。
- `t3.nano` Amazon Linux 2 を実行する Amazon EC2 インスタンス
- IAM ロールとインスタンス プロファイル
- エンドポイントと Amazon EC2 インスタンスの Amazon VPC セキュリティグループとセキュリティグループルール

ターゲットアーキテクチャ

図に示す内容は以下のとおりです。

1. ユーザーが割り当てられている IAM ロールには、次の操作を実行する権限があります。
   - Amazon EC2 インスタンスの認証、認可、接続
   - Session Management (IAM) でセッションを開始
2. ユーザーは Session Manager から SSH セッションを開始します。
3. Session Manager はユーザーを認証し、関連する IAM ポリシーの権限を検証し、設定を確認し、SSM エージェントにメッセージを送信して双方向接続を開きます。
4. ユーザーは Amazon EC2 メタデータを介して SSH パブリックキーを踏み台ホストにプッシュします。これは各接続の前に行う必要があります。SSH 公開鍵は 60 秒間使用可能です。

5. 踏み台ホストは、Systems Manager と Amazon EC2 のインターフェイス VPC エンドポイントと通信します。
6. ユーザーは TLS 1.2 で暗号化された双方向通信チャネルにより、Session Manager を通じて踏み台ホストにアクセスします。

自動化とスケール

このアーキテクチャを自動的にデプロイしまたは拡張するには、次のオプションを使用します。

- 継続的な統合および継続的な提供 (CI/CD) パイプラインで、アーキテクチャをデプロイできます。
- コードを変更して、踏み台ホストのインスタンスタイプを変更できます。
- コードを変更して複数の踏み台ホストをデプロイできます。`bastion-host/main.tf` ファイルの `aws_instance` リソースブロックに、`count` メタ引数を追加します。詳細については、「Terraform のドキュメント」を参照してください。

ツール

AWS のサービス

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Identity and Access Management (IAM) は、リソースの使用を認証および認可されたユーザーを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間が短縮され、 AWS リソースを大規模に安全に管理できます。このパターンは、システムマネージャーの機能である「Session Manager」 を使用します。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。このパターンは「Terraform CLI」を使用しています。

コードリポジトリ

このパターンのコードは、GitHub·内の「Session Manager と Amazon EC2 Instance Connectを使用して踏み台ホストにアクセスする」リポジトリで利用できます。

ベストプラクティス

- コードのセキュリティと品質を向上させるために、自動コードスキャンツールの使用をお勧めします。このパターンは、IaC 用の静的コード分析ツールである「Checkov」を使用してスキャンされました。最低でも、`terraform validate` と `terraform fmt -check -recursive` Terraform コマンドを使用して基本的な検証とフォーマットのチェックを行うことをお勧めします。
- IaC の自動テストを追加することは良い習慣です。Terraform コードをテストするさまざまな方法の詳細については、「HashiCorp Terraform のテスト」(Terraform ブログ記事) を参照してください。
- デプロイ中、Terraform はAmazon EC22 インスタンスが置き換えられます。これにより、パッチやアップグレードを含む新しいバージョンのオペレーティングシステムがデプロイされます。デプロイスケジュールの頻度が低い場合、インスタンスに最新のパッチが適用されないため、セキュリティ上のリスクが生じる可能性があります。デプロイされた Amazon EC2 インスタンスにセキュリティパッチを頻繁に更新して適用することが重要です。詳細については、「Amazon EC2 での更新管理」を参照してください。
- このパターンは概念実証であるため、 AWS などの 管理ポリシーを使用しますAmazonSSMManagedInstanceCore。 AWS 管理ポリシーは一般的なユースケースを対象としますが、最小特権のアクセス許可は付与しません。ユースケースに応じて、このアーキテクチャにデプロイされたリソースに最小特権のアクセス権限を付与するカスタムポリシーを作成することをお勧めします。詳細については、「 AWS 管理ポリシーの使用を開始し、最小特権のアクセス許可に移行する」を参照してください。
- SSH キーへのアクセスを保護し、キーを安全な場所に保存するにはパスワードを使用してください。
- 踏み台ホストのロギングとモニタリングを設定します。記録とモニタリングは、運用面とセキュリティ面の両方の観点から、システムを保守する上で重要な部分です。踏み台ホストの接続とアク

ティビティをモニタリングする方法は複数あります。詳細については、Systems Manager ドキュメントの次のトピックを参照してください。

- モニタリング AWS Systems Manager
- でのログ記録とモニタリング AWS Systems Manager
- 「セッションアクティビティの監査」
- 「セッションアクティビティのログ記録」

エピック

リソースのデプロイ

タスク	説明	必要なスキル
コードリポジトリを複製します。	1. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 2. 次のコマンドを入力します。 `git clone https://github.com/aws-samples/secured-bastion-host-terraform.git`	DevOps エンジニア、開発者
ディレクトリから Terraform を開始します。	このステップは最初のデプロイにのみ必要です。このパターンをレデプロイする場合、次の手順に進んでください。 クローンしたリポジトリのルートディレクトリに、以下のコマンドを入力します。	DevOps エンジニア、開発者、Terraform

タスク	説明	必要なスキル

	• $S3_STATE_BUCKET は、Terraform 状態を含む Amazon S3 バケットの名前です。
	• $PATH_TO_STATE_FILE は infra/bastion-host/tetfstate など Terraform ステートファイルのキーです。
	• $AWS_REGION は、Amazon S3 バケットがデプロイされているリージョンです。
	
	`terraform init \`
	`  -backend-config="bucket=$S3_STATE_BUCKET" \`
	`  -backend-config="key=$PATH_TO_STATE_FILE" \`
	`  -backend-config="region=$AWS_REGION`
	
	Note
	または、config.tf ファイルを開き、terraform セクションでこれらの値を手動で指定することもできます。	

タスク	説明	必要なスキル
リソースのデプロイ	1. クローン作成したリポジトリのルート ディレクトリで、次のコマンドを入力します。 `terraform apply -var-file="dev.tfvars"` 2. に適用されるすべての変更のリストを確認し AWS アカウント、デプロイを確認します。 3. すべてのリソースがデプロイされるまでお待ちください。	DevOps エンジニア、開発者、Terraform

ローカル環境の設定

タスク	説明	必要なスキル
SSH 接続を設定します。	SSH 設定を更新して、Session Manager を介して SSH 接続を有効にします。手順については、「Session Manager の SSH 接続を許可」を参照してください。これにより、許可されたユーザーは、プロキシコマンドを入力し、Session Manager セッションを開始し、双方向接続を介してすべてのデータを転送することができます。	DevOps エンジニア
SSH キーを生成します。	次のコマンドを入力して、ローカルで非公開と公開 SSH	DevOps エンジニア、開発者

タスク	説明	必要なスキル
	キーペアを生成します。このキーペアで踏み台ホストに接続します。 `ssh-keygen -t rsa -f my_key`	

Session Manager で踏み台ホストに接続

タスク	説明	必要なスキル
インスタンスの ID を取得します。	1. デプロイされた踏み台ホストに接続するには、Amazon EC2 インスタンスの ID が必要です。ID を検索するには、次のいずれかの操作を行います • Amazon EC2 コンソールを開きます。ナビゲーションペインで、[インスタンス] を選択してください。踏み台ホストのインスタンスを見つける • で AWS CLI、次のコマンドを入力します。 `aws ec2 describe-instances` 結果をフィルタリングするには、次のコマンドを入力します。ここで $BASTION_HOST_TAG は踏み台ホストに割り	AWS 全般

タスク	説明	必要なスキル
	当てたタグです。このタグのデフォルト値は sandbox-dev-bastion-host です。 `aws ec2 describe-instances \ --filters "Name=tag:Name,Values=$BASTION_HOST_TAG" \ --output text \ --query 'Reservations[*].Instances[*].InstanceId' \ --output text` 2. Amazon EC2 インスタンスの ID をコピーします。この値は後で使用します。	

タスク	説明	必要なスキル
SSH パブリックキーを送信します。	**Note** このセクションでは、踏み台ホストのインスタンスメタデータにパブリックキーをアップロードします。キーがアップロードされたら、60 秒以内に踏み台ホストとの接続を開始してください。60 秒後、パブリックキーは削除されます。詳細については、このパターンの「トラブルシューティング」セクションを参照してください。踏み台ホストに接続する前にキーが削除されないように、次の手順をすぐに実行してください。 1. Amazon EC2 Instance Connect を使用して、踏み台ホストに SSH キーを送信します。次のコマンドを入力します。 • `$INSTANCE_ID` は Amazon EC2 インスタンスの ID です。 • `$PUBLIC_KEY_FILE` は、`my_key.pub` など	AWS 全般

タスク	説明	必要なスキル

	のパブリックキーファイルへのパスです。
	
	Important
	プライベートキーではなく、パブリックキーを使用してください。
	
	``
	aws ec2-instance-connect send-ssh-public-key \
	    --instance-id $INSTANCE_ID \
	    --instance-os-user ec2-user \
	    --ssh-public-key file://$PUBLIC_KEY_FILE
	``
	
	2. キーが正常にアップロードされたことを示すメッセージが表示されるまでお待ちください。直ちに次のステップに進んでください。	

タスク	説明	必要なスキル
踏み台ホストに接続します。	1. 次のコマンドを入力して、Session Manager経由で踏み台ホストに接続します。 • `$PRIVATE_KEY_FILE` は my_key などプライベートキーへのパスです。 • `$INSTANCE_ID` は Amazon EC2 インスタンスの ID です。 `ssh -i $PRIVATE_KEY_FILE ec2-user@$INSTANCE_ID` 2. `yes` を入力して接続を確認します。Session Manager を使用して SSH 接続が開きます。 **Note** 踏み台ホストとの SSH 接続を開く方法は他にもあります。詳細については、このパターンの「追加情報」セクションの「踏み台ホストとの SSH 接続を確立するための代替方法」を参照してください。	AWS 全般

(オプション) クリーンアップする

タスク	説明	必要なスキル
デプロイされたリソースを削除します。	1. デプロイされたすべてのリソースを削除するには、クローンされたリポジトリのルートディレクトリから次のコマンドを実行します。 `terraform destroy -var-file="dev.tfvars"` 2. リソースの削除を確認します。	DevOps エンジニア、開発者、Terraform

トラブルシューティング

問題	ソリューション
踏み台ホストに接続しようと試みる時の `TargetNotConnected` エラー	1. 「Amazon EC2 のドキュメント」の「インスタンスの再起動」の手順に従って、踏み台ホストを再起動します。 2. インスタンスが正常に再起動したら、パブリックキーを踏み台ホストに再送信し、接続を再試行します。
踏み台ホストに接続しようと試みる時の `Permission denied` エラー	パブリックキーが踏み台ホストにアップロードされたら、60 秒以内に接続を開始してください。60 秒が経過すると、そのキーは自動的に削除され、そのキーを使用してインスタンスに接続できなくなります。その場合は、手順を繰り返してそのキーをインスタンスに再送信できます。

関連リソース

AWS ドキュメント

- AWS Systems Manager Session Manager (Systems Manager ドキュメント)
- の Session Manager プラグインをインストールする AWS CLI (Systems Manager ドキュメント)
- 「Session Manager の SSH 接続を許可」(Systems Manager のドキュメント)
- 「EC2 Instance Connect の使用について」(Amazon EC2 のドキュメント)
- 「EC2 Instance Connect で接続」(Amazon EC2 のドキュメント)
- 「Amazon EC2 の Identity and Access Management」(Amazon EC2 のドキュメント)
- 「IAM ロールを使用して、Amazon EC2 インスタンスで実行されているアプリケーションにアクセス許可を付与する」(IAM ドキュメント)
- IAM におけるセキュリティのベストプラクティス」(IAM のドキュメント)
- 「セキュリティグループを使用してリソースへのトラフィックを制御する」(Amazon VPC ドキュメント)

その他のリソース

- 「Terraform 開発者向けウェブページ」
- 「コマンド：検証」(Terraform のドキュメント)
- 「コマンド：fmt」(Terraform のドキュメント)
- 「HashiCorp Terraform のテスト」(HashiCorp ブログ記事)
- 「Checkov webpage」

追加情報

踏み台ホストとの SSH 接続を確立するための代替方法

ポート転送

この `-D 8888` オプションを使用して、動的ポートフォワーディングで SSH 接続を開くことができます。詳細については、explainshell.com の手順を参照してください。以下は、ポート転送を使用して SSH 接続を開くコマンドの例です。

```
ssh -i $PRIVATE_KEY_FILE -D 8888 ec2-user@$INSTANCE_ID
```

この接続は、ローカルブラウザからのトラフィックを踏み台ホスト経由で転送できる SOCKS プロキシを開くようなものです。Linux または MacOS をご使用の場合、すべてのオプションを表示するには、`man ssh` を入力します。SSH リファレンスマニュアルが表示されます。

提供されたスクリプトを使用

「エピック」セクションで、Session Manager で踏み台ホストに接続に説明されている手順を手動で実行する代わりに、コードリポジトリに含まれている connect.sh スクリプトを使用することができます。このスクリプトは SSH キーペアを生成し、パブリックキーを Amazon EC2 インスタンスにプッシュして、踏み台ホストとの接続を開始します。コマンドを実行すると、タグとキー名を引数として渡します。以下はスクリプトを実行するコマンドの例です。

```
./connect.sh sandbox-dev-bastion-host my_key
```

AWS Managed Microsoft AD およびオンプレミスの Microsoft Active Directory を使用して DNS 解決を一元化する

作成者: Brian Westmoreland (AWS)

概要

このパターンは、 AWS Directory Service for Microsoft Active Directory （ AWS Managed Microsoft AD) と Amazon Route 53 の両方を使用して AWS 、マルチアカウント環境内で DNS 解決を一元化するためのガイダンスを提供します。このパターンでは、DNS 名前空間はオンプレミス AWS DNS 名前空間のサブドメインです。このパターンでは、オンプレミスの DNS ソリューションが Microsoft Active Directory を使用する場合 AWS にクエリを に転送するようにオンプレミスの DNS サーバーを設定する方法についても説明します。

前提条件と制限

前提条件

- を使用して設定された AWS マルチアカウント環境 AWS Organizations。
- ネットワーク接続は 間で確立されます AWS アカウント。
- AWS とオンプレミス環境との間で確立されたネットワーク接続 (AWS Direct Connect または任意のタイプの VPN 接続を使用 ）。
- AWS Command Line Interface （ AWS CLI) ローカルワークステーションで設定します。
- AWS Resource Access Manager （ AWS RAM) アカウント間で Route 53 ルールを共有するために使用されます。したがって、エピックセクションで説明されているように、 AWS Organizations 環境内で共有を有効にする必要があります。

機能制限

- AWS Managed Microsoft AD Standard Edition には 5 つの共有の制限があります。
- AWS Managed Microsoft AD Enterprise Edition には 125 共有の制限があります。
- このパターンのソリューションは、共有 AWS リージョン をサポートする に限定されます AWS RAM。

製品バージョン

- Windows Server 2008、2012、2012 R2、または 2016 で実行されている Microsoft Active Directory。

アーキテクチャ

ターゲットアーキテクチャ

この設計では、 AWS Managed Microsoft AD は共有サービスにインストールされます AWS アカウント。これは要件ではありませんが、このパターンはこの設定を前提としています。別の AWS Managed Microsoft AD で を設定する場合は AWS アカウント、それに応じて工ピックセクションのステップを変更する必要がある場合があります。

この設計では、Route 53 Resolver を使用して、Route 53 ルールによる名前解決案が適用されます。オンプレミスの DNS ソリューションが Microsoft DNS を使用する場合、会社の DNS 名前空間(`aws.company.com`) のサブドメインである AWS 名前空間(`company.com`)の条件付き転送ルールを作成するのは簡単ではありません。従来の条件付きフォワーダーを作成しようとすると、エラーになります。これは、Microsoft アクティブディレクトリが、 `company.com` のどのサブドメインに対してもすでに権限があると見なされているためです。このエラーを回避するには、まずその名前空間の権限を委任するために、 `aws.company.com` の委任を作成する必要があります。作成後には、条件付きフォワーダーを作成できます。

各スポークアカウントの Virtual Private Cloud (VPC) は、ルート名前空間に基づいて独自の DNS AWS 名前空間を持つことができます。この設計では、各スポークアカウントが、ベースの AWS 名前空間にアカウント名の省略形を追加します。スポークアカウントのプライベートホストゾーンが作成されると、ゾーンはスポークアカウントのローカル VPC と中央 AWS ネットワークアカウントの VPC に関連付けられます。これにより、中央 AWS ネットワークアカウントはスポークアカウントに関連する DNS クエリに応答できます。これにより、Route 53 と の両方 AWS Managed Microsoft AD が協力して、 AWS 名前空間 () を管理する責任を共有しますaws.company.com。

自動化とスケール

この設計では、Route 53 Resolver エンドポイントを使用して、 AWS とオンプレミス環境の間で DNS クエリをスケーリングします。Route 53 Resolver の各エンドポイントは、複数のエラスティックネットワークインターフェイス (複数のアベイラビリティーゾーンにわたって分散している) で構成され、各ネットワークインターフェースは 1 秒あたり最大 10,000 件のクエリを処理できます。Route 53 Resolver は、エンドポイントあたり最大 6 つの IP アドレスが適用されるため、この

設計は複数のアベイラビリティーゾーンにまたがって 1 秒あたり最大 60,000 件の DNS クエリが適用され、高い可用性を実現します。

さらに、このパターンでは、 内の将来の成長が自動的に考慮されます AWS。オンプレミスで設定された DNS 転送ルールは、新しい VPCs と、それらに関連付けられたプライベートホストゾーンが追加されたことをサポートするために変更する必要はありません AWS。

ツール

AWS サービス

- AWS Directory Service for Microsoft Active Directory では、ディレクトリ対応のワークロードと AWS リソースが で Microsoft Active Directory を使用できるようになります AWS クラウド。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- AWS Resource Access Manager （ AWS RAM ） は、 間でリソースを安全に共有 AWS アカウントし、運用上のオーバーヘッドを削減し、可視性と監査可能性を提供します。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

ツール

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。このパターンでは、 AWS CLI を使用して Route 53 認可を設定します。

エピック

AWS Managed Microsoft AD ディレクトリを作成して共有する

タスク	説明	必要なスキル
デプロイします AWS Managed Microsoft AD。	1. 新しいディレクトリを作成し設定します。詳細な手順については、「 AWS Directory Service 管理ガイド」の「 の作成 AWS Managed Microsoft AD」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
	2. AWS Managed Microsoft AD ドメインコントローラーの IP アドレスを記録します。 これらは、後の手順で参照します。	
ディレクトリを共有します。	ディレクトリが構築されたら、 AWS 組織 AWS アカウント 内の他の と共有します。手順については、「 AWS Directory Service 管理ガイド」の「ディレクトリを共有する」を参照してください。 **Note** AWS Managed Microsoft AD Standard Edition には 5 つの共有の制限があります。Enterprise Edition には 125 つの共有の制限があります。	AWS 管理者

Route 53 を設定

タスク	説明	必要なスキル
Route 53 Resolvers を作成します。	Route 53 リゾルバーは、AWS とオンプレミスデータセンター間の DNS クエリ解決を容易にします。 1. Route 53 開発者ガイドの「手順」 に従って、Route	AWS 管理者

タスク	説明	必要なスキル
	53 Resolver をインストールします。 2. 高可用性を実現するために、中央 AWS ネットワークアカウント VPC 内の少なくとも 2 つのアベイラビリティーゾーンのプライベートサブネットで Route 53 Resolver を設定します。 **Note** 中央 AWS ネットワークアカウント VPC の使用は必須ではありませんが、残りのステップではこの設定を前提としています。	

タスク	説明	必要なスキル

Route 53 ルールを作成します。	特定のユースケースでは多数の Route 53 ルールが必要になる場合がありますが、ベースラインとして以下のルールを設定する必要があります
	• 中央ネットワークアカウントのアウトバウンド Route 53 Resolver を使用したオンプレミス名前空間 (company.com) の送信ルール。ターゲット IP アドレスはオンプレミス DNS サーバーです。
	  • このルールを中央ネットワークアカウント VPC に関連付けます。
	• 中央ネットワークアカウントのアウトバウンド Route 53 Resolver を使用した AWS 名前空間 (aws.company.com) の送信ルール。ターゲット IP アドレスは、中央ネットワークアカウントのインバウンド Route 53 Resolver IP アドレスです。
	  • このルールを中央 AWS ネットワークアカウント VPC (Route 53 Resolver を格納) に関連付けないでください。
	• AWS Managed Microsoft AD ドメインコントローラーを指す AWS 名前空間
		AWS 管理者

タスク	説明	必要なスキル
	(aws.company.com) の 2 番目の送信ルール (前のエピックの IPs を使用) 。 • このルールを中央 AWS ネットワークアカウント VPC (Route 53 Resolver を格納) に関連付けます。 • このルールを他の と共有したり、関連付けたりしないでください AWS アカウント。 詳細については、Route 53 開発者ガイドの「転送ルールの管理」 を参照してください。	

タスク	説明	必要なスキル
Route 53 プロファイルを設定します。	Route 53 プロファイルは、スポークアカウントとルールを共有するために使用されます。 1. 「Route 53 デベロッパーガイド」の手順に従って、中央ネットワークアカウントに新しい Route 53 プロファイルを作成します。 2. オンプレミス名前空間 (company.com) のルールをプロファイルに追加します。 3. Route 53 インバウンドリゾルバーの IP アドレスをターゲットとする AWS 名前空間 (aws.company.com) の最初のルールをプロファイルに追加します。 4. Route 53 プロファイルを AWS 組織と共有します。 5. 各スポークアカウントの Route 53 Profile リソース共有を受け入れます。 6. Route 53 プロファイルを各スポークアカウント VPC に関連付けます。	AWS 管理者

オンプレミスのアクティブディレクトリ DNS を設定

タスク	説明	必要なスキル
委任を作成します。	Microsoft DNS スナップイン (dnsmgmt.msc) を使用して、Active Directory の名前空間の company.com に対して、新しい委任を作成します。委任されたドメインの名前は aws であるはずです。これにより、委任 aws.company.com の完全に指定されたドメイン名 (FQDN) が作成されます。ネームサーバーの IP 値には AWS Managed Microsoft AD ドメインコントローラーの IP アドレスを使用し、名前server.aws.company.com には を使用します。(この委任は、委任よりも優先されるこの名前空間に対して条件付きフォワーダーが作成されるため、冗長性のためにのみ使用されます)。	アクティブディレクトリ
条件付きフォワーダーを作成します。	Microsoft DNS スナップイン (dnsmgmt.msc) を使用して、 aws.company.com の新しい条件付きフォワーダーを作成します。 条件付きフォワーダーのターゲット AWS アカウント に対して、中央 DNS の AWS インバウンド Route 53 リゾルバーの IP アドレスを使用します。	アクティブディレクトリ

スポーク用の Route 53 プライベートホストゾーンを作成する AWS アカウント

タスク	説明	必要なスキル
Route 53 プライベートホストゾーンを作成します。	Route 53 プライベートホストゾーンを各スポークアカウントで作成します。このプライベートホストゾーンをスポークアカウント VPC に関連付けます。詳細な手順については、Route 53 開発者ガイドの「プライベートホストゾーンの作成」を参照してください。	AWS 管理者
許可を作成します。	を使用して AWS CLI 、中央 AWS ネットワークアカウント VPC の認可を作成します。各スポークのコンテキストから次のコマンドを実行します AWS アカウント。 `aws route53 create-vpc-association-authorization --hosted-zone-id <hosted-zone-id> \ --vpc VPCRegion=<region>,VPCId=<vpc-id>` 各パラメータの意味は次のとおりです。 • `<hosted-zone-id>` は、スポークアカウントの Route 53 プライベートホストゾーンです。	AWS 管理者

タスク	説明	必要なスキル
	• <region> および <vpc-id>は、中央 AWS ネットワークアカウント VPC の AWS リージョン および VPC ID です。	
関連付けを行います。	を使用して、中央 AWS ネットワークアカウント VPC の Route 53 プライベートホストゾーンの関連付けを作成します AWS CLI。中央 AWS ネットワークアカウントのコンテキストから次のコマンドを実行します。 `aws route53 associate-vpc-with-hosted-zone --hosted-zone-id <hosted-zone-id> \ --vpc VPCRegion=<region>,VPCId=<vpc-id>` 各パラメータの意味は次のとおりです。 • <hosted-zone-id> は、スポークアカウントの Route 53 プライベートホストゾーンです。 • <region> および <vpc-id>は、中央 AWS ネットワークアカウントの AWS リージョン および VPC ID です。	AWS 管理者

関連リソース

- Route 53 Resolver を使用してマルチアカウント環境で DNS 管理を簡素化する (AWS ブログ記事)
- の作成 AWS Managed Microsoft AD (AWS Directory Service ドキュメント)
- AWS Managed Microsoft AD ディレクトリの共有 (AWS Directory Service ドキュメント)
- とは Amazon Route 53 Resolver (Amazon Route 53 ドキュメント)
- プライベートホストゾーンの作成 (Amazon Route 53 ドキュメント)
- Amazon Route 53 プロファイルとは (Amazon Route 53 ドキュメント)

Amazon CloudWatch オブザーバビリティアクセスマネージャーを使用してモニタリングを一元化

作成者: Anand Krishna Varanasi (AWS)、Jimmy Morgan (AWS)、Ashish Kumar (AWS)、Balaji Vedagiri (AWS)、JAGDISH KOMAKULA (AWS)、Satra Chandra Pothula (AWS)、Vivek Thangamuthu (AWS)

概要

オブザーバビリティは、アプリケーションのモニタリング、理解、トラブルシューティングに不可欠です。 AWS Control Tower またはランディングゾーンの実装と同様に、複数のアカウントにまたがるアプリケーションは、多数のログとトレースデータを生成します。問題の迅速なトラブルシューティングを行ったり、ユーザー分析やビジネス分析を理解したりするには、すべてのアカウントにまたがる共通のオブザーバビリティプラットフォームが必要です。Amazon CloudWatch オブザーバビリティアクセスマネージャーは、複数のアカウントログに一元的にアクセスし制御できます。

オブザーバビリティアクセスマネージャーを使用して、ソースアカウントによって生成されたオブザーバビリティデータログを表示および管理できます。ソースアカウントは AWS アカウント 、リソースのオブザーバビリティデータを生成する個々のアカウントです。オブザーバビリティデータは、ソースアカウントとモニタリングアカウントの間で共有されます。共有オブザーバビリティデータには、Amazon CloudWatch のメトリクス、Amazon CloudWatch Logs のログ、トレースを含めることができます AWS X-Ray。詳細については、「オブザーバビリティアクセスマネージャー 文書」を参照してください。

このパターンは、複数の で実行されるアプリケーションまたはインフラストラクチャを持ち AWS アカウント 、ログを表示するために共通の場所を必要とするユーザーを対象としています。これらのアプリケーションやインフラストラクチャの状態や状態を監視するために、Terraform を使用してオブザーバビリティアクセスマネージャーをセットアップする方法を説明します。このソリューションは、複数の方法でインストールできます。

- 手動でセットアップしたスタンドアロンの Terraform モジュールとして
- 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインの使用
- AWS Control Tower Account Factory for Terraform (AFT) などの他のソリューションと統合する

エピックセクションの手順では、手動の実装について説明しています。AFT のインストール手順については、GitHub Observability Access Manager リポジトリの README ファイルを参照してください。

前提条件と制限

前提条件

- Terraform は、システムまたは自動パイプラインにインストールまたは参照されています。(最新バージョンの使用をお勧めします。)
- 中央監視アカウントとして使用できるアカウント。他のアカウントは、ログを表示するために、中央監視アカウントへのリンクを作成します。
- (オプション) GitHub、 AWS CodeCommit Atlassian Bitbucket、または同様のシステムなどのソースコードリポジトリ。自動 CI/CD パイプラインを使用している場合、ソースコードリポジトリは必要ありません。
- (オプション) GitHub でコードレビューとコードコラボレーションのためのプルリクエスト (PR) を作成する権限。

制約事項

オブザーバビリティアクセスマネージャーには、以下のサービスクォータがあります。これらのクォータは変更できません。この特徴量を導入する前に、これらのクォータを検討します。詳細については、「CloudWatch ドキュメント」の「CloudWatch Service Quotas」を参照してください。

- ソースアカウントリンク: 各ソースアカウントを最大 5 つの監視アカウントにリンクできます。
- シンク: アカウントに対して複数のシンクを構築できますが、 ごとに 1 つのシンクのみが許可され AWS リージョン ます。

加えて:

- シンクとリンクは同じ で作成する必要があります AWS リージョン。クロスリージョンにすることはできません。

クロスリージョンとクロスアカウントのモニタリング

クロスリージョン、クロスアカウントモニタリングでは、次のいずれかのオプションを選択できます。

- アラームとメトリクスのクロスアカウントおよびクロスリージョン CloudWatch ダッシュボードを作成します。このオプションはログとトレースをサポートしていません。
- Amazon OpenSearch Service を使用して一元的なログ記録を実装します。

- すべてのテナントアカウントからリージョンごとに 1 つのシンクを作成し、メトリクスを集中モニタリングアカウント (このパターンで説明) にプッシュしてから、CloudWatch メトリクスストリームを使用して、共通の外部宛先または Datadog、Dynatrace、Sumo Logic、Splunk、New Relic などのサードパーティーのモニタリング製品にデータを送信します。

アーキテクチャ

コンポーネント

CloudWatch Observability Access Manager は、クロスアカウントオブザーバビリティを可能にする 2 つの主要なコンポーネントで構成されています。

- シンクは、ソースアカウントがオブザーバビリティデータを中央モニタリングアカウントに送信できるようにします。シンクは基本的に、ソースアカウントが接続するためのゲートウェイジャンクションを提供します。シンクゲートウェイまたは接続は 1 つしかありませんが、複数のアカウントが接続できます。
- 各ソースアカウントには、シンクゲートウェイジャンクションへのリンクがあり、オブザーバビリティデータはこのリンクを介して送信されます。各ソースアカウントからリンクを作成する前に、シンクを作成する必要があります。

アーキテクチャ

次の図表は、オブザーバビリティアクセスマネージャーとそのコンポーネントの説明です。

ツール

AWS のサービス

- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。

ツール

- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。
- AWS Control Tower Account Factory for Terraform (AFT) は、アカウントのプロビジョニングとカスタマイズに役立つ Terraform パイプラインを設定します AWS Control Tower。オプションで AFT を使用して、複数のアカウントにまたがるオブザーバビリティアクセスマネージャーを大規模にセットアップできます。

コードリポジトリ

このパターンのコードは、GitHub 内の「オブザーバビリティアクセス マネージャー」リポジトリで利用できます。

ベストプラクティス

- AWS Control Tower 環境では、ログ記録アカウントを中央モニタリングアカウント (シンク) としてマークします。
- に複数のアカウントを持つ複数の組織がある場合は AWS Organizations、個々のアカウントではなく組織を設定ポリシーに含めることをお勧めします。アカウントの数が少ない場合や、アカウントがシンク設定ポリシーの組織に含まれていない場合は、代わりに個別のアカウントを含めることを決定できます。

エピック

シンクモジュールをセットアップします

タスク	説明	必要なスキル
リポジトリをクローン作成します。	GitHub オブザーバビリティアクセスマネージャーのリポジトリをクローニングします： `git clone https://github.com/aws-samples/cloudwatch-observability-access-manager-terraform`	AWS DevOps、クラウド管理者、AWS 管理者

タスク	説明	必要なスキル
シンクモジュールのプロパティ値を指定します。	`main.tf` ファイル (リポジトリの `deployments/aft-account-customizations/LOGGING/terraform/` フォルダー内)で、以下のプロパティの値を指定します： • `sink_name` ：CloudWatch シンクの名前。 • `allowed_oam_resource_types` ：Observability Access Manager は現在、CloudWatch メトリクス、ロググループ、 AWS X-Ray トレースをサポートしています。 • `allowed_source_accounts` ：セントラル CloudWatch シンクアカウントにログを送信できるソースアカウント。 • `allowed_source_organizations` ：中央の CloudWatch シンクアカウントにログを送信できるソース AWS Control Tower 組織。 詳細については、 AWS CloudFormation ドキュメントの「AWS::Oam::Sink」を参照してください。	AWS DevOps、クラウド管理者、AWS 管理者

タスク	説明	必要なスキル

シンクモジュールをインストールします。	モニタリングアカウントとして AWS アカウント 選択した の認証情報をエクスポートし、オブザーバビリティアクセスマネージャーシンクモジュールをインストールします。
	
	`Terraform Init`
	`Terrafom Plan`
	`Terraform Apply`
		AWS DevOps、クラウド管理者、AWS 管理者

リンクモジュールをセットアップします。

タスク	説明	必要なスキル

リンクモジュールのプロパティ値を指定します。	`main.tf` ファイル (リポジトリの `deployments/aft-account-customizations/LOGGING/terraform/` フォルダー内) で、以下のプロパティの値を指定します：
	
	• `account_label` ：次のいずれかの値を使用します：
	  • `$AccountName` ：アカウントの名前。
	  • `$AccountEmail` ：メールドメインを含む、グローバルに一意のメールアドレス (例、`hello@example.com`)
		AWS DevOps、クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	• `$AccountEmailNoDomain` ：ドメイン名を含まないメールアドレス。 • `allowed_oam_resource_types` ：Observability Access Manager は現在、CloudWatch メトリクス、ロググループ、 AWS X-Ray トレースをサポートしています。 詳細については、 AWS CloudFormation ドキュメントの「AWS::Oam::Link」を参照してください。	

個々のアカウントにリンクモジュールをインストールします。	個々のアカウントの認証情報をエクスポートし、オブザーバビリティアクセスマネージャーリンクモジュールをインストールします
	
	`Terraform Plan`
	`Terraform Apply`
	
	リンクモジュールはアカウントごとに個別に設定することも、AFT を使用してこのモジュールを多数のアカウントに自動的にインストールすることもできます。
		AWS DevOps、クラウド管理者、クラウドアーキテクト

シンクからリンクへの接続を承認

タスク	説明	必要なスキル
ステータスメッセージ。	1. モニタリングアカウントにサインインします。 2. CloudWatch コンソールを開きます。 3. 左のナビゲーションペインの [設定] を選択します。 右側に、緑色のチェックマークが付いた「監視アカウントの有効化」というステータスメッセージが表示されます。つまり、モニタリングアカウントには、他のアカウントのリンクが接続されるオブザーバビリティアクセスマネージャーシンクがあることを意味します。	
リンクとシンク間の接続を承認します。	1. ステータスメッセージの下のアカウントをリンクするリソースオプションを選択します。この情報により、これがモニタリングアカウントであることが確認され、テナントソースアカウントから共有されるデータ (ログ、メトリクス、トレース) が一覧表示され、そしてアカウントラベルがとして $AccountName して表示されます。	AWS DevOps、クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	この画面では、テナントアカウントをモニタリングアカウントにリンクするためのオプションが 2 つあります。組織レベルの承認とアカウントレベルの承認です。オプションごとに、承認用の AWS CloudFormation テンプレートをダウンロードするか、各アカウントを個別に承認するかを選択できます。 2. 簡単化するために、任意のアカウントを選択して各アカウントレベルで承認します。このオプションでは、アカウントの承認リンクが表示されます。 3. [URL をコピーする] を選択して、リンクをコピーします。 4. 各ソースアカウントにサインインします。 5. ブラウザウィンドウにリンクを貼り付け、そして [シンクに接続するリンクを承認] を選択します。 6. 追加のソースアカウントにも同じ手順を繰り返します。 詳細については、CloudWatch ドキュメントの「モニタリン	

タスク	説明	必要なスキル
	グアカウントとソースアカウントをリンクする」を参照してください。	

クロスアカウントオブザーバビリティデータを検証

タスク	説明	必要なスキル
クロスアカウントデータを表示します。	1. モニタリングアカウントにサインインします。 2. CloudWatch コンソールを開きます。 3. 左ナビゲーションペインで、クロスアカウントログ、メトリクス、トレースを表示するオプションを選択します。	AWS DevOps、クラウド管理者、クラウドアーキテクト

(オプション) ソースアカウントがモニタリングアカウントを信頼できるようにする

タスク	説明	必要なスキル
他のアカウントのメトリクス、ダッシュボード、ログ、ウィジェット、アラームを表示します。	追加の特徴量として、CloudWatch メトリクス、ダッシュボード、ログ、ウィジェット、アラームを他のアカウントと共有できます。各アカウントは、CloudWatch-CrossAccountSharingRole と呼ばれる IAM ロールを使用して、このデータにアクセスします。	AWS DevOps、クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	中央モニタリングアカウントと信頼関係にあるソースアカウントは、このロールを引き受け、モニタリングアカウントのデータを表示できます。 CloudWatch には、ロールを作成するためのサンプル CloudFormation スクリプトが用意されています。[IAM でロールを管理する] を選択し、データを表示したいアカウントでこのスクリプトを実行します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": [ "arn:aws:iam::XXXXXXXXXX:root", "arn:aws:iam::XXXXXXXXXX:root", "arn:aws:iam::XXXXXXXXXX:root", "arn:aws:iam::XXXXXXXXXX:root" ] }},`	

タスク	説明	必要なスキル

	<pre>"Action":
	"sts:AssumeRole"
	}
	]
	}</pre>
	詳細については、CloudWatch ドキュメントの「CloudWatch でのクロスアカウント機能の有効化」を参照してください。 CloudWatch	

(オプション) モニタリングアカウントから、クロスアカウント/クロスリージョンを表示

タスク	説明	必要なスキル
クロスアカウント、クロスリージョンアクセスを設定します。	中央監視アカウントでは、オプションでアカウントセレクターを追加して、認証なしで簡単にアカウントを切り替えたり、そのデータを表示したりできます。 1. モニタリングアカウントにサインインします。 2. CloudWatch コンソールを開きます。 3. 左のナビゲーションペインの [設定] を選択します。 4. クロスアカウント/クロスリージョンを表示 セクションで、設定 を選択します。 5. 有効化を選択し、セレクターをコンソールに表	AWS DevOps、クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	示チェックボックスを選択します。 6. 以下のいずれかのオプションを選択します： • アカウント ID 入力: このオプションでは、クロスアカウントデータを表示するためにアカウントを変更するたびに、手動でアカウント ID を入力するように提示します。 • AWS Organization アカウントセレクタ：CloudWatch を と統合 AWS Organizationsしている場合、このオプションは組織内のアカウントの完全なリストを含むドロップダウンセレクタを提供します。 • カスタムアカウントセレクター: このオプションでは、アカウント ID のリストを手動で入力してセレクターに入力できます。 7. [Save changes] (変更の保存) をクリックします。 ステップや情報については、CloudWatch ユーザーガイドの「クロスアカウントクロスリージョン CloudWatch	

タスク	説明	必要なスキル
	コンソール」を参照してください。	

関連リソース

- 「CloudWatch クロスアカウントオブザーバビリティ」 (Amazon CloudWatch ドキュメント)
- 「Amazon CloudWatch オブザーバビリティアクセスマネージャー API リファレンス」 (Amazon CloudWatch ドキュメント)
- 「リソース:aws_oam_sink」 (テラフォームドキュメント)
- 「データソース: aws_oam_link」 (テラフォームドキュメンテーション)
- CloudWatchObservabilityAccessManager (AWS Boto3 ドキュメント)

起動時に EC2 インスタンスに必須タグが欠けていないか確認する

作成者：Susanne Kangnoh (AWS) と Archit Mathur (AWS)

概要

Amazon Elastic Compute Cloud (Amazon EC2) は、アマゾン ウェブ サービス (AWS) クラウドでスケーラブルなコンピューティングキャパシティーを提供します。Amazon EC2 の使用により、ハードウェアに事前投資する必要がなくなり、アプリケーションをより速く開発およびデプロイできます。

タグを使用して、さまざまな方法で AWS リソースを分類できます。EC2 インスタンスのタグ付けは、アカウントに多数のリソースがあり、タグに基づいて特定のリソースをすばやく識別する場合に役立ちます。タグを使用すると、EC2 インスタンスにカスタムメタデータを割り当てることができます。各タグは、ユーザー定義のキーと値で構成されます。組織の要件に適合する一連の一貫したタグを作成することをお勧めします。

このパターンは、EC2 インスタンスで特定のタグをモニタリングするのに役立つ AWS CloudFormation テンプレートを提供します。このテンプレートは、AWS CloudTrail の TagResource イベントまたは UntagResource イベントを監視する Amazon CloudWatch Events イベントを作成して、新しい EC2 インスタンスのタグ付けまたはタグの削除を検出します。定義済みのタグが欠けている場合、AWS Lambda 関数を呼び出し、Amazon Simple Notification Service (Amazon SNS) を使用して指定のメールアドレスに違反メッセージを送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 提供される Lambda コードをアップロードする Amazon Simple Storage Service (Amazon S3) バケット。
- 違反の通知を受信する E メールアドレス

機能制限

- このソリューションは、CloudTrail TagResource イベントまたは UntagResource イベントをサポートします。他のイベントの通知は作成されません。
- このソリューションはタグキーのみをチェックします。キー値はモニタリングしません。

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- AWS CloudFormation テンプレートは、さまざまな AWS リージョンとアカウントに複数回使用できます。各リージョンまたはアカウントで、テンプレートを 1 回実行するだけで済みます。

ツール

サービス

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) は、クラウド内で安全で再サイズを変更できるコンピューティング性能を提供するウェブサービスです。ウェブスケールのクラウドコンピューティングを開発者が簡単に利用できるように設計されています。
- AWS CloudTrail - CloudTrail は、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査に役立つ AWS のサービスです。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。CloudWatch Events は、運用上の変更が生じると同時にそれらを認識し、環境に応答するためのメッセージを送信する、機能をアクティブ化する、変更を行う、および状態情報を収集することによって、必要に即した是正措置を講じます。
- AWS Lambda – Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- 「Amazon S3」 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS)は、アプリケーション、エンドユーザー、およびデバイスでクラウドから通知を瞬時に送受信できるようにするウェブサービスです。

コード

このパターンでは、次の 2 つのファイルを含む添付ファイルを使用します。

- `index.zip`は、このパターンの Lambda コードを含む圧縮ファイルです。
- `ec2-require-tags.yaml` は、Lambda コードをデプロイする CloudFormation テンプレートです。

これらのファイルの使用方法については、「エピック」セクションを参照してください。

エピック

Lambda コードをデプロイします。

タスク	説明	必要なスキル
S3 バケットにコードをアップロードします。	新しい S3 バケットを作成するか、既存の S3 バケットを使用して、添付 `index.zip` ファイル (Lambda コード) をアップロードします。このバケットは、モニタリング対象のリソース (EC2 インスタンス) と同じ AWS リージョンに存在する必要があります。	クラウドアーキテクト
CloudFormation テンプレートをデプロイする。	S3 バケットと同じ AWS リージョンで Cloudformation コンソールを開き、添付ファイルで提供されている `ec2-require-tags.yaml` ファイルをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を入力します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。この S3 バケットには Lambda コードの .zip ファイルが含まれており、CloudFormation のテンプレートおよびモニタリング対象のリソースと同じ AWS リージョンに存在する必要があります。	クラウドアーキテクト
S3 キーを入力します。	S3 バケット内の Lambda コードの.zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例：`index.zip` または `controls/index.zip`)。	クラウドアーキテクト
メールアドレスを入力します。	違反の通知を受信するメールアドレスを入力します。	クラウドアーキテクト
ロギングレベルを定義する。	ロギングレベルと冗長性を指定します。`Info` はアプリケーションの進行状況に関する詳細な情報メッセージを指定するもので、デバッグにのみ使用してください。`Error` は、アプリケーションの実行を継続できるエラーイベントを指定します。`Warning` は潜在的に有害な状況を示します。	クラウドアーキテクト
必要なタグキーを入力します。	確認するタグキーを入力します。複数のキーを指定する場	クラウドアーキテクト

タスク	説明	必要なスキル
	合は、スペースを入れずにカンマで区切ります。(たとえば、`ApplicationId,CreatedBy,Environment,Organization` は 4 つのキーを検索します)。CloudWatch Events イベントはこれらのタグキーを検索し、見つからない場合は通知を送信します。	

サブスクリプションを確認

タスク	説明	必要なスキル
メールサブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定した E メールアドレスにサブスクリプション E メールメッセージが送信されます。通知を受信するには、この E メールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「バケットの作成」(Amazon S3 ドキュメント)
- 「オブジェクトのアップロード」(Amazon S3 ドキュメント)
- 「Amazon EC2 リソースのタグ付け」(Amazon EC2 ドキュメント)
- CloudTrail を使用して API コールでトリガーする CloudWatch Events ルールの作成 (Amazon CloudWatchドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Session Manager を使用して Amazon EC2 インスタンスに接続

作成者: Jason Cornick (AWS), Abhishek Bastikoppa (AWS), 及び Yaniv Ron (AWS)

概要

このパターンでは、AWS Systems Manager の機能である セッションマネージャーを使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに接続する方法について説明します。このパターンを使用して、ウェブブラウザから EC2 インスタンスで bash コマンドを実行できます。Session Manager では、インバウンドポートを開く必要はなく、また EC2 インスタンスのパブリック IP アドレスも必要ありません。さらに、さまざまな Secure Shell (SSH) キーでの要塞ホストの管理を不要にします。Session Manager のアクセスを AWS 識別とアクセス管理(IAM) ポリシーで管理し、インスタンスアクセスやアクションなどの重要な情報を記録するログを設定します。

このパターンでは、IAM ロールを設定し、それをAmazon マシンイメージ (AMI) を使用してプロビジョニングする Linux EC2 インスタンスに関連付けます。次に、Amazon CloudWatch Logsでロギングを設定し、セッションマネージャーを使用して、インスタンスとのセッションを開始します。

このパターンでは、Amazon Web Services (AWS) クラウドの Linux EC2 インスタンスに接続しますが、この方法を使用して、オンプレミスサーバーや他の仮想マシンなどの他のサーバーとの接続に Session Manager を使用することもできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- マネージド型ノードにアクセスする権限。手順については、「マネージド型ノードへのユーザーセッションアクセスの制御」 を参照してください。
- `ssm`、`ec2`、`ec2messages`、`ssmmessages`、および `s3` の VPC エンドポイント。手順については、Systems Manager ドキュメントの「VPC エンドポイントの作成」 を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- セッションマネージャー
- Amazon EC2

- CloudWatch ログ

ターゲットアーキテクチャ

1. ユーザーは IAM を通じて 自分のID と認証情報を確認7します。
2. ユーザーはSession Managerから SSH セッションを開始し、EC2 インスタンスに API 呼び出しを送信します。
3. EC2 インスタンスにインストールされている AWS Systems Manager SSM エージェントは、Session Managerに接続してコマンドを実行します。
4. 監査とモニタリングの目的で、Session Managerはロギングデータを CloudWatch ログに送信します。あるいは、Amazon Simple Storage Service (Amazon S3) バケットに、ログデータを送信することもできます。詳細については、「Amazon S3 を使用したセッションデータの口グ作成」 (システムマネージャードキュメント)を参照してください。

ツール

AWS サービス

- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、 AWS からのログを一元化することを支援して、ログを監視して安全にアーカイブできるようにします。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。このパターンは、Amazon マシンイメージ (AMI) を使用して、Linux EC2 インスタンスをプロビジョニングします。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。このパターンは、システムマネージャーの機能である「Session Manager」 を使用します。

ベストプラクティス

AWS Well-Architected フレームワークの「セキュリティの柱」について、詳しく読み、暗号化オプションを検討し、「Session Managerのセットアップ」（システムマネージャードキュメント）のセキュリティ推奨事項を適用することを推奨します。

エピック

インフラストラクチャを設定します。

タスク	説明	必要なスキル
IAM ロールを作成します。	SSM エージェントの IAM ロールを作成します。「AWS サービスのロールの作成」(IAM ドキュメント)の手順に従い、次のことに注意します： 1. AWSサービスで [EC2] を選択します。 2. [許可] で、`AmazonSSMManagedInstanceCore` を選択します。 3. [ロール名] に `EC2_SSM_Role` を入力します。	AWS システム管理者
EC2 インスタンスを作成します。	1. EC2 インスタンスを作成します。「インスタンスの起動」(Amazon EC2 ドキュメント)の手順に従い、次のことに注意します： a. [名前とタグ] セクションで、[その他のタグを追加] を選択します。[キー] に「`Name`」と入力し、[値] に「`Productio`	AWS システム管理者

タスク	説明	必要なスキル
	n_Server_One 」と入力します。 b. SSM エージェントがプリインストールされている Amazon Linux AMI を選択します。完全なリストについては、「SSM エージェントがプリインストールされている AMI」(システムマネージャードキュメント)」を参照してください。 c. [詳細情報] セクションで、IAM インスタンスプロファイルで、[EC2_SSM_Role] を選択します。 2. Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/) を開きます。 3. ナビゲーションペインで、[Fleet Manager] を選択します。 4. インスタンスがマネージドノードのリストに表示されていることを検証します。	

タスク	説明	必要なスキル
ログ記録をセットアップする。	1. CloudWatch Logs にロググループを作成します。「ロググループの作成」(CloudWatch ログドキュメント)」の手順に従います。新しいロググループ`SessionManager` に名前をつけます。 2. Session Managerのログを設定します。「Amazon CloudWatch Logs を使用してセッションデータをログに記録」（システムマネージャードキュメント）の手順に従い、次のことに注意してください： a. 暗号化済みの CloudWatch ロググループにのみ許可 を選択しないでください。 b. [リストからロググループを選択] で、[SessionManager] を選択します。	AWS システム管理者

インスタンスに接続する

タスク	説明	必要なスキル
EC2 インスタンスに接続します。	1. Systems Manager コンソールでセッションを開始する方法 。手順については、「セッションを開始する	AWS システム管理者

タスク	説明	必要なスキル
	」(Systems Manager のドキュメント)を参照してください。ターゲットインスタンスの場合、Production_Server_One インスタンスの左側にあるオプションボタンを選択します。 2. 接続が完了した後に、いくつかの bash コマンドを実行します。 3. システムマネージャーコンソールで、セッションを終了します。手順については、「セッションの終了」(システムマネージャードキュメント) を参照してください。	
ロギングを検証する。	1. CloudWatch Logs で、ロググループのログストリームを開きます。説明については、「ログデータを表示」(CloudWatch ログドキュメント)を参照してください。 2. ログデータに、前の記事で実行したコマンドが一覧表示されていることを確認します。	AWS システム管理者

トラブルシューティング

問題	ソリューション
IAM に関する問題	サポートについては、「トラブルシューティング」(IAM ドキュメント)」を参照してください。

関連リソース

- 「Session Managerの前提条件を完了」 (システムマネージャードキュメント)
- 「 Amazon CloudWatchでログとモニタリングの設計と実行 」(AWS 規範ガイダンス)

AWS CodePipeline に適用されない AWS リージョンにパイプラインを作成

作成者：Anand Krishna Varanasi (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

AWS CodePipeline は、Amazon Web Services (AWS) の一連の DevOps ツールの一部である継続的デリバリー (CD) オーケストレーションサービスです。さまざまなソース (バージョン管理システムやストレージソリューションなど)、AWS や AWS パートナーの継続的インテグレーション (CI) 製品およびサービス、オープンソース製品と統合して、アプリケーションとインフラストラクチャを迅速にデプロイするためのエンドツーエンドのワークフローサービスを提供します。

ただし、CodePipeline はすべての AWS リージョンに適用されなく、AWS CI/CD サービスを接続する目に見えないオーケストレーターがあると便利です。このパターンでは、CodePipeline がまだ適用されない AWS リージョンで AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy などの AWS CI/CD サービスを使用してエンドツーエンドのワークフローパイプラインを実装する方法を示しています。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Cloud Development Kit (AWS CDK) CLI バージョン 2.28 以降

アーキテクチャ

ターゲットテクノロジースタック

次の図表では、アフリカ (ケープタウン) リージョンなど、CodePipeline が適用されないリージョンで作成されたパイプラインを示しています。開発者は CodeDeploy 設定ファイル (デプロイライフサイクルフックスクリプト)とも呼ばれる) を CodeCommit がホストする Git リポジトリにプッシュします。(このパターンで提供されている GitHub リポジトリ を参照してください。) Amazon EventBridge ルールは、CodeBuild を自動で開始します。

CodeDeploy 設定ファイルは、パイプラインのソースステージの一部として CodeCommit から取得され、CodeBuild に転送されます。

次のフェーズで、CodeBuild は次のタスクを実行します。

1. アプリケーションのソースコードの TAR ファイルをダウンロードします。AWS Systems Manager のキャパシティとしての パラメータストアを使用して、このファイルの名前を設定できます。
2. CodeDeploy 設定ファイルをダウンロードします。
3. アプリケーションソースコードと、アプリケーションタイプに固有の CodeDeploy 設定ファイルを組み合わせたアーカイブを作成します。
4. Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに対する CodeDeploy デプロイをセットアップします。

ツール

AWS サービス

- AWS CodeBuild は完全マネージド型の構築サービスです。ソースコードのコンパイル、ユニットテストの実行、すぐにデプロイできるアーティファクトの生成を行います。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodeDeploy は、Amazon EC2 またはオンプレミスインスタンス、AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) に対するデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。

コード

このパターンのコードは、GitHub 内の「CodePipeline 適用されないリージョン」リポジトリで利用できます。

エピック

開発者ワークステーションをセットアップ

タスク	説明	必要なスキル
AWS CLI のインストール。	手順については、AWS CDKのドキュメントを参照してください。	AWS DevOps
SQL クライアントをインストールします。	ローカルコンピュータにインストールされた Git クライアントを使用してコミットを作成してから、それらのコミットを CodeCommit リポジトリにプッシュできます。Git クライアントで CodeCommit をセットアップするには、CodeCommit のドキュメント を参照してください。	AWS DevOps
npm をインストールします。	npm のパッケージマネージャーをインストールします。詳細については、npmドキュメントを参照してください。	AWS DevOps

パイプラインのセットアップ

タスク	説明	必要なスキル
コードリポジトリを複製します。	以下のコマンドを実行して、GitHub CodePipeline 適用されないリージョン のリポジトリをローカルマシンにクローンします。	DevOps エンジニア

タスク	説明	必要なスキル
	`git clone https://github.com/aws-samples/invisible-codepipeline-unsupported-regions`	

タスク	説明	必要なスキル
cdk.json にパラメーターを設定します。	cdk.json セクションで、次のパラメーターの値を指定します。 `"pipeline_account" :"XXXXXXXXXXXX", "pipeline_region":" us-west-2", "repo_name": "app-dev-repo", "ec2_tag_key": "test-vm", "configName" : "cbdeployconfig", "deploymentGroupName": "cbdeploygroup", "applicationName" : "cbdeployapplication", "projectName" : "CodeBuildProject"` 各パラメータの意味は次のとおりです。 • pipeline_account はパイプラインが構築される AWS アカウントです。 • pipeline_region はパイプラインが構築される AWS リージョンです。 • コードリポジトリの名前 (など)。 • ec2_tag_key はコードをデプロイする EC2 インスタンスにアタッチされたタグです。	AWS DevOps

タスク	説明	必要なスキル
	• `configName` はCodeDeploy コンフィギュレーションファイルの名前です • CodeDeploy デプロイグループの名前。 • `applicationName` は CodeDeploy アプリケーション名です。 • `projectName` はCodeBuild プロジェクト名です。	

AWS CDK コンストラクトライブラリをセットアップします。	複製された GitHub リポジトリで、次のコマンドを使用して AWS CDK 構築ライブラリをインストールし、アプリケーションをビルドし、合成してアプリケーションの AWS CloudFormation テンプレートを生成します。
	`npm i aws-cdk-lib`
	`npm run build`
	`cdk synth`
		AWS DevOps

サンプルアプリケーションをデプロイするには	適用されないリージョン (`af-south-1` など)で次のコマンドを実行してコードをデプロイします。 `cdk deploy`	AWS DevOps

CodeDeploy の CodeCommit リポジトリをセットアップします

タスク	説明	必要なスキル
アプリケーションの CI/CD をセットアップします。	cdk.json ファイルに指定された CodeCommit リポジトリを複製して (デフォルトで app-dev-repo と呼ばれ)、アプリケーションの CI/CD パイプラインを設定します。 `git clone https://git-codecommit.us-west-2.amazonaws.com/v1/repos/app-dev-repo` リポジトリ名とリージョンは、cdk.json ファイルに入力した値によって異なります。	AWS DevOps

パイプラインを削除します。

タスク	説明	必要なスキル
デプロイインストラクションを使用してパイプラインをテストします。	GitHub CodePipelineサポートされていないリージョン リポジトリの CodeDeploy_Files フォルダには、CodeDeploy にアプリケーションをデプロイするように指示するサンプルファイルが含まれています。appspec.yml ファイルは、アプリケーションデプロイのフローを制御するフックを含む CodeDeploy 設定ファイ	AWS DevOps

タスク	説明	必要なスキル
	ルです。サンプルファイル `index.html` 、`start_server.sh` 、`stop_server.sh` 、及び `install_dependencies.sh` を使用して、Apache でホストされているウェブサイトを更新します。これらは例です。GitHub リポジトリのコードを使用して、あらゆるタイプのアプリケーションをデプロイします。ファイルが CodeCommit リポジトリにプッシュされる場合、非表示のパイプラインが自動的に開始されます。デプロイ結果については、CodeBuild コンソールと CodeDeploy コンソールの個々のフェーズの結果を確認します。	

関連リソース

- 開始方法 (AWS CDK ドキュメント)
- Cloud Development Kit (CDK) の紹介 (AWS ワークショップスタジオ)
- AWS CDK ワークショップ

AWS CDK アスペクトとエスケープハッチを使用してデフォルトのロール名をカスタマイズする

作成者: SANDEEP SINGH (AWS) と James Jacob (AWS)

概要

このパターンは、 AWS Cloud Development Kit (AWS CDK) コンストラクトによって作成されるロールのデフォルト名をカスタマイズする方法を示しています。組織で命名規則に基づいて特定の制約がある場合、ロール名のカスタマイズが必要になることがよくあります。たとえば、組織で AWS Identity and Access Management (IAM) アクセス許可の境界や、ロール名に特定のプレフィックスを必要とするサービスコントロールポリシー (SCPs) を設定することができます。このような場合、AWS CDK コンストラクトによって生成されたデフォルトのロール名は、これらの規則を満たさない場合があり、変更する必要がある場合があります。このパターンは、エスケープハッチと の側面を使用して、これらの要件に対処します AWS CDK。エスケープハッチを使用してカスタムロール名を定義し、すべてのロールにカスタム名を適用する側面を定義して、組織のポリシーと制約を確実に順守します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS CDK ドキュメントで指定されている前提条件

制約事項

- アスペクトはリソースタイプに基づいてリソースをフィルタリングするため、すべてのロールが同じプレフィックスを共有します。ロールごとに異なるロールプレフィックスが必要な場合は、他のプロパティに基づく追加のフィルタリングが必要です。たとえば、 AWS Lambda 関数に関連付けられたロールに異なるプレフィックスを割り当てるには、特定のロール属性またはタグでフィルタリングし、Lambda 関連のロールに 1 つのプレフィックスを、他のロールに別のプレフィックスを適用できます。
- IAM ロール名の最大長は 64 文字であるため、この制限を満たすには、変更されたロール名をトリミングする必要があります。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。特定のエンドポ

イントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ターゲットテクノロジースタック

- AWS CDK
- AWS CloudFormation

ターゲット アーキテクチャ

- AWS CDK アプリケーションは 1 つ以上の AWS CloudFormation スタックで構成され、 AWS リソースを管理するために合成およびデプロイされます。
- レイヤー 2 (L2) コンストラクトによって公開されていない AWS CDKマネージドリソースのプロパティを変更するには、エスケープハッチを使用して基盤となる CloudFormation プロパティ (この場合はロール名) を上書きし、 の側面を使用して AWS CDK スタック合成プロセス中に AWS CDK アプリケーション内のすべてのリソースにロールを適用します。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK コマンドラインインターフェイス (AWS CDK CLI) (AWS CDK Toolkit とも呼ばれます) は、 AWS CDK アプリの操作に役立つコマンドラインクラウド開発キットです。CLI `cdk` コマンドは、 AWS CDK アプリを操作するための主要なツールです。アプリケーションを実行し、定義したアプリケーションモデルを調査し、 によって生成された CloudFormation テンプレートを生成してデプロイします AWS CDK。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。

コードリポジトリ

このパターンのソースコードとテンプレートは、GitHub CDK Aspects Override リポジトリで入手できます。

ベストプラクティス

TypeScript AWS CDK で を使用して、 規範ガイダンスウェブサイトのIaC プロジェクトを作成するためのベストプラクティス」を参照してください。 AWS

エピック

CLI AWS CDK のインストール

タスク	説明	必要なスキル
CLI AWS CDK をインストールします。	CLI AWS CDK をグローバルにインストールするには、 コマンドを実行します。 `npm install -g aws-cdk`	AWS DevOps
バージョンを確認します。	コマンドを実行します。 `cdk --version` CLI のバージョン 2 AWS CDK を使用していることを確認します。	AWS DevOps
AWS CDK 環境をブートストラップします。	AWS CloudFormation テンプレートをデプロイする前に、使用する アカウントと AWS リージョン を準備します。 コマンドを実行します。 `cdk bootstrap <account>/<Region>`	AWS DevOps

タスク	説明	必要なスキル
	詳細については、 AWS ドキュメントのAWS CDK「ブートストラップ」を参照してください。	

AWS CDK アプリをデプロイして側面の使用をデモンストレーションする

タスク	説明	必要なスキル

プロジェクトをセットアップします。	1. このパターンの GitHub リポジトリをローカルコンピュータにクローンします。
	`git clone https://github.com/aws-samples/cdk-aspects-override`
	2. ローカルコンピュータのプロジェクトディレクトリに移動します。
	3. プロジェクトの依存関係をインストールします。
	`npm ci`
		AWS DevOps

によって割り当てられたデフォルトのロール名を使用してスタックをデプロイします AWS CDK。	Lambda 関数と関連するロールを含む 2 つの CloudFormation スタック (`ExampleStack1` および `ExampleStack2`) をデプロイします。	AWS DevOps

タスク	説明	必要なスキル
	`npm run deploy:ExampleAppWithoutAspects` コードはロールプロパティを明示的に渡さないため、ロール名は によって作成されます AWS CDK。 出力例については、「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル

側面を持つスタックをデプロイします。	このステップでは、 AWS CDK プロジェクトにデプロイされているすべての IAM ロールにプレフィックスを追加して、ロール名の規則を適用する側面を適用します。アスペクトは lib/aspects.ts ファイルで定義されます。この側面では、エスケープハッチを使用して、プレフィックスを追加してロール名を上書きします。アスペクトは、 bin/app-with-aspects.ts ファイルのスタックに適用されます。この例で使用されるロール名のプレフィックスは ですdev-unicorn 。
	
	1. bin/app-with-aspects.ts ファイルを編集します。
	2. ファイルで、プレフィックス ROLE_NAME_PREFIX を使用して変数を更新しますdev-unicorn 。
	
	``
	const app = new cdk.App();
	
	// Define a prefix for the role names
	const ROLE_NAME_PREFIX = 'dev-unicorn';
	``
		AWS DevOps

タスク	説明	必要なスキル

	`// Instantiate the RoleNamingConventionAspect with the desired prefix const roleNamingConventionAspect = new RoleNamingConventionAspect(ROLE_NAME_PREFIX);`
	3. 以下の側面で AWS CDK アプリケーションをデプロイします。
	`npm run deploy:ExampleAppWithAspects`
	出力例については、「追加情報」セクションを参照してください。	

リソースをクリーンアップする

タスク	説明	必要なスキル
AWS CloudFormation スタックを削除します。	このパターンの使用が完了したら、次のコマンドを実行してリソースをクリーンアップし、追加コストが発生しないようにします。 `cdk destroy --all -f && cdk --app npx ts-node bin/app-with-aspects.ts' destroy --all -f`	AWS DevOps

トラブルシューティング

問題	ソリューション
を使用して問題が発生する AWS CDK。	AWS CDK ドキュメントの「一般的なAWS CDK 問題のトラブルシューティング」を参照してください。

関連リソース

- AWS Cloud Development Kit (AWS CDK)
- AWS CDK ドキュメント
- AWS CDK GitHub の
- ハッチのエスケープ
- 側面と AWS CDK

追加情報

側面 AWS CloudFormation なしで によって作成されたロール名

```
Outputs:
ExampleStack1WithoutAspects.Function1RoleName = example-stack1-without-as-
Function1LambdaFunctionSe-y7FYTY6FXJXA
ExampleStack1WithoutAspects.Function2RoleName = example-stack1-without-as-
Function2LambdaFunctionSe-dDZV4rkWqWnI
...

Outputs:
ExampleStack2WithoutAspects.Function3RoleName = example-stack2-without-as-
Function3LambdaFunctionSe-ygMv49iTyMq0
```

側面 AWS CloudFormation を持つ によって作成されたロール名

```
Outputs:
ExampleStack1WithAspects.Function1RoleName = dev-unicorn-
Function1LambdaFunctionServiceRole783660DC
ExampleStack1WithAspects.Function2RoleName = dev-unicorn-
Function2LambdaFunctionServiceRole2C391181
```

```
...

Outputs:
ExampleStack2WithAspects.Function3RoleName = dev-unicorn-
Function3LambdaFunctionServiceRole4CAA721C
```

Amazon EC2 にプライベート静的 IP を使用して Cassandra クラスターをデプロイしてリバランスを回避する

ディピン・ジェイン (AWS) によって作成されました

概要

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのプライベート IP は、そのライフサイクルを通じて保持されます。ただし、プライベート IP は、Amazon マシンイメージ (AMI) のアップグレード中など、計画的または予期しないシステムクラッシュ時に変更される可能性があります。シナリオによっては、プライベートの静的 IP を保持することで、ワークロードのパフォーマンスと復旧時間を向上させることができます。たとえば、Apache Cassandra シードノードに静的 IP を使用すると、クラスターにリバランスのオーバーヘッドが発生するのを防ぐことができます。

Amazon EC2 インスタンスの Cassandra クラスターにセカンダリのelastic network interface をアタッチして、リホスト中に IP を静的に保つ方法。このパターンは Cassandra クラスターに焦点を当てていますが、この実装はプライベートの静的 IP の利点を活用するあらゆるアーキテクチャに使用できます。

前提条件と制限

前提条件

- Amazon Web Services (AWS) アカウント。

製品バージョン

- DataStax バージョン 5.11.1
- オペレーティングシステム： (Ubuntu 18.04 LTS)

アーキテクチャ

ソースアーキテクチャ

ソースは、オンプレミスの仮想マシン (VM) 上の Cassandra クラスターでも、AWS クラウドの EC2 インスタンスでもかまいません。このシナリオを以下に図表で示します。この例には、3 つのシードノードと 1 つの管理ノードの 4 つのクラスターノードが含まれています。ソースアーキテクチャでは、各ノードに 1 つのネットワークインターフェースが接続されています。

ターゲット アーキテクチャ

デスティネーションクラスターは、次の図に示すように、各ノードにセカンダリ elastic network interface がアタッチされた EC2 インスタンスでホストされます。

自動化とスケール

AWS Knowledge Center のビデオ で説明されているように、2 つ目のelastic network interface を EC2 Auto Scaling グループに自動的にアタッチすることもできます。

エピック

Amazon EC2上のカサンドラクラスターを設定します。

タスク	説明	必要なスキル
EC2 ノードを起動して Cassandra クラスターをホストします。	Amazon EC2 コンソール で、AWS アカウントの Ubuntu ノード用に 4 つの EC2 インスタンスを起動します。Cassandraクラスターには3つの (シード) ノードが使用され、4番目のノードは DataStax Enterprise (DSE) OpsCenterをインストールするクラスター管理ノードとして機能します。手順については、Amazon EC2 のドキュメントを参照してください。	クラウドエンジニア
ノード通信を確認する。	4 つのノードがデータベースとクラスター管理ポートを介して相互に通信できることを確認します。	ネットワークエンジニア

タスク	説明	必要なスキル
DSE OpsCenterを管理ノードにインストールします。	管理ノードのDebianパッケージからDSE OpsCenter 6.1をインストールします。手順については、DataStax のドキュメントを参照してください。	DBA

タスク	説明	必要なスキル
セカンダリネットワークインターフェイスを設定する	Cassandra は、そのノードの EC2 インスタンスの IP アドレスに基づいて、各ノードの汎用固有識別子 (UUID) を生成します。この UUID はリング上の仮想ノード (vnode) を配布するために使用されます。Cassandra を EC2 インスタンスにデプロイすると、インスタンスが作成されると自動的に IP アドレスが割り当てられます。 計画的または予期しない停止が発生した場合、新しい EC2 インスタンスの IP アドレスとデータ配信が変更され、リング全体のバランスを調整する必要があります。これは望ましくありません。割り当てられた IP アドレスを保持するには、固定 IP アドレスの セカンダリ elastic network interface を使用してください。 1. Amazon EC2 コンソール で、ネットワークインターフェイス、ネットワークインターフェイスの作成を選択します。 2. サブネット,に EC2 インスタンスを作成した サブネットを選択します。	クラウドエンジニア

タスク	説明	必要なスキル
	3. プライベート IPv4 アドレスが自動的に割り当てられます。 4. Private IPv4 アドレスは、セキュリティグループを選択し、ネットワークインターフェイスの作成 を選択します。 ネットワークインターフェイスの作成に関する詳細については、Amazon EC2 ドキュメント を参照してください。	
セカンダリネットワークインターフェースをクラスターノードにアタッチします。	1. Amazon EC2 コンソールで、インスタンス を選択します。 2. 前に作成した EC2 インスタンスのチェックボックスをオンにします。 3. アクション、ネットワーク、ネットワークインターフェイスの添付 の順に選択します。 4. 前のステップで作成したネットワークインターフェイスをオンにし、添付.を選択します。 ネットワークインターフェイスの作成に関する詳細については、Amazon EC2 ドキュメント を参照してください。	クラウドエンジニア

タスク	説明	必要なスキル
Amazon EC2 にルートを追加して、非対称ルーティングに対処します。	2 つ目のネットワークインターフェースをアタッチすると、ネットワークは非対称ルーティングを実行する可能性が非常に高くなります。これを回避するには、新しいネットワークインターフェースにルートを追加します。 非対称ルーティングの詳細な説明と修正方法については、AWS ナレッジセンターのビデオ、またはマルチホームサーバーでの非対称ルーティングの克服 (Patrick McManus による Linux Journal の記事、2004 年 4 月 5 日) を参照してください。	ネットワークエンジニア
セカンダリネットワークインターフェイス IP を指すように DNS エントリを更新します。	ノードの完全修飾ドメイン名 (FQDN) をセカンダリネットワークインターフェイスの IP に設定します。	ネットワークエンジニア
DSE OpsCenterを使用してCassandraクラスターをインストールして構成します。	クラスター・ノードにセカンダリ・ネットワーク・インターフェースが用意できたら、Cassandra クラスターをインストールして構成できます。	DBA

ノード障害からクラスターを回復する

タスク	説明	必要なスキル
クラスターシードノード用の AMI を作成します。	ノードに障害が発生した場合にデータベースバイナリで復元できるように、ノードのバックアップを作成します。手順については、Amazon EC2 のドキュメントのAMIの作成を参照してください。	バックアップ管理者
ノード障害から回復します。	障害が発生したノードを AMI から起動した新しい EC2 インスタンスと交換し、障害が発生したノードのセカンダリネットワークインターフェイスを接続します。	バックアップ管理者
Cassandra クラスターが正常であることを確認します。	交換用ノードが起動したら、DSE OpsCenterでクラスターの状態を確認します。	DBA

関連リソース

- DebianパッケージからのDSE OpsCenter 6.1のインストール (DataStaxドキュメント)
- Ubuntu EC2 インスタンスでセカンダリネットワークインターフェイスを機能させる方法 (AWS ナレッジセンターの動画)
- Amazon EC2 で Apache カサンドラを実行するためのベストプラクティス (AWS ブログ記事)

AWS Transit Gateway Connect を使用して VRF を AWS に拡張する

作成者: Adam Till (AWS)、Yashar Araghi (AWS)、Vikas Dewangan (AWS)、Mohideen HajaMohideen (AWS)

概要

仮想ルーティングと転送 (VRF) は従来のネットワークの機能です。分離された論理ルーティングドメインをルートテーブル形式で使用して、同じ物理インフラストラクチャ内のネットワークトラフィックを分離します。オンプレミスネットワークを AWS に接続するときに VRF 分離をサポートするように AWS Transit Gateway を構成できます。このパターンでは、サンプルアーキテクチャを使用してオンプレミスの VRF をさまざまなトランジットゲートウェイルートテーブルに接続します。

このパターンは、AWS Direct Connect のトランジット仮想インターフェイス (VIF) と Transit Gateway の Connect アタッチメントを使用して VRF を拡張します。トランジット VIF は、Direct Connect ゲートウェイに関連付けられた 1 つまたは複数の Amazon VPC トランジットゲートウェイにアクセスするために使用されます。Transit Gateway Connect アタッチメントは、Transit Gateway を VPC で実行しているサードパーティの仮想アプライアンスと接続します。Transit Gateway Connect アタッチメントは、総称ルーティングカプセル化 (GRE)トンネルプロトコルをサポートして高パフォーマンスを実現し、動的ルーティングのためにボーダーゲートウェイプロトコル (BGP) をサポートします。

このパターンで説明するアプローチには、以下の利点があります。

- Transit Gateway Connect を使用すると、Transit Gateway Connect ピアに最大1,000のルートをアドバタイズし、そこから最大5,000のルートを受信できます。Transit Gateway Connect なしで Direct Connect トランジット VIF 機能を使用すると、Transit Gateway あたり 20 プレフィックスに制限されます。
- 顧客が使用している IP アドレススキーマに関係なくトラフィックの分離を維持し、Transit Gateway Connect を使用して AWS でホストされたサービスを提供できます。
- VRF トラフィックはパブリック仮想インターフェイスを通過する必要はありません。これにより、多くの組織のコンプライアンス要件やセキュリティ要件を簡単に遵守できます。
- 各 GRE トンネルは最大 5 Gbps をサポートし、Transit Gateway の Connect アタッチメントごとに最大 4 つの GRE トンネルを設定できます。これは、最大 1.25 Gbps をサポートする AWS Site-to-Site VPN 接続など、他の多くの接続タイプよりも高速です。

前提条件と制限

前提条件

- 必要な AWS アカウントが作成されていること (詳細についてはアーキテクチャを参照してください)
- アカウントごとに AWS Identity and Access Management (IAM) ロールを引き受ける権限。
- 各アカウントの IAM ロールには、AWS Transit Gateway と AWS Direct Connect リソースをプロビジョニングするためのアクセス権限が必要です。詳細については、「トランジットゲートウェイの認証とアクセス制御」および「Direct Connect のアイデンティティとアクセス管理」を参照してください。
- Direct Connect の接続が正常に作成されている。詳細については、「接続ウィザードを使用して接続を作成する」を参照してください。

制約事項

- プロダクション、QA、開発アカウントの VPC への Transit Gateway アタッチメントには制限があります。詳細については、「VPC への Transit Gateway アタッチメント」を参照してください。
- Direct Connect ゲートウェイの作成および使用には制限があります。詳細については、「AWS Direct Connect」を参照してください。

アーキテクチャ

ターゲットアーキテクチャ

以下のサンプルアーキテクチャは、Transit Gateway Connect アタッチメントを使用してトランジット VIF をデプロイするための再利用可能なソリューションを提供します。このアーキテクチャは、複数の Direct Connect ロケーションを使用することで耐障害性を実現します。詳細については、Direct Connect ドキュメントの「最大限の耐障害性」を参照してください。オンプレミスネットワークにはプロダクション、QA、開発用の VRF があり、これらは AWS に拡張され、専用のルートテーブルを使用して分離されます。

AWS 環境では、Direct Connect アカウントとネットワークハブアカウントの 2 つのアカウントが VRF の拡張専用です。Direct Connect アカウントには、各ルーターの接続とトランジット VIF が含まれています。トランジット VIF は Direct Connect アカウントから作成しますが、ネットワークハ

ブアカウントにデプロイして、ネットワークハブアカウントの Direct Connect ゲートウェイに関連付けることができます。ネットワークハブアカウントは、Direct Connect ゲートウェイを所有しています。AWS リソースは次のように接続されます。

1. トランジット VIF は、Direct Connect ロケーションのルーターを、Direct Connect アカウントの AWS Direct Connect に接続します。
2. トランジット VIF は、Direct Connect をネットワークハブアカウントの Direct Connect ゲートウェイに接続します。
3. トランジットゲートウェイアソシエーションは、Direct Connect ゲートウェイをネットワークハブアカウント内のトランジットゲートウェイに接続します。
4. Transit Gateway Connect アタッチメントは、Transit Gateway をプロダクション、QA、開発アカウントの VPC に接続します。

トランジット VIF アーキテクチャ

次の図は、トランジット VIF 構成の詳細を示しています。このサンプルアーキテクチャでは、トンネルソースに VLAN を使用していますが、ループバックを使用することもできます。

以下は、トランジット VIF の AS 番号 (ASN) などの構成詳細です。

リソース	項目	[Detail] (詳細)
ルーター 01	ASN	65534
ルーター 02	ASN	65534
ルーター 03	ASN	65534
ルーター 04	ASN	65534
Direct Connect ゲートウェイ	ASN	64601
トランジットゲートウェイ	ASN	64600
	CIDR ブロック	10.100.254.0/24

Transit Gateway Connect のアーキテクチャ

次の図と表は、Transit Gateway Connect アタッチメントを介して単一の VRF を構成する方法を示しています。VRF を追加する場合は、一意のトンネル ID、トランジットゲートウェイ GRE IP アドレス、および CIDR ブロック内の BGP を割り当てます。ピア GRE IP アドレスは、トランジット VIF のルーターピア IP アドレスと一致します。

次の表には、ルーター構成の詳細が記載されています。

ルーター	トンネル	IP アドレス	ソース	デスティネーション
ルーター 01	トンネル 1	169.254.101.17	VLAN 60 169.254.100.1	10.100.254.1
ルーター 02	トンネル 11	169.254.101.81	VLAN 61 169.254.100.5	10. 100.254.11「」
ルーター 03	トンネル 21	169.254.101.145	VLAN 62 169.254.100.9	10. 100.254.21「」
ルーター 04	トンネル 31	169.254.101.209	VLAN 63 169.254.100.13	10. 100.254.31「」

次の表には、トランジットゲートウェイ構成の詳細が記載されています。

トンネル	トランジットゲートウェイ GRE IP アドレス	ピア GRE IP アドレス	CIDR ブロック内の BGP
トンネル 1	10.100.254.1	VLAN 60 169.254.100.1	169.254.101.16/29
トンネル 11	10. 100.254.11「」	VLAN 61	169.254.101.80/29

		169.254.100.5	
トンネル 21	10. 100.254.21「」	VLAN 62	169.254.101.144/29
		169.254.100.9	
トンネル 31	10. 100.254.31「」	VLAN 63	169.254.101.208/29
		169.254.100.13	

デプロイメント

「エピック」セクションでは、 1 つの VRF に対する複数のカスタマールーターの構成例をデプロイする方法について説明します。ステップ 1 ~ 5 の完了後、AWS に拡張する新しい VRF ごとにステップ 6 ~ 7 を実行して、新しい Transit Gateway Connect アタッチメントを作成できます。

1. トランジットゲートウェイを作成します。
2. 各 VRF の Transit Gateway ルートテーブルを作成します。
3. トランジット仮想インターフェイスを作成します。
4. Direct Connect ゲートウェイを作成します。
5. Direct Connect ゲートウェイ仮想インターフェイスと、許可されたプレフィックスを持つゲートウェイアソシエーションを作成します。
6. Transit Gateway Connect アタッチメントを作成します。
7. Transit Gateway Connect ピアを作成します。
8. Transit Gateway Connect アタッチメントをルートテーブルに関連付けます。
9. ルートをルーターにアドバタイズします。

ツール

サービス

- AWS Direct Connect は、標準のイーサネット光ファイバーケーブルを介して内部ネットワークを Direct Connect の場所にリンクします。この接続を使用すると、ネットワークパスのインターネットサービスプロバイダーを回避してパブリック AWS サービスに対する仮想インターフェイスを直接作成できます。

- AWS Transit Gatewayは、仮想プライベートクラウド (VPC) とオンプレミスネットワークを接続する中央ハブです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

アーキテクチャを計画する

タスク	説明	必要なスキル
カスタムアーキテクチャ図を作成します。	1. 「添付ファイル」セクションで、ダイアグラムテンプレートをダウンロードします。 2. 添付の図を Microsoft Office PowerPoint で開きます。 3. 「アーキテクチャの概要」スライドで、使用中の環境に合わせてアーキテクチャ図をカスタマイズします。AWS 環境に拡張する必要があるオンプレミス VRF を特定します。 4. トランジット VIF スライドで、アーキテクチャ図をカスタマイズします。ルーター、Direct Connect ゲートウェイ、トランジットゲートウェイの AS 番号を特定します。トランジット VIF の両端にある IP アドレスを特定します。	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル
	5. 「Transit Gateway Connect」スライドで、VRF ごとにアーキテクチャ図をカスタマイズします。ルーターと Transit Gateway Connect ピアの構成に必要な IP アドレスをすべて特定します。	

Transit Gateway リソースの作成

タスク	説明	必要なスキル
トランジットゲートウェイを作成します。	1. ネットワークハブアカウントにサインインします。 2. 「トランジットゲートウェイの作成」の指示に従います。このパターンでは、次の点に注意してください。 • [Amazon side Autonomous System Number (ASN)] (Amazon 側の自律システム番号 (ASN)) に、一意の ASN を入力します。この例では、ASN は `64600` です。 • DNS サポートを選択します。 • このサンプルアーキテクチャでは、VPN ECMP サポート、デフォルトルートテーブルアソシエーション、デフォルトルート	ネットワーク管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	テーブルプロパゲーション、マルチキャストサポートは必要ありません。 • トランジットゲートウェイ CIDR ブロックには、トランジットゲートウェイの IPv4 CIDR ブロックを入力します。この例では、CIDR ブロックは `10.100.254.0/24` です。	
トランジットゲートウェイルートテーブルを作成します。	「トランジットゲートウェイルートテーブルの作成」の指示に従います。このパターンでは、次の点に注意してください。 • [名前タグ] に、トランジットゲートウェイルートテーブルの名前を提供します。VRF に対応する名前 (`routetable-dev-vrf` など) を使用することをお勧めします。 • [トランジットゲートウェイ ID] で、先に作成したトランジットゲートウェイを選択します。	クラウドアーキテクト、ネットワーク管理者

トランジット仮想インターフェイスの作成

タスク	説明	必要なスキル
トランジット仮想インターフェイスを作成します。	1. Direct Connect アカウントにサインインします。 2. 「Direct Connect ゲートウェイと接続するトランジット仮想インターフェイスを作成する」の指示に従います。このパターンでは、次の点に注意してください。 • [仮想インターフェイス名] に、トランジット VIF の名前を入力します。ルーターに対応する名前 (`transit-vif-router01` など) を使用することをお勧めします。 • [接続] には、`router-01` などのルーターを選択します。 • [仮想インターフェースの所有者] には、ネットワークハブアカウントのアカウント ID を入力します。手順については、「AWS アカウント ID を表示する」を参照してください。 • Direct Connect ゲートウェイの場合は、何も選択しないでください。Direct Connect ゲート	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル

	ウェイは次のステップで接続します。
	• VLAN には、ルーターの VLAN (60 など) を入力します。
	• BGP ASN の場合は、ルーターの ASN (65534 など) を入力します。
	• [追加設定] で、以下を実行します。
	  • [IPv4] を選択します。
	  • [ルーターピア IP] には、ルーターピア IP アドレス (169.254.100.1 など) を入力します。
	  • Amazon ルーターピア IP には、Amazon ルーターのピア IP (169.254.100.2 など) を入力します。
	  • BGP 認証キーにはパスワードが必要です。これを空白のままにすると、AWS はこのアカウントでのみアクセス可能なキーを作成します。
	3. これらの手順を繰り返して、VRF のすべてのトランジット VIF を作成します。	

Direct Connect のリソースの作成

タスク	説明	必要なスキル
Direct Connect ゲートウェイを作成します。	1. ネットワークハブアカウントにサインインします。 2. 「Direct Connect ゲートウェイの作成」の指示に従ってください。このパターンでは、次の点に注意してください。 • Amazon 側 ASN の場合は、Direct Connect ゲートウェイの ASN (64601 など) を入力します。 • 仮想プライベートゲートウェイは選択しないでください。	クラウドアーキテクト、ネットワーク管理者
Direct Connect ゲートウェイをトランジット VIF に接続します。	1. ネットワークハブアカウントで、AWS Direct Connect コンソール (https://console.aws.amazon.com/directconnect/v2/home) を開きます。 2. ナビゲーションペインで、[Virtual Interfaces] を選択します。 3. 新しいトランジット VIF を選択し、[承認] を選択します。 4. 作成した Direct Connect ゲートウェイを選択します。	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル
	5. これらの指示を各トランジット VIF に対して繰り返します。	
許可されたプレフィックスを使用して Direct Connect ゲートウェイアソシエーションを作成します。	ネットワークハブアカウントで、「トランジットゲートウェイを関連付ける方法」の指示に従います。このパターンでは、次の点に注意してください。 • [トランジットゲートウェイ ID] で、先に作成したトランジットゲートウェイを選択します。 • [許可するプレフィックス] には、トランジットゲートウェイに割り当てられた CIDR ブロック (`10.100.254.0/24` など) を入力します。 このアソシエーションを作成すると、Direct Connect Gateway リソースタイプの Transit Gateway アタッチメントが自動的に作成されます。このアタッチメントは、トランジットゲートウェイのルートテーブルに関連付ける必要はありません。	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル
Transit Gateway Connect アタッチメントを作成します。	1. ネットワークハブアカウントで、Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。 2. ナビゲーションペインで [Transit Gateway アタッチメント] を選択します。 3. [Transit Gateway アタッチメントの作成] を選択します。 4. [名前タグ] で、アタッチメントの名前を入力します。VRF に対応する名前 (PROD-VRF など) を使用することをお勧めします。 5. [トランジットゲートウェイ ID] で、先に作成したトランジットゲートウェイを選択します。 6. [アタッチメントタイプ] で、[接続] を選択します。 7. [トランスポートアタッチメント ID] には、先に作成した Direct Connect ゲートウェイを選択します。 8. [Transit Gateway アタッチメントの作成] を選択します。 9. 拡張する VRF ごとに、このステップを繰り返します。	クラウドアーキテクト、ネットワーク管理者

タスク	説明	必要なスキル
Transit Gateway Connect ピアを作成します。	1. ネットワークハブアカウントで、「Transit Gateway Connect ピア (GRE トンネル) を作成する」の指示に従います。このパターンでは、次の点に注意してください。 • [名前タグ] に Transit Gateway Connect ピアの名前を入力します。ルーターに対応する名前 (`connectpeer-router01` など) を使用することをお勧めします。 • [トランジットゲートウェイ GRE アドレス] には、トランジットゲートウェイ CIDR ブロックから割り当てられた IP アドレス (`10.100.254.1` など) を入力します。 • [ピア GRE アドレス] には、トランジット VIF 用にルーター上に作成された VLAN に割り当てられた IP アドレス (`169.254.100.1` など) を入力します。AWS が IP アドレスにアクセスできれば、ピア GRE アドレスには VLAN や Loopback などの任意の	

タスク	説明	必要なスキル
	インターフェイスを使用できます。 • BGP 内部 CIDR ブロック (IPv4) の場合は、BGP 内部 CIDR ブロック IP アドレス (169.254.101.16/29 など) を入力します。 • ピア ASN には、ルーターの ASN (65534 など) を入力します。 2. これらの手順を繰り返して、ルータごとに GRE トンネルを作成します。	

ルーターにルートをアドバタイズする

タスク	説明	必要なスキル
ルートをアドバタイズします。	新しいTransit Gateway Connect アタッチメントを、この VRF 用に作成したルートテーブルに関連付けます。たとえば、プロダクション Transit Gateway Connect アタッチメントを Production-VRF ルートテーブルに関連付けます。 ルーターにアドバタイズされるプレフィックス用の静的ルートを作成します。	ネットワーク管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	1. ネットワークハブアカウントにサインインします。 2. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。 3. ナビゲーションペインで、[Transit Gateways]、[Transit Gateway ルートテーブル] の順に選択します。 4. `Production-VRF` ルートテーブルを選択します。 5. [アクション] メニューで [静的ルートの作成] を選択します。 6. CIDR には、ターゲット VPC の Transit Gateway アタッチメントにアドバタイズされたルートの CIDR ブロック (`10.100.1.0/24` など) を入力します。 7. [アタッチメントの選択] には、関連する Transit Gateway Connect アタッチメントを選択します。 8. [静的ルートの作成] を選択します。	

関連リソース

ドキュメント

- Direct Connect ドキュメント
  - Direct Connect ゲートウェイの操作
  - トランジットゲートウェイの関連付け
  - AWS Direct Connect 仮想インターフェイス
- Transit Gateway ドキュメント
  - トランジットゲートウェイの使用
  - Direct Connect ゲートウェイへのトランジットゲートウェイアタッチメント
  - Transit Gateway Connect アタッチメントと Transit Gateway Connect ピア
  - Transit Gateway の 接続 アタッチメントを作成する

ブログの投稿

- AWS Transit Gateway Connect によるハイブリッドネットワークのセグメント化
- AWS Transit Gateway Connect を使用して VRF を拡張し、IP プレフィックスアドバタイズメントを増やす

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS KMS キーのキーの状態が変更されたときに Amazon SNS 通知を受け取る

作成者:シュバム・ハルソラ (AWS)、アロマル・ラージ・ジャヤラジャン (AWS)、ナヴディープ・パリーク (AWS)

概要

AWS Key Management Service (AWS KMS) キーに関連付けられているデータとメタデータは、キーが削除されると失われます。削除は元に戻せず、失われたデータ (暗号化されたデータを含む) を回復することはできません。AWS KMS キーの「キーステータス」の変更を通知する通知システムを設定することで、データ損失を防ぐことができます。

このパターンは、Amazon EventBridge と Amazon Simple Notification Service (Amazon SNS) を使用して AWS KMS キーのキーの状態が `Disabled` または `PendingDeletion` に変わるたびに自動通知を発行することで、AWS KMS キーのステータス変化を監視する方法を示しています。たとえば、ユーザーが AWS KMS キーを無効化または削除しようとすると、試行されたステータス変更の詳細が記載されたメール通知が届きます。このパターンを使用して、AWS KMS キーの削除をスケジュールすることもできます。

前提条件と制限

前提条件

- AWS Identity and Access Management (IAM) ユーザーがいるアクティブな AWS アカウント
- 「AWS KMS key」

アーキテクチャ

テクノロジースタック

- Amazon EventBridge
- AWS Key Management Service (AWS KMS)
- Amazon Simple Notification Service (Amazon SNS)

ターゲットアーキテクチャ

次の図は、AWS KMS キーの状態の変化を検出するための自動モニタリングおよび通知プロセスを構築するためのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーが AWS KMS キーの削除を無効化またはスケジュールします。
2. EventBridge ルールは、スケジュール `Disabled` または `PendingDeletion` イベントを評価します。
3. EventBridge ルールは Amazon SNS トピックを呼び出します。
4. Amazon SNS はユーザーに E メール通知メッセージを送信します。

> **Note**
>
> 組織のニーズに合わせて E メールメッセージをカスタマイズできます。AWS KMS キーが使用されているエンティティに関する情報を含めることをお勧めします。これにより、ユーザーは AWS KMS キーを削除した場合の影響を理解できます。AWS KMS キーが削除される 1 日または 2 日前に送信されるリマインダーメール通知をスケジュールすることもできます。

自動化とスケール

AWS CloudFormation スタックは、このパターンが機能するために必要なすべてのリソースとサービスをデプロイします。このパターンは、1 つのアカウントに個別に実装することも、「AWS CloudFormation StackSets」を使用して AWS Organizations の複数の独立したアカウントまたは「組織単位」に実装することもできます。

ツール

- 「AWS CloudFormation」を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントと AWS リージョンにわたってライフサイクル全体にわたってリソースを管理できます。このパターンの CloudFormation テンプレートは、必要なすべての AWS リソースを記述し、CloudFormation はそれらのリソースをプロビジョニングして構成してくれる。
- 「Amazon EventBridge」は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、お客様独自のアプ

リケーション、AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを AWS Lambda などのターゲットにルーティングします。EventBridge は、イベント駆動型アーキテクチャを構築するプロセスを簡素化します。

- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

コード

このパターンのコードは、GitHub 内の「Monitor の AWS KMS キー無効化および予定削除」リポジトリで利用できます。

エピック

CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	以下のコマンドを実行して、GitHub「Monitor AWS KMS キー無効化およびスケジュール削除」リポジトリをローカルマシンに複製します。 `git clone https://github.com/aws-samples/aws-kms-deletion-notification`	AWS 管理者、クラウドアーキテクト
テンプレートのパラメータを更新します。	コードエディターで、リポジトリから複製した `Alerting-KMS-Events.yaml` CloudFormation テンプレートを開き、次のパラメーターを更新します。	AWS 管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	• `DestinationEmailAddress` には、SNS 通知の受信に使用する予定のアクティブなメールアドレスを入力します。 • `SNSTopicName` には、SNS トピックの名前を入力します。	
CloudFormation のテンプレートをデプロイします。	1. AWS マネジメントコンソールにサインインし、「CloudFront コンソール」を開きます。 2. ナビゲーションペインで [スタックの作成] を選択し、[新しいリソース (標準) を使用] を選択します。 3. 「リソースの識別」ページで「次へ」を選択します。 4. 「テンプレートの指定」ページの「テンプレートソース」で、「テンプレートファイルのアップロード」を選択します。 5. [ファイルを選択] を選択し、クローンした GitHub リポジトリから `Alerting-KMS-Events.yaml` ファイルを選択して、[次へ] を選択します。 6. スタック名に対して、スタック名を入力します。 7. [送信] を選択します。	AWS 管理者、クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションメールを確認します。	CloudFormation のテンプレートが正常にデプロイされると、Amazon SNS は CloudFormation テンプレートで指定されたメールアドレスにサブスクリプションの確認メッセージを送信します。 通知を受信するには、このメールのサブスクリプションを確認する必要があります。詳細については、Amazon SNS 開発者ガイドの「サブスクリプションの確認」を参照してください。	AWS 管理者、クラウドアーキテクト

サブスクリプション通知をテストする

タスク	説明	必要なスキル
AWS KMS キーを無効にします。	1. AWS マネジメントコンソールにサインインし、「AWS KMS コンソール」を開きます。 2. リージョンを変更するには、現在表示されているリージョン名を選択してから、切り替えたいリージョンを選択します。 3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。	AWS 管理者

タスク	説明	必要なスキル
	4. 有効または無効にする AWS KMS キーのチェックボックスを選択します。 5. AWS KMS キーを無効にするには、[キーアクション]、[無効] の順に選択します。	
サブスクリプションを検証します。	Amazon SNS の通知メールが届いたことを確認します。	AWS 管理者

リソースをクリーンアップする

タスク	説明	必要なスキル
CloudFormation スタックを削除します。	1. AWS マネジメントコンソールにサインインし、「CloudFront コンソール」を開きます。 2. ナビゲーションペインで、[Stacks] を選択します。 3. 以前に作成したスタックを選択し、[削除] を選択します。	AWS 管理者

関連リソース

- 「AWS CloudFormation」(AWS ドキュメント)
- 「AWS CloudFormation コンソールでのスタックの作成」(AWS CloudFormation ドキュメント)
- 「AWS でのイベント駆動型アーキテクチャの構築」(AWS Workshop Studio ドキュメント)
- 「AWS Key Management Service ベストプラクティス」(AWS ホワイトペーパー)
- 「AWS Key Management Service のセキュリティベストプラクティス」(AWS KMS 開発者ガイド)

追加情報

Amazon SQS は、デフォルトで送信中の暗号化を提供します。キュリティのベストプラクティスに合わせて、AWS KMS カスタマーマネージドキーを使用して Amazon SNS のサーバー側の暗号化を有効にすることもできます。

非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約

アダム・スパイサー (AWS) によって作成されました

概要

Amazon Web Services(AWS)は、トランジットゲートウェイアタッチメントとロードバランサーエンドポイント (AWS Network Firewall またはサードパーティアプライアンスをサポートするため) の両方に仮想プライベートクラウド(VPC)の専用サブネットを使用することを推奨するベストプラクティスを公開しています。これらのサブネットは、これらのサービスのエラスティックネットワークインターフェイスを格納するために使用されます。AWS Transit Gateway とゲートウェイロードバランサーの両方を使用する場合、VPC の各アベイラビリティーゾーンに 2 つのサブネットが作成されます。VPC の設計方法により、このような余分なサブネットは /28 マスクより小さくすることはできず、ルーティング可能なワークロードに使用できたはずの貴重なルーティング可能な IP スペースを消費する可能性があります。このパターンは、ルーティング不可能なClassless Inter-Domain Routing (CIDR) 範囲をこれらの専用サブネットに使用して、ルーティング可能な IP スペースを節約する方法を示しています。

前提条件と制限

前提条件

- ルーティング可能な IP スペースのマルチ VPC 戦略
- 使用しているサービス (トランジットゲートウェイアタッチメント、ゲートウェイロードバランサー または Network Firewall エンドポイント) のルーティング不可能なCIDR範囲

アーキテクチャ

ターゲット アーキテクチャ

このパターンには 2 つのリファレンスアーキテクチャが含まれます。1 つのアーキテクチャにはトランジットゲートウェイ (TGW) アタッチメント用のサブネットと ゲートウェイ ロードバランサーエンドポイント (GWLBE) 用のサブネットがあり、もう 1 つのアーキテクチャには TGW アタッチメント専用のサブネットがあります。

アーキテクチャ 1 – アプライアンスへの入力ルーティングを備えた TGW 接続 VPC

次の図は、2 つのアベイラビリティーゾーンにまたがる VPC のリファレンスアーキテクチャを示しています。インレスでは、VPC は イングレスルーティングパターン を使用して、パブリックサブネット宛のトラフィックを Bump-in-the-Wire アプライアンスに転送し、ファイアウォールインスペクションを行います。TGW アタッチメントは、プライベートサブネットから別の VPC への出力をサポートします。

このパターンでは、TGW アタッチメントサブネットと GwLBe サブネットにルーティング不可能な CIDR 範囲を使用します。TGW ルートテーブルでは、このルーティング不能な CIDR は、より具体的なルートのセットを使用してブラックホール (静的) ルートに設定されています。ルートが TGW ルートテーブルに伝達される場合、これらのより具体的なブラックホールルートが適用されます。

この例では、/23 ルーティング可能な CIDR が分割され、ルーティング可能なサブネットに完全に割り当てられます。

アーキテクチャ 2 — TGW 接続 VPC

次の図は、2 つのアベイラビリティーゾーンにまたがる VPC のリファレンスアーキテクチャを示しています。TGW アタッチメントは、プライベートサブネットから別の VPC へのアウトバウンドトラフィック (下り) をサポートします。TGW アタッチメントサブネットにのみルーティング不可能な CIDR 範囲を使用します。TGW ルートテーブルでは、このルーティング不能な CIDR は、より具体的なルートのセットを使用してブラックホール (静的) ルートに設定されています。ルートが TGW ルートテーブルに伝達される場合、これらのより具体的なブラックホールルートが適用されます。

この例では、/23 ルーティング可能な CIDR が分割され、ルーティング可能なサブネットに完全に割り当てられます。

ツール

AWS のサービスと設定されているリソース

- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。このパターンでは、VPC セカンダリ CIDR を使用して、ワークロード CIDR 内のルーティング可能な IP スペースを確保します。

- インターネットゲートウェイのイングレスルーティング (エッジアソシエーション) は、専用の非ルーティングサブネットの ゲートウェイロードバランサーエンドポイントとともに使用できます。
- AWS Transit Gateway は VPC とオンプレミスネットワークを接続する一元的ハブです。このパターンでは、VPC はトランジットゲートウェイに集中的に接続され、Transit Gateway アタッチメントはルーティングできない専用のサブネットに配置されます。
- ゲートウェイロードバランサーを使用すると、ファイアウォール、侵入検知および防止システム、ディープパケットインスペクションシステムなどの仮想アプライアンスをデプロイ、スケール、管理できます。ゲートウェイは、すべてのトラフィックの単一の入口と出口の役割を果たします。このパターンでは、ゲートウェイロードバランサーのエンドポイントは、ルーティングできない専用のサブネットで使用できます。
- AWS Network Firewall は、ステートフルでマネージド型のネットワークファイアウォールならびに侵入検知および防止サービスです。このパターンでは、ゲートウェイロードバランサーのエンドポイントは、ルーティングできない専用のサブネットで使用できます。

コードリポジトリ

このパターンのランブックと AWS CloudFormation テンプレートは、GitHub のルーティング不可能なセカンダリ CIDR パターンリポジトリにあります。サンプルファイルを使用して、環境内に作業ラボをセットアップできます。

ベストプラクティス

AWS Transit Gateway

- 各 Transit Gateway VPC アタッチメントに個別のサブネットを使用します。
- ルーティング不可能なセカンダリ CIDR 範囲から /28 サブネットを Transit Gateway アタッチメントサブネットに割り当てます。
- 各トランジットゲートウェイのルートテーブルに、ルーティングできない CIDR 範囲の静的でより具体的なルートをブラックホールとして追加します。

ゲートウェイロードバランサーとイングレス・ルーティング

- イングレスルーティングを使用して、インターネットからのトラフィックをゲートウェイロードバランサーエンドポイントに転送します。
- 各ゲートウェイロードバランサーエンドポイントに個別のサブネットを使用します。

- ゲートウェイロードバランサーエンドポイントサブネットに、ルーティング不可能なセカンダリ CIDR 範囲から /28 サブネットを割り当てます。

エピック

VPCの作成

タスク	説明	必要なスキル
ルーティング不可能な CIDR 範囲を決定します。	Transit Gateway アタッチメントサブネットと (オプションで) ゲートウェイLoad Balancer または Network Firewall エンドポイントサブネットに使用される、ルーティング不可能なCIDR範囲を決定します。この CIDR 範囲は VPC のセカンダリ CIDR として使用されます。VPC のプライマリ CIDR 範囲またはより大きなネットワークからルーティング可能であってはなりません。	クラウドアーキテクト
VPC のルーティング可能な CIDR 範囲を決定します。	VPC に使用されるルーティング可能な CIDR 範囲のセットを決定します。この CIDR 範囲は VPC のプライマリ CIDR として使用されます。	クラウドアーキテクト
VPCの作成	VPC を作成し、トランジットゲートウェイにアタッチします。各 VPC には、前の 2 つのステップで決定した範囲に基づいて、ルーティング可能なプライマリ CIDR 範囲と	クラウドアーキテクト

タスク	説明	必要なスキル
	ルーティング不可能なセカンダリ CIDR 範囲が必要です。	

Transit ゲートウェイ のブラックホールルートの設定

タスク	説明	必要なスキル
より具体的なルーティング不可能な CIDR をブラックホールとして作成する。	各トランジットゲートウェイのルートテーブルには、ルーティング不可能なCIDR用に作成されたブラックホールルートのセットが必要です。これらは、セカンダリ VPC CIDR からのトラフィックがルーティング不可能なままになり、大規模なネットワークに漏れることがないように設定されています。これらのルートは、VPC のセカンダリ CIDR として設定されているルーティング不可能な CIDR よりも具体的でなければなりません。たとえば、ルーティング不可能なセカンダリ CIDR が 100.64.0.0/26 の場合、トランジットゲートウェイのルートテーブル内のブラックホールルートは 100.64.0.0/27 と 100.64.0.32/27 である必要があります。	クラウドアーキテクト

関連リソース

- ゲートウェイロードバランサーをデプロイするためのベストプラクティス
- ゲートウェイロードバランサーを使用した分散型検査アーキテクチャ
- ネットワーキングイマージョンデー – インターネットから VPC ファイアウォールラボ
- Transit Gateway 設計のベストプラクティス

追加情報

ルーティング不可能なセカンダリ CIDR 範囲は、大量の IP アドレスを必要とする大規模なコンテナデプロイメントを扱う場合にも役立ちます。このパターンをプライベート NAT ゲートウェイで使用すると、ルーティング不可能なサブネットを使用してコンテナデプロイメントをホストできます。詳細については、ブログ記事 プライベート NAT ソリューションでプライベート IP の枯渇を解決する方法 を参照してください。

コードリポジトリ AWS Service Catalog を使用して で Terraform 製品をプロビジョニングする

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Tamilselvan P (AWS)

概要

AWS Service Catalog はHashiCorp Terraform 設定のガバナンスによるセルフサービスプロビジョニングをサポートしています。Terraform を使用する場合は、Service Catalog を単一のツールとして使用して、 内の Terraform 設定を大規模 AWS に整理、管理、配布できます。Service Catalog の主な機能には、標準化および事前承認されたコードとしてのインフラストラクチャ (IaC) テンプレートのカタログ化、アクセスコントロール、最小特権アクセスによるクラウドリソースのプロビジョニング、バージョニング、数千の共有 AWS アカウント、タグ付けなどがあります。エンジニア、データベース管理者、データサイエンティストなどのエンドユーザーは、アクセスできる製品とバージョンのリストを表示し、1 つのアクションでデプロイできます。

このパターンは、Terraform コードを使用して AWS リソースをデプロイするのに役立ちます。GitHub リポジトリの Terraform コードには、Service Catalog からアクセスできます。このアプローチを使用して、製品を既存の Terraform ワークフローと統合します。管理者は、Terraform を使用して Service Catalog ポートフォリオを作成し、 AWS Launch Wizard 製品を追加できます。

このソリューションの利点は次のとおりです。

- Service Catalog のロールバック機能のため、デプロイ中に問題が発生した場合は、製品を以前のバージョンに戻すことができます。
- 製品バージョンの違いを簡単に特定できます。これにより、デプロイ中の問題を解決できます。
- GitHub や GitLab GitLab など、Service Catalog でリポジトリ接続を設定できます。製品の変更は、 リポジトリから直接行うことができます。

の全体的な利点については AWS Service Catalog、「サービスカタログとは」を参照してください。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- ZIP 形式の Terraform 設定ファイルを含む GitHub、BitBucket、またはその他のリポジトリ。

- AWS Serverless Application Model コマンドラインインターフェイス (AWS SAM CLI) がインストールされています。
- AWS Command Line Interface （ AWS CLI) をインストールして設定します。
- Go、インストール済み。
- Python バージョン 3.9、インストール済み。 AWS SAM CLI にはこのバージョンの Python が必要です。
- Service Catalog 製品とポートフォリオにアクセスして管理するための AWS Lambda 関数とアクセス許可を記述して実行するアクセス許可。

アーキテクチャ

この図表は、次のワークフローを示しています：

1. Terraform 設定の準備ができると、開発者はすべての Terraform コードを含む .zip ファイルを作成します。開発者は、.zip ファイルを Service Catalog に接続されているコードリポジトリにアップロードします。
2. 管理者は、Terraform 製品を Service Catalog のポートフォリオに関連付けます。管理者は、エンドユーザーが製品をプロビジョニングできるようにする起動制約も作成します。
3. Service Catalog では、エンドユーザーは Terraform 設定を使用して AWS リソースを起動します。デプロイする製品バージョンを選択できます。

ツール

AWS のサービス

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Service Catalog は、承認された IT サービスのカタログを一元管理するために役立ちます AWS。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。

その他のサービス

- Go は、Google がサポートするオープンソースのプログラミング言語です。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

Service Catalog を介してデプロイできるサンプル Terraform 設定が必要な場合は、GitHub Amazon Macie Organization Setup Using Terraform リポジトリで設定を使用できます。このリポジトリでコードサンプルを使用する必要はありません。

ベストプラクティス

- Terraform 設定ファイル (`terraform.tfvars`) で変数の値を指定する代わりに、Service Catalog を使用して製品を起動するときに変数値を設定します。
- 特定のユーザーまたは管理者にのみポートフォリオへのアクセスを許可します。
- 最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

ローカルワークステーションをセットアップする

タスク	説明	必要なスキル
(オプション) Docker をインストールします。	開発環境で AWS Lambda 関数を実行する場合は、Docker をインストールします。手順については、Docker ドキュメントの「Docker Engine のインストール」を参照してください。	DevOps エンジニア
Terraform AWS Service Catalog 用の エンジンをインストールします。	1. 次のコマンドを入力して、AWS Service Catalog Engine for Terraform リポジトリのクローンを作成します。	DevOps エンジニア、AWS 管理者

タスク	説明	必要なスキル

	`git clone https://github.com/aws-samples/service-catalog-engine-for-terraform-os.git`
	2. クローンされたリポジトリのルートディレクトリに移動します。
	3. 次のコマンドを入力します。これにより、 エンジンがインストールされます。
	`run ./bin/bash/deploy-tre.sh -r`
	デフォルトプロファイルの AWS リージョン セットは、自動インストール中に使用されることはありません。代わりに、このコマンドを実行するときに リージョンを指定します。	

GitHub リポジトリを接続する

タスク	説明	必要なスキル
GitHub リポジトリへの接続を作成します。	1. にサインインし AWS Management Console、デベロッパーツールコンソールを開きます。開発者ツールコンソールにアクセスするには、 AWS CodePipel	AWS 管理者

タスク	説明	必要なスキル
	ine や などのサービスを選択します AWS CodeDeploy。 2. 左側のナビゲーションペインで、設定を選択し、接続を選択します。 3. [接続を作成] を選択します。 4. Terraform ソースコードを維持するリポジトリを選択します。例えば、Bitbucket、GitHub、または GitHub Enterprise Server を選択できます。 5. 接続の名前を入力し、Connect を選択します。 6. プロンプトが表示されたら、リポジトリを認証します。 認証が完了すると、接続が作成され、ステータスがアクティブに変わります。	

Service Catalog で Terraform 製品を作成する

タスク	説明	必要なスキル
Service Catalog 製品を作成します。	1. AWS Service Catalog コンソールを開きます。 2. 「管理」セクションに移動し、[製品リスト] を選択します。	AWS 管理者

タスク	説明	必要なスキル
	3. [製品の作成] を選択します。 4. 「製品の詳細」セクションの「製品の作成」ページで、外部製品タイプを選択します。 Service Catalog は、この製品タイプを使用して Terraform Community Edition 製品をサポートします。 5. Service Catalog 製品の名前と所有者を入力します。 6. CodeStar プロバイダーを使用してコードリポジトリを指定するを選択します。 7. リポジトリに次の情報を入力します。 • を使用してプロバイダーに接続する AWS CodeConnections – 前に作成した接続を選択します。 • リポジトリ – リポジトリを選択します。 • ブランチ – ブランチを選択します。 • テンプレートファイルパス – コードテンプレートファイルが保存されているパスを選択します。ファイル名は で終わる必要がありますtar.gz。	

タスク	説明	必要なスキル
	8. バージョン名と説明に、製品バージョンに関する情報を入力します。 9. [製品の作成] を選択します。	
ポートフォリオを作成します。	1. AWS Service Catalog コンソールを開きます。 2. 管理セクションに移動し、ポートフォリオを選択します。 3. [ポートフォリオの作成] を選択します。 4. 次の値を入力します。 • ポートフォリオ名 - `Sample terraform` • ポートフォリオの説明 — `Sample portfolio for Terraform configurations` • 所有者 – E メールアドレスなどの連絡先情報 5. [Create] (作成) を選択します。	AWS 管理者

タスク	説明	必要なスキル
Terraform 製品をポートフォリオに追加します。	1. AWS Service Catalog コンソールを開きます。 2. 「管理」セクションに移動し、[製品リスト] を選択します。 3. 前に作成した Terraform 製品を選択します。 4. アクションを選択し、ポートフォリオに製品を追加するを選択します。 5. `Sample terraform` ポートフォリオを選択します。 6. [製品をポートフォリオに追加] を選択します。	AWS 管理者

タスク	説明	必要なスキル
アクセスポリシーを作成します。	1. AWS Identity and Access Management (IAM) コンソールを開きます。 2. ナビゲーションペインで、ポリシー を選択してください。 3. コンテンツペインで、[ポリシーの作成] を選択します。 4. [JSON] オプションを選択します。 5. このパターンの追加情報セクションのアクセスポリシーにサンプル JSON ポリシーを入力します。 6. [Next (次へ)] を選択します。 7. 確認と作成 ページで、ポリシー名ボックスに「」と入力しますTerraformResourceCreationAndArtifactAccessPolicy 。 8. [Create policy] を選択します。	AWS 管理者

タスク	説明	必要なスキル
カスタム信頼ポリシーを作成します。	1. IAM コンソールを開きます。 2. ナビゲーションペインで [Roles (ロール)] を選択します。 3. [Create role] を選択します。 4. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。 5. JSON ポリシーエディタで、このパターンの追加情報セクションの信頼ポリシーにサンプル JSON ポリシーを入力します。 6. [Next (次へ)] を選択します。 7. アクセス許可ポリシーで、`Terraform ResourceCreationAndArtifactAccessPolicy` 以前に作成した を選択します。 8. [Next (次へ)] を選択します。 9. 「ロールの詳細」の「ロール名」ボックスに「」と入力します`SCLaunch-product` 。	AWS 管理者

タスク	説明	必要なスキル
	Important ロール名は で始まる必要がありますSCLaunch。 10[ロールの作成] を選択します。	

タスク	説明	必要なスキル
Service Catalog 製品に起動制約を追加します。	1. 管理者権限を持つユーザー AWS Management Console として にサインインします。 2. AWS Service Catalog コンソールを開きます。 3. ナビゲーションペインで、ポートフォリオを選択します。 4. 前に作成したポートフォリオを選択します。 5. [ポートフォリオの詳細] ページで、[制約] タブを選択し、[制約の作成] を選択します。 6. ForProduct で、前に作成した Terraform 製品を選択します。 7. 起動制約の「メソッド」で「ロール名の入力」を選択します。 8. ロール名ボックスに、 と入力しますSCLaunch-product 。 9. [Create] (作成) を選択します。	AWS 管理者

タスク	説明	必要なスキル
製品へのアクセスを許可します。	1. AWS Service Catalog コンソールを開きます。 2. ナビゲーションペインで、ポートフォリオを選択します。 3. 前に作成したポートフォリオを選択します。 4. Accesstab を選択し、Grant access を選択します。 5. ロールタブを選択し、この製品をデプロイするためのアクセス権を持つロールを選択します。 6. [Grant access (アクセス権の付与)] を選択します。	AWS 管理者

タスク	説明	必要なスキル
製品を起動します。	1. Service Catalog 製品をデプロイするアクセス許可を持つユーザー AWS Management Console として にサインインします。 2. AWS Service Catalog コンソールを開きます。 3. ナビゲーションペインで、[Products] (製品) を選択します。 4. 前に作成した製品を選択し、Launch 製品を選択します。 5. 製品名を入力し、必要なパラメータを定義します。 6. [製品の起動] を選択します。	DevOps エンジニア

デプロイメントを確認する

タスク	説明	必要なスキル
デプロイを検証します。	Service Catalog プロビジョニングワークフローには 2 つの AWS Step Functions ステートマシンがあります。 • `ManageProvisionedProductStateMachine` – Service Catalog は、新しい Terraform 製品をプロビジョニングするとき、および既存の Terraform プロ	DevOps エンジニア

タスク	説明	必要なスキル
	ビジョニング済み製品を更新するときに、このステートマシンを呼び出します。 • `TerminateProvisionedProductStateMachine` – Service Catalog は、既存の Terraform プロビジョニング済み製品を終了するときに、このステートマシンを呼び出します。 `ManageProvisionedProductStateMachine` ステートマシンのログをチェックして、製品がプロビジョニングされたことを確認します。 1. にサインインし AWS Management Console、AWS Step Functions コンソールを開きます。 2. 左側のナビゲーションペインで、State マシンを選択します。 3. 選択 `ManageProvisionedProductStateMachine` 。 4. 実行リストで、プロビジョニングされた製品 ID を入力して実行を見つけます。	

タスク	説明	必要なスキル
	Note 状態ファイルのバックエンドバケット名は で始まりますsc-terraform-engine-state-。 5. 必要なリソースがすべてアカウントで作成されていることを確認します。	

インフラストラクチャをクリーンアップする

タスク	説明	必要なスキル
プロビジョニング済み製品を削除します。	1. Service Catalog 製品をデプロイするアクセス許可を持つユーザー AWS Management Console として にサインインします。 2. AWS Service Catalog コンソールを開きます。 3. 左側のナビゲーションで、プロビジョニング済み製品を選択します。 4. 作成した製品を選択します。 5. Actions リストで、Terminate を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	6. 確認テキストボックスに「」と入力しterminate、「プロビジョニング済み製品の終了」を選択します。 7. これらのステップを繰り返して、プロビジョニングされたすべての製品を削除します。	

タスク	説明	必要なスキル
Terraform AWS Service Catalog 用の エンジンを削除します。	1. 管理者権限を持つユーザー AWS Management Console として にサインインします。 2. Amazon Simple Storage Service (Amazon S3) コンソールを開きます。 3. ナビゲーションペインで、バケットを選択します。 4. `sc-terraform-engine-logging-XXXX` バケットを選択します。 5. 空 を選択します。 6. 次のバケットに対してステップ 4～5 を繰り返します。 • `sc-terraform-engine-state-XXXX` • `terraform-engine-bootstrap-XXXX` 7. AWS CloudFormation コンソールを開き、正しい にいることを確認します AWS リージョン。 8. 左側のナビゲーションで、スタックを選択します。 9. を選択し`SAM-TRE`、Delete を選択します。スタックが削除されるまで待ちます。	AWS 管理者

タスク	説明	必要なスキル
	10.を選択しBootstrap-TRE、削除を選択します。スタックが削除されるまで待ちます。	

関連リソース

AWS ドキュメント

- Terraform 製品の開始方法

Terraformのドキュメント

- Terraform のインストール
- Terraform バックエンド設定
- Terraform AWS プロバイダーのドキュメント

追加情報

アクセスポリシー

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/servicecatalog:provisioning": "true"
                }
            }
        },
        {
            "Action": [
```

```
                "s3:CreateBucket*",
                "s3:DeleteBucket*",
                "s3:Get*",
                "s3:List*",
                "s3:PutBucketTagging"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "resource-groups:CreateGroup",
                "resource-groups:ListGroupResources",
                "resource-groups:DeleteGroup",
                "resource-groups:Tag"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "tag:GetResources",
                "tag:GetTagKeys",
                "tag:GetTagValues",
                "tag:TagResources",
                "tag:UntagResources"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

信頼ポリシー

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GivePermissionsToServiceCatalog",
            "Effect": "Allow",
            "Principal": {
                "Service": "servicecatalog.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account_id:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
TerraformExecutionRole*",
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
ServiceCatalogExternalParameterParserRole*",
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
ServiceCatalogTerraformOSParameterParserRole*"
                    ]
                }
            }
        }
    ]
}
```

Amazon SES を使用して、単一メールアドレスで複数の AWS アカウントを登録する

作成者: Joe Wozniak (AWS) および Shubhangi Vishwakarma (AWS)

概要

このパターンでは、 に関連付けられている E メールアドレスから実際の E メールアドレスを切り離す方法について説明します AWS アカウント。アカウントの作成時に一意の E メールアドレスを指定 AWS アカウント する必要があります。一部の組織では、 が管理するチームが、メッセージングチームで多くの一意の E メールアドレスを管理する負担を引き受け AWS アカウント る必要があります。これは、多くの を管理する大規模な組織では難しい場合があります AWS アカウント。さらに、E メールシステムが、E メールフィルタリング: サブアドレス拡張 (RFC 5233) で定義されているとおりに、プラスのアドレス指定またはサブアドレス指定を許可しない場合、プラス記号 (+) と識別子を E メールアドレスのローカル部分の末尾に追加します。このadmin+123456789123@example.comパターンは、この制限を克服するのに役立ちます。

このパターンは、 AWS アカウント 所有者が 1 つの E メールアドレスを複数の E メールアドレスに関連付けることを可能にする、一意の E メールアドレス販売ソリューションを提供します AWS アカウント。 AWS アカウント 所有者の実際の E メールアドレスは、テーブル内で生成された E メールアドレスに関連付けられます。このソリューションは、一意のメールアカウントのすべての受信メールを処理し、各アカウントの所有者を検索して、受信したメッセージを所有者に転送します。

前提条件と制限

前提条件

- への管理アクセス AWS アカウント。
- 開発環境へのアクセス権。
- （ オプション) AWS Cloud Development Kit (AWS CDK) ワークフローと Python プログラミング言語に精通していれば、問題のトラブルシューティングや変更に役立ちます。

機能制限

- 全販売メールアドレスは 64 文字長です。詳細については、 AWS Organizations API リファレンスのCreateAccount」を参照してください。

製品バージョン

- Node.js バージョン 12.7.0 以降
- Python 3.9 以降
- Python パッケージ pip と virtualenv
- AWS CDK バージョン 2.23.0 以降
- Docker 20.10.x 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation スタック
- AWS Lambda 関数
- Amazon Simple Email Service (Amazon SES) ルールとルールセット
- AWS Identity and Access Management (IAM) ロールとポリシー
- Amazon Simple Storage Service (Amazon S3) バケットとバケットポリシー
- AWS Key Management Service （ AWS KMS) キーとキーポリシー
- Amazon Simple Notification Service (Amazon SNS) のトピックとトピックポリシー
- Amazon DynamoDB テーブル

ターゲットアーキテクチャ

この図は、以下の 2 つのフローを示しています。

- メールアドレス販売フロー: この図では、メールアドレス販売フロー (下のセクション) は通常、アカウント販売ソリューションまたは外部自動化で開始、または手動で呼び出されます。リクエストでは、必要なメタデータを含むペイロードで Lambda 関数が呼び出されます。この関数はこの情報を使用して一意のアカウント名とメールアドレスを生成し、DynamoDB データベースに保存して、呼び出し元に値を返します。これらの値を使用して、新しい AWS アカウント （ 通常は を使用) を作成できます AWS Organizations。
- メール転送フロー: このフローは、前の図の上部セクションに示されています。E メールアドレス販売フローから生成されたアカウント E メールを使用して が作成されると、 AWS アカウントはその E メールアドレスにアカウント登録の確認や定期的な通知など、 AWS さまざまな E メールを送信します。このパターンのステップに従って、ドメイン全体の E メールを受信するように

Amazon SES AWS アカウント で を設定します。このソリューションでは、Lambda がすべての受信メールを処理し、T0 アドレスが DynamoDB テーブルにあるかどうかを確認し、代わりにアカウントオーナーのメールアドレスにメッセージを転送できるようにする転送ルールを設定します。このプロセスを使用すると、アカウントオーナーは複数のアカウントを単一メールアドレスに関連付けできます。

自動化とスケール

このパターンでは AWS CDK 、 を使用してデプロイを完全に自動化します。このソリューションでは、ニーズに合わせて自動的にスケールする (またはスケールするように設定できる) AWS マネージドサービスを使用します。Lambda 関数には、スケーリングのニーズを満たすために追加の設定が必要な場合があります。詳細については、Lambda ドキュメントの「Lambda 関数のスケーリングについて」を参照してください。

ツール

AWS サービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドを通じて AWS のサービスを操作するのに役立つオープンソースツールです。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Email Service (Amazon SES) - ユーザー自身のメールアドレスとドメインを使用してメールを送受信する上で役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータの保存、保護、取得に役立つクラウドベースのオブジェクトストレージサービスです。

デプロイに必要なツール

- AWS CLI および IAM アクセスを持つ開発環境 AWS アカウント。詳細については、関連リソースセクションのリンクを参照してください。
- 開発システムに以下をインストールします。
  - Git コマンドラインツールは、Git ダウンロードウェブサイトから入手できます。
  - AWS CLI のアクセス認証情報を設定する AWS CDK。詳細については、AWS CLI のドキュメントを参照してください。
  - Python バージョン 3.9 以降。Python ダウンロードウェブサイトから入手できます。
  - Python パッケージ pip と virtualenv。インストール手順については、pip ドキュメントと virtualenv ドキュメントを参照してください。
  - Node.js バージョン 12.7.0 以降。インストール手順については、Node.js ドキュメントを参照してください。
  - AWS CDK バージョン 2.23.0 以降。インストール手順については、 AWS CDK ドキュメントを参照してください。
  - Docker バージョン 20.10.x 以降。インストール手順については、Docker ドキュメントを参照してください。

コード

このパターンのコードは、GitHub AWS アカウント Factory の E メールリポジトリで入手できます。

エピック

ターゲットデプロイ環境を割り当てる

タスク	説明	必要なスキル
を特定または作成します AWS アカウント。	E メールソリューションをデプロイするために、完全な管理アクセス権 AWS アカウン	AWS 管理者、クラウド管理者

タスク	説明	必要なスキル
	ト がある既存または新規の を特定します。	
デプロイ環境を設定します。	次の手順に従って、使い易いデプロイ環境を構成し、依存関係を設定します。 1. 「 ツール」セクションに記載されているツールを使用して、開発環境をセットアップします。 2. コマンドを使用して 、GitHub AWS アカウントファクトリ E メールリポジトリコードベースを開発環境にクローンします。 `git clone https://github.com/aws-samples/aws-account-factory-email` 3. `requirements.txt` ファイル (リポジトリのルート) で、 で始まる行を、環境で実行 AWS CDK されている のバージョンと一致する`aws-cdk-lib==` ように更新します。バージョンを特定するには、`cdk --version` コマンドを使用します。	AWS DevOps、アプリ開発者

検証済みドメインをセットアップする

タスク	説明	必要なスキル
ドメインを特定して割り当てます。	メール転送機能には専用ドメインが必要です。Amazon SES で検証できるドメインまたはサブドメインを特定して割り当てます。このドメインは、E メール転送ソリューション AWS アカウント がデプロイされている 内で受信 E メールを受信できる必要があります。 ドメイン要件: • ドメインは標準ドメインまたはサブドメインである必要があります。 • ドメインは組織外からの E メールの受信に使用されるため、外部 DNS で解決可能である必要があります。	クラウド管理者、ネットワーク管理者、DNS 管理者
ドメインを検証します。	特定したドメインが受信メールの受け入れに使用できることを確認します。 Amazon SES ドキュメントの Amazon SES E メールを受信するドメインの検証の手順を実行します。これには、ドメインの DNS レコードを担当する個人またはチームとの調整が必要です。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
MX レコードをセットアップします。	AWS アカウント とリージョンの Amazon SES エンドポイントを指す MX レコードを使用してドメインを設定します。詳細については、Amazon SES ドキュメントの Amazon SES E メール受信用 MX レコードの公開を参照してください。	クラウド管理者、ネットワーク管理者、DNS 管理者

メールの販売と転送ソリューションをデプロイする

タスク	説明	必要なスキル
のデフォルト値を変更しますcdk.json。	デプロイ後にソリューションが正しく動作するように、cdk.json ファイル (リポジトリのルート内) のデフォルト値の一部を編集します。 1. 前に確認したドメイン名と一致するように SES_DOMAIN_NAME 値を変更します。 2. SES_DOMAIN_NAME の同じドメインを含むように ADDRESS_FROM 値を変更します。アドレスのローカル部分はクラウドチームが決定する必要があります。このアドレスは、ソリューションから転送されるすべてのメールの FROM アドレスとなります。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	3. 一致しない受信メッセージの転送先となるメールアドレスと一致するように ADDRESS_ADMIN 値を変更します。この値は、有効かつ動作中のメールアドレスである必要があります。	

タスク	説明	必要なスキル

メールの販売と転送ソリューションをデプロイします。	1. Python 仮想環境を作成する
	`python -m venv .venv`
	2. Python 仮想環境をアクティブ化する
	`source .venv/bin/activate`
	または、Windows プラットフォームで、以下を使用します。
	`% .venv\Scripts\activate.bat`
	3. Python のすべての要件をエラーなしでインストールする
	`pip install -r requirements.txt`
	4. CloudFormation テンプレートを合成する
	`cdk synth`
	エラーがなく、完全な CloudFormation テンプレートに期待どおりの出力が含まれていることを確認します。
	5. (オプション) AWS CDK コードを現在の AWS アカ
		アプリ開発者、AWS DevOps

タスク	説明	必要なスキル

	ウント またはリージョンに初めてデプロイする場合は、環境をブートストラップします。詳細については、 AWS CDK ドキュメントのAWS CDK「ブートストラップ」を参照してください。
	`cdk bootstrap aws://AWS-ACCOUNT-NUMBER/REGION`
	AWS-ACCOUNT-NUMBER および REGION を実際の値に置き換えます。
	6. ソリューションをデプロイする
	`cdk bootstrap`
	`cdk deploy`
	コマンドはエラーなしで実行する必要があります。	

タスク	説明	必要なスキル
ソリューションがデプロイされていることを確認します。	テストを開始する前に、ソリューションが正常にデプロイされたことを確認します。 1. AWS CloudFormation コンソールを開き、 という名前を含む CloudFormation スタックを探します `AwsMailFwdStack` 。 2. この `AwsMailFwdStack` スタックに以下のリソースがあることを確認します。 • Lambda 関数 • Amazon SES のルールとルールセット • IAM ロールとポリシー • Amazon S3 バケットとバケットポリシー • AWS KMS キーとキーポリシー • Amazon SNS トピックとトピックポリシー • DynamoDB テーブル	アプリ開発者、AWS DevOps

メールの販売と転送が想定どおりに動作することを確認する

タスク	説明	必要なスキル
API が動作していることを確認します。	このステップでは、ソリューションの API にテストデータを送信し、ソリューションが期待どおりの出力を生成し、バックエンド操作が期待どお	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	りに実行されていることを確認します。 テスト入力を使用して、販売メールの Lambda 関数を手動で実行します。(例については、sample_vend_request.json ファイルを参照してください。) `OwnerAddress` の場合は、有効なメールアドレスを使用します。API は、期待どおりの値を含むアカウント名とアカウントメールを返す必要があります。	

タスク	説明	必要なスキル
メールが転送中であることを確認します。	このステップでは、システム経由でテストメールを送信し、メールが想定される受信者に転送されていることを確認します。 1. 最後の手順でアカウントメールを入手します。 2. テスト件名と本文を記載したメールをこのアドレスに送信します。 3. アカウントオーナーのメールアドレスにメールを受信したことを確認します。 4. 受信したメールの FROM アドレスが、cdk.json の ADDRESS_FROM 設定と一致していることを確認します。 5. 受信したメールの件名と本文が元の送信メッセージと同じであることを確認します。	アプリ開発者、AWS DevOps

トラブルシューティング

問題	ソリューション
システムが期待どおりにメールを転送しません。	設定が正しいことを確認する 1. ドメインの Amazon SES 検証プロセスが完了しているはずです。

問題	ソリューション
	2. ドメインは、 AWS アカウント および リージョンの Amazon SES エンドポイントを指す MX レコードで適切に設定する必要があります。詳細については、Amazon SES ドキュメントの Amazon SES E メール受信用 MX レコードの公開を参照してください。 ドメインの設定を検証したら、次の手順を実行します。 1. ソリューションをデプロイしたアカウントとリージョンの Amazon CloudWatch コンソールを開き、ナビゲーションペインで CloudWatch ロググループに移動します。 2. `SesMailForwardLogGroup` のロググループのリストを検索します。 3. このグループのログを調べて、メールの販売と転送処理中にエラーが発生していないかを確認します。

問題	ソリューション
AWS CDK スタックをデプロイしようとすると、次のようなエラーが表示されます。 「テンプレートのフォーマットエラー: 認識されないリソースタイプ」	ほとんどの場合、このエラーメッセージは、ターゲットにしているリージョンに利用可能なすべての AWS サービスがないことを意味します。Amazon EC2 インスタンスを使用してソリューションをデプロイしている場合は、インスタンスが実行されているリージョンとは異なるリージョンをターゲットにしている可能性があります。 **Note** デフォルトでは、 は で設定したリージョンとアカウントに AWS CDK デプロイされます AWS CLI。 考えられる解決策 1. このソリューションに必要なすべてのサービス (このパターンの前半の「ターゲットテクノロジースタック」セクションを参照) が、リージョン別の AWS サービスを確認して、ターゲット AWS リージョン とする にあるかどうかを調査します。 2. EC2 インスタンスを使用していて、インスタンスが実行されているリージョンとは異なるリージョンをターゲットにしている場合は、ソリューションをデプロイ AWS CLI する前に、必ず `AWS_DEFAULT_REGION` 環境変数を設定するか、 でリージョンを設定します。詳細については、 AWS CLI ドキュメントの「 の環境変数の設定 AWS CLI」を参照してください。または、AWS CDK 環境のドキュメントの手順に従って、リポジトリのルートにある `app.py` ファイルを変更

問題	ソリューション
	して、ハードコードされたアカウント ID とリージョンを含めることもできます。
ソリューションをデプロイすると、次のエラーメッセージが表示されます。 「デプロイに失敗しました: エラー: AWSMailFWDStack: SSM パラメーター/cdk-bootstrap/hnb659fds/version が見つかりません。環境はブートストラップされていますか? 「cdk bootstrap」を実行してください。	ターゲットとする AWS アカウント およびリージョンに AWS CDK リソースをデプロイしたことがない場合は、エラーが示すように、まず `cdk bootstrap` コマンドを実行する必要があります。ブートストラップコマンドを実行した後もこのエラーが続く場合は、開発環境が実行されているリージョンとは異なるリージョンにソリューションをデプロイしようとしている可能性があります。 この問題を解決するには、ソリューションをデプロイ AWS CLI する前に、 `AWS_DEFAULT_REGION` 環境変数を設定するか、 でリージョンを設定します。または、AWS CDK 環境のドキュメントの手順に従って、リポジトリのルートにある `app.py` ファイルを変更して、ハードコードされたアカウント ID とリージョンを含めることもできます。

関連リソース

- のインストールについては AWS CLI、「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。
- IAM アクセス認証情報 AWS CLI を使用して を設定する方法については、「 の設定 AWS CLI」を参照してください。
- のヘルプについては AWS CDK、「 の開始方法 AWS CDK」を参照してください。

追加情報

コスト

このソリューションをデプロイすると、次のサービスの使用に関連するコストが AWS アカウント所有者に発生する可能性があります。 これらのサービスの請求方法を理解して、潜在的な費用を認識しておくことが重要です。価格設定情報については、次のページを参照してください。

- Amazon SES の価格設定
- Amazon S3 の価格設定
- AWS KMS 料金
- AWS Lambda 料金
- Amazon DynamoDB の価格設定

単一アカウントの AWS 環境でハイブリッドネットワークの DNS 解決を設定

作成者:Abdullahi Olaoye (AWS)

概要

このパターンでは、管理オーバーヘッドなしで、オンプレミスリソース、AWS リソース、インターネットDNSクエリのエンドツーエンドDNS解決を可能にする、完全なハイブリッドドメインネームシステム (DNS) アーキテクチャを設定する方法を示しています。このパターンは、ドメイン名に基づいて AWS から送信される DNS クエリの送信先を決定する、 Amazon Route 53 Resolver 転送ルールを設定する方法を示しています。オンプレミスリソースの DNS クエリは、オンプレミスの DNS レゾルバ に転送されます。AWS リソースの DNS クエリとインターネット DNS クエリは Route 53 Resolver によって解決されます。

このパターンでは、AWS シングルアカウント環境のハイブリッド DNS 解決を対象としています。AWS マルチアカウント環境でのアウトバウンド DNS クエリの設定については、「マルチアカウント AWS 環境でのハイブリッドネットワークの DNS 解決の設定」パターンを参照してください。

前提条件と制限

前提条件

- AWS アカウント
- AWS アカウントに 仮想プライベートクラウド (VPC) を作成
- AWS 仮想プライベートネットワーク (AWS VPN) または AWS Direct Connect を使用した、オンプレミスの環境とお客様の VPC 間のネットワーク接続
- オンプレミス DNS レゾルバ の IP アドレス (VPC からアクセス可能)
- オンプレミスレゾルバ に転送するドメイン/サブドメイン名 (たとえば、 onprem.mydc.com)
- AWS プライベートホストゾーンのドメイン/サブドメイン名 (たとえば、myvpc.cloud.com)

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Route 53 プライベートホストゾーン
- Amazon Route 53 Resolver

- Amazon VPC
- AWS VPN または Direct Connect

ターゲットアーキテクチャ

ツール

- Amazon Route 53 Resolver は、ハイブリッドクラウド全体でシームレスな DNS クエリ解決を可能にすることで、企業のお客様がハイブリッドクラウドをより簡単に利用できるようにします。DNS エンドポイントと条件付き転送ルールを作成して、オンプレミスデータセンターと VPC の間の DNS 名前空間を解決できます。
- Amazon Route 53プライベートホストゾーンは、Amazon VPC サービスで作成する 1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナです。

エピック

プライベートホストゾーンを設定

タスク	説明	必要なスキル
myvpc.cloud.com などの AWS リザーブドドメイン名に、Route 53 プライベートホストゾーンを作成します。	このゾーンには、オンプレミス環境から解決する必要がある AWS リソースの DNS レコードが格納されます。手順については、Route 53 ドキュメントの「プライベートホストゾーンの作成」を参照してください。	ネットワーク管理者、システム管理者
プライベートホストゾーンを VPC に関連付けます。	VPC のリソースが、このプライベートホストゾーンの DNS レコードを解決できるようにするには、VPC をホストゾーンに関連付ける必要がありま	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	す。手順については、Route 53 ドキュメントの「プライベートホストゾーンの作成」を参照してください。	

Route 53 Resolver エンドポイントの設定

タスク	説明	必要なスキル
インバウンドエンドポイントを作成します。	Route 53 Resolverはインバウンド エンドポイントを使用して、DNS クエリをオンプレミスネットワークから Route 53 Resolver に転送します。手順については、Route 53 ドキュメントの「VPC へのインバウンド DNS クエリの転送」を参照してください。受信エンドポイント IP アドレスをメモしておきます。	ネットワーク管理者、システム管理者
アウトバウンドエンドポイントを作成します。	Route 53 Resolver は、アウトバウンドエンドポイントを使用して DNS クエリをオンプレミスの DNS レゾルバ に送信します。手順については、Route 53 ドキュメントの「アウトバウンド DNS クエリのネットワークへの転送」を参照してください。出力エンドポイント ID を書きとめておきます。	ネットワーク管理者、システム管理者

転送ルールを設定して VPC に関連付け

タスク	説明	必要なスキル
オンプレミスドメインのルールを作成します。	このルールは、オンプレミスドメイン (onprem.mydc.com など) の DNS クエリを、すべてオンプレミスの DNS レゾルバ に転送するよう Route 53 Resolver に指示します。このルールを作成するには、オンプレミスの DNS レゾルバの IP アドレスと Route 53 Resolverのアウトバウンドエンドポイント ID が必要です。手順については、Route 53 ドキュメントの「転送ルールの管理」 を参照してください。	ネットワーク管理者、システム管理者
転送ルールを VPC に関連付けます。	転送ルールを有効にする VPC に関連付ける必要があります。その後、Route 53 Resolver はドメインを解決する際にルールを考慮します。手順については、Route 53 ドキュメントの「転送ルールの管理」 を参照してください。	ネットワーク管理者、システム管理者

オンプレミスの DNS レゾルバ の設定

タスク	説明	必要なスキル
オンプレミスの DNS レゾルバ で条件付き転送を設定します。	DNS クエリをオンプレミス環境から Route 53 プライベートホストゾーンに送信するには、オンプレミスの DNS	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	レゾルバ で条件付き転送を設定する必要があります。これにより、AWS ドメイン (myvpc.cloud.com など) のすべての DNS クエリを Route 53 Resolverのインバウンドエンドポイント IP アドレスに転送するよう DNS レゾルバに指示します。	

エンドツーエンドDNS解決をテスト

タスク	説明	必要なスキル
AWS からオンプレミス環境への DNS解決をテストします。	VPC のサーバーから、オンプレミスドメイン (server1.onprem.mydc.com など) の DNS クエリを実行します。	ネットワーク管理者、システム管理者
オンプレミス環境から AWS への DNS 解決案をテストします。	オンプレミスサーバーから、AWS ドメイン (server1.myvpc.cloud.com など) の DNS 解決を実行します。	ネットワーク管理者、システム管理者

関連リソース

- 「Amazon Route 53 および AWS Transit Gateway を使用したハイブリッドクラウドの一元化 DNS 管理」 (AWS ネットワークおよびコンテンツ配信ブログ)
- 「Route 53 Resolverでマルチアカウント環境の DNS 管理を簡素化」 (AWS セキュリティブログ)
- 「プライベートホストゾーンの操作」 (Route 53 ドキュメント)
- 「Route 53 Resolverの使用を開始」 (Route 53 ドキュメント)

AWS CloudFormation を使用して Amazon EC2 に UiPath RPA ボットを自動的にセットアップする

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Tamilselvan P (AWS)

概要

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにロボティックプロセスオートメーション (RPA) ボットをデプロイする方法を説明しています。ここでは、EC2 Image Builder パイプラインを使用して、カスタム Amazon マシンイメージ (AMI) を作成します。AMI は、EC2 インスタンスをデプロイするためのオペレーティングシステム (OS) とプリインストールされたソフトウェアを含む事前構成された仮想マシン (VM)イメージです。このパターンでは、AWS CloudFormation テンプレートを使用して UiPath Studio コミュニティエディションをカスタム AMI にインストールします。UiPath は、ロボットをセットアップしてタスクを自動化するのに役立つ RPA ツールです。

このソリューションの一部として、EC2 Windows インスタンスはベース AMI を使用して起動され、UiPath Studio アプリケーションがインスタンスにインストールされます。このパターンは、Microsoft システム準備 (Sysprep) ツールを使用し、のカスタマイズされた Windows インストールを重複します。その後、ホスト情報を削除し、インスタンスから最終的な AMI を作成します。これにより、独自の命名規則とモニタリング設定で最終的な AMI を使用し、インスタンスをオンデマンドで起動できます。

> **Note**
>
> このパターンでは、RPA ボットの使用に関する情報は提供されません。詳細については、UiPath ドキュメントを参照してください。このパターンを使用して、独自の要件に基づいてインストール手順をカスタマイズすることで、他の RPA ボットアプリケーションを設定することもできます。

このパターンには次のような自動化と利点があります。

- アプリケーションのデプロイと共有: アプリケーションのデプロイ用に Amazon EC2 AMI を構築し、EC2 Image Builder パイプラインを通じて複数のアカウントで共有できます。このパイプラインでは、AWS CloudFormation テンプレートを Infrastructure as Code (IaC) スクリプトとして使用します。

- Amazon EC2 のプロビジョニングとスケーリング: CloudFormation IaC テンプレートは、カスタムコンピュータ名シーケンスと Active Directory 参加の自動化を提供します。
- オブザーバビリティとモニタリング: このパターンは Amazon EC2 メトリクス (CPU やディスク使用量など) のモニタリングに役立つ Amazon CloudWatch ダッシュボードを設定します。
- RPA がビジネスにもたらすメリット: RPA は割り当てられたタスクをロボットが自動的かつ一貫して実行できるため、精度が向上します。また、RPA は付加価値のない業務を排除し、繰り返しの多い作業を処理するため、スピードと生産性も向上します。

前提条件と制限

前提条件

- アクティブな アカウント
- CloudFormation テンプレートをデプロイするための Identity and Access Management (IAM) 許可
- EC2 Image Builder でクロスアカウント AMI ディストリビューションをセットアップするための IAM ポリシー

アーキテクチャ

1. 管理者は `ec2-image-builder.yaml` ファイルにベースの Windows AMI を指定し、CloudFormation コンソールにスタックをデプロイします。
2. CloudFormation スタックは、以下のリソースを含む EC2 Image Builder パイプラインをデプロイします。
   - `Ec2ImageInfraConfiguration`
   - `Ec2ImageComponent`
   - `Ec2ImageRecipe`
   - `Ec2AMI`
3. EC2 Image Builder パイプラインは、ベース AMI を使用して一時的な Windows EC2 インスタンスを起動し、必要なコンポーネント (この場合は UiPath Studio) をインストールします。
4. EC2 Image Builder はすべてのホスト情報を削除し、Windows サーバーから AMI を作成します。
5. カスタム AMI で `ec2-provisioning yaml` ファイルを更新し、独自の要件に基づいて多数の EC2 インスタンスを起動します。

6. Count マクロは、CloudFormation テンプレートを使用してデプロイします。このマクロは CloudFormation リソースの Count プロパティを提供するので、同じタイプの複数のリソースを簡単に指定できます。
7. CloudFormation `ec2-provisioning.yaml` ファイル内のマクロの名前を更新し、スタックをデプロイします。
8. 管理者は要件に基づいて `ec2-provisioning.yaml` ファイルを更新し、スタックを起動します。
9. このテンプレートは UiPath Studio アプリケーションで EC2 インスタンスをデプロイします。

ツール

サービス

- AWS CloudFormation を使用すると、インフラストラクチャリソースを自動的かつ安全な方法でモデル化および管理できます。
- Amazon CloudWatch は、AWS、オンプレミス、その他のクラウド上のリソースとアプリケーションを観察およびモニタリングするのに役立ちます。
- Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウド内で安心で再サイズを変更できるコンピューティング性能を提供するウェブサービスです。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- EC2 Image Builder は、AWS またはオンプレミスで使用する仮想マシンとコンテナイメージの構築、テスト、デプロイを簡素化します。
- Amazon EventBridge は、AWS、既存のシステム、または Software as a Service (SaaS) アプリケーション全体にわたって、イベント駆動型アプリケーションを大規模に構築できるよう支援します。
- Identity and Access Management (IAM) は、AWS リソースへのアクセスを安全にコントロールするのに役立ちます。IAM を使用すると、ユーザーがアクセスできる AWS のリソースを制御するアクセス許可を集中管理できます。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- AWS Lambda は、サーバーレスのイベント駆動型のコンピューティングサービスで、サーバーのプロビジョニングや管理を行わなくても、実質あらゆるタイプのアプリケーションやバックエンドサービスのコードを実行できます。200 以上の AWS サービスや SaaS アプリケーションから Lambda 関数を呼び出すことができ、使用した分のみ料金が発生します。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。

- AWS Systems Manager Agent (SSM Agent) は、Systems Manager が EC2 インスタンス、エッジデバイス、オンプレミスサーバー、仮想マシン (VM) を更新、管理、構成するのに役立ちます。

コードリポジトリ

このパターンのコードは、GitHub 内の「CloudFormation を使用した UiPath RPA ボットのセットアップ」リポジトリで利用できます。このパターンでは、AWS CloudFormation マクロリポジトリから入手できるマクロも使用しています。

ベストプラクティス

- AWS は毎月新しい Windows AMI をリリースしています。これには最新の OS パッチ、ドライバー、起動エージェントが含まれます。新しいインスタンスを起動する際、または独自のカスタムイメージを作成する際は、最新の AMI を使用してください。
- イメージのビルド時には、Windows または Linux で使用可能なすべてのセキュリティパッチを適用してください。

エピック

ベースイメージ用のイメージパイプラインをデプロイする

タスク	説明	必要なスキル
EC2 Image Builder パイプラインをセットアップします。	1. CloudFormation を使用した UiPath RPA ボットのセットアップリポジトリをクローン作成するか、リポジトリから`ec2-image-builder.yaml` テンプレートをダウンロードします。 2. AWS マネジメントコンソールにサインインし、AWS CloudFront コンソールを開きます。 3. [スタックの作成] を選択してください。	AWS DevOps

タスク	説明	必要なスキル
	4. [Specify template (テンプレートの指定)] セクションで、[Upload a template file (テンプレートファイルのアップロード)] を選択します。 5. コンピュータにある `ec2-image-builder.yaml` テンプレートを見つけてアップロードし、[次へ] を選択します。 6. スタックの入力パラメータを指定するか、デフォルト値をそのまま使用します。[Next (次へ)] を選択します。 **Note** パラメータの数と値は、入力値によって異なる場合があります。 7. オプションで、スタックオプションを構成し、[次へ] を選択します。 8. スタックの詳細を確認します。 9. 画面の最後で、機能を確認するチェックボックスを選択し、[送信] を選択します。	

タスク	説明	必要なスキル
	10スタックの進行状況をモニタリングします。ステータスが CREATE_COMPLETE になったら、デプロイは準備完了です。	
EC2 Image Builder の設定を表示します。	EC2 Image Builder の設定には、インフラストラクチャ構成、ディストリビューション設定、セキュリティスキャン設定が含まれます。設定を表示するには: 1. EC2 Image Builder コンソールを開きます。 2. ナビゲーションペインで、Image Builder のさまざまな設定に移動します。 **Note** ベストプラクティスとして、EC2 Image Builder の更新は CloudFormation テンプレートを通じてのみ行う必要があります。	AWS DevOps

タスク	説明	必要なスキル
イメージパイプラインを表示します。	デプロイされたイメージパイプラインを表示するには: 1. EC2 Image Builder コンソールのナビゲーションペインで [イメージパイプライン] を選択します。 2. 作成したイメージパイプラインを選択します。 3. 出力イメージ、イメージレシピ、インフラストラクチャ構成、ディストリビューション設定、Amazon EventBridge ルール、タグの構成の詳細を表示します。	AWS DevOps

タスク	説明	必要なスキル
Image Builder のログを表示します。	EC2 Image Builder のログは CloudWatch ロググループに集約されます。CloudWatch コンソールでのログを表示するには: 1. CloudWatch コンソールを開きます。 2. ナビゲーションペインで、[ログ]、[ロググループ] の順に選択します。 3. ロググループの名前を選択します。EC2 Image Builder のログはロググループ `/aws/imagebuilder/XXX` に集約されます。 4. イメージパイプラインの実行中にエラーが発生していないか、それぞれのログストリームの最新のログを確認してください。 EC2 Image Builder のログも S3 バケットに保存されます。バケット内のログを表示するには: 1. Amazon S3 コンソールを開きます。 2. [バケット] リストで、バケット名を選択します。ログは S3 バケット `<stack-name>-XXXXXX` に集約されます。	AWS DevOps

タスク	説明	必要なスキル
S3 バケットに UiPath ファイルをアップロードする。	1. UiPath Studio 用の `.msi` ファイルを https://download.uipath.com/UiPathStudioCommunity.msi からダウンロードします。 2. S3 バケットに ファイルをアップロードします。 3. `ec2-image-builder.yaml` テンプレートのユーザーデータセクションの 310 行目にあるバケット名とファイルキーを更新します。	AWS DevOps

Count マクロのデプロイとテスト

タスク	説明	必要なスキル
Count マクロをデプロイします。	1. Count CloudFormation マクロをクローン作成またはダウンロードします。 2. `Count` フォルダに移動します。 3. CloudFormation アーティファクトを保存するには S3 バケットが必要です。S3 バケットがまだない場合は、`aws s3 mb s3://<bucket name>` の名前で作成します。 4. Count マクロテンプレートをパッケージ化します。テンプレートは Serverless	DevOps エンジニア

タスク	説明	必要なスキル

	Application Model (SAM) を使用するため、デプロイする前に変換する必要があります。
	
	`aws cloudformation package \ --template-file template.yaml \ --s3-bucket <your bucket name here> \ --output-template-file packaged.yaml`
	
	以下に例を示します。
	
	`aws cloudformation package \ --template-file template.yaml \ --s3-bucket count-macro-ec2 \ --output-template-file packaged.yaml`
	
	5. パッケージ化されたテンプレートを使用して CloudFormation スタックを作成するには
	
	`aws cloudformation deploy \ --stack-name Count-macro \ --template-file packaged.yaml \`	

タスク	説明	必要なスキル
	`--capabilities CAPABILITY_IAM` コンソールを使用する場合は、前のエピックまたは CloudFormation ドキュメントの指示に従ってください。	
Count マクロをテストします。	マクロの機能をテストするには、マクロに付属しているサンプルテンプレートを起動してみてください。 `aws cloudformation deploy \ --stack-name Count-test \ --template-file test.yaml \ --capabilities CAPABILITY_IAM`	DevOps エンジニア

CloudFormation スタックをデプロイし、カスタムイメージでインスタンスをプロビジョニングする

タスク	説明	必要なスキル
Amazon EC2 プロビジョニングテンプレートをデプロイする。	CloudFormation を使用して EC2 イメージパイプラインをデプロイするには: 1. GitHub リポジトリから `ec2-provisioning.yaml` テンプレートをダウンロードするか、リポジトリをクローン作成した場合	AWS DevOps

タスク	説明	必要なスキル
	はコンピューター上でテンプレートを検索します。 2. CloudFormation コンソールを開きます。 3. 最初のエピックの手順を繰り返して (または CloudFormation ドキュメントの指示に従って) `ec2-provisioning.yaml` をデプロイします。	
Amazon EC2 の設定を表示する。	Amazon EC2 の構成には、セキュリティ、ネットワーク、ストレージ、ステータスチェック、モニタリング、タグの構成が含まれます。これらの構成を表示するには: 1. Amazon EC2 コンソールを開きます。 2. ナビゲーションペインで [インスタンス] を選択し、Amazon EC2 プロビジョニングテンプレートで作成した EC2 インスタンスを選択します。 3. インスタンスの概要で、タブを選択すると、対応する Amazon EC2 設定が表示されます。	AWS DevOps

タスク	説明	必要なスキル
CloudWatch ダッシュボードを表示する。	1. CloudWatch コンソールを開きます。 2. ナビゲーションペインで、ダッシュボードを選択します。 3. スタック名のあるダッシュボードを選択します。 **Note** スタックをプロビジョニングした後、ダッシュボードにメトリクスを入力するのに時間がかかります。 ダッシュボードには次のメトリクスが表示されます: `CPUUtilization` 、`DiskUtilization` 、`MemoryUtilization` 、`NetworkIn` 、`NetworkOut` 、`StatusCheckFailed` 。	AWS DevOps

タスク	説明	必要なスキル
メモリとディスクの使用状況に関するカスタムメトリックスを表示する。	1. CloudWatch コンソールで [ダッシュボード] を選択します。 2. ナビゲーションペインで、[Metrics]、[All metrics] を選択します。 3. [カスタム名前空間]、[CWAgent] を選択します。	AWS DevOps
メモリとディスクの使用状況に関するアラームを表示する。	1. CloudWatch コンソールで、ナビゲーションペインの [ダッシュボード] を選択します。 2. [Add alarm] (アラームを追加) を選択します。	AWS DevOps
スナップショットのライフサイクルルールを確認します。	1. Amazon EC2 コンソールを開きます。 2. ナビゲーションペインでLifecycle Managerを選択します。 3. AMI ライフサイクルの設定を確認します。	AWS DevOps

環境を削除する (オプション)

タスク	説明	必要なスキル
スタックを削除します。	PoC またはパイロットプロジェクトが完了したら、作成したスタックを削除して、これらのリソースに対して課金されないようにすることをお勧めします。	AWS DevOps

タスク	説明	必要なスキル
	1. AWS CloudFormation コンソールを開きます。 2. ナビゲーションペインで [スタック] を選択し、前に作成したスタックのうち、削除するスタックのうちの1つまたは両方を選択します。スタックは現在実行中である必要があります。 3. [スタックの詳細] ペインで、[削除] を選択します。 4. プロンプトが表示されたら、[スタックの削除] を選択します。 **Important** スタックの削除オペレーションは、開始後に停止できません。スタックは DELETE_IN_PROGRESS 状態になります。 削除が失敗した場合、スタックは DELETE_FAILED 状態になります。解決策については、AWS CloudFormation トラブルシューティングドキュメントの「スタックの削除失敗」を参照してください。	

タスク	説明	必要なスキル
	スタックが誤って削除されないように保護する方法については、AWS CloudFormation ドキュメントの「スタックが削除されないように保護する」を参照してください。	

トラブルシューティング

問題	ソリューション
Amazon EC2 プロビジョニングテンプレートをデプロイすると、「トランスフォーム 123xxxx:: Count から不正な形式の応答を受信しました」というエラーが表示される。	これは既知の問題です。(AWS CloudFormation マクロリポジトリのカスタムソリューションと PR を参照してください)。 この問題を解決するには、AWS Lambda コンソールを開き、GitHub リポジトリのコンテンツで `index.py` を更新します。

関連リソース

GitHub リポジトリ

- CloudFormation を使用した UiPath RPA ボットのセットアップ
- Count CloudFormation Macro

AWS リファレンス

- AWS CloudFormation コンソールでのスタックの作成 (CloudFormation ドキュメント)
- CloudFormation のトラブルシューティング (CloudFormation ドキュメント)
- Amazon EC2 Linux インスタンスのメモリとディスクのメトリクスのモニタリング (Amazon EC2 ドキュメント)
- CloudWatch エージェントを使用して Windows サーバー上のパフォーマンスモニターのメトリクスを表示する方法を教えてください。(AWS re: POST の記事)

その他の参考資料

- UiPathドキュメント
- Syspreped AMI でホスト名を設定する (Brian Beachによるブログ記事)
- パラメータが変更されたときに Cloudformation にマクロを使用してテンプレートを再処理させるにはどうすればよいですか? (スタックオーバーフロー)

AWS で可用性の高い PeopleSoft アーキテクチャを設定する

作成者: Ramanathan Muralidhar (AWS)

概要

PeopleSoft のワークロードを AWS に移行する場合、回復性は重要な目標です。これにより、PeopleSoft アプリケーションの可用性が常に高くなり、障害から迅速に回復できるようになります。

このパターンは、ネットワーク、アプリケーション、データベースの各層で高可用性 (HA) を確保するための AWS での PeopleSoft アプリケーションのアーキテクチャを設計する方法を提供します。データベース層には、Amazon Relational Database Service (Amazon RDS) for Oracle、Amazon RDS for SQL Server データベースを使用します。このアーキテクチャには、Amazon Route 53、Amazon Elastic Compute Cloud (Amazon EC2) Linux インスタンス、Amazon Elastic Block Storage (Amazon EBS)、Amazon Elastic File System (Amazon EFS)、Application Load Balancer などの AWS サービスも含まれており、スケーラブルです。

Oracle PeopleSoft は、人員管理や他のビジネス運営のためのツールとアプリケーション一式を提供しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS での設定に必要なライセンスを持つ PeopleSoft 環境
- AWS アカウントで設定された仮想プライベートクラウド (VPC) には、次のリソースがあります。
  - 少なくとも 2 つのアベイラビリティーゾーン
  - 各アベイラビリティーゾーンに 1 つのパブリックサブネットと 3 つのプライベートサブネットがあります
  - NAT ゲートウェイとインターネットゲートウェイ
  - トラフィックをルーティングする各サブネットのルートテーブル
  - 組織の標準に従って PeopleSoft アプリケーションのセキュリティを確保するために定義されたネットワークアクセスコントロールリスト (ネットワーク ACL) とセキュリティグループ

機能制限

- このパターンは、高可用性 (HA) ソリューションを実現します。　ディザスタリカバリ (DR) シナリオはサポートされていません。　まれに、HA 実装の AWS リージョン全体が停止した場合、アプリケーションは使用できなくなります。

製品バージョン

- PeopleTools 8.52 以降を実行している PeopleSoft アプリケーション

アーキテクチャ

ターゲット アーキテクチャ

PeopleSoft 実稼働アプリケーションのダウンタイムや停止は、アプリケーションの可用性に影響を与え、ビジネスに多大な混乱をもたらします。

PeopleSoft 実稼働アプリケーションは、常に高い可用性が得られるように設計することをお勧めします。これは、単一障害点をなくし、信頼性の高いクロスオーバーポイントまたはフェイルオーバーポイントを追加し、障害を検出することで実現できます。次の図では、PeopleSoft on AWS の HA アーキテクチャを示しています。

このアーキテクチャのデプロイでは、PeopleSoft データベースとして Amazon RDS for Oracle を使用し、Red Hat Enterprise Linux (RHEL) 上で実行される EC2 インスタンスを使用しています。Amazon RDS for SQL Server を Peoplesoft データベースとして使用することもできます。

このアーキテクチャには、以下のコンポーネントが含まれています。

- Amazon Route 53 は、インターネットから PeopleSoft アプリケーションにリクエストをルーティングするためのドメインネームサーバー (DNS) として使用されます。
- AWS WAF は、可用性に影響を与え、セキュリティを侵害し、過剰なリソースを消費する可能性のある一般的なウェブの脆弱性とボットからの保護に役立ちます。AWS Shield Advanced (図には示されていません) は、はるかに幅広い保護を提供します。
- Application Load Balancer は、ウェブサーバーをターゲットとした高度なリクエストルーティングにより、HTTP トラフィックと HTTPS トラフィックの負荷を分散します。
- PeopleSoft アプリケーションをサポートするウェブサーバー、アプリケーションサーバー、プロセススケジューラーサーバー、および Elasticsearch サーバーは、複数のアベイラビリティーゾーンで実行され、Amazon EC2 Auto Scaling を使用します。

- PeopleSoft アプリケーションが使用するデータベースは、Amazon RDS 上でマルチ AZ 構成で動作します。
- PeopleSoft アプリケーションが使用するファイル共有は Amazon EFS で設定され、インスタンス間でファイルにアクセスするために使用されます。
- Amazon マシンイメージ (AMI) は、Amazon EC2 Auto Scaling によって使用され、必要なときに PeopleSoft コンポーネントが迅速に複製されます。
- NAT ゲートウェイを使用すると、プライベートサブネット内のインスタンスは VPC 外のサービスに接続できますが、外部サービスはそれらのインスタンスとの接続を開始できません。
- インターネットゲートウェイは、VPC とインターネットとの間の通信を可能にする VPC コンポーネントであり、冗長性と高い可用性を備えており、水平スケーリングが可能です。
- パブリックサブネットの踏み台ホストは、インターネットや内部ネットワークなどの外部ネットワークから、オンプレミスサブネット上のサーバーへのアクセスを提供します。踏み台ホストは、プライベートサブネット内のサーバーへの制御されたセキュアなアクセスを提供します。

アーキテクチャの詳細

PeopleSoft データベースは、マルチ AZ 設定の Amazon RDS for Oracle (または Amazon RDS for SQL Server) データベースに格納されています。 Amazon RDS の Amazon RDS マルチ AZ は、2 つのアベイラビリティーゾーン間でデータベース更新を複製して耐久性と可用性を高める機能です。Amazon RDS は、予定されたメンテナンスや予定外の中断に備えて、自動的にスタンバイデータベースにフェイルオーバーします。

PeopleSoft のウェブ層と中間層は EC2 インスタンスにインストールされます。これらのインスタンスは複数のアベイラビリティーゾーンに分散され、Auto Scaling グループによって結合されます。これにより、これらのコンポーネントでは常に高可用性が保証されます。アプリケーションが常に利用可能で、必要に応じてスケールできるように、必要最小限のインスタンス数が維持されます。

OEM EC2 インスタンスには、現行世代の EC2 インスタンスタイプを使用することをお勧めします。AWS Nitro System で構築されたインスタンスなどの現世代のインスタンスタイプは、ハードウェア仮想マシン (HVM) をサポートします。HVM AMI は、拡張ネットワーキングのメリットを活用する場合に必要であり、セキュリティも強化されています。各 Auto Scaling グループの一部である EC2 インスタンスは、インスタンスの交換またはスケールアップ時に独自の AMI を使用します。EC2 インスタンスタイプは、PeopleSoft アプリケーションが処理する負荷と、Oracle が PeopleSoft アプリケーションとPeopleTools のバージョンに推奨される最小値に基づいて選択することをお薦めします。ハードウェア要件とソフトウェア要件の詳細については、Oracle サポートのウェブサイトを参照してください。

PeopleSoft のウェブ層と中間層は Amazon EFS マウントを共有して、レポート、データファイル、および (必要に応じて) PS_HOME ディレクトリを共有します。Amazon EFS は、パフォーマンスとコストの観点から、各アベイラビリティーゾーンのマウントターゲットで設定されています。

Application Load Balancer は、PeopleSoft アプリケーションにアクセスするトラフィックをサポートするようにプロビジョニングされ、異なるアベイラビリティーゾーンにまたがるウェブサーバー間でトラフィックの負荷を分散します。Application Load Balancer は、最低 2 つのアベイラビリティーゾーンで HA を提供するネットワークデバイスです。　ウェブサーバーは、ロードバランサー設定を使用してトラフィックを異なるアプリケーションサーバーに分散します。ウェブサーバーとアプリケーションサーバー間のロードバランシングにより、負荷がインスタンス全体に均等に分散され、インスタンスの過負荷によるボトルネックやサービスの中断を回避できます。

Amazon Route 53 は、インターネットから Application Load Balancer にトラフィックをルーティングする DNS サービスとして使用されます。Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

HA の詳細

- データベース: Amazon RDS のマルチ AZ 機能は、同期レプリケーションを使用して、複数のアベイラビリティゾーンで 2 つのデータベースを実行します。これにより、自動フェイルオーバーによる可用性の高い環境が構築されます。　Amazon RDS にはフェイルオーバーイベント検出機能があり、イベントが発生すると自動フェイルオーバーを開始します。Amazon RDS API を使用して手動フェイルオーバーを開始することもできます。詳細な説明については、ブログ記事「Amazon RDS Under The Hood:マルチ AZ」を参照してください。フェイルオーバーはシームレスで、フェイルオーバーが発生するとアプリケーションは自動的にデータベースに再接続します。ただし、フェイルオーバー中のプロセス スケジューラ ジョブはいずれもエラーが発生し、再送信する必要があります。
- PeopleSoft アプリケーションサーバー:アプリケーションサーバーは複数のアベイラビリティーゾーンに分散され、Auto Scaling グループが定義されています。　インスタンスに障害が発生した場合、Auto Scaling グループは、アプリケーションサーバーテンプレートの AMI から複製された正常なインスタンスに、そのインスタンスを置き換えます。具体的には、jolt プーリングが有効になっているため、アプリケーションサーバーのインスタンスが停止すると、セッションは自動的に別のアプリケーションサーバーにフェイルオーバーされ、Auto Scaling グループは自動的に別のインスタンスを起動し、アプリケーションサーバーを起動して Amazon EFS マウントに登録します。新しく作成されたアプリケーションサーバーは、ウェブサーバー内の PSSTRSETUP.SH スクリプトを使用して自動的にウェブサーバーに追加されます。これにより、PeopleSoft アプリケーションの可用性が常に高く、障害から迅速に回復できるようになります。

- プロセス スケジューラー: プロセス スケジューラーサーバーは複数のアベイラビリティゾーンに分散しており、Auto Scaling グループが定義されています。インスタンスに障害が発生した場合、Auto Scaling グループは、プロセス スケジューラー サーバーテンプレートの AMI からクローンされた正常なインスタンスに、そのインスタンスを置き換えます。具体的には、プロセス スケジューラインスタンスが停止すると、Auto Scaling グループは自動的に別のインスタンスを起動し、プロセス スケジューラを起動します。インスタンスに障害が発生した場合、実行中のジョブはすべて再送信する必要があります。これにより、には、プロセス スケジューラの可用性が常に高く、障害から迅速に回復できるようになります。
- Elasticsearch サーバー: Elasticsearch サーバーには、Auto Scaling グループが定義されています。インスタンスに障害が発生した場合、Auto Scaling グループは、Elasticsearch サーバーテンプレートの AMI からクローンされた正常なインスタンスに、そのインスタンスを置き換えます。具体的には、Elasticsearch インスタンスが停止すると、リクエストを処理する Application Load Balancer がエラーを検出し、そのインスタンスへのトラフィックの送信を停止します。Auto Scaling グループは自動的に別のインスタンスを起動し、Elasticsearch インスタンスを起動します。Elasticsearch インスタンスがバックアップされると、Application Load Balancer はインスタンスが正常であることを検出し、リクエストの送信を再開します。これにより、Elasticsearch サーバーの可用性が常に高く、障害から迅速に回復できるようになります。
- ウェブサーバー: ウェブサーバーには、Auto Scaling グループが定義されています。インスタンスに障害が発生した場合、Auto Scaling グループは、ウェブサーバーテンプレートの AMI からクローンされた正常なインスタンスに、そのインスタンスを置き換えます。具体的には、ウェブサーバーのインスタンスが停止すると、リクエストを処理する Application Load Balancer がエラーを検出し、そのインスタンスへのトラフィックの送信を停止します。Auto Scaling グループは自動的に別のインスタンスを起動し、ウェブサーバーのインスタンスを起動します。ウェブサーバーのインスタンスがバックアップされると、Application Load Balancer はインスタンスが正常であることを検出し、リクエストの送信を再開します。これにより、ウェブサーバーの可用性が常に高く、障害から迅速に回復できるようになります。

ツール

AWS サービス

- アプリケーションロードバランサーは、受信アプリケーショントラフィックを複数のアベイラビリティーゾーンの EC2 インスタンスなどの複数のターゲットに分散します。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Elastic File System (Amazon EFS) は、AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。

ベストプラクティス

運用のベストプラクティス

- PeopleSoft on AWS を実行する場合、Route 53 を使用してインターネットとローカルからのトラフィックをルーティングします。プライマリ DB インスタンスを使用できない場合は、フェイルオーバーオプションを使用してトラフィックをディザスタリカバリ (DR) サイトに再ルーティングします。
- Application Load Balancerは、常に PeopleSoft 環境の前で使用してください。これにより、トラフィックの負荷がウェブサーバーにセキュアに分散されます。
- Application Load Balancer のターゲットグループ設定で、ロードバランサーが生成した Cookie で維持設定がオンになっていることを確認します。

> **Note**
>
> 外部シングルサインオン (SSO) を使用する場合は、アプリケーションベースの Cookie を使用する必要がある場合があります。これにより、ウェブサーバーとアプリケーションサーバー間の接続が一貫していることが保証されます。

- PeopleSoft 実稼働アプリケーションの場合、Application Load Balancer のアイドルタイムアウトは、使用するウェブ プロファイルで設定されている値と一致する必要があります。これにより、ユーザーセッションがロードバランサーのレイヤーで期限切れになるのを防止できます。
- PeopleSoft 実稼働アプリケーションの場合は、アプリケーションサーバーの「再利用回数」をメモリーリークを最小限に抑える値に設定します。
- このパターンで説明されているように PeopleSoft 実稼働アプリケーションに Amazon RDS データベースを使用している場合は、高可用性を実現するためにマルチ AZ 形式で実行します。

- データベースが PeopleSoft 実稼働アプリケーションの EC2 インスタンスで実行されている場合は、高可用性を実現するためにスタンバイデータベースが別のアベイラビリティーゾーンで実行されていることを確認してください。
- DR では、Amazon RDS データベースまたは EC2 インスタンスに、実稼働データベースとは別の AWS リージョンにスタンバイが設定されていることを確認してください。これにより、そのリージョンで障害が発生した場合に、アプリケーションを別のリージョンに切り替えることができます。
- DR では、Amazon Elastic ディザスタリカバリを使用して、実稼働コンポーネントとは別のリージョンにアプリケーションレベルのコンポーネントを設定します。これにより、そのリージョンで障害が発生した場合に、アプリケーションを別のリージョンに切り替えることができるようになります。
- PeopleSoft のレポート、添付ファイル、およびデータファイルを保存するには、Amazon EFS (中程度の I/O 要件の場合) または Amazon FSx (厳しい I/O 要件の場合) を使用します。これにより、コンテンツは 1 か所に保存され、インフラストラクチャ内のどこからでもアクセスできます。
- Amazon CloudWatch (基本および詳細) を使用して、PeopleSoft アプリケーションが使用している AWS クラウドのリソースをほぼリアルタイムでモニタリングします。これにより、問題が即座にプッシュされ、環境の可用性に影響を与える前に迅速に対処できます。
- Amazon RDS データベースを PeopleSoft データベースとして使用している場合は、拡張モニタリングを使用してください。この機能により、CPU、メモリ、ファイルシステム I/O、ディスク I/O など、50 を超えるメトリクスにアクセスできます。
- AWS CloudTrail を使用して、PeopleSoft アプリケーションが使用している AWS リソースでの API コールをモニタリングします。これにより、セキュリティ分析、リソース変更の追跡、およびコンプライアンスのモニタリングを行うことができます。

セキュリティベストプラクティス

- SQL インジェクションやクロスサイトスクリプティング (XSS) 攻撃など、一般的なエクスプロイトから PeopleSoft アプリケーションを保護するには、AWS WAF を使用します。カスタマイズされた検出および軽減サービスには、AWS Shield Advanced の使用を検討してください。
- PeopleSoft アプリケーションのセキュリティを確保するために、トラフィックを HTTP から HTTPS に自動的にリダイレクトするルールを Application Load Balancer に追加します。
- Application Load Balancer に別のセキュリティグループを設定します。このセキュリティグループは HTTPS/HTTP インバウンドトラフィックのみを許可し、アウトバウンドトラフィックは許可しません。これにより、意図したトラフィックのみが許可されるため、アプリケーションのセキュリティが確保されます。

- アプリケーションサーバー、ウェブサーバー、データベースにはプライベートサブネットを使用し、アウトバウンドのインターネットトラフィックには NAT ゲートウェイを使用します。これにより、アプリケーションをサポートするサーバーにパブリックにアクセスできないようにし、必要なサーバーにのみパブリック アクセスを提供します。
- PeopleSoft の実稼働環境と非実稼働環境では、異なる VPC を使用します。 AWS Transit Gateway、VPC ピアリング、ネットワーク ACL、セキュリティグループを使用して VPC 間で、また必要に応じてオンプレミスデータセンターでトラフィックフローを制御します。
- 最小権限の原則に従います。PeopleSoft アプリケーションが使用する AWS リソースへのアクセス権は、必ず必要なユーザーにのみ付与されます。タスクの実行に必要な最低限の権限のみを付与します。詳細については、AWS Well-Architected フレームワークの「セキュリティの柱」を参照してください。
- 可能な限り、AWS Systems Manager を使用して PeopleSoft アプリケーションが使用する EC2 インスタンスにアクセスします。

信頼性のベストプラクティス

- Application Load Balancer を使用するときは、有効なアベイラビリティゾーンごとに ターゲットを 1 つ登録します。これにより、ロードバランサーの効率が最高になります。
- PeopleSoft の実稼働環境ごとに、アプリケーションにアクセスするための URL、統合ブローカーにサービスを提供する URL、レポートを表示するための URL 、この 3 つの異なる URL を用意することをお勧めします。可能であれば、各 URL には独自の専用ウェブサーバーとアプリケーションサーバーが必要です。この URL にはそれぞれ異なる機能があり、アクセスが制御されるため、このデザインは PeopleSoft アプリケーションのセキュリティを高めるのに役立ちます。さらに、基盤となるサービスに障害が発生した場合の影響範囲を最小限に抑えることができます。
- PeopleSoft アプリケーションのロードバランサーのターゲットグループに対してヘルスチェックを構成することをお勧めします。ヘルスチェックは、EC2 インスタンスではなく、ウェブサーバーで実行する必要があります。これにより、ウェブサーバーがクラッシュしたり、ウェブサーバーをホストする EC2 インスタンスがダウンした場合でも、Application Load Balancer は正確な情報を反映します。
- PeopleSoft の実稼働アプリケーションでは、ウェブサーバーを少なくとも 3 つのアベイラビリティゾーンに分散させることをお勧めします。これにより、アベイラビリティゾーンの 1 つがダウンした場合の高可用性を常に維持できます。
- PeopleSoft 実稼働アプリケーションでは、jolt プール (`joltPooling=true`) を有効にします。これにより、パッチ適用や仮想マシン障害によってサーバーが停止した場合でも、アプリケーションが別のアプリケーションサーバーに確実にフェイルオーバーされます。

- PeopleSoft 実稼働アプリケーションの場合は、`DynamicConfigReload` を 1 に設定します。この設定は、PeopleTools バージョン 8.52 以降でサポートされています。サーバーを再起動せずに、新規アプリケーションサーバーをウェブサーバーに動的に追加します。
- PeopleTools パッチを適用するときのダウンタイムを最小限に抑えるには、ウェブサーバーとアプリケーションサーバーの Auto Scaling グループの起動設定にブルー/グリーンデプロイ方法を使用してください。詳細については、ホワイトペーパー「AWSデプロイオプションの概要」を参照してください。
- AWS Backup を使用して、AWS 上の PeopleSoft アプリケーションをバックアップします。AWS Backup は費用対効果が高く、フルマネージド型のポリシーベースのサービスで、大規模なデータ保護を簡素化します。

パフォーマンスに関するベストプラクティス

- ビジネスで環境全体のトラフィックを暗号化する必要がある場合を除き、PeopleSoft 環境でパフォーマンスを最適化するには、Application Load Balancer で SSL を終了します。
- Amazon Simple Notification Service (Amazon SNS) や CloudWatch などの AWS サービスのインターフェイス VPC エンドポイントを作成して、トラフィックが常に内部に送られるようにします。これは費用対効果が高く、アプリケーションのセキュリティを確保します。

コスト最適化のベストプラクティス

- PeopleSoft 環境で使用されるすべてのリソースにタグを付け、コスト配分タグを有効にします。これらのタグは、リソースコストの表示と管理に役立ちます。
- PeopleSoft 実稼働アプリケーションでは、ウェブサーバーとアプリケーションサーバーに Auto Scaling グループを設定します。 これにより、アプリケーションをサポートするための最小限のウェブサーバーとアプリケーションサーバーが維持されます。Auto Scaling グループポリシーを使用して、必要に応じてサーバーをスケールアップまたはスケールダウンできます。
- 請求アラームを使用すると、コストが指定した予算のしきい値を超えたときにアラートを受け取ることができます。

サステナビリティのベストプラクティス

- Infrastructure as Code (IaC) を使用して PeopleSoft 環境を維持してください。これにより、一貫性のある環境を構築し、変更管理を維持できます。

エピック

PeopleSoft データベースを Amazon RDS に移行する

タスク	説明	必要なスキル
DB サブネットグループを作成します。	Amazon RDS コンソールのナビゲーションペインで [サブネットグループ] を選択し、複数のアベイラビリティゾーンにサブネットを持つ Amazon RDS DB サブネットグループを作成します。これは、Amazon RDS データベースをマルチ AZ 設定で実行する時に必要です。	クラウド管理者
Amazon RDS データベースを作成します。	PeopleSoft HA 環境用に選択した AWS リージョンのアベイラビリティゾーンに Amazon RDS データベースを作成します。Amazon RDS データベースを作成するときは、マルチ AZ オプション (スタンバイインスタンスを作成) と前のステップで作成したデータベースのサブネットグループを必ず選択してください。詳細については、「Amazon RDS ドキュメント」を参照してください。	クラウド管理者、Oracle データベース管理者
PeopleSoft データベースを Amazon RDS に移行します。	AWS Database Migration Service (AWS DMS) を使用して、既存の PeopleSoft データベースを Amazon RDS データベースに移行します。詳細については、「AWS DMS ド	クラウド管理者、PeopleSoft DBA

タスク	説明	必要なスキル
	キュメント」および AWS ブログ記事「DMS を使用してほぼゼロのダウンタイムで Oracle データベースを移行する」参照してください。	

Amazon EFS ファイルシステムをマウントする

タスク	説明	必要なスキル
ファイルシステムを作成します。	Amazon EFS コンソールで、ファイルシステムを作成し、各アベイラビリティゾーンのターゲットをマウントします。手順については、「Amazon EC2 ドキュメント」を参照してください。ファイルシステムを作成したら、その DNS 名を書き留めます。ファイルシステムをマウントするときに、この情報を使用します。	クラウド管理者

PeopleSoft アプリケーションとファイルシステムを設定する

タスク	説明	必要なスキル
EC2 インスタンスを起動します。	PeopleSoft アプリケーションの EC2 インスタンスを起動します。手順については、「Amazon EC2 ドキュメント」を参照してください。	クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル
	• [名前] に APP_TEMPLATE と入力します。 • [OS イメージ] には、Red Hat を選択してください。 • [インスタンスタイプ] には、PeopleSoft アプリケーションに適したインスタンスタイプを選択します。詳細については、「アーキテクチャ」セクションの「アーキテクチャの詳細」を参照してください。	
PeopleSoft をインスタンスにインストールします。	作成した EC2 インスタンスに PeopleSoft アプリケーションと PeopleTools をインストールします。手順については、「Oracle ドキュメント」を参照してください。	クラウド管理者、PeopleSoft 管理者
アプリケーションサーバーを作成します。	AMI テンプレート用のアプリケーションサーバーを作成し、Amazon RDSデータベースに正常に接続していることを確認します。	クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル

Amazon EFS ファイルシステムをマウントします。	ルートユーザーとして EC2 インスタンスにログインし、次のコマンドを実行してサーバー上の PSFTMNT というフォルダに、Amazon EFS ファイルシステムをマウントします。
	`sudo su -`
	`mkdir /psftmnt`
	`cat /etc/fstab`
	次の行を /etc/fstab ファイルに追加します。ファイルシステムを作成した際に書き留めた DNS 名を使用します。
	`fs-09e064308f11453 88.efs.us-east-1.a mazonaws.com:/ / psftmnt nfs4 nfsvers=4 .1,rsize=1048576,w size=1048576,hard, timeo=600,retrans= 2,noresvport,_netdev 0 0`
	`mount -a`
		クラウド管理者、PeopleSoft 管理者

アクセス許可を確認します。	PeopleSoft ユーザーが適切にアクセスできるように、PSFTMNT フォルダに適切な権限があることを確認してください。	クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル
追加のインスタンスを作成します。	このエピックの前のステップを繰り返して、プロセス スケジューラー、ウェブサーバー、Elasticsearch サーバー用のテンプレートインスタンスを作成します。 これらのインスタンスには、PRCS_TEMPLATE 、WEB_TEMPLATE 、SRCH_TEMPLATE という名前を付けます。ウェブサーバーには、joltPooling=true と DynamicConfigReload=1 を設定します。	クラウド管理者、PeopleSoft 管理者

スクリプトを作成してサーバーを設定します。

タスク	説明	必要なスキル
アプリケーションサーバーをインストールするスクリプトを作成します。	Amazon EC2 APP_TEMPLATE インスタンスで、PeopleSoft ユーザーとして次のスクリプトを作成します。appstart.sh という名前を付け、PS_HOME ディレクトリに配置します。このスクリプトを使用してアプリケーションサーバーを起動し、Amazon EFS マウントにサーバー名を記録します。 `#!/bin/ksh`	PeopleSoft 管理者

タスク	説明	必要なスキル
	``` . /usr/homes/hcmdemo /.profile. psadmin -c configure -d HCMDEMO psadmin -c parallelb oot -d HCMDEMO touch /psftmnt/`echo  $HOSTNAME` ```	

タスク	説明	必要なスキル

プロセス スケジューラサーバーをインストールするスクリプトを作成します。	Amazon EC2 PRCS_TEMPLATE インスタンスで、PeopleSoft ユーザーとして次のスクリプトを作成します。prcsstart.sh という名前を付け、PS_HOME ディレクトリに配置します。このスクリプトを使用して、プロセス スケジューラサーバーを起動します。
	
	`#!/bin/ksh`
	`. /usr/homes/hcmdemo/.profile`
	`/* The following line ensures that the process scheduler always has a unique name during replacement or scaling activity. */`
	`sed -i "s/.*PrcsServerName.*/`hostname -I | awk -F. '{print "PrcsServerName=PSUNX"$3$4}'`/" $HOME/appserv/prcs/*/psprcs.cfg`
	`psadmin -p configure -d HCMDEMO`
	`psadmin -p start -d HCMDEMO`
		PeopleSoft 管理者

タスク	説明	必要なスキル

Elasticsearch サーバーをインストールするスクリプトを作成します。	Amazon EC2 SRCH_TEMPLATE インスタンスで、Elasticsearch ユーザーとして次のスクリプトを作成します。srchstart.sh という名前を付け、HOME ディレクトリに配置します。
	`#!/bin/ksh`
	`/* The following line ensures that the correct IP is indicated in the elasticsearch.yaml file. */`
	`sed -i "s/.*network.host.*/`hostname -I
		awk '{print "host:"$0}'`/" $ES_HOME_DIR/config/elasticsearch.yaml`
		`nohup $ES_HOME_DIR/bin/elasticsearch &`
			PeopleSoft 管理者

タスク	説明	必要なスキル

ウェブサーバーをインストールするスクリプトを作成します。	Amazon EC2 WEB_TEMPLATE インスタンスで、ウェブサーバーユーザーとして、HOME ディレクトリに次のスクリプトを作成します。
	
	renip.sh: このスクリプトは、AMI からクローンした際にウェブサーバに正しい IP が割り当てられていることを確認します。
	
	`#!/bin/ksh`
	`hn=`hostname``
	`/* On the following line, change the IP with the hostname with the hostname of the web template. */`
	`for text_file in `find * -type f -exec grep -l '<hostname-of-the-web-template>' {} \;`do`
	`sed -e 's/<hostname-of-the-web-template>/'$hn'/g' $text_file > temp`
	`mv -f temp $text_file`
	`done`
	
	psstrsetup.sh ：このスクリプトは、ウェブサーバーが現在実行している正しいアプリケーションサーバー IP を使用することを確認します。jolt ポート上の各アプリケーショ
		PeopleSoft 管理者

タスク	説明	必要なスキル
	ンサーバーへの接続を試み、設定ファイルに追加します。 `#!/bin/ksh` `c2=""` `for ctr in `ls -1 /psftmnt/*.internal` `do` `c1=`echo $ctr | awk -F "/" '{print $3}'` `/* In the following lines, 9000 is the jolt port. Change it if necessary. */` `if nc -z $c1 9000 2> /dev/null; then` `if [[ $c2 = "" ]]; then` `c2="psserver="`echo $c1`":9000"` `else` `c2=`echo $c2`","`echo $c1`":9000"` `fi` `fi` `done` webstart.sh：このスクリプトは前の 2 つのスクリプトを実行して、ウェブサーバーを起動します。 `#!/bin/ksh` `/* Change the path in the following if necessary. */` `cd /usr/homes/hcmdemo` `./renip.sh` `./psstrsetup.sh`	

タスク	説明	必要なスキル
	`webserv/peoplesoft/bin/startPIA.sh`	
crontab エントリを追加します。	Amazon EC2 `WEB_TEMPLATE` インスタンスで、ウェブサーバーユーザーとして、crontab に次のスクリプトを作成します。時間とパスを変更して、必要な値を反映させます。このエントリにより、ウェブサーバの `configuration.properties` ファイル内にあるアプリケーションサーバーエントリが常に正しいことが保証されます。 `* * * * * /usr/homes/hcmdemo/psstrsetup.sh`	PeopleSoft 管理者

AMI と Auto Scaling グループテンプレートを作成する

タスク	説明	必要なスキル
アプリケーションサーバーテンプレート用の AMI を作成します。	Amazon EC2 コンソールで、Amazon EC2 `APP_TEMPLATE` インスタンスの AMI イメージを作成します。AMI を `PSAPPSRV-SCG-VER1` と名付けます。手順については、「Amazon EC2 ドキュメント」を参照してください。	クラウド管理者、PeopleSoft 管理者
別サーバー用の AMI を作成します。	このエピックの前のステップを繰り返して、プロセス ス	クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル
	ケジューラー、Elasticsearch サーバー、ウェブサーバー用のテンプレートインスタンスを作成します。	

アプリケーションサーバーの Auto Scaling グループの起動テンプレートを作成します。	アプリケーションサーバーの Auto Scaling グループの起動テンプレートを作成します。テンプレートに PSAPPSRV_TEMPLATE.　という名前を付けます。テンプレートで、APP_TEMPLATE　インスタンス用に作成した AMI を選択します。手順については、「Amazon EC2 ドキュメント」を参照してください。
	• 起動テンプレートで、要件に基づいたインスタンスタイプを選択します。
	• 「詳細情報」セクションの「ユーザーデータ」フィールドに、次のエントリを追加します。パスとユーザー情報が正しいことを確認してください。前のステップで appstart.sh　スクリプトを作成しました。
	`#! /bin/ksh`
	`su -c “/usr/homes/hcmdemo/appstart.sh” - hcmdemo`
		クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル

プロセス スケジューラサーバーの Auto Scaling グループの起動テンプレートを作成します。	前のステップを繰り返して、プロセス スケジューラサーバーの Auto Scaling グループの起動テンプレートを作成します。テンプレートに PSPRCS_TEMPLATE という名前を付けます。テンプレートで、プロセススケジューラ用に作成した AMI を選択します。
	• 「詳細情報」セクションの「ユーザーデータ」フィールドに、次のエントリを追加します。パスとユーザー情報が正しいことを確認してください。前のステップで prcsstart.sh スクリプトを作成しました。
	`#! /bin/ksh`
	`su -c “/usr/hom`
	`es/hcmdemo/prcssta`
	`rt.sh” - hcmdemo`
		クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル

Elasticsearch サーバーの Auto Scaling グループの起動テンプレートを作成します。	前のステップを繰り返して、Elasticsearch サーバーの Auto Scaling グループの起動テンプレートを作成します。テンプレートに SRCH_TEMPLATE という名前を付けます。テンプレートで、検索サーバー用に作成した AMI を選択します。
	• 「詳細情報」セクションの「ユーザーデータ」フィールドに、次のエントリを追加します。パスとユーザー情報が正しいことを確認してください。前のステップで srchstart.sh スクリプトを作成しました。
	`#! /bin/ksh`
	`su -c “/usr/homes/essearch/srchstart.sh” - essearch`
		クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル

ウェブサーバーの Auto Scaling グループの起動テンプレートを作成します。	前のステップを繰り返して、ウェブサーバーの Auto Scaling グループの起動テンプレートを作成します。テンプレートに WEB_TEMPLATE という名前を付けます。テンプレートで、ウェブサーバー用に作成した AMI を選択します。
	• 「詳細情報」セクションの「ユーザーデータ」フィールドに、次のエントリを追加します。パスとユーザー情報が正しいことを確認してください。前のステップで webstart.sh スクリプトを作成しました。
	`#! /bin/ksh`
	`su -c “/usr/homes/hcmdemo/webstart.sh” - hcmdemo`
		クラウド管理者、PeopleSoft 管理者

Auto Scaling グループを作成する

タスク	説明	必要なスキル
アプリケーションサーバー の Auto Scaling グループを作成します。	Amazon EC2 コンソールで、PSAPPSRV_TEMPLATE テンプレートを使用して、アプリケーションサーバー用の PSAPPSRV_ASG という名前の Auto Scaling グループを作	クラウド管理者、PeopleSoft 管理者

タスク	説明	必要なスキル
	成します。手順については、「Amazon EC2 ドキュメント」を参照してください。 • [インスタンス起動オプションの選択] ページで、正しい VPC を選択し、異なるアベイラビリティーゾーンから複数のサブネットを選択します。 • [詳細オプションの設定] ページでは、ロードバランサーを選択しないでください。 • [グループサイズとスケーリングポリシーの設定] ページでは、システムの設計で指定する負荷量と、スケーリングポリシーを使用するかどうかに応じて設定を選択します。任意の時点で少なくとも 1 つのインスタンスをサービストラフィックに使用できるように、必要な最小容量を少なくとも 2 に設定することをお勧めします。自動スケーリングポリシーの詳細については、「Amazon EC2 ドキュメント」を参照してください。	

タスク	説明	必要なスキル
別サーバーの Auto Scaling グループを作成します。	このエピックの前のステップを繰り返して、プロセス スケジューラーサーバー、Elasticsearch サーバー、ウェブサーバーの Auto Scaling グループを作成します。	クラウド管理者、PeopleSoft 管理者

ターゲットグループの作成と設定

タスク	説明	必要なスキル
ウェブサーバーのターゲットグループを作成します。	Amazon EC2 コンソールで、ウェブサーバーのターゲットグループを作成します。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。ポートをウェブサーバーがリッスンしているポート番号に設定します。	クラウド管理者
ヘルスチェックを設定します。	ヘルスチェックの値がビジネス要件を反映した正しい値になっていることを確認してください。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。	クラウド管理者
Elasticsearch サーバーのターゲットグループを作成します。	前のステップを繰り返して、Elasticsearch サーバー用に PSFTSRCH というターゲットグループを作成し、正しい	クラウド管理者

タスク	説明	必要なスキル
	Elasticsearch ポートを設定します。	
Auto Scaling グループにターゲットグループを追加します。	先ほど作成した PSPIA_ASG というウェブサーバーの Auto Scaling グループを開きます。[ロードバランサー] タブで [編集] を選択してから、PSFTWEB ターゲットグループを Auto Scaling グループに追加します。 Elasticsearch Auto Scaling グループ PSSRCH_ASG に対してこのステップを繰り返し、以前に作成したターゲットグループ PSFTSRCHを追加します。	クラウド管理者
セッションの維持設定を行います。	ターゲットグループ PSFTWEB で、[属性] タブを選択して、[編集] を選択した後、セッションの維持設定を行います。維持設定タイプでは、ロードバランサーが生成した Cookie を選択して、期間を 1 に設定します。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。 ターゲットグループ PSFTSRCH にも同様に、このステップを繰り返します。	クラウド管理者

Application Load Balancer を作成して設定します。

タスク	説明	必要なスキル
ウェブサーバーのロードバランサーを作成します。	PSFTLB という名前の Application Load Balancer を作成して、ウエブサーバーへのトラフィックの負荷を分散させます。詳細については、「Elastic Load Balancing ドキュメント」を参照してください。 • ロードバランサーの名前を指定します。 • [スキーム] で、[インターネット接続] を選択します。 • [ネットワークマッピング] セクションで、正しい VPC と、異なるアベイラビリティーゾーンの少なくとも 2 つのパブリックサブネットを選択します。 • [リスナーとルーティング] セクションで、ターゲットグループ PSFTWEB を選択し、正しいプロトコルとポート番号を指定します。	クラウド管理者
Elasticsearch サーバーのロードバランサーを作成します。	PSFTSCH という名前の Application Load Balancer を作成して、Elasticsearch サーバーへのトラフィックの負荷を分散させます。 • ロードバランサーの名前を指定します。	クラウド管理者

タスク	説明	必要なスキル
	• [スキーム] で [内部] を選択します。 • [ネットワークマッピング] セクションで、正しい VPC とプライベートサブネットを選択します。 • [リスナーとルーティング] セクションで、ターゲットグループ PSFTSRCH を選択し、正しいプロトコルとポート番号を指定します。	
Route 53 を設定します。	Amazon Route 53 コンソールで、PeopleSoft アプリケーションにサービスを提供するホストゾーンにレコードを作成します。手順については、「Amazon Route 53ドキュメント」を参照してください。これにより、すべてのトラフィックがロードバランサーを通過するようになります。PSFTLB	クラウド管理者

関連リソース

- Oracle PeopleSoftのウェブサイト
- AWS ドキュメント

AWS Elastic Disaster Recoveryで Oracle JD Edwards EnterpriseOne のディザスタリカバリをセットアップする

作成者: Thanigaivel Thirumalai (AWS)

概要

自然災害、アプリケーション障害、またはサービスの中断による災害は、収益に悪影響を及ぼし、企業アプリケーションのダウンタイムを引き起こします。JD Edwards EnterpriseOne Enterprise Resource Planning (ERP) システムやその他のミッションクリティカルかつビジネスクリティカルなソフトウェアを導入する企業にとって、このような事象の影響を軽減するためのディザスタリカバリ (DR) 計画が不可欠です。

このパターンは、企業が JD Edwards EnterpriseOne アプリケーションの DR オプションとして AWS Elastic Disaster Recovery を使用する方法を説明しています。また、Elastic Disaster Recovery のフェイルオーバーとフェイルバックを使用して、AWS クラウドの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされているデータベースのクロスリージョン DR 戦略を構築する手順についても概説しています。

> **Note**
>
> このパターンでは、クロスリージョン DR 実装のプライマリリージョンとセカンダリリージョンを AWS でホストする必要があります。

Oracle JD Edwards EnterpriseOne は、さまざまな業界の中規模から大規模企業を対象とした統合型 ERP ソフトウェアソリューションです。

AWS Elastic Disaster Recovery は、手頃な価格のストレージ、最小限のコンピューティング、ポイントインタイムリカバリを使用して、オンプレミスおよびクラウドベースのアプリケーションを迅速かつ確実にリカバリすることで、ダウンタイムとデータ損失を最小限に抑えます。

AWS には 4 つのコアとなる DR アーキテクチャパターンがあります。このドキュメントでは、パイロットライト戦略を使用したセットアップ、構成、最適化に焦点を当てています。この戦略は、ソースデータベースからデータを複製するためのレプリケーションサーバーを最初にプロビジョニングし、DR ドリルとリカバリの開始時にのみ実際のデータベースサーバーをプロビジョニングする、低コストの DR 環境を構築するのに役立ちます。この戦略により、DR リージョンでデータベースサーバーを維持する費用が不要になります。代わりに、レプリケーションサーバーとして機能する小規模な EC2 インスタンスの料金のみが発生します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Oracle Database または Microsoft SQL Server 上で実行されている JD Edwards EnterpriseOne アプリケーション。サポートされているデータベースは、マネージド EC2 インスタンスで実行中の状態になっている必要があります。このアプリケーションには、1 つの AWS リージョンにインストールされているすべての JD Edwards EnterpriseOne 基本コンポーネント (エンタープライズサーバー、HTML サーバー、データベースサーバー) が含まれている必要があります。
- Elastic Disaster Recovery サービスをセットアップするための AWS Identity and Access Management (IAM) ロール。
- 必要な接続構成に従って構成されたElastic Disaster Recovery を実行するためのネットワーク。

制約事項

- データベースが Amazon Relational Database Service (Amazon RDS) でホストされている場合を除き、このパターンを使用してすべての層を複製できます。その場合は、Amazon RDS のクロスリージョンコピー機能を使用することをお勧めします。
- Elastic Disaster Recovery は CloudEndure Disaster Recovery と互換性がありませんが、CloudEndure Disaster Recovery からアップグレードできます。詳細については、「Elastic Disaster Recovery ドキュメント」の「よくある質問」を参照してください。
- Amazon Elastic Block Store (Amazon EBS) では、スナップショットを作成できるレートに制限があります。Elastic Disaster Recovery を使用すると、1つのAWS アカウントに最大300台のサーバーを複製できます。より多くのサーバーを複製するには、複数の AWS アカウントまたは複数のターゲット AWS リージョンを使用できます。(Elastic Disaster Recovery はアカウントとリージョンごとに個別に設定する必要があります)。詳細については、「Elastic Disaster Recovery ドキュメント」の「ベストプラクティス」を参照してください。
- ソースワークロード (JD Edwards EnterpriseOne アプリケーションとデータベース) は EC2 インスタンスでホストされている必要があります。このパターンは、オンプレミスや他のクラウド環境にあるワークロードをサポートしていません。
- このパターンは JD Edwards EnterpriseOne コンポーネントに焦点を当てています。完全な DR と事業継続計画 (BCP) には、次のような他のコアサービスも含める必要があります。
  - ネットワーク (仮想化プライベートクラウド、サブネット、セキュリティグループ)
  - アクティブディレクトリ

- Amazon WorkSpaces
- エラスティックロードバランシング
- Amazon Relational Database Service (Amazon RDS) などのマネージド型データベースサービス

前提条件、構成、制限に関する追加情報については、「ElasticDisaster Recoveryドキュメント」を参照してください。

製品バージョン

- Oracle JD Edwards EnterpriseOne (Oracleの最小技術要件に基づく、Oracle と SQL Server がサポートするバージョン)

アーキテクチャ

ターゲットテクノロジースタック

- 本番用と非本番用の単一リージョンの単一仮想プライベートクラウド (VPC)、およびDR用の2つ目のリージョン
- サーバー間のレイテンシーを低く抑えるための単一のアベイラビリティーゾーン
- ネットワークトラフィックを分散して、複数のアベイラビリティーゾーンにわたるアプリケーションのスケーラビリティと可用性を向上させる Application Load Balancer
- ドメインネームシステム (DNS) 構成を提供する Amazon Route 53
- Amazon WorkSpaces は、ユーザーにクラウドによるデスクトップエクスペリエンスを提供します
- バックアップ、ファイル、オブジェクトを保存するための Amazon Simple Storage Service (Amazon S3)
- Amazon CloudWatch を使用したアプリケーションのロギング、モニタリング、およびアラーム
- ディザスタリカバリのための Amazon Elastic Disaster Recovery

ターゲットアーキテクチャ

次の図は、Elastic Disaster Recoveryを使用した JD Edwards EnterpriseOne のクロスリージョンディザスタリカバリアーキテクチャを示しています。

手順

ここでは、プロセスの概要を示します。詳細については、エピックセクションを参照ください。

- Elastic Disaster Recovery のレプリケーションは、初回同期から開始します。初回同期中に、AWS Replication Agent はソースディスクのすべてのデータをステージングエリアサブネット内の適切なリソースに複製します。
- 連続レプリケーションは、初回同期が完了した後も無期限に継続されます。
- エージェントをインストールしてレプリケーションを開始したら、サービス固有の構成と Amazon EC2 起動テンプレートを含む起動パラメータを確認します。ソースサーバーがリカバリ準備完了と表示されたら、インスタンスを起動できます。
- Elastic Disaster Recovery が起動操作を開始するために一連の API 呼び出しを発行すると、リカバリインスタンスが起動設定に従ってすぐに AWS で起動されます。このサービスはスタートアップ時に自動的に変換サーバーをスピンアップします。
- 変換が完了して使用できる状態になると、新しいインスタンスが AWS でスピンアップされます。起動時のソースサーバーの状態は、起動したインスタンスに関連付けられたボリュームによって表されます。変換プロセスでは、インスタンスが AWS でネイティブに起動するように、ドライバ、ネットワーク、オペレーティングシステムのライセンスを変更します。
- 起動後、新しく作成されたボリュームはソースサーバーと同期されなくなります。AWS Replication Agent は、引き続きソースサーバーへの変更をステージングエリアボリュームに定期的に複製しますが、起動されたインスタンスにはそれらの変更は反映されません。
- 新しいドリルインスタンスまたはリカバリインスタンスを開始すると、データは常にソースサーバーからステージングエリアのサブネットに複製された最新の状態に反映されます。
- ソースサーバーがリカバリ準備完了と表示されたら、インスタンスを起動できます。

> **Note**
>
> このプロセスは、プライマリ AWS リージョンから DR リージョンへのフェイルオーバーと、復旧時にプライマリサイトにフェイルバックするという両方の方法で機能します。完全にオーケストレーションされた方法で、ターゲットマシンからソースマシンへのデータレプリケーションの方向を逆転させることで、フェイルバックに備えることができます。

このパターンで説明されているプロセスの利点には次のようなものがあります。

- 柔軟性: レプリケーションサーバーは、データセットとレプリケーション時間に基づいてスケールアウトとスケールインを行うため、ソースワークロードやレプリケーションを中断することなく DR テストを実行できます。

- 信頼性: レプリケーションは堅牢で、無停止で、継続的です。
- 自動化: このソリューションでは、テスト、リカバリ、フェイルバックのための統一された自動化プロセスを実現します。
- コストの最適化: 必要なボリュームだけを複製して料金を支払い、DR サイトのコンピュートリソースの料金が発生するのは、それらのリソースが有効化された場合のみです。コスト最適化レプリケーションインスタンス (コンピューティング最適化インスタンスタイプの使用を推奨) は、複数のソース、または大きな EBS ボリュームを持つ単一のソースに使用できます。

自動化とスケール

大規模にディザスタリカバリを実行すると、JD Edwards EnterpriseOne サーバは環境内の他のサーバに依存することになります。以下に例を示します。

- 起動時に JD Edwards EnterpriseOne がサポートするデータベースに接続する JD Edwards EnterpriseOne アプリケーションサーバーは、そのデータベースに依存しています。
- 認証が必要で、起動時にドメインコントローラーに接続してサービスを開始する必要がある JD Edwards EnterpriseOne サーバーは、ドメインコントローラーに依存しています。

この理由から、フェイルオーバータスクを自動化することをお勧めします。たとえば、AWS Lambda または AWS Step Functions を使用して JD Edwards EnterpriseOne のスタートアップスクリプトとロードバランサーの変更を自動化することで、エンドツーエンドのフェイルオーバープロセスを自動化できます。詳細については、ブログ記事「Elastic Disaster Recovery によるスケーラブルなディザスタリカバリ計画の作成」を参照してください。

ツール

サービス

- Amazon Elastic Block Store (Amazon EBS) は、EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Elastic Disaster Recovery は、手頃な価格のストレージ、最小限のコンピューティング、ポイントインタイムリカバリを使用して、オンプレミスおよびクラウドベースのアプリケーションを迅速かつ確実にリカバリすることで、ダウンタイムとデータ損失を最小限に抑えます。

- Amazon Virtual Private Cloud (Amazon VPC) では、リソースの配置、接続、セキュリティなど、仮想化ネットワーク環境を完全に制御できます。

ベストプラクティス

一般的なベストプラクティス

- 実際にリカバリイベントが発生した場合にどうするかについて、書面による計画を立ててください。
- Elastic Disaster Recoveryを正しく構成した後、必要に応じてオンデマンドで構成を作成できる AWS CloudFormation テンプレートを作成します。サーバーとアプリケーションを起動する順序を決定し、リカバリ計画に記録します。
- 定期的にドリルを実施してください (Amazon EC2 の標準料金が適用されます)。
- Elastic Disaster Recovery コンソールまたはプログラムを使用して、進行中のレプリケーションの状態をモニタリングします。
- インスタンスを終了する前に、ポイントインタイムスナップショットを保護して確認してください。
- AWS Replication Agent をインストールするための IAM ロールを作成します。
- 実際の DR シナリオでリカバリインスタンスの終了保護を有効にします。
- 実際にリカバリイベントが発生した場合でも、リカバリインスタンスを起動したサーバーに対して Elastic Disaster Recovery コンソールの [との接続解除] アクションを使用しないでください。接続解除を実行すると、ポイントインタイム (PIT) リカバリポイントを含む、これらのソースサーバーに関連するすべてのレプリケーションリソースが終了します。
- PIT ポリシーを変更して、スナップショットの保存日数を変更します。
- Elastic Disaster Recovery 起動設定の起動テンプレートを編集して、ターゲットサーバーの正しいサブネット、セキュリティグループ、インスタンスタイプを設定します。
- Lambda または Step 関数を使用して JD Edwards EnterpriseOne のスタートアップスクリプトとロードバランサーの変更を自動化することで、エンドツーエンドのフェイルオーバープロセスを自動化します。

JD Edwards EnterpriseOne の最適化と考慮事項

- PrintQueue をデータベースに移動します。
- MediaObjects をデータベースに移動します。

- ログと temp フォルダーをバッチサーバーとロジックサーバーから除外します。
- Oracle WebLogic から temp フォルダを除外します。
- フェイルオーバー後のスタートアップスクリプトを作成します。
- SQL Server 用の tempdb を除外します。
- Oracle 用の temp ファイルを除外します。

エピック

初期タスクと構成を行う

タスク	説明	必要なスキル
レプリケーションネットワークをセットアップする。	JD Edwards EnterpriseOne システムをプライマリ AWS リージョンに実装し、DR 用の AWS リージョンを特定します。「Elastic Disaster Recovery ドキュメント」の「レプリケーションネットワーク要件」セクションの手順に従って、レプリケーションと DR ネットワークの計画と設定を行います。	AWS 管理者
RPO と RTO を決定します。	アプリケーションサーバーとデータベースの目標復旧時間 (RTO) と目標復旧時点 (RPO) を特定します。	クラウドアーキテクト、DR アーキテクト
Amazon EFS のレプリケーションを有効にします。	該当する場合は、AWS DataSync、rsync、またはその他の適切なツールを使用して、Amazon Elastic File System (Amazon EFS) などの共有ファイルシステムの AWS プライマリから DR リージョ	クラウド管理者

タスク	説明	必要なスキル
	ンへのレプリケーションを有効にします。	
DR が発生した場合は DNS を管理する。	DRドリルまたは実際のDR中にドメインネームシステム (DNS) を更新するプロセスを特定します。	クラウド管理者
設定用の IAM ロールを作成します。	「Elastic Disaster Recovery ドキュメント」の「Elastic Disaster Recovery の初期化と権限」セクションの指示に従って、AWS サービスを初期化および管理するための IAM ロールを作成します。	クラウド管理者
VPC ピアリングのセットアップ	ソース VPC とターゲット VPC がピアリングされ、相互にアクセス可能であることを確認します。構成手順については、Amazon VPC のドキュメントを参照してください。	AWS 管理者

Elastic Disaster Recovery レプリケーション設定の構成

タスク	説明	必要なスキル
Elastic Disaster Recovery を初期化します。	Elastic Disaster Recovery コンソールを開き、ターゲット AWS リージョン (データをレプリケートしてリカバリインスタンスを起動する場所) を選択してから、[デフォルトのレプリケーション設定の設定] を選択します。	AWS 管理者

タスク	説明	必要なスキル
レプリケーションサーバーをセットアップします。	1. レプリケーションサーバーのセットアップペインで、ステージングエリアのサブネットとレプリケーションサーバーのインスタンスタイプを入力します。デフォルトでは、`t3.small` インスタンスタイプが選択されます。この構成は要件に基づいて行い、インスタンスの価格を必ず考慮してください。詳細については「アマゾン EC2 料金」を参照してください。 2. [サービスアクセス] セクションで[詳細を表示] を選択し、サービスの初期化中に作成されたサービスにリンクされたロールと追加のポリシーを確認します。 3. [Next (次へ)] を選択します。	AWS 管理者

タスク	説明	必要なスキル
ボリュームとセキュリティグループを構成します。	1. ボリュームとセキュリティグループペインで、レプリケーションサーバーの EBS ボリュームタイプを選択し、Amazon EBS 暗号化を [デフォルト] に設定します。 2. [常に Elastic Disaster Recovery セキュリティグループを使用する] を選択すると、Elastic Disaster Recovery はデフォルトのセキュリティグループを自動的にアタッチしてモニタリングします。 3. [Next (次へ)] を選択します。	AWS 管理者

タスク	説明	必要なスキル
その他のブローカーを構成する	1. 追加構成ペインで、データルーティングとスロットリング、PIT ポリシー、タグを構成します。 • データルーティングとスロットリングは、外部サーバーからレプリケーションサーバーへのデータの流れを制御します。[データレプリケーションにプライベート IP を使用] を選択します。そうしないと、レプリケーションサーバーには自動的にパブリック IP が割り当てられ、データがパブリックインターネット上に流出します。 • [ポイントインタイム (PIT) ポリシー] セクションで、スナップショットが不要になるまでの期間を決定する保持ポリシーを構成します。デフォルトの保持期間は 7 日間です。 • [タグ] セクションで、AWS アカウントで Elastic Disaster Recovery により作成されたリソースにカスタムタグを追加します。 2. [次へ] を選択し、次のペインで設定を確認し、[デフォ	AWS 管理者

タスク	説明	必要なスキル
	ルトの作成] を選択してデフォルトテンプレートを作成します。	

AWS レプリケーションエージェントをインストールする

タスク	説明	必要なスキル
IAM ロールを作成します。	`AWSElasticDisasterRecoveryAgentInstallationPolicy` ポリシーを含む IAM ロールを作成します。[Select AWS access type] セクションで、[Programmatic Access] を選択します。アクセスキー ID とシークレットアクセスキーを書き留めます。この情報は、AWS Replication Agentのインストール時に必要になります。	AWS 管理者
要件を確認します。	「Elastic Disaster Recovery ドキュメント」に記載されている、AWS Replication Agent をインストールするための前提条件を確認して完了してください。	AWS 管理者
AWS レプリケーションエージェントをインストールします。	オペレーティングシステムのインストール手順に従い、AWS Replication Agent をインストールします。 • Microsoft Windows の場合: セットアップファイルをダ	AWS 管理者

タスク	説明	必要なスキル

	ウンロードし、.exe ファイルを管理者として実行します。 プロンプトに応答しインストールを完了します。
	• Linux の場合: 次のコマンドを (表示されている順序で) コピーし、Secure Shell (SSH)に貼り付けます。最初のコマンドでインストーラーをダウンロードし、2 番目のコマンドでインストーラーを実行します。
	`wget -O ./aws-replication-installer-init.py https://aws-elastic-disaster-recovery-us-west-2.s3.amazonaws.com/latest/linux/aws-replication-installer-init.py`
	`sudo python3 aws-replication-installer-init.py`
	プロンプトに応答しインストールを完了します。
	Note
	リージョンを反映するように URL を変更します。	

タスク	説明	必要なスキル
	残りのサーバーについても同様のステップを繰り返します。	
レプリケーションをモニタリングします。	Elastic Disaster Recovery のソースサーバーペインに戻り、レプリケーションの状態をモニタリングします。データ転送のサイズによっては、初回同期に時間がかかります。 ソースサーバーが完全に同期されると、サーバーの状態は [準備完了] に更新されます。つまり、ステージングエリアにレプリケーションサーバーが作成され、EBS ボリュームがソースサーバーからステージングエリアにレプリケートされたことを意味します。	AWS 管理者

起動設定の構成

タスク	説明	必要なスキル
起動設定を編集します。	ドリルインスタンスとリカバリインスタンスの起動設定を更新するには、Elastic Disaster Recovery コンソールでソースサーバーを選択し、[アクション]、[起動設定の編集] を選択します。または、ソースサーバーページでソースマシンを選択し、次	AWS 管理者

タスク	説明	必要なスキル
	に [起動設定] タブを選択することもできます。このタブには、[一般起動設定] と [EC2 起動テンプレート] の 2 つのセクションがあります。	

タスク	説明	必要なスキル
一般起動構成を行います。	要件に応じて、一般起動設定を修正します。 • インスタンスタイプの適切なサイジング: [ベーシック] を選択すると、Elastic Disaster Recovery は Amazon EC2 起動テンプレートで選択したインスタンスタイプを迂回し、ソースサーバーのオペレーティングシステム、CPU、RAM に基づいてインスタンスタイプを自動的に選択します。 • プライベート IP のコピー: Elastic Disaster Recovery で、ドリルインスタンスまたはリカバリーインスタンスが使用するプライベート IP が、ソースサーバーで使用されるプライベート IP と一致するようにするかどうかを選択します。[はい] を選択した場合は、Amazon EC2 起動テンプレートで設定したサブネットの IP 範囲にプライベート IP アドレスが含まれていることを確認してください。 詳細については、「Elastic Disaster Recovery ドキュメン	AWS 管理者

タスク	説明	必要なスキル
	ト」の「一般起動設定」を参照してください。	
Amazon EC2 起動テンプレートを構成します。	Elastic Disaster Recovery は、Amazon EC2 起動テンプレートを使用して、各ソースサーバーのドリルインスタンスとリカバリインスタンスを起動します。起動テンプレートは、AWS Replication Agent のインストール後、Elastic Disaster Recovery に追加するソースサーバーごとに自動的に作成されます。 Elastic Disaster Recovery で使用する場合は、Amazon EC2 起動テンプレートをデフォルトの起動テンプレートとして設定する必要があります。 詳細については、「Elastic Disaster Recovery ドキュメント」の「EC2 起動テンプレート」を参照してください。	AWS 管理者

DR ドリルとフェイルオーバーの開始

タスク	説明	必要なスキル
ドリルを開始する	1. Elastic Disaster Recovery コンソールで、ソースサーバーページを開き、ソースサーバーのステータスが	AWS 管理者

タスク	説明	必要なスキル
	[準備完了] になっていることを確認します。 2. DR ドリルを実行したいソースサーバーをすべて選択します。 3. [リカバリジョブの開始] メニューから [ドリルの開始] を選択し、適切なポイントインタイムスナップショットを選択します。これにより、選択したソースサーバーのリカバリジョブが開始されます。[リカバリジョブ履歴] タブでジョブのステータスをモニタリングできます。 起動したドリルインスタンスは、リカバリインスタンスページにも表示されます。 **Note** ソースサーバーへのさらなる変更は、ドリルインスタンスではなくレプリケーションサーバーに同期されます。 4. DR ドリルインスタンスをテストして検証します。 5. リカバリインスタンスページでドリルインスタンス	

タスク	説明	必要なスキル
	を選択し、[アクション]、[との接続解除] を選択します。これにより、AWS Replication Agent がリカバリインスタンスから削除され、リカバリインスタンスに関連するすべてのリソースが Elastic Disaster Recovery から削除されます。 6. [リカバリインスタンスを削除] を選択します。これにより、Elastic Disaster Recovery コンソールからインスタンスの表現が削除され、インスタンスと Elastic Disaster Recovery サービスとの関連付けが完全に解除されます。基盤となる EC2 インスタンスは削除されません。 7. Amazon EC2 コンソールから DR ドリルインスタンスを終了します。 詳細については、「Elastic Disaster Recovery ドキュメント」の「フェイルオーバーの準備」を参照してください。	

タスク	説明	必要なスキル
ドリルを検証します。	前のステップでは、DR リージョンで新しいターゲットインスタンスを起動しました。ターゲットインスタンスは、起動を開始したときに作成されたスナップショットに基づくソースサーバーのレプリカです。 この手順では、Amazon EC2 ターゲットマシンに接続して、想定どおりに動作していることを確認します。 1. Amazon EC2 コンソールを開きます。 2. [インスタンス (実行中)] を選択します。 3. ターゲットインスタンスを選択し、プライベート IPv4 アドレスを書き留めます。 4. EC2 インスタンスに接続できることと、JD Edwards EnterpriseOne と関連コンポーネントが期待どおりに複製されていることを確認してください。	

タスク	説明	必要なスキル
フェイルオーバーを開始します。	フェイルオーバーとは、プライマリシステムからセカンダリシステムにトラフィックをリダイレクトすることです。Elastic Disaster Recovery は、AWS でリカバリインスタンスを起動することでフェイルオーバーを実行するのに役立ちます。リカバリインスタンスが起動したら、プライマリシステムからのトラフィックをこれらのインスタンスにリダイレクトします。 1. Elastic Disaster Recovery コンソールで [ソースサーバー]ページを開き、ソースサーバーの [リカバリ準備完了] 列が [準備完了] で、[データレプリケーションステータス] 列が [正常] になっていることを確認します。 2. ソースサーバーを選択します。[リカバリジョブの開始] メニューから、[リカバリを開始] を選択します。 3. リカバリインスタンスを起動するポイントインタイムスナップショットを選択し、[リカバリの開始] を選択します。 これにより、リカバリジョブが開始されます。リカバ	AWS 管理者

タスク	説明	必要なスキル
	リーインスタンスページでジョブのステータスをモニタリングできます。 4. リカバリインスタンスをテストして検証します。必要に応じて DNS 構成を調整し、JD Edwards EnterpriseOne アプリケーションをデータベースに接続します。 5. 変更はすべて新しいリカバリインスタンスに書き込まれているので、ソース JD Edwards EnterpriseOne サーバーを接続解除して使用停止できます。 6. 「 Replication Agent のインストール」エピックで説明されているプロセスに従って、リカバリインスタンスを DR リージョンのソースサーバーとして登録します。 詳細については、「Elastic Disaster Recovery ドキュメント」の「フェイルオーバーの実行」を参照してください。	

タスク	説明	必要なスキル
フェイルバックを開始します。	フェイルバックを開始するプロセスは、フェイルオーバーを開始するプロセスと似ています。 1. プライマリリージョンで Elastic Disaster Recovery コンソールを開きます。リカバリインスタンスページに移動し、ドリルインスタンスを選択して、[アクション]、[との接続解除]、[リカバリインスタンスの削除] を選択します。 2. DR リージョンで Elastic Disaster Recovery コンソールを開きます。AWS Replication Agent をインストールして、新しい JD Edwards EnterpriseOne サーバーを DR リージョンのソースサーバーとして登録します。データは、新しいステージングサブネットにプロビジョニングされた新しいレプリケーションサーバーと同期されます。 **Note** 新しい JD Edwards EnterpriseOne サーバーがソースサーバーとして登録されると、Elastic	AWS 管理者

タスク	説明	必要なスキル
	Disaster Recovery コンソールに 2 つのソースサーバーが表示される場合があります。1 つはプライマリ EC2 インスタンスから作成されたサーバーで、もう 1 つは復旧インスタンスから作成された新しいサーバーです。混乱を避けるためにサーバーには正しくタグ付けし、できれば新しいサーバーを起動テンプレートに追加することをお勧めします。 3. プライマリリージョンから DR レプリケーションを再起動するには、起動したリカバリインスタンスと DR リージョンの Elastic Disaster Recovery コンソールの関連付けを解除し、ホストをプライマリリージョンのソースサーバーとして登録します。 詳細については、「Elastic Disaster Recovery ドキュメント」の「フェイルバックの実行」を参照してください。	

タスク	説明	必要なスキル
JD Edwards EnterpriseOne コンポーネントを起動する。	1. データベースサーバーにログインして JD Edwards EnterpriseOne データベースを起動します。 2. データベースが稼働しているときに、JD Edwards EnterpriseOne ロジックサーバーとバッチサーバーを起動します。 3. ウェブサーバーで WebLogic を起動し、JAS サーバーで JAS インスタンスを起動します。 4. プロビジョニングサーバーと SM コンソールのサーバーで WebLogic を起動します。 5. これらのサーバーで SM Agent を起動します。 6. JD Edwards EnterpriseOne へのログインが正しく機能することを確認します。 JD Edwards EnterpriseOne リンクが機能するためには、Route 53 とApplication Load Balancer の変更を組み込む必要があります。 これらのステップは、Lambda、Step Functions、およびSystems Manager (Run	JD Edwards EnterpriseOne CNC

タスク	説明	必要なスキル
	Command) を使用して自動化できます。 **Note** Elastic Disaster Recovery は、オペレーティングシステムとファイルシステムをホストするソース EC2 インスタンス EBS ボリュームのブロックレベルのレプリケーションを実行します。Amazon EFS を使用して作成された共有ファイルシステムは、このレプリケーションには含まれません。最初のエピックで説明したように、AWS DataSync を使用して共有ファイルシステムを DR リージョンに複製し、その複製されたファイルシステムを DR システムにマウントできます。	

トラブルシューティング

問題	ソリューション
ソースサーバーのデータレプリケーションステータスが [停止] で、複製が遅延している。詳細を確認すると、データレプリケーションステータスには[エージェントが見つかりません]と表示される。	停止したソースサーバーが稼働中であることを確認してください。 **Note** ソースサーバーがダウンすると、レプリケーションサーバーは自動的に終了します。 遅延の問題については、「Elastic Disaster Recovery ドキュメント」の「レプリケーション遅延の問題」を参照してください。
RHEL 8.2 でソース EC2 インスタンスに AWS Replication Agent をインストールしようとしたら、ディスクのスキャン後に失敗した。`aws_replication_agent_installer.log` を確認すると、カーネルヘッダーが見当たらないことがわかった。	RHEL 8、CentOS 8、または Oracle Linux 8 に AWS Replication Agent をインストールする前に、以下を実行してください。 `sudo yum install elfutils-libelf-devel` 詳細については、「Elastic Disaster Recovery ドキュメント」の「Linux のインストール要件」を参照してください。
Elastic Disaster Recovery コンソールで、ソースサーバーが [準備完了] となって遅延し、データレプリケーションのステータスが [停止中] と表示される。 AWS Replication Agent が使用できない時間によっては、ステータスに大きな遅延と表示される場合があるが、問題は変わらない。	オペレーティングシステムコマンドを使用して、AWS Replication Agent がソース EC2 インスタンスで実行されていること、またはインスタンスが実行中であることを確認します。 問題を修正すると、Elastic Disaster Recovery はスキャンを再開します。すべてのデータが同期され、レプリケーションステータスが [正常]

問題	ソリューション
	になるまで待ってから DR ドリルを開始してください。
初回のレプリケーションで大きい遅延が発生する。Elastic Disaster Recovery コンソールを見ると、ソースサーバーの初回同期ステータスが非常に遅い。	「Elastic Disaster Recovery ドキュメント」の「レプリケーション遅延の問題」セクションに記載されているレプリケーション遅延の問題を確認してください。 レプリケーションサーバーは、組み込み関数が原因で負荷を処理できない場合があります。その場合は、AWS テクニカルサポートチームに相談した上でインスタンスタイプをアップグレードしてみてください。

関連リソース

- Elastic Disaster Recoveryユーザーガイド
- Elastic Disaster Recovery によるスケーラブルなディザスタリカバリ計画の作成 (AWS ブログ記事)
- Elastic Disaster Recovery - 技術入門 (AWS スキルビルダーコース、ログインが必要)
- Elastic Disaster Recovery クイックスタートガイド

マルチリージョン、マルチアカウント組織で AWS CloudFormation ドリフト検出を設定する

作成者: Ram Kandaswamy (AWS)

概要

アマゾン ウェブ サービス (AWS) のユーザーは、 AWS CloudFormation スタックのドリフトなど、リソース設定の不一致を効率的に検出し、できるだけ早く修正する方法を探します。これは、 AWS Control Tower が使用される場合に特に当てはまります。

このパターンは、統合されたリソース設定を変更し、結果に基づいて、問題を効率的に解決する規範的なソリューションを提供します。このソリューションは、複数の 、複数の アカウント AWS リージョン、またはその両方の組み合わせに複数の AWS CloudFormation スタックが作成されるシナリオ向けに設計されています。このソリューションの目標は以下のとおりです。

- ドリフト検出プロセスを簡素化する
- 通知と警告を設定する
- 統合レポートを設定する

前提条件と制限

前提条件

- AWS Config モニタリングする必要があるすべてのリージョンとアカウントで有効になっている

機能制限

- 生成されたレポートは、カンマ区切り値 (CSV) と JSON 出力形式のみをサポートしています。

アーキテクチャ

次の図は、複数の accounts. AWS Config rules で AWS Organizations 設定されたアカウント間の通信を示しています。

ワークフローには以下のステップが含まれます。

1. AWS Config ルールはドリフトを検出します。

2. 他のアカウントで見つかったドリフト検出結果は、管理アカウントに送信されます。
3. Amazon CloudWatch ルールは AWS Lambda 関数を呼び出します。
4. Lambda 関数は、集計結果について AWS Config ルールをクエリします。
5. Lambda 関数は Amazon Simple Notification Service (Amazon SNS) に通知し、ドリフトの E メール通知を送信します。

自動化とスケール

ここで紹介するソリューションは、追加のリージョンとアカウントの両方に合わせてスケールできます。

ツール

AWS のサービス

- AWS Config は、 内の AWS リソースの設定の詳細ビューを提供します AWS アカウント。これには、リソース間の関係と設定の履歴が含まれるため、時間の経過と共に設定と関係がどのように変わるかを確認できます。
- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

エピック

のドリフト検出を自動化する AWS CloudFormation

タスク	説明	必要なスキル
アグリゲータを作成します。	1. にサインイン AWS Management Console し、 AWS Config コンソールを https://console.aw	クラウドアーキテクト

タスク	説明	必要なスキル
	s.amazon.com/config:// www.com で開きます。 2. 管理アカウントにアグリゲータを作成します。 3. がソースアカウントからデータを取得できるようにAWS Config 、データレプリケーションが有効になっていることを確認します。 4. 該当するすべてのリージョンとアカウントを選択します。に基づいてアカウントを選択できます AWS Organizations。組織内の新しいアカウントは自動的にアグリゲータの一部になるため、このアプローチをお勧めします。	
AWS マネージドルールを作成します。	`cloudformation-stack-drift-detection-check` AWSマネージドルールを追加します。このルールには `cloudformationArn` のパラメータ値が必要です。 スタックのドリフトを検出する権限を持つ IAM ロールの Amazon リソースネーム (ARN) です。ロールには、がロールを引き受け AWS Config ることができる信頼ポリシーが必要です。	クラウドアーキテクト

タスク	説明	必要なスキル
アグリゲータの詳細なクエリセクションを作成します。	以下のクエリを作成して、複数のソースからドリフトスタックを取得できます。 `SELECT resourceId, configuration.driftInformation.stackDriftStatus WHERE resourceType = 'AWS::CloudFormation::Stack' AND configuration.driftInformation.stackDriftStatus IN ('DRIFTED')`	クラウドアーキテクト、開発者
クエリとパブリッシュを自動化します。	1. アタッチされているコードを使用して Lambda 関数を作成します。Lambda は、Lambda 関数の環境変数として提供される SNS トピックに結果を発行します。 2. アラートを受信するには、SNS トピックへの E メールサブスクリプションを作成します。	クラウドアーキテクト、開発者
CloudWatch ルールを作成します。	スケジュールベースの CloudWatch ルールを作成して、アラートを行う Lambda 関数を呼び出します。	クラウドアーキテクト

関連リソース

リソース

- とは AWS Config
- 「マルチアカウント、マルチリージョンのデータ集計」
- 「スタックとリソースに対するアンマネージド型構成変更の検出」
- IAM: IAM ロールを特定の に渡す AWS のサービス
- 「Amazon SNS とは」

追加情報

考慮事項

特定の間隔で API コールを含むカスタムソリューションを使用して、各 CloudFormation スタックまたはスタックセットでドリフト検出を開始する代わりに、このパターンで示されているソリューションを使用することをお勧めします。特定の間隔で API コールを使用するカスタムソリューションは、多数の API コールを引き起こし、パフォーマンスに影響を与える可能性があります。API コールの数が多いため、スロットリングが発生する可能性があります。もう 1 つの潜在的な問題として、リソースの変更をスケジュールのみに基づいて特定すると、検出に遅延が発生します。

スタックセットはスタックで構成されているため、このソリューションを使用できます。スタックインスタンスの詳細は、ソリューションの一部としても入手できます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

S3 バケットを AWS CloudFormation スタックとして正常にインポートしました

作成者: Ram Kandaswamy (AWS)

概要

Amazon Simple Storage Service (Amazon S3) バケットなどのAmazon Web Services (AWS) リソースを使用し、コードとしての infrastructure as code (IaC) アプローチを使用したい場合は、リソースを AWS CloudFormation にインポートしてスタックとして管理できます。

このパターンは、S3 バケットを AWS CloudFormation スタックとして正常にインポートするための手順を示しています。このパターンの方法を使用すると、S3 バケットを 1 回のアクションでインポートした場合に発生する可能性のあるエラーを回避できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 既存の S3 バケットおよび S3 バケットポリシー。この詳細については、「AWS ナレッジセンターの AWS Config ルール s3-bucket-ssl-requests-only に準拠するために、どのような S3 バケットポリシーを使用すべきか」を参照してください。
- 既存の AWS Key Management Service (AWS KMS) キーとそのエイリアス。詳細については、AWS KMS ドキュメント「アラートの操作」を参照してください。
- サンプル `CloudFormation-template-S3-bucket` AWS CloudFormation テンプレート (添付)。お使いのコンピュータにダウンロードされます。

アーキテクチャ

この図表は、次のワークフローを示しています：

1. ユーザーは JSON 形式または YAML 形式の AWS CloudFormation テンプレートを作成します。
2. このテンプレートは、S3 バケットをインポートするための AWS CloudFormation スタックを作成します。
3. AWS CloudFormation スタックは、テンプレートで指定した S3 バケットを管理します。

テクノロジースタック

- AWS CloudFormation
- AWS Identity and Access Management (IAM)
- AWS KMS
- Amazon S3

ツール

- 「AWS CloudFormation」— AWS CloudFormation は、AWS インフラストラクチャのデプロイメントを予測可能かつ繰り返し作成し、プロビジョニングするのに役立ちます。
- 「AWS Identity and Access Management (IAM)」— IAM は、AWS サービスへのアクセスをセキュアに制御するためのウェブサービスです。
- 「AWS KMS」— AWS Key Management Service (AWS KMS) は、クラウド向けに拡張された暗号化およびキー管理サービスです。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。

エピック

AWS KMS keyベースの暗号化を使用して S3 バケットを AWS CloudFormation スタックとしてインポートする

タスク	説明	必要なスキル

S3 バケットと KMS キーをインポートするテンプレートを作成します。	ローカルコンピュータで、次のサンプルテンプレートを使用して S3 バケットと KMS キーをインポートするテンプレートを作成します。
	
	`AWSTemplateFormatVersion: 2010-09-09`
	
	`Parameters:`
		AWS DevOps

タスク	説明	必要なスキル

	<pre> bucketName:
	
	    Type: String
	
	Resources:
	
	  S3Bucket:
	
	    Type: 'AWS::S3:
	:Bucket'
	
	    DeletionPolicy:
	 Retain
	
	    Properties:
	
	      BucketName: !Ref
	 bucketName
	
	      BucketEncryption:
	
	        ServerSid
	eEncryptionConfigu
	ration:
	
	          - ServerSid
	eEncryptionByDefault:
	
	              SSEAlgori
	thm: 'aws:kms'
	
	              KMSMaster
	KeyID: !GetAtt
	
	                -
	 KMSS3Encryption
	
	                - Arn
	
	  KMSS3Encryption:</pre>	

タスク	説明	必要なスキル

	<pre> Type: 'AWS::KMS
	::Key'
	
	    DeletionPolicy:
	 Retain
	
	    Properties:
	
	      Enabled: true
	
	      KeyPolicy: !Sub
	 |-
	
	        {
	
	            "Id": "key-
	consolepolicy-3",
	
	            "Version":
	 "2012-10-17",
	
	            "Statemen
	t": [
	
	                {
	
	 "Sid": "Enable IAM User
	 Permissions",
	
	 "Effect": "Allow",
	
	 "Principal": {
	
	 "AWS": ["arn:aws:iam::
	${AWS::AccountId}:roo
	t"]
	
	                    },</pre>	

タスク	説明	必要なスキル

	<pre>"Action": "kms:*",
	
	"Resource": "*"
	}
	}
	]
	}
	
	EnableKey Rotation: true</pre>	

タスク	説明	必要なスキル
スタックを作成します。	1. AWS マネジメントコンソールにサインインし、AWS CloudFormation コンソールを開き、[スタックの表示]、[スタックの作成]、[既存のリソースを使用 (リソースのインポート)] の順に選択します。 2. [テンプレートファイルをアップロード] を選択し、先ほど作成したテンプレートファイルをアップロードします。 3. スタックの名前を入力し、必要に応じて残りのオプションを設定します。 4. [スタックの作成] を選択し、スタックのステータスが `IMPORT_COMPLETE` に変わるのを待ちます。	AWS DevOps

タスク	説明	必要なスキル

KMS キーエイリアスを作成します。	1. AWS CloudFormation コンソールで [スタック] を選択し、前に作成したスタックの名前を選択し、[テンプレート] ペインを選択して、[デザイナーで表示] を選択します。
	2. テンプレートの `Resource` セクションに次のスニペットを追加し、[スタックを作成] を選択してウィザードを完了します。
	
	`KMSS3EncryptionAlias:`
	
	`  Type: 'AWS::KMS ::Alias'`
	
	`  DeletionPolicy: Retain`
	
	`  Properties:`
	
	`  AliasName: alias/S3BucketKey`
	
	`  TargetKeyId: !Ref KMSS3Encryption`
	
	これに関する詳細については、AWS CloudFormation ドキュメントの「AWS CloudFormation スタックの更新」を参照してください。
		AWS DevOps

タスク	説明	必要なスキル

S3 バケットポリシーを含むようにスタックを更新する。	1. AWS CloudFormation コンソールで [スタック] を選択し、前に作成したスタックの名前を選択し、[テンプレート] ペインを選択して、[デザイナーで表示] を選択します。
	2. テンプレートの `Resource` セクションに次のスニペットを追加し、[スタックを作成] を選択してウィザードを完了します。
	
	`S3BucketPolicy:`
	
	`  Type: 'AWS::S3::BucketPolicy'`
	
	`  Properties:`
	
	`    Bucket: !Ref S3Bucket`
	
	`    PolicyDocument: !Sub |-`
	
	`      {`
	
	`  "Version": "2008-10-17",`
	
	`                  "Id": "restricthttp",`
	
	`  "Statement": [`
		AWS DevOps

タスク	説明	必要なスキル

	{
	
	"Sid": "denyhttp",
	
	"Effect": "Deny",
	
	"Principal": {
	
	"AWS": "*"
	
	},
	
	"Action": "s3:*",
	
	"Resource": ["arn:aws:s3:::${S3Bucket}","arn:aws:s3:::${S3Bucket}/*"],
	
	"Condition": {
	
	"Bool": {
	
	"aws:SecureTransport": "false"
	
	}
	
	}	

タスク	説明	必要なスキル
	}] } **Note** この S3 バケットポリシーには、安全でない API コールを制限する拒否ステートメントがあります。	

タスク	説明	必要なスキル
キーポリシーを更新します。	1. AWS CloudFormation コンソールで [スタック] を選択し、前に作成したスタックの名前を選択し、[テンプレート] ペインを選択して、[デザイナーで表示] を選択します。 2. テンプレートの KMS リソースを変更して、管理者が KMS キーを管理できるようにするキーポリシーを含めます。 3. [スタックの作成] を選択し、[次へ] を選択して、要件に従ってウィザードを完了します。 詳細については、AWS KMS ドキュメントの「 のキーポリシー AWS KMS」を参照してください。	AWS 管理者

タスク	説明	必要なスキル

リソースレベルのタグを追加します。	1. AWS CloudFormation コンソールで [スタック] を選択し、前に作成したスタックの名前を選択し、[テンプレート] ペインを選択して、[デザイナーで表示] を選択します。
	2. テンプレートの Amazon S3 リソース `Properties` セクションに次のスニペットを追加し、[スタックを作成] を選択してウィザードを完了します。
	`Tags:`
	`  - Key: createdBy`
	`    Value: Cloudformation`
		AWS DevOps

関連リソース

- 「既存のリソースを AWS CloudFormation の管理に取り込む」
- 「AWS re: Invent 2017: AWS CloudFormation のディープダイブ」(ビデオ)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS DataSync を使用して、さまざまな AWS リージョンの Amazon EFS ファイルシステム間でデータを同期する

作成者: Sarat Chandra Pothula (AWS) と Aditya Ambati (AWS)

概要

このソリューションは、さまざまな AWS リージョンの Amazon Elastic File System (Amazon EFS) インスタンス間で効率的かつ安全なデータ同期のための堅牢なフレームワークを提供します。このアプローチはスケーラブルで、制御されたクロスリージョンデータレプリケーションを提供します。このソリューションは、ディザスタリカバリとデータ冗長性戦略を強化できます。

AWS Cloud Development Kit (AWS CDK) を使用することで、このパターンは をInfrastructure as Code (IaC) アプローチとして使用してソリューションリソースをデプロイします。AWS CDK アプリケーションは、重要な AWS DataSync、Amazon EFS、Amazon Virtual Private Cloud (Amazon VPC)、および Amazon Elastic Compute Cloud (Amazon EC2) リソースをデプロイします。この IaC は、AWS のベストプラクティスと完全に一致した、繰り返し可能でバージョン管理されたデプロイプロセスを提供します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 「AWS コマンドラインインターフェイス (AWS CLI) バージョン 2.9.11 以降、インストールおよび設定済み」
- AWS CDK バージョン 2.114.1 以降、インストールおよびブートストラップ
- NodeJS バージョン 20.8.0 以降、インストール済み

制約事項

- このソリューションは、データ転送レート、サイズ制限、リージョンの可用性など、DataSync と Amazon EFS の制限を継承します。詳細については、「AWS DataSync クォータ」と「Amazon EFS クォータ」を参照してください。
- このソリューションは Amazon EFS のみをサポートしています。DataSync は、Amazon Simple Storage Service (Amazon S3) や Amazon FSx for Lustre などの他の AWS サービスをサポートしています。ただし、このソリューションでは、データを他のサービスと同期するために変更が必要です。

アーキテクチャ

このソリューションは、次の AWS CDK スタックをデプロイします。

- Amazon VPC スタック – このスタックは、プライマリ AWS リージョンとセカンダリ AWS リージョンの両方で、サブネット、インターネットゲートウェイ、NAT ゲートウェイを含む仮想プライベートクラウド (VPC) リソースを設定します。
- Amazon EFS スタック – このスタックは、Amazon EFS ファイルシステムをプライマリリージョンとセカンダリリージョンにデプロイし、それぞれの VPCs に接続します。
- Amazon EC2 スタック – このスタックは、プライマリリージョンとセカンダリリージョンで EC2 インスタンスを起動します。これらのインスタンスは、Amazon EFS ファイルシステムをマウントするように設定され、共有ストレージへのアクセスを許可します。
- DataSync ロケーションスタック – このスタックは、 というカスタムコンストラクト`DataSyncLocationConstruct`を使用して、プライマリリージョンとセカンダリリージョンに DataSync ロケーションリソースを作成します。これらのリソースは、データ同期のエンドポイントを定義します。
- DataSync タスクスタック – このスタックは、 というカスタムコンストラクト`DataSyncTaskConstruct`を使用して、プライマリリージョンに DataSync タスクを作成します。このタスクは、DataSync の送信元と送信先の場所を使用して、プライマリリージョンとセカンダリリージョン間でデータを同期するように設定されています。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- 「AWS DataSync」は、ファイルまたはオブジェクトデータを AWS ストレージサービス間で移動したり、AWS ストレージサービス間で移動したりするのに役立つオンラインデータ転送および検出サービスです。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

- Amazon Elastic File System (Amazon EFS) は、AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

コードリポジトリ

このパターンのコードは、GitHub Amazon EFS Cross-Region DataSync Project リポジトリで入手できます。

ベストプラクティス

TypeScript で AWS CDK を使用して IaC プロジェクトを作成するためのベストプラクティスで説明されているベストプラクティスに従います。

エピック

AWS CDK アプリをデプロイする

タスク	説明	必要なスキル
プロジェクトリポジトリのクローンを作成します。	次のコマンドを入力して、Amazon EFS クロスリージョン DataSync プロジェクトリポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/aws-efs-crossregion-datasync.git`	AWS DevOps
npm の依存関係をインストールします。	次のコマンドを入力します。 `npm ci`	AWS DevOps

タスク	説明	必要なスキル

プライマリリージョンとセカンダリリージョンを選択します。	クローンされたリポジトリで、 src/infa ディレクトリに移動します。Launcher.ts ファイルで、 PRIMARY_AWS_REGION と SECONDARY_AWS_REGION の値を更新します。対応するリージョンコードを使用します。
	
	`const primaryRegion = { account: account, region: '<PRIMARY_AWS_REGION>' };`
	`const secondaryRegion = { account: account, region: '<SECONDARY_AWS_REGION>' };`
		AWS DevOps

環境を起動します。	次のコマンドを入力して、使用する AWS アカウントと AWS リージョンをブートストラップします。 `cdk bootstrap <aws_account>/<aws_region>` 詳細については、AWS CDK ドキュメントの「ブートストラップ」を参照してください。	AWS DevOps

タスク	説明	必要なスキル
AWS CDK スタックを一覧表示します。	次のコマンドを入力して、アプリ内の AWS CDK スタックのリストを表示します。 `cdk ls`	AWS DevOps
AWS CDK スタックを合成します。	次のコマンドを入力して、AWS CDK アプリケーションで定義された各スタックの AWS CloudFormation テンプレートを生成します。 `cdk synth`	AWS DevOps
AWS CDK アプリをデプロイします。	次のコマンドを入力して、変更に対する手動承認を必要とせずに、すべてのスタックを AWS アカウントにデプロイします。 `cdk deploy --all --require-approval never`	AWS DevOps

デプロイを検証する

タスク	説明	必要なスキル
プライマリリージョンの EC2 インスタンスにログインします。	1. AWS Systems Manager の一機能である Session Manager を使用して、プライマリリージョンの EC2 インスタンスにログインします。手順については、AWS Systems Manager Session	AWS DevOps

タスク	説明	必要なスキル
	Manager を使用して Linux インスタンスに接続する」を参照してください。 2. ディレクトリを Amazon EFS マウントパスに変更します。 `cd /mnt/efs`	

一時ファイルを作成します。	次のコマンドを入力して、Amazon EFS マウントパスに一時ファイルを作成します。
	`sudo dd if=/dev/zero \`
	`of=tmptst.dat \`
	`bs=1G \`
	`seek=5 \`
	`count=0`
	
	`ls -lrt tmptst.dat`
		AWS DevOps

タスク	説明	必要なスキル

DataSync タスクを開始します。	次のコマンドを入力して、一時ファイルをプライマリリージョンからセカンダリリージョンにレプリケートします。ここで、 <ARN-task> は DataSync タスクの Amazon リソースネーム (ARN) です。
	`aws datasync start-task-execution \`
	`    --task-arn <ARN-task>`
	コマンドは、タスク実行の ARN を次の形式で返します。
	`arn:aws:datasync:<region>:<account-ID>:task/task-execution/<exec-ID>`
		AWS DevOps

タスク	説明	必要なスキル
データ転送のステータスを確認します。	次のコマンドを入力して DataSync 実行タスクを記述します。ここで、 <ARN-task-execution> はタスク実行の ARN です。 `aws datasync describe-task-execution \ --task-execution-arn <ARN-task-execution>` DataSync タスクは、PrepareStatus 、、および VerifyStatus のすべての の値が の場合TransferStatus に完了しますSUCCESS。	AWS DevOps
セカンダリリージョンの EC2 インスタンスにログインします。	1. AWS Systems Manager の一機能である Session Manager を使用して、セカンダリリージョンの EC2 インスタンスにログインします。手順については、AWS Systems Manager Session Manager を使用して Linux インスタンスに接続する」を参照してください。 2. ディレクトリを Amazon EFS マウントパスに変更します。 `cd /mnt/efs`	AWS DevOps

タスク	説明	必要なスキル
レプリケーションを検証します。	次のコマンドを入力して、一時ファイルが Amazon EFS ファイルシステムに存在することを確認します。 `ls -lrt tmptst.dat`	AWS DevOps

関連リソース

AWS ドキュメント

- AWS CDK API リファレンス
- Amazon EFS を使用した AWS DataSync 転送の設定
- AWS DataSync 転送に関する問題のトラブルシューティング

その他の AWS リソース

- AWS DataSync FAQs

LocalStack および Terraform Tests を使用して AWS インフラストラクチャをテストする

作成者: Ivan Girardi (AWS) と Ioannis Kalyvas (AWS)

概要

このパターンは、 AWS 環境にインフラストラクチャをプロビジョニングすることなく、Terraform AWS で のInfrastructure as Code (IaC) をローカルでテストするのに役立ちます。Terraform Tests フレームワークを LocalStack と統合します。LocalStack Docker コンテナは、さまざまな をエミュレートするローカル開発環境を提供します AWS のサービス。これにより、 でコストを発生させることなく、インフラストラクチャのデプロイをテストして反復処理できます AWS クラウド。

ソリューションは次の利点があります。

- コストの最適化 – LocalStack に対してテストを実行すると、 を使用する必要がなくなります AWS のサービス。これにより、これらの AWS リソースの作成、運用、変更に関連するコストが発生するのを防ぐことができます。
- 速度と効率 – ローカルでのテストは通常、 AWS リソースのデプロイよりも高速です。この迅速なフィードバックループにより、開発とデバッグが高速化されます。LocalStack はローカルで実行されるため、インターネット接続なしで Terraform 設定ファイルを開発およびテストできます。Terraform 設定ファイルをローカルでデバッグし、すぐにフィードバックを受け取ることができるため、開発プロセスが効率化されます。
- 一貫性と再現性 – LocalStack は、テストのための一貫した環境を提供します。この整合性により、外部 AWS の変更やネットワークの問題に関係なく、テストで同じ結果が得られるようになります。
- 分離 — LocalStack によるテストにより、ライブ AWS リソースや本番環境に誤って影響することを防ぎます。この分離により、さまざまな設定を安全に実験およびテストできます。
- 自動化 – 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインとの統合により、Terraform 設定ファイルを自動的にテストできます。パイプラインは、デプロイ前に IaC を徹底的にテストします。
- 柔軟性 – さまざまな および サービス設定をシミュレートして AWS リージョン AWS アカウント、本番環境により近いものにすることができます。

前提条件と制限

前提条件

- Docker をインストールする
- デフォルトの Docker ソケット () へのアクセスを有効にします/var/run/docker.sock。詳細については、LocalStack のドキュメントを参照してください。
- Docker Compose のインストール
- Terraform バージョン 1.6.0 以降をインストールする
- Terraform CLI のインストール
- Terraform AWS プロバイダーを設定する
- （オプション) AWS Command Line Interface () をインストールして設定しますAWS CLI。LocalStack AWS CLI で を使用する方法の例については、「LocalStack と Terraform Tests を使用する GitHub Test インフラストラクチャ」リポジトリを参照してください。 AWS LocalStack

制約事項

- このパターンでは、Amazon Simple Storage Service (Amazon S3) AWS Lambda、 AWS Step Functions、および Amazon DynamoDB リソースをテストするための明示的な例を示します。ただし、このソリューションを拡張して追加の AWS リソースを含めることができます。
- このパターンでは、Terraform テストをローカルで実行する手順を示しますが、テストを任意の CI/CD パイプラインに統合できます。
- このパターンでは、LocalStack Community イメージを使用する手順を示します。LocalStack Pro イメージを使用している場合は、LocalStack Pro のドキュメントを参照してください。
- LocalStack は、さまざまな AWS APIs。完全なリストについては、AWS 「サービス機能のカバレッジ」を参照してください。一部の高度な機能では、LocalStack Pro のサブスクリプションが必要になる場合があります。

アーキテクチャ

次の図は、このソリューションのアーキテクチャを示しています。主なコンポーネントは、ソースコードリポジトリ、CI/CD パイプライン、および LocalStack Docker コンテナです。LocalStack Docker コンテナは、以下を AWS のサービス ローカルでホストします。

- ファイルを保存するための Amazon S3 バケット
- モニタリングとログ記録のための Amazon CloudWatch
- サーバーレスコードを実行するための AWS Lambda 関数

- マルチステップワークフローをオーケストレーションするための AWS Step Functions ステートマシン
- NoSQL データを保存するための Amazon DynamoDB テーブル

この図表は、次のワークフローを示しています：

1. Terraform 設定ファイルをソースコードリポジトリに追加してコミットします。
2. CI/CD パイプラインは変更を検出し、静的 Terraform コード分析のビルドプロセスを開始します。パイプラインは LocalStack Docker コンテナを構築して実行します。その後、パイプラインはテストプロセスを開始します。
3. パイプラインは、LocalStack Docker コンテナでホストされている Amazon S3 バケットにオブジェクトをアップロードします。
4. オブジェクトをアップロードすると、 AWS Lambda 関数が呼び出されます。
5. Lambda 関数は、Amazon S3 イベント通知を CloudWatch ログに保存します。
6. Lambda 関数は AWS Step Functions ステートマシンを起動します。
7. ステートマシンは Amazon S3 オブジェクトの名前を DynamoDB テーブルに書き込みます。
8. CI/CD パイプラインのテストプロセスでは、アップロードされたオブジェクトの名前が DynamoDB テーブルのエントリと一致することを確認します。また、S3 バケットが指定された名前でデプロイされていること、および AWS Lambda 関数が正常にデプロイされたことも検証します。

ツール

AWS のサービス

- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functions はサーバーレスオーケストレーションサービスであり、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立ちます。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしてのPlatform as a Service (PaaS) 製品のセットです。
- Docker Compose は、マルチコンテナアプリケーションを定義して実行するためのツールです。
- LocalStack は、単一のコンテナで実行されるクラウドサービスエミュレータです。LocalStack を使用すると、 に接続せずに AWS のサービス、 を使用するローカルマシンでワークロードを実行できます AWS クラウド。
- Terraform は HashiCorp の IaC ツールで、クラウドおよびオンプレミスリソースの作成と管理に役立ちます。
- Terraform Tests は、統合またはユニットテストに類似したテストを通じて Terraform モジュール設定の更新を検証するのに役立ちます。

コードリポジトリ

このパターンのコードは、LocalStack と Terraform Tests リポジトリを使用した GitHub Test インフラストラクチャで使用できます。 AWS LocalStack

ベストプラクティス

- このソリューションは、Terraform 設定ファイルで指定された AWS インフラストラクチャをテストし、それらのリソースを にデプロイしません AWS クラウド。リソースをデプロイする場合は、最小特権の原則 (IAM ドキュメント) に従い、Terraform バックエンド (Terraform ドキュメント) を適切に設定します。
- LocalStack を CI/CD パイプラインに統合する場合は、LocalStack Docker コンテナを特権モードで実行しないことをお勧めします。詳細については、「ランタイム権限と Linux 機能 (Docker ドキュメント) 」および「セルフマネージドランナーのセキュリティ (GitLab ドキュメント) 」を参照してください。 GitLab

エピック

解決策をデプロイする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	bash シェルで、次のコマンドを入力します。これにより、GitHub から LocalStack と Terraform Tests リポジトリを使用してテスト AWS インフラストラクチャのクローンが作成されます。 `git clone https://github.com/aws-samples/localstack-terraform-test.git`	DevOps エンジニア

LocalStack コンテナを実行します。	1. 次のコマンドを入力して、クローンされたリポジトリに移動します。
	`cd localstack-terraform-test`
	2. 次のコマンドを入力して、LocalStack Docker コンテナをデタッチモードで起動します。
	`docker-compose up -d`
	3. LocalStack Docker コンテナが動作するまで待ちます。
		DevOps エンジニア

Terraform を初期化します。	Terraform を初期化するには、次のコマンドを入力します。	DevOps エンジニア

タスク	説明	必要なスキル
	`terraform init`	

Terraform テストを実行します。	1. Terraform Tests を実行するには、次のコマンドを入力します。
	`terraform test`
	2. すべてのテストが正常に完了したことを確認します。出力は次の例のようになります:
	`Success! 3 passed, 0 failed.`
		DevOps エンジニア

リソースをクリーンアップします。	LocalStack コンテナを破棄するには、次のコマンドを入力します。 `docker-compose down`	DevOps エンジニア

トラブルシューティング

問題	ソリューション
`Error: reading DynamoDB Table Item (Files|README.md): empty` terraform test コマンドを実行すると、 の結果が表示されます。	1. `terraform test` コマンドを再入力します。 2. それでもエラーが解決しない場合は、main.tf://www.com を編集して、スリープタイムアウトを 15 秒を超える値に増やします。 `resource "time_sleep" "wait" {`

問題	ソリューション

	create_duration = "15s"
	triggers = {
	s3_object = local.key_json
	}
	}

関連リソース

- Terraform の開始方法: AWS CDK および AWS CloudFormation エキスパート向けのガイダンス (AWS 規範ガイダンス)
- Terraform AWS プロバイダーを使用するためのベストプラクティス (AWS 規範ガイダンス)
- 新しい Terraform Test Framework AWS を使用した Terraform CI/CD と でのテスト (AWS ブログ記事)
- LocalStack Cloud Emulator を使用したソフトウェア配信の高速化 AWS Marketplace (AWS ブログ記事)

追加情報

GitHub Actions との統合

GitHub Actions を使用して、CI/CD パイプラインに LocalStack テストと Terraform テストを統合することができます。詳細については、GitHub Actions のドキュメントを参照してください。以下は、GitHub Actions 設定ファイルの例です。

```
name: LocalStack Terraform Test

on:
  push:
    branches:
      - '**'

  workflow_dispatch: {}

jobs:
  localstack-terraform-test:
    runs-on: ubuntu-latest
```

```
    steps:
    - uses: actions/checkout@v4

    - name: Build and Start LocalStack Container
      run: |
        docker compose up -d

    - name: Setup Terraform
      uses: hashicorp/setup-terraform@v3
      with:
        terraform_version: latest

    - name: Run Terraform Init and Validation
      run: |
        terraform init
        terraform validate
        terraform fmt --recursive --check
        terraform plan
        terraform show

    - name: Run Terraform Test
      run: |
        terraform test

    - name: Stop and Delete LocalStack Container
      if: always()
      run: docker compose down
```

SAP ペースメーカークラスターを ENSA1 から ENSA2 にアップグレード

作成者: Gergely Cserdi (AWS) と Balazs Sandor Skublics (AWS)

概要

このパターンでは、スタンドアロンエンキューサーバー (ENSA1) に基づく SAP Pacemaker クラスターを ENSA2 にアップグレードする場合の手順と考慮事項を説明します。このパターンの情報は、SLES (SLES)Linux Enterprise Server (SLES) と Red Hat Enterprise Linux (RHEL) オペレーティングシステムの両方に適用されます。

SAP NetWeaver 7.52 または S/4HANA 1709 およびそれ以前のバージョンの Pacemaker クラスターは、ENSA1 アーキテクチャで動作し ENSA1 専用に設定されています。Amazon Web Services (AWS) で SAP ワークロードを実行して、ENSA2 への移行を検討している場合、SAP、SUSE、RHEL のドキュメントには包括的な情報が記載されていないことに気付くかもしれません。このパターンは、ENSA1 から ENSA2 にアップグレードするために SAP パラメータと Pacemaker クラスターを再設定するために必要な技術的ステップを説明します。SUSE システムの例を示していますが、RHEL クラスターでも概念は同じです。

注: ENSA1 と ENSA2 は SAP アプリケーションのみに関係する概念なので、このパターンの情報は SAP HANA や他のタイプのクラスターには当てはまりません。

技術的には、ENSA2 はエンキューレプリケーター 2 の有無にかかわらず使用できます。ただし、高可用性 (HA) と (クラスターソリューションによる) フェイルオーバー自動化には エンキューレプリケーター 2 が必要です。このパターンでは、ENSA2 クラスターという用語は、スタンドアロンエンキューサーバー 2 とエンキューレプリケーター 2 を備えたクラスターを指します。

前提条件と制限

前提条件

- SLES または RHEL の Pacemaker と Corosync を使用する、動作中の ENSA1 ベースのクラスターです。
- 少なくとも 2 つの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスです。そこでは、(ABAP) SAP セントラルサービス (ASCS/SCS) インスタンスとエンキューレプリケーションサーバー (ERS) インスタンスが実行されています。
- SAP アプリケーションとクラスターの管理に関する知識です。

- ルートユーザーとして Linux 環境にアクセスします。

機能制限

- ENSA1 ベースのクラスターに、2つのノードのアーキテクチャのみが適用されます。
- ENSA2 ベースのクラスターが、7.52 以前のバージョンの SAP NetWeaver にはデプロイされません。
- クラスターの EC2 インスタンスは、異なる AWS アベイラビリティーゾーンにある必要があります。

製品バージョン

- SAP NetWeaver バージョン 7.52 以降
- S/4HANA 2020 以降では、ENSA2 クラスターのみが適用
- カーネル 7.53 以降では、ENSA2 とエンキューレプリケーター 2 に適用
- SAP アプリケーションバージョン 12 以降のSLES
- ハイアベイラビリティ (HA) バージョン 7.9 以降の SAP 用 RHEL

アーキテクチャ

ソーステクノロジースタック

- SAP カーネル 7.53 以降を搭載した SAP NetWeaver 7.52
- SLES または RHEL オペレーティングシステム

ターゲットテクノロジースタック

- SAP カーネル 7.53 以降を搭載した SAP NetWeaver 7.52 (ABAP プラットフォーム搭載の S/4HANA 2020 を含む)
- SLES または RHEL オペレーティングシステム

ターゲットアーキテクチャ

次の図表は、ENSA2 クラスターに基づく ASCS/SCS インスタンスと ERS インスタンスの HA 構成を示しています。

ENSA1 クラスターと ENSA2 クラスターの比較

SAP は ENSA1 の後継として ENSA2 を導入しました。ENSA1 ベースのクラスターには、エラーが発生する場合、 ASCS/SCS インスタンスが ERS にフェイルオーバーする 2 ノードアーキテクチャが適用されます。この制限は、フェイルオーバー後に ASCS/SCS インスタンスが ERS ノードの共有メモリからロックテーブル情報を取り戻す方法によるものです。エンキューレプリケーター 2 を搭載した ENSA2 ベースのクラスターでは、ASCS/SCS インスタンスがネットワーク経由で ERS インスタンスからロック情報を収集できるため、この制限がなくなります。ASCS/SCS インスタンスは ERS ノードにフェイルオーバーする必要がなくなるため、ENSA2 ベースのクラスターは 3 つ以上のノードを持つことができます。(ただし、2 ノードの ENSA2 クラスター環境では、ASCS/SCS インスタンスは ERS ノードにフェイルオーバーされます。クラスターに他にフェイルオーバーするノードがないためです。ENSA2 は SAP カーネル 7.50 以降に適用されますが、いくつかの制限があります。エンキューレプリケーター 2 が適用される HA セットアップの場合、最小要件は NetWeaver 7.52 です (「SAP OSS ノート 2630416」 を参照)。S/4HANA 1809 にはデフォルトで推奨されている ENSA2 アーキテクチャが付属していますが、S/4HANA はバージョン 2020 以降には ENSA2 のみ適用されます。

自動化とスケール

ターゲットアーキテクチャの HA クラスタにより、ASCS は他のノードに自動的にフェイルオーバーされます。

ENSA2 ベースのクラスターに移動するシナリオ

ENSA2 ベースのクラスターへのアップグレードには、主に2つのシナリオがあります:

- シナリオ 1: SAP リリースとカーネルバージョンに ENSA2が適用されることを仮定して、SAP のアップグレードや S/4HANA の変換を伴わずに ENSA2 にアップグレードすることを選択します。
- シナリオ 2: SUM を使用して ENSA2 へのアップグレードまたは変換 (たとえば、S/4HANA 1809 以降へ) の一環として移動します。

「エピック」 セクションでは、2つのシナリオのステップについて説明します。最初のシナリオでは、ENSA2 のクラスター構成を変更する前に SAP 関連のパラメータを手動で設定する必要があります。二つ目のシナリオでは、バイナリと SAP 関連のパラメータは SUM によってデプロイされます。残る作業は HA のクラスター構成を更新することだけです。SUM を使用した後にも SAP パラメータを検証することを推奨します。ほとんどの場合、S/4HANA 変換がクラスタアップグレードの主な理由です。

ツール

- OS パッケージマネージャーには、Zypper (SLES の場合) または YUM (RHEL の場合) ツールを推奨します。
- クラスター管理には、crm(SLES の場合) または pcs (RHEL の場合) シェルを推奨します。
- SAPControl などの SAP インスタンス管理ツール。
- (オプション) S/4HANA 変換アップグレードの SUM ツール。

ベストプラクティス

- AWS で SAP ワークロードを使用する際のベストプラクティスについては、AWS Well-Architected フレームワークの「SAP Lens」を参照してください。
- ENSA2 マルチノードアーキテクチャのクラスターノードの数 (奇数または偶数)を考慮します。
- SAP S/4-HA-CLU 1.0 認定基準に沿って、SLES 15 用の ENSA2 クラスターをセットアップします。
- ENSA2 にアップグレードする前に、必ず既存のクラスターとアプリケーションの状態を保存またはバックアップするようにします。

エピック

ENSA2 の SAP パラメータを手動で設定する (シナリオ 1 のみ)

タスク	説明	必要なスキル

デフォルトのプロファイルにパラメータを設定します。	同じ SAP リリースを保持しながら ENSA2 にアップグレードする場合や、ターゲットリリースのデフォルトが ENSA1 である場合、デフォルトプロファイル (DEFAULT.PFL ファイル) のパラメータを次の値に設定します。
	`enq/enable=TRUE`
	`enq/serverhost=sapascsvirt`
		SAP

タスク	説明	必要なスキル

	enq/serverinst=10
	    (instance number of
	 ASCS/SCS instance)
	enque/process_loc
	ation=REMOTESA
	enq/replicatorhost=sa
	persvirt
	enq/replicatorinst=11
	    (instance number of
	 ERS instance)
	
	ここで、sapascsvirt がASCS インスタンスの仮想ホスト名で、sapersvirt はERS インスタンスの仮想ホスト名です。これらはターゲット環境に合わせて変更できます。
	
	Note
	このアップグレードオプションを使用するには、SAP リリースとカーネルバージョンがENSA2 とエンキューレプリケーター 2 をサポートしている必要があります。	

タスク	説明	必要なスキル
ASCS/SCS インスタンスプロファイルを設定します。	同じ SAP リリースを保持しながら ENSA2 にアップグレードする場合や、ターゲットリリースのデフォルトが ENSA1 である場合、ASCS/SCS インスタンスプロファイルに次のパラメータを設定します。 ENSA1 が定義されているプロファイルのセクションは以下のように表示されます。 `#------------------ ------------------ ------------------ --------- Start SAP enqueue server #------------------- ------------------ ------------------ ------- _EN = en.sap$(S APSYSTEMNAME)$(INS TANCE_NAME) Execute_04 = local rm - f $(_EN) Execute_05 = local ln - s -f $(DIR_EXECUTABLE)/ enserver$(FT_EXE) $(_EN) Start_Program_01 = local $(_EN) pf=$(_PF)` このセクションを ENSA2 用に再設定するには:	SAP

タスク	説明	必要なスキル
	1. _EN プログラムのプレフィックスを、SAP (OSS ノート 2501860) からの最新情報に基づいて _ENQ に変更します (SAP ONE サポートラウンチパッドのユーザーアカウントが必要です。) 2. エンキューサーバーのバイナリを enserver から enq_server に変更します。 3. 新パラメータ enq/server/replication/enable を TRUE に設定します。 4. Autostart = 0 を確保します。 変更後、このプロファイルセクションは以下のように見えるでしょう。 `#------------------ ------------------ ------------------ --------- Start SAP enqueue server #------------------- ------------------ ------------------ ------- _ENQ = enq.sap$( SAPSYSTEMNAME)$(IN STANCE_NAME)`	

タスク	説明	必要なスキル

	`Execute_04 = local rm -f $(_ENQ)`
	`Execute_05 = local ln -s -f $(DIR_EXECUTABLE)/enq_server$(FT_EXE) $(_ENQ)`
	`Start_Program_01 = local $(_ENQ) pf=$(_PF)`
	`...`
	`enq/server/replication/enable = TRUE`
	`Autostart = 0`
	
	Important
	_ENQ では、再起動オプションが有効になっていません。RestartProgram_01 が _ENQ に設定されている場合、StartProgram_01 に変更しこれにより、SAP がサービスを再起動したり、クラスターが管理するリソースに干渉したりするのを防止します。	

タスク	説明	必要なスキル
ERS プロファイルを設定します。	同じ SAP リリースを保持しながら ENSA2 にアップグレードする場合、またはターゲットリリースのデフォルトが ENSA1 である場合、次のパラメータを ERS インスタンスプロファイルに設定します。 エンキューレプリケーターが定義されているセクションを見つけます。以下に似ているものになります。 `#------------------ ------------------ ------------------- Start enqueue replicati on server #------------------- ------------------ ------------------ _ER = er.sap$(S APSYSTEMNAME)$(INS TANCE_NAME) Execute_03 = local rm - f $(_ER) Execute_04 = local ln - s -f $(DIR_EXECUTABLE)/ enrepserver$(FT_EXE) $(_ER) Start_Program_00 = local $(_ER) pf=$(_PF) NR=$(SCSID)` このセクションをエンキューレプリケーター 2 に再設定する場合：	SAP

タスク	説明	必要なスキル
	1. SAPからの最新のノート（OSS ノート2501860）に基づいて、_ER プログラムのプレフィックスを _ENQR に変更します。「SAP ONE サポートラウンチパッドのユーザーアカウント」が必要です。） 2. エンキューレプリケータのバイナリを enrepserver の代わりに enq_replicator に変更します。 3. Autostart = 0 を確保します。 変更後、このプロファイルセクションは以下のように表示されます。 `#------------------ ------------------ ------------------- Start enqueue replicati on server #------------------- ------------------ ------------------ _ENQR = enqr.sap$ (SAPSYSTEMNAME)$(I NSTANCE_NAME) Execute_01 = local rm - f $(_ENQR) Execute_02 = local ln - s -f $(DIR_EXECUTABLE)/ enq_replicator$(FT _EXE) $(_ENQR)`	

タスク	説明	必要なスキル
	`Start_Program_00 = local $(_ENQR) pf= $(_PF) NR=$(SCSID) … Autostart = 0` **Important** _ENQR では、再起動オプションが有効になっていません。RestartProgram_01 が _ENQR に設定されている場合、StartProgram_01 に変更します。これにより、SAP がサービスを再起動したり、クラスターが管理するサービスに干渉したりすることを防止します。	

タスク	説明	必要なスキル

SAP スタートサービスを再起動します。	このエピックで前述したプロファイルを変更した後に、ASCS/SCS と ERS の両方の SAP スタートサービスを再起動します。
	
	`sapcontrol -nr 10 -function RestartService SCT`
	
	`sapcontrol -nr 11 -function RestartService SCT`
	
	ここで SCT は SAP システム ID を指し、ASCS/SCS インスタンスと ERS インスタンスのインスタンス番号がそれぞれ 10 と 11 であると仮定します。
		SAP

ENSA2 用にクラスターを再構成します (両方のシナリオも必要)。

タスク	説明	必要なスキル
SAP リソースエージェントのバージョン番号を検証します。	SUM を使用して SAP を S/4HANA 1809 以降にアップグレードして、SUM は SAP プロファイルのパラメータ変更を処理します。クラスターのみが手動調整が必要です。ただし、クラスターを変更する前に、パラメータ設定を確認することを推奨します。	AWS システム管理者

タスク	説明	必要なスキル

	Note
	このエピックの例は、SUSE オペレーティングシステムを使用していることを前提としています。RHEL を使用する場合、Zypper や crm の代わりに YUM や pcs シェルなどのツールを使用する必要があります。
	
	アーキテクチャ内の両方のノードをチェックして、`resource-agents` パッケージが SAP が推奨する最小バージョンと一致していることを確認します。SLES については、SAP OSS ノート 2641019 を確認します。RHEL については、SAP OSS ノート 2641322 を確認します。(SAP Notes では 「SAP ONE サポートラウンチパッドのユーザーアカウント」 が必要です。)
	
	`sapers:sctadm 23>`
	`zypper search -s -i`
	`resource-agents`
	`Loading repository`
	`data...`
	`Reading installed`
	`packages...`	

タスク	説明	必要なスキル
	`S | Name | Type | Version | Arch | Repository --+------------ ----+---------+--- ------------------ ---------------+-- ------+----------- ------------------ i | resource-agents | package | 4.8.0+git30.d0077df0-150300.8.28.1 | x86_64 | SLE-Product-HA15-SP3-Updates` 必要に応じて、resource-agents バージョンを更新します。	
クラスター設定をバックアップします。	CRM クラスター構成を次のようにバックアップします。 `crm configure show > /tmp/cluster_config_backup.txt`	AWS システム管理者
メンテナンスモードを設します。。	クラスターをメンテナンスモードに設定します。 `crm configure property maintenance-mode="true"`	AWS システム管理者

タスク	説明	必要なスキル

クラスター設定を確認します	現在のクラスター設定をチェックします。
	
	`crm configure show`
	
	以下は、完全な出力からの抜粋です：
	
	`node 1: sapascs`
	`node 2: sapers`
	`...`
	`primitive rsc_sap_S`
	`CT_ASCS10 SAPInstance`
	` \`
	`operations $id=rsc_s`
	`ap_SCT_ASCS10-oper`
	`ations \`
	`op monitor interval=120`
	` timeout=60 on-fail=r`
	`estart \`
	`params InstanceN`
	`ame=SCT_ASCS10_sap`
	`ascsvirt START_PRO`
	`FILE="/sapmnt/SCT/`
	`profile/SCT_ASCS10`
	`_sapascsvirt" \`
	`  AUTOMATIC_RECOVER=`
	`false \`
	`meta resource-stickines`
	`s=5000 failure-t`
	`imeout=60 migration-`
	`threshold=1 priority=`
	`10`
	`primitive rsc_sap_S`
	`CT_ERS11 SAPInstance \`
	`operations $id=rsc_s`
	`ap_SCT_ERS11-opera`
	`tions \`
	`op monitor interval=120`
	` timeout=60 on-fail=r`
	`estart \`
		AWS システム管理者

タスク	説明	必要なスキル
	params InstanceN ame=SCT_ERS11_sape rsvirt START_PRO FILE="/sapmnt/SCT/ profile/SCT_ERS11_ sapersvirt" \ AUTOMATIC_RECOVER= false IS_ERS=true \ meta priority=1000 ... colocation col_sap_S CT_no_both -5000: grp_SCT_ERS11 grp_SCT_ASCS10 location loc_sap_S CT_failover_to_ers rsc_sap_SCT_ASCS10 \ rule 2000: runs_ers_SCT eq 1 order ord_sap_S CT_first_start_asc s Optional: rsc_sap_S CT_ASCS10:start rsc_sap_SCT_ERS11: stop symmetrical=false ... ここで sapascsvirt はASCS インスタンスの仮想ホスト名、sapersvirt はERS インスタンスの仮想ホスト名、SCT はSAP システム ID を指します。	

タスク	説明	必要なスキル
フェイルオーバーコロケーションの制約を削除します。	前の例では、ロケーション制約 `loc_sap_SCT_failover_to_ers` は、フェイルオーバー時に ASCS の ENSA1 特徴量が常に ERS インスタンスに従うように指定されています。ENSA2 では、ASCS は参加しているすべてのノードに自由にフェイルオーバーできるはずなので、この制約を取り除くことができます。 `crm configure delete loc_sap_SCT_failover_to_ers`	AWS システム管理者

タスク	説明	必要なスキル
プリミティブを調整します。	ASCS と ERS の SAPInstance プリミティブにも若干の変更を加える必要があります。 ENSA1 用に設定された ASCS SAP インスタンスプリミティブの例を次に示します。 `primitive rsc_sap_S CT_ASCS10 SAPInstance \ operations $id=rsc_s ap_SCT_ASCS10-oper ations \ op monitor interval=120 timeout=60 on-fail=r estart \ params InstanceN ame=SCT_ASCS10_sap ascsvirt START_PRO FILE="/sapmnt/SCT/ profile/SCT_ASCS10 _sapascsvirt" \ AUTOMATIC_RECOVER= false \ meta resource-stickines s=5000 failure-t imeout=60 migration- threshold=1 priority= 10`  ENSA2 にアップグレードするには、この設定を次のように変更します。  `primitive rsc_sap_S CT_ASCS10 SAPInstance \`	AWS システム管理者

タスク	説明	必要なスキル
	operations $id=rsc_sap_SCT_ASCS10-operations \ op monitor interval=120 timeout=60 on-fail=restart \ params InstanceName=SCT_ASCS10_sapascsvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ASCS10_sapascsvirt" \ AUTOMATIC_RECOVER=false \ meta resource-stickiness=3000 これは ENSA1 に設定された ERS SAPInstance プリミティブの例です。 primitive rsc_sap_SCT_ERS11 SAPInstance \ operations $id=rsc_sap_SCT_ERS11-operations \ op monitor interval=120 timeout=60 on-fail=restart \ params InstanceName=SCT_ERS11_sapersvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ERS11_sapersvirt" \ AUTOMATIC_RECOVER=false IS_ERS=true \ meta priority=1000	

タスク	説明	必要なスキル

	ENSA2 にアップグレードするには、この設定を次のように変更します。
	
	`primitive rsc_sap_SCT_ERS11 SAPInstance \ operations $id=rsc_sap_SCT_ERS11-operations \ op monitor interval=120 timeout=60 on-fail=restart \ params InstanceName=SCT_ERS11_sapersvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ERS11_sapersvirt" \ AUTOMATIC_RECOVER=false IS_ERS=true`
	
	プリミティブは、さまざまな方法で変更できます。例えば、次の例のように、vi などの工ディタで修正できます。
	
	`crm configure edit rsc_sap_SCT_ERS11`	

タスク	説明	必要なスキル
メンテナンスモードを無効にします。	クラスターのメンテナンスモードを無効にします。 `crm configure property maintenance-mode="false"` クラスターがメンテナンスモードを終了すると、新しい ENSA2 設定で ASCS インスタンスと ERS インスタンスをオンラインにしようとします。	AWS システム管理者

(オプション) クラスターノードを追加

タスク	説明	必要なスキル
ベストプラクティスをレビューします。	より多くのノードを追加する前に、使うノードの数が奇数か、偶数かなどのベストプラクティスを必ず理解するようにしてください。	AWS システム管理者
ノードを追加します。	もっと多くのノードを追加するには、オペレーティングシステムの更新、既存のノードと一致するソフトウェアパッケージのインストール、マウントを使用可能にするなど、一連のタスクが必要です。SAP ソフトウェアプロビジョニングマネージャー (SWPM) の追加ホストの準備オプションを使用して、ホストの SAP	AWS システム管理者

タスク	説明	必要なスキル
	特定のベースラインを作成できます。詳細について、次のセクションの SAP ガイドをご覧ください。	

関連リソース

SAP と SUSE のリファレンス

SAP ノートにアクセスするには、 SAP ONE サポートラウンチパッドのユーザーアカウントが必要です。詳細については、「SAP サポートウェブサイト」 を参照してください。

- 「SAP ノート 2501860 – ABAP 7.52 の SAP NetWeaver アプリケーションサーバーのドキュメント」
- 「SAP NOTE 2641019 – SUSE HA 環境のENSA2 のインストールと ENSA1 から ENSA2 にアップデート」
- 「SAP ノート 2641322 – SAP の Red Hat HA ソリューションを使用する場合の ENSA2 のインストールと ENSA1 から ENSA2 にアップデート」
- 「SAP ノート 2711036 – HA 環境のスタンドアロンエンキューサーバー 2 の使用」
- 「スタンドアロンエンキューサーバー2 」 (SAP ドキュメント)
- 「SAP S/4 HANA – エンキューレプリケーション 2 ハイアベイラビリティクラスタ-のセットアップガイド」 (SUSE ドキュメント)

AWS リファレンス

- 「SAP HANA on AWS: SLES と RHEL のハイアベイラビリティ設定ガイド」
- 「SAP Lens - AWS Well-Architected フレームワーク」

異なる AWS アカウント間の VPC で一貫したアベイラビリティーゾーンを使用する

アダム・スパイサー (AWS) によって作成されました

概要

Amazon Web Services (AWS) クラウドでは、アベイラビリティーゾーンには AWS アカウントによって異なる名前と、その場所を識別する「アベイラビリティーゾーン ID (AZ ID)」があります。AWS CloudFormation を使用して仮想プライベートクラウド (VPC) を作成する場合、サブネットを作成するときにアベイラビリティーゾーンの名前または ID を指定する必要があります。複数のアカウントで VPC を作成する場合、アベイラビリティーゾーン名はランダム化されます。つまり、サブネットはアカウントごとに異なるアベイラビリティーゾーンを使用します。

複数のアカウントに同じアベイラビリティーゾーンを使用するには、各アカウントのアベイラビリティーゾーン名を同じ AZ ID にマッピングする必要があります。たとえば、次の図は、`use1-az6` AZ ID が AWS アカウント A では `us-east-1a`、AWSアカウントZでは `us-east-1c` という名前になっていることを示している。

このパターンは、サブネット内の同じアベイラビリティーゾーンを使用するための、クロスアカウントでスケーラブルなソリューションを提供することで、ゾーンの一貫性を確保するのに役立ちます。ゾーンの整合性により、クロスアカウントのネットワークトラフィックがアベイラビリティーゾーン間のネットワークパスを回避できるため、データ転送コストを削減し、ワークロード間のネットワークレイテンシーを短縮できます。

このパターンは、AWS CloudFormation「アベイラビリティーゾーン」プロパティの代替アプローチです。

前提条件と制限

前提条件

- 同じ AWS リージョンの少なくとも 2 つのアクティブな AWS アカウント。
- リージョン内の VPC 要件をサポートするのに必要なアベイラビリティーゾーンの数を評価してください。
- サポートする必要のある各アベイラビリティーゾーンの AZ ID を特定して記録します。詳細については、AWS ResAWS Resource Access Manager ドキュメントの「AWS リソースのアベイラビリティーゾーン ID」を参照してください。

- AZ ID の順序付きカンマ区切りリスト。たとえば、リストの最初のアベイラビリティーゾーンは az1 としてマッピングされ、2 番目のアベイラビリティーゾーンは az2 としてマップされます。このマッピング構造は、カンマで区切られたリストが完全にマップされるまで続きます。マッピングできる AZ ID の数に上限はありません。
- ローカルマシンにコピーされた GitHub「マルチアカウントアベイラビリティーゾーンのマッピング」リポジトリの `az-mapping.yaml` ファイル

アーキテクチャ

次の図は、アカウントにデプロイされ、AWS Systems Manager Parameter Store 値を作成するアーキテクチャを示しています。これらのパラメータストア値は、アカウントに VPC を作成するときに消費されます。

この図表は、次のワークフローを示しています：

1. このパターンのソリューションは、VPC のゾーン整合性を必要とするすべてのアカウントにデプロイされます。
2. このソリューションでは、AZ ID ごとにパラメータストア値を作成し、新しいアベイラビリティーゾーン名を保存します。
3. AWS CloudFormation テンプレートは、各パラメータストア値に保存されているアベイラビリティーゾーン名を使用するため、ゾーンの一貫性が確保されます。

次の図は、このパターンのソリューションを使用して VPC を作成するワークフローを示しています。

この図表は、次のワークフローを示しています：

1. VPC を作成するためのテンプレートを AWS CloudFormation に送信します。
2. AWS CloudFormation は各アベイラビリティーゾーンのパラメータストア値を解決し、各 AZ ID のアベイラビリティーゾーン名を返します。
3. VPC は、ゾーンの整合性に必要な正しい AZ ID で作成されます。

このパターンのソリューションをデプロイしたら、Parameter Store 値を参照するサブネットを作成できます。AWS CloudFormation を使用する場合、次の YAML 形式のサンプルコードからアベイラビリティーゾーンのマッピングパラメータ値を参照できます。

```
Resources:
    PrivateSubnet1AZ1:
        Type: AWS::EC2::Subnet
        Properties:
            VpcId: !Ref VPC
            CidrBlock: !Ref PrivateSubnetAZ1CIDR
            AvailabilityZone:
                !Join
                    - ''
                    - - '{{resolve:ssm:/az-mapping/az1:1}}'
```

このサンプルコードは、GitHub「マルチアカウントアベイラビリティーゾーンマッピングリ」ポジトリの `vpc-example.yaml` ファイルに含まれています。ゾーンの整合性を保つために、パラメータストアの値に合わせた VPC とサブネットを作成する方法について説明します。

テクノロジースタック

- AWS CloudFormation
- AWS Lambda
- Systems Manager パラメータストア

自動化とスケール

このパターンは、AWS CloudFormation StackSets または AWS Control Tower 向けカスタマイズソリューションを使用して、すべての AWS アカウントにデプロイできます。詳細については、AWS Cloudformation ドキュメントの「AWS CloudFormation StackSets 使用」と、AWS ソリューションライブラリの「AWS Control Tower のカスタマイズ」を参照してください。

AWS CloudFormation テンプレートをデプロイしたら、パラメータストアの値を使用するようにテンプレートを更新し、パイプラインに、または要件に応じて VPC をデプロイできます。

ツール

AWS サービス

- 「AWS CloudFormation」は、AWS リソースのモデル化と設定、迅速かつ一貫したプロビジョニング、ライフサイクル全体にわたる管理を支援します。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理およびプロビジョニングできます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「AWS Systems Manager Parameter Store」は AWS Systems Manager の一機能です。設定データ管理と機密管理のための安全な階層型ストレージを提供します。

コード

このパターンのコードは、GitHub 内の「マルチアカウントアベイラビリティーゾーンのマッピング」リポジトリで利用できます。

エピック

AZ マッピング.yaml ファイルをデプロイします。

タスク	説明	必要なスキル
リージョンの必要なアベイラビリティーゾーンを決定します。	1. リージョンで一貫して使用する必要がある AZ ID を決定してください。 2. これらの AZ ID は、カンマ区切りのリストに、適用したい順序に記録します。たとえば、リストの最初のアベイラビリティーゾーンは az1 としてマッピングされ、2 番目のアベイラビリティーゾーンは az2 としてマッピングされます。マッ	クラウドアーキテクト

タスク	説明	必要なスキル
	ピングできる AZ ID の数に上限はありません。	
az-mapping.yaml ファイルをデプロイします。	`az-mapping.yaml` ファイルを使用して、必要なすべての AWS アカウントに AWS CloudFormation スタックを作成します。`AZIds` パラメータには、先ほど作成したカンマ区切りリストを使用します。 「AWS CloudFormation StackSets」または「AWS Control Tower ソリューション」のカスタマイズを使用することをお勧めします。	クラウドアーキテクト

VPC をアカウントにデプロイ

タスク	説明	必要なスキル
AWS CloudFormation テンプレートをカスタマイズします。	AWS CloudFormation を使用してサブネットを作成する場合、以前に作成したパラメータストア値を使用するようにテンプレートをカスタマイズします。 サンプルテンプレートについては、GitHub「マルチアカウントアベイラビリティーゾーンのマッピング」リポジトリにある `vpc-example.yaml` ファイルを参照してください。	クラウドアーキテクト

タスク	説明	必要なスキル
VPC をデプロイします。	カスタマイズした AWS CloudFormation テンプレートをアカウントにデプロイします。これにより、リージョン内の各 VPC は、サブネットに使用されるアベイラビリティーゾーンでゾーン整合性を保ちます。	クラウドアーキテクト

関連リソース

- 「AWS リソースのアベイラビリティーゾーン ID」(AWS Resource Access Manager のドキュメント)
- 「AWS:: EC2:: サブネット」(AWS CloudFormation ドキュメント)

アクセスコントロールと自動化に IAM ポリシーでユーザー IDs

作成者: Srinivas Ananda Babu (AWS) と Ram Kandaswamy (AWS)

概要

このパターンでは、 AWS Identity and Access Management (IAM) でユーザー名ベースのポリシーを使用する潜在的な落とし穴、ユーザー IDs を使用する利点、およびこのアプローチを と統合して AWS CloudFormation 自動化する方法を説明します。

では AWS クラウド、IAM サービスは、ユーザー ID とアクセスコントロールを正確に管理するのに役立ちます。ただし、IAM ポリシー作成でユーザー名に依存すると、予期しないセキュリティリスクやアクセスコントロールの問題が発生する可能性があります。例えば、新しい従業員である John Doe がチームに加わり、ユーザー名 を使用して IAM ユーザーアカウントを作成し`j.doe`、ユーザー名を参照する IAM ポリシーを通じてアクセス許可を付与します。John が退職すると、アカウントは削除されます。この問題は、新しい従業員である Jane Doe がチームに加わり、`j.doe`ユーザー名が再作成されたときに発生します。既存のポリシーは、John Doe と同じアクセス許可を Jane Doe に付与するようになりました。これにより、潜在的なセキュリティとコンプライアンスの悪夢が生じます。

新しいユーザーの詳細を反映するように各ポリシーを手動で更新することは、特に組織の成長に応じて、時間がかかり、エラーが発生しやすいプロセスです。解決策は、一意で変更不可能なユーザー ID を使用することです。IAM ユーザーアカウントを作成すると、 は IAM ユーザーに一意のユーザー ID (またはプリンシパル ID) を AWS 割り当てます。これらのユーザー IDs を IAM ポリシーで使用して、ユーザー名の変更や再利用の影響を受けない、一貫性のある信頼性の高いアクセスコントロールを確保できます。

たとえば、ユーザー ID を使用する IAM ポリシーは次のようになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::example-bucket",
            "Principal": { "AWS": "arn:aws:iam::123456789012:user/abcdef01234567890" }
        }
      ]
}
```

IAM ポリシーでユーザー IDs を使用する利点は次のとおりです。

- 一意性。ユーザー IDsはすべての で一意であるため AWS アカウント、正確で一貫性のあるアクセス許可アプリケーションを提供します。
- イミュータビリティ。ユーザー IDsは変更できないため、ポリシーでユーザーを参照するための安定した識別子を提供します。
- 監査とコンプライアンス。 AWS のサービス 多くの場合、ログと監査証跡にユーザー IDs が含まれているため、特定のユーザーにアクションを簡単にトレースできます。
- 自動化と統合。 AWS APIs、SDKs、またはオートメーションスクリプトでユーザー IDs を使用すると、プロセスはユーザー名の変更の影響を受けません。
- 将来性。最初からポリシーでユーザー IDs を使用すると、潜在的なアクセスコントロールの問題やポリシーの広範な更新を防ぐことができます。

オートメーション

などの Infrastructure as Code (IaC) ツールを使用すると AWS CloudFormation、ユーザー名ベースの IAM ポリシーの落とし穴が引き続き問題を引き起こす可能性があります。`Ref` 組み込み関数を呼び出すと、IAM ユーザーリソースはユーザー名を返します。組織のインフラストラクチャが進化するにつれて、IAM ユーザーアカウントなどのリソースを作成および削除するサイクルは、ユーザー名を再利用すると意図しないアクセスコントロールの問題につながる可能性があります。

この問題に対処するには、ユーザー IDsを CloudFormation テンプレートに組み込むことをお勧めします。ただし、この目的でユーザー IDsを取得するのは難しい場合があります。ここでカスタムリソースが役立ちます。CloudFormation カスタムリソースを使用して、 AWS APIs または外部サービスと統合することで、サービスの機能を拡張できます。特定の IAM ユーザーのユーザー ID を取得するカスタムリソースを作成することで、CloudFormation テンプレート内でユーザー ID を使用できるようになります。このアプローチにより、ユーザー IDs を参照するプロセスが合理化され、自動化ワークフローが堅牢で将来も確実に維持されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- クラウド管理者が AWS CloudFormation テンプレートを実行するための IAM ロール

制約事項

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ターゲットアーキテクチャ

次の図は、 AWS CloudFormation がサポートするカスタムリソース AWS Lambda を使用して IAM ユーザー ID を取得する方法を示しています。

自動化とスケール

CloudFormation テンプレートは、異なる AWS リージョン および アカウントに複数回使用できます。各リージョンまたはアカウントで 1 回のみ実行できます。

ツール

AWS サービス

- IAM – AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- AWS CloudFormation - AWS リソースを AWS CloudFormation モデル化してセットアップできるため、リソースの管理に費やす時間を減らし、 が実行されるアプリケーションに集中できます AWS。必要な AWS リソースを記述するテンプレートを作成すると、CloudFormation がそれらのリソースのプロビジョニングと設定を処理します。
- AWS Lambda – は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートするコンピューティングサービス AWS Lambda です。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。

ベストプラクティス

ゼロから開始する場合やグリーンフィールドデプロイを計画する場合は、一元化されたユーザー管理AWS IAM Identity Centerに を使用することを強くお勧めします。IAM Identity Center は既存の ID プロバイダー (Active Directory や Okta など) と統合してユーザー ID をフェデレーションするため

AWS、IAM ユーザーを直接作成および管理する必要はありません。このアプローチは、一貫したアクセスコントロールを保証するだけでなく、ユーザーライフサイクル管理を簡素化し、 AWS 環境全体のセキュリティとコンプライアンスを強化するのに役立ちます。

エピック

アクセス許可を検証する

タスク	説明	必要なスキル
AWS アカウント と IAM ロールを検証します。	に CloudFormation テンプレートをデプロイするアクセス許可を持つ IAM ロールがあることを確認します AWS アカウント。 この手順の最後のステップで、CloudFormation コンソール AWS CLI の代わりに を使用してテンプレートをデプロイする場合は、 AWS CLI コマンドを実行するための一時的な認証情報も設定する必要があります。手順については、IAM ドキュメントを参照してください。	クラウドアーキテクト

CloudFormation テンプレートを構築する

タスク	説明	必要なスキル
CloudFormation テンプレートを作成します。	1. CloudFormation ドキュメントの手順に従って CloudFormation テンプレートを作成します。JSON 形式または YAML 形式を使用できます。このパターン	AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル
	は、YAML 形式を使用していることを前提としています。 2. テンプレートを という名前で保存しますget_unique_user_id.yaml 。	

ユーザー名の入力パラメータを追加します。	CloudFormation テンプレートの Parameters セクションに次のコードを追加します。
	`Parameters:`
	`  NewIamUserName:`
	`  Type: String`
	`  Description: Unique username for the new IAM user`
	このパラメータは、ユーザーにユーザー名の入力を求めます。
		AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル

カスタムリソースを追加して IAM ユーザーを作成します。	CloudFormation テンプレートの `Resources` セクションに次のコードを追加します。
	
	`Resources:`
	`  rNewIamUser:`
	`  Type: 'AWS::IAM ::User'`
	`  Properties:`
	`  UserName: !Ref NewIamUserName`
	
	このコードは、`NewIamUserName` パラメータで指定された名前で IAM ユーザーを作成する CloudFormation リソースを追加します。
		AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル
Lambda 関数の実行ロールを追加します。	このステップでは、IAM を取得するアクセス許可を AWS Lambda 関数に付与する IAM ロールを作成します `UserId`。Lambda が実行するために必要な以下の最小限のアクセス許可を指定します。 • `logs:CreateLogStream` • `logs:PutLogEvents` • `CreateLogGroup` • `iam:GetUser` • `lambda.amazonaws.com` 用の `AssumeRole` 実行ロールを作成する手順については、Lambda ドキュメントを参照してください。このロールは、Lambda 関数を作成する次のステップで参照します。	AWS 管理者、クラウドアーキテクト

タスク	説明	必要なスキル

Lambda 関数を追加して、一意の IAM を取得しますUserId。	このステップでは、一意の IAM を取得するための Python ランタイムを使用して Lambda 関数を定義しますUserId。これを行うには、CloudFormation テンプレートの Resources セクションに次のコードを追加します。を、最後のステップで作成した実行ロールの名前<<ROLENAME>> に置き換えます。
	
	`GetUserLambdaFunction:`
	`Type: 'AWS::Lambda::Function'`
	`Properties:`
	`Handler: index.handler`
	`Role: <<ROLENAME>>`
	`Timeout: 30`
	`Runtime: python3.11`
	`Code:`
	`ZipFile: |`
	`import cfnresponse, boto3`
	`def handler(event, context):`
	`try:`
	`print(event)`
	`user = boto3.client('iam').get_user(UserName=event['ResourceP`
		AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル
	roperties'][NewIa mUserName'])['User'] cfnrespon se.send(event, context, cfnrespon se.SUCCESS, {'NewIamU serId': user['Use rId'], 'NewIamUs erPath': user['Pat h'], 'NewIamUserArn': user['Arn']}) except Exception as e: cfnrespon se.send(event, context, cfnrespon se.FAILED, {'NewIamU ser': str(e)})	
カスタムリソースを追加します。	CloudFormation テンプレートの Resources セクションに次のコードを追加します。 rCustomGetUniqueUs erId: Type: 'Custom:: rCustomGetUniqueUs erIdWithLambda' Properties: ServiceToken: ! GetAtt GetUserLa mbdaFunction.Arn NewIamUserName: ! Ref NewIamUserName このカスタムリソースは Lambda 関数を呼び出して IAM を取得しますUserID。	AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル

CloudFormation 出力を定義します。	CloudFormation テンプレートの Outputsセクションに次のコードを追加します。
	
	`Outputs:`
	`  NewIamUserId:`
	`  Value: !GetAtt`
	`  rCustomGetUniqueUs`
	`erId.NewIamUserId`
	
	これにより、UserID新しい IAM ユーザーの IAM が表示されます。
		AWS DevOps、クラウドアーキテクト

テンプレートを保存します。	CloudFormation テンプレートに変更を保存します。	AWS DevOps、クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル

CloudFormation のテンプレートをデプロイします。	CloudFormation コンソールを使用してget_unique_user_id.yaml テンプレートをデプロイするには、CloudFormation ドキュメントの指示に従います。
	
	または、次の AWS CLI コマンドを実行してテンプレートをデプロイすることもできます。
	
	`aws cloudformation`
	`  create-stack \`
		AWS DevOps、クラウドアーキテクト

タスク	説明	必要なスキル

	`--stack-name DemoNewUser \`
	`--template-body file://get_unique_user_id.yaml \`
	`--parameters ParameterKey=NewIamUserName,ParameterValue=demouser \`
	`--capabilities CAPABILITY_NAMED_IAM`	

関連リソース

- CloudFormation コンソールからスタックを作成する (CloudFormation ドキュメント)
- Lambda-backed カスタムリソース (CloudFormation ドキュメント)
- 一意の識別子 (IAM ドキュメント)
- AWS リソースで一時的な認証情報を使用する (IAM ドキュメント)

Account Factory for Terraform (AFT) のコードをローカルで検証する

アレクサンドル・ポップ (AWS) とミハル・ゴーニアック (AWS) が制作

概要

注意: AWS CodeCommit は新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、 AWS Control Tower Account Factory for Terraform (AFT) によって管理される HashiCorp Terraform コードをローカルでテストする方法を示しています。Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するための Infrastructure as Code (IaC) ツールです。AFT は、複数の AWS アカウント をプロビジョニングおよびカスタマイズするのに役立つ Terraform パイプラインを設定します AWS Control Tower。

コード開発時には、Terraform Infrastructure as Code (IaC) を AFT パイプラインの外部でローカルでテストすると役立つ場合があります。このパターンは、次を実行する方法を説明しています。

- AFT 管理アカウントの AWS CodeCommit リポジトリに保存されている Terraform コードのローカルコピーを取得します。
- 取得したコードを使用して AFT パイプラインをローカルでシミュレートします。

このプロシージャは、通常の AFT パイプラインに含まれていない Terraform コマンドを実行する場合にも使用できます。たとえば、このメソッドを使用して、`terraform validate`、`terraform plan`、`terraform destroy`や`terraform import`などのコマンドを実行できます。

前提条件と制限

前提条件

- が使用するアクティブな AWS マルチアカウント環境 AWS Control Tower
- 完全にデプロイされた AFT 環境。
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- AWS CLI の認証情報ヘルパー AWS CodeCommit、インストールおよび設定
- Python 3.x
- Git　ローカルマシンにインストールされて設定されている 。
- `git-remote-commit` ユーティリティ、インストールおよび設定済み

- 「Terraform」がインストールされ、構成されている (ローカルの Terraform パッケージのバージョンは AFT デプロイメントで使用されているバージョンと一致している必要がある)

機能制限

- このパターンは、、AFT AWS Control Tower、または特定の Terraform モジュールに必要なデプロイ手順には適用されません。
- この手順でローカルに生成された出力は、AFT パイプラインのランタイムログには保存されません。

アーキテクチャ

ターゲットテクノロジースタック

- デプロイ内に AWS Control Tower デプロイされた AFT インフラストラクチャ
- Terraform
- Git
- AWS CLI バージョン 2

自動化とスケール

このパターンは、単一の AFT 管理の AFT グローバルアカウントカスタマイズのために Terraform コードをローカルで呼び出す方法を示しています AWS アカウント。Terraform コードを検証したら、マルチアカウント環境の残りのアカウントにも適用できます。詳細については、 AWS Control Tower ドキュメントの「カスタマイズの再呼び出し」を参照してください。

同様のプロセスを使用して、ローカルターミナルで AFT アカウントのカスタマイズを実行することもできます。AFT アカウントのカスタマイズから Terraform コードをローカルで呼び出すには、AFT 管理アカウントの CodeCommit から 「aft-グローバル-アカウント-カスタマイズ リポジトリ」の代わりに「aft-アカウント-カスタマイズ」リポジトリを複製します。

ツール

AWS サービス

- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境のセットアップと管理に役立ちます。

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

その他のサービス

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。
- 「Git」はオープンソースの分散型バージョン管理システムです。

コード

以下は、AFT が管理する Terraform コードをローカルで実行するために使用できる bash スクリプトの例です。このスクリプトを使用するには、このパターンの「エピック」セクションの指示に従ってください。

```
#! /bin/bash
# Version: 1.1 2022-06-24 Unsetting AWS_PROFILE since, when set, it interferes with
 script operation
#          1.0 2022-02-02 Initial Version
#
# Purpose: For use with AFT: This script runs the local copy of TF code as if it were
 running within AFT pipeline.
#        * Facilitates testing of what the AFT pipline will do
#           * Provides the ability to run terraform with custom arguments (like 'plan'
 or 'move') which are currently not supported within the pipeline.
#
# © 2021 Amazon Web Services, Inc. or its affiliates. All Rights Reserved.
# This AWS Content is provided subject to the terms of the AWS Customer Agreement
# available at http://aws.amazon.com/agreement or other written agreement between
# Customer and either Amazon Web Services, Inc. or Amazon Web Services EMEA SARL or
 both.
#
# Note: Arguments to this script are passed directly to 'terraform' without parsing nor
 validation by this script.
#
# Prerequisites:
#    1. local copy of ct GIT repositories
#    2. local backend.tf and aft-providers.tf filled with data for the target account
 on which terraform is to be run
#       Hint: The contents of above files can be obtain from the logs of a previous
 execution of the AFT pipeline for the target account.
```

```
#    3. 'terraform' binary is available in local PATH
#    4. Recommended: .gitignore file containing 'backend.tf', 'aft_providers.tf' so the
 local copy of these files are not pushed back to git

readonly credentials=$(aws sts assume-role \
    --role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output
 text ):role/AWSAFTAdmin \
    --role-session-name AWSAFT-Session \
    --query Credentials )

unset AWS_PROFILE
export AWS_ACCESS_KEY_ID=$(echo $credentials | jq -r '.AccessKeyId')
export AWS_SECRET_ACCESS_KEY=$(echo $credentials | jq -r '.SecretAccessKey')
export AWS_SESSION_TOKEN=$(echo $credentials | jq -r '.SessionToken')
terraform "$@"
```

エピック

サンプルコードをローカルファイルとして保存します。

タスク	説明	必要なスキル
サンプルコードをローカルファイルとして保存します。	1. このパターンの「コード」「 」セクションにある bash スクリプトの例をコピーして、コードエディターに貼り付けます。 2. ファイル に名前を付けct_terraform.sh 、ファイルを ~/scripts やなどの専用フォルダ内にローカルに保存します~/bin。	AWS 管理者
サンプルコードを実行可能にします。	ターミナルウィンドウを開き、次のいずれかを実行して AWS AFT 管理アカウントを認証します。	AWS 管理者

タスク	説明	必要なスキル
	• AFT 管理アカウントへのアクセスに必要なアクセス許可で設定された既存のAWS CLI プロファイルを使用します。プロファイルを使用するには、次のコマンドを実行します。 `export AWS_PROFILE=<aft account profile name>` • 組織が SSO を使用して にアクセスする場合は AWS、組織の SSO ページで AFT 管理アカウントの認証情報を入力します。 **Note** 組織に、 AWS 環境に認証情報を提供するカスタムツールがある場合もあります。	

タスク	説明	必要なスキル

正しい で AFT 管理アカウントへのアクセスを確認します AWS リージョン。	**⚠ Important**
	AFT 管理アカウントで認証したのと同じターミナルセッションを使用していることを確認してください。
	
	1. 次のコマンド AWS リージョン を実行して、AFT デプロイの に移動します。
	`export AWS_REGIO N=<aft_region>`
	2. 正しいアカウントにいることを確認します。
	a. 次のコマンドを実行してください。
	`aws code-commit list-repositories`
	b. 出力にリストされているリポジトリが、AFT 管理アカウントにあるリポジトリの名前と一致していることを確認します。
		AWS 管理者

タスク	説明	必要なスキル

AFT リポジトリコードを保存する新しいローカルディレクトリを作成します。	同じターミナルセッションから、次のコマンドを実行します。
	
	`mkdir my_aft`
	`cd my_aft`
		AWS 管理者

タスク	説明	必要なスキル

リモート AFT リポジトリコードを複製します。	1. ローカルターミナルから次のコマンドを実行します。
	`git clone codecommit::$AWS_REGION://aft-global-customizations`
	Note
	簡単にするために、この手順と AFT はメインコードブランチのみを使用します。コード分岐を使用するには、ここにもコード分岐コマンドを入力します。しかし、AFT オートメーションがメインブランチからのコードを適用すると、メインブランチ以外から適用された変更はすべてロールバックされます。
	2. クローンされたディレクトリに移動します。
	`cd aft-global-customizations/terraform`
		AWS 管理者

AFT パイプラインをローカルで実行するために必要な Terraform 設定ファイルを作成します。

タスク	説明	必要なスキル
以前に実行した AFT パイプラインを開き、Terraform 設定ファイルをローカルフォルダーにコピーします。	**Note** このエピックで作成された `backend.tf` および `aft-providers.tf` 設定ファイルは、AFT パイプラインをローカルで実行するために必要です。これらのファイルはクラウドベースの AFT パイプライン内で自動的に作成されますが、パイプラインをローカルで実行するには手動で作成する必要があります。AFT パイプラインをローカルで実行するには、パイプラインの実行を表す 1 つのファイルセットが必要です AWS アカウント。 1. AWS Control Tower 管理アカウントの認証情報を使用して にサインインし AWS Management Console、AWS CodePipeline コンソールを開きます。AFT をデプロイした AWS リージョン のと同じ にいることを確認します。	AWS 管理者

タスク	説明	必要なスキル
	2. 左のナビゲーションペインの [パイプライン] を選択します。 3. #########-カスタマイズ-パイプラインを選択します。(########## は、Terraform コードをローカルで実行するために使用する AWS アカウント ID です)。 4. 「最後にマークされた実行」に「成功」の値が表示されていることを確認します。値が異なる場合は、AFT パイプラインでカスタマイズを再呼び出しする必要があります。詳細については、 AWS Control Tower ドキュメントの「カスタマイズの再呼び出し」を参照してください。 5. 最新のランタイムを選択すると、詳細が表示されます。 6. 「Apply-AFT-グローバルカスタマイズ」セクションで「Apply-Terraform」ステージを見つけてください。 7. 「Apply-Terraform」ステージの「詳細」セクションを選択します。 8. 「Apply-Terraform」ステージのランタイムログを検索してください。	

タスク	説明	必要なスキル

	9. ランタイムログで、次の行で始まるセクションと終わるセクションを探します。
	
	`"\n\n aft-providers.tf ... "\n \n backend.tf"`
	
	10.これら2つのラベル間の出力をコピーし、ローカルTerraformフォルダ（ターミナルセッションのカレントワーキングディレクトリ）内にaft-providers.tf という名前のローカルファイルとして保存します。
	
	自動的に生成されたproviders.tf ステートメントの例
	
	`## Autogenerated providers.tf ## ## Updated on: 2022-05-31 16:27:45 ## provider "aws" { region = "us-east-2" assume_role { role_arn = "arn:aws:iam::########:role/AWSAFTExecution" } default_tags { tags = {`	

タスク	説明	必要なスキル

	<pre>managed_by
	= "AFT"
	}
	}
	}</pre>
	11.ランタイムログで、次の行で始まるセクションと終わるセクションを探します。
	<pre>"\n\n tf
	...
	"\n \n backend.tf"</pre>
	12.これら2つのラベル間の出力をコピーし、ローカルTerraformフォルダ（ターミナルセッションのカレントワーキングディレクトリ）内にtf という名前のローカルファイルとして保存します。
	
	自動生成された backend.tf ステートメントの例
	<pre>## Autogenerated
	backend.tf ##
	## Updated on: 2022-05-3
	1 16:27:45 ##
	terraform {
	required_version =
	">= 0.15.0"
	backend "s3" {
	region =
	"us-east-2"
	bucket =
	"aft-backend-#####</pre>	

タスク	説明	必要なスキル

	<pre>#######-primary-region"
	    key = "############-aft-global-customizations/terraform.tfstate"
	    dynamodb_table = "aft-backend-###########"
	    encrypt = "true"
	    kms_key_id = "########-####-####-####-############"
	    role_arn = "arn:aws:iam::#########:role/AWSAFTExecution"
	  }
	}</pre>
	Note
	backend.tf および aft-providers.tf ファイルは AWS アカウント、特定の、AFT デプロイ、およびフォルダに関連付けられます。これらのファイルは、同じ AFT デプロイ内の aft-global-customizations リポジトリと aft-account-customizations リポジトリにあるかどうかによって	

タスク	説明	必要なスキル
	も異なります。必ず、同じランタイムリストから両方のファイルを生成してください。	

サンプルの bash スクリプトを使用して AFT パイプラインをローカルで実行します。

タスク	説明	必要なスキル
検証したい Terraform の設定の変更を実装します。	1. 以下のコマンドを実行して、複製された「aft-グローバル-カスタマイズ」リポジトリに移動します。 `cd aft-global-customizations/terraform` **Note** ファイル `backend.tf` と `aft-providers.tf` はこのディレクトリにあります。このディレクトリには、「aft-グローバル-カスタマイズ」リポジトリの Terraform ファイルも含まれています。 2. ローカルでテストしたい Terraform コードの変更を	AWS 管理者

タスク	説明	必要なスキル
	設定ファイルに組み込みます。	

タスク	説明	必要なスキル

ct_terraform.sh スクリプトを実行し、出力を確認します。	1. 「sh」スクリプトを含むローカルフォルダに移動します。
	2. 変更した Terraform コードを検証するには、以下のコマンドを実行してct_terraform.sh スクリプトを実行します。
	`~/scripts/ct_terraform.sh apply`
	`terraform --help`
	Note
	このステップでは、任意の Terraform コマンドを実行できます。次のコマンドを実行して、Terraform コマンドの全リストを表示します。
	3. コマンド出力を確認し、変更をコミットして AFT リポジトリにプッシュする前に、コードの変更をローカルでデバッグします。
		AWS 管理者

タスク	説明	必要なスキル
	Important • ローカルで行われ、リモートリポジトリにプッシュバックされない変更は一時的なものであり、実行中の AFT パイプラインオートメーションによっていつでも元に戻すことができます。 • AFT オートメーションは他のユーザーや AFT オートメーショントリガーによって呼び出される可能性があるため、いつでも実行できます。 • AFT は常にリポジトリのメインブランチからコードを適用し、コミットされていない変更はすべて元に戻します。	

ローカルコードの変更を AFT リポジトリにプッシュバックする

タスク	説明	必要なスキル
`.gitignore` ファイルへの `backend.tf` および `aft-providers.tf` ファイルへの参照を追加します。	以下のコマンドを実行して、作成した`backend.tf` と`aft-providers.tf` ファイルを`.gitignore` ファイルに追加します。	AWS 管理者

タスク	説明	必要なスキル

	`echo backend.tf >> .gitignore`
	`echo aft-providers.tf >>.gitignore`
	
	Note
	ファイルを `.gitignore` ファイルに移動すると、コミットされず、リモート AFT リポジトリにプッシュバックされません。	

タスク	説明	必要なスキル

コード変更をリモート AFT リポジトリにコミットしてプッシュします。	1. 新しい Terraform 設定ファイルをリポジトリに追加するには、次のコマンドを実行します。
	`git add <filename>`
	2. 変更をコミットして CodeCommitt のリモート AFT リポジトリにプッシュするには、次のコマンドを実行します。
	`git commit -a`
	`git push`
	Important
	導入するコードは、この時点で 1 つの AWS アカウント みに適用されるまで、この手順に従って変更されます。
		AWS 管理者

変更を複数のアカウントにロールアウトする

タスク	説明	必要なスキル
AFT によって管理されるすべてのアカウントに変更をロールアウトします。	AFT によって管理 AWS アカウント される複数の に変更をロールアウトするには、 AWS Control Tower ドキュメント	AWS 管理者

タスク	説明	必要なスキル
	の「カスタマイズの再呼び出し」の手順に従います。	

その他のパターン

- リードレプリカを使用して Amazon RDS Customの Oracle PeopleSoft に HA を追加
- パブリック IP アドレスからのアクセスを許可する AWS セキュリティグループを自動的に監査する
- で Landing Zone Accelerator を使用してアカウントの作成を自動化する AWS
- AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する
- AWS リソース評価を自動化する
- AWS CDK を使用して AWS Service Catalog ポートフォリオと製品のデプロイを自動化する
- DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する
- AWS CloudFormation スタックと関連リソースの削除を自動化する
- Terraform を使用して Amazon Managed Grafana での Amazon MWAA カスタムメトリクスの取り込みと視覚化を自動化する
- Amazon MQ で RabbitMQ 設定を自動化する
- 全体の Amazon RDS インスタンスのレプリケーションを自動化する AWS アカウント
- Cloud Custodian と AWS CDK を使用して、Systems Manager の AWS マネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする
- AWS CDK を使用してマイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する
- CodeCommit 内のモノリポジトリの変更を自動的に検出し、さまざまな CodePipeline パイプラインを開始する
- AWS Config のカスタム修復ルールを使用して AWS CloudTrail を自動的に再有効化する
- AWS DataOps Development Kit を使用して Google Analytics データを取り込み、変換、分析するためのデータパイプラインを構築する
- Amazon EC2 Auto Scaling と Systems Manager を搭載した Micro Focus Enterprise Server PAC を構築する
- GitHub Actions と Terraform を使用して Docker イメージを構築して Amazon ECR にプッシュする
- Terraform を使用して AWS Organizations の IAM アクセスキー管理を一元化する
- Terraform を使用して AWS Organizations でソフトウェアパッケージの配布を一元化する
- サーバーレスアプローチを使用して AWS サービスを連結する

- Hybrid Linked Mode を使用して VMware Cloud on AWS へのデータセンター拡張を構成する
- を使用して Amazon Bedrock でモデル呼び出しログ記録を設定する AWS CloudFormation
- AWS 上の SQL Server の Always On アベイラビリティグループで読み取り専用ルーティングを構成する
- VMware vRealize Automation を VMware Cloud on AWSにプロビジョニングするように設定
- 、Angular AWS Amplify、および Module Federation を使用してマイクロフロントエンド用のポータルを作成する
- 組織内でクロスアカウント Amazon EventBridge 接続を作成する
- Java および Python プロジェクト用の動的 CI パイプラインを自動的に作成
- VMware Cloud on AWS を使用して VMware SDDC on AWS をデプロイする
- プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API を内部 Web サイトにデプロイする
- Amazon EKS クラスターをデプロイおよびデバッグ
- と CloudFormation を使用して AWS Control Tower コントロールをデプロイ AWS CDK および管理します。
- Terraform を使用した AWS Control Tower コントロールのデプロイと管理
- Terraform を使用して CloudWatch Synthetics Canary をデプロイする
- Terraform と DRA を使用して高性能データ処理用の Lustre ファイルシステムをデプロイする
- Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする
- Terraform を使用して AWS Wavelength ゾーンにリソースをデプロイする
- Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする
- CA 証明書の有効期限が切れている Amazon RDS および Aurora データベースインスタンスを検出する
- AWS ランディングゾーンの設計を文書化する
- IAM プロファイルが EC2 インスタンスと確実に関連付けられているようにします。
- AWS Organizations 内の組織全体の AWS Backup レポートを CSV ファイルとしてエクスポートする
- Amazon Personalize を使用して、パーソナライズされ再ランク付けされたレコメンデーションを生成します
- Amazon Data Firehose リソースが AWS KMS キーで暗号化されていない場合の識別とアラート
- ブートストラップパイプラインを使用して Account Factory for Terraform (AFT) を実装する

- Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する
- Kubernetes DaemonSet を使用して Amazon EKS ワーカーノードに SSM エージェントをインストールします
- prebootstrapコマンドを使用して、SSM エージェントと CloudWatch エージェントを Amazon EKS ワーカーノードにインストールします。
- VMware vRealize Network Insight を VMware Cloud on AWS と統合する
- を使用してアクセス AWS IAM Identity Center 許可セットをコードとして管理する AWS CodePipeline
- 複数の AWS アカウントと AWS リージョンで AWS Service Catalog 製品を管理
- AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。
- DNS レコードを Amazon Route 53 プライベートホストゾーンに一括で移行する
- Oracle E-Business Suite を Amazon RDS Custom に移行
- Oracle PeopleSoft を Amazon RDS Custom に移行
- AWS MGN を使用して RHEL BYOL システムを AWS ライセンス込みのインスタンスに移行する
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX
- 組織間でデータを共有するための最小限の実行可能なデータスペースを設定する
- Amazon ElastiCache クラスターの保存時の暗号化をモニタリングする
- セキュリティグループの ElastiCache クラスターをモニタリング
- AWS のサービスを使用して SAP RHEL Pacemaker クラスターをモニタリングする
- AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する
- 複数の VPC から中央のAWS のサービスエンドポイントにプライベートにアクセスする
- GitHub Actions を使用して AWS CloudFormation テンプレートに基づいて AWS Service Catalog 製品をプロビジョニングする
- ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする
- AWS Lambda オートメーションを使用して 全体の Amazon EC2 エントリを AWS アカウント から削除 AWS Managed Microsoft AD する
- AWS Lambda オートメーションを使用して AWS アカウント から同じ の Amazon EC2 AWS Managed Microsoft AD エントリを削除する
- コンテナを再起動せずにデータベースの認証情報をローテーションする

- IAM ユーザーが作成されたときに通知を送信
- VMware Aria Operations for Logs を使用して VMware Cloud on から Splunk AWS にログを送信する
- セルベースのアーキテクチャ用にサーバーレスセルルーターを設定する
- AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する
- NICE EnginFrame と NICE DCV セッションマネージャーを使用して自動スケーリング仮想デスクトップインフラストラクチャを設定する
- Amazon RDS Custom でアクティブスタンバイデータベースを使用して Oracle E-Business Suite の HA/DR アーキテクチャを設定する
- マルチアカウント AWS 環境でハイブリッドネットワークの DNS 解決を設定する
- Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする
- Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする
- Application Load Balancer を使用して Amazon ECS の相互 TLS でアプリケーション認証を簡素化する
- AWS Private CA と AWS RAM を使用してプライベート証明書の管理を簡素化する
- SageMaker AI と Hybrid を使用して、ローカル開発からスケーラブルな実験まで機械学習ワークフローを合理化する
- AWS Organizations を使用して Transit Gateway アタッチメントに自動的にタグを付ける
- Amazon RDS Custom for Oracle 上の Oracle PeopleSoft アプリケーションの移行ロール
- Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める

ウェブおよびモバイルアプリ

トピック

- Amazon Cognito と AWS Amplify UI を使用して既存の React アプリケーションユーザーを認証する
- AWS CodeCommit リポジトリから最新の AWS Amplify ウェブアプリケーションを継続的にデプロイする
- AWS Amplify を使用して React アプリケーションを作成し、Amazon Cognito による認証を追加する
- 、Angular AWS Amplify、および Module Federation を使用してマイクロフロントエンド用のポータルを作成する
- React ベースのシングルページアプリケーションを Amazon S3 と CloudFront にデプロイする
- プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API を内部 Web サイトにデプロイする
- Amazon QuickSight ダッシュボードをローカルの Angular アプリケーションに埋め込みます。
- Green Boost を使用したフルスタックのクラウドネイティブなウェブアプリケーション開発を探索する
- AWS CodeBuild を使用して GitHub から Node.js アプリケーションのユニットテストを実行する
- AWS Lambda を使用して六角形アーキテクチャで Python プロジェクトを構築する
- その他のパターン

Amazon Cognito と AWS Amplify UI を使用して既存の React アプリケーションユーザーを認証する

作成者: Daniel Kozhemyako (AWS)

概要

このパターンは、 AWS Amplify UI ライブラリと Amazon Cognitouser プールを使用して、既存のフロントエンド React アプリケーションに認証機能を追加する方法を示しています。

このパターンでは、Amazon Cognito を使用して、アプリケーションの認証、認可、ユーザー管理を提供します。また、 の機能をユーザーインターフェイス (UI) 開発に拡張するオープンソースライブラリである Amplify UI のコンポーネントも使用します。 AWS Amplify Authenticator UI コンポーネントはログインセッションを管理し、Amazon Cognito を介してユーザーを認証するクラウド接続ワークフローを実行します。

このパターンを実装すると、ユーザーは次の認証情報のいずれかを使用してサインインできます。

- ユーザー名とパスワード
- Apple、Facebook、Google、Amazon などのソーシャル ID プロバイダー
- SAML 2.0 互換または OpenID Connect (OIDC) 互換のエンタープライズ ID プロバイダー

> **Note**
>
> カスタム認証 UI コンポーネントを作成するには、オーセンティケーター UI コンポーネントをヘッドレスモードで実行できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- React 18.2.0 以降のウェブアプリケーション
- Node.js および npm 6.14.4 以降がインストールされている

制約事項

- このパターンは React Web アプリケーションのみに適用されます。
- このパターンは、事前構築済みの Amplify UIコンポーネントを使用します。このソリューションには、カスタム UI コンポーネントの実装に必要な手順は含まれません。

製品バージョン

- Amplify UI 6.1.3 以降 (Gen 1)
- Amplify 6.0.16 以降 (Gen 1)

アーキテクチャ

ターゲット アーキテクチャ

次の図は、Amazon Cognito を使用して React ウェブアプリケーションのユーザーを認証するアーキテクチャを示しています。

ツール

AWS のサービス

- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。

その他のツール

- Amplify UI は、クラウドに接続できるカスタマイズ可能なコンポーネントを提供するオープンソースの UI ライブラリです。
- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。

ベストプラクティス

新しいアプリケーションを構築する場合は、Amplify Gen 2 を使用することをお勧めします。

エピック

Amazon Cognito ユーザープールを作成する

タスク	説明	必要なスキル
ユーザープールを作成します。	Amazon Cognito ユーザープールを作成します。ユースケースに合わせて、ユーザープールのサインインオプションとセキュリティ要件を設定します。	アプリ開発者
アプリクライアントを追加します。	ユーザープールアプリクライアントを設定します。このクライアントは、アプリケーションが Amazon Cognito ユーザープールとやり取りするために必要です。	アプリ開発者

Amazon Cognito ユーザープールを Authenticator UI コンポーネントと統合する

タスク	説明	必要なスキル
依存関係をインストールします。	`aws-amplify` および `@aws-amplify/ui-react` パッケージをインストールするには、アプリケーションのルートディレクトリから次のコマンドを実行します。 `npm i @aws-amplify/ui-react aws-amplify`	アプリ開発者
ユーザープールを設定します。	次の例に基づいて、 `aws-exports.js` ファイルを作成し、 `src`フォルダに保存し	アプリ開発者

タスク	説明	必要なスキル

	ます。ファイルには、次の情報が含まれている必要があります。
	
	• AWS リージョン Amazon Cognito ユーザープールが
	• Amazon Cognito ユーザープール ID
	• アプリクライアント ID
	
	`// replace the user pool region, id, and app client id details`
	`const awsmobile = {`
	`"aws_project_region": "put_your_region_here",`
	`"aws_cognito_region": "put_your_region_here",`
	`"aws_user_pools_id": "put_your_user_pool_id_here",`
	`"aws_user_pools_web_client_id": "put_your_user_pool_app_id_here"`
	`}`
	
	`export default awsmobile;`	

タスク	説明	必要なスキル

Amplify サービスをインポートして設定します。	1. アプリケーションのエントリポイントファイル (などApp.js) で、次のコード行を入力してaws-exports.js ファイルをインポートしてロードします。
	`import { Amplify } from 'aws-amplify'; import awsExports from './aws-exports';`
	2. 次の例に基づいて、 aws-exports.js ファイルを使用して Amplify クライアントを設定します。
	`// Configure Amplify in index file or root file Amplify.configure({ ...awsExports });`
	詳細については、Amplify ドキュメントの「Amplify カテゴリを設定する」を参照してください。
		アプリ開発者

タスク	説明	必要なスキル

Authenticator UI コンポーネントを追加します。	Authenticator UI コンポーネントを表示するには、アプリケーションのエントリポイントファイル (App.js) に次のコード行を追加します。
	
	`import { Authenticator } from '@aws-amplify/ui-react';`
	`import '@aws-amplify/ui-react/styles.css';`
	
	Note
	サンプルコードスニペットは、UI Authenticator コンポーネントと Amplify UI styles.css ファイルをインポートします。これは、コンポーネントの事前構築されたテーマを使用するときに必要です。
	
	UI コンポーネントには 2 Authenticator つの戻り値があります。
	
	- ユーザー詳細
	- ユーザーをサインアウトするために呼び出すことができる関数
		アプリ開発者

タスク	説明	必要なスキル
	次のコンポーネント例を参照してください。 `function App() { return ( <Authenticator> {({ signOut, user }) => ( <div> <p>Welcome {user.username}</p> <button onClick={signOut}>Sign out</button> </div> )} </Authenticator> ); }` **Note** App.js ファイル例については、このパターンの???「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル

(オプション) セッション情報を取得します。	ユーザーが認証されると、AAmplify クライアントからセッションに関するデータを取得できます。たとえば、ユーザーのセッションから JSON Web トークン (JWT) を取得して、そのユーザーのセッションからバックエンド API への要求を認証できます。
	
	JWT を含む要求ヘッダーの次の例を参照してください。
	
	`import { fetchAuthSession } from 'aws-amplify/auth';`
	`(await fetchAuthSession()).tokens?.idToken?.toString();`
		アプリ開発者

トラブルシューティング

問題	ソリューション
新規ユーザーはアプリケーションにサインアップできません。	次のように、Amazon Cognito ユーザープールが、ユーザーがユーザープールにサインアップできるように設定されていることを確認します。 • にサインインし AWS Management Console、Amazon Cognito コンソールを開きます。 • 左のナビゲーションペインで、[ユーザープール] を選択します。

問題	ソリューション
	• リストから ユーザープールを選択します。 • [General Settings (全般設定)] で、[Policies (ポリシー)] を選択します。 • [Allow users to sign themselves up (ユーザーにサインアップを許可)] を選択します。
v5 から v6 にアップグレードすると、認証コンポーネントが動作しなくなりました。	Auth カテゴリは、Amplify v6 の機能的なアプローチと名前付きパラメータに移行しました。これで、 aws-amplify/auth パスから直接機能 APIsをインポートする必要があります。詳細については、Amplify ドキュメントの「v5 から v6 への移行」を参照してください。

関連リソース

- Amazon Cognito の開始方法 (AWS ウェブサイト)
- 新しい React アプリを作成する (React ドキュメント)
- Amazon Cognito とは (Amazon Cognito ドキュメント)
- Amplify UI ライブラリ (Amplify ドキュメント)

追加情報

App.js ファイルには、次のコードが含まれている必要があります。

```
import './App.css';
import { Amplify } from 'aws-amplify';
import awsExports from './aws-exports';
import { fetchAuthSession } from 'aws-amplify/auth';
import { Authenticator } from '@aws-amplify/ui-react';
import '@aws-amplify/ui-react/styles.css';
Amplify.configure({ ...awsExports });
let token = (await fetchAuthSession()).tokens?.idToken?.toString();
function App() {
  return (
      <Authenticator>
        {({ signOut, user }) => (
```

```
            <div>
              <p>Welcome {user.username}</p>
                <p>Your token is: {token}</p>
              <button onClick={signOut}>Sign out</button>
            </div>
        )}
      </Authenticator>
  );
}

export default App;
```

AWS CodeCommit リポジトリから最新の AWS Amplify ウェブアプリケーションを継続的にデプロイする

ディークシトゥル・ペンタコタ（AWS）とサイ・カタカム（AWS）によって作成されました

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら。

最新のウェブアプリケーションは、すべてのアプリケーションコンポーネントを静的ファイルにパッケージ化する単一ページのウェブアプリケーションとして構築されます。AWS Amplify ホスティング を使用すると、Git ベースのリポジトリで管理される最新のウェブアプリケーションをビルド、デプロイ、ホストする継続的インテグレーションおよび継続的デプロイ (CI/CD) パイプラインを構築できます。Amplify Hosting をコードリポジトリに接続すると、コミットごとにアプリケーションのフロントエンドとバックエンドをデプロイする単一のワークフローが開始されます。これにより、アプリケーションのフロントエンドとバックエンドの間の矛盾は解消され、デプロイが正常に完了した場合にのみ、ウェブアプリケーションが更新されます。

このパターンでは、AWS CodeCommit リポジトリを使用して最新のウェブアプリケーションを管理します。このインストラクションのサンプルウェブアプリケーションは React SPA フレームワークを使用しています。ただし、Amplify Hostingは、Angular、Vue、Next.js など他の多くのSPAフレームワークをサポートしており、Gatsby、Hugo、Jekyllなどのシングルサイトジェネレーターもサポートしています。

このパターンは、以下のサービスとコンセプトの経験がある AWS ビルダーを対象としています。

- AWS CodeCommit
- AWS Amplify ホスティング
- React
- JavaScript
- Node.js
- npm
- Git

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amplify と CodeCommit でリソースを作成するための権限。詳細については、「Amplify の Identity および Access Management」と「AWS CodeCommit の Identity および Access Management」を参照してください。
- AWS コマンドラインインターフェイス (AWS CLI)は、「インストール」および「設定」されています。
- テキストエディタまたはコードエディタ。
- CodeCommit、Git 認証情報を使用した HTTPS ユーザーのセットアップ
- AAmplify の「IAMサービスロール」です。
- npm と Node.js が「インストールされました」 (npm ドキュメント)。

制約事項

- このパターンでは、API、認証、データベースなど、Amplify アプリケーションのバックエンドの開発と統合については説明していません。バックエンドの詳細については、Amplify ドキュメントの「バックエンドの作成」を参照してください。

製品バージョン

- AWS CLI バージョン 2
- Node.js バージョン 16.x 以降

アーキテクチャ

ターゲットテクノロジースタック

- React SPA を含む AWS CodeCommit リポジトリ
- AWS Amplify ホスティングワークフロー

ターゲットアーキテクチャ

ツール

AWS サービス

- AWS Amplify ホスティングは、継続的なデプロイでフルスタックのサーバーレスウェブアプリケーションをホストするための Git ベースのワークフローを提供します。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

その他のツール

- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。

エピック

「CodeCommit リポジトリを作成」

タスク	説明	必要なスキル
リポジトリを作成します。	手順については、CodeCommit ドキュメントの「AWS CodeCommit リポジトリの作成」を参照してください。	AWS DevOps
リポジトリをクローン作成します。	手順については、CodeCommit ドキュメントの「リポジトリをクローンして CodeCommitリポジトリに接続する」を参照してください。サインインするように求められたら、GitHub の認証情報を入力します。	アプリ開発者

Go アプリケーションを作成します

タスク	説明	必要なスキル

新しいアプリケーションを作成するには	1. 次のコマンドを入力して、クローンされたリポジトリに移動します。<repo name>をCodeCommit リポジトリの名前に置き換えます。
	`$ cd <repo name>`
	2. 以下のコマンドを入力して、クローンされたリポジトリに新しい React アプリケーションを作成します。
	`$ npx create-react-app .`
	3. アプリケーションをコーディングし、次のコマンドを入力して、アプリケーションを起動します。
	`$ npm start`
	カスタム React アプリケーションの作成について詳しくは、「リアクトアプリの作成」ドキュメンテーションの「リアクトアプリの作成」の説明を参照してください。AAmplify ドキュメントの「フロントエンドのデプロイ」の手順に従って、サンプ
		アプリ開発者

タスク	説明	必要なスキル
	ルReactアプリケーションをAAmplify アカウントにデプロイすることもできます。	

ブランチを作成してコードをプッシュします。	1. 次のコマンドを入力して、新しいブランチをローカルに作成します。ここで、<branch>は新しいブランチに割り当てる名前を指定します。
	`$ git checkout -b <branch>`
	2. 次のコマンドを入力して、ブランチを CodeCommit リポジトリにプッシュします。ここで、<branch>は前のステップで割り当てた名前です。詳細については、「コミットを行う」を参照してください。
	`$ git push --set-upstream origin <branch>`
		アプリ開発者

AWS Amplify ホスティングにアプリケーションをデプロイ

タスク	説明	必要なスキル
Amplify をリポジトリConnect。	手順については、Amplify Hosting ドキュメントの「リポジトリのConnect」を参照してください。AWS CodeCommit と、以前に作成	アプリ開発者

タスク	説明	必要なスキル
	したリポジトリとブランチを選択します。	

フロントエンドのビルド設定を定義します。	手順については、Amplify Hosting ドキュメントの「フロントエンドのビルド設定の確認」を参照してください。デフォルトをそのまま使用するか、以下を入力します。
	
	`Build settings:`
	`version: 0.1`
	`frontend:`
	`  phases:`
	`  preBuild:`
	`  commands:`
	`  - npm ci`
	`  build:`
	`  commands:`
	`  - npm run build`
	`  artifacts:`
	`  baseDirectory:`
	`build`
	`  files:`
	`  - '**/*'`
	`  cache:`
	`  paths:`
	`  - node_modules/`
	`**/*`
		アプリ開発者

確認とデプロイ	手順については、Amplify Hosting ドキュメントの「保存とデプロイ」を参照してください。デプロイプロセスが完了するまでお待ちください。	アプリ開発者

継続的デプロイを検証してください。

タスク	説明	必要なスキル
初期デプロイを検証してください。	デプロイプロセスが完了したら、「ドメイン」でリンクを選択します。アプリケーションが想定どおりに動作していることを確認します。	アプリ開発者
変更をコードリポジトリにプッシュします。	ローカルワークステーションでコードを編集し、変更を CodeCommit リポジトリにプッシュします。Amplify Hostingはリポジトリ内の変更を検出し、ビルドとデプロイのプロセスを自動的に開始します。アプリケーションの更新がドメインに表示されていることを確認します。	アプリ開発者

関連リソース

AWS CodeCommit ドキュメント

- AWS CodeCommit 用のセットアップ
  - Git 認証情報を使用した HTTPS ユーザーのセットアップ
  - AWS CLI 認証情報ヘルパーを使用した、Linux、macOS、または UNIX 上の AWS CodeCommit リポジトリへの HTTPS 接続のセットアップ手順
- AWS CodeCommit の開始方法

AWS Amplify ホスティングドキュメント

- 既存のコードを使い始める
- カスタムドメインのセットアップ

React リソース

- React アプリウェブサイトを作成
- React アプリドキュメンテーションの作成
- React アプリケーションリポジトリの作成 (GitHub)

AWS Amplify を使用して React アプリケーションを作成し、Amazon Cognito による認証を追加する

作成者：Rishi Singla (AWS)

概要

このパターンは、AWS Amplify を使用して React ベースのアプリケーションを作成する方法と、Amazon Cognito を使用してフロントエンドに認証を追加する方法を説明しています。AWS Amplify は、AWS でのモバイルアプリとウェブアプリケーションの開発を加速するための一連のツール (オープンソースフレームワーク、ビジュアル開発環境、コンソール) とサービス (ウェブアプリと静的ウェブサイトホスティング) から構成されています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- マシンにインストールされている「Node.js」と「npm」

製品バージョン

- Node.js バージョン 10.x 以降 (バージョンを確認するにはターミナルウィンドウで `node -v` を実行)
- npm バージョン 6.x 以降 (バージョンを確認するにはターミナルウィンドウで `npm -v` を実行)

アーキテクチャ

ターゲットテクノロジースタック

- 「AWS Amplify」
- Amazon Cognito

ツール

- 「Amplify コマンドラインインターフェイス (CLI)」
- 「Amplify ライブラリ」(オープンソースのクライアントライブラリ)

- 「Amplify Studio」(ビジュアルインターフェイス)

エピック

AWS Amplify CLI をインストール

タスク	説明	必要なスキル

Amplify CLI をインストールします。	Amplify CLI は、React アプリケーション用の AWS クラウドサービスを作成するための統合ツールチェーンです。Amplify CLI をインストールするには、実行
	`npm install -g @aws-amplify/cli`
	npm は、新しいメジャーバージョンが利用可能になると通知します。その場合は、次のコマンドを使用して npm のバージョンをアップグレードします。
	`npm install -g npm@9.8.0`
	9.8.0 はインストールしたいバージョンを指します。
		アプリ開発者

React アプリを作成

タスク	説明	必要なスキル

React アプリを作成します。	新しい React アプリを作成するには、コマンドを使用します。
	`npx create-react-app amplify-react-application`
	`ampify-react-application` はアプリの名前です。
	アプリが正常に作成されたら、次のメッセージが表示されます。
	`Success! Created amplify-react-application`
	React アプリ用にさまざまなサブフォルダーを含むディレクトリーが作成されます。
		アプリ開発者

ローカルマシンでアプリを起動します。	前のステップで作成したディレクトリ `amplify-react-application` に移動し、以下のコマンドを実行します。 `amplify-react-application% npm start` ローカルマシンで React アプリが起動します。	アプリ開発者

Amplify CLI の設定

タスク	説明	必要なスキル
AWS アカウントに接続するように Amplify を設定します。	次のコマンドを実行して Amplify を設定します。 `amplify-react-application % amplify configure` Amplify CLI では、次の手順に従い、AWS アカウントへのアクセスを設定するように求められます。 1. AWS 管理者アカウントにサインインします。 2. 使用しようとする AWS リージョンを指定します。 3. プログラムによるアクセスを持つ AWS Identity and Access Management (IAM) ユーザーを作成し、ユーザーにアクセス権限 `AdministratorAccess-Amplify` ポリシーを添付します。 4. [アクセスキー ID] と [シークレットアクセスキーS] を作成し、コピーします。 5. これらの詳細をターミナルに入力します。 6. プロファイル名を作成するか、デフォルトのプロファイルを使用します。	AWS 全般、アプリ開発者

タスク	説明	必要なスキル

	Warning
	このシナリオでは、プログラムによるアクセスと長期的な認証情報を持つ IAM ユーザーが必要です。これはセキュリティリスクをもたらします。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。アクセスキーは、必要に応じて更新できます。詳細については、「IAM ユーザーガイド」の「アクセスキーの更新」を参照してください。
	
	これらのステップは、ターミナルに次のように表示されます。
	
	`Follow these steps to set up access to your AWS account:`
	`Sign in to your AWS administrator account: https://console.aws.amazon.com/`
	`Press Enter to continue`	

タスク	説明	必要なスキル

	<pre>Specify the AWS Region
	? region: us-east-1
	Follow the instructions
	 at
	https://docs.am
	plify.aws/cli/start/
	install/#configure-
	the-amplify-cli
	to complete the user
	 creation in the AWS
	 console
	https://console.aws
	.amazon.com/iamv2/
	home#/users/create
	Press Enter to continue
	Enter the access key of
	 the newly created user:
	? accessKeyId:
	 ********************
	? secretAccessKey:
	 ******************
	******************
	****
	This would update/cr
	eate the AWS Profile in
	 your local machine
	? Profile Name: new
	
	Successfully set up the
	 new user.</pre>
	これらの手順の詳細については、Amplify Dev Center の「ドキュメント」を参照してください。	

Amplify を初期化

タスク	説明	必要なスキル

Amplify を初期化します。	1. 新しいディレクトリで Amplify を初期化するには、以下を実行します。
	`amplify init`
	Amplify はプロジェクト名と設定パラメータの入力を求めます
	2. すべてのパラメーターを指定し、Y を押して、指定した構成でプロジェクトを初期化します。
	`Project information`
	`| Name: amplifyreactproject`
	`| Environment: dev`
	`| Default editor: Visual Studio Code`
	`| App type: javascript`
	`| Javascript framework: react`
	`| Source Directory Path: src`
	`| Distribution Directory Path: build`
		アプリ開発者、AWS 全般

タスク	説明	必要なスキル

	<pre>| Build Command: npm
	 run-script build
	
	| Start Command: npm
	 run-script start</pre>
	3. 前のステップで作成した最初のプライベートサブネットを選択します。リソースは、作成したAmplify プロジェクトの dev 環境にデプロイされます。
	4. リソースが作成されたことを確認するには、「AWS Amplify コンソール」を開き、リソースの作成に使用された AWS CloudFormation テンプレートと詳細を表示します。
	
	<pre>Deploying root stack
	 amplifyreactproject
	 [==================
	==----------------
	----] 2/4
	amplify-amplif
	yreactproject-d…
	 AWS::CloudFormatio
	n::Stack
	 CREATE_IN_PROGRESS
	
	UnauthRole
	 AWS::IAM:
	:Role
	 CREATE_COMPLETE</pre>	

タスク	説明	必要なスキル

	`DeploymentBucket AWS::S3::Bucket CREATE_IN_PROGRESS`
	
	`AuthRole AWS::IAM::Role CREATE_COMPLETE`	

フロントエンドに認証を追加します。

タスク	説明	必要なスキル
認証を追加します。	`amplify add <category>` コマンドで、ユーザーログインやバックエンド API などの機能を追加できます。このステップでは、コマンドで認証を追加します。 Amplify は、Amazon Cognito、フロントエンドライブラリ、およびドロップイン認証システム UI コンポーネントを備えたバックエンド認証サービスを提供します。ユーザーサインアップ、ユーザーサインイン、多要素認証、ユーザーサインアウト、パスワードレスサインインなどの機能があります。Amazon、Google と Facebook などのフェデレーションアイデンティティプロバイダーと統合して	アプリ開発者、AWS 全般

タスク	説明	必要なスキル

	ユーザーアイデンティティを認証することもできます。Amplify 認証カテゴリは、API、分析、ストレージなどの他のAmplify カテゴリとシームレスに統合されるため、認証されたユーザーと認証されていないユーザーとの認可ルールを定義できます。
	
	1. React アプリの認証を設定するには、以下のコマンドを実行します。
	
	`amplify-react-application1 % amplify add auth`
	
	これにより、次の情報とプロンプトが表示されます。ビジネス要件とセキュリティ要件に応じて適切な構成を選択できます。
	
	`Using service: Cognito, provided by: awscloudformation The current configured provider is Amazon Cognito. Do you want to use the default authentication and security configuration? (Use arrow keys)`
	
	`# Default configuration`	

タスク	説明	必要なスキル

	<pre> Default configura
	tion with Social
	 Provider (Federati
	on)
	
	  Manual configura
	tion
	
	  I want to learn
	 more.</pre>
	2. 簡単な例として、デフォルト設定を選択して、ユーザーのログインメカニズム (この場合は電子メール) を選択します。
	
	<pre>How do you want users
	 to be able to sign
	 in?
	 Username
	
	# Email
	
	  Phone Number
	
	  Email or Phone
	 Number
	
	  I want to learn
	 more.</pre>
	3. 詳細設定を省略して、認証リソースの追加を完了します。
	
	<pre>Do you want to
	 configure advanced</pre>	

タスク	説明	必要なスキル
	settings? (Use arrow keys) # No, I am done. Yes, I want to make some additional changes. 4. ローカルバックエンドリソースを構築してクラウドにプロビジョニングします。 amplify-react-application1 % amplify push このコマンドは、お使いのアカウントで Congito ユーザープールに適切な変更を加えます。 5. [Y] を押して、CloudFormationを使用してauth リソースを設定します。 これにより、以下のリソースを設定します。 UserPool AWS::Cognito::UserPool CREATE_COMPLETE UserPoolClientWeb AWS::Cognito::UserPoolClient CREATE_COMPLETE	

タスク	説明	必要なスキル
	UserPoolClientWeb AWS::Cognito::UserPoolClient CREATE_COMPLETE UserPoolClientRole AWS::IAM::Role CREATE_COMPLETE UserPoolClientLambda AWS::Lambda::Function CREATE_COMPLETE UserPoolClientLambdaPolicy AWS::IAM::Policy CREATE_COMPLETE UserPoolClientLogPolicy AWS::IAM::Policy CREATE_IN_PROGRESS 「AWS Cognito コンソール」を使用してこれらのリソースを表示することもできます (Cognito ユーザープールと アイデンティティプールを探してください)。 このステップでは、React アプリの `src` フォルダー内の `aws-exports.js` ファイルを Cognito ユーザープールとアイデンティ	

タスク	説明	必要なスキル
	ティプールの設定で更新します。	

[App.js] ファイルを変更してください。

タスク	説明	必要なスキル
App.js ファイルを変更してください。	この src フォルダーで、App.js ファイルを開いて修正します。変更されたファイルは以下のようになります。 `{ App.Js File after modifications: import React from 'react'; import logo from './logo.svg'; import './App.css'; import { Amplify } from 'aws-amplify'; import { withAuthenticator, Button, Heading } from '@aws-amplify/ui-react'; import awsconfig from './aws-exports'; Amplify.configure(awsconfig); function App({ signOut }) { return (<div> <h1>Thankyou for doing verification</h1> <h2>My Content</h2>`	アプリ開発者

タスク	説明	必要なスキル
	<button onClick={ signOut}>Sign out</ button>); } export default withAuthenticator(App);	
React パッケージをインポートします。	この App.js ファイルは 2 つの React パッケージをインポートされます。次のコマンドを使用してパッケージをインストールします。 amplify-react-appl ication1 % npm install --save aws-amplify @aws-amplify/ui-react	アプリ開発者

React アプリを起動し、認証を確認します。

タスク	説明	必要なスキル
アプリを起動します。	ローカルマシンで React アプリを起動します。 amplify-react-appl ication1 % npm start	アプリ開発者、AWS 全般
認証を確認します。	アプリが認証パラメータの入力を求めるメッセージを表示するかを確認します。(この例では、サインイン方法とし	アプリ開発者、AWS 全般

タスク	説明	必要なスキル
	て電子メールを設定しています)。 フロントエンド UI では、ログイン認証情報の入力が求められ、アカウントを作成するオプションが表示されるはずです。 Amplify ビルドプロセスを設定して、継続的デプロイメントワークフローの一部としてバックエンドを追加することもできます。ただし、このパターンではそのオプションは対象外です。	

関連リソース

- 「開始方法」(npm ドキュメント)
- 「スタンドアロンの AWS アカウントを作成する」(AWS アカウント管理のドキュメント)
- 「AWS Amplify のドキュメント」
- 「Amazon Cognito のドキュメント」

、Angular AWS Amplify、および Module Federation を使用してマイクロフロントエンド用のポータルを作成する

作成者: Milena Godau (AWS) と Pedro Garcia (AWS)

概要

マイクロフロントエンドアーキテクチャを使用すると、複数のチームがフロントエンドアプリケーションのさまざまな部分を個別に作業できます。各チームは、アプリケーションの他の部分に干渉することなく、フロントエンドのフラグメントを開発、構築、デプロイできます。エンドユーザーの観点から見ると、単一のまとまりのあるアプリケーションのように見えます。ただし、さまざまなチームによって公開される複数の独立したアプリケーションとやり取りしています。

このドキュメントでは、、Angular フロントエンドフレームワークAWS Amplify、およびモジュールフェデレーションを使用してマイクロフロントエンドアーキテクチャを作成する方法について説明します。このパターンでは、マイクロフロントエンドはシェル (または親) アプリケーションによってクライアント側で結合されます。シェルアプリケーションは、マイクロフロントエンドを取得、表示、統合するコンテナとして機能します。シェルアプリケーションは、さまざまなマイクロフロントエンドをロードするグローバルルーティングを処理します。@angular-architects/module-federation プラグインは、モジュールフェデレーションを Angular と統合します。を使用して、シェルアプリケーションとマイクロフロントエンドをデプロイします AWS Amplify。エンドユーザーはウェブベースのポータルからアプリケーションにアクセスします。

ポータルは垂直に分割されます。つまり、マイクロフロントエンドはビュー全体またはビューのグループであり、同じビューの一部ではありません。したがって、シェルアプリケーションは一度に 1 つのマイクロフロントエンドのみをロードします。

マイクロフロントエンドはリモートモジュールとして実装されます。シェルアプリケーションは、これらのリモートモジュールを遅延してロードします。これにより、マイクロフロントエンドの初期化が必要になるまで延期されます。このアプローチでは、必要なモジュールのみをロードすることで、アプリケーションのパフォーマンスを最適化します。これにより、初期ロード時間が短縮され、全体的なユーザーエクスペリエンスが向上します。さらに、webpack 設定ファイル (webpack.config.js) を使用してモジュール間で共通の依存関係を共有します。この手法は、コードの再利用を促進し、重複を減らし、バンドルプロセスを合理化します。

前提条件と制限

前提条件

- アクティブな AWS アカウント

- Node.js と npm がインストールされている
- Amplify CLI、インストール済み
- Angular CLI、インストール済み
- を使用するアクセス許可 AWS Amplify
- 「アンギュラーに精通していること」

製品バージョン

- Angular CLI バージョン 13.1.2 以降
- @angular-architects/module-federation バージョン 14.0.1 以降
- webpack バージョン 5.4.0 以降
- AWS Amplify 第 1 世代

制約事項

マイクロフロントエンドアーキテクチャは、スケーラブルで回復力のあるウェブアプリケーションを構築するための強力なアプローチです。ただし、このアプローチを採用する前に、以下の潜在的な課題を理解することが重要です。

- 統合 – 主な課題の 1 つは、モノリシックフロントエンドと比較して複雑さが増す可能性があることです。複数のマイクロフロントエンドのオーケストレーション、マイクロフロントエンド間の通信の処理、共有依存関係の管理がより複雑になる場合があります。さらに、マイクロフロントエンド間の通信に関連するパフォーマンスオーバーヘッドが発生する場合があります。この通信により、レイテンシーが増加し、パフォーマンスが低下する可能性があります。これは、効率的なメッセージングメカニズムとデータ共有戦略を通じて対処する必要があります。
- コードの重複 – 各マイクロフロントエンドは個別に開発されるため、共通機能または共有ライブラリのコードが重複するリスクがあります。これにより、アプリケーション全体のサイズが増加し、メンテナンスの課題が発生する可能性があります。
- 調整と管理 – 複数のマイクロフロントエンドにわたる開発とデプロイのプロセスを調整するのは難しい場合があります。分散アーキテクチャでは、一貫したバージョニングの確保、依存関係の管理、コンポーネント間の互換性の維持がより重要になります。シームレスなコラボレーションとデリバリーには、明確なガバナンス、ガイドライン、自動化されたテストとデプロイパイプラインを確立することが不可欠です。
- テスト — マイクロフロントエンドアーキテクチャのテストは、モノリシックフロントエンドのテストよりも複雑になる場合があります。コンポーネント間の統合テストとend-to-endテストを実行

し、複数のマイクロフロントエンドにわたって一貫したユーザーエクスペリエンスを検証するために、追加の労力と特殊なテスト戦略が必要です。

マイクロフロントエンドアプローチにコミットする前に、マイクロフロントエンドの理解と実装 AWSを確認することをお勧めします。

アーキテクチャ

マイクロフロントエンドアーキテクチャでは、各チームが機能を個別に開発およびデプロイします。次の図は、複数の DevOps チームがどのように連携するかを示しています。ポータルチームがシェルアプリケーションを開発します。シェルアプリケーションはコンテナとして機能します。他の DevOps チームによって発行されたマイクロフロントエンドアプリケーションを取得、表示、統合します。を使用して AWS Amplify 、シェルアプリケーションとマイクロフロントエンドアプリケーションを公開します。

以下は、アーキテクチャ図を示しています。

1. ポータルチームはシェルアプリケーションを開発および維持します。シェルアプリケーションは、ポータル全体を構成するために、マイクロフロントエンドの統合とレンダリングを調整します。
2. チーム A と B は、ポータルに統合されている 1 つ以上のマイクロフロントエンドまたは機能を開発および維持します。各チームは、それぞれのマイクロフロントエンドで個別に作業できます。
3. エンドユーザーは Amazon Cognito を使用して認証します。
4. エンドユーザーがポータルにアクセスし、シェルアプリケーションがロードされます。ユーザーが移動すると、シェルアプリケーションはルーティングを処理し、リクエストされたマイクロフロントエンドを取得し、バンドルをロードします。

ツール

AWS のサービス

- AWS Amplify は、フロントエンドのウェブ開発者とモバイル開発者がフルスタックアプリケーションをすばやく構築するのに役立つ、専用のツールと機能のセットです AWS。このパターンでは、Amplify CLI を使用して Amplify マイクロフロントエンドアプリケーションをデプロイします。

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

その他のツール

- @angular-architects/module-federation は、Angular をモジュールフェデレーションと統合するプラグインです。
- Angular は、モダンでスケーラブル、テスト可能な単一ページアプリケーションを構築するためのオープンソースのウェブアプリケーションフレームワークです。これは、コードの再利用とメンテナンスを促進するモジュール式およびコンポーネントベースのアーキテクチャに従います。
- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計されたイベント駆動型の JavaScript ランタイム環境です。
- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。
- Webpack モジュールフェデレーションは、マイクロフロントエンドやプラグインなど、個別にコンパイルおよびデプロイされたコードをアプリケーションにロードするのに役立ちます。

コードリポジトリ

このパターンのコードは、Angular および Module Federation GitHub リポジトリを使用して Micro-frontend ポータルで使用できます。 GitHub このリポジトリには、次の 2 つのフォルダが含まれています。

- `shell-app` には、シェルアプリケーションのコードが含まれています。
- `feature1-app` には、サンプルのマイクロフロントエンドが含まれています。シェルアプリケーションはこのマイクロフロントエンドを取得し、ポータルアプリケーション内のページとして表示します。

ベストプラクティス

マイクロフロントエンドアーキテクチャには多くの利点がありますが、複雑さも生じます。以下は、スムーズな開発、高品質のコード、優れたユーザーエクスペリエンスのためのベストプラクティスです。

- 計画とコミュニケーション – コラボレーションを合理化するには、事前計画、設計、明確なコミュニケーションチャネルに投資します。

- 設計の一貫性 – 設計システム、スタイルガイド、コンポーネントライブラリを使用して、マイクロフロントエンド全体で一貫したビジュアルスタイルを適用します。これにより、一貫性のあるユーザーエクスペリエンスが提供され、開発が加速されます。
- 依存関係管理 – マイクロフロントエンドは独立して進化するため、標準化された契約とバージョニング戦略を採用して依存関係を効果的に管理し、互換性の問題を防止します。
- マイクロフロントエンドアーキテクチャ – 独立した開発とデプロイを可能にするには、各マイクロフロントエンドにカプセル化された機能に対する明確で明確に定義された責任が必要です。
- 統合と通信 – スムーズな統合を促進し、競合を最小限に抑えるには、APIs、イベント、共有データモデルなど、マイクロフロントエンド間の明確な契約と通信プロトコルを定義します。
- テストと品質保証 — マイクロフロントエンドのテスト自動化と継続的統合パイプラインを実装します。これにより、全体的な品質が向上し、手動テストの労力が削減され、マイクロフロントエンドインタラクション間の機能を検証できます。
- パフォーマンスの最適化 – パフォーマンスメトリクスを継続的にモニタリングし、マイクロフロントエンド間の依存関係を追跡します。これにより、ボトルネックを特定し、最適なアプリケーションパフォーマンスを維持できます。この目的には、パフォーマンスモニタリングと依存関係分析ツールを使用します。
- デベロッパーエクスペリエンス – 明確なドキュメント、ツール、例を提供することで、デベロッパーエクスペリエンスに焦点を当てます。これにより、開発を合理化し、新しいチームメンバーをオンボーディングできます。

エピック

シェルアプリケーションを作成する

タスク	説明	必要なスキル
シェルアプリケーションを作成します。	1. Angular CLI で、次のコマンドを入力します。 `ng new shell --routing` 2. 次のコマンドを入力して、プロジェクトフォルダに移動します。	アプリ開発者

タスク	説明	必要なスキル
	`cd shell` **Note** シェルアプリケーションとマイクロフロントエンドアプリケーションのフォルダとプロジェクト構造は完全に独立している可能性があります。これらは独立した Angular アプリケーションとして処理できます。	
プラグインをインストールします。	Angular CLI で、次のコマンドを入力して @angular-architects/module-federation プラグインをインストールします。 `ng add @angular-architects/module-federation --project shell --port 4200`	アプリ開発者

タスク	説明	必要なスキル
マイクロフロントエンド URL を環境変数として追加します。	1. environment.ts ファイルを開きます。 2. `environment` オブジェクト`mfe1URL: 'http://localhost:5000'` に を追加します。 `export const environment = { production: false, mfe1URL: 'http://localhost:5000', };` 3. environment.ts ファイルを保存して閉じます。	アプリ開発者

タスク	説明	必要なスキル

ルーティングを定義します。	1. app-routing.module.ts ファイルを開きます。
	2. Angular CLI で次のコマンドを入力して、@angular-architects/module-federation プラグインからloadRemoteModule モジュールをインポートします。
	`import { loadRemoteModule } from '@angular-architects/module-federation';`
	3. デフォルトルートを次のように設定します。
	`{ path: '', pathMatch: 'full', redirectTo: 'mfe1' },`
	4. マイクロフロントエンドのルートを設定します。
	`{ path: 'mfe1', loadChildren: () => loadRemoteModule({ type: 'module',`
		アプリ開発者

タスク	説明	必要なスキル
	<pre>remoteEntry: `${environment.mfe1URL}/remoteEntry.js`, exposedModule: './Module' }) .then(m => m.Mfe1Module) },</pre> 5. app-routing.module.ts ファイルを保存して閉じます。	
mfe1 モジュールを宣言します。	1. src フォルダで、decl.d.ts という名前の新しいファイルを作成します。 2. decl.d.ts ファイルを開きます。 3. ファイルに以下を追加します。 <pre>declare module 'mfe1/Module';</pre> 4. decl.d.ts ファイルを保存して閉じます。	アプリ開発者

タスク	説明	必要なスキル
マイクロフロントエンドのプリロードを準備します。	マイクロフロントエンドをプリロードすると、ウェブパックは共有ライブラリとパッケージを適切にネゴシエートできます。 1. main.ts ファイルを開きます。 2. コンテンツを以下に置き換えます。 ``` import { loadRemoteEntry } from '@angular-architects/module-federation'; Promise.all([loadRemoteEntry(`${environment.mfe1URL}/remoteEntry.js`, 'mfe1'),]) .catch(err => console.error('Error loading remote entries', err)) .then(() => import('./bootstrap')) .catch(err => console.error(err)); ``` 3. main.ts ファイルを保存して閉じます。	アプリ開発者

タスク	説明	必要なスキル
HTML コンテンツを調整します。	1. app.component.html ファイルを開きます。 2. コンテンツを以下に置き換えます。 `<h1>Shell application is running!</h1><router-outlet></router-outlet>` 3. app.component.html ファイルを保存して閉じます。	アプリ開発者

マイクロフロントエンドアプリケーションを作成する

タスク	説明	必要なスキル

マイクロフロントエンドを作成します。	1. Angular CLI で、次のコマンドを入力します。
	`ng new mfe1 --routing`
	2. 次のコマンドを入力して、プロジェクトフォルダに移動します。
	`cd mfe1`
		アプリ開発者

プラグインをインストールします。	@angular-architects/module-federation プラグインをインストールするには、次のコマンドを入力します。 `ng add @angular-architects/module-`	アプリ開発者

タスク	説明	必要なスキル
	`federation --project mfe1 --port 5000`	

モジュールとコンポーネントを作成します。	次のコマンドを入力してモジュールとコンポーネントを作成し、リモートエントリモジュールとしてエクスポートします。
	`ng g module mfe1 --routing`
	`ng g c mfe1`
		アプリ開発者

デフォルトのルーティングパスを設定します。	1. mfe-routing.module.ts ファイルを開きます。 2. デフォルトルートを次のように設定します。 `{ path: '', component: Mfe1Component },` 3. mfe-routing.module.ts ファイルを保存して閉じます。	アプリ開発者

タスク	説明	必要なスキル

mfe1 ルートを追加します。	1. app-routing.module.ts ファイルを開きます。
	2. デフォルトルートを次のように設定します。
	`{ path: '', pathMatch: 'full', redirectTo: 'mfe1' },`
	3. 次のmfe1ルートを追加します。
	`{ path: 'mfe1', loadChildren: () => import('./mfe1/mfe1.module').then((m) => m.Mfe1Module), },`
	4. app-routing.module.ts ファイルを保存して閉じます。
		アプリ開発者

タスク	説明	必要なスキル

webpack.config.js ファイルを編集します。	1. webpack.config.js ファイルを開きます。
	2. For remotes セクションを編集して、以下と一致させます。
	
	`// For remotes (please adjust)`
	`name: "mfe1",`
	`filename: "remoteEntry.js",`
	`exposes: {`
	`'./Module': './src/app/mfe1/mfe1.module.ts',`
	`},`
	
	3. shared セクションで、mfe1アプリケーションがシェルアプリケーションと共有している依存関係を追加します。
	
	`shared: share({`
	`"@angular/core": { singleton: true, strictVersion: true, requiredVersion: 'auto' },`
	`"@angular/common": { singleton: true, strictVersion: true, requiredVersion: 'auto' },`
	`"@angular/common/http": { singleton: true, strictVersion: true,`
		アプリ開発者

タスク	説明	必要なスキル
	``` requiredVersion: 'auto' }, "@angular/ router": { singleton : true, strictVer sion: true, requiredV ersion: 'auto' }, ...shared Mappings.getDescri ptors() }) ``` 4. webpack.config.js ファイルを保存して閉じます。	
HTML コンテンツを調整します。	1. app.component.html ファイルを開きます。 2. 内容を以下に置き換えます。 ``` <router-outlet></r outer-outlet> ``` 3. app.component.html ファイルを保存して閉じます。	アプリ開発者

アプリケーションをローカルで実行する

タスク	説明	必要なスキル
mfe1 アプリケーションを実行します。	1. mfe1 アプリケーションを起動するには、次のコマンドを入力します。 ``` npm start ```	アプリ開発者

タスク	説明	必要なスキル
	2. ウェブブラウザで、 にアクセスしますhttp://localhost:5000 。 3. マイクロフロントエンドをスタンドアロンで実行できることを確認します。mfe1アプリケーションはエラーなしで適切にレンダリングする必要があります。	
シェルアプリケーションを実行します。	1. 次のコマンドを入力して、シェルアプリケーションを起動します。 `npm start` 2. ウェブブラウザで、 にアクセスしますhttp://localhost:4200/mfe1 。 3. mfe1 マイクロフロントエンドがシェルアプリケーションに埋め込まれていることを確認します。ポータルアプリケーションはエラーなしで適切にレンダリングされ、mfe1アプリケーションはその中に埋め込まれている必要があります。	アプリ開発者

マイクロフロントエンドロードエラーを処理するためにシェルアプリケーションをリファクタリングする

タスク	説明	必要なスキル

モジュールとコンポーネントを作成します。	シェルアプリケーションのルートフォルダで、次のコマンドを入力して、エラーページのモジュールとコンポーネントを作成します。
	`ng g module error-page --routing`
	`ng g c error-page`
		アプリ開発者

HTML コンテンツを調整します。	1. error-page.component.html ファイルを開きます。 2. 内容を以下に置き換えます。 `<p>Sorry, this page is not available.</p>` 3. error-page.component.html ファイルを保存して閉じます。	アプリ開発者
デフォルトのルーティングパスを設定します。	1. error-page-routing.module.ts ファイルを開きます。 2. デフォルトルートを次のように設定します。 `{ path: '', component: ErrorPageComponent },`	アプリ開発者

タスク	説明	必要なスキル
	3. error-page-routing.module.ts ファイルを保存して閉じます。	

タスク	説明	必要なスキル

マイクロフロントエンドをロードする関数を作成します。	1. app-routing.module.ts ファイルを開きます。
	2. 次の関数を作成します。
	
	`function loadMFE(url: string) { return loadRemoteModule({ type: 'module', remoteEntry: `${url}/remoteEntry.js`, exposedModule: './Module' }) .then(m => m.Mfe1Module) .catch( () => import('./error-page/error-page.module').then(m => m.ErrorPageModule) ); }`
	
	3. mfe1 ルートを次のように変更します。
	
	`{ path: 'mfe1', loadChildren: () => loadMFE(environment.mfe1URL) },`
	
	4. app-routing.module.ts ファイルを保存して閉じます。
		アプリ開発者

タスク	説明	必要なスキル

エラー処理をテストします。	1. まだ実行されていない場合は、次のコマンドを入力してシェルアプリケーションを起動します。
	`npm start`
	2. ウェブブラウザで、 にアクセスしますhttp://localhost:4200/mfe1 。
	3. エラーページがレンダリングされていることを確認します。次のテキストが表示されます。
	`Sorry, this page is not available.`
		アプリ開発者

を使用してアプリケーションをデプロイする AWS Amplify

タスク	説明	必要なスキル
マイクロフロントエンドをデプロイします。	1. Amplify CLI で、マイクロフロントエンドアプリケーションのルートフォルダに移動します。 2. Amplify を初期化するには、次のコマンドを入力します。 `amplify init` 3. Amplify プロジェクトの名前を入力するように求	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	められたら、Enter キーを押します。これにより、package.json ファイルの名前が再利用されます。 4. 上記の設定でプロジェクトを初期化するように求められたら、「」と入力しますYes。 5. 認証方法を選択するように求められたら、 を選択しますAWS Profile。 6. 使用するプロファイルを選択します。 7. Amplify がプロジェクトを初期化するのを待ちます。このプロセスが完了すると、ターミナルに確認メッセージが表示されます。 8. 次のコマンドを入力して、Amplify ホスティングカテゴリをマイクロフロントエンドに追加します。 `amplify add hosting` 9. プラグインモジュールを選択するように求められたら、 を選択しますHosting with Amplify Console。 10. タイプを選択するように求められたら、 を選択しますManual deployment 。	

タスク	説明	必要なスキル

	11次のコマンドを入力して、プロジェクトの npm 依存関係をインストールします。
	`npm install`
	12次のコマンドを入力して、アプリケーションを Amplify コンソールに発行します。
	`amplify publish -y`
	公開が完了すると、Amplify はマイクロフロントエンドの URL を返します。
	13URL をコピーします。シェルアプリケーションを更新するには、この値が必要です。	

タスク	説明	必要なスキル

シェルアプリケーションをデプロイします。	1. src/app/environments フォルダで environment.prod.ts ファイルを開きます。
	2. mfe1URL 値をデプロイされたマイクロフロントエンドの URL に置き換えます。
	`export const environment = { production: true, mfe1URL: 'https://<env>.<Amplify-app-ID>.amplifyapp.com' };`
	3. environment.prod.ts ファイルを保存して閉じます。
	4. Amplify CLI で、シェルアプリケーションのルートフォルダに移動します。
	5. Amplify を初期化するには、次のコマンドを入力します。
	`amplify init`
	6. Amplify プロジェクトの名前を入力するように求められたら、Enter キーを押します。これにより、package.json ファイルの名前が再利用されます。
	7. 上記の設定でプロジェクトを初期化するように求められたら、「」と入力しますYes。
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル

	8. 認証方法を選択するように求められたら、 を選択しますAWS Profile。
	9. 使用するプロファイルを選択します。
	10Amplify がプロジェクトを初期化するのを待ちます。このプロセスが完了すると、ターミナルに確認メッセージが表示されます。
	11Amplify ホスティングカテゴリをシェルアプリケーションに追加します。
	`amplify add hosting`
	12.プラグインモジュールを選択するように求められたら、 を選択しますHosting with Amplify Console。
	13.タイプを選択するように求められたら、 を選択しますManual deployment 。
	14次のコマンドを入力して、プロジェクトの npm 依存関係をインストールします。
	`npm install`
	15次のコマンドを入力して、シェルアプリケーションを Amplify コンソールに発行します。	

タスク	説明	必要なスキル
	`amplify publish -y` 公開が完了すると、Amplify はデプロイされたシェルアプリケーションの URL を返します。 16.シェルアプリケーションの URL を書き留めます。	

タスク	説明	必要なスキル
CORS を有効にします。	シェルアプリケーションとマイクロフロントエンドアプリケーションは異なるドメインで個別にホストされるため、マイクロフロントエンドでクロスオリジンリソース共有 (CORS) を有効にする必要があります。これにより、シェルアプリケーションは別のオリジンからコンテンツをロードできます。CORS を有効にするには、カスタムヘッダーを追加します。 1. Amplify CLI で、マイクロフロントエンドのルートフォルダに移動します。 2. 次のコマンドを入力します。 `amplify configure hosting` 3. カスタム設定を設定するように求められたら、「」と入力しますY。 4. にサインインし AWS Management Console、Amplify コンソールを開きます。 5. マイクロフロントエンドを選択します。 6. ナビゲーションペインで、ホスティングを選択	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル

	し、カスタムヘッダーを選択します。
	7. [編集] を選択します。
	8. カスタムヘッダーの編集ウィンドウで、次のように入力します。
	`customHeaders:`
	`  - pattern: '*.js'`
	`  headers:`
	`  - key: Access-Control-Allow-Origin`
	`  value: '*'`
	`  - key: Access-Control-Allow-Methods`
	`  value: 'GET, OPTIONS'`
	`  - key: Access-Control-Allow-Headers`
	`  value: '*'`
	9. [Save] を選択します。
	10.マイクロフロントエンドを再デプロイして、新しいカスタムヘッダーを適用します。	

タスク	説明	必要なスキル
シェルアプリケーションで書き換えルールを作成します。	Angular シェルアプリケーションは、HTML5 ルーティングを使用するように設定されています。ユーザーがハード更新を実行すると、Amplify は現在の URL からページをロードしようとします。これにより、403 エラーが生成されます。これを回避するには、Amplify コンソールに書き換えルールを追加します。 書き換えルールを作成するには、次の手順に従います。 1. Amplify CLI で、シェルアプリケーションのルートフォルダに移動します。 2. 次のコマンドを入力します。 `amplify configure hosting` 3. カスタム設定を設定するように求められたら、「」と入力しますY。 4. Amplify コンソールを開きます。 5. シェルアプリケーションを選択します。 6. ナビゲーションペインで [ホスティング] を選択し、[書き換えとリダイレクト] を選択します。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	7. [書き換えとリダイレクト] ページで、[リダイレクトの管理] を選択します。 8. テキストエディタを開く を選択します。 9. JSON エディタで、次のリダイレクトを入力します。 `[ { "source": "/<*>", "target": "/index.html", "status": "404-200", "condition": null } ]` 10. [Save] を選択します。	
ウェブポータルをテストします。	1. ウェブブラウザで、デプロイされたシェルアプリケーションの URL を入力します。 2. シェルアプリケーションとマイクロフロントエンドが正しくロードされていることを確認します。	アプリ開発者

リソースをクリーンアップする

タスク	説明	必要なスキル
アプリケーションを削除します。	シェルアプリケーションとマイクロフロントエンドアプリケーションが不要になった場合は、それらを削除します。これにより、使用していないリソースの料金を防ぐことができます。 1. にサインインし AWS Management Console、Amplify コンソールを開きます。 2. マイクロフロントエンドを選択します。 3. ナビゲーションペインで、アプリ設定を選択し、全般設定を選択します。 4. [Delete app] (アプリの削除) を選択します。 5. 確認ウィンドウで、「」と入力しdelete、「アプリの削除」を選択します。 6. これらのステップを繰り返して、シェルアプリケーションを削除します。	AWS 全般

トラブルシューティング

問題	ソリューション
amplify init コマンドの実行時に AWS プロファイルを使用できません	AWS プロファイルを設定していない場合でも、 amplify init コマンドを続行できます。ただし、認証方法の入力を求められたら、 AWS access keysオプションを選択する必要があります。 AWS アクセスキーとシークレットキーを使用可能にします。 または、 の名前付きプロファイルを設定することもできます AWS CLI。手順については、AWS CLI ドキュメントの「設定と認証情報ファイルの設定」を参照してください。
リモートエントリのロード中にエラーが発生しました	シェルアプリケーションの main.ts ファイルにリモートエントリをロードするときにエラーが発生した場合は、environment.mfe1URL 変数が正しく設定されていることを確認してください。この変数の値は、マイクロフロントエンドの URL である必要があります。
マイクロフロントエンドにアクセスするときの 404 エラー	など、ローカルマイクロフロントエンドにアクセスしようとすると 404 エラーが発生した場合はhttp://localhost:4200/mfe1 、以下を確認してください。 • シェルアプリケーションの場合、app-routing.module.ts ファイル内のルーティング設定が正しく設定されていること、およびloadRemoteModule 関数がマイクロフロントエンドを適切に呼び出していることを確認します。 • マイクロフロントエンドの場合、webpack.config.js ファイルexposesの設定が正しいこ

問題	ソリューション
	とを確認し、remoteEntry.js ファイルが正しく生成されていることを確認します。

追加情報

AWS ドキュメント

- でのマイクロフロントエンドの理解と実装 AWS (AWS 規範ガイダンス)
- Amplify CLI (Amplify ドキュメント)
- Amplify ホスティング (Amplify ドキュメント)

その他のリファレンス

- モジュールフェデレーション
- Node.js
- 角度
- @angular-architects/module-federation

React ベースのシングルページアプリケーションを Amazon S3 と CloudFront にデプロイする

作成者: Jean-Baptiste Guillois (AWS)

概要

シングルページアプリケーション (SPA) は、JavaScript API を使用して表示される Web ページのコンテンツを動的に更新する Web サイトまたは Web アプリケーションです。このアプローチでは、Web ページ全体をサーバーからリロードするのではなく、新しいデータのみを更新するため、Web サイトのユーザーエクスペリエンスとパフォーマンスが向上します。

このパターンは、Amazon Simple Storage Service (Amazon S3) と Amazon CloudFront で React に記述された SPA のコーディングとホスティングに段階的にアプローチします。このパターンの SPA は、Amazon API Gateway で設定され、Amazon CloudFront ディストリビューションを介して公開される REST API を使用して、クロスオリジンリソース共有 (CORS) 管理を簡素化します。Amazon CloudFront

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Node.js と npm、インストールおよび設定済み。詳細については、Node.js ドキュメントのダウンロードセクションを参照してください。
- Yarn がインストールされ、設定されている。詳細については、Yarn ドキュメントを参照してください。
- Git、インストールおよび設定済み。詳細については、GitHub ドキュメントを参照してください。

アーキテクチャ

このアーキテクチャは、 AWS CloudFormation (infrastructure as code) を使用して自動的にデプロイされます。Amazon S3 などのリージョンサービスを使用して静的アセットを保存し、Amazon CloudFront と Amazon API Gateway を使用してリージョン API (REST) エンドポイントを公開します。アプリケーションログは Amazon CloudWatch を使用して収集されます。すべての AWS

API コールが監査されます AWS CloudTrail。すべてのセキュリティ設定 (ID やアクセス許可など) は AWS Identity and Access Management (IAM) で管理されます。静的コンテンツは Amazon CloudFront コンテンツ配信ネットワーク (CDN) で配信され、DNS クエリは Amazon Route 53 によって処理されます。

ツール

AWS サービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- Amazon CloudFront は、世界中のデータセンターネットワークを通じて配信することで、ウェブコンテンツの配信を高速化します。これにより、レイテンシーが減少し、パフォーマンスが向上します。
- AWS CloudTrail は、 のガバナンス、コンプライアンス、運用リスクを監査するのに役立ちます AWS アカウント。
- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コード

このパターンのサンプルアプリケーションコードは、GitHub のReact ベースの CORS シングルページアプリケーションのリポジトリにあります。

ベストプラクティス

Amazon S3 オブジェクトストレージを使用すると、アプリケーションの静的アセットを、安全で耐障害性が高く、パフォーマンスが高く、費用対効果の高い方法で保存できます。このタスクに専用

コンテナや Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用する必要はありません。

Amazon CloudFront コンテンツ配信ネットワークを使用すると、ユーザーがアプリケーションにアクセスするときに発生する可能性のあるレイテンシーを減らすことができます。ウェブアプリケーションファイアウォール (AWS WAF) をアタッチして、悪意のある攻撃からアセットを保護することもできます。

エピック

アプリケーションをローカルにビルドおよびデプロイする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	次のコマンドを実行して、サンプルアプリケーションのリポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/react-cors-spa react-cors-spa && cd react-cors-spa`	アプリ開発者、AWS DevOps
アプリケーションをローカルにデプロイします。	1. プロジェクトディレクトリで、`npm install` コマンドを実行してアプリケーションの依存関係を開始します。 2. `yarn dev` コマンドを実行して、アプリケーションをローカルで起動します。	アプリ開発者、AWS DevOps
アプリケーションにローカルでアクセスします。	ブラウザウィンドウを開き、`http://localhost:3000` URL を入力してアプリ	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	ケーションにアクセスします。	

アプリケーションをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation テンプレートをデプロイします。	1. にサインインし AWS Management Console、AWS CloudFormation コンソールを開きます。 2. [Create Stack (スタックの作成)] を選択してから、[With new resources (standard) (新しいリソースを使用 (標準))] を選択します。 3. [Upload a template file (テンプレートファイルをアップロード)] を選択します。 4. [Choose file (ファイルを選択)] を選択し、クローンされたリポジトリから `react-cors-spa-stack.yaml` ファイルを選択してから、[Next(次へ)] を選択します。 5. スタックの名前を入力してから、[Next (次へ)] を選択します。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	6. デフォルトのオプションを保持するには、[Next (次へ)] を選択します。 7. スタックの設定を確認してから、[Create stack (スタックの作成)] を選択します。	
アプリケーションソースファイルをカスタマイズします。	1. スタックがデプロイされたら、出力タブを開き、`Bucket`名前と`APIDomain` 値を特定します。 2. REST API の CloudFront ディストリビューションドメインをコピーします。 3. に移動し`<project_root>/src/pages/index.tsx` 、このドメインを`index.tsx` ファイルの 13 行目の`APIEndPoint` 変数値に挿入するか貼り付けます。	アプリ開発者
アプリケーションパッケージをビルドします。	プロジェクトディレクトリで `yarn build` コマンドを実行して、アプリケーションパッケージをビルドします。	アプリ開発者

タスク	説明	必要なスキル
アプリケーションパッケージをデプロイします。	1. Amazon S3 コンソールを開きます。 2. CloudFormation スタックによって以前に作成された S3 バケットを特定して選択します。 3. [Upload (アップロード)] を選択してから、[Add files (ファイルの追加)] を選択します。 4. `out` フォルダの内容を選択します。 5. フォルダの追加 を選択し、`_next`ディレクトリを選択します。 **⚠ Important** コンテンツではなく `_next`ディレクトリを選択します。 6. [Upload (アップロード)] を選択し、S3 バケットにファイルとディレクトリをアップロードします。	アプリ開発者、AWS DevOps

アプリケーションをテストする

タスク	説明	必要なスキル
アプリケーションにアクセスしてテストします。	ブラウザウィンドウを開き、CloudFront ディストリビューションドメイン (以前に	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	デプロイした CloudFormation スタックからのSPADomain 出力) を貼り付けてアプリケーションにアクセスします。	

リソースをクリーンアップする

タスク	説明	必要なスキル
S3 バケットのコンテンツを削除します。	1. Amazon S3 コンソールを開き、スタックによって以前に作成されたバケット (名前が で始まる最初のバケットreact-cors-spa-) を選択します。 2. [Empty (空)] を選択してバケットのコンテンツを削除します。 3. スタックによって以前に作成された 2 番目のバケット (名前が react-cors-spa- で始まり、-logs で終わる 2 番目のバケット) を選択します。 4. [Empty (空)] を選択してバケットのコンテンツを削除します。	AWS DevOps、アプリケーション開発者
AWS CloudFormation スタックを削除します。	1. AWS CloudFormation コンソールを開き、前に作成したスタックを選択します。	AWS DevOps、アプリケーション開発者

タスク	説明	必要なスキル
	2. 削除を選択し、スタックとすべての関連リソースを削除します。	

関連リソース

ウェブアプリケーションをデプロイしてホストするには、AWS Amplify ホスティングを使用することもできます。ホスティングは、継続的なデプロイでフルスタックのサーバーレスウェブアプリケーションをホストするための Git ベースのワークフローを提供します。Amplify ホスティングは の一部でありAWS Amplify、フロントエンドのウェブおよびモバイル開発者がフルスタックアプリケーションを迅速かつ簡単に構築できるようにする専用のツールと機能を提供します AWS。

追加情報

403 エラーを生成する可能性のあるユーザーによってリクエストされた無効な URLs を処理するために、CloudFront ディストリビューションで設定されたカスタムエラーページは 403 エラーをキャッチし、それらをアプリケーションエントリポイント () にリダイレクトしますindex.html。

CORS の管理を簡素化するために、REST API は CloudFront ディストリビューションを通じて公開されます。

プライベートエンドポイントと Application Load Balancer を使用して、Amazon API Gateway API を内部 Web サイトにデプロイする

作成者: Saurabh Kothari (AWS)

概要

このパターンは、オンプレミスネットワークからアクセスできる内部 Web サイトに Amazon API Gateway API をデプロイする方法を示しています。プライベートエンドポイント、Application Load Balancer、AWS PrivateLink、Amazon Route 53 で設計されたアーキテクチャを使用して、プライベート API のカスタムドメイン名を作成する方法を学習します。このアーキテクチャは、API でのドメインベースのルーティングに役立つカスタムドメイン名とプロキシサーバーの使用の意図しない結果を防ぎます。たとえば、ルーティングできないサブネットに仮想プライベートクラウド (VPC) エンドポイントをデプロイする場合、ネットワークは API ゲートウェイに到達できません。一般的な解決策は、カスタムドメイン名を使用してからルーティング可能なサブネットに API をデプロイすることですが、プロキシ設定でトラフィック (`execute-api.{region}.vpce.amazonaws.com`) が AWS Direct Connect に渡されると、他の内部サイトが機能しなくなる場合があります。最後に、このパターンは、インターネットからアクセスできないプライベート API とカスタムドメイン名を使用するための組織の要件を満たす上で役立ちます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Web サイトと API のサーバー名表示 (SNI) 証明書
- オンプレミス環境から AWS Direct Connect または AWS Site-to-Site VPN を使用して設定された AWS アカウントへの接続
- オンプレミスネットワークから解決され、DNS クエリを Route 53 に転送する、対応するドメイン (domain.com など) を含むプライベートホストゾーン
- オンプレミスネットワークからアクセス可能なルーティング可能なプライベートサブネット

制限

ロードバランサー、ルール、その他のリソースのクォータ (以前は制限と呼ばれてい) の詳細にについては、Elastic Load Balancing ドキュメントの Application Load Balancer のクォータを参照してください。

アーキテクチャ

テクノロジースタック

- Amazon API Gateway
- Amazon Route 53
- Application Load Balancer
- AWS Certificate Manager
- AWS PrivateLink

ターゲットアーキテクチャ

次の図は、Application Load Balancer のリスナールールに基づき、Web トラフィックを Web サイトターゲットグループ、または API ゲートウェイターゲットグループに誘導するApplication Load Balancer を VPC にデプロイする方法を示しています。API ゲートウェイのターゲットグループは、API ゲートウェイの VPC エンドポイントの IP アドレスのリストです。API ゲートウェイは、API をそのリソースポリシーでプライベートにするように設定されています。このポリシーは、特定の VPC エンドポイント以外からの呼び出しをすべて拒否します。API ゲートウェイのカスタムドメイン名は API とそのステージに api.domain.com を使用するように更新されました。Application Load Balancer のルールが追加され、ホスト名に基づきトラフィックがルーティングされます。

この図表は、次のワークフローを示しています：

1. オンプレミスネットワークのユーザーが内部 Web サイトにアクセスします。リクエストは ui.domain.com と api.domain.com に送信されます。次に、リクエストはルーティング可能なプライベートサブネットの内部 Application Load Balancer に解決されます。SSL は ui.domain.com と api.domain.com の Application Load Balancer で終了します。
2. Application Load Balancer に設定されたリスナールールは、ホストヘッダーをチェックします。

   a。ホストヘッダーが api.domain.com の場合、リクエストは API ゲートウェイのターゲットグループに転送されます。Application Load Balancer は、ポート 443 を介して API ゲートウェイへの新しい接続を開始します。

   b。ホストヘッダーが ui.domain.com の場合、リクエストは Web サイトのターゲットグループに転送されます。

3. リクエストが API ゲートウェイに到達すると、API ゲートウェイに設定されたカスタムドメインマッピングによってホスト名と実行する API が決定されます。

自動化とスケール

このパターンの手順は、AWS CloudFormation または AWS Cloud Development Kit (AWS CDK) を使用することで自動化できます。API ゲートウェイコールのターゲットグループを設定するには、カスタムリソースを使用して VPC エンドポイントの IP アドレスを取得する必要があります。describe-vpc-endpoints と describe-network-interfaces への APIコールは IP アドレスとセキュリティグループを返し、これを使用して IP アドレスの API ターゲットグループを作成できます。

ツール

- Amazon API Gateway は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- Amazon Route 53 は、高可用性でスケーラブルな DNS Web サービスです。
- AWS Certificate Manager (ACM) は、 AWS Web サイトとアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書とキーの作成、保存、更新に役立ちます。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS PrivateLink は、VPC から VPC の外部サービスへのプライベート接続の作成に役立ちます。

エピック

SNI 証明書を作成する

タスク	説明	必要なスキル
SNI 証明書を作成し、その証明書を ACM にインポートします。	1. ui.domain.com と api.domain.com 用の SNI 証明書を作成します。詳細については、Amazon CloudFront ドキュメントの CloudFront で HTTPS リクエストを処理する方法の選択 を参照してください。	ネットワーク管理者

タスク	説明	必要なスキル
	2. SNI 証明書を AWS Certificate Manager (ACM) にインポートします。詳細については、AWS ドキュメントの証明書を AWS Certificate Manager にインポートするを参照してください。	

VPC エンドポイントをルーティングできないプライベートサブネットにデプロイする

タスク	説明	必要なスキル
API Gateway にインターフェイス VPC エンドポイントを作成します。	インターフェイス VPC エンドポイントを作成するには、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントのインターフェイス VPC エンドポイントを使用して AWS サービスにアクセスするの指示に従います。	クラウド管理者

Application Load Balancer を設定する

タスク	説明	必要なスキル
アプリケーションのターゲットグループを作成します。	アプリケーションの UI リソースのターゲットグループを作成します。	クラウド管理者
API ゲートウェイエンドポイントのターゲットグループを作成します。	1. IP アドレスタイプでターゲットグループを作成してから、API ゲートウェイエンドポイントの VPC エンドポイントの IP アドレスを	クラウド管理者

タスク	説明	必要なスキル
	ターゲットグループに追加します。 2. 成功コード 403 を使用してターゲットグループのヘルスチェックを設定します。API Gateway の VPC エンドポイントは、ターゲットグループのヘルスチェックによってヘッダーなしで呼び出されたときに 403 コードが返されるため、403 が必要です。	
Application Load Balancer を作成します。	1. ルーティング可能なプライベートサブネットに Application Load Balancer (内部) を作成します。 2. 443 リスナーをApplication Load Balancer に追加し、ACM から証明書を選択します。	クラウド管理者
リスナールールを作成します。	リスナールールを作成して、以下を実行します。 1. ホスト api.domain.com を API ゲートウェイターゲットグループに転送 2. ホスト ui.domain.com を UI リソースのターゲットグループに転送	クラウド管理者

Route 53 を設定する

タスク	説明	必要なスキル
プライベートホストゾーンを作成します。	domain.com のプライベートホストゾーンを作成します。	クラウド管理者
ドメインレコードを作成します。	以下の レコードセットを作成します。 • 値が Application Load Balancer の DNS 名に設定された API • 値が Application Load Balancer の DNS 名に設定された UI	クラウド管理者

API ゲートウェイにプライベート API エンドポイントを作成する

タスク	説明	必要なスキル
プライベート API エンドポイントを作成して設定します。	1. プライベート API エンドポイントを作成するには、API ゲートウェイドキュメントの Amazon API Gateway でプライベート API を作成するの手順に従います。 2. VPC エンドポイントからの API へのコールのみを許可するようにリソースポリシーを設定します。詳細については、API Gateway ドキュメントの API Gateway リソースポリシーで API へ	アプリ開発者、クラウド管理者

タスク	説明	必要なスキル
	のアクセスを制御するを参照してください。	
カスタムドメイン名を作成します。	1. api.domain.com のカスタムドメイン名を作成します。詳細については、API ゲートウェイドキュメントの REST API のカスタムドメイン名のセットアップするを参照してください。 2. 作成した API とステージを選択します。 詳細については、API ゲートウェイドキュメントの REST API の API マッピングを使用するを参照してください。	クラウド管理者

関連リソース

- Amazon API Gateway
- Amazon Route 53
- Application Load Balancer
- AWS PrivateLink
- AWS Certificate Manager

Amazon QuickSight ダッシュボードをローカルの Angular アプリケーションに埋め込みます。

ショーン・グリフィン (AWS) とミレーナ・ゴダウ (AWS) が作成

概要

このパターンは、Amazon QuickSight ダッシュボードをローカルでホストされている Angular アプリケーションに埋め込んで開発またはテストするためのガイダンスを提供します。QuickSight の「埋め込み分析特徴量」は、この機能をネイティブではサポートしていません。既存のダッシュボードと Angular の知識を備えた QuickSight アカウントが必要です。

埋め込まれた QuickSight ダッシュボードを使用する場合、通常、ダッシュボードを表示するには Web サーバー上でアプリケーションをホストする必要があります。これにより、変更内容を Web サーバーに継続的にプッシュして、すべてが正しく動作していることを確認する必要があるため、開発がより困難になります。このパターンは、ローカルでホストされたサーバーを稼働させ、QuickSight の埋め込み分析を使用して開発プロセスをより簡単かつ合理化する方法を示しています。

前提条件と制限

前提条件

- 「Amazon Web Services (AWS) (AWS) アカウント」
- 「セッションキャパシティ料金が設定されたアクティブな QuickSight アカウント」
- 「QuickSight 埋め込み SDK がインストールされました」
- 「Angular CLI がインストールされました」
- 「アンギュラーに精通していること」
- 「mkcert がインストールされました」

機能制限

- このパターンは、ANONYMOUS (公開アクセス可能な) 認証タイプを使用して QuickSight ダッシュボードを埋め込む際のガイダンスを提供します。埋め込みダッシュボードで AWS Identity and Access Management (IAM) または QuickSight 認証を使用している場合は、提供されたコードは適用されません。ただし、「エピック」セクションの Angular アプリケーションをホストする手順は引き続き有効です。

- GetDashboardEmbedUrl API ANONYMOUS をアイデンティティタイプで使用するには、QuickSight キャパシティ料金プランが必要です。

バージョン

- 「アンギュラー CLI バージョン 13.3.4」
- 「QuickSight 埋め込み SDK バージョン 2.3.1」

アーキテクチャ

テクノロジースタック

- アンギュラフロントエンド
- AWS Lambda と Amazon API Gateway バックエンド

アーキテクチャ

このアーキテクチャでは、API ゲートウェイ の HTTP API により、ローカルの Angular アプリケーションが Lambda 関数を呼び出すことができます。Lambda 関数は QuickSight ダッシュボードを埋め込むための URL を返します。

自動化とスケール

AWS CloudFormation または AWS サーバーレスアプリケーションモデル (AWS SAM) を使用してバックエンドデプロイを自動化できます。

ツール

ツール

- 「Angular CLI」は、Angular アプリケーションをコマンドシェルから直接初期化、開発、スキャフォールド、保守するために使用するコマンドラインインターフェイスツールです。
- 「QuickSight 埋め込み SDK」は、QuickSight ダッシュボードを HTML に埋め込むために使用されます。
- 「mkcert」は、ローカルで信頼される開発用証明書を作成するためのシンプルなツールです。設定は不要です。QuickSight ではダッシュボードの埋め込みに HTTPS リクエストしか許可されないため、mkcert が必要です。

AWS サービス

- Amazon API Gateway は、あらゆる規模の REST、HTTP、WebSocket API を作成、公開、管理、モニタリング、保護するための AWS のサービスです。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「Amazon QuickSight」は、ビジュアルの構築、アドホック分析の実行、データからのビジネス上の洞察の取得に使用できるビジネス分析サービスです。

エピック

埋め込み URL を生成

タスク	説明	必要なスキル
EmbedURL ポリシーを作成します。	以下のプロパティを含む QuickSightGetDashboardEmbedUrl という名前の IAM ポリシーを作成します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "quicksight:GetDashboardEmbedUrl", "quicksight:GetAnonymousUserEmbedUrl" ],`	AWS 管理者

タスク	説明	必要なスキル

	<pre>"Resource": "*"
	}
	]
	}</pre>	

タスク	説明	必要なスキル

Lambda 関数を作成します。	1. Lambda コンソールで「関数ページ」を開きます。
	
	2. [関数の作成] を選択します。
	
	3. Author from scratch (製作者を最初から) を選択します。
	
	4. [関数名] に「`get-qs-embed-url` 」と入力します。
	
	5. [Runtime] (ランタイム) では、[Python 3.9] を選択します。
	
	6. [関数の作成] を選択します。
	
	7. コードタブでは、下記のコードを Lambda 関数にコピーします。
	
	`import json`
	`import boto3`
	`from botocore.exceptions import ClientError`
	`import time`
	`from os import environ`
	
	`qs = boto3.client('quicksight',region_name='us-east-1')`
	`sts = boto3.client('sts')`
		アプリ開発者

タスク	説明	必要なスキル

	ACCOUNT_ID = boto3.client('sts').get_caller_identity().get('Account')
	DASHBOARD_ID = environ['DASHBOARD_ID']
	
	def getDashboardURL(accountId, dashboardId, quicksightNamespace, resetDisabled, undoRedoDisabled):
	  try:
	  response = qs.get_dashboard_embed_url(
	  AwsAccountId = accountId,
	  DashboardId = dashboardId,
	  Namespace = quicksightNamespace,
	  IdentityType = 'ANONYMOUS',
	  SessionLifetimeInMinutes = 600,
	  UndoRedoDisabled = undoRedoDisabled,
	  ResetDisabled = resetDisabled
	  )
	  return response
	
	  except ClientError as e:
	  print(e)
	  return "Error generating embeddedURL: " + str(e)	

タスク	説明	必要なスキル

	``
	def lambda_handler(eve
	nt, context):
	  url = getDashbo
	ardURL(ACCOUNT_ID,
	 DASHBOARD_ID,
	 "default", True, True)
	['EmbedUrl']
	  return {
	  'statusCode':
	 200,
	  'url': url
	  }
	``
	8. [デプロイ] を選択します。	

タスク	説明	必要なスキル
ダッシュボード ID を環境変数として追加します。	Lambda 関数に環境変数として、DASHBOARD_ID を追加します。 1. 設定タブで、環境変数、編集、環境変数の追加を選択します。 2. キー DASHBOARD_ID で環境変数を追加します。 3. DASHBOARD_ID の値を取得するには、QuickSight のダッシュボードに移動し、ブラウザの URL の末尾にある UUID をコピーします。たとえば、URL が `https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/<dashboard-id>` の場合、URL <dashboard-id> の一部をキー値として指定します。 4. [Save] を選択します。	アプリ開発者

タスク	説明	必要なスキル
Lambda 関数のアクセス許可を追加します。	Lambda 関数の実行ロールを変更し、それに QuickSightGetDashboardEmbedURL ポリシーを追加します。 1. 設定タブで権限を選択し、ロール名を選択します。 2. ポリシーをアタッチを選択し、`QuicksightGetDashboardEmbedUrl` を検索してチェックボックスを選択し、ポリシーをアタッチを選択します。	アプリ開発者

タスク	説明	必要なスキル
Lambda 関数をテストします。	テストイベントを作成して実行します。この関数はテストイベントのデータを使用しないため、「Hello World」テンプレートを使用できます。 1. [テスト] タブを選択します。 2. テストイベントに名前を付け、Save を選択します。 3. テストを選択して、Lambda 関数をテストします。レスポンスは次のようになります。 `{ "statusCode": 200, "url": "\"https://us-east-1.quicksight.aws.amazon.com/embed/f1acc0786687783b9a4543a05ba929b3a/dashboards/..." }` **Note** 「前提条件と制限」セクションで説明したように、QuickSight アカウントはセッションキャパシティ料金プランの対象である必要があります。そうしない	アプリ開発者

タスク	説明	必要なスキル
	と、エラーメッセージが表示されます。	

タスク	説明	必要なスキル
API ゲートウェイで API を作成します。	1. 「API ゲートウェイコンソール」で API の作成を選択し、REST API、ビルドの順に選択します。 • API 名では、`qs-embed-api` を入力します。 • API の作成 を選択します。 2. アクションでメソッドの作成を選択します。 • GET を選択し、チェックマークを選択して確定します。 • 統合タイプとして、Lambda 関数を選択します。 • Lambda 関数の場合、`get-qs-embed-url` を入力します。 • [Save] を選択します。 • Lambda 関数に権限を追加するダイアログボックスで、 OK を選択します。 3. CORS を有効にします。 • アクションでCORS を有効にするを選択します。 • Access-Control-Allow-Origin では、`'https://my-qs-app.net:4200'` を入力します。	アプリ開発者

タスク	説明	必要なスキル
	• CORS を有効にして既存の CORS ヘッダーを置換を選択して、確認します。 4. API をデプロイします。 • [Actions] (アクション) で [Deploy API] (API のデプロイ) を選択します。 • [デプロイされるステージ] で、[新しいステージ] を選択します。 • [Stage name (ステージ名)] に `dev` と入力します。 • デプロイを選択します。 • 呼び出し URL をコピーします。 **Note** `my-qs-app.net` は任意のドメインにすることができます。別のドメイン名を使用する場合は、ステップ 3 で Access-Control-Allow-Origin 情報を更新し、`my-qs-app.net` 以降の手順で変更してください。	

Angular アプリケーションを作成します。

タスク	説明	必要なスキル

Angular CLI を使用してアプリケーションを作成します。	1. アプリケーションの作成
	`ng new quicksight-app --defaults`
	`cd quicksight-app/src/app`
	2. ダッシュボードコンポーネントを作成します。
	`ng g c dashboard`
	3. `src/environments/environment.ts` ファイルに移動し、`apiUrl: '<Invoke URL from previous steps>'` 環境オブジェクトに追加します。
	`export const environment = { production: false, apiUrl: '<Invoke URL from previous steps>', };`
		アプリ開発者

QuickSight 埋め込み SDK を追加します。	1. プロジェクトのルートフォルダで次のコマンドを実行して QuickSight Embedding SDK をインストールします。	アプリ開発者

タスク	説明	必要なスキル
	``` npm i amazon-qu icksight-embedding- sdk ``` 2. src フォルダで、新しい decl.d.ts ファイルを次の内容でを作成します。 ``` declare module 'amazon-quicksight- embedding-sdk'; ```	

タスク	説明	必要なスキル

ダッシュボード.component.ts ファイルにコードを追加します。	`import { Component, OnInit } from '@angular/core';`
	`import { HttpClient } from '@angular/common/http';`
	`import * as QuicksightEmbedding from 'amazon-quicksight-embedding-sdk';`
	`import { environment } from "../../environments/environment";`
	`import { take } from 'rxjs';`
	`import { EmbeddingContext } from 'amazon-quicksight-embedding-sdk/dist/types';`
	`import { createEmbeddingContext } from 'amazon-quicksight-embedding-sdk';`
	
	`@Component({`
	`  selector: 'app-dashboard',`
	`  templateUrl: './dashboard.component.html',`
	`  styleUrls: ['./dashboard.component.scss']`
	`})`
	`export class DashboardComponent implements OnInit {`
	
	`  constructor(private http: HttpClient) { }`
		アプリ開発者

タスク	説明	必要なスキル

	loadingError = false;
	dashboard: any;
	
	ngOnInit() {
	this.GetDashboardU
	RL();
	}
	
	public GetDashbo
	ardURL() {
	this.http.get(envi
	ronment.apiUrl)
	.pipe(
	take(1),
	)
	.subscribe((data:
	any) => this.Dash
	board(data.url));
	}
	
	public async Dashboard
	(embeddedURL: any) {
	var containerDiv
	= document.getElemen
	tById("dashboardCo
	ntainer") || '';
	const frameOptions
	= {
	url: embeddedURL,
	container:
	containerDiv,
	height: "850px",
	width: "100%",
	resizeHei
	ghtOnSizeChangedEv
	ent: true,
	}
	const embedding
	Context: Embedding
	Context = await
	createEmbeddingCon
	text();	

タスク	説明	必要なスキル

	<pre>this.dashboard = embeddingContext.embedDashboard(frameOptions);
	}
	}</pre>	

ダッシュボード.component.html ファイルにコードを追加します。	以下のコードを src/app/dashboard/dashboard.component.html ファイルに追加します。 <pre> </pre>	アプリ開発者
app.component.html ファイルを変更して、ダッシュボードコンポーネントをロードします。	1. src/app/app.component.html ファイルの内容を削除します。 2. 以下を追加します。 <pre><app-dashboard></app-dashboard></pre>	アプリ開発者
HttpClientModule を app.module.ts ファイルにインポートします。	1. src/app/app.module.ts ファイルの先頭に下記の内容を追加します。 <pre>import { HttpClientModule } from '@angular/common/http';</pre> 2. imports の配列に HttpClientModule を追加してあなたの AppModule とします。	アプリ開発者

Angular アプリケーションをホストします。

タスク	説明	必要なスキル

mkcert を設定します。	**Note**
	次のコマンドは、Unix または MacOS マシン用です。Windows を使用している場合は、対応する echo コマンドの 「追加情報」セクションを参照してください。
	
	1. マシンにローカル認証機関 (CA) を作成します。
	`mkcert -install`
	2. 常にローカル PC `my-qs-app.net` にリダイレクトするように設定します。
	`echo "127.0.0.1 my-qs-app.net" | sudo tee -a /private/etc/hosts`
	3. Angular `src` プロジェクトのディレクトリにいることを確認します。
	`mkcert my-qs-app.net 127.0.0.1`
		アプリ開発者

タスク	説明	必要なスキル
ドメインを許可するように QuickSight を設定します。	1. QuickSight では、右上にある名前を選択し、QuickSight を管理を選択します。 2. ドメインと埋め込みに移動します。 3. `https://my-qs-app.net:4200` を許可されたドメインとして追加。	AWS 管理者
ソリューションをテストします。	次のコマンドを実行して、ローカルの Angular 開発サーバーを開始します。 `ng serve --host my-qs-app.net --port 4200 --ssl --ssl-key "./src/my-qs-app.net-key.pem" --ssl-cert "./src/my-qs-app.net.pem" -o` これにより、以前に作成したカスタム証明書で Secure Sockets Layer (SSL) が有効になります。 ビルドが完了すると、ブラウザウィンドウが開き、Angularでローカルにホストされている埋め込みQuickSightダッシュボードを表示できます。	アプリ開発者

関連リソース

- 「Angular ウェブサイト」

- 「匿名 (未登録) ユーザー向けの QuickSight データダッシュボードの埋め込み」(QuickSight ドキュメント)
- 「QuickSight 埋め込み SDK」
- 「モックサートツール」

追加情報

Windows を使用している場合は、管理者としてコマンドプロンプトウィンドウを実行し、次のコマンドを使用して常にローカル PC `my-qs-app.net` にリダイレクトするように設定します。

```
echo 127.0.0.1 my-qs-app.net >> %WINDIR%\System32\Drivers\Etc\Hosts
```

Green Boost を使用したフルスタックのクラウドネイティブなウェブアプリケーション開発を探索する

作成者: Ben Stickley (AWS)、Amiin Samatar (AWS)

概要

変化する開発者のニーズに対応するため、Amazon Web Services (AWS は、クラウドネイティブなウェブアプリケーションを効率的に開発するための重要なニーズを認識しています。AWS は、AWS クラウドへのウェブアプリケーションのデプロイに関連する一般的な障害を克服できるようサポートすることに重点を置いています。このパターンは、TypeScript、AWS Cloud Development Kit (AWS CDK)、React、Node.js などの最新テクノロジーの機能を活用することで、開発プロセスを合理化および迅速化することを目的としています。

Green Boost (GB) ツールキットを基盤としたこのパターンは、AWS の広範な機能を最大限に活用するウェブアプリケーションをコンストラクトするための実践的なガイドとなります。Amazon Aurora PostgreSQL 互換エディションと統合された基本的な CRUD (作成、読み取り、更新、削除) ウェブアプリケーションをデプロイするプロセスを案内する、包括的なロードマップして機能します。これは、Green Boost コマンドラインインターフェイス (Green Boost CLI) を使用して、ローカル開発環境を確立することによって実現されます。

アプリケーションのデプロイが成功した後、このパターンではインフラストラクチャ設計、バックエンドとフロントエンドの開発、視覚化のための cdk-dia などの基本的なツールを含む、ウェブアプリケーションの主要なコンポーネントを詳細に調査し、効率的なプロジェクト管理を促進します。

前提条件と制限

前提条件

- インストール済みGit
- Visual Studio Code (VS Code) をインストール済み
- AWS コマンドラインインターフェイス (AWS CLI) をインストール済み
- AWS CDK ツールキット をインストール済み
- Node.js 18 をインストール済み、または pnpm を搭載した Node.js 18 がアクティブ
- pnpm をインストール済み (Node.js のインストールに含まれない場合)
- TypeScript、AWS CDK、Node.js、React に関する基本的な知識
- アクティブな AWS アカウント

- us-east-1 の AWS CDK を使用して、ブートストラップされた AWS アカウント。Amazon CloudFront Lambda @Edge 関数をサポートするには us-east-1 AWS リージョンが必要です。
- ターミナル環境における AWS セキュリティ認証情報 (AWS_ACCESS_KEY_ID を含む) の適切な設定
- Windows ユーザーの場合、管理者モードのターミナル (pnpm がノードモジュールを処理する方法に対応)

製品バージョン

- AWS SDK for JavaScript バージョン 3
- AWS CDK バージョン 2
- AWS CLI バージョン 2.2
- Node.js バージョン 18
- React バージョン 18

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL 互換エディション
- Amazon CloudFront
- Amazon CloudWatch
- Amazon Elastic Compute Cloud (Amazon EC2)
- AWS Lambda
- AWS Secrets Manager
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)
- AWS WAF

ターゲット アーキテクチャ

次の図は、ユーザーリクエストが Amazon CloudFront、AWS WAF、AWS Lambda を通過し S3 バケット、Aurora データベース、EC2 インスタンスとやり取りされ、最終的に開発者に届くことを示

しています。一方、管理者は通知とモニタリングの目的で Amazon SNS と Amazon CloudWatch を使用します。

デプロイ後のアプリケーションをより詳しく表示するには、次の例のように cdk-dia を使用して図を作成できます。

これらの図は、ウェブアプリケーションのアーキテクチャを 2 つの異なる視点から示しています。cdk-dia 図では、AWS CDK インフラストラクチャの詳細な技術情報を示しており、Amazon Aurora PostgreSQL 互換や AWS Lambda などの特定の AWS サービスが強調されています。これとは対照的に、もう 1 つの図は、データの論理的な流れとユーザーとのやり取りを強調した、より広い視点を採用しています。主な違いは詳細レベルにあります。cdk-dia は技術的な複雑さについて詳しく調べていますが、最初の図はよりユーザー中心のビューを提供しています。

cdk-dia 図の作成については、「AWS CDK を使用してアプリケーションのインフラストラクチャを理解する」というエピックで説明しています。

ツール

AWS サービス

- Amazon Aurora PostgreSQL 互換エディションは、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援する、フルマネージド型で ACID 互換のリレーショナルデータベースエンジンです。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Amazon CloudFront は、世界中のデータセンターネットワークを通じて配信することで、ウェブコンテンツの配信を高速化します。これにより、レイテンシーが減少し、パフォーマンスが向上します。
- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。このパターンは、AWS Systems Manager Session Manager を使用します。
- Amazon Simple Storage Service (Amazon S3) は、任意の数のデータを保存、保護、検索できるクラウドベースのオブジェクトストレージサービスです。Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーやメールアドレスを含む、パブリッシャーとクライアント間のメッセージの交換を調整して管理するのに役立ちます。
- AWS WAF は、保護されたウェブアプリケーションリソースに転送される HTTP と HTTPS リクエストをモニタリングするのに役立つウェブアプリケーションのファイアウォールです。

その他のツール

- Git はオープンソースの分散バージョンの管理システムです。
- Green Boost は、AWS でウェブアプリケーションを構築するためのツールキットです。
- Next.js は、機能を追加したり最適化したりするための React フレームワークです。
- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- pgAdmin は PostgreSQL 用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。
- pnpm は Node.js プロジェクトの依存関係を管理するパッケージマネージャーです。

ベストプラクティス

以下の推奨事項の詳細については、エピック セクションを参照してください。

- Amazon CloudWatch ダッシュボードとアラームを使用してインフラストラクチャをモニタリングします。

- cdk-nag を使用して静的 Infrastructure as Code (IaC) 分析を実行することで、AWS のベストプラクティスを実施します。
- Systems Manager セッションマネージャを使用して、SSH (Secure Shell) トンネル経由でデータベースポートの転送を確立します。これは、公開 IP アドレスを使用するよりもセキュアです。
- `pnpm audit` を実行して脆弱性を管理します。
- ESLint を使用して静的な TypeScript コード分析を実行し、Prettier を使用してコードフォーマットを標準化することでベストプラクティスを実施します。

エピック

Aurora PostgreSQL 互換の CRUD ウェブアプリケーションのデプロイ

タスク	説明	必要なスキル
Green Boost CLI をインストールします。	Green Boost CLI をインストールするには、次のコマンドを実行します。 `pnpm add -g gboost`	アプリ開発者
GB アプリを作成します。	1. Green Boost を使用してアプリを作成するには、`gboost create` コマンドを実行します。 2. `CRUD App with Aurora PostgreSQL` テンプレートを選択します。	アプリ開発者
依存関係をインストールしてアプリをデプロイします。	1. `cd <your directory>` プロジェクトディレクトリに移動します。 2. `pnpm i` コマンドを使用して、依存関係をインストールします。 3. `cd infra` ディレクトリに移動します。	アプリ開発者

タスク	説明	必要なスキル
	4. アプリをローカルにデプロイするには、pnpm deploy:local コマンドを実行します: これは cdk deploy ... で定義されている infra/package.json コマンドのエイリアスです。 デプロイが完了するまでお待ちください (約 20 分)。待っている間、CloudFormation コンソールで AWS CloudFormation スタックをモニタリングできます。コードマップで定義されたコンストラクトが、デプロイされたリソースにどのようにマッピングされるかに注目してください。CloudFormation コンソールの CDK コンストラクトツリービューを確認してください。	

タスク	説明	必要なスキル
アプリにアクセスします。	GB アプリをローカルにデプロイしたら、CloudFront URL を使用してアクセスできます。URL はターミナル出力に表示されますが、見つけるのは少し難しいかもしれません。URL を素早く見つけるには、次の手順に従います。 1. `pnpm deploy:local` コマンドを実行するターミナルを開きます。 2. ターミナル出力で、次に示すテキストに類似したセクションを探します。 `myapp5stickbui9C39A55A.CloudFrontDomainName = d1q16n5pof924c.cloudfront.net` この URL は、デプロイに対して一意になります。 代わりに、Amazon CloudFront コンソールにアクセスして、Amazon CloudFront URL を確認することもできます。 1. AWS マネジメントコンソールにサインインし、CloudFront サービスに移動します。	アプリ開発者

タスク	説明	必要なスキル
	2. リスト内で、最新のデプロイ済みディストリビューションを確認します。 ディストリビューションに関連付けられたドメイン名をコピーします。`your-unique-id.cloudfront.net` のようになります。	

Amazon CloudWatch を使用してモニタリングする

タスク	説明	必要なスキル
CloudWatch ダッシュボードを表示します。	1. CloudWatch コンソールを開き、[ダッシュボード] を選択します。 2. <appId>-<stageName>-dashboard という名前のダッシュボードを選択します。 3. ダッシュボードを表示します。どのリソースをモニタリングしますか？ どのようなメトリクスが記録されていますか？ このダッシュボードは、オープンソースのコンストラクト cdk-monitoring-constructs によって実装されています。	アプリ開発者
アラートを有効にします。	CloudWatch ダッシュボードは、ウェブアプリケーションをアクティブにモニタリング	アプリ開発者

タスク	説明	必要なスキル

	するのに役立ちます。ウェブアプリを受動的にモニタリングするには、アラートを有効にします。
	
	1. モニタースタックを定義する `/infra/src/app/stateless/monitor-stack.ts` に移動します。
	2. 次の行のコメントを解除し、`admin@example.com` をメールアドレスに置き換えます。
	
	`onAlarmTopic.addSubscription(new EmailSubscription("admin@example.com"));`
	
	3. ファイルの最上部に、以下の情報を追加します。
	
	`import { EmailSubscription } from "aws-cdk-lib/aws-sns-subscriptions";`
	
	4. `infra/` 内で、次のコマンドを実行します。
	
	`cdk deploy "*/monitor" --exclusively.`
	
	5. モニタリングアラームの発生時に開始する SNS トピックへの登録を確認する	

タスク	説明	必要なスキル
	には、メールメッセージ内のリンクを選択します。	

AWS CDK を使用してアプリのインフラストラクチャを理解する

タスク	説明	必要なスキル
アーキテクチャ図を作成します。	cdk-dia を使用して、ウェブアプリのアーキテクチャ図を生成します。アーキテクチャを視覚化することで、チームメンバー間の理解とコミュニケーションが向上します。システムコンポーネントとその関係の明確な概要が示されています。 1. Graphviz をインストールします。 2. `infra/` 内で、`pnpm cdk-dia` コマンドを実行します。 3. `infra/diagram.png` を表示します。	アプリ開発者
cdk-nag を使用してベストプラクティスを実施します。	cdk-nag を使用すると、ベストプラクティスを実施し、セキュリティの脆弱性や設定ミスのリスクを軽減することで、セキュアでコンプライアンスを確保したインフラストラクチャを維持できます。 1. cdk-nag のベストプラクティスの実施については	アプリ開発者

タスク	説明	必要なスキル
	、AWS ソリューションライブラリのルールパックにあるチェック項目を含む、「ルール」セクションをご覧ください。 2. cdk-nag がどのようにルールを実行するかを確認するには、コードを変更します。例えば、`infra/src/app/stateful/data-stacks.ts` で `storageEncrypted: true` を `storageEncrypted: false` に変更します。 3. `infra/` 内で、`cdk synth "*/data"` コマンドを実行します。合成中に、ルール違反を意味するビルドエラーが発生します。 `AwsSolutions-RDS2: The RDS instance or Aurora DB cluster does not have storage encryption enabled.` このエラーは、cdk-nag がどのようにインフラストラクチャーのベストプラクティスを実施し、セキュリティ上の設定ミスを防ぐためのセキュリティメカニズ	

タスク	説明	必要なスキル

	ムであるかを示しています。
	4. 必要に応じて、異なる範囲でルールを抑制することもできます。例えば、AWSSolutions-RDS2 を抑制するには、DbIamCluster のインスタンス化の下に次のコードを追加します。
	`NagSuppressions.addResourceSuppressions(`
	`  cluster.node.findChild("Resource"),`
	`  [`
	`  {`
	`  id: "AwsSolutions-RDS2",`
	`  reason: "Customer requirement necessitates having unencrypted DB storage",`
	`  },`
	`  ],`
	`);`
	5. 抑制した後、もう一度 cdk synth "*/data" を実行します。これで、AWS CDK アプリが正常に合成されます。抑制されたルールはすべて infra/cdk.out/assembly-<appId>-<stageName>/AwsSolutions-<appId>	

タスク	説明	必要なスキル
	-<stageName>-${stackId}-NagReport.csv で確認できます。	

データベース設定とスキーマを評価する

タスク	説明	必要なスキル
環境変数を追加します。	必要な環境変数を取得するには、以下の手順に従います。 1. DB_BASTION_ID を検索するには、コンソールにサインインし、EC2 コンソールに移動します。[インスタンス (実行中)] を選択し、<stageName>-ssm-db-bastion 名が含まれている行を探します。インスタンス ID は i- で始まります。 2. DB_ENDPOINT を検索するには、Amazon Relational Database Service (Amazon RDS) コンソールで DB インスタンスを選択し、DB 識別子が <appId>-<stageName>-data- で始まるリージョンクラスターを選択します。rds.amazonaws.com で終わるライターインスタンスのエンドポイントを探します。	アプリ開発者
ポート転送を確立します。	ポート転送を確立するには、以下の手順に従います。	アプリ開発者

タスク	説明	必要なスキル
	1. AWS Systems Manager Session Manager プラグインをインストールします。 2. `core/` で `pnpm db:connect` を実行してポート転送を開始し、踏み台ホストを介してセキュアな接続を確立します。 3. ターミナルに `Waiting for connections...` テキストが表示されると、EC2 踏み台ホストを経由してローカルマシンと Aurora サーバーの間に SSH トンネルが正常に確立されています。	
Systems Manager Session Manager のタイムアウトを調整します。	(オプション) デフォルトの 20 分のセッションタイムアウトが短すぎる場合は、Systems Manager コンソールで [セッションマネージャ]、[設定]、[編集]、[アイドルセッションタイムアウト] の順に選択して、最大 60 分まで増やすことができます。	アプリ開発者

タスク	説明	必要なスキル
データベースを視覚化します。	pgAdmin は、PostgreSQL データベースを管理するための使いやすいオープンソースツールです。データベースタスクを簡素化し、データベースを効率的に作成、管理、最適化できるようにします。 このセクションでは、pgAdmin のインストールから、その機能を PostgreSQL データベース管理に使用する方法について説明します。 1. オブジェクトエクスプローラで、サーバーのコンテキスト (右クリック) メニューを開いてから、[登録]、[サーバー] を選択します。 2. [全般] タブの [名前] フィールドに <appId><stageName> と入力します。 3. DB パスワードを取得するには、AWS Secrets Manager コンソールを開き、<appId>-<stageName>-data スタックの CDK によって生成と記述されたシークレットを選択し、[シークレット値] カードを選択します。[シークレット値の取得] を選択し、[シークレット値] と [パスワード] のキーをコピーします。	アプリ開発者

タスク	説明	必要なスキル
	4. [接続] タブで、[ホスト名/アドレス] フィールドに 0.0.0 と入力し、[ユーザー名] フィールドに <appId>_admin と入力します。[パスワード] フィールドには、以前に取得したシークレットを使用します。[パスワードを保存?] フィールド で [はい] を選択します。 5. [保存] を選択します。 6. テーブルを表示するには、<appId>-<stageName>、[データベース]、[<appId>_db]、[スキーマ]、<appId>、[テーブル] に移動します。 7. [アイテム] テーブルのコンテキスト (右クリック) メニューを開き、[データの表示/編集]、[すべての行] を選択します。 8. テーブルを調べます。	

Node.js でデバッグする

タスク	説明	必要なスキル
アイテム作成のユースケースをデバッグします。	アイテム作成のユースケースをデバッグするには、次の手順に従います。 1. `core/src/modules/item/create-item.us`	アプリ開発者

タスク	説明	必要なスキル

	e-case.ts ファイルを開き、次のコードを挿入します。
	
	``
	import { fileURLTo
	Path } from "node:url
	";
	
	// existing create-it
	em.use-case.ts code
	 here
	
	if (process.argv[1]
	 === fileURLTo
	Path(import.meta.u
	rl)) {
	  createItemUseCase(
	{
	    description:
	 "Item 1's Descripti
	on",
	    name: "Item 1",
	  });
	}
	``
	
	2. 前の手順で追加したコードにより、モジュールを直接実行するときに createItemUseCase 関数が呼び出されるようになりました。行ごとのデバッグを開始するコードブロック内で、行にブレークポイントを設定します。
	
	1. VS Code JavaScript デバッグターミナルを開き、pnpm tsx core/	

タスク	説明	必要なスキル
	`src/modules/item/create-item.use-case.ts` を実行して、行ごとのデバッグでコードを実行します。別の方法として、`console.log` ステートメントを使用することもできますが、複雑なビジネスロジックを処理する場合は print ステートメントでは不十分な場合があります。行ごとのデバッグを実行すると、より多くのコンテキストが得られます。	

フロントエンドを開発する

タスク	説明	必要なスキル
開発サーバーを設定します。	1. `ui/` に移動し、`pnpm dev` を実行して Next.js 開発サーバーを起動します。 2. `http://localhost:3000` で、ローカルにウェブアプリケーションにアクセスします。Next.js 開発サーバーは、React コンポーネントに加えられた編集に関するフィードバックを高速更新で瞬時に返すように設定されています。 3. アプリバーの色をカスタマイズします。`ui/src/components/theme/`	アプリ開発者

タスク	説明	必要なスキル
	theme.tsx ファイルを開き、アプリバーのテーマを定義している部分を見つけます。[colorSchemes.light.palette.primary] セクションで、メイン値を colors.lagoon から colors.carrot に更新します。この変更を追加した後、ファイルを保存し、ブラウザで更新内容を観察します。 4. テキストやコンポーネントを変更したり、新しいページを追加するなど、いろいろ試してください。	

Green Boost を使用したツーリング

タスク	説明	必要なスキル
monorepo と pnpm パッケージマネージャーを設定します。	1. GB リポジトリのルートの下にある pnpm-workspace.yaml を確認し、ワークスペースがどのように定義されるかに注目します。ワークスペースの詳細については、pnpm のドキュメントを参照してください。 2. ui/package.json を確認し、 "<appId>/core": "workspace:^", というパッケージ	アプリ開発者

タスク	説明	必要なスキル
	名の core/ 内でどのようにワークスペースを参照しているかに注目します。 3. TypeScript と ESLint の設定が、packages/ 内で定義されているユーティリティパッケージにどのように一元化されているか観察します。その後、core/ infra/ ui/ などのアプリケーションパッケージがこの設定を使用します。これは、アプリをスケールする場合や、設定コードを複製せずにユーティリティパッケージを参照できるアプリケーションパッケージをさらに定義する場合に役立ちます。	

タスク	説明	必要なスキル
pnpm スクリプトを実行します。	リポジトリのルートで以下のコマンドを実行します。 1. `pnpm lint` を実行します。このコマンドは ESLint を使用して静的コード分析を実行します。 2. `pnpm typecheck` を実行します。このコマンドは、TypeScript コンパイラを実行してコードのタイプをチェックします。 3. `pnpm test` を実行します。このコマンドは Vitest を実行してユニットテストを実行します。 これらのコマンドが、すべてのワークスペースでどのように実行されるかに注目してください。これらのコマンドは、各ワークスペースの `package.json#scripts` フィールドで定義されます。	アプリ開発者

タスク	説明	必要なスキル
ESLint を使用して静的コード分析を行います。	ESLint の静的コード分析機能をテストするには、以下の作業を行います。 1. まず、VS Code ESLint エクステンション (ID: `dbaeumer.vscode-eslint`) がインストールされていることを確認します。エラーをインラインで確認するには、VS Code Error Lens (ID:`usernamehw.errorlens`) もインストールすることをお勧めします。 2. コードには、次の例に示すように、`eval()` 関数を使用する 1 行のコードが格納されています。 `const userInput = "import("fs").then((fs) => console.log(fs.readFileSync("/etc/passwd", { encoding: "utf8" })))"; eval(userInput);` **Important** これはテストのみを目的としています。`eval()` の使用は潜在的に危険であると考えられており、セ	アプリ開発者

タスク	説明	必要なスキル
	キュリティ上のリスクがあるため避けるべきです。 3. eval() 行を追加したら、コードエディタを開き、ESLint がコードの臭いを示す赤い破線を使用していることを確認します。 4. ESLint プラグインと設定については、packages/eslint-config-{node,next}/.eslintrc.cjs を参照してください。	

タスク	説明	必要なスキル
依存関係と脆弱性を管理します。	1. 共通脆弱性識別子 (CVE) を特定するには、リポジトリのルートで `pnpm audit` を実行します。 「既知の脆弱性は見つかりませんでした」と、表示されるはずです。 2. `pnpm add minimist@0.2.3` を実行して意図的に攻撃されやすいパッケージを `core/` にインストールし、`pnpm audit` を実行します。報告されている脆弱性に注目してください。 3. `pnpm remove minimist` を実行して、`core/` で脆弱なパッケージをアンインストールします。	アプリ開発者

タスク	説明	必要なスキル
Husky を使用してフックをプリコミットします。	1. リポジトリ全体の TypeScript ファイルにいくつかの小さな変更を追加します。この変更は、コメントの追加のように基本的な内容でもかまいません。 2. `git add -A` と `git commit -m "test husky"` を使用してこれらの変更をステージングしてコミットします。 `.husky/pre-commit` で定義されている Husky プリコミット フックトリガーは、コマンド `pnpm lint-staged` を実行します。 3. lint-staged が、Git `*/.lintstagedrc.js` によってステージングされたファイルに対して、リポジトリ全体のファイルで指定されたコマンドを実行する方法を確認してください。 これらのツールは、不正なコードがアプリケーションに侵入するのを防止するためのメカニズムです。	アプリ開発者

インフラストラクチャをティアダウンする

タスク	説明	必要なスキル
アカウントからデプロイを削除します。	1. `infra/` で `pnpm destroy:local` を実行して、最初のエピックでプロビジョニングされたインフラストラクチャをティアダウンします。 2. `pnpm destroy:local` が完了したら 15 分間待ち、Lambda コンソールでアプリケーション ID を検索して、予約されている Lambda@Edge 関数を削除します。Lambda @Edge 関数は複製されるため、削除が困難です。Lambda @Edge 関数の削除の詳細については、CloudFront のドキュメントを参照してください。	アプリ開発者

トラブルシューティング

問題	ソリューション
ポート転送を確立できない	AWS 認証情報が正しく設定され、必要な権限があることを確認してください。 踏み台ホスト ID (`DB_BASTION_ID`) とデータベースエンドポイント (`DB_ENDPOINT`) の環境変数が正しく設定されていることを再確認してください。

問題	ソリューション
	それでも問題が解決しない場合は、SSH 接続と Session Manager のトラブルシューティングに関する AWS ドキュメントを参照してください。
`localhost:3000` で、ウェブサイトが読み込まれない	ターミナルの出力に、転送アドレスを含めてポート転送が成功したことが示されていることを確認します。 ローカルマシンでポート 3000 を使用しているプロセスが競合していないことを確認します。 Green Boost アプリケーションが正しく設定され、想定されるポート (3000) で実行されていることを確認します。 ウェブブラウザで、ローカル接続をブロックする可能性のあるセキュリティ拡張機能や設定がないか確認してください。
ローカルデプロイ中のエラーメッセージ (`pnpm deploy:local`)	エラー メッセージを注意深く確認して、問題の原因を特定します。 必要な環境変数と設定ファイルが正しく設定されていることを確認します。

関連リソース

- AWS SDK ドキュメント
- Green Boost ドキュメント
- Next.js ドキュメント
- Node.js ドキュメント
- React ドキュメント
- TypeScript ドキュメント

AWS CodeBuild を使用して GitHub から Node.js アプリケーションのユニットテストを実行する

作成者: Thomas Scott (AWS)、Jean-Baptiste Guillois (AWS)

概要

このパターンは、Node.js ゲーム API のサンプルソースコードと主要ユニットテストコンポーネントを提供します。また、継続的インテグレーションおよび継続的デリバリー (CI/CD) ワークフローの一環として、AWS CodeBuild を使用した、GitHub リポジトリーからのこれらのユニットテストの実行方法も含まれます。

ユニットテストは、ユニットと呼ばれるアプリケーションのさまざまな部分が個別に独立してテストされ、正しく動作するかどうかがテストされるソフトウェア開発プロセスです。テストはコードの品質を検証し、期待どおりに機能することを確認します。テストを参考にすれば、他の開発者も簡単にコードベースに慣れることができます。ユニットテストは、将来のリファクタリング時間を短縮し、エンジニアがコードベースに素早く慣れるようにする上で役立ち、期待される動作に自信を与えます。

ユニットテストでは、AWS Lambda 関数など、個々の関数をテストします。ユニットテストを作成するには、テストフレームワークとテスト (アサーション) を検証する方法が必要です。このパターンのコード例では、Mocha テストフレームワークと Chai アサーションライブラリを使用します。

ユニットテストとテストコンポーネントの例の詳細については、「追加情報」セクションを参照してください。

前提条件と制限

- 正しい CodeBuild 権限があるアクティブな AWS アカウント
- GitHub アカウント (「サインアップの手順」を参照)
- Git (「インストール手順」を参照)
- 変更を行い、コードを GitHub にプッシュするためのコードエディタ

アーキテクチャ

このパターンは、次の図に示すアーキテクチャを実装します。

ツール

ツール

- Git は、コード開発に使用できるバージョン管理システムです。
- AWS CodeBuild は、ソースコードをコンパイルし、テストを実行し、デプロイ可能なソフトウェアパッケージを生成するフルマネージドの継続的統合サービスです。CodeBuild を使用すると、ビルドサーバーのプロビジョニング、管理、スケーリングが不要になります。CodeBuild では、継続的にスケーリングされ、複数のビルドが同時にプロセスされるため、ビルドの実行までキューで待機することはありません。パッケージ済みのビルド環境を使用、またはご自分のビルドツールを使用するカスタムビルド環境を作成できることですぐに開始できます。CodeBuild では毎分ごとに使用するコンピューティングリソースの使用量が有料になります。

コード

このパターンのソースコードは、GitHub のサンプルゲームユニットテストアプリケーションリポジトリにあります。このサンプルから独自の GitHub リポジトリを作成 (オプション 1)、またはこのパターンのサンプルリポジトリを直接使用 (オプション 2) できます。次のセクションの各オプションの指示に従います。従うオプションは、ユースケースによって異なります。

エピック

オプション 1 - CodeBuild を使用して個人用 GitHub リポジトリでユニットテストを実行

タスク	説明	必要なスキル
サンプルプロジェクトに基づいて独自の GitHub リポジトリを作成します。	1. GitHub にログインします。 2. 新しいレポジトリを作成します。手順については、GitHub のドキュメントを参照してください。 3. サンプルリポジトリをクローンして、アカウントの新しいリポジトリにプッシュします。	アプリ開発者、AWS 管理者、AWS DevOps
新しい CodeBuild プロジェクトを作成します。	1. AWS マネジメントコンソールにサインイン	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	し、https://console.aws.amazon.com/codesuite/codebuild/home://www.com で CodeBuild コンソールを開きます。 2. Create build project (ビルドプロジェクトの作成)を選択します。 3. プロジェクト設定セクションのプロジェクト名で、aws-tests-sample-node-js と入力します。 4. ソースセクションの ソースプロバイダで、 [GitHub] を選択します。 5. リポジトリでは、[Repository in my GitHub account (自分の GitHub アカウントのリポジトリ)] を選択し、URL を新しく作成した GitHub リポジトリに貼り付けます。 6. プライマリソースの Webbook イベントセクションで、[Rebuild every time a code change is pushed to this repository (コードの変更がこのリポジトリにプッシュされるたびに再構築する) をオンにします。 7. イベントタイプで、[PUSH(プッシュ)] を選択します。	

タスク	説明	必要なスキル
	8. 環境セクションで、マネージドイメージ、Amazon Linux、および最新のイメージを選択します。 9. 他のすべてのオプションはデフォルト設定のままにしてから、[Create build project (ビルドプロジェクトの作成)] を選択します。	
ビルドを開始します。	確認 ページで、[Start build (ビルドの開始)] を選択してビルドを実行します。	アプリ開発者、AWS 管理者、AWS DevOps

オプション 2 - CodeBuild を使用してパブリック GitHub リポジトリでユニットテストを実行

タスク	説明	必要なスキル
新しい CodeBuild ビルドプロジェクトを作成します。	1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/codesuite/codebuild/home://www.com で CodeBuild コンソールを開きます。 2. Create build project (ビルドプロジェクトの作成)を選択します。 3. プロジェクト設定セクションのプロジェクト名で、aws-tests-sample-node-js と入力します。	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	4. ソースセクションの ソースプロバイダで、 [GitHub] を選択します。 5. リポジトリには [Public repository (パブリックリポジトリ)] を選択してから、URL: https://github.com/aws-samples/node-js-tests-sample を貼り付けます。 6. 環境セクションで、マネージドイメージ、Amazon Linux、および最新のイメージを選択します。 7. 他のすべてのオプションはデフォルト設定のままにしてから、[Create build project (ビルドプロジェクトの作成)] を選択します。	
ビルドを開始します。	確認 ページで、[Start build (ビルドの開始)] を選択してビルドを実行します。	アプリ開発者、AWS 管理者、AWS DevOps

ユニットテストの分析

タスク	説明	必要なスキル
テスト結果を表示します。	CodeBuild コンソールで、CodeBuild ジョブのユニットテスト結果を確認します。結果は追加情報セクションに表示されている結果と一致するはずです。	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	これらの結果は、GitHub リポジトリと CodeBuild の統合を検証します。	
Webbook を適用します。	これで Webhook を適用できるようになったため、コードの変更をリポジトリのメインブランチにプッシュするたびに自動的にビルドを開始できます。手順については、CodeBuild ドキュメントを参照してください。	アプリ開発者、AWS 管理者、AWS DevOps

関連リソース

- サンプルゲームユニットテストアプリケーション(サンプルコードを含む GitHub リポジトリ)
- AWS CodeBuild ドキュメント
- GitHub Webbook イベント(CodeBuild ドキュメント)
- 新しいリポジトリの作成(GitHub ドキュメント)

追加情報

ユニットテスト結果

プロジェクトが正常にビルドされると、CodeBuild コンソールに次のテスト結果が表示されるはずです。

ユニットテストコンポーネントの例

このセクションでは、ユニットテストで使用される 4 種類のテストコンポーネント (アサーション、スパイ、スタブ、モック) について説明します。各コンポーネントの簡単な説明とコード例が含まれています。

アサーション

アサーションは期待される結果の検証に使用されます。これは特定の関数からの期待される応答を検証するため、重要なテストコンポーネントです。次のサンプルアサーションは、新しいゲームを初期化するときに、返される ID が 0 から 1000 の間であることを検証します。

```
const { expect }  = require('chai');
const { Game } = require('../src/index');

describe('Game Function Group', () => {
 it('Check that the Game ID is between 0 and 1000', function() {
      const game = new Game();
      expect(game.id).is.above(0).but.below(1000)
 });
});
```

スパイ

スパイは関数を実行中に何が起きているかの観察に使用されます。たとえば、関数が正しく呼び出されたことを確認する必要があります。次の例は、Game クラスオブジェクトで開始メソッドと停止メソッドが呼び出されることを示しています。

```
const { expect }  = require('chai');
const { spy } = require('sinon');

const { Game } = require('../src/index');

describe('Game Function Group', () => {
   it('should verify that the correct function is called', () => {
      const spyStart = spy(Game.prototype, "start");
      const spyStop = spy(Game.prototype, "stop");

      const game = new Game();
      game.start();
      game.stop();

      expect(spyStart.called).to.be.true
      expect(spyStop.called).to.be.true
    });
});
```

スタブ

スタブは関数のデフォルトレスポンスのオーバーライドに使用されます。これは関数が外部リクエストをする場合に特に便利です。ユニットテストから外部リクエストをしないようにするためです。(外部リクエストは、異なるコンポーネント間のリクエストを物理的にテストできる統合テストにより適しています)。次の例では、スタブが getId 関数からのリターン ID を強制します。

```
const { expect }  = require('chai');
const {.stub } = require('sinon');

const { Game } = require('../src/index');

describe('Game Function Group', () =>  {
   it('Check that the Game ID is between 0 and 1000', function() {
      let generateIdStub = stub(Game.prototype, 'getId').returns(999999);

      const game = new Game();

      expect(game.getId).is.equal(999999);

      generateIdStub.restore();
    });
});
```

モック

モックは、さまざまなシナリオをテストする動作があらかじめプログラムされたフェイクメソッドです。モックはスタブの拡張版と見なすことができ、複数のタスクを同時に実行できます。以下の例では、モックを使用して 3 つのシナリオを検証します。

- 関数の呼び出し
- 引数による関数の呼び出し
- 関数は整数 9 を返す

```
const { expect }  = require('chai');
const {.mock } = require('sinon');

const { Game } = require('../src/index');

describe('Game Function Group', () =>  {
   it('Check that the Game ID is between 0 and 1000', function() {
      let mock = mock(Game.prototype).expects('getId').withArgs().returns(9);
```

```
      const game = new Game();
      const id = get.getId();

      mock.verify();
      expect(id).is.equal(9);
    });
});
```

AWS Lambda を使用して六角形アーキテクチャで Python プロジェクトを構築する

作成者: Furkan Oruc (AWS)、Dominik Goby (AWS)、Darius Kunce (AWS)、および Michal Ploski (AWS)

概要

このパターンは、AWS Lambda を使用して Python プロジェクトを六角形アーキテクチャで構築する方法を示しています。このパターンでは、Infrastructure as Code (IaC) ツールとして AWS Cloud Development Kit (AWS CDK)、REST API として Amazon API Gateway、パーシスタンスレイヤーとして Amazon DynamoDB を使用しています。六角形アーキテクチャは、ドメイン主導型の設計原則に従っています。六角形アーキテクチャでは、ソフトウェアはドメイン、ポート、アダプターの 3 つのコンポーネントで構成されます。六角形アーキテクチャとその利点の詳細については、「 で六角形アーキテクチャを構築する」ガイドを参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Python の使用経験
- AWS Lambda、AWS CDK、Amazon API Gateway、DynamoDB に精通していること
- GitHub アカウント (「サインアップの手順」を参照)
- Git (「インストール手順」を参照)
- 変更を行い、コードを GitHub にプッシュするためのコードエディタ (Visual Studio Code や JetBrains PyCharm など)
- Docker がインストールされ、Docker デーモンが起動して実行されていること

製品バージョン

- Git バージョン 2.24.3 以降
- Python バージョン 3.7 以降。
- AWS CDK v2
- Poetry バージョン 1.1.13 以降

- Python バージョン 1.25.6 以降向け AWS Lambda Powertools
- pytest バージョン 7.1.1 以降
- Moto バージョン 3.1.9 以降
- pPydantic バージョン 1.9.0 以降
- Boto3 バージョン 1.22.4 以降
- mypy-boto3-dynamodb バージョン 1.24.0 以降

アーキテクチャ

ターゲットテクノロジースタック

ターゲットテクノロジースタックは、API Gateway、Lambda、および DynamoDB を使用する Python サービスで構成されます。このサービスは DynamoDB アダプターを使用してデータを保持します。エントリポイントとして Lambda を使用する関数を提供します。このサービスは Amazon API Gateway を使用して REST API を公開します。API は AWS Identity and Access Management (IAM) を使用してクライアントの認証を行います。

ターゲットアーキテクチャ

実装を説明するために、このパターンではサーバーレスターゲットアーキテクチャをデプロイします。クライアントは API Gateway エンドポイントにリクエストを送信できます。API Gateway は、六角形アーキテクチャパターンを実装するターゲット Lambda 関数にリクエストを転送します。Lambda 関数は、DynamoDB テーブルで作成、読み取り、更新、および削除 (CRUD) 操作を実行します。

> **Important**
>
> このパターンは PoC 環境でテストされました。アーキテクチャを実稼働環境にデプロイする前に、セキュリティレビューを実施して脅威モデルを特定し、安全なコードベースを作成する必要があります。

API は、製品エンティティに対する 5 つの操作をサポートします。

- `GET /products` はすべての製品を返します。

- POST /products は新しい製品を作成します。
- GET /products/{id} は特定の商品を返します。
- PUT /products/{id} は特定の製品を更新します。
- DELETE /products/{id} は特定の製品を削除します。

以下のフォルダ構造を使用して、六角形アーキテクチャパターンに従ってプロジェクトを整理できます。

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to execute commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events triggered via the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
|--- libraries/  # List of 3rd party libraries used by the Lambda function
infra/  # infrastructure code
simple-crud-app.py  # AWS CDK v2 app
```

ツール

サービス

- Amazon API Gateway は、デベロッパーがあらゆる規模で API の公開、保守、モニタリング、セキュリティ保護を簡単に行えるフルマネージドサービスです。
- Amazon DynamoDB は、あらゆる規模で高性能アプリケーションを実行できるように設計された、完全マネージド型のサーバーレスのキーバリュー型 NoSQL データベースです。
- AWS Lambda は、サーバーレスのイベント駆動型のコンピューティングサービスで、サーバーのプロビジョニングや管理を行わなくても、実質あらゆるタイプのアプリケーションやバックエンドサービスのコードを実行できます。200 以上の Software as a Service (SaaS) アプリケーションから Lambda 関数を起動できます。お支払いいただくのは使用した分のみです。

ツール

- このパターンでは、コード開発のバージョン管理システムとして Git が使用されます。
- このパターンのプログラミング言語には Python が使用されています。Python は、高レベルのデータ構造とオブジェクト指向プログラミングへのアプローチを提供します。AWS Lambda には Python サービスの操作を簡素化する組み込みの Python ランタイムが用意されています。
- このパターンの開発とテストには、Visual Studio Code が IDE として使用されています。Python 開発をサポートする任意の IDE (PyCharm など) を使用できます。
- AWS Cloud Development Kit (AWS CDK) は、使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースを定義できるオープンソースのソフトウェア開発フレームワークです。このパターンでは、CDK を使用してクラウドインフラストラクチャをコードとして記述し、デプロイします。
- Poetry はパターン内の依存関係を管理するために使用されます。
- Docker は AWS CDK が Lambda パッケージとレイヤーを構築するために使用します。

コード

このパターンのコードは、GitHub 内の「Lambda 六角形アーキテクチャのサンプル」リポジトリで利用できます。

ベストプラクティス

実稼働環境でこのパターンを使用するには、次のベストプラクティスに従ってください。

- AWS Key Management Service (AWS KMS) のカスタマーマネージドキーを使用して Amazon CloudWatch Logs グループと Amazon DynamoDB テーブルを暗号化します。
- 組織のネットワークからのアクセスのみを許可するように Amazon API Gateway 用 WAF を構成します。
- IAM がニーズを満たさない場合は、API Gateway 認可の他のオプションを検討してください。たとえば、Amazon Cognito ユーザープールや API Gateway Lambda オーソライザーを使用できます。
- DynamoDB バックアップの使用
- 仮想プライベートクラウド (VPC) のデプロイで Lambda 関数を構成し、ネットワークトラフィックをクラウド内に維持します。
- クロスオリジンリソースシェアリング (CORS) プリフライトの許可オリジン構成を更新して、リクエスト元のオリジンドメインのみにアクセス制限します。

- cdk-nag を使用して、AWS CDK コードのセキュリティベストプラクティスを確認します。
- コードスキャンツールを使用して、コードの一般的なセキュリティ問題を見つけることを検討してください。たとえば、Bandit は Python コードの一般的なセキュリティ問題を検出するために設計されたツールです。Pip-audit は Python 環境をスキャンして、既知の脆弱性があるパッケージを探します。

このパターンでは、AWS X-Ray を使用して、アプリケーションのエントリポイント、ドメイン、アダプタを通じてリクエストをトレースします。AWS X-Ray は、開発者がボトルネックを特定して高いレイテンシーを洗い出し、アプリケーションのパフォーマンスを向上させるのに役立ちます。

エピック

プロジェクトを初期化する

タスク	説明	必要なスキル
独自のリポジトリを作成します。	1. GitHub にログインします。 2. 新しいレポジトリを作成します。手順については、GitHub のドキュメントを参照してください。 3. このパターンのサンプルリポジトリをクローン作成して、アカウントの新しいリポジトリにプッシュします。	アプリ開発者
依存関係をインストールします。	1. Poetry をインストールします。 `pip install poetry` 2. ルートディレクトリからパッケージをインストールします。次のコマンドで、アプリケーションと AWS CDK パッケージをインス	アプリ開発者

タスク	説明	必要なスキル

	トールします。これにより、ユニットテストの実行に必要な開発パッケージもインストールされます。インストールされたパッケージはすべて新しい仮想環境に置かれます。
	`poetry install`
	3. インストールされたパッケージをグラフィカルに表示するには、次のコマンドを実行します。
	`poetry show --tree`
	4. すべての依存関係を更新します。
	`poetry update`
	5. 新しく作成した仮想環境内で新しいシェルを開きます。ここにインストール済みの依存関係がすべて含まれています。
	`poetry shell`	

タスク	説明	必要なスキル
IDE を構成する。	Visual Studio Code をお勧めしますが、Python をサポートする任意の IDE でもかまいません。次の手順は、Visual Studio Code 用です。 1. `.vscode/settings` ファイルを更新します。 `{ "python.testing.pytestArgs": [ "app/adapters/tests", "app/entrypoints/api/tests", "app/domain/tests" ], "python.testing.unittestEnabled": false, "python.testing.pytestEnabled": true, "python.envFile": "${workspaceFolder}/.env", }` 2. プロジェクトのルートディレクトリに `.env` ファイルを作成します。これにより、`PYTHONPATH` にプロジェクトのルートディレクトリが含まれ、pytest がそのルートディレクトリを検索してすべてのパッケー	アプリ開発者

タスク	説明	必要なスキル
	ジを正しく検出できるようになります。 `PYTHONPATH=.`	
ユニットテストを実行する。オプション 1: Visual Studio コードを使用する。	1. Poetry によって管理されている仮想環境の Python インタープリターを選択してください。 2. テストエクスプローラーからテストを実行します。	アプリ開発者

ユニットテストを実行する。オプション 2: シェルコマンドを使用する。	1. 仮想環境内で新しいシェルを起動します。
	`poetry shell`
	2. ルートディレクトリから `pytest` のコマンドを実行します。
	`python -m pytest`
	または、Poetry からコマンドを直接実行できます。
	`poetry run python -m pytest`
		アプリ開発者

アプリケーションをデプロイしてテストする

タスク	説明	必要なスキル
一時的な認証情報をリクエストします。	`cdk deploy` の実行時にシェルに AWS 認証情報を保存す	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	るには、AWS IAM アイデンティティセンター (AWS シングルサインオンの後継) を使用して一時的な認証情報の作成を行います。手順については、ブログ記事「IAM アイデンティティセンター で CLI を使用するための短期認証情報を取得する方法」を参照してください。	

タスク	説明	必要なスキル

アプリケーションをデプロイします。	1. AWS CDK v2 をインストールします。
	`npm install -g aws-cdk`
	詳細については、CDK ドキュメントを参照してください。
	2. AWS CDK をアカウントとリージョンにブートストラップします。
	`cdk bootstrap aws://12345678900/us-east-1 --profile aws-profile-name`
	3. AWS プロファイルを使用して、アプリケーションを AWS CloudFormation スタックとしてデプロイします。
	`cdk deploy --profile aws-profile-name`
		アプリ開発者、AWS DevOps

API をテストする。オプション 1: コンソールを使用する。	API Gateway コンソールを使用して API メソッドをテストします。API の操作とリクエスト/レスポンスメッセージの詳細については、GitHub リポジトリの「readme ファイルの API 使用状況セクション」を参照してください。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
API をテストする。オプション 2: Postman を使用する。	Postman などのツールを使用する場合: 1. スタンドアロンアプリケーションまたはブラウザ拡張機能として Postman をインストールします。 2. API Gateway のエンドポイント URL をコピーします。次のような形式になります。 `https://{api-id}.execute-api.{region}.amazonaws.com/{stage}/{path}` 3. [認可] タブで AWS 署名を構成します。手順については、API Gateway REST API の IAM 認証の有効化に関する AWS re:Post の記事を参照してください。 4. API エンドポイントにリクエストを送信するには、Postman を使用します。	アプリ開発者、AWS DevOps

サービスの開発

タスク	説明	必要なスキル
ビジネスドメインのユニットテストを書きます。	1. `test_` ファイル名のプレフィックスを使用して、`app/domain/tests` フォルダ	アプリ開発者

タスク	説明	必要なスキル

	に Python ファイルを作成します。
	2. 次の例を使用して、新しいビジネスロジックをテストする新しいテストメソッドを作成します。
	
	`def test_create_product_should_store_in_repository():`
	`    # Arrange`
	`    command = create_product_command.CreateProductCommand(`
	`        name="Test Product",`
	`        description="Test Description",`
	`    )`
	`    # Act`
	`    create_product_command_handler.handle_create_product_command(`
	`        command=command, unit_of_work=mock_unit_of_work`
	`    )`
	`# Assert`
	
	3. app/domain/commands フォルダにコマンドクラスを作成します。
	4. 機能が新しい場合は、app/domain/command	

タスク	説明	必要なスキル
	_handlers フォルダにコマンドハンドラーのスタブを作成します。 5. まだビジネスロジックがないので、ユニットテストを実行して失敗することを確認してください。 `python -m pytest`	

タスク	説明	必要なスキル

コマンドとコマンドハンドラーを実装します。	1. 新しく作成したコマンドハンドラーファイルにビジネスロジックを実装します。
	2. 外部システムとやり取りする依存関係ごとに、app/domain/ports フォルダに抽象クラスを宣言します。
	
	`class ProductsRepository(ABC):`
	`  @abstractmethod`
	`  def add(self, product: product.Product) -> None:`
	`  ...`
	
	`class UnitOfWork(ABC):`
	`  products: ProductsRepository`
	
	`  @abstractmethod`
	`  def commit(self) -> None:`
	`  ...`
	
	`  @abstractmethod`
	`  def __enter__(self) -> typing.Any:`
	`  ...`
	
	`  @abstractmethod`
	`  def __exit__(self, *args) -> None:`
	`  ...`
		アプリ開発者

タスク	説明	必要なスキル

	3. 抽象ポートクラスを型アノテーションとして使用して、新たに宣言された依存関係を受け入れるようにコマンドハンドラーの署名を更新します。
	
	`def handle_create_product_command(`
	`  command: create_product_command.CreateProductCommand,`
	`  unit_of_work: unit_of_work.UnitOfWork,`
	`) -> str:`
	`  ...`
	
	4. ユニットテストを更新して、コマンドハンドラーに宣言されたすべての依存関係の動作をシミュレートします。
	
	`  # Arrange`
	`  mock_unit_of_work = unittest.mock.create_autospec(`
	`  spec=unit_of_work.UnitOfWork, instance=True`
	`  )`
	`  mock_unit_of_work.products = unittest.mock.create_autospec(`
	`  spec=unit_of_work.ProductsR`	

タスク	説明	必要なスキル
	epository, instance= True)	
	5. テスト内のアサーションロジックを更新して、想定される依存関係の呼び出しがあるかどうかを確認します。 ``` # Assert mock_unit _of_work.commit.as sert_called_once() product = mock_unit_of_work. products.add.call_ args.args[0] assertpy. assert_that(produc t.name).is_equal_t o("Test Product") assertpy. assert_that(produc t.description).is_ equal_to("Test Description") ``` 6. ユニットテストを実行して成功することを確認します。 ``` python -m pytest ```	

タスク	説明	必要なスキル

セカンダリアダプターの統合テストを書く。	1. test_ をファイル名のプレフィックスとして使用して、app/adapters/tests フォルダにテストファイルを作成します。
	2. Moto ライブラリを使用して AWS サービスをモックします。
	`@pytest.fixture`
	`def mock_dynamodb():`
	`  with moto.mock_dynamodb():`
	`  yield boto3.resource("dynamodb", region_name="eu-central-1")`
	3. アダプターの統合テスト用の新しいテストメソッドを作成します。
	`def test_add_and_commit_should_store_product(mock_dynamodb):`
	`  # Arrange`
	`  unit_of_work = dynamodb_unit_of_work.DynamoDBUnitOfWork(`
	`  table_name=TEST_TABLE_NAME, dynamodb_client=mock_dynamodb.meta.client`
	`  )`
	`  current_time = datetime.datetime.`
		アプリ開発者

タスク	説明	必要なスキル

	now(datetime.timezone.utc).isoformat()
	
	    new_product_id = str(uuid.uuid4())
	    new_product = product.Product(
	        id=new_product_id,
	        name="test-name",
	        description="test-description",
	        createDate=current_time,
	        lastUpdateDate=current_time,
	    )
	
	    # Act
	    with unit_of_work:
	        unit_of_work.products.add(new_product)
	        unit_of_work.commit()
	
	    # Assert
	4. app/adapters フォルダにアダプタークラスを作成します。ports フォルダの抽象クラスを基本クラスとして使用します。
	5. まだロジックがないので、ユニットテストを実行して	

タスク	説明	必要なスキル
	失敗することを確認します。 `python -m pytest`	

タスク	説明	必要なスキル

セカンダリアダプタを実装する。	1. 新しく作成したアダプターファイルにロジックを実装します。
	2. テストアサーションを更新します。
	
	`# Assert`
	`    with unit_of_work_readonly:`
	`        product_from_db = unit_of_work_readonly.products.get(new_product_id)`
	
	`    assertpy.assert_that(product_from_db).is_not_none()`
	`    assertpy.assert_that(product_from_db.dict()).is_equal_to(`
	`        {`
	`            "id": new_product_id,`
	`            "name": "test-name",`
	`            "description": "test-description",`
	`            "createDate": current_time,`
	`            "lastUpdateDate": current_time,`
	`        }`
	`    )`
		アプリ開発者

タスク	説明	必要なスキル
	3. ユニットテストを実行して成功することを確認します。 `python -m pytest`	

タスク	説明	必要なスキル
エンドツーエンドテストを作成する。	1. test_ をファイル名のプレフィックスとして使用して、app/entrypoints/api/tests フォルダにテストファイルを作成します。 2. テストで Lambda を呼び出すために使用される Lambda コンテキストフィクスチャを作成します。 3. API 呼び出し用のテストメソッドを作成します。	アプリ開発者

```
@pytest.fixture
def lambda_co
ntext():
    @dataclass
    class LambdaCon
text:
        function_
name: str = "test"
        memory_li
mit_in_mb: int = 128
        invoked_f
unction_arn: str =
 "arn:aws:lambda:eu
-west-1:809313241:
function:test"
        aws_reque
st_id: str =
 "52fdfc07-2182-154
f-163f-5f0f9a621d7
2"

 return LambdaCon
text()
```

タスク	説明	必要なスキル

	<pre>def test_crea
	te_product(lambda_
	context):
	    # Arrange
	    name = "TestName"
	    description =
	 "Test description"
	    request =
	 api_model.CreatePr
	oductRequest(name=
	name, descripti
	on=description)
	
	    minimal_event =
	 api_gateway_proxy_
	event.APIGatewayPr
	oxyEvent(
	        {
	            "path": "/
	products",
	            "httpMeth
	od": "POST",
	            "requestC
	ontext": { #
	 correlation ID
	
	  "requestId":
	 "c6af9ac6-7b61-11e
	6-9a41-93e8deadbee
	f"
	            },
	            "body":
	 json.dumps(request
	.dict()),
	        }
	    )
	
	    create_pr
	oduct_func_mock =
	 unittest.mock.crea
	te_autospec(</pre>	

タスク	説明	必要なスキル

	<pre>spec=crea
	te_product_command
	_handler.handle_cr
	eate_product_comma
	nd
	)
	handler.c
	reate_product_comm
	and_handler.handle
	_create_product_co
	mmand = (
	create_pr
	oduct_func_mock
	)
	
	# Act
	handler.h
	andler(minimal_eve
	nt, lambda_context)</pre>
	4. まだロジックがないので、ユニットテストを実行して失敗することを確認します。
	<pre>python -m pytest</pre>	

タスク	説明	必要なスキル

プライマリアダプタを実装する。	1. API ビジネスロジック用の関数を作成し、API リソースとして宣言します。
	
	`@tracer.capture_method`
	`@app.post("/products")`
	`@utils.parse_event(model=api_model.CreateProductRequest, app_context=app)`
	`def create_product(`
	`request: api_model.CreateProductRequest,`
	`) -> api_model.CreateProductResponse:`
	`"""Creates a product."""`
	`...`
	
	Note
	表示されるすべてのデコレータは、AWS Lambda Powertools for Python ライブラリの機能です。詳細については、AWS Lambda Powertools for Python ウェブサイトを参照してください。
		アプリ開発者

タスク	説明	必要なスキル

	2. API ロジックを実装します。
	
	`id=create_product_command_handler.handle_create_product_command(`
	`command=create_product_command.CreateProductCommand(`
	`name=request.name,`
	`description=request.description,`
	`),`
	`unit_of_work=unit_of_work,`
	`)`
	`response = api_model.CreateProductResponse(id=id)`
	`return response.dict()`
	
	3. ユニットテストを実行して成功することを確認します。
	
	`python -m pytest`	

関連リソース

APG ガイド

- で六角形アーキテクチャを構築する

AWS リファレンス

- AWS Lambda ドキュメント
- AWS CDK ドキュメント
  - 初めての CDK アプリケーション
- API Gateway ドキュメント
  - IAM アクセス許可により API へのアクセスを制御する
  - API Gateway コンソールを使用して REST API メソッドをテストする
- Amazon DynamoDB ドキュメント

ツール

- git-scm.com ウェブサイト
- Git のインストール
- 新しい GitHub リポジトリの作成
- Python ウェブサイト
- 「AWS Lambda Powertools for Python」
- Postman ウェブサイト
- Python モックオブジェクトライブラリ
- Poetry ウェブサイト

IDE

- Visual Studio Code ウェブサイト
- PyCharm ウェブサイト

その他のパターン

- Amazon Cognito アイデンティティプールを使用して ASP.NET Core アプリケーションから AWS のサービスにアクセスする
- AWS Fargate、AWS PrivateLink、Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします。
- AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします
- ある の AWS CodeCommit リポジトリ AWS アカウント を別のアカウントの Amazon SageMaker AI Studio Classic に関連付ける
- AWS CloudFormation スタックと関連リソースの削除を自動化する
- を使用して IP アドレスまたは位置情報に基づいてアクセスを制限する AWS WAF
- DevOps プラクティスと AWS Cloud9 を使用して、マイクロサービスで疎結合アーキテクチャを構築する
- AWS Amplify を使用してサーバーレスの React Native モバイルアプリを構築する
- AWS CodeCommit、AWS CodePipeline、AWS Device Farm を使用して iOS アプリを構築してテストする
- NLog を使用して Amazon CloudWatch Logs 内の.NET アプリケーションの口ギングを設定します
- CodePipeline と HashiCorp Packer を使用してパイプラインと AMI を作成
- CodePipeline を使用してパイプラインを作成し、アーティファクトの更新をオンプレミスの EC2 インスタンスにデプロイ
- Amazon ECS タスク定義を作成し、Amazon EFS を使用して EC2 インスタンスにファイルシステムをマウントする
- gRPC ベースのアプリケーションを Amazon EKS クラスターにデプロイし、Application Load Balancer でアクセスする
- ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation
- Terraform を使用して CloudWatch Synthetics Canary をデプロイする
- AWS Fargate を使用して Amazon ECS に Java マイクロサービスをデプロイする
- Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする
- Terraform を使用して AWS Wavelength ゾーンにリソースをデプロイする
- Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する

- メッセージキューを Microsoft Azure Service Bus から Amazon SQS に移行
- .NET アプリケーションを Microsoft Azure App Serviceから AWS Elastic Beanstalk に移行
- バイナリメソッドを使用してオンプレミスの Go ウェブアプリケーションを AWS Elastic Beanstalk に移行します
- AWS を使用してオンプレミス SFTP サーバーを に移行する AWS Transfer for SFTP
- IBM WebSphere アプリケーションサーバーから Amazon EC2 の Apache Tomcat に移行
- IBM WebSphere アプリケーションサーバーから Amazon EC2 上の Apache Tomcat への自動スケーリングによる移行
- Oracle GlassFish から AWS Elastic Beanstalk へ移行する
- AWS App2Container を使用したオンプレミスの Java プリケーションの AWS への移行
- OpenText TeamSite ワークロードの AWS クラウド への移行
- ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行
- ASP.NET ウェブフォームアプリケーションを AWS で最新化
- Amazon EC2 Linux インスタンスで ASP.NET Core ウェブ API Docker コンテナを実行する
- Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する
- AWS で可用性の高い PeopleSoft アーキテクチャを設定する
- 自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する
- Amazon Bedrock AWS Step Functions を使用して の状態をトラブルシューティングする
- Network Firewall を使用して、アウトバウンドトラフィックのサーバー名表示から DNS ドメイン名をキャプチャする
- フラスコと AWS Elastic Beanstalk を使用して AI/ML モデルの結果を視覚化

IoT

トピック

- AWS IoT 環境内のセキュリティイベントのログ記録とモニタリングを設定する
- データレイク内の AWS IoT SiteWise メタデータ属性を抽出してクエリする
- クライアントデバイスによる AWS IoT Greengrass のセットアップとトラブルシューティング
- その他のパターン

AWS IoT 環境内のセキュリティイベントのログ記録とモニタリングを設定する

作成者:Prateek Prakash (AWS)

概要

特に組織は何十億ものデバイスを IT 環境に接続しているため、モノのインターネット (IoT) 環境の安全を確保することは重要な優先事項です。このパターンは、 の IoT 環境全体でセキュリティイベントのログ記録とモニタリングを実装するために使用できるリファレンスアーキテクチャを提供します AWS クラウド。通常、 の IoT 環境 AWS クラウド には次の 3 つのレイヤーがあります。

- 関連するテレメトリデータを生成する IoT デバイス。
- AWS IoT IoT デバイスを他のデバイスや に接続する サービス (、AWS IoT CoreAWS IoT Device Management、 などAWS IoT Device Defender) AWS のサービス。
- テレメトリデータ AWS のサービス を処理し、さまざまなビジネスユースケースに役立つインサイトを提供するバックエンド。

AWS IoT レンズ - AWS Well-Architected Framework ホワイトペーパーが提供するベストプラクティスは、クラウドベースのアーキテクチャを確認および改善し、設計上の意思決定によるビジネスへの影響をよりよく理解するのに役立ちます。重要な推奨事項は、デバイスと でアプリケーションログとメトリクスを分析することです AWS クラウド。これは、さまざまなアプローチや手法 (「脅威モデリング」など) を活用して、潜在的なセキュリティ問題を検出するために監視する必要のあるメトリクスとイベントを特定することで実現できます。

このパターンでは、 AWS IoT および セキュリティサービスを使用して、 上の IoT 環境のセキュリティログ記録とモニタリングリファレンスアーキテクチャを設計および実装する方法について説明します AWS クラウド。このアーキテクチャは、既存の AWS セキュリティのベストプラクティスに基づいて構築され、IoT 環境に適用されます。

前提条件と制限

前提条件

- 既存のランディングゾーン環境。詳細については、 AWS 「 規範ガイダンス」ウェブサイトの「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。
- ランディングゾーンでは以下のアカウントが利用可能である必要があります。

- ログアーカイブアカウント — このアカウントは、ランディングゾーンの組織単位 (OU) 内のアカウントのログ情報にアクセスする必要があるユーザー向けです。詳細については、 AWS 「 規範ガイダンス」ウェブサイトの「セキュリティAWS リファレンスアーキテクチャ」ガイドの「セキュリティ OU – ログアーカイブアカウント」セクションを参照してください。
- セキュリティアカウント — セキュリティチームとコンプライアンスチームは、このアカウントを監査や緊急のセキュリティ運用に使用します。このアカウントは Amazon GuardDuty の管理者アカウントとしても指定されています。管理者アカウントのユーザーは、自分のアカウントおよびすべてのメンバーアカウントの GuardDuty の検出結果を表示および管理できるだけでなく、GuardDuty の設定もできます。詳細については、GuardDuty ドキュメントの「GuardDuty での複数のアカウントの管理」を参照してください。 GuardDuty
- IoT アカウント — このアカウントは IoT 環境用です。

アーキテクチャ

このパターンは、 AWS ソリューションライブラリからの集中ロギングソリューションを拡張して、セキュリティ関連の IoT イベントを収集して処理します。集中ロギングソリューションはセキュリティアカウントにデプロイされ、 Amazon CloudWatch logs の収集、分析、表示を 1 つのダッシュボードで行えます。このソリューションは、複数のソースからのログファイルを統合、管理、分析します。最後に、集中型ロギングソリューションでは Amazon OpenSearch Service と OpenSearch ダッシュボードも使用して、すべてのログイベントを統合して表示します。

次のアーキテクチャ図は、 の IoT セキュリティログ記録およびリファレンスアーキテクチャの主要なコンポーネントを示しています AWS クラウド。

この図表は、次のワークフローを示しています：

1. IoT モノは、異常なセキュリティイベントがないか監視する必要があるデバイスです。これらのデバイスは エージェントを実行して、セキュリティイベントまたはメトリクスを AWS IoT Core および に発行します AWS IoT Device Defender。
2. AWS IoT ログ記録を有効にすると、 は、メッセージブローカーとルールエンジンを介してデバイスから Amazon CloudWatch Logs に渡される各メッセージに関する進行状況イベント AWS IoT を送信します。CloudWatch Logs サブスクリプションを使用して、「集中型ロギングソリューション」にイベントをプッシュできます。詳細については、 AWS IoT Core ドキュメントのAWS IoT 「メトリクスとディメンション」を参照してください。

3. AWS IoT Device Defender は、IoT デバイスの安全でない設定とセキュリティメトリクスをモニタリングするのに役立ちます。異常が検出されると、アラームは Amazon Simple Notification Service (Amazon SNS) に通知します。Amazon SNS にはサブスクライバーとして AWS Lambda 関数があります。Lambda 関数はアラームをメッセージとして CloudWatch Logs に送信します。CloudWatch Logs サブスクリプションを使用して、集中型ロギングソリューションにイベントをプッシュできます。詳細については、 AWS IoT Core ドキュメントの「監査チェック」、「デバイス側のログを CloudWatch にアップロードする」、AWS IoT 「ログ記録の設定」を参照してください。
4. AWS CloudTrail は、変更を行う AWS IoT Core コントロールプレーンアクション (APIs。CloudTrail がランディングゾーン実装の一部として設定されると、CloudWatch Logs にイベントが送信されます。サブスクリプションを使用して、一元化されたログ記録ソリューションにイベントをプッシュできます。
5. AWS Config マネージドルールまたはカスタムルールは、IoT 環境の一部であるリソースを評価します。CloudWatch CloudWatch Logs をターゲットとする CloudWatch イベントを使用して、「コンプライアンス変更通知」をモニタリングします。コンプライアンス変更通知が CloudWatch Logs に送信されたら、サブスクリプションを使用して集中型ロギングソリューションにイベントをプッシュできます。
6. Amazon GuardDuty は CloudTrail 管理イベントを継続的に分析し、既知の悪意のある IP アドレス、異常な位置情報、または匿名化プロキシから AWS IoT Core エンドポイントに対して行われた API コールを特定するのに役立ちます。CloudWatch Logs のロググループをターゲットとする CloudWatch Events を使用して GuardDuty 通知をモニタリングします。GuardDuty 通知が CloudWatch Logs に送信されると、サブスクリプションを使用して集中型モニタリングソリューションにイベントをプッシュしたり、セキュリティアカウントの GuardDuty コンソールを使用して通知を表示したりできます。
7. AWS Security Hub は、セキュリティのベストプラクティスを使用して IoT アカウントをモニタリングします。CloudWatch Logs のロググループをターゲットとして CloudWatch イベントを使用して、Security Hub 通知を監視します。Security Hub 通知が CloudWatch Logs に送信されたら、サブスクリプションを使用して集中型モニタリングソリューションにイベントをプッシュするか、セキュリティアカウントの Security Hub コンソールを使用して通知を表示します。
8. Amazon Detective は情報を評価および分析して根本原因を分離し、IoT アーキテクチャ内の AWS IoT エンドポイントやその他のサービスへの異常な呼び出しに対してセキュリティ検出結果に対してアクションを実行します。
9. Amazon Athena は、ログアーカイブアカウントに保存されているログにクエリを実行することで、セキュリティ結果の理解を深め、傾向や悪意のあるアクティビティを特定します。

ツール

- 「Amazon Athena」は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析することを容易にするインタラクティブなクエリサービスです。
- AWS CloudTrail は、 のガバナンス、コンプライアンス、運用およびリスク監査を可能にするのに役立ちます AWS アカウント。
- Amazon CloudWatch は、 AWS リソースと で実行されるアプリケーションを AWS リアルタイムでモニタリングします。 CloudWatch を使用してメトリクスを収集および追跡できます。メトリクスとは、リソースやアプリケーションについて測定できる変数です。
- Amazon CloudWatch Logs は、使用するすべてのシステム、アプリケーション、および からの口グ AWS のサービス を一元化します。ログを表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。
- AWS Config は、 内の AWS リソースの設定の詳細ビューを提供します AWS アカウント。
- 「Amazon Detective」を使用すると、セキュリティに関する検出結果や不審なアクティビティの根本原因を簡単に分析、調査、および迅速に特定できます。
- AWS Glue は、データの分類、クリーニング、強化、さまざまなデータストアとデータストリーム間の信頼性の高い移動を簡単かつ費用対効果の高い方法で実現する、フルマネージド型の抽出、変換、ロード (ETL) サービスです。
- 「Amazon GuardDuty」は、継続的なセキュリティモニタリングサービスです。
- AWS IoT Core は、インターネットに接続されたデバイス (センサー、アクチュエータ、組み込みデバイス、ワイヤレスデバイス、スマートアプライアンスなど) が MQTT、HTTPS、LoRaWAN AWS クラウド 経由で に接続するための安全な双方向通信を提供します。
- AWS IoT Device Defender は、デバイスの設定の監査、異常動作の検出を目的とした接続デバイスのモニタリング、セキュリティリスクの軽減を行うことができるセキュリティサービスです。
- Amazon OpenSearch Service は、 で OpenSearch クラスターのデプロイ、運用、スケーリングを容易にするマネージドサービスです AWS クラウド。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合することができるアカウント管理サービスです。
- AWS Security Hub は、 のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスに照らして環境をチェックするのに役立ちます。
- Amazon Virtual Private Cloud (Amazon VPC) は、 の論理的に分離されたセクション AWS クラウド をプロビジョニングし、定義した仮想ネットワークで AWS リソースを起動できます。仮想

ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

ランディングゾーン環境に IoT アカウントを設定する

タスク	説明	必要なスキル
IoT アカウントのセキュリティガードレールを検証します。	CloudTrail、 AWS Config、GuardDuty、Security Hub のガードレールが IoT アカウントで有効になっていることを確認します。	AWS 管理者
IoT アカウントがセキュリティアカウントのメンバーアカウントとして設定されていることを確認します。	IoT アカウントがセキュリティアカウントの GuardDuty と Security Hub のメンバーアカウントとして設定され、関連付けられていることを確認します。 詳細については、GuardDuty ドキュメントの「Managing GuardDuty accounts with AWS Organizations」および Security Hub ドキュメントの「Managing administrator and member accounts」を参照してください。 GuardDuty	AWS 管理者
ログのアーカイブを検証します。	CloudTrail、 AWS Config、および VPC フローログがログアーカイブアカウントに保存されていることを確認します。	AWS 管理者

集中型ロギングソリューションをセットアップします。

タスク	説明	必要なスキル
セキュリティアカウントに集中ロギングソリューションを設定します。	AWS Management Console セキュリティアカウントの にサインインし、 AWS ソリューションライブラリから集中ロギングソリューションを設定し、Amazon OpenSearch Service と OpenSearch Dashboards で CloudWatch Logs を収集、分析、表示します。 詳細については、「 AWS ソリューションライブラリ」の「集中ロギング実装ガイド」の「集中ロギングソリューションを使用した 1 つのダッシュボードでの Amazon CloudWatch Logs の収集、分析、表示」を参照してください。	AWS 管理者

IoT アカウントで AWS リソースを設定および設定する

タスク	説明	必要なスキル
AWS IoT ログ記録を設定します。	IoT アカウントの AWS Management Console にサインインします。CloudWatch Logs にログを送信する AWS IoT Core ように を設定および設定します。	AWS 管理者

タスク	説明	必要なスキル
	詳細については、 AWS IoT Core ドキュメントの「CloudWatch Logs を使用したAWS IoT ログ記録の設定とモニタリング」を参照してください。 AWS IoT CloudWatch	
セットアップします AWS IoT Device Defender。	IoT リソースを監査し、異常を検出する AWS IoT Device Defender ように を設定します。 詳細については、 AWS IoT Core ドキュメントの「 の開始 AWS IoT Device Defender方法」を参照してください。	AWS 管理者
CloudTrail のセットアップ	CloudWatch Logs にイベントを送信するように CloudTrail を設定します。 詳細については、CloudTrail ドキュメントの「CloudWatch Logs へのイベントの送信」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
AWS Config および AWS Config ルールを設定します。	AWS Config と必要な AWS Config ルールを設定します。 詳細については、 AWS Config ドキュメントの「コンソール AWS Config での設定」および「AWS Config ルールの追加」を参照してください。	AWS 管理者
GuardDuty のセットアップ	CloudWatch Logs の ロググループをターゲットとして Amazon CloudWatch Events に検出結果を送信するように GuardDuty をセットアップして設定します。 詳細については、GuardDuty ドキュメントのAmazon CloudWatch Events を使用した GuardDuty の検出結果へのカスタムレスポンスの作成」を参照してください。 GuardDuty	AWS 管理者
Security Hub を設定します。	Security Hub をセットアップし、「CIS AWS Foundations ベンチマーク」および「AWS Foundational セキュリティベストプラクティス」標準を有効にします。 詳細については、Security Hub ドキュメントの「自動応答と修復」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
Amazon Detective の設定	Detective を設定して、セキュリティ検出結果の分析を容易にします。 詳細については、Amazon Detective ドキュメントの「Amazon Detective の開始方法」を参照してください。	AWS 管理者
Amazon Athena と をセットアップします AWS Glue。	Athena と をセットアップ AWS Glue して、セキュリティインシデントの調査を実行する AWS のサービス ログをクエリします。 詳細については、Amazon Athena ドキュメントのAWS のサービス「ログのクエリ」を参照してください。	AWS 管理者

関連リソース

- 「ランディングゾーンとは何ですか?」

データレイク内の AWS IoT SiteWise メタデータ属性を抽出してクエリする

作成者: Ambarish Dongaonkar (AWS)

概要

AWS IoT SiteWise は、アセットモデルと階層を使用して、産業機器、プロセス、施設を表します。各モデルまたはアセットには、環境固有の複数の属性が含まれる場合があります。メタデータ属性の例には、アセットの設置場所または物理的な場所、プラントの詳細、機器識別子などがあります。これらの属性値は資産測定データを補完し、ビジネス価値を最大化します。機械学習 (ML) は、このメタデータをさらに詳しく把握し、エンジニアリングタスクを効率化します。

ただし、メタデータ属性を AWS IoT SiteWise サービスから直接クエリすることはできません。属性をクエリ可能にするには、属性を抽出してデータレイクに取り込む必要があります。このパターンでは、Python スクリプトを使用してすべての AWS IoT SiteWise アセットの属性を抽出し、Amazon Simple Storage Service (Amazon S3) バケットのデータレイクに取り込みます。このプロセスが完了したら、Amazon Athena の SQL クエリを使用して、 AWS IoT SiteWise メタデータ属性や測定データセットなどの他のデータセットにアクセスできます。メタデータ属性情報は、 AWS IoT SiteWise モニターやダッシュボードを操作する場合にも役立ちます。Amazon S3 バケット内の抽出された属性を使用して Amazon Amazon QuickSight ダッシュボードを構築することもできます。

パターンにはリファレンスコードがあり、 AWS Lambda や などのユースケースに最適なコンピューティングサービスを使用してコードを実装できます AWS Glue。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Lambda 関数または AWS Glue ジョブをセットアップするためのアクセス許可。
- Amazon S3 バケット。
- アセットモデルと階層は で設定されます AWS IoT SiteWise。詳細については、 AWS IoT SiteWise ドキュメントの「アセットモデルの作成」を参照してください。

アーキテクチャ

Lambda 関数または AWS Glue ジョブを使用して、このプロセスを完了できます。モデル数が 100 未満で、各モデルの属性が平均 15 個以下の場合は、Lambda を使用することをお勧めします。他のすべてのユースケースでは、 を使用することをお勧めします AWS Glue。

以下の図で、ソリューションアーキテクチャとワークフローを示します。

1. スケジュールされた AWS Glue ジョブまたは Lambda 関数が実行されます。からアセットメタデータ属性を抽出 AWS IoT SiteWise し、Amazon S3 バケットに取り込みます。
2. AWS Glue クローラはAmazon S3バケット内の抽出データをクロールし、 にテーブルを作成します AWS Glue Data Catalog。
3. 標準の SQL を使用して、Amazon Athena は のテーブルをクエリします AWS Glue Data Catalog。

自動化とスケール

AWS IoT SiteWise アセット設定の更新頻度に応じて、Lambda 関数または AWS Glue ジョブを毎日または毎週実行するようにスケジュールできます。

サンプルコードが処理できる AWS IoT SiteWise アセットの数に制限はありませんが、多数のアセットがあると、プロセスの完了に必要な時間が長くなる可能性があります。

ツール

- 「Amazon Athena」はインタラクティブなクエリサービスで、Amazon S3 内のデータをスタンダード SQL を使用して直接分析できます。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS IoT SiteWise は、産業機器から大規模にデータを収集、モデル化、分析、視覚化するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。

エピック

ジョブまたは関数をセットアップする

タスク	説明	必要なスキル
IAM のアクセス許可を構成するには	IAM コンソールで、Lambda 関数または AWS Glue ジョブが引き受ける IAM ロールにアクセス許可を付与して、以下を実行します。 • AWS IoT SiteWise サービスからの読み取り • Amazon S3 バケットへの書き込み 詳細については、IAM ドキュメントの「 のロール AWS のサービスの作成」を参照してください。	AWS 全般
Lambda 関数または AWS Glue ジョブを作成します。	Lambda を使用している場合は、新しい Lambda 関数を作成します。[Runtime] (ランタイム) では、[Python] を選択します。詳細については、Lambda ドキュメントの「Python を使用した Lambda 関数の構築」を参照してください。	AWS 全般

タスク	説明	必要なスキル
	を使用している場合は AWS Glue、 AWS Glue コンソールで新しい Python シェルジョブを作成します。詳細については、 AWS Glue ドキュメントの「Python シェルジョブの追加」を参照してください。	
Lambda 関数または AWS Glue ジョブを更新します。	新しい Lambda 関数または AWS Glue ジョブを変更し、「追加情報」セクションにコードサンプルを入力します。必要な場合、ユースケースに合わせてコードを変更します。詳細については、Lambda ドキュメントの「コンソールエディタを使用してコードを編集する」およびAWS Glue ドキュメントの「スクリプトの使用」を参照してください。	AWS 全般

ジョブまたは関数を実行する

タスク	説明	必要なスキル
Lambda 関数または AWS Glue ジョブを実行します。	Lambda 関数または AWS Glue ジョブを実行します。詳細については、Lambda ドキュメントの「Lambda 関数を呼び出す」または AWS Glue ドキュメントの「トリガーを使用したジョブの開始」を参照してください。これにより、 AWS IoT SiteWise	AWS 全般

タスク	説明	必要なスキル
	階層内のアセットとモデルのメタデータ属性が抽出され、指定された Amazon S3 バケットに保存されます。	
AWS Glue クローラをセットアップします。	CSV 形式のファイルに必要な形式分類子を使用して AWS Glue クローラを設定します。Lambda 関数または AWS Glue ジョブで使用される Amazon S3 バケットとプレフィックスの詳細を使用します。詳細については、 AWS Glue ドキュメントの「クローラの定義」を参照してください。	AWS 全般
AWS Glue クローラを実行します。	クローラを実行して、Lambda 関数または AWS Glue ジョブによって作成されたデータファイルを処理します。クローラは、指定された にテーブルを作成します AWS Glue Data Catalog。詳細については、 AWS Glue ドキュメントの「」または「トリガーを使用したクローラの開始」を参照してください。	AWS 全般

タスク	説明	必要なスキル
メタデータ属性をクエリします。	Amazon Athena を使用して、ユースケース AWS Glue Data Catalog に応じて標準の SQL を使用して をクエリします。メタデータ属性テーブルは、他のデータベースやテーブルと結合できます。詳細については、Amazon Athena ドキュメントの「開始方法」を参照してください。	AWS 全般

関連リソース

- Amazon Athena ドキュメント
- AWS Glue ドキュメント
- AWS IoT SiteWise API リファレンス
- AWS IoT SiteWise ユーザーガイド
  - 入門
  - 産業用アセットのモデリング
  - アセットモデル間の関係 (階層) を定義する
  - アセットの関連付けと関連付け解除
  - AWS IoT SiteWise デモの作成
- IOTSiteWise (Python 用 SDK ドキュメント)
- Lambda ドキュメント

追加情報

Code

提供されているサンプルコードは参照用であり、使用状況に合わせて必要に応じてカスタマイズできます。

```
# Following code can be used in an AWS Lambda function or in an AWS Glue Python shell
 job.
# IAM roles used for this job need read access to the AWS IoT SiteWise service and
 write access to the S3 bucket.
sw_client = boto3.client('iotsitewise')
s3_client = boto3.client('s3')
output = io.StringIO()

attribute_list=[]
bucket = '{3_bucket name}'
prefix = '{s3_bucket prefix}'
output.write("model_id,model_name,asset_id,asset_name,attribuet_id,attribute_name,attribute_val
\n")

m_resp = sw_client.list_asset_models()
for m_rec in m_resp['assetModelSummaries']:
     model_id = m_rec['id']
     model_name = m_rec['name']

     attribute_list.clear()
     dam_response = sw_client.describe_asset_model(assetModelId=model_id)
     for rec in dam_response['assetModelProperties']:
         if 'attribute' in rec['type']:
            attribute_list.append(rec['name'])

     response = sw_client.list_assets(assetModelId=model_id, filter='ALL')
     for asset in response['assetSummaries']:
         asset_id = asset['id']
         asset_name = asset['name']
         resp = sw_client.describe_asset(assetId=asset_id)
         for rec in resp['assetProperties']:
            if rec['name'] in attribute_list:
                p_resp = sw_client.get_asset_property_value(assetId=asset_id,
 propertyId=rec['id'])
                if 'propertyValue' in p_resp:
                    if p_resp['propertyValue']['value']:
                        if 'stringValue' in p_resp['propertyValue']['value']:
                             output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['stringValue']) + "\n")

                        if 'doubleValue' in p_resp['propertyValue']['value']:
```

```
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['doubleValue']) + "\n")
                        if 'integerValue' in p_resp['propertyValue']['value']:
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['integerValue']) + "\n")
                         if 'booleanValue' in p_resp['propertyValue']['value']:
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['booleanValue']) + "\n")

output.seek(0)
s3_client.put_object(Bucket=bucket, Key= prefix + '/data.csv', Body=output.getvalue())
output.close()
```

クライアントデバイスによる AWS IoT Greengrass のセットアップとトラブルシューティング

マルアン・セフィアー二とアカランカ・デ・シルバ (AWS) によって作成されました

概要

AWS IoT Greengrass は、エッジデバイスで IoT ソフトウェアを構築、デプロイ、管理するためのモノのインターネット (IoT) エッジランタイムおよびクラウドサービスです。AWS IoT Greengrass のユースケースには以下が含まれます。

- AWS IoT Greengrass ゲートウェイをホームオートメーションのハブとして使用するスマートホーム
- AWS IoT Greengrass が製造現場からのデータの取り込みとローカル処理を容易にするスマートファクトリー

AWS IoT Greengrass は、通常は AWS IoT Core に直接接続する他のエッジデバイス (クライアントデバイスとも呼ばれる) の安全で認証された MQTT 接続エンドポイントとして機能します。この機能は、クライアントデバイスが AWS IoT Core エンドポイントに直接ネットワークアクセスできない場合に役立ちます。

AWS IoT Greengrass は、以下のユースケースでクライアントデバイスで使用するようにセットアップできます。

- クライアントデバイスが AWS IoT Greengrass にデータを送信する場合
- AWS IoT Greengrass が AWS IoT Core にデータを転送するには
- AWS IoT Core ルールエンジンの高度な AWS IoT Core ルールエンジン特徴量を活用するには

これらの機能を使用するには、AWS IoT Greengrass デバイスに次のコンポーネントをインストールして設定する必要があります。

- MQTT ブローカー
- MQTT ブリッジ
- クライアントデバイス認証
- IP ディテクター

さらに、クライアントデバイスから公開されるメッセージは JSON 形式または 「プロトコルバッファ (protobuf)」 形式である必要があります。

このパターンでは、これらの必要なコンポーネントをインストールして設定する方法を説明し、トラブルシューティングのヒントやベストプラクティスを提供します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 「AWS Command Line Interface (AWS CLI) バージョン 2」
- Python 3.7 以降を実行する 2 台のクライアントデバイス
- Java ランタイム環境 (JRE) バージョン 8 以降と 「Amazon Corretto 11」 または 「OpenJDK 11」 を実行している 1 つのコアデバイス

制約事項

- AWS IoT Core を利用できる AWS リージョンを選択する必要があります。AWS IoT Core のリージョンの最新リストについては、「リージョン別の AWS サービス」 を参照してください。
- コアデバイスには、少なくとも 172 MB の RAM と 512 MB のディスクスペースが必要です。

アーキテクチャ

このパターンのソリューションアーキテクチャを次の図に示します。

アーキテクチャには以下が含まれます。

- 2 つのクライアントデバイス 各デバイスには、プライベートキー、デバイス証明書、ルート認証局 (CA) 証明書が含まれます。MQTT クライアントを含む AWS IoT デバイス SDK も各クライアントデバイスにインストールされます。
- AWS IoT Greengrass がデプロイされたコアデバイスには、以下のコンポーネントが含まれます。
  - MQTT ブローカー
  - MQTT ブリッジ

- クライアントデバイス認証
- IP ディテクター

このアーキテクチャは、以下のシナリオをサポートします。

- クライアントデバイスは MQTT クライアントを使用して、コアデバイスの MQTT ブローカーを介して相互に通信できます。
- クライアントデバイスは、コアデバイスの MQTT ブローカーと MQTT ブリッジを介してクラウド内の AWS IoT Core と通信することもできます。
- クラウド内の AWS IoT Core は、MQTT テストクライアント、コアデバイスの MQTT ブリッジ、および MQTT ブローカーを介してクライアントデバイスにメッセージを送信できます。

クライアントデバイスとコアデバイス間の通信の詳細については、「追加情報」 セクションを参照してください。

ツール

AWS サービス

- 「AWS IoT Greengrass」 は、デバイス上で IoT アプリケーションを構築、デプロイ、管理するのに役立つオープンソースの IoT エッジランタイムおよびクラウドサービスです。
- 「AWS IoT Core」 は、インターネットに接続されたデバイスが AWS クラウドに接続するための安全な双方向通信を提供します。
- 「AWS IoT Device SDK」 はソフトウェア開発キットで、その中に、オープンソースライブラリ、サンプル付きのデベロッパーガイド、および移植ガイドが含まれているので、選択したプラットフォーム上で革新的な IoT 製品またはソリューションを構築できます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

ベストプラクティス

- 変換や条件付きアクションなどの AWS IoT Core ルールエンジンの高度な特徴量を利用するには、クライアントデバイスからのメッセージのペイロードを JSON または Protobuf 形式にする必要があります。
- MQTT ブリッジを双方向通信を許可するように設定します。

- AWS IoT Greengrass で IP ディテクタコンポーネントを設定してデプロイし、コアデバイスの IP アドレスが MQTT ブローカー証明書のサブジェクト代替名 (SAN) フィールドに含まれるようにします。

エピック

コアデバイスをセットアップ

タスク	説明	必要なスキル
コアデバイスに AWS IoT Greengrass セットアップします。	「開発者ガイド」 の指示に従って AWS IoT Greengrass Core ソフトウェアをインストールします。	AWS IoT Greengrass

インストールの状態を確認します。	次のコマンドを使用して、コアデバイスで AWS IoT Greengrass サービスの状態を確認します。
	`sudo systemctl status greengrass.service`
	コマンドの期待される出力は次のようになります。
	`Launched Nucleus successfully`
		AWS 全般

IAM ポリシーを設定し、それを Greengrass サービスロールにアタッチします。	1. MQTT ブリッジとの間の通信を許可する IAM ポリシーを作成します。ポリシーの例を次に示します。 `{ "Version": "2012-10-17", "Statement": [`	AWS 全般

タスク	説明	必要なスキル

	<pre>{
	"Effect": "Allow",
	"Action": [
	"iot:*"
	],
	"Resource": "*"
	},
	{
	"Sid": "GreengrassActions",
	"Effect": "Allow",
	"Action": [
	"greengrass:*"
	],
	"Resource": "*"
	}
	]
	}</pre>	

	2. ポリシーを Greengrass サービスロールにアタッチします。サービスロールを取得するには、次のコマンドを使用します。 <pre>aws greengrassv2 get-service-role-for-account --region <region></pre>	

タスク	説明	必要なスキル
	where <region> はお客様の AWS リージョンを指します。	
AWS IoT Greengrass コアデバイスに必要なコンポーネントを設定してデプロイします。	次のコンポネントを設定、デプロイします。 • greengrass.clientdevices.mqtt.Moquette (「設定の詳細」を参照) • greengrass.clientdevices.mqtt.Bridge (「設定の詳細」と次のタスクを参照) • greengrass.clientdevices.Auth (「設定の詳細」と次のタスク以降のタスクを参照) • aws.greengrass.clientdevices.IPDetector (「設定の詳細を参照」)	AWS IoT Greengrass

タスク	説明	必要なスキル
MQTT ブリッジが双方向通信を許可していることを確認します。	クライアントデバイスと AWS IoT Core の間で MQTT メッセージをリレーするには、MQTT ブリッジコンポーネントを設定とデプロイして、リレーするトピックを指定します。例を示します。 `{ "mqttTopicMapping": { "ClientDevicesToCloud": { "topic": "dt/#", "source": "LocalMqtt", "target": "IotCore" }, "CloudToClientDevices": { "topic": "cmd/#", "source": "IotCore", "target": "LocalMqtt" } } }`	AWS IoT Greengrass

タスク	説明	必要なスキル

認証コンポーネントがクライアントデバイスに接続してトピックをパブリッシュまたはサブスクライブできることを確認します。	次の aws.greengrass.clientdevices.Auth 設定では、すべてのクライアントデバイスに接続したり、メッセージを公開したり、すべてのトピックをサブスクライブすることを許可しています。
	
	{
	"deviceGroups": {
	"formatVersion": "2021-03-05",
	"definitions": {
	"MyPermissiveDeviceGroup": {
	"selectionRule": "thingName: *",
	"policyName": "MyPermissivePolicy"
	}
	},
	"policies": {
	"MyPermissivePolicy": {
	"AllowAll": {
	"statementDescription": "Allow client devices to perform all actions.",
	"operations": [
	"*"
	],
	"resources": [
	"*"
	]
	}
	}
		AWS IoT Greengrass

タスク	説明	必要なスキル
	<pre> } } }</pre>	

クライアントデバイスをセットアップする

タスク	説明	必要なスキル
AWS IoT Device SDK をインストールします。	AWS IoT Device SDK をクライアントデバイスにインストールします。サポートされている言語と関連する SDK の全リストについては、「AWS IoT Core のキュメンテーション」 を参照してください。 たとえば、AWS IoT Device SDK for Python SDK for Python SDK は 「GitHub にあります」。この SDK をインストールするには: 1. GitHub リポジトリの 「前提条件ページ」 の指示に従って、Python 3.7 以降がインストールされていることを確認します。 2. pip コマンドを使用して、SDK をインストールします。 MacOS と Linux の場合: <pre>python3 -m pip install awsiotsdk</pre>	AWS 全般 IoT

タスク	説明	必要なスキル

	Windows の場合:
	`python -m pip install awsiotsdk`
	
	代わりに、SDK をソースリポジトリからインストールすることもできます。
	`# Create a workspace directory to hold all the SDK files`
	`mkdir sdk-workspace`
	`cd sdk-workspace`
	`# Clone the repository`
	`git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git`
	`# Install using Pip (use 'python' instead of 'python3' on Windows)`
	`python3 -m pip install ./aws-iot-device-sdk-python-v2`	

タスク	説明	必要なスキル
モノの作成	1. AWS IoT コンソールで、[Get started] (今すぐ始める) ボタンが表示された場合はそれをクリックします。それ以外の場合は、ナビゲーションペインで Security 、Policies を選択します。 2. ポリシーはまだ作成されていませんというダイアログボックスが表示された場合、ポリシーの作成を選択します。それ以外の場合は、[作成] を選択します。 3. AWS IoT ポリシーの名前 (例: `ClientDevicePolicy`) を入力します。 4. テートメントの追加セクションで、既存のポリシーを次の JSON コードに置き換えます。 `<region>` と `<account>` をお客様の AWS リージョンと AWS アカウント番号に置き換えます。 `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "iot:Connect",`	AWS IoT Core

タスク	説明	必要なスキル
	<pre>"Resource": "arn:aws:iot:region:account:client/*" }, { "Effect": "Allow", "Action": "iot:Publish", "Resource": "*" }, { "Effect": "Allow", "Action": "iot:Receive", "Resource": "*" }, { "Effect": "Allow", "Action": "iot:Subscribe", "Resource": "*" }, { "Effect": "Allow", "Action": ["iot:GetThingShadow", "iot:UpdateThingShadow", "iot:DeleteThingShadow"], "Resource": "arn:aws:iot:region:account:thing/*"</pre>	

タスク	説明	必要なスキル
	<pre> }] }</pre> 5. [Create] (作成) を選択します。 6. ナビゲーションペインの「AWS IoT コンソール」では、管理、モノの順に選択します。 7. まだモノがありませんというダイアログボックスが表示された場合、モノの登録を選択します。それ以外の場合は、[作成] を選択します。 8. [AWS IoT モノを作成する] ページで、[単一のモノを作成する] を選択します。 9. [Add your device to the device registry (デバイスレジストリへのデバイスの追加)] ページで、IoT モノの名前 (例: `ClientDevice1`) を入力し、[次へ] を選択します。 **Note** 作成後にモノの名前は変更できません。名前を変更するには、新しいモノを作成して、新しい名前を	

タスク	説明	必要なスキル
	付け、古いモノを削除する必要があります。 10[モノに証明書を追加] ページで、[証明書の作成] を選択します。 11[ダウンロード] リンクを選択して、証明書、プライベートキー、ルート CA 証明書をダウンロードします。 ⚠ Important これは、証明書とプライベートキーをダウンロードする唯一の機会です。 12証明書を有効にするには、[Activate] (有効化) を選択します。デバイスが AWS IoT に接続するには、証明書がアクティブである必要があります。 13[ポリシーのアタッチ] を選択します。 14.自分用のポリシーを追加」では、ClientDevice Policy、Thing の登録を選択します。	

タスク	説明	必要なスキル
CA 証明書を Greengrass コアデバイスからダウンロードします。	Greengrass Core デバイスがオフライン環境で動作することが予想される場合は、Greengrass コア CA 証明書をクライアントデバイスで使用できるようにする必要があります。これにより、クライアントデバイスが MQTT ブローカーの証明書 (Greengrass コア CA によって発行される) を検証できるようになります。そのため、この証明書のコピーを入手することが重要です。CA 証明書をダウンロードするには、次のいずれかの方法を使用します。 • PC から AWS IoT Greengrass デバイスにネットワークアクセスできる場合は、`https://<device IP>:8883` ウェブブラウザに入力して MQTT ブローカー証明書と CA 証明書を表示します。CA 証明書をクライアントデバイスに保存することもできます。 • または、 OpenSSL のコマンドラインを使用できます。 `openssl s_client -showcerts -connect <device IP>:8883`	AWS 全般

タスク	説明	必要なスキル
認証情報をクライアントデバイスにコピーします。	Greengrass コア CA 証明書、デバイス証明書、および秘密鍵をクライアントデバイスにコピーします。	AWS 全般

タスク	説明	必要なスキル
クライアントデバイスをコアデバイスと関連付けます。	クライアントデバイスをコアデバイスに関連付けて、コアデバイスを検出できるようにします。その後、「Greengrass ディスカバリー API」 を使用して、関連するコアデバイスの接続、情報と証明書を取得することができます。詳細については、AWS IoT Greengrass ドキュメントの 「クライアントデバイスを関連付ける」 を参照してください。 1. 「AWS IoT Greengrass コンソール」 で、コアデバイスを選択します。 2. 管理するコアデバイスを選択します。 3. コアデバイスの詳細ページで、[Client devices] (クライアントデバイス) タブを選択します。 4. [Associated client devices] (関連付けられているクライアントデバイス) セクションで、[Associate client devices] (クライアントデバイスを関連付ける) を選択します。 5. [Associate client devices with core device] (クライアントデバイスをコアデバイスに関連付ける) モーダル	AWS IoT Greengrass

タスク	説明	必要なスキル
	で、関連付ける各クライアントデバイスに対して次の操作を行います。 a. クライアントデバイスとして関連付ける AWS IoT の名前を入力します。 b. [追加] を選択します。 6. [関連付ける] を選択してください。 関連付けたクライアントデバイスで Greengrass ディスカバリ API を使用して、このコアデバイスを検出できるようになりました。	

データを送受信します。

タスク	説明	必要なスキル
あるクライアントデバイスから別のクライアントデバイスにデータを送信する。	デバイスの MQTT クライアントを使用して、`dt/client 1/sensor` トピックに関するメッセージを公開します。	AWS 全般
クライアントデバイスから AWS IoT Core にデータを送信します。	デバイスの MQTT クライアントを使用して、`dt/client 1/sensor` トピックに関するメッセージを公開します。 MQTT テストクライアントで、デバイスがメッセージを送信しているトピックを購読するか、すべてのトピックを	AWS 全般

タスク	説明	必要なスキル
	# に登録します (「詳細」 を参照)。	
AWS IoT Core からクライアントデバイスにメッセージを送信します。	MQTT テストクライアントページの トピックへの発行タブの トピック名フィールドに、メッセージのトピック名を入力します。この例では、`cmd/client1` トピックにを使用します。	AWS 全般

トラブルシューティング

問題	ソリューション
サーバー証明書エラーを確認できません。	このエラーは、TLS ハンドシェイク中に MQTT ブローカーが提示した証明書を MQTT クライアントが検証できない場合に発生します。最も一般的な理由は、MQTT クライアントが CA 証明書を持っていないことです。以下の手順に従って、CA 証明書が MQTT クライアントに提供されていることを確認します。 1. PC から AWS IoT Greengrass デバイスにネットワークアクセスできる場合は、`https://<device IP>:8883` ブラウザウィンドウに入力して MQTT ブローカー証明書と CA 証明書を表示します。CA 証明書をクライアントデバイスに保存することもできます。 または、OpenSSL コマンドラインを使用します。

問題	ソリューション

	`openssl s_client -showcerts -connect <device IP>:8883`
	2. Moquette CA 証明書と Greengrass Core CA 証明書の内容をファイルに保存し、コマンドを使用してデコードされた内容を表示します。
	`openssl x509 -in <Name of CA>.pem -text`
	Moquette CA 証明書には、次の例のように SAN フィールドが表示されるはずです。
	`X509v3 Subject Alternative Name: IP Address:XXX.XXX.XXX.XXX, IP Address:127.0.0.1, DNS:localhost`

サーバー名を確認できませんというエラーです。	このエラーは、MQTT クライアントが正しいサーバーに接続していることを検証できない場合に発生します。最も一般的な理由は、Greengrass デバイスの IP アドレスが証明書の SAN フィールドに記載されていないことです。 前のソリューションの指示に従って MQTT ブローカー証明書を取得し、「追加情報」セクションで説明したように、SAN フィールドに AWS IoT Greengrass デバイスの IP アドレスが含まれていることを確認します。そうでない場合は、IP ディテクターコンポーネントが正しくインストールされていることを確認し、コアデバイスを再起動します。

問題	ソリューション
組み込みクライアントデバイスから接続する場合のみサーバー名を確認できない	組み込みデバイスで使用される一般的な TLS ライブラリである Mbed TLS は、現在、Mbed TLS ライブラリコードに示されているように、証明書の SAN フィールドでの DNS 名検証のみをサポートしています。コアデバイスには独自のドメイン名がなく、IP アドレスに依存しているため、Mbed TLS を使用する TLS クライアントは TLS ハンドシェイク中にサーバー名の検証に失敗し、接続障害の原因となります。「x509_crt_check_san 関数」 を使用して Mbed TLS ライブラリに SAN IP アドレス検証を追加することをお勧めします。

関連リソース

- 「AWS IoT Greengrass ドキュメント」
- 「AWS IoT Core キュメント」
- 「MQTT ブローカーコンポーネント」
- 「MQTT ブリッジコンポーネント」
- 「クライアントデバイス認証コンポーネント」
- 「IP ディテクターコンポーネント」
- 「AWS IoT デバイス SDK」
- 「AWS IoT Greengrass によるローカルクライアントデバイスの実装」 (AWS ブログ記事)
- 「RFC 5280 — インターネット X.509 パブリックキーインフラストラクチャ証明書と証明書失効リスト (CRL) プロファイル」

追加情報

このセクションでは、クライアントデバイスとコアデバイスの間の通信に関する追加情報を提供します。

MQTT ブローカーはコアデバイスのポート 8883 で TLS クライアント接続を試行します。次の図は、MQTT ブローカーのサーバー証明書の例を示しています。

サンプル証明書には以下の詳細が表示されます。

- 証明書は AWS IoT Greengrass Core CA によって発行されます。この証明書はローカルでコアデバイスに固有です。つまり、ローカル CA として機能します。
- この証明書は、次の図に示す、クライアントの認証コンポーネントによって毎週自動的にローテーションされます。この間隔は、クライアント認証コンポーネントの設定で設定できます。

- サブジェクト代替名 (SAN) は TLS クライアント側でのサーバー名検証において重要な役割を果たします。これにより、TLS クライアントは正しいサーバーに接続でき、TLS セッション設定中の中間者攻撃を回避できます。サンプル証明書の SAN フィールドは、このサーバーがローカルホスト (ローカル UNIX ドメインソケット) をリッスンしていて、ネットワークインターフェースの IP アドレスが 192.168.1.12 であることを示しています。

TLS クライアントは、証明書の SAN フィールドを使用して、サーバー検証中に正規のサーバーに接続していることを確認します。これとは対照的に、HTTP サーバーとブラウザー間の通常の TLS ハンドシェイクでは、サーバー検証プロセス中に、共通名 (CN) フィールドまたは SAN フィールドのドメイン名を使用して、ブラウザーが実際に接続しているドメインをクロスチェックします。コアデバイスにドメイン名がない場合は、SAN フィールドに含まれる IP アドレスが同じ目的を果たします。詳細については、RFC 5280 — Internet X.509 公開鍵インフラストラクチャ証明書および証明書失効リスト (CRL) プロファイルの 「サブジェクト代替名」セクションを参照してください。

AWS IoT Greengrass の IP ディテクターコンポーネントは、証明書の SAN フィールドに正しい IP アドレスが含まれていることを保証します。

この例の証明書は、ローカル CA として動作する AWS IoT Greengrass デバイスによって署名されています。TLS クライアント (MQTT クライアント) はこの CA を認識しないため、次のような CA 証明書を提供する必要があります。

その他のパターン

- AWS IoT Greengrass を使用して IoT データをコスト効率よく直接 Amazon S3 に取り込む

移行とモダナイゼーション

トピック

- 移行
- モダナイゼーション
- メインフレーム

移行

トピック

- Microsoft Excel と Python を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成
- 自動ポートフォリオ検出の開始方法
- オンプレミスの Cloudera ワークロードを AWS 上の Cloudera データプラットフォームに移行する
- Microsoft SQL サーバーを AWS クラウドに移行した後に接続エラーを解決する
- RHEL ソースサーバーを再起動した後、SELinux を無効にせずに AWS レプリケーションエージェントを自動的に再起動する
- リアーキテクト
- リホスト
- リロケート
- リプラットフォーム
- ワークロード別の移行パターン
- その他のパターン

Microsoft Excel と Python を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成

作成者：Venkata Naveen Koppula (AWS)

概要

このパターンは、Microsoft Excel と Python を使用して「AWS Database Migration Service」(AWS DMS) 用の AWS CloudFormation テンプレートを自動的に作成する手順の概要を示しています。

AWS DMS を使用してデータベースを移行するには、通常、AWS DMS タスクをプロビジョニングするための AWS CloudFormation テンプレートの作成が必要です。従来、AWS CloudFormation テンプレートを作成するには JSON または YAML プログラミング言語に関する知識が必要です。このツールで、Excel の基本知識とターミナルまたはコマンドウィンドウを使用して Python スクリプトを実行する方法のみ必要です。

このツールは入力として、移行するテーブルの名前、AWS DMS エンドポイントの Amazon リソースネーム (ARN) および AWS DMS レプリケーションインスタンスを含む Excel ワークブックを使用します。その後、このツールは、必要な AWS DMS タスク用の AWS CloudFormation テンプレートを生成します。

詳細な手順と背景情報については、AWS Database ブログの「Microsoft Excel を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成」というブログ記事を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Microsoft Excel バージョン 2016 以降
- Python バージョン 2.7 以降
- [xlrd] Python モジュール (コマンドプロンプトで [pip install xlrd] コマンドを使用してインストール済み)
- AWS DMS ソースとターゲットエンドポイント、AWS DMS レプリケーションインスタンス

制約事項

- スキーマ、テーブルと関連する列の名前は、宛先エンドポイントで小文字に変換されます。
- このツールは、AWS DMS エンドポイントとレプリケーションインスタンスの作成に関する問題に対応していません。
- 現在、このツールは AWS DMS タスクごとに 1 つのスキーマのみサポートしています。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスのデータベース
- Microsoft Excel

ターゲットテクノロジースタック

- AWS CloudFormation テンプレート
- AWS クラウド内のデータベース

アーキテクチャ

ツール

- 「Pycharm IDE」または Python バージョン 3.6 をサポートする任意の統合開発環境 (IDE)
- Microsoft Office 2016 (Microsoft Excel 用)

エピック

ネットワーク、AWS DMS レプリケーションインスタンス、エンドポイントの設定

タスク	説明	必要なスキル
必要に応じて、サービスクォータの増加をリクエストします。	必要に応じて、AWS DMS タスク用の Service Quotas の増加をrリクエストします。	AWS 全般

タスク	説明	必要なスキル
AWS リージョン、仮想プライベートクラウド (VPC)、CIDR 範囲、アベイラビリティーゾーンとサブネットを設定します。		AWS 全般
AWS DMS レプリケーションインスタンスを設定します。	AWS DMS レプリケーションインスタンスは、オンプレミスと AWS データベースの両方に接続できます。	AWS 全般
AWS DMS エンドポイントを設定します。	ソースデータベースとターゲットデータベースの両方のエンドポイントを構成します。	AWS 全般

AWS DMS タスクとタグ用のワークシートを準備

タスク	説明	必要なスキル
テーブルリストを設定します。	移行に関係するテーブルをすべて一覧表示します。	データベース
タスクワークシートを準備します。	設定したテーブルリストを使用して Excel ワークシートを準備します。	AWS 全般、Microsoft Excel
タグワークシートを準備します。	AWS DMS タスクに添付する AWS リソースタグの詳細を示します。	AWS 全般、Microsoft Excel

ツールをダウンロードして実行します。

タスク	説明	必要なスキル
GitHub リポジトリからテンプレート生成ツールをダウンロードして抽出します。	GitHub リポジトリ：https://github.com/aws-samples/dms-cloudformation-templates-generator/	
ツールを実行します。	「参考資料とヘルプ」に記載されているブログ記事の詳細な手順に従ってください。	

関連リソース

- Microsoft Excel を使用したAWS DMSタスク用の AWS CloudFormation テンプレートの作成(ブログ記事)
- 「DMS CloudFormation Templates Generator (GitHub リポジトリ)」
- 「Python のドキュメンテーション」
- 「xlrd の説明とダウンロード」
- AWS DMS のドキュメント
- 「AWS CloudFormation のドキュメント」

自動ポートフォリオ検出の開始方法

作成者:プラティック・チュナワラ (AWS) とロドルフォ・ジュニア・セラダ (AWS)

概要

アプリケーションとサーバーを Amazon Web Services (AWS) クラウドに移行する場合、特にサーバーが 300 を超える大規模な移行の場合、ポートフォリオを評価してメタデータを収集するのは重要な課題です。自動ポートフォリオ検出ツールを使用すると、ユーザー数、使用頻度、依存関係、アプリケーションのインフラストラクチャに関する情報など、アプリケーションに関する情報を収集するのに役立ちます。この情報は、類似の特性を持つアプリケーションを適切に優先順位付けしてグループ化できるように、移行の段階を計画する際に不可欠です。検出ツールを使用すると、ポートフォリオチームが手動でメタデータを収集しなくても検出ツールの結果を検証できるため、ポートフォリオチームとアプリケーション所有者間のコミュニケーションが効率化されます。このパターンでは、自動検出ツールを選択する際の主な考慮事項と、それを環境内にデプロイしてテストする方法に関する情報について説明します。

このパターンには、大まかなアクティビティをまとめた独自のチェックリストを作成するための出発点となるテンプレートが含まれています。チェックリストの隣には、実行責任者、説明責任者、相談先、報告先 (RACI) のマトリックス用のテンプレートがあります。この RACI マトリックスを使用して、チェックリストの各タスクの責任者を決定できます。

エピック

ディスカバリーツールの選択

タスク	説明	必要なスキル
検出ツールがユースケースに適しているかどうかを判断します。	ディスカバリーツールはユースケースに最適なソリューションではないかもしれません。検出ツールの選択、調達、準備、導入に必要な時間を考慮してください。環境内のエージェントレス検出ツール用にスキャニングアプライアンスをセットアップしたり、対象範囲内のすべてのワークロードにエージェント	移行リーダー、移行エンジニア

タスク	説明	必要なスキル
	をインストールしたりするには、4 ~ 8 週間かかる場合があります。導入後、検出ツールがアプリケーションワークロードをスキャンし、アプリケーションスタック分析を実行してメタデータを収集するまでに 4 ~ 12 週間かかります。移行するサーバーが 100 台未満の場合は、自動検出ツールによるメタデータのデプロイと収集に必要な時間よりも早く、メタデータを手動で収集して依存関係を分析できる可能性があります。	
ディスカバリーツールを選択します。	「追加情報」セクションの「自動検出ツールを選択する際の考慮事項」を確認してください。ユースケースに合った検出ツールを選択するための適切な基準を決定し、その基準に照らして各ツールを評価します。自動検出ツールの包括的なリストについては、「検出、計画、推奨移行ツール」を参照してください。	移行リーダー、移行エンジニア

インストールの準備

タスク	説明	必要なスキル
導入前チェックリストを準備する。	ツールをデプロイする前に完了しなければならないタスクのチェックリストを作成しま	構築リード、移行エンジニア、移行リーダー、ネットワーク管理者

タスク	説明	必要なスキル
	す。例については、Flexera ドキュメンテーション Web サイトの「導入前チェックリスト」を参照してください。	
ネットワーク要件を準備する。	ツールを実行してターゲットサーバーにアクセスするために必要なポート、プロトコル、IP アドレス、ルーティングをプロビジョニングします。詳細については、検出ツールのインストールガイドを参照してください。例については、フレクセラのドキュメンテーションウェブサイトの「導入要件」を参照してください。	移行エンジニア、ネットワーク管理者、クラウドアーキテクト
アカウントと認証情報の要件を準備します。	対象サーバーにアクセスし、ツールのすべてのコンポーネントをインストールするのに必要な認証情報を特定します。	クラウド管理者、AWS 全般、移行エンジニア、移行リーダー、ネットワーク管理者、AWS 管理者
ツールをインストールするアプライアンスを準備します。	ツールコンポーネントをインストールするアプライアンスがツールの仕様とプラットフォーム要件を満たしていることを確認してください。	移行エンジニア、移行リーダー、ネットワーク管理者
変更指示書を準備します。	組織の変更管理プロセスに従って、必要な変更指示を準備し、これらの変更指示が承認されていることを確認します。	ビルドリード、移行リード

タスク	説明	必要なスキル
利害関係者に要件を送ります。	導入前のチェックリストとネットワーク要件を利害関係者に送信します。利害関係者は、導入を進める前に必要な要件を確認、評価、準備する必要があります。	ビルドリード、移行リード

ツールをデプロイ

タスク	説明	必要なスキル
インストーラをダウンロードします。	インストーラーまたは仮想マシンイメージをダウンロードします。仮想マシンのイメージは通常、オープン仮想化フォーマット (OVF) で提供されます。	ビルドリード、移行リード
ファイルを展開します。	インストーラーを使用している場合は、インストーラーをオンプレミスサーバーにダウンロードして実行する必要があります。	ビルドリード、移行リード
ツールをサーバーにデプロイします。	次のように、対象のオンプレミスサーバーに検出ツールをデプロイします。 • ソースファイルが仮想マシンイメージの場合は、VMware などの仮想マシン環境にデプロイします。 • ソースファイルがインストーラーの場合は、インス	構築リーダー、移行リーダー、ネットワーク管理者

タスク	説明	必要なスキル
	トーラーを実行してツールをインストールして設定します。	
検出ツールにログインします。	画面に表示される指示に従って、ログインしてツールの開始方法	移行リード、構築リード
製品をアクティベートします。	ライセンスキーを入力します。	ビルドリード、移行リード
ツールを設定します。	Windows、VMware、簡易ネットワーク管理プロトコル (SNMP)、セキュアシェルプロトコル (SSH)、またはデータベースの認証情報など、ターゲットサーバーへのアクセスに必要な認証情報を入力します。	ビルドリード、移行リード

ツールをテストしてください。

タスク	説明	必要なスキル
テストサーバーを選択してください。	検出ツールのテストに使用できる、本番用ではないサブネットまたは IP アドレスをいくつか特定します。これにより、スキャンを迅速に検証し、エラーを迅速に特定してトラブルシューティングし、テストを本番環境から切り離すことができます。	構築リーダー、移行リーダー、ネットワーク管理者

タスク	説明	必要なスキル
選択したテストサーバーのスキャンを開始します。	エージェントレス検出ツールの場合は、選択したテストサーバーのサブネットまたは IP アドレスを検出ツールコンソールに入力し、スキャンを開始します。 エージェントベースの検出ツールの場合は、選択したテストサーバーにエージェントをインストールします。	構築リーダー、移行リーダー、ネットワーク管理者
スキャン結を確認します。	テストサーバーのスキャン結果を確認します。エラーが見つかった場合は、エラーのトラブルシューティングと修正を行います。エラーと解決策を文書化してください。将来この情報を参照し、ポートフォリオランブックにこの情報を追加できます。	構築リーダー、移行リーダー、ネットワーク管理者
テストサーバーを再スキャンしてください。	再スキャンが完了したら、エラーがなくなるまでスキャンを繰り返します。	構築リーダー、移行リーダー、ネットワーク管理者

関連リソース

AWS リソース

- AWS クラウド 移行のためのアプリケーションポートフォリオ評価ガイド
- 「検出、計画、推奨の移行ツール」

よく使われるディスカバリーツールの導入ガイド

- 「RN150 仮想アプライアンスをデプロイします」(Flexera ドキュメント)
- 「ギャザラーのインストール」(ModelizeIT ドキュメント)
- 「オンプレミス分析サーバーのインストール」(ModelizeIT ドキュメント)

追加情報

自動検出ツールを選択する際の考慮事項

検出ツールにはそれぞれ利点と制限があります。ユースケースに適したツールを選択する際には、次の点を考慮してください。

- ポートフォリオ評価の目標を達成するのに必要なメタデータのすべてではないにしても、ほとんどを収集できる発見ツールを選択してください。
- ツールではサポートされていないため、手動で収集する必要のあるメタデータを特定してください。
- 検出ツールの要件を利害関係者に提供し、関係者がサーバー、ネットワーク、認証情報の要件など、社内のセキュリティ要件とコンプライアンス要件に基づいてツールを見直し、評価できるようにします。
  - ツールでは、対象となるワークロードにエージェントをインストールする必要がありますか?
  - このツールでは、環境内に仮想アプライアンスを設定する必要がありますか?
- データレジデンシーの要件を決定してください。データを環境外に保存したくない組織もあります。これに対処するには、ツールの一部のコンポーネントをオンプレミス環境にインストールする必要がある場合があります。
- ツールが対象ワークロードのオペレーティングシステム (OS) と OS バージョンをサポートしていることを確認してください。
- ポートフォリオにメインフレーム、ミッドレンジ、レガシーサーバーが含まれているかどうかを確認してください。ほとんどの検出ツールはこれらのワークロードを依存関係として検出できますが、使用率やサーバーの依存関係など、デバイスの詳細を取得できないツールもあります。Device42 と ModernizeIT の検出ツールはどちらもメインフレームとミッドレンジのサーバーをサポートしています。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

オンプレミスの Cloudera ワークロードを AWS 上の Cloudera データプラットフォームに移行する

作成者: Battulga Purevragchaa (AWS)、Nijjwol Lamsal (パートナー）、Nidhi Gupta (AWS)

概要

このパターンでは、オンプレミスの Cloudera 分散 Hadoop (CDH)、Hortonworks データプラットフォーム (HDP)、および Cloudera データプラットフォーム (CDP) のワークロードを AWS 上の CDP パブリッククラウドに移行するための概要レベルの手順を説明しています。Cloudera プロフェッショナルサービスおよびシステムインテグレーター (SI) と提携して、これらのステップを実装することをお勧めします。

Cloudera のお客様がオンプレミスの CDH、HDP、CDP のワークロードをクラウドに移行したいと思う理由はたくさんあります。一般的な理由は以下のとおりです。

- データレイクハウスやデータメッシュなどの新しいデータプラットフォームパラダイムの採用を効率化します。
- ビジネスの俊敏性を高め、既存のデータ資産へのアクセスと推論を民主化します。
- 総保有コスト (TCO) が低くなります
- ワークロードの伸縮自在性を強化
- 従来のオンプレミスのインストールベースと比較して、スケーラビリティを高め、データサービスのプロビジョニングにかかる時間を大幅に短縮できます。
- レガシーハードウェアを廃止、ハードウェアの更新サイクルを大幅に短縮
- 従量制料金を活用してください。この価格は、Cloudera ライセンスモデル (CCU) を使用した AWS 上の Cloudera ワークロードにも適用されます。
- 継続的インテグレーションおよび継続的デリバリー (CI/CD) のプラットフォームにより、より迅速な導入や統合の強化を活用する
- 単一の統合プラットフォーム (CDP) で複数のワークロードに対応

Clouderaは、機械学習、データエンジニアリング、データウェアハウス、オペレーショナルデータベース、ストリーム処理 (CSP)、データセキュリティとガバナンスなど、主要なワークロードをすべてサポートします。Cloudera はこれらのワークロードを長年にわたりオンプレミスで提供してきました。ワークロードマネージャーとレプリケーションマネージャーを備えた CDP パブリッククラウドを使用することで、これらのワークロードを AWS クラウドに移行できます。

Cloudera Shared Data Experience (SDX) では、これらのワークロード全体で共有メタデータカタログが提供されるため、一貫したデータ管理と運用が容易になります。SDX には、脅威から保護するための包括的できめ細かなセキュリティと、ペイメントカード業界データセキュリティ標準 (PCI DSS) や GDPR などの標準に準拠するための監査および検索機能の統合ガバナンスも含まれています。

一見してのCDP の移行

ワークロード	ソースワークロード	CDH、HDP、および CDP プライベートクラウド
	ソース環境	• Windows、Linux • オンプレミス、コロケーション、または AWS 以外の環境
	送信先ワークロード	AWS 上の CDP パブリッククラウド
	送信先環境	• 導入モデル:顧客アカウント • 運用モデル:カスタマー/Cloudera コントロールプレーン
移行	移行戦略 (「7Rs」)	リホスト、リプラットフォーム、リファクタリング
	これはワークロードのバージョンアップですか？	はい
	移行期間	• 導入:顧客アカウント、仮想プライベートクラウド (VPC)、CDP Public Cloud 顧客管理環境の作成に約 1 週間かかります。 • 移行期間:ワークロードの複雑さと規模にもよりますが、1 ~ 4 か月。

コスト	AWS でのワークロードの実行コスト	• 高いレベルでは、AWS への CDH ワークロード移行のコストは、AWS 上に新しい環境を構築することを前提としている。これには、新しい環境向けのコンピューティングリソースとライセンスソフトウェアのプロビジョニングに加えて、人員の時間と労力を考慮することが含まれます。 • Cloudera のクラウド使用量ベースの価格モデルでは、バースト機能と自動スケーリング機能を柔軟に活用できます。詳細については、Cloudera ウェブサイトの「CDP パブリッククラウドサービス料金」をご覧ください。 • Cloudera のエンタープライズ「データハブ」は Amazon Elastic Compute Cloud (Amazon EC2) をベースに、従来のクラスターを忠実にモデル化しています。データハブは「カスタマイズ」可能ですが、これはコストに影響します。 • 「CDP パブリッククラウドデータウェアハウス」、「Cloudera 機械学習」、および「Cloudera データエンジニアリング (CDE)」はコンテナベースで、自動的に

		スケーリングするように設定できます。
インフラストラクチャー契約とフレームワーク	システム要件	「前提条件」セクションを参照してください。
	SLA	「CDP パブリッククラウドに関する Cloudera サービスレベル契約」を参照してください。
	DR	Cloudera ドキュメントの「ディザスタリカバリ」を参照してください。
	ライセンスと運用モデル (ターゲット AWSアカウントの場合)	Bring Your Own License (BYOL) モデル
コンプライアンス	セキュリティ要件	Cloudera ドキュメントの「Cloudera セキュリティの概要」を参照してください。
	その他の「コンプライアンス認証」	「一般データ保護規則 (GDPR)」の遵守と「CDP トラストセンター」については、Cloudera のウェブサイトに掲載されている情報をご覧ください。

前提条件と制限

前提条件

- アカウント、リソース、サービス、アクセス許可 (AWS ID & アクセス管理 (IAM) ロールやポリシー設定など) を含む「AWS アカウント要件」
- Cloudera ウェブサイトから「CDP をデプロイするための前提条件」

移行には以下の役割と専門知識が必要です。

ロール	スキルと責任
移行リード	経営陣のサポート、チームコラボレーション、計画、実装、評価を保証する
Cloudera SME	CDH、HDP、CDP の管理、システム管理、アーキテクチャに関する専門スキル
AWS アーキテクト	AWS のサービス、ネットワーク、セキュリティ、アーキテクチャのスキル

アーキテクチャ

適切なアーキテクチャを構築することは、移行とパフォーマンスを確実に期待に応えるための重要なステップです。移行作業がこのプレイブックの前提条件を満たすようにするには、仮想プライベートクラウド (VPC) ホストインスタンスまたは CDP 上の AWS クラウド内のターゲットデータ環境が、オペレーティングシステムとソフトウェアのバージョン、および主要なマシン仕様の点でソース環境と同等である必要があります。

以下の図 (「Cloudera Shared Data Experience データシート」から許可を得て複製) は、CDP 環境のインフラストラクチャコンポーネントと、階層またはインフラストラクチャコンポーネントがどのように相互作用するかを示しています。

アーキテクチャには以下の CDP コンポーネントが含まれます。

- データハブは Cloudera Runtime を搭載したワークロードクラスターを起動および管理するためのサービスです。データハブのクラスター定義を使用して、カスタムユースケース向けにワークロードクラスターをプロビジョニングしてアクセスしたり、カスタムクラスター構成を定義したりできます。詳細については、「Cloudera のウェブサイト」を参照してください。
- データフローとストリーミングは、データが移動する中で企業が直面する主な課題に対処します。以下のような管理をしています。
  - 高ボリュームおよび高スケールでのリアルタイムデータストリーミングの処理
  - ストリーミングデータの出所とリネージのトラッキング
  - エッジアプリケーションとストリーミングソースの管理と監視

詳細については、Cloudera ウェブサイトの「Cloudera DataFlow」と「CSP」を参照してください。

- データエンジニアリングには、組織がデータパイプラインとワークフローを構築し維持するのに役立つデータ統合、データ品質、データガバナンスが含まれます。詳細については、「Cloudera のウェブサイト」を参照してください。Cloudera Data Engineering ワークロードの「AWS でのコスト削減を促進するスポットインスタンスのサポート」についてご覧ください。
- Data Warehouse を利用すると、ワークロードの需要に応じて自動的にスケーリングする独立したデータウェアハウスとデータマートを作成できます。このサービスでは、データウェアハウスとデータマートごとに独立したコンピューティングインスタンスと自動最適化が可能になり、SLA を満たしながらコストを節約できます。詳細については、「Cloudera のウェブサイト」を参照してください。AWS 上の Cloudera データウェアハウスの「コスト管理」と「自動スケーリング」について学びましょう。
- CDP のオペレーショナルデータベースは、スケーラブルで高性能なアプリケーションのための信頼性が高く柔軟な基盤を提供します。運用とウェアハウジングの統合プラットフォーム内で、従来の構造化データだけでなく新しい非構造化データも提供する、リアルタイムで、いつでも利用可能な、スケーラブルなデータベースを実現します。詳細については、「Cloudera のウェブサイト」を参照してください。
- Machine Learning は、セルフサービスのデータサイエンスとデータエンジニアリング機能をエンタープライズデータクラウド内の単一のポータブルサービスに統合するクラウドネイティブな機械学習プラットフォームです。機械学習と人工知能 (AI) をどこにでもデータにスケーラブルに展開できます。詳細については、「Cloudera のウェブサイト」を参照してください。

AWS 上の CDP

以下の図 (Cloudera ウェブサイトからの許可を得て改変) は、AWS 上の CDP の大まかなアーキテクチャを示しています。CDP は「独自のセキュリティモデル」を実装して、アカウントとデータフローの両方を管理しています。これらは「クロスアカウントロール」を使用して「IAM」と統合されます。

CDP コントロールプレーンは、独自の VPC の Cloudera マスターアカウントにあります。各顧客アカウントには独自のサブアカウントと固有の VPC があります。クロスアカウントIAMロールとSSLテクノロジーは、コントロールプレーン間の管理トラフィックを、各顧客VPC 内のインターネットでルーティング可能なパブリックサブネットにあるカスタマーサービスにルーティングします。お客様のVPC では、Cloudera Shared Data Experience (SDX) が統合ガバナンスとコンプラ

イアンスを備えたエンタープライズクラスのセキュリティを実現し、データからより迅速に洞察を引き出すことができます。SDX は Cloudera のすべての製品に組み込まれている設計哲学です。「SDX」と「AWS 向けの CDP パブリッククラウドネットワークアーキテクチャ」の詳細については、「Cloudera のドキュメント」を参照してください。

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「Amazon Elastic Kubernetes Service (Amazon EKS)」は、AWS で Kubernetes を実行する際に役立ち、独自の Kubernetes コントロールプレーンまたはノードをインストールまたは維持する必要はありません。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

オートメーションとツール

- その他のツールとしては、「Cloudera Backup データリカバリ (BDR)」、「AWS Snowball」、「AWS Snowmobile」を使用して、オンプレミスの CDH、HDP、CDP から AWS がホストする CDP へのデータ移行を支援できます。
- 新規導入には、「CDP 用 AWS パートナーソリューション」を使用することをお勧めします。

エピック

移行の準備をする

タスク	説明	必要なスキル
Cloudera チームと連携してください。	Cloudera は顧客との標準化されたエンゲージメントモデ	移行リード

タスク	説明	必要なスキル
	ルを追求しており、貴社のシステムインテグレーター (SI) と協力して同じアプローチを推進することができます。Cloudera のカスタマーチームに連絡すれば、プロジェクトを開始するためのガイダンスや必要な技術リソースを提供してもらえます。Cloudera チームに連絡することで、移行日が近づいたら、必要なすべてのチームが移行の準備を整えることができます。 Cloudera プロフェッショナルサービスに連絡して、Cloudera の導入をパイロット版から本番環境に迅速に、低コストで、最高のパフォーマンスで移行できます。サービスの詳細なリストについては、「Cloudera ウェブサイト」を参照してください。	
VPC 用の CDP パブリッククラウド環境を AWS 上に作成します。	Cloudera プロフェッショナルサービスまたは SI と連携して CDP パブリッククラウドを計画し、AWS 上の VPC にデプロイします。	Cloudera 中小企業のクラウドアーキテクト

タスク	説明	必要なスキル
移行するワークロードに優先順位を付け、評価する。	オンプレミスのすべてのワークロードを評価して、移行が最も簡単なワークロードを決定します。ミッションクリティカルではないアプリケーションは、顧客への影響が最小限になるため、最初に移行するのが最適です。ミッションクリティカルなワークロードは、他のワークロードの移行に成功したら、最後に保存しておきます。 **Note** 一時的な (CDP データエンジニアリング) ワークロードは、永続的な (CDP データウェアハウス) ワークロードよりも移行が容易です。移行の際には、データ量と場所を考慮することも重要です。課題としては、データをオンプレミス環境からクラウドに継続的に複製することや、データをクラウドに直接インポートするようにデータインジェストパイプラインを変更することが挙げられます。	移行リード

タスク	説明	必要なスキル
CDH、HDP、CDP、レガシーアプリケーションの移行アクティビティについて話し合う。	Cloudera ワークロードマネージャーを使用して、以下のアクティビティを検討し、計画を開始してください。 • AWS 環境にコピーするデータとワークロード • クラウド対応データ • リソースを使い果たし、他のテナントに迷惑をかける騒がしい隣人 • 伸縮自在なワークロード • 運用上のオーバーヘッドが高い小規模クラスター	移行リード

タスク	説明	必要なスキル
Cloudera レプリケーションマネージャーの要件と推奨事項をすべて記入してください。	Cloudera プロフェッショナルサービスおよび SI と協力して、AWS 上の CDP パブリッククラウド環境にワークロードを移行する準備をしてください。 以下の要件と推奨事項を理解しておくと、Replication Manager サービスのインストール中およびインストール後に発生する一般的な問題を回避するのに役立ちます。 • Replication Managerのサポートドキュメントを確認して、環境とシステムの要件を満たしていることを確認してください。詳細については、Cloudera ウェブサイトの「CDP パブリッククラウドレプリケーションマネージャーサポートマトリックス」を参照してください。 • Replication Manager アプリとデータライフサイクルマネージャ (DLM) エンジンがインストールされるノードへのルートアクセスは必要ありません。 • 将来的に Hive レプリケーションを使用しないことが確実でない限り、Replication Manager の初回インストール時に Apache	移行リード

タスク	説明	必要なスキル
	Hive をインストールします。Replication Manager で HDFS レプリケーションポリシーを作成した後で Hive をインストールする場合は、Hive を追加した後ですべての HDFS レプリケーションポリシーを削除してから再作成する必要があります。 • Replication Manager で使用されるクラスターは、対称的な構成になっている必要があります。レプリケーション・リレーションシップ内の各クラスターは、セキュリティ (Kerberos)、ユーザー管理 (LDAP/AD)、Knox Proxyに関してまったく同じように構成されている必要があります。Hadoop 分散ファイルシステム (HDFS)、Apache Hive、Apache Knox、Apache Ranger、Apache Atlas などのクラスターサービスは、高可用性 (HA) を実現するために異なる構成にすることができます。たとえば、ソースクラスターとターゲットクラスターには HA 構成と非 HA 構成が別々になっている場合があります。	

CDP を AWS に移行する

タスク	説明	必要なスキル
Cloudera ワークロードマネージャーを使用して、開発/テスト環境の最初のワークロードを移行します。	SI は、最初のワークロードを AWS クラウドに移行するのに役立ちます。これは、顧客向けのものでもミッションクリティカルなものでもないアプリケーションでなければなりません。開発/テスト移行の理想的な候補は、CDP Data Engineering ワークロードなど、クラウドで簡単にデータを取り込めるアプリケーションです。これは、中断のないアクセスを必要とする多数のユーザーがいる可能性のある CDP データウェアハウスワークロードのような永続的なワークロードと比較して、通常はアクセスするユーザーが少ない一時的なワークロードです。データエンジニアリングのワークロードは永続的ではないため、何か問題が発生した場合のビジネスへの影響を最小限に抑えることができます。ただし、これらのジョブはプロダクションレポートにとって重要になる可能性があるため、影響の少ないデータエンジニアリングのワークロードを最初に優先してください。	移行リード

タスク	説明	必要なスキル
必要に応じて移行手順を繰り返します。	Cloudera ワークロードマネージャーは、クラウドに最適なワークロードを特定するのに役立ちます。クラウドのパフォーマンス評価、ターゲット環境のサイジング/キャパシティプラン、レプリケーションプランなどの指標を提供します。移行に最適な候補は、季節的なワークロード、臨時のレポート、リソースをあまり消費しない断続的なジョブです。 Cloudera Replication Manager は、データをオンプレミスからクラウドへ、そしてクラウドからオンプレミスへと移動します。 ワークロードマネージャーを使用して、データウェアハウス、データエンジニアリング、機械学習のワークロード、アプリケーション、パフォーマンス、インフラストラクチャ容量をプロアクティブに最適化します。データウェアハウスをモダナイズする方法の詳細なガイドについては、「Cloudera ウェブサイト」をご覧ください。	Cloudera SME

関連リソース

Cloudera ドキュメント

- 「CDP、Cloudera マネージャー、レプリケーションマネージャーへのクラシッククラスターの登録:」
  - 「マネジメントコンソール」
  - 「レプリケーションマネージャーハイブレプリケーション」
- 「Sentry レプリケーション」
- 「Sentry アクセス許可」
- 「データハブクラスター計画チェックリスト」
- 「ワークロードマネージャーアーキテクチャ」
- 「レプリケーションマネージャー要件」
- 「Cloudera データプラットフォームのオブザーバビリティ」
- 「AWS 要件」

AWS ドキュメント

- クラウドへのデータ移行

Microsoft SQL サーバーを AWS クラウドに移行した後に接続エラーを解決する

作成者: Premkumar Chelladurai (AWS)

概要

Windows Server 2008 R2、2012、または 2012 R2 で実行中の Microsoft SQL Server をAmazon Web Services (AWS) クラウド上の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに移行後、SQL Server への接続が失敗し、次のエラーが表示されます。

- `[Microsoft][ODBC SQL Server Driver][DBNETLIB] General Network error`
- `ERROR [08S01] [Microsoft][SQL Native Client]Communication link failure. System.Data.SqlClient.SqlException: A transport-level error has occurred when sending the request to the server. (provider: TCP Provider, error: 0 - An existing connection was forcibly closed by the remote host.)`
- `TCP Provider: The semaphore timeout period has expired`

このパターンでは、Windows Server 2008 R2、2012、または 2012 R2 で実行中の SQL Server のオペレーティングシステム (OS) とネットワークインターフェイスレベルで Windows スケーラブルネットワークパック (SNP) 機能をオフにすることでこれらのエラーを解決する方法を説明します。

前提条件と制限

前提条件

- Windows サーバーの管理者特権。
- AWS アプリケーション移行サービスを移行ツールとして使用した場合は、次の Windows Server バージョンのいずれかが必要です。
  - Windows Server 2008 R2 サービスパック 1、2012、または 2012 R2
- CloudEndure Migration を移行ツールとして使用した場合は、次の Windows Server バージョンのいずれかが必要です。
  - Windows Server 2003 R2 Service Pack 3、2008、2008 R2 サービスパック 1、2012、または 2012 R2

ツール

- 「Amazon EC2」— Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動し、スケールアウトまたはスケールインできます。
- Windows サーバー — Windows Server は、接続されたアプリケーション、ネットワーク、および Web サービスのインフラストラクチャを構築するプラットフォームです。

エピック

OS レベルと伸縮性ネットワークインターフェイスレベルで SNP 機能をオフにする

タスク	説明	必要なスキル
OS レベルで SNP 機能をオフにします。	1. Windows Server にサインインし、管理者としてコマンドプロンプトを開きます。 2. `netsh int tcp show global` コマンドを実行します。 3. 出力で、`Receive-Side Scaling` 、`Chimney Offload` または `enabled` がモードになっているかどうかを確認します。いずれかが `enabled` の場合は、次のコマンドを実行します。 • `netsh int tcp set global chimney=disabled` • `netsh int tcp set global rss=disabled`	AWS 管理者、AWS システム管理者、移行エンジニア、クラウド管理者

タスク	説明	必要なスキル
伸縮性のあるネットワークインターフェイスレベルで SNP 機能をオフにします。	1. [Start (スタート)] を選択してから、[Enter (実行キー)] を押します。 2. [Elastic Network Adapter (伸縮性ネットワークアダプター)] を右クリックします。 3. ポップアップメニューで、[Properties (プロパティ)] を選択します。 4. [Ethernet Adapter Properties (イーサネットアダプターのプロパティ)] ウィンドウで、[Configure (設定)] を選択します。 5. [Amazon Elastic Network Adapter Properties (Amazon 伸縮性ネットワークアダプターのプロパティ)] ウィンドウで、[Advanced (詳細設定)] タブを選択します。 6. プロパティセクションで、オフロードと RSS をすべてオフにします。	AWS 管理者、クラウド管理者、AWS システム管理者

関連リソース

- RSS、NetDMA などの高度なネットワークパフォーマンス機能のトラブルシューティング方法

RHEL ソースサーバーを再起動した後、SELinux を無効にせずに AWS レプリケーションエージェントを自動的に再起動する

アニル・クナパレディ(AWS)、シャンムガム・シャンカー(AWS)、ベンカトラマナ・チンサ(AWS)によって作成されました

概要

AWS アプリケーション移行サービスは、Red Hat Enterprise Linux (RHEL) ワークロードのAmazon Web Services (AWS) クラウドへの移行を簡素化、促進、自動化するのに役立ちます。ソースサーバーをアプリケーション移行サービスに追加するには、サーバーに AWS Replication エージェントをインストールします。

アプリケーション移行サービスでは、ブロックレベルの非同期レプリケーションをリアルタイムで行うことができます。つまり、レプリケーションプロセス全体を通して通常の IT 運用を継続できるということです。 これらの IT 運用では、移行中に RHEL ソースサーバーの再起動または再起動が必要になる場合があります。この場合、AWS Replication エージェントは自動的に再起動せず、データレプリケーションは停止します。通常、セキュリティ強化版 Linux (SELinux) を無効モードまたは許可モードに設定すると、AWS レプリケーションエージェントが自動的に再起動されます。ただし、組織のセキュリティポリシーで SELinux の無効化が禁止されている場合や、ファイルのラベルを変更 しなければならない場合もあります。

このパターンは、RHEL ソースサーバーが再起動または移行中に再起動したときに SELinux をオフにせずに AWS Replication エージェントを自動的に再起動する方法を示しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS クラウドに移行したいオンプレミスの RHEL ワークロード。
- アプリケーション移行サービスコンソールから初期化されたアプリケーション移行サービス。初期化が必要なのは、このサービスを初めて使用するときだけです。手順については、AWS Service Catalog のドキュメントを参照してください。
- Application Migration Service 用の既存の AWS Identity and Access Management (IAM) ポリシー。詳細については、Android アプリケーション ID のドキュメントを参照してください。

バージョン

- RHEL バージョン 7 以降

ツール

AWS サービス

- AWS Application Migration Service は、高度に自動化されたリフトアンドシフト (リホスト) ソリューションです。これにより、AWS へのアプリケーションの移行を簡略化、迅速化し、コストを削減します。

Linux コマンド

次の表は、RHEL ソースサーバーで実行する Linux コマンドの一覧です。これらは、このパターンのエピックやストーリーにも記載されています。

コマンド	説明
`#systemctl –version`	システムバージョンを識別します。
`#systemctl list-units --type=service`	RHEL サーバーで利用できるすべてのアクティブなサービスを一覧表示します。
`#systemctl list-units --type=service | grep running`	RHEL サーバー上で現在実行中のすべてのサービスを一覧表示します。
`#systemctl list-units --type=service | grep failed`	RHEL サーバーの再起動または再起動後にロードに失敗したすべてのサービスを一覧表示します。
`restorecon -Rv /etc/rc.d/init.d/aws-replication-service`	コンテキストを `aws-replication-service` に変更します。
`yum install policycoreutils*`	SELinux システムの運用に必要なポリシーコアユーティリティをインストールします。
`ausearch -c "insmod" --raw | audit2allow -M my-modprobe`	監査ログを検索し、ポリシーのモジュールを作成します。
`semodule -i my-modprobe.pp`	ポリシーを有効にします。

`cat my-modprobe.te`	`my-modprobe.te` ファイルの内容を表示します。
`semodule -l | grep my-modprobe`	ポリシーが SELinux モジュールに読み込まれているかどうかを確認します。

エピック

AWS レプリケーションエージェントをインストールし、RHEL ソースサーバーを再起動します。

タスク	説明	必要なスキル
アクセスキーとシークレットアクセスキーを使用してApplication Migration Service ユーザーを作成します。	AWS Replication エージェントをインストールするには、必要な AWS 認証情報を使用してアプリケーション移行サービスユーザーを作成する必要があります。手順については、AWS Service Catalog のドキュメントを参照してください。	移行エンジニア
AWS レプリケーションエージェントをインストールします。	1. AWS マネジメントコンソールにサインインし、AWS Service Catalog コンソール (https://console.aws.amazon.com/kms) を開きます。 2. アプリケーション移行サービスのドキュメント の指示に従ってレプリケーション設定を行います。 3. アプリケーション移行サービスのドキュメント の指示に従ってレプリケーション設定を行います。	移行エンジニア

タスク	説明	必要なスキル
	4. Source Servers ページで RHEL ソースサーバーを選択し、次に Replication を選択して初期レプリケーションを開始します。詳細については、Android アプリケーション ID のドキュメントを参照してください。	
RHEL ソースサーバーを再起動または再起動します。	データレプリケーションのステータスが移行ダッシュボードで停止と表示されたら、RHEL ソースサーバーを再起動または再起動します。	移行エンジニア
データレプリケーションステータス	1 時間待ってから、Migration ダッシュボードでデータ複製のステータスをもう一度確認します。これは正常状態である必要があります。	移行エンジニア

RHEL ソースサーバーの AWS レプリケーションエージェントのステータスを確認する

タスク	説明	必要なスキル
システムバージョンを識別します。	RHEL ソースサーバーのコマンドラインインターフェースを開き、以下のコマンドを実行してシステムバージョンを確認します。 `#systemctl -version`	移行エンジニア
すべてのアクティブなサービスを一覧表示します。	RHEL サーバー上で利用可能なすべてのアクティブなサー	移行エンジニア

タスク	説明	必要なスキル
	ビスを一覧表示するには、以下のコマンドを実行します。 `#systemctl list-units --type=service`	
実行中のすべてのサービスを一覧表示します。	RHEL サーバー上で現在実行中のすべてのサービスを一覧表示します。 `#systemctl list-units --type=service | grep running`	移行エンジニア
ロードに失敗したサービスをすべて一覧表示します。	RHEL サーバーの再起動または再起動後にロードに失敗したすべてのサービスを一覧表示します。 `#systemctl list-units --type=service | grep failed`	移行エンジニア

SELinux モジュールを作成して実行します。

タスク	説明	必要なスキル
セキュリティコンテキストを変更します。	RHEL ソースサーバーのコマンドラインインターフェイスで、次のコマンドを実行してセキュリティコンテキストを AWS レプリケーションサービスに変更します。	移行エンジニア

タスク	説明	必要なスキル
	`restorecon -Rv /etc/rc.d/init.d/aws-replication-service`	
コアユーティリティをインストールします。	SELinux システムとそのポリシーの運用に必要なコアユーティリティをインストールするには、以下のコマンドを実行します。 `yum install policycoreutils*`	移行エンジニア
監査ログを検索し、ポリシーのモジュールを作成します。	コマンドを実行します。 `ausearch -c "insmod" --raw | audit2allow -M my-modprobe`	移行エンジニア
my-modprobe-te ファイルの内容を表示します。	この `my-modprobe.te` ファイルは audit2allow コマンドによって生成されます。このファイルには SELinux ドメイン、ポリシーソースディレクトリ、サブディレクトリが含まれ、ドメインに関連するアクセスベクトルルールとトランジションが指定されます。次のコマンドを実行して、ファイルの内容を表示します。 `cat my modprobe.te`	移行エンジニア

タスク	説明	必要なスキル
ポリシーを有効にします。	モジュールを挿入してポリシーパッケージをアクティブにするには、以下のコマンドを実行します。 `semodule -i my-modprobe.pp`	移行エンジニア
モジュールがロードされているか確認してください。	コマンドを実行します。 `semodule -l | grep my-modprobe` SELinux モジュールがロードされると、移行中に SELinux を無効 モードまたは許可モードに設定する必要がなくなります。	移行エンジニア
RHEL ソースサーバーを再起動または再起動し、データ複製のステータスを確認します。	AWS Migration Service コンソールを開き、データ複製の進行状況に移動して、RHEL ソースサーバーを再起動または再起動します。これで、RHEL ソースサーバーの再起動後にデータ複製が自動的に再開されるはずです。	移行エンジニア

関連リソース

- アプリケーション移行サービスドキュメント
- テクニカルトレーニング資料
- AWS レプリケーションエージェントの問題のトラブルシューティング
- Application Migration Service ポリシー

リアーキテクト

トピック

- Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換
- Aurora PostgreSQL-Compatible でのアプリケーションユーザーとロールの作成
- PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行
- Aurora PostgreSQL 互換のファイルエンコーディングを使用して BLOB ファイルを TEXT にロード
- AWS SCT と AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行する AWS CLIAWS CloudFormation
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- Oracle SERIALLY_REUSABLE プラグマパッケージを PostgreSQL に移行
- Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行
- 関数ベースのインデックスを Oracle から PostgreSQL に移行する
- エクステンションを使用して Oracle のネイティブ関数を PostgreSQL に移行
- AWS DMS を使用して Db2 データベースを Amazon EC2 から Aurora MySQL 互換のデータベースに移行する
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon EC2 から Amazon DocumentDB に移行します
- オンプレミスの ThoughtSpot Falcon データベースを Amazon Redshift に移行する
- AWS DMS を使用して Oracle データベースを Amazon DynamoDB に移行する
- AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行
- Amazon RDS for Oracle から Amazon RDS for MySQL に移行する
- AWS DMS と AWS SCT を使用して IBM Db2 on Amazon EC2 から Aurora PostgreSQL 互換に移行する
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- マテリアライズドビューと AWS DMS を使用して Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行

- AWS DMS と AWS SCT を使用して Oracle on Amazon EC2 から Amazon RDS for MySQL に移行する
- AWS DMS を使用して Oracle から Amazon DocumentDB に移行する
- AWS DMS と AWS SCT を使用して、Oracle データベースを Amazon EC2 から Amazon RDS for MariaDB に移行する
- AWS DMS と AWS SCT を使用してオンプレミスの Oracle データベースを Amazon RDS for MySQL に移行する
- Oracle バイスタンダーと AWS DMS を使用して、オンプレミスの Oracle データベースを Amazon RDS for PostgreSQL に移行する
- Oracle GoldenGate を使用して Oracle Database から Amazon RDS for PostgreSQL に移行
- AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行
- オンプレミスの Oracle データベースから Aurora PostgreSQL にデータを移行する
- AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する
- AWS DMS を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- AWS SCT データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- AWS SCT データ抽出エージェントを使用して Teradata データベースを Amazon Redshift に移行
- AWS SCT データ抽出エージェントを使用してオンプレミスの Vertica データベースを Amazon Redshift に移行する
- レガシーアプリケーションを Oracle Pro*C から ECPG に移行する
- 仮想生成列をOracleから PostgreSQL に移行
- Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする
- Oracle から Amazon Aurora PostgreSQL への移行後にデータベースオブジェクトを検証する

Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換

作成者：Naresh Damera (AWS)

概要

Oracle 用 Amazon Relational Database Service (Amazon RDS) から Amazon Aurora PostgreSQL 互換エディションへの移行中に、 AWS Database Migration Service () で移行を検証するときにデータの不一致が発生する可能性がありますAWS DMS。この不一致を防止するために、VARCHAR2 (1) データ型をブール型データ型に変換できます。

VARCHAR2 データ型には可変長のテキスト文字列が格納され、VARCHAR2 (1) は文字列の長さが 1 文字または 1 バイトであることを示します。VARCHAR2 の詳細については、Oracle のビルドインデータ型 (Oracle ドキュメント) を参照してください。

このパターンでは、サンプルソースデータテーブル列の VARCHAR2 (1) データは、Y (はい)、または N (いいえ) です。 このパターンには、 AWS DMS と AWS Schema Conversion Tool （ AWS SCT) を使用して、このデータ型を VARCHAR2(1) の Y 値と N 値からブール値の true 値または false 値に変換する手順が含まれています。

対象者

このパターンは、 を使用して Oracle データベースを Aurora PostgreSQL 互換に移行した経験があるユーザーに推奨されます AWS DMS。移行が完了したら、「Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL」（ AWS SCT ドキュメント) の推奨事項に従ってください。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 認証情報、権限、セキュリティグループの設定など、環境が Aurora に対応していることを確認します。詳細について、Amazon Aurora 用の環境のセットアップ (Aurora ドキュメント)」を参照してください。
- VARCHAR2 (1) データを含むテーブル列を含むソース Amazon RDS for Oracle データベース。
- Amazon Aurora PostgreSQL 互換のターゲットデータベースインスタンス。詳細については、データベースクラスターを作成して Aurora PostgreSQL データベースクラスターのデータベースに接続 (Aurora ドキュメント) を参照してください。

製品バージョン

- Oracle バージョン 12.1.0.2 以降用の Amazon RDS for Oracle
- AWS DMS バージョン 3.1.4 以降。詳細については、「Using an Oracle database as a source for AWS DMS and Using a PostgreSQL database as a target for AWS DMS (AWS DMS documentation)」を参照してください。最も包括的なバージョンと機能のサポート AWS DMS には、 の最新バージョンを使用することをお勧めします。
- AWS Schema Conversion Tool （ AWS SCT) バージョン 1.0.632 以降。最も包括的なバージョンと機能のサポート AWS SCT には、 の最新バージョンを使用することをお勧めします。
- Aurora には、Aurora PostgreSQL 互換のデータベースエンジンバージョン (Aurora ドキュメント) に一覧表示された PostgreSQL バージョンが適用されます。

アーキテクチャ

ソーステクノロジースタック

Amazon RDS for Oracle データベースインスタンス

ターゲットテクノロジースタック

Amazon Aurora PostgreSQL 互換エディションに基づく DB インスタンス上のデータベース

ソースアーキテクチャとターゲットアーキテクチャ

ツール

AWS のサービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- AWS Database Migration Service （ AWS DMS ） は、データストアを に移行する AWS クラウドか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役立ちます。
- Oracle 用 Amazon Relational Database Service (Amazon RDS) は、 で Oracle リレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。

- AWS Schema Conversion Tool （ AWS SCT ） は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行をサポートします。

その他のサービス

- 「Oracle SQL Developer」 は、従来のデプロイとクラウドベースのデプロイの両方で Oracle データベースの開発と管理を簡素化する統合開発環境です。このパターンでは、このツールを使用して Amazon RDS for Oracle データベースインスタンスに接続し、データをクエリします。
- pgAdmin はPostgreSQL用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。このパターンでは、このツールを使用して Aurora データベースインスタンスに接続し、データをクエリします。

エピック

移行の準備をする

タスク	説明	必要なスキル
データベース移行レポートを作成します。	1. で AWS SCT、データベース移行評価レポートを作成します。詳細については、移行評価レポートの作成 を参照してください。 2. 移行評価レポートのアクション項目を確認して実行します。評価レポートの詳細については、評価レポートのアクション項目を参照してください。	DBA、開発者
外部キーの制約とトリガーをターゲットデータベースにドロップします。	PostgreSQLでは、トリガーを使用して、外部キーが実装されます。全ロードフェーズ中、 は各テーブルを一度に 1 つずつ AWS DMS ロードします。次のいずれかの方法を	DBA、開発者

タスク	説明	必要なスキル
	使用して、全ロード中に外部キーの制約を無効にすることを強くお勧めします。 • インスタンスからすべてのトリガーを一時的に無効にして、全ロードを終了します。 • PostgreSQL では、`session_replication_role` パラメータを使用します。 外部キー制約を無効にすることが不可能な場合は、親テーブルと子テーブルに固有のプライマリデータの AWS DMS 移行タスクを作成します。	

ターゲットデータベースのプライマリキーとユニークキーを無効にします。	次のコマンドを使用して、ターゲットデータベースの主キーと制約を無効にします。これにより、初期ロードタスクのパフォーマンスを改善します。
	`ALTER TABLE <table> DISABLE PRIMARY KEY;`
	`ALTER TABLE <table> DISABLE CONSTRAINT <constraint_name>;`
		DBA、開発者

タスク	説明	必要なスキル
初期ロードタスクを作成します。	で AWS DMS、初期ロードの移行タスクを作成します。手順については、タスクの作成を参照してください。移行方法は、既存のデータを移行するを選択します。この移行メソッドは APIで `Full Load` と呼び出されます。このタスクはまだ開始しないでください。	DBA、開発者

タスク	説明	必要なスキル
初期ロードタスクのタスク設定を編集します。	タスク設定を編集してデータ検証を追加します。これらの検証設定は JSON ファイルで作成する必要があります。手順と例については、タスク設定を指定 を参照してください。以下の検証を追加します。 • ターゲットデータベースで VARCHAR2 (1) データが正確にブーリンに変換されていることを検証するには、このパターンの追加情報 セクションのデータ検証スクリプトにコードを追加します。検証スクリプトは、ターゲットテーブルのブーリン値 1 を Y に、0 を N に変換し、ターゲットテーブルの値をソーステーブルと比較します。 残りのデータ移行を検証するには、タスクでデータ検証を有効にします。詳細については、データ検証のタスク設定をご参照ください。	AWS 管理者、データベース管理者

タスク	説明	必要なスキル
継続的レプリケーション タスクを作成します。	で AWS DMS、ターゲットデータベースをソースデータベースと同期させる移行タスクを作成します。手順については、タスクの作成を参照してください。移行方法には、データ変更のみを複製を選択します。このタスクはまだ開始しないでください。	DBA

移行タスクをテストします。

タスク	説明	必要なスキル
テストのサンプルデータを作成します。	ソースデータベースで、テストのデータを含むサンプルテーブルを作成します。	開発者
競合するアクティビティがないことを確認します。	`pg_stat_activity` を使用して、移行に影響する可能性のあるサーバのアクティビティがないか確認します。詳細については、統計コレクター（PostgreSQL ドキュメント）を参照してください。	AWS 管理者
AWS DMS 移行タスクを開始します。	AWS DMS コンソールの Dashboard ページで、前のエピックで作成した初期ロードと継続的なレプリケーションタスクを開始します。	AWS 管理者
タスクとテーブルのロード状態を監視します。	移行中は、タスクの状態とテーブルの状態 を監視します。初期化ロードタスクが完	AWS 管理者

タスク	説明	必要なスキル
	了した場合、テーブル統計 タブで次の操作を行います。 • ロード状態 は、テーブル完了であるはずです。 • 検証状態は、検証済みでなければなりません。	
移行結果を検証します。	pgAdmin を使用して、ターゲットデータベースのテーブルをクエリします。クエリが成功する場合、データが正常に移行されたことを示します。	開発者
ターゲットデータベースにプライマリキーと外部キーを追加します。	ターゲットデータベースにプライマリキーと外部キーを作成します。詳細については、テーブルを変更 (PostgreSQL ウェブサイト) を参照してください。	DBA
テストデータをクリーンアップします。	ソースデータベースとターゲットデータベースで、ユニットテストに作成されたデータをクリーンアップします。	開発者

カットオーバー

タスク	説明	必要なスキル
移行が完了しました。	前のエピックスを繰り返して、実際のソースデータを使用して、移行タスクをテストします。これにより、ソース	開発者

タスク	説明	必要なスキル
	からターゲットデータベースにデータが移行されます。	
ソースデータベースとターゲットデータベースが同期していることを確認します。	ソースデータベースとターゲットデータベースが同期していることを確認します。詳細と手順については、AWS DMS 「データ検証」を参照してください。	開発者
ソースデータベース	Amazon RDS for Oracle データベースを停止します。詳細については、一時的に Amazon RDS DB インスタンスを停止するを参照してください。ソースデータベースを停止すると、 の初期ロードと継続的なレプリケーションタスク AWS DMS が自動的に停止します。これらのタスクを停止するために追加のアクションは必要ありません。	開発者

関連リソース

AWS リファレンス

- AWS DMS と を使用して Oracle データベースを Aurora PostgreSQL に移行する AWS SCT (AWS 規範ガイダンス)
- Oracle を Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL に変換する (AWS SCT ドキュメント)
- AWS DMS の仕組み (AWS DMS ドキュメント)

その他のリファレンス

- ブーリアンデータ型 (PostgreSQL ドキュメント)
- Oracle ビルトインデータ型 (Oracle ドキュメント)
- pgAdmin (pgAdmin ウェブサイト)
- SQL 開発者 (Oracleのウェブサイト)

チュートリアルと動画

- の開始方法 AWS DMS
- Amazon RDS の開始方法
- の概要 AWS DMS (動画)
- Amazon RDS を理解する

追加情報

データ検証スクリプト

次のデータ検証スクリプトは、1 を Y、0 を N に変換します。 これにより、 AWS DMS タスクが正常に完了し、テーブルの検証に合格します。

```
{
"rule-type": "validation",
"rule-id": "5",
"rule-name": "5",
"rule-target": "column",
"object-locator": {
"schema-name": "ADMIN",
"table-name": "TEMP_CHRA_BOOL",
"column-name": "GRADE"
},
"rule-action": "override-validation-function",
"target-function": "case grade when '1' then 'Y' else 'N' end"
        }
```

case スクリプトのステートメントが検証を実行します。検証が失敗した場合、 はターゲットデータベースインスタンスの public.awsdms_validation_failures_v1 テーブルにレコード AWS DMS を挿入します。このレコードには、テーブル名、エラー時間、およびソーステーブルとターゲットテーブルの不一致値に関する詳細が含まれます。

このデータ検証スクリプトを AWS DMS タスクに追加せず、データがターゲットテーブルに挿入されると、 AWS DMS タスクは検証状態を不一致レコードとして表示します。

AWS SCT 変換中、 AWS DMS 移行タスクは VARCHAR2(1) データ型のデータ型をブール値に変更し、"N0"列にプライマリキー制約を追加します。

Aurora PostgreSQL-Compatible でのアプリケーションユーザーとロールの作成

Abhishek Verma (AWS) によって作成されました

概要

Amazon Aurora PostgreSQL-Compatible Edition に移行する場合、ソースデータベースに存在するデータベースユーザーとロールは Aurora PostgreSQL -Compatible データベースで作成する必要があります。Aurora PostgreSQL-Compatible でユーザーとロールを作成するには、次の 2 つの方法があります。

- ターゲットではソースデータベースと同様のユーザーとロールを使用します。この方法では、ユーザーとロールのデータ定義言語 (DDL) がソースデータベースから抽出されます。その後、それらは変換され、ターゲットの Aurora PostgreSQL-Compatible データベースに適用されます。たとえば、ブログ記事「SQL で Oracle から PostgreSQL にユーザ、ロールと権限をマッピング」では、Oracle ソースデータベースエンジンからの抽出の使用について説明しています。
- 開発、管理およびデータベースでのその他の関連操作の実行によく使用される、標準化されたユーザーとロールを使用してください。これには、各ユーザーが実行する読み取り専用、読み取り/書き込み、開発、管理とデプロイ操作が含まれています。

このパターンには、標準化されたユーザーとロールのアプローチに必要な Aurora PostgreSQL -Compatible でのユーザーとロールの作成に必要なグラントが含まれています。ユーザーとロールの作成手順は、データベースユーザーに与える権限を最小限に抑えるというセキュリティポリシーに従っています。次の表は、ユーザー、対応するロールとデータベース上の詳細を示しています。

[ユーザー]	ロール	目的
`APP_read`	`APP_RO`	スキーマ APP の読み取り専用アクセスに使用されます
`APP_WRITE`	`APP_RW`	スキーマ APP の書き込み操作と読み取り操作に使用されます
`APP_dev_user`	`APP_DEV`	スキーマ APP_DEV の開発目的で使用され、スキーマ APP への読み取り専用アクセス権が付与されます

Admin_User	rds_superuser	データベースの管理者操作を実行するために使用されます
APP	APP_DEP	APP スキーマの下にオブジェクトを作成し、または APP スキーマにオブジェクトをデプロイするために使用されます

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント
- PostgreSQL データベース、Amazon Aurora PostgreSQL-Compatible Edition データベースまたは PostgreSQL データベース用の Amazon Relational Database Service (Amazon RDS)

製品バージョン

- PostgreSQL のすべてのバージョン

アーキテクチャ

ソーステクノロジースタック

- 任意のデータベース

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL 互換

ターゲットアーキテクチャ

次の図は、Aurora PostgreSQL-Compatible データベースのユーザーロールとスキーマアーキテクチャを示しています。

自動化とスケール

このパターンには、ユーザー、ロールとスキーマ作成スクリプトが含まれており、ソースまたはターゲットデータベースの既存ユーザーに影響を与えることなく複数回実行できます。

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援する、フルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。

その他のサービス

- 「psql」は、PostgreSQL データベースをインストールするたびにインストールされるターミナルベースのフロントエンドツールです。SQL、PL-PGSQL とオペレーティングシステムのコマンドを実行するためのコマンドラインインターフェースを備えています。
- 「pgAdmin」 はPostgreSQL用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

エピック

ユーザーとロールの作成

タスク	説明	必要なスキル
デプロイユーザーの作成	デプロイメントユーザー APP は、デプロイ中にデータベースオブジェクトの作成と変更に使用されます。以下のスクリプトを使用して、スキーマ APP にデプロイユーザーロール APP_DEP を作成します。アクセス権を検証して、このユーザーには必要なスキーマ APP 内のオブジェクトを作成	DBA

タスク	説明	必要なスキル

	する権限のみが付与されていることを確認します。
	
	1. 管理者ユーザーに接続して、スキーマを作成します。
	
	`CREATE SCHEMA APP;`
	
	2. ユーザーを作成します。
	
	`CREATE USER APP WITH PASSWORD <password> ;`
	
	3. ロールを作成します。
	
	`CREATE ROLE APP_DEP ;`
	`GRANT all on schema APP to APP_DEP ;`
	`GRANT USAGE ON SCHEMA APP to APP_DEP ;`
	`GRANT connect on database <db_name> to APP_DEP ;`
	`GRANT APP_DEP to APP;`
	
	4. 権限をテストするには、APP に接続してテーブルを作成します。
	
	`set search_path to APP;`
	`SET`
	`CREATE TABLE test(id integer ) ;`
	`CREATE TABLE`
	
	5. 権限を確認してください。	

タスク	説明	必要なスキル
	`select schemaname , tablename , tableowner from pg_tables where tablename like 'test' ;` `schemaname	tablename	tableowner` `APP	test	APP`	

タスク	説明	必要なスキル

読み取り専用ユーザーを作成します。	読み取り専用ユーザー APP_read は、スキーマ APP の読み取り専用操作を実行するために使用されます。以下のスクリプトを使用して読み取り専用ユーザーを作成します。アクセス権を検証して、このユーザーがスキーマ APP 内のオブジェクトを読み取る権限のみを持っていることおよびスキーマ APP に作成された新しいオブジェクトに対する読み取りアクセス権を自動的に付与する権限を持っていることを確認します。
	
	1. APP_read ユーザーを作成します。
	
	`create user APP_read ;`
	`alter user APP_read`
	`with password`
	`'your_password' ;`
	
	2. ロールを作成します。
	
	`CREATE ROLE APP_ro ;`
	`GRANT SELECT ON ALL`
	`TABLES IN SCHEMA APP`
	`TO APP_RO ;`
	`GRANT USAGE ON SCHEMA`
	`APP TO APP_RO`
	`GRANT CONNECT ON`
	`DATABASE testdb TO`
	`APP_RO ;`
	`GRANT APP_RO TO`
	`APP_read;`
		DBA

タスク	説明	必要なスキル

	3. 権限をテストするには、APP_read ユーザーを使用してログインします。
	
	`set search_path to APP ;`
	`create table test1( id integer) ;`
	`ERROR: permission denied for schema APP`
	`LINE 1: create table test1( id integer) ;`
	`insert into test values (34) ;`
	`ERROR: permission denied for table test`
	`SQL state: 42501`
	`select from test`
	`no rows selected`	

タスク	説明	必要なスキル

読み取り/書き込みユーザーを作成します。	読み取り/書き込みユーザー APP_WRITE は、スキーマ APP の読み取りと書き込み操作の実行に使用されます。以下のスクリプトを使用して読み取り/書き込みユーザーを作成し、そのユーザーに APP_RW ロールを付与します。アクセス権を検証して、このユーザーがスキーマ APP 内のオブジェクトに対してのみ読み取り権限と書き込み権限を持っていることを確認し、スキーマ APP 内に作成された新しいオブジェクトに対する読み取りと書き込みアクセス権を自動的に付与します。
	
	1. ユーザーを作成します。
	
	`CREATE USER APP_WRITE ; alter user APP_WRITE with password 'your_password' ;`
	
	2. ロールを作成します。
	
	`CREATE ROLE APP_RW; GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA APP TO APP_RW ; GRANT CONNECT ON DATABASE postgres to APP_RW ; GRANT USAGE ON SCHEMA APP to APP_RW ;`	

タスク	説明	必要なスキル
	``` ALTER DEFAULT PRIVILEGES IN SCHEMA APP GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO APP_RW ; GRANT APP_RW to APP_WRITE ``` 3. 権限をテストするには、APP_WRITE ユーザーを使用してログインします。 ``` SET SEARCH_PATH to APP; CREATE TABLE test1(id integer) ; ERROR: permission denied for schema APP LINE 1: create table test1(id integer) ; SELECT * FROM test ; id ---- 12 INSERT INTO test values (31) ; INSERT 0 1 ```	

タスク	説明	必要なスキル

管理者ユーザーを作成します。	管理者ユーザー Admin_User は、データベースの管理操作を実行するために使用されます。これらの操作の例は CREATE ROLE と CREATE DATABASE Admin_User です。rds_superuser はビルトインロールを使用してデータベースの管理操作を実行します。以下のスクリプトを使用して、データベース内の管理者ユーザー Admin_User の権限を作成してテストします。
	
	1. ユーザーを作成し、ロールを付与します。
	
	`create user Admin_User WITH PASSWORD ‘Your password’ ALTER user Admin_user CREATEDB; ALTER user Admin_user CREATEROLE;`
	
	2. 権限をテストするには、Admin_User ユーザーからログインします。
	
	`SELECT * FROM APP.test ; id ---- 31 CREATE ROLE TEST ;`
		DBA

タスク	説明	必要なスキル
	``` CREATE DATABASE test123 ; ```	

タスク	説明	必要なスキル

開発ユーザーを作成します。	開発ユーザー APP_dev_user には、ローカルスキーマ APP_DEV にオブジェクトを作成する権限とスキーマ APP の読み取りアクセス権が付与されます。以下のスクリプトを使用して、データベース内のユーザー APP_dev_user の権限を作成してテストします。
	
	1. ユーザーを作成します。
	
	`CREATE USER APP1_dev_user with password ‘your password’;`
	
	2. App_dev_user のためにAPP_DEV スキーマを作成します。
	
	`CREATE SCHEMA APP1_DEV ;`
	
	3. APP_DEV ロールを作成し、
	
	`CREATE ROLE APP1_DEV ;`
	`GRANT APP1_RO to APP1_DEV ;`
	`GRANT SELECT ON ALL TABLES IN SCHEMA APP1_DEV to APP1_dev_user`
	`GRANT USAGE, CREATE ON SCHEMA APP1_DEV to APP1_DEV_USER`
	`GRANT APP1_DEV to APP1_DEV_USER ;`
		DBA

タスク	説明	必要なスキル

	4. 権限をテストするには、APP_dev_user からログインします。
	
	`CREATE TABLE APP1_dev.test1( id integer ) ;`
	`CREATE TABLE`
	`INSERT into APP1_dev.test1 ( select * from APP1.test );`
	`INSERT 0 1`
	`CREATE TABLE APP1.test4 ( id int) ;`
	`ERROR: permissio n denied for schema APP1`
	`LINE 1: create table APP1.test4 ( id int) ;`	

関連リソース

PostgreSQL ドキュメント

- 「ロールの作成」
- 「ユーザーの作成」
- 「定義済みのロール」

追加情報

PostgreSQL 14 エンハンスメント

PostgreSQL 14 には、事前定義されたロールのセットが用意されており、一般的に必要とされる特定の特権的な機能や情報へのアクセスを可能にしています。管理者 (CREATE ROLE 権限を持つロールを含む) は、これらのロールや環境内の他のロールをユーザーに付与して、指定された機能や情報にアクセスできるようにすることができます。

管理者は GRANT コマンドを使用してこれらのロールへのアクセス権をユーザーに付与できます。たとえば、pg_signal_backend ロールに Admin_User を付与するには、以下のコマンドを実行します。

```
GRANT pg_signal_backend TO Admin_User;
```

この pg_signal_backend ロールは、管理者が信頼できる非スーパーユーザーロールが他のバックエンドにシグナルを送信できるようにすることを目的としています。詳細については、「PostgreSQL 14 の機能拡張」を参照してください。

アクセスの微調整

場合によっては、より詳細なアクセス (テーブルベースのアクセスや列ベースのアクセスなど) をユーザーに提供することが必要になる場合があります。このような場合は、追加ロールを作成して、それらの権限をユーザーに付与することができます。詳細については、「PostgreSQL Grants」を参照してください。

PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート

ハリクリシュナ・ボーガッダ (AWS) によって作成されました

概要

エンタープライズディザスタリカバリ (DR) のベストプラクティスは、基本的に、最小限の介入で、理想的にはデータを失うことなく、災害に耐え (事業継続)、通常の運用を再開できる (事業再開) 耐障害性のあるハードウェアおよびソフトウェアシステムの設計と実装です。エンタープライズ DR の目標を達成するために耐障害性のある環境を構築するには、費用と時間がかかる場合があり、企業による強いコミットメントが必要です。

Oracle Database には、Oracle データを保護する他のどのアプローチよりも最高レベルのデータ保護と可用性を提供する 3 つの異なる DR アプローチがあります。

- Oracle データ損失ゼロ・リカバリー・アプライアンス
- Oracle Active Data Guard
- Oracle GoldenGate

このパターンは、Amazon Aurora Global Database を使用して Oracle GoldenGate DR をエミュレートする方法を提供します。リファレンスアーキテクチャでは、3 つの AWS リージョンにわたる DR に Oracle GoldenGate を使用しています。このパターンは、ソースアーキテクチャを Amazon Aurora PostgreSQL 互換エディションに基づくクラウドネイティブ Aurora グローバルデータベースにリプラットフォーム化する手順を示しています。

Aurora Global Database は、グローバルフットプリントを持つアプリケーション向けに設計されています。1 つの Aurora データベースは、最大 5 つのセカンダリリージョンの複数の AWS リージョンにまたがっています。Aurora グローバルデータベースには次の特徴量があります。

- 物理ストレージレベルのレプリケーション
- 低レイテンシーのグローバル読み取り
- 地域全体の障害からの迅速なディザスタリカバリ
- クロスリージョン集の高速移行
- リージョン間のレプリケーション遅延が少ない
- データベースへのパフォーマンスへの影響はほとんど、あるいはまったくない

Aurora Global Database 特徴量と利点については、「Amazon Aurora Global Database の使用」 を参照してください。計画外フェイルオーバーとマネージドフェールオーバーの詳細については、「Amazon Aurora Global Database でのフェイルオーバーの使用」 を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- アプリケーション接続用の Java Database Connectivity (JDBC) PostgreSQL ドライバー。
- Amazon Aurora PostgreSQL 互換エディションに基づく Aurora グローバルデータベース
- Oracle Real Application クラスター (RAC) データベースが Aurora PostgreSQL 互換の Aurora グローバルデータベースに移行されました

Aurora Global Database の制限

- Aurora グローバルデータベースが AWS リージョンに使用不可です。サポートされているリージョンのリストについては、「Aurora PostgreSQL による Aurora グローバルデータベース」 を参照してください。
- サポートされていない特徴量や Aurora グローバルデータベースのその他の制限については、「Amazon Aurora Global Database の制限事項」 を参照してください。

製品バージョン

- Amazon Aurora PostgreSQL 互換エディションバージョン 10.14 以降

アーキテクチャ

ソーステクノロジースタック

- Oracle RAC 4 ノードデータベース
- Oracle GoldenGate

ソースアーキテクチャ

次の図は、Oracle GoldenGate を使用してレプリケートされた、異なる AWS リージョンにある 4 ノード Oracle RAC を持つ 3 つのクラスターを示しています。

ターゲットテクノロジースタック

- Aurora PostgreSQL と互換性のある 3 クラスターの Amazon Aurora Global Database。プライマリリージョンに 1 つのクラスター、異なるセカンダリリージョンに 2 つのクラスターがあります

ターゲットアーキテクチャ

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」 は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。
- 「Amazon Aurora global databases」は、複数の AWS リージョンにまたがり、低レイテンシーのグローバル読み取りを提供し、AWS リージョン全体に影響が及ぶ可能性のある機能停止から、すばやい復旧を行います。

エピック

リーダー DB インスタンスのリージョンを追加

タスク	説明	必要なスキル
1 つ以上のセカンダリの Aurora クラスターを接続します。	[AWS マネジメントコンソール] で、[Amazon Aurora] を選択します。プライマリクラスターを選択し、アクションを選択し、ドロップダウンリストからリージョンを追加を選択します。	DBA
インスタンスクラスを選択します。	セカンダリクラスターのインスタンスクラスは変更できま	DBA

タスク	説明	必要なスキル
	す。ただし、プライマリクラスターインスタンスクラスと同じにしておくことをお勧めします。	
3 番目のリージョンを追加します。	このエピックの手順を繰り返して、3 番目のリージョンにクラスターを追加します。	DBA

Aurora グローバルデータベースのフェイルオーバー

タスク	説明	必要なスキル
Aurora グローバルデータベース からプライマリクラスターを削除します。	1. データベースページで プライマリクラスターを選択します。 2. セカンダリクラスターにフェイルオーバーするには、グローバルから削除を選択します。	DBA
新しく昇格させたクラスターに書き込みトラフィックを振り向けるようにアプリケーションを設定し直します。	アプリケーションのエンドポイントを、新しくプロモートされたクラスターのエンドポイントに変更します。	DBA
使用できないクラスターへの書き込み操作をすべて停止します。	削除したクラスターに対するアプリケーションとデータ操作言語 (DML) アクティビティを停止します。	DBA
新しいAurora グローバルデータベースを作成します。	今、新しく昇格させたクラスターをプライマリクラスターとして Aurora グローバルデータベースを作成します。	DBA

プライマリクラスターを起動します。

タスク	説明	必要なスキル
グローバルデータベースから起動するプライマリクラスターを選択します。	Amazon Aurora コンソールのグローバルデータベースセットアップで、プライマリクラスターを選択します。	DBA
クラスターを開始します。	アクションドロップダウンリストで開始を選択します。このプロセスには時間がかかる場合があります。画面を更新してステータスを確認するか、操作完了後にステータス列でクラスターの現在の状態を確認します。	DBA

リソースをクリーンアップします。

タスク	説明	必要なスキル
残りの二次クラスターを削除します。	フェイルオーバーパイロットが完了したら、グローバルデータベースからセカンダリクラスターを解除します。	DBA
プライマリクラスターを削除します。	クラスタを削除します。	DBA

関連リソース

- 「Amazon Aurora Global Database の使用」
- 「Amazon Aurora Global Database を使用した Aurora PostgreSQL ディザスタリカバリソリューション」(ブログ記事)

SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する

作成者: Rajkumar Raghuwanshi (AWS) と Jagadish Kantubugata (AWS)

概要

このパターンでは、SQL Server から Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL PostgreSQL 互換に移行するときに、E メールアドレスに Secure Hash Algorithm 1 (SHA1) ハッシュを実装する方法について説明します。E メールアドレスは、個人を特定できる情報 (PII) の例です。PII は、直接表示したり、他の関連データとペアになったりするときに、個人のアイデンティティを合理的に推測するために使用できる情報です。

このパターンは、さまざまなデータベース照合順序と文字エンコーディングにわたって一貫したハッシュ値を維持するという課題をカバーし、PostgreSQL 関数とトリガーを使用するソリューションを提供します。このパターンは SHA1 ハッシュに焦点を当てていますが、PostgreSQL の `pgcrypto`モジュールでサポートされている他のハッシュアルゴリズムに合わせて調整できます。機密データを処理する場合は、ハッシュ戦略のセキュリティへの影響を常に考慮し、セキュリティの専門家に相談してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ソース SQL Server データベース
- ターゲット PostgreSQL データベース (Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換)
- PL/PgSQL コーディングの専門知識

制約事項

- このパターンでは、ユースケースに基づいてデータベースレベルの照合順序を変更する必要があります。
- 大規模なデータセットに対するパフォーマンスへの影響は評価されていません。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS 「リージョン別のサービス」を参照してください。特定のエンドポ

イントについては、「サービスエンドポイントとクォータ」を参照して、サービスのリンクを選択します。

製品バージョン

- Microsoft SQL Server 2012 以降

アーキテクチャ

ソーステクノロジースタック

- SQL Server
- 特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。

ターゲットテクノロジースタック

- PostgreSQL
- `pgcrypto` 拡張機能

自動化とスケール

- メンテナンスを容易にするために、ハッシュ関数をストアドプロシージャとして実装することを検討してください。
- 大規模なデータセットの場合は、パフォーマンスを評価し、バッチ処理またはインデックス作成戦略を検討します。

ツール

AWS のサービス

- Amazon Aurora PostgreSQL 互換は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つフルマネージドの ACID 準拠のリレーショナルデータベースエンジンです。
- AWS Database Migration Service （ AWS DMS ） を使用すると、データストアを に移行する AWS クラウド か、クラウドとオンプレミスのセットアップの組み合わせ間で移行できます。
- Amazon Relational Database Service Amazon RDS for PostgreSQL は、 で PostgreSQL リレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。

- AWS Schema Conversion Tool （ AWS SCT ） は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行をサポートします。

その他のツール

- 「pgAdmin」 は PostgreSQL 用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。
- SQL Server Management Studio (SSMS) は、SQL インフラストラクチャを管理するための統合環境です。

ベストプラクティス

- ターゲットデータベース側で特殊文字を処理するには、適切な照合設定を使用します。
- ASCII 以外の文字のアドレスなど、さまざまな E メールアドレスを使用して徹底的にテストします。
- アプリケーションレイヤーとデータベースレイヤー間の大文字と小文字の処理の一貫性を維持します。
- ハッシュ値を使用してクエリのパフォーマンスをベンチマークします。

エピック

ソースハッシュの実装を分析する

タスク	説明	必要なスキル
SQL Server コードを確認します。	SHA1 ハッシュを生成する SQL Server コードを確認するには、次の手順を実行します。 • SHA1 ハッシュの既存の SQL Server 実装を分析します。 • ハッシュ生成に使用される正確な方法を特定します。	データエンジニア、DBA、アプリ開発者

タスク	説明	必要なスキル
	• 入力パラメータと出力形式を文書化します。 • データ型の変換を確認します。 • 照合設定とその影響を調べます。	
ハッシュアルゴリズムとデータ変換を文書化します。	ハッシュアルゴリズムとデータ変換を正確に文書化するには、以下を実行します。 • ハッシュプロセスの詳細な技術ドキュメントを作成します。 • step-by-step変換ロジックを文書化します。 • 入力および出力形式とデータ型を指定します。 • エッジケースと特殊文字処理を含めます。	アプリケーション開発者、データエンジニア、DBA

PostgreSQL ハッシュ関数を作成する

タスク	説明	必要なスキル
pgcrypto 拡張機能を作成します。	pgcrypto 拡張機能を作成するには、 pgAdmin/psql を使用して次のコマンドを実行します。 `CREATE EXTENSION pgcrypto;`	DBA、データエンジニア

タスク	説明	必要なスキル
PostgreSQL 関数を実装します。	次の PostgreSQL 関数を実装して、SQL Server ハッシュロジックをレプリケートします。大まかに言うと、この関数は次のステップを使用します。 1. 必要に応じて、入力を大文字に変換します。 2. 入力の SHA1 ハッシュを作成します。 3. このハッシュの最後の 10 バイト (80 ビット) を取得します。 4. これらのバイトを 64 ビット整数に変換します。 `CREATE OR REPLACE FUNCTION utility.hex_to_bigint ( par_val character varying, par_upper character varying DEFAULT 'lower'::character varying) RETURNS bigint LANGUAGE 'plpgsql' AS $BODY$ DECLARE retnumber bigint; digest_bytes bytea; BEGIN if lower(par_upper) = 'upper' then`	データエンジニア、DBA、アプリ開発者

タスク	説明	必要なスキル

	<pre>digest_bytes := digest(upper(par_val), 'sha1');
	    else
	digest_bytes := digest((par_val), 'sha1');
	    end if;
	    retnumber := ('x' || encode(substring(digest_bytes, length(digest_bytes)-10+1), 'hex'))::bit(64)::bigint;
	    RETURN retnumber;
	END;
	$BODY$;</pre>	

タスク	説明	必要なスキル

関数をテストします。	関数をテストするには、SQL Server のサンプルデータを使用して、一致するハッシュ値を検証します。次のコマンドを実行してください。
	
	`select 'alejandr o_rosalez@example. com' as Email, utility.hex_to_big int('alejandro_ros alez@example.com', 'upper') as HashValue;`
	
	`--OUTPUT /* email hashvalue "alejandro_rosale z@example.com" 451 397011176045063 */`
		アプリ開発者、DBA、データエンジニア

自動ハッシュのトリガーを実装する

タスク	説明	必要なスキル
関連するテーブルにトリガーを作成します。	挿入または更新時にハッシュ値を自動的に生成するトリガーを関連するテーブルに作成するには、次のコマンドを実行します。 `CREATE OR REPLACE FUNCTION update_em ail_hash() RETURNS TRIGGER`	アプリケーション開発者、データエンジニア、DBA

タスク	説明	必要なスキル

	AS $$
	BEGIN
	    NEW.email_hash =
	 utility.hex_to_big
	int(NEW.email,
	 'upper');
	    RETURN NEW;
	END;
	$$ LANGUAGE plpgsql;
	
	CREATE TRIGGER email_has
	h_trigger BEFORE INSERT
	 OR UPDATE ON users
	 FOR EACH ROW EXECUTE
	 FUNCTION update_em
	ail_hash();	

既存のデータの移行

タスク	説明	必要なスキル
移行スクリプトを開発するか、 を使用します AWS DMS。	移行スクリプトを作成するか AWS DMS 、 を使用して既存のデータのハッシュ値 (BIGINTソースシステムに として保存されているハッシュ値を含む) を入力します。以下のタスクを実行します。 • ハッシュ値を使用してデータ転送用の移行スクリプトを作成します。 • 適切な変換ルールを使用して AWS DMS タスクを設定します。	データエンジニア、アプリ開発者、DBA

タスク	説明	必要なスキル
	• でソースエンドポイントとターゲットエンドポイントを設定します AWS DMS。 • エラー処理と口グ記録メカニズムを実装します。 • 大規模なデータセットのバッチ処理戦略を設計します。 • データ検証用の検証クエリを作成します。	
新しい PostgreSQL ハッシュ関数を使用します。	新しい PostgreSQL ハッシュ関数を使用して整合性を確保するには、以下を実行します。 • ハッシュの整合性を検証するための検証手順を実装します。 • ソースシステムとターゲットシステム間の比較スクリプトを作成します。 • ハッシュ値検証の自動テストを設定します。 • 不一致と解決手順を文書化します。	アプリ開発者、DBA、DevOps エンジニア

アプリケーションクエリを更新する

タスク	説明	必要なスキル
アプリケーションクエリを特定します。	ハッシュ化された値を使用するアプリケーションクエリを	アプリ開発者、DBA、データエンジニア

タスク	説明	必要なスキル
	特定するには、以下を実行します。 • ハッシュ値を使用して、アプリケーションコードベースでクエリを分析します。 • ハッシュオペレーションを参照するストアドプロシージャと関数を確認します。 • クエリパフォーマンスメトリクスと実行計画を文書化します。 • ハッシュベースのルックアップへの依存関係を特定します。 • 影響を受けるアプリケーションコンポーネントをマッピングします。	

タスク	説明	必要なスキル
クエリを変更します。	必要に応じて、新しい PostgreSQL ハッシュ関数を使用するようにクエリを変更します。以下の操作を実行します。 • PostgreSQL ハッシュ関数を使用するように既存のクエリをリファクタリングします。 • ストアドプロシージャと関数を更新します。 • 新しいクエリパターンを実装してテストします。 • 変更されたクエリのパフォーマンスを最適化します。	アプリ開発者、DBA、データエンジニア

テストと検証

タスク	説明	必要なスキル
テストを実行します。	本番データのサブセットを使用して徹底的なテストを実行するには、以下を実行します。 • データサブセット検証のテストプランを作成します。 • 本番データの代表的なサンプルを抽出します。 • 適切な設定でテスト環境を設定します。	アプリケーション開発者、データエンジニア、DBA

タスク	説明	必要なスキル
	• データのロードと変換テストを実行します。 • ボリュームテストとストレステストを実行します。	
ハッシュ値が一致することを確認します。	SQL Server と PostgreSQL の間でハッシュ値が一致することを確認するには、以下を実行します。 • ハッシュ値の比較スクリプトを開発します。 • ハッシュマッチングの検証レポートを作成します。 • 自動検証手順を実装します。 • 見つかった不一致があれば文書化します。 • ハッシュの不一致を分析して解決します。	アプリケーション開発者、データエンジニア、DBA

タスク	説明	必要なスキル
アプリケーションの機能を確認します。	移行されたデータと新しいハッシュ実装を使用してアプリケーションの機能を確認するには、以下を実行します。 • end-to-endアプリケーションテストを実行します。 • ハッシュ化されたデータを使用してすべてのアプリケーション機能を検証します。 • 新しい実装でアプリケーションのパフォーマンスをテストします。 • API の統合と依存関係を検証します。	アプリ開発者、DBA、データエンジニア

トラブルシューティング

問題	ソリューション
ハッシュ値が一致しません。	ソースとターゲット間の文字エンコードと照合を確認します。詳細については、「Amazon Aurora および Amazon RDS での PostgreSQL での照合変更の管理」(AWS ブログ) を参照してください。

関連リソース

AWS ブログ

- Amazon Aurora と Amazon RDS で PostgreSQL の照合変更を管理する
- ベストプラクティスと フィールドから学んだ教訓を使用して SQL Server を Amazon Aurora PostgreSQL に移行する

その他のリソース

- PostgreSQL pgcrypto モジュール (PostgreSQL ドキュメント)
- PostgreSQL トリガー関数 (PostgreSQL ドキュメント)
- SQL Server HASHBYTES 関数 (Microsoft ドキュメント)

Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行

作成者: Pinesh Singal (AWS)

概要

多くの移行策略やアプローチは、複数のフェーズに分かれて、数週間から数ヶ月間かかることもあります。この間、PostgreSQL DB インスタンスに移行するソース Oracle DB インスタンスのパッチ適用またはアップグレードが原因で、遅延が発生する可能性があります。このような状況を回避するには、残りの Oracle データベースコードを PostgreSQL データベースコードに段階的に移行することを推奨します。

このパターンでは、最初の移行後に多数のトランザクションが実行される、そして PostgreSQL データベースに移行する必要があるマルチテラバイトの Oracle DB インスタンスに対して、ダウンタイムのない、段階的な移行戦略を提供します。このパターンのステップバイステップのアプローチを使用すれば、 Amazon Web Services (AWS) マネジメントコンソールにサインインする必要なく、Oracle DB インスタンスのAmazon Relational Database Service (Amazon RDS) を Amazon RDS for PostgreSQL インスタンスに段階的に移行できます。

このパターンでは、「Oracle SQL Developer」 を使用して、ソース Oracle データベース内の二つのスキーマの差分を見つけます。次に、AWS Schema Conversion Tool (AWS SCT) を使用して、Amazon RDS for Oracle データベーススキーマオブジェクトを Amazon RDS for PostgreSQL データベーススキーマオブジェクトに変換します。次に、Windows コマンドプロンプトで Python スクリプトを実行して、ソースデータベースオブジェクトへの段階的な変更のために AWS SCT オブジェクトを作成します。

> **Note**
>
> 本番ワークロードを移行する前に、テスト環境または非本番環境でこのパターンのアプローチの概念実証 (PoC) を実行することをお勧めします。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 既存の Amazon RDS for Oracle DB インスタンス。

- 既存の Amazon RDS for PostgreSQL DB インスタンス。
- AWS SCT、Oracle および PostgreSQL データベースエンジンの JDBC ドライバーにインストールされ、設定されたAWS SAWS SCT。これの詳細については、AWS SCT ドキュメントの「AWS SCTをインストール」 、および「必要なデータベースドライバーをインストール」 を参照してください。
- インストール、設定されたOracle SQL Developer。これの詳細については、「Oracle SQL Developer」 ドキュメントを参照してください。
- ローカルのコンピュータにダウンロードされた、`incremental-migration-sct-sql.zip` ファイル(添付)。

機能制限

- ソースである Amazon RDS for Oracle DB インスタンスの最小要件は次のとおりです:
  - Enterprise、Standard、Standard One、Standard Two エディションの、10.2 以降 (バージョン 10.x)、11g (バージョン 11.2.0.3.v1 以降)、および12.2、18c までのOracle バージョン
- ターゲット Amazon RDS for PostgreSQL DB インスタンスの最小要件は次のとおりです：
  - PostgreSQL バージョン 9.4 以降 (バージョン 9.x 用)、10.x、11.x。
- このパターンでは、Oracle SQL Developer を使用します。他のツールを使用してスキーマの差分を見つけてエクスポートすると、結果が異なる場合があります。
- Oracle SQL Developer より生成される 「SQL スクリプト」では、変換エラーが発生する可能性があるため、手動で移行を行う必要があります。
- AWS SCT のソースとターゲットのテスト接続が失敗した場合、着信トラフィックを受け入れるために、JDBC ドライバーバージョンと Virtual Private Cloud (VPC) セキュリティグループの受信ルールを設定することを確保します。

製品バージョン

- Amazon RDS for Oracle DB インスタンスバージョン 12.1.0.2 (バージョン 10.2 以降)
- Amazon RDS for PostgreSQL DB インスタンスバージョン 11.5 (バージョン 9.4 以降)
- Oracle SQL Developer バージョン 19.1 以降
- AWS SCT バージョン 1.0.632 以降

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for Oracle DB インスタンス

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL インスタンス

ソースアーキテクチャとターゲットアーキテクチャ

次の図表は、Amazon RDS for Oracle DB インスタンスを Amazon RDS for PostgreSQL DB インスタンスに移行する方法を示しています。

この図表は、次の移行ワークフローを示しています：

1. Oracle SQL Developer を開き、ソースデータベースとターゲットデータベースに接続します。
2. 「差分レポート」 を生成し、次医スキーマ差分オブジェクトの SQL スクリプトファイルを生成します。差分レポートの詳細については、Oracle ドキュメントの「詳細な差分レポート」 を参照してください。
3. AWS SCT を設定し、Python コードを実行します。
4. SQL スクリプトファイルは Oracle から PostgreSQL に変換します。
5. ターゲット PostgreSQL DB インスタンスで SQL スクリプトファイルを実行します。

自動化とスケール

Python スクリプトに、1 つのプログラム内の複数の機能に関するパラメータやセキュリティ関連の変更を追加することで、この移行を自動化できます。

ツール

- 「AWS SCT」 — AWS Schema Conversion Tool (AWS SCT) は、既存のデータベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換します。
- 「Oracle SQL Developer」 — Oracle SQL Developer は、従来のデプロイメントとクラウドベースのデプロイメントの両方で Oracle データベースの開発と管理を簡素化する統合開発環境 (IDE) です。

Code

`incremental-migration-sct-sql.zip` ファイル (添付) には、このパターンの完全なソースコードが含まれています。

エピック

ソースデータベースのスキーマの差分に対応する SQL スクリプトファイルを作成します。

タスク	説明	必要なスキル
Oracle SQL Developer でデータベース差分を実行します。	1. ソース Oracle DB インスタンスにサインインし、ツールを選択し、次にデータベース差分を選択します。 2. ソース接続でソースデータベースを選択します。 3. デスティネーション接続で、更新した、またはパッチを適用したソースデータベースを選択します。 4. 要件に応じて残りのオプションを設定し、[次へ] を選択してから、[完成] を選択して 差分レポートを生成します。	DBA
SQL スクリプトファイルを生成します。	スクリプトを生成を選択して、SQL ファイルで差分を生成します。 これにより、AWS SCT がデータベースを Oracle から PostgreSQL に変換するために使用する SQL スクリプトファイルが生成されます。	DBA

Python スクリプトを使用して、AWS SCT にターゲット DB オブジェクトを作成します。

タスク	説明	必要なスキル
Windows コマンドプロンプトを使用して AWS SCT を設定します。	1. プレインストールされている AWS SCT フォルダーから `AWSSchemaConversionToolBatch.jar` ファイルをコピーし、作業ディレクトリに貼り付けます。 2. `run_aws_sct_sql.py` ファイルから（`incremental-migration-sct-sql.zip` 添付フォルダーにある）Python コードをデプロイします。これにより、ソースデータベースとターゲットデータベース環境設定の詳細を含む、.xml ファイルと .sct ファイルが `projects` ディレクトリに作成されます。また、Oracle SQL Developer で生成した SQL スクリプトファイルも読み取られます。最後に、`output` ディレクトリに .sql ファイルオブジェクトを作成します。 3. 次の形式を使用して、`database_migration.txt` ファイル内のソース環境設定とターゲット環境設定の詳細を設定します:	DBA

タスク	説明	必要なスキル

	`#source_vendor,source_hostname,source_dbname,source_user,source_pwd,source_schema,source_port,source_sid,target_vendor,target_hostname,target_user,target_pwd,target_dbname,target_port`
	
	`ORACLE,myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com,ORCL,orcl,orcl1234,orcl,1521,ORCL,POSTGRESQL,mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com,pguser,pgpassword,pgdb,5432`
	
	4. 要件に応じて AWS SCT 設定パラメータを変更し、SQL スクリプトファイルを `input` サブディレクトリの作業ディレクトリにコピーします。	

タスク	説明	必要なスキル
Python スクリプトを実行する。	1. 次のコマンドを使用して、Python スクリプトを実行します：「`$ python run_aws_sct_sql.py database_migration.txt` 」 2. これにより、 DB オブジェクトの SQL ファイルが作成されます。変換エラーの未変換コードは手動で変換できます。	DBA
Amazon RDS for PostgreSQL でオブジェクトを作成	SQL ファイルを実行し、Amazon RDS for PostgreSQL DB インスタンスにオブジェクトを作成します。	DBA

関連リソース

- Amazon RDS 上の Oracle
- 「Amazon RDS 上の PostgreSQL」
- 「AWS SCT ユーザーインターフェイスの使用」
- 「 AWS SCTのソースとしての Oracle の使用」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Aurora PostgreSQL 互換のファイルエンコーディングを使用して BLOB ファイルを TEXT にロード

作成者：Bhanu Ganesh Gudivada (AWS) と Jeevan Shetty (AWS)

概要

多くの場合、移行中に、ローカルファイルシステムのファイルからロードされた非構造化データや構造化データを処理しなければならない場合があります。データがデータベースの文字セットと異なる文字セットになっている場合もあります。

これらのファイルには以下の種類のデータが格納されています。

- メタデータ – このデータはファイル構造を記述します。
- 半構造化データ – JSON や XML などの特定の形式のテキスト文字列です。このようなデータについて、「常に '<' で始まる」や「改行文字を一切含まない」などのアサーションができる場合があります。
- 全文 – このデータには通常、改行文字や引用符を含むあらゆる種類の文字が含まれます。UTF-8 のマルチバイト文字で構成されている場合もあります。
- バイナリデータ – このデータには、バイト、または NULL とファイル末尾マーカーを含むバイトの組み合わせが含まれる場合があります。

これらの種類のデータが混在してロードするのは難しい場合があります。

このパターンでは、オンプレミスの Oracle データベース、Amazon Web Services (AWS) クラウドの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにある Oracle データベース、および Oracle データベースの Amazon Relational Database Service (Amazon RDS) で使用できます。例として、このパターンでは Amazon Aurora PostgreSQL 互換エディションを使用しています。

Oracle データベースでは、`BFILE` (バイナリファイル) ポインタ、`DBMS_LOB` パッケージ、および Oracle システム関数を使用して、ファイルからロードし、文字エンコーディングを使用して CLOB に変換できます。PostgreSQL は Amazon Aurora PostgreSQL 互換エディションデータベースへの移行時に BLOB データ型が適用されないため、これらの関数は PostgreSQL 互換のスクリプトに変換する必要があります。

このパターンでは、Amazon Aurora PostgreSQL 互換データベースの 1 つのデータベース列にファイルをロードする方法が 2 つあります。

- 方法 1 —エンコードオプション付けの aws_s3 拡張機能の table_import_from_s3 関数を使用して、 Amazon Simple Storage Service (Amazon S3) バケットからデータをインポートします。
- 方法 2 – データベースの外部で 16 進数にエンコードし、次にデータベース内で TEXT を表示するようにデコードします。

Aurora PostgreSQL 互換は、aws_s3 の拡張機能つ直接統合されているため、アプローチ 1 を使用することを推薦します。

このパターンでは、マルチバイト文字と独自の形式を持つ E メールテンプレートを含むフラットファイルを Amazon Aurora PostgreSQL 互換データベースにロードする例を使用します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon RDS インスタンスまたは Aurora PostgreSQL 互換インスタンス
- SQL およびリレーショナルデータベース管理システム (RDBMS) の基本的な理解
- 1 つの Amazon Simple Storage Service (Amazon S3) バケット
- OracleとPostgreSQLのシステム機能に関する知識
- RPM Package HexDump-XXD-0.1.1 (Amazon Linux 2 に含まれています)

> **Note**
>
> Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。

機能制限

- TEXT データ型の場合、保存できる最長の文字列は約 1 GB です。

製品バージョン

- Auroraには Amazon Aurora PostgreSQL 更新 に記載されている PostgreSQL バージョンが適用されます。

アーキテクチャ

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換

ターゲット アーキテクチャ

アプローチ 1 – aws_s_s3.table_import_from_s3 を使用

オンプレミスサーバーから、マルチバイト文字とカスタムフォーマットの E メールテンプレートを含むファイルが Amazon S3 に転送されます。このパターンで提供されるカスタムデータベース関数は、`file_encoding` 付けの `aws_s3.table_import_from_s3` 関数を使用して、ファイルをデータベースにロードし、クエリ結果を TEXT データ型として返します。

1. ファイルはステージング S3 バケットに転送されます。
2. ファイルは Amazon Aurora PostgreSQL 互換データベースにアップロードされます。
3. pgAdmin クライアントを使用して、カスタム関数 `load_file_into_clob` が Aurora データベースにデプロイされます。
4. カスタム関数が内部的に file_encoding `table_import_from_s3` と組み合わせて使用します。この関数からの出力は、`array_to_string` と `array_agg` を使用して、TEXT 出力として取得されます。

方法 2 – データベースの外部では 16 進数にエンコードし、データベース内の TEXT を表示するにはデコードします。

オンプレミスサーバーまたはローカルファイルシステムのファイルは 16 進ダンプに変換されます。次に、ファイルが TEXT フィールドとして、 PostgreSQL にインポートされます。

1. `xxd -p` オプションを使用して、コマンドラインでファイルを 16 進数ダンプに変換します。
2. `\copy` オプションを使用して、 16 進数ダンプファイルを Aurora PostgreSQL 互換にアップロードし、16 進数ダンプファイルをバイナリにデコードします。
3. バイナリデータを TEXT エンコードしてとして返します。

ツール

AWS サービス

- Amazon Aurora PostgreSQL 互換版は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型のリレーショナルデータベースエンジンです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

その他のツール

- PgAdmin4 は PostgreSQL のオープンソースの管理および開発プラットフォームです。PgAdmin4 は Linux、UNIX、Mac OS、および Windows で使用して PostgreSQL を管理できます。

エピック

アプローチ 1：Amazon S3 からの Aurora PostgreSQL 互換のデータをインポート

タスク	説明	必要なスキル
EC2 インスタンスを起動します。	インスタンスを起動する手順については、インスタンスを起動する を参照してください。	DBA
PostgreSQL クライアント pgAdmin ツールをインストールします。	pgAdmin をダウンロードし、インストールします。	DBA
IAM ポリシーを作成します。	ファイルが保存される S3 バケットへのアクセスを許可する `aurora-s3-access-pol` と名前付けられたAWS アイデンティティアクセス管理(IAM) ポリシーを作成します。次のコードを使用して、S3 バケットの名で	DBA

タスク	説明	必要なスキル

	<bucket-name> を置き換わります。
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"s3:GetObject",
	"s3:AbortMultipartUpload",
	"s3:DeleteObject",
	"s3:ListMultipartUploadParts",
	"s3:PutObject",
	"s3:ListBucket"
	],
	"Resource": [
	"arn:aws:s3:::<bucket-name>/*",
	"arn:aws:s3:::<bucket-name>"
	]
	}
	]
	}	

タスク	説明	必要なスキル
Amazon S3 から Aurora PostgreSQL 互換にオブジェクトをインポートするための IAM ロールを作成します。	次のコードを使用して、AssumeRole 信頼関係を持ち、`aurora-s3-import-role` と名前付けられた IAM ロールを作成します。`AssumeRole` は、ユーザーの代わりに Aurora が他の AWS のサービスにアクセスすることを許可します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow","Principal": { "Service": "rds.amazonaws.com" },"Action": "sts:AssumeRole" } ] }`	DBA

タスク	説明	必要なスキル
IAM ロールをクラスターに関連付けます。	IAM ロールを Aurora PostgreSQL 互換データベースクラスターに関連付けるには、次の AWS CLI コマンドを実行します。Aurora PostgreSQL 互換データベースをホストする AWS アカウントの ID を <Account-ID> に変更します。これにより、Aurora PostgreSQL 互換データベースが S3 バケットにアクセスできるようになります。 `aws rds add-role-to-db-cluster --db-cluster-identifier aurora-postgres-cl --feature-name s3Import --role-arn arn:aws:iam::<Account-ID>:role/aurora-s3-import-role`	DBA
このファイルを Amazon S3 にアップロードします。	1. このパターンの追加情報セクションでは、`salary.event.notification.email.vm` と名前付けられたファイルに電子メールテンプレートのコードをコピーします。 2. ファイルを S3 バケットにアップロードします。	DBA、アプリ所有者

タスク	説明	必要なスキル
カスタム関数をデプロイします。	1. 追加情報セセクションから、カスタム関数の `load_file_into_clob` SQL ファイルの内容を一時テーブルにコピーします。 2. Aurora PostgreSQL 互換データベースにログインし、pgAdmin クライアントを使用してデータベーススキーマにデプロイします。	アプリ所有者、DBA

タスク	説明	必要なスキル

データをデータベースにインポートするために、カスタム関数を実行します。	次の SQL コマンドを実行し、山括弧内の項目を適切な値に置き換えます。
	
	`select load_file _into_clob('aws-s3 -import-test'::tex t,'us-west-1'::tex t,'employee.salary .event.notificatio n.email.vm'::text);`
	
	コマンドを実行する前に、山括弧内の項目を、次の例に示されているように、適切な値に置き換わります。
	
	`Select load_file _into_clob('aws-s3 -import-test'::tex t,'us-west-1'::tex t,'employee.salary .event.notificatio n.email.vm'::text);`
	
	このコマンドは Amazon S3 からファイルをロードし、TEXT として出力を返します。
		アプリ所有者、DBA

アプローチ 2：テンプレートファイルをローカル Linux システムの 16 進数ダンプに変換

タスク	説明	必要なスキル
テンプレートファイルを 16 進数ダンプに変換します。	**Note** Hexdump ユーティリティは、バイナリファイルの内容を 16 進数、10 進数、8 進数、または ASCII 形式で表示します。 `hexdump` コマンドが `util-linux` パッケージの一部で、、Linux ディストリビューションにプリインストールされています。Hexdump RPM パッケージは Amazon Linux 2 の一部でもあります。(: Amazon Linux 2 はサポート終了に近づいています。 詳細については、「Amazon Linux 2 のFAQs」を参照してください。) ファイルの内容を 16 進数ダンプに変換するには、次のシェルコマンドを実行します。 `xxd -p </path/file.vm> | tr -d '\n' > </path/file.hex>`	DBA

タスク	説明	必要なスキル
	パスとファイルを次の例に示されているように、適切な値に置き換わります。 `xxd -p employee.salary.event.notification.email.vm | tr -d '\n' > employee.salary.event.notification.email.vm.hex`	

タスク	説明	必要なスキル
hexdump ファイルをデータベーススキーマにロードします。	次のコマンドを使用して、hexdump ファイルを Aurora PostgreSQL 互換データベースにロードします。 1. Aurora PostgreSQL データベースにログインし、`email_template_hex` という新しいテーブルを作成します。 ``` CREATE TABLE email_template_hex(hex_data TEXT); ``` 2. 次のコマンドを使用して、ローカルファイルシステムからファイルを DB スキーマにロードします。 ``` \copy email_template_hex FROM '/path/file.hex'; ``` パスをローカルファイルシステム上の場所に置き換えます。 ``` \copy email_template_hex FROM '/tmp/employee.salary.event.notification.email.vm.hex'; ``` 3. `email_template_bytea` というテーブルをもう 1 つ作成します。	DBA

タスク	説明	必要なスキル
	CREATE TABLE email_template_bytea(hex_data bytea); 4. email_template_hex からデータを email_template_bytea に挿入します。 INSERT INTO email_template_bytea (hex_data) (SELECT decode(hex_data, 'hex') FROM email_template_hex limit 1); 5. 16進数バイトコードを TEXT データとして返すには、次のコマンドを実行します。 SELECT encode(hex_data::bytea, 'escape') FROM email_template_bytea;	

関連リソース

リファレンス

- PostgreSQL データベースを AWS Database Migration Service のソースとして使用する
- Oracle データベース 19c から PostgreSQL 互換 (12.4) 対応 Amazon Aurora への移行プレイブック

- IAM ポリシーの作成
- IAM ロールと Amazon Aurora MySQL DB クラスターの関連付け
- pgAdmin

チュートリアル

- 「Amazon RDS の開始方法」
- Oracle から Amazon Aurora への移行

追加情報

load_file_into_clob カスタム関数

```
CREATE OR REPLACE FUNCTION load_file_into_clob(
    s3_bucket_name text,
    s3_bucket_region text,
    file_name text,
    file_delimiter character DEFAULT '&'::bpchar,
    file_encoding text DEFAULT 'UTF8'::text)
    RETURNS text
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
    blob_data BYTEA;
    clob_data TEXT;
    l_table_name CHARACTER VARYING(50) := 'file_upload_hex';
    l_column_name CHARACTER VARYING(50) := 'template';
    l_return_text TEXT;
    l_option_text CHARACTER VARYING(150);
    l_sql_stmt CHARACTER VARYING(500);

BEGIN

    EXECUTE format ('CREATE TEMPORARY TABLE %I (%I text, id_serial serial)',
 l_table_name, l_column_name);

    l_sql_stmt := 'select ''(format text, delimiter ''''' || file_delimiter || ''''',
 encoding ''''' || file_encoding ||  ''''')'' ';
```

```
    EXECUTE FORMAT(l_sql_stmt)
    INTO l_option_text;

    EXECUTE FORMAT('SELECT aws_s3.table_import_from_s3($1,$2,$6,
 aws_commons.create_s3_uri($3,$4,$5))')
    INTO l_return_text
    USING l_table_name, l_column_name, s3_bucket_name,
 file_name,s3_bucket_region,l_option_text;

    EXECUTE format('select array_to_string(array_agg(%I order by id_serial),E''\n'')
 from %I', l_column_name, l_table_name)
    INTO clob_data;

    drop table file_upload_hex;

    RETURN clob_data;
END;
$BODY$;
```

E メールテンプレート

```
##############################################################################################
##
 ##
##    johndoe Template Type: email
 ##
##    File: johndoe.salary.event.notification.email.vm
 ##
##    Author: Aimée Étienne    Date 1/10/2021
      ##
##  Purpose: Email template used by EmplmanagerEJB to inform a johndoe they         ##
##        have been given access to a salary event
 ##
##    Template Attributes:
 ##
##        invitedUser - PersonDetails object for the invited user
  ##
##        salaryEvent - OfferDetails object for the event the user was given access
 ##
##        buyercollege - CompDetails object for the college owning the salary event
 ##
```

```
##        salaryCoordinator - PersonDetails of the salary coordinator for the event
 ##
##        idp - Identity Provider of the email recipient
 ##
##        httpWebRoot - HTTP address of the server
 ##
##
 ##
################################################################################

$!invitedUser.firstname $!invitedUser.lastname,

Ce courriel confirme que vous avez ete invite par $!salaryCoordinator.firstname $!
salaryCoordinator.lastname de $buyercollege.collegeName a participer a l'evenement
 "$salaryEvent.offeringtitle" sur johndoeMaster Sourcing Intelligence.

Votre nom d'utilisateur est $!invitedUser.username

Veuillez suivre le lien ci-dessous pour acceder a l'evenement.

${httpWebRoot}/myDashboard.do?idp=$!{idp}

Si vous avez oublie votre mot de passe, utilisez le lien "Mot de passe oublie" situe
 sur l'ecran de connexion et entrez votre nom d'utilisateur ci-dessus.

Si vous avez des questions ou des preoccupations, nous vous invitons a
 communiquer avec le coordonnateur de l'evenement $!salaryCoordinator.firstname $!
salaryCoordinator.lastname au ${salaryCoordinator.workphone}.

*******

johndoeMaster Sourcing Intelligence est une plateforme de soumission en ligne pour les
 equipements, les materiaux et les services.

Si vous avez des difficultes ou des questions, envoyez un courriel a
 support@johndoeMaster.com pour obtenir de l'aide.
```

AWS SCT と AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行する AWS CLIAWS CloudFormation

作成者: Pinesh Singal (AWS)

概要

このパターンは、 AWS Command Line Interface（) を使用して、Oracle DB インスタンス用のマルチテラバイト Amazon Relational Database Service (Amazon RDS) を Amazon RDS for PostgreSQL DB インスタンスに移行する方法を示していますAWS CLI。このアプローチはダウンタイムを最小限に抑え、 にサインインする必要はありません AWS Management Console。

このパターンは、 AWS Schema Conversion Tool（ AWS SCT) コンソールと AWS Database Migration Service（ AWS DMS) コンソールを使用して、手動設定や個々の移行を回避するのに役立ちます。このソリューションは、複数のデータベースに 1 回限りの設定をセットアップし、 AWS DMS で AWS SCT と を使用して移行を実行します AWS CLI。

このパターンでは、 を使用してデータベーススキーマオブジェクト AWS SCT を Amazon RDS for Oracle から Amazon RDS for PostgreSQL に変換し、 AWS DMS を使用してデータを移行します。で Python スクリプトを使用して AWS CLI、 AWS CloudFormation テンプレートを使用して AWS SCT オブジェクトと AWS DMS タスクを作成します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 既存の Amazon RDS for Oracle DB インスタンス。
- 既存の Amazon RDS for PostgreSQL DB インスタンス。
- スクリプトを実行するための Windows または Linux OS を搭載した Amazon Elastic Compute Cloud (Amazon EC2) インスタンスまたはローカルマシン。
- 次の AWS DMS 移行タスクタイプの理解: `full-load`、`cdc`、`full-load-and-cdc`。 詳細については、 AWS DMS ドキュメントの「タスクの作成」を参照してください。
- AWS SCT Oracle および PostgreSQL データベースエンジン用の Java Database Connectivity (JDBC) ドライバーを使用してインストールおよび設定されている 。詳細については、 AWS SCT ドキュメントの「インストールと設定 AWS SCT」を参照してください。
- インストールされた AWS SCT フォルダから作業ディレクトリにコピーされた `AWSSchemaConversionToolBatch.jar` ファイル。

- `cli-sct-dms-cft.zip` ファイル (添付) はダウンロードされ、作業ディレクトリに抽出されます。
- レ AWS DMS プリケーションインスタンスエンジンの最新バージョン。詳細については、 AWS サポート ドキュメントとAWS DMS リリースノートのAWS DMS 「レプリケーションインスタンスの作成方法」を参照してください。
- AWS CLI バージョン 2。スクリプトが実行されている EC2 インスタンスまたは OS のアクセスキー ID、シークレットアクセスキー、およびデフォルト AWS リージョン 名を使用してインストールおよび設定されます。詳細については、 AWS CLI ドキュメントの「 の最新バージョンのインストールまたは更新 AWS CLI」および「 の設定の構成 AWS CLI」を参照してください。
- AWS CloudFormation テンプレートに精通していること。詳細については、 AWS CloudFormation ドキュメントの「 AWS CloudFormation の仕組み」を参照してください。
- Python バージョン 3。スクリプトが実行されている EC2 インスタンスまたは OS にインストールされ、設定されます。詳細については、Python のドキュメントを参照してください。

機能制限

- ソースである Amazon RDS for Oracle DB インスタンスの最小要件は次のとおりです:
  - Enterprise、Standard、Standard One、Standard Two エディションの Oracle バージョン 12c (12.1.0.2、12.2.0.1)、18c (18.0.0.0)、および 19c (19.0.0.0)。
  - Amazon RDS は Oracle 18c (18.0.0.0) をサポートしていますが、Oracle はend-of-support以降に 18c のパッチを提供しなくなったため、このバージョンは非推奨パスにあります。詳細については、Amazon RDS ドキュメントの「Amazon RDS for Oracle」 を参照してください。
  - Amazon RDS for Oracle 11g はサポートされなくなりました。
- ターゲット Amazon RDS for PostgreSQL DB インスタンスの最小要件は次のとおりです。
  - PostgreSQL バージョン 9 (9.5 および 9.6)、10.x、11.x、12.x、および 13.x

製品バージョン

- Amazon RDS for Oracle DB インスタンスバージョン 12.1.0.2 以降
- Amazon RDS for PostgreSQL DB インスタンスバージョン 11.5 以降
- AWS CLI バージョン 2
- の最新バージョン AWS SCT
- Python 3 の 最新バージョン

アーキテクチャ

ソーステクノロジースタック

- 「Amazon RDS for Oracle」

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、 AWS DMS および Python スクリプトを使用した Amazon RDS for Oracle DB インスタンスの Amazon RDS for PostgreSQL DB インスタンスへの移行を示しています。

この図表は、次の移行ワークフローを示しています：

1. Python スクリプトは、 AWS SCT を使用してソース DB インスタンスとターゲット DB インスタンスに接続します。
2. ユーザーは Python スクリプト AWS SCT で開始し、Oracle コードを PostgreSQL コードに変換して、ターゲット DB インスタンスで実行します。
3. Python スクリプトは、ソース DB インスタンスとターゲット DB インスタンスの AWS DMS レプリケーションタスクを作成します。
4. ユーザーは Python スクリプトをデプロイして AWS DMS タスクを開始し、データ移行の完了後にタスクを停止します。

自動化とスケール

この移行を自動化するには、Python スクリプトにパラメータとセキュリティ関連の変更を追加して、機能を追加します。

ツール

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドを通じて AWS のサービスを操作するのに役立つオープンソースツールです。
- AWS CloudFormation は、AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じて管理するために役立

ちます。このパターンは、Python スクリプトを使用して.csv入力ファイルを.json入力ファイルに変換します。.json ファイルは、Amazon リソースネーム (ARNs)、移行タイプ、タスク設定、テーブルマッピングを使用して複数の AWS DMS レプリケーションタスクを作成する AWS CloudFormation スタックを作成する AWS CLI コマンドで使用されます。

- AWS Database Migration Service （ AWS DMS ） は、データストアを AWS クラウド に移行するか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役立ちます。このパターンでは、 AWS DMS を使用して、コマンドラインで実行される Python スクリプトを使用してタスクを作成、開始、停止し、 AWS CloudFormation テンプレートを作成します。
- AWS Schema Conversion Tool （ AWS SCT ） は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行をサポートします。このパターンでは、インストールされた AWS SCT ディレクトリの AWSSchemaConversionToolBatch.jar ファイルが必要です。

Code

cli-sct-dms-cft.zip ファイル (添付) には、このパターンの完全なソースコードが含まれています。

エピック

でデータベースオブジェクトを設定 AWS SCT および作成する AWS CLI

タスク	説明	必要なスキル
から実行 AWS SCT するように を設定します AWS CLI。	1. 次の形式を使用して、database_migration.txt ファイル内のソース環境設定とターゲット環境設定の詳細を設定します: `#source_vendor,source_hostname,source_dbname,source_user,source_pwd,source_schema,source_port,source_sid,target_vendor,target_hostname,target_us`	DBA

タスク	説明	必要なスキル
	er,target_pwd,targ et_dbname,target_p ort ORACLE,myoracl edb.cokmvis0v46q.u s-east-1.rds.amazo naws.com,ORCL,orcl ,orcl1234,orcl,152 1,ORCL,POSTGRESQL, mypgdbinstance.cok mvis0v46q.us-east- 1.rds.amazonaws.co m,pguser,pgpasswor d,pgdb,5432 2. 、project_settings.x ml 、 の各ファイルの要 件に従ってOracle_PG _Test_Batch.xml 、 AWS SCT 設定パラメータ を変更しますORACLE-or cl-to-POSTGRESQL.x ml 。	

タスク	説明	必要なスキル
Python `run_aws_sct.py` スクリプトを実行します。	次のコマンドを使用して、`run_aws_sct.py` Python スクリプトを実行します。 `$ python run_aws_sct.py database_migration.txt` Python スクリプトは、データベースオブジェクトを Oracle から PostgreSQL に変換し、PostgreSQL 形式の SQL ファイルを作成します。このスクリプトは PDF ファイル も作成します。これにより`Database migration assessment report` 、データベースオブジェクトの詳細なレコメンデーションと変換統計が提供されます。	DBA
Amazon RDS for PostgreSQL にオブジェクトを作成します。	1. 必要に応じて AWS SCT、によって生成された SQL ファイルを手動で変更します。 2. SQL ファイルを実行し、Amazon RDS for PostgreSQL DB インスタンスにオブジェクトを作成します。	DBA

および を使用して AWS DMS タスクを設定 AWS CLI および作成する AWS CloudFormation

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	にサインインし AWS Management Console、AWS DMS コンソールを開き、要件に応じて設定されたレプリケーションインスタンスを作成します。 詳細については、 AWS DMS ドキュメントの「レプリケーションインスタンスの作成」および AWS サポート ドキュメントのAWS DMS 「レプリケーションインスタンスの作成方法」を参照してください。	DBA
ソースエンドポイントを作成します。	AWS DMS コンソールで、エンドポイントを選択し、要件に応じて Oracle データベースのソースエンドポイントを作成します。 **Note** 追加の接続属性は `-2値numberDataTypeScale` である必要があります。 詳細については、 AWS DMS ドキュメントの「ソースエンドポイントとターゲットエン	DBA

タスク	説明	必要なスキル
	ドポイントの作成」を参照してください。	
ターゲットエンドポイントを作成します。	AWS DMS コンソールで、エンドポイントを選択し、要件に応じて PostgreSQL データベースのターゲットエンドポイントを作成します。 詳細については、 AWS DMS ドキュメントの「ソースエンドポイントとターゲットエンドポイントの作成」を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル

から実行する AWS DMS レプリケーションの詳細を設定します AWS CLI。	次の形式を使用して、 AWS DMS ソースエンドポイント ARN、ターゲットエンドポイント ARN、レプリケーションインスタンス ARN を使用して、 `dms-arn-list.txt` ファイル内のソースエンドポイントとターゲットエンドポイント、レプリケーションの詳細を設定します。
	
	`#sourceARN,targetARN,repARN`
	`arn:aws:dms:us-east-1:123456789012:endpoint:EH7AINRUDZ5GOYIY6HVMXECMCQ`
	`arn:aws:dms:us-east-1:123456789012:endpoint:HHJVUV57N7O3CQF4PJZKGIOYY5`
	`arn:aws:dms:us-east-1:123456789012:rep:LL57N77AQQAHHJF4PJFHNEDZ5G`
		DBA

タスク	説明	必要なスキル
Python dms-create-task.py スクリプトを実行して AWS DMS タスクを作成します。	1. 次のコマンドを使用して、dms-create-task.py Python スクリプトを実行します。 `$ python dms-create-task.py database_migration.txt dms-arn-list.txt <cft-stack-name> <migration-type>` 各パラメータの意味は次のとおりです。 • database_migration.txt はデータベース移行テキストファイルです。 • dms-arn-list.txt はの ARN リストです AWS DMS。 • <cft-stack-name> はユーザー定義の AWS CloudFormation スタック名です。 • <migration-type> が、full-load 、cdc、または full-load-and-cdc の場合。 2. 移行タイプに応じて、次のコマンドを使用して 3 種類の AWS DMS タスクを作成できます。	DBA

タスク	説明	必要なスキル
	• `$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack full-load` • `$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack cdc` • `$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack full-load-and-cdc`	
AWS DMS タスクの準備が整っていることを確認します。	AWS DMS コンソールで、Readyステータス セクションで AWS DMS タスクのステータスを確認します。	DBA

を使用して AWS DMS タスクを開始および停止する AWS CLI

タスク	説明	必要なスキル
AWS DMS タスクを開始します。	次のコマンドを使用して、`dms-start-task.py` Python スクリプトを実行します。	DBA

タスク	説明	必要なスキル
	`$ python dms-start-task.py start '<cdc-start-datetime>'` **Note** 開始日時は 'DD-MON-YYYY' または 'YYYY-MM-DDTHH:MI:SS' 形式 (例: '01-Dec-2019' または '2018-03-08T12:12:12') である必要があります。 AWS DMS タスクのステータスは、 AWS DMS コンソールのタスクページのテーブル統計タブで確認できます。	

タスク	説明	必要なスキル
データを検証します。	1. フルロード移行が完了すると、タスクは CDC に対して継続的に実行され続けます。 2. CDC が完了するか、移行する必要がなくなったら、移行タスクの結果と Oracle および PostgreSQL データベース内のデータを確認して検証します。 AWS DMS コンソールのタスクページのデータベース移行タスクのテーブル統計タブでステータスとカウント列 (`Validation state` 、`Validation pending`、`Validation failed`、、、`Validation details`) を確認することで`Validation suspended` 、データを検証できます。 詳細については、 AWS DMS ドキュメントAWS DMS のデータ検証を参照してください。	DBA

タスク	説明	必要なスキル
AWS DMS タスクを停止します。	次のコマンドを使用して、Python スクリプトを実行します。 `$ python dms-start-task.py stop` **Note** AWS DMS タスクは、検証failedステータスに応じて、 ステータスで停止することがあります。詳細については、次のセクションをご覧ください。	DBA

トラブルシューティング

問題	ソリューション
AWS SCT ソースとターゲットのテスト接続が失敗します。	受信トラフィックを受け入れるように JDBC ドライバーバージョンと VPC セキュリティグループのインバウンドルールを設定します。
ソースまたはターゲットエンドポイントのテスト実行が失敗します。	エンドポイント設定とレプリケーションインスタンスのステータスを確認します `Available` 。エンドポイントの接続ステータスが `Successful` であるかどうかを確認します。 詳細については、 AWS サポート ドキュメントの「AWS DMS エンドポイントの接続障害のト

問題	ソリューション
	ラブルシューティング方法」を参照してください。
全ロード実行は失敗します。	ソースデータベースとターゲットデータベースのデータ型とサイズが一致しているかどうかを確認します。 詳細については、 AWS DMS ドキュメントの「 での移行タスクのトラブルシューティング AWS DMS」を参照してください。
検証実行エラーが発生します。	プライマリキー以外のテーブルは検証されないため、テーブルにプライマリキーがあるかどうかを確認します。 テーブルにプライマリキー、エラーがある場合は、ソースエンドポイントの追加の接続属性に `numberDataTypeScale=-2` があることを確認してください。 詳細については、 AWS DMS ドキュメントの「Oracle をソースとして使用する場合のエンドポイント設定 AWS DMS」、OracleSettings」、および「トラブルシューティング」を参照してください。

関連リソース

- のインストールと設定 AWS SCT
- の概要 AWS DMS (ビデオ)
- AWS CLI および PowerShell の CloudFormation スタックオペレーションコマンドの例
- のユーザーインターフェイスの操作 AWS SCT
- のソースとしての Oracle データベースの使用 AWS DMS
- を使用した Oracle データベースへの接続 AWS SCT
- のターゲットとしての PostgreSQL データベースの使用 AWS DMS

- 「データ移行のソース」
- 「データ移行のターゲット」
- クラウドフォーメーション (AWS CLI ドキュメント)
- create-stack (AWS CLI ドキュメント)
- dms (AWS CLI ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します

作成者: Pinesh Singal (AWS)

概要

このパターンは、Amazon Relational Database Service (Amazon RDS) を、Amazon Amazon Web Services (AWS) クラウド上の Amazon RDS for PostgreSQL データベースに移行する際のガイダンスを提供します。データベース間の接続を暗号化するために、このパターンでは Amazon RDS と AWS Database Migration Service (AWS DMS) の認証局 (CA) と SSL モードを使用します。

このパターンは、トランザクション数が多い数テラバイトの Oracle ソースデータベースに対して、ダウンタイムをほとんどまたはまったく発生させないオンライン移行戦略を示しています。データセキュリティのため、このパターンではデータ転送時に SSL を使用します。

このパターンでは、AWS Schema Conversion Tool (AWS SCT) を使用して Amazon RDS for Oracle データベーススキーマを Amazon RDS for PostgreSQL スキーマに変換します。このパターンでは、AWS SCT を使用してデータを Amazon RDS for Oracle から Amazon RDS for PostgreSQL に移行します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- rds-ca-rsa2048-g1 のみで設定された Amazon RDS データベース認証局 (CA)
  - rds-ca-2019 証明書は 2024 年 8 月に期限切れになりました。
  - rds-ca-2015 証明書は 2020 年 3 月 5 日に期限切れになりました。
- AWS SCT
- AWS DMS
- pgAdmin
- SQL ツール (SQL Developer や SQL*Plus など)

機能制限

- Amazon RDS for Oracle — 最小要件は、エンタープライズエディションおよびスタンダード 2 エディションの Oracle バージョン 19c です。

- Amazon RDS for PostgreSQL — 最小要件は PostgreSQL バージョン 12 以降 (バージョン 9.x 以降用) です。

製品バージョン

- Amazon RDS for Oracle DB インスタンスバージョン 12.1.0.2 以降
- Amazon RDS for PostgreSQL データベースバージョン 11.5 インスタンス

アーキテクチャ

ソーステクノロジースタック

- バージョン 12.1.0.2.v18 の Amazon RDS for Oracle データベースインスタンス。

ターゲットテクノロジースタック

- AWS DMS
- バージョン 11.5 以降のAmazon RDS for PostgreSQL データベースインスタンス

ターゲットアーキテクチャ

次の図は、Oracle (ソース) データベースと PostgreSQL (ターゲット) データベース間のデータ移行アーキテクチャのアーキテクチャを示しています。このアーキテクチャには以下が含まれます。

- 仮想プライベートクラウド (VPC)
- アベイラビリティゾーン
- プライベートサブネット
- Amazon RDS for Oracle のデータベース
- AWS DMS レプリケーションインスタンスを作成します。
- RDS for PostgreSQL データベース

ソースデータベースとターゲットデータベースの接続を暗号化するには、Amazon RDS と AWS DMS で CA と SSL モードを有効にする必要があります。

ツール

AWS サービス

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「OracleのAmazon Relational Database Service (Amazon RDS)」 によって、AWS クラウドでOracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。
- 「Amazon Relational Database Service (Amazon RDS)」 を使用して、AWS クラウドでの PostgreSQL リレーショナルデータベースをセットアップ、運用、スケーリングできます。
- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。

その他のサービス

- 「pgAdmin」 は PostgreSQL 用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

ベストプラクティス

Amazon RDS では、AWS セキュリティのベストプラクティスとして、新しい CA 証明書を提供しています。新しい証明書とサポートされている AWS リージョンの詳細については、「SSL/TLS を使用して DB インスタンスまたはクラスターへの接続を暗号化する」を参照してください。

RDS インスタンスが現在 CA 証明書 にあり`rds-ca-2019`、 にアップグレードする場合は`rds-ca-rsa2048-g1`、「DB インスタンスまたはクラスターを変更して CA 証明書を更新する」または「メンテナンスを適用して CA 証明書を更新する」の手順に従います。

エピック

Amazon RDS for Oracle インスタンスの設定

タスク	説明	必要なスキル
Oracle データベースインスタンスを作成します。	AWS アカウントにサインインし、AWS マネジメントコンソールを開いて Amazon RDS コ	AWS 全般、DBA

タスク	説明	必要なスキル
	ンソールに移動します。コンソールで データベースの作成を選択し、Oracle を選択します。	
セキュリティグループを設定します。	インバウンドおよびアウトバウンドのセキュリティグループを設定します。	AWS 全般
オプショングループを作成します。	Amazon RDS for Oracle データベースと同じ VPC およびセキュリティグループにオプショングループを作成します。オプションには SSL を選択します。ポートには 2484 (SSL 接続用) を選択します。	AWS 全般
オプション設定を行います。	以下の設定を使用します。 • `SQLNET.CIPHER_SUITE : SSL_RSA_WITH_AES_256_CBC_SHA` • `SQLNET.SSL_VERSION : 1.2 or 1.0`	AWS 全般
RDS for Oracle DB インスタンスを変更します。	CA 証明書を rds-ca-rsa2048-g1 に設定します。 オプショングループ で、以前に作成したオプショングループを添付します。	AWS 全般、DBA

タスク	説明	必要なスキル

RDS for Oracle DB インスタンスが使用可能であることを確認します。	Amazon RDS for Oracle データベースインスタンスが稼働中であり、データベーススキーマにアクセスできることを確認します。
	
	RDS for Oracle DB に接続するには、sqlplus コマンドラインからコマンドを使用します。
	
	`$ sqlplus orcl/**** @myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com:1521/ORCL`
	`SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 15 18:11:07 2019`
	`Copyright (c) 1982, 2016, Oracle. All rights reserved.`
	`Last Successful login time: Mon Dec 16 2019 23:17:31 +05:30`
	`Connected to:`
	`Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production`
	`With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options`
	`SQL>`
		DBA

タスク	説明	必要なスキル
RDS for Oracle データベースにオブジェクトとデータを作成します。	オブジェクトを作成し、スキーマにデータを挿入します。	DBA

Amazon RDS for PostgreSQL インスタンスの設定

タスク	説明	必要なスキル
RDS for PostgreSQL データベースを作成します。	Amazon RDS コンソールのデータベースの作成ページで、PostgreSQL を選択して Amazon RDS for PostgreSQL データベースインスタンスを作成します。	AWS 全般、DBA
セキュリティグループを設定します。	インバウンドおよびアウトバウンドのセキュリティグループを設定します。	AWS 全般
パラメータグループを作成します。	PostgreSQL バージョン 11.x を使用している場合は、パラメータグループを作成して SSL パラメータを設定します。PostgreSQL バージョン 12 では、SSL パラメータグループはデフォルトで有効になっています。	AWS 全般
パラメータの編集。	`rds.force_ssl` パラメータを 1 (オン)に変更します。 デフォルトでは、 `ssl` パラメータが 1 (オン)です。`rds.force_ssl` パラメータを 1 に設定すると、す	AWS 全般

タスク	説明	必要なスキル
	べての接続が SSL モードでのみ接続するように強制されます。	
RDS for PostgreSQL DB インスタンスを更新します。	CA 証明書を rds-ca-rsa2048-g1 に設定します。 PostgreSQL のバージョンに応じて、デフォルトのパラメータグループまたは以前に作成したパラメータグループを添付します。	AWS 全般、DBA

タスク	説明	必要なスキル

RDS for PostgreSQL DB インスタンスが使用可能であることを確認します。	Amazon RDS for PostgreSQL データベースが稼働していることを確認します。
	
	この psql コマンドは、コマンドラインから sslmode set を使用して SSL 接続を確立します。
	
	1 つの方法は、sslmode=1 パラメーターグループに設定し、psql コマンドに sslmode パラメーターを含めずに接続を使用することです。
	
	次の出力は、SSL 接続が確立されたことを示しています。
	
	`$ psql -h mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com -p 5432 "dbname=pgdb user=pguser"`
	`Password for user pguser:`
	`psql (11.3, server 11.5)`
	`SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)`
	`Type "help" for help.`
	`pgdb=>`
		DBA

タスク	説明	必要なスキル

	2 つ目の方法は、sslmode=1 パラメータグループに設定し、その sslmode パラメータを psql コマンドに含めることです。
	
	次の出力は、SSL 接続が確立されたことを示しています。
	
	`$ psql -h mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com -p 5432 "dbname=pgdb user=pguser sslmode=require"`
	`Password for user pguser:`
	`psql (11.3, server 11.5)`
	`SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)`
	`Type "help" for help.`
	`pgdb=>`	

AWS SCT の設定と実行

タスク	説明	必要なスキル
AWS SCT をインストールします。	AWS SCT アプリケーションの最新バージョンをインストールします。	AWS 全般
JDBC ドライバーを使用して AWS SCT を設定します。	Oracle (「ojdbc8.jar」) と PostgreSQL (postgresq	AWS 全般

タスク	説明	必要なスキル
	l-42.2.5.jar) 用の Java データベース接続 (JDBC) ドライバーをダウンロードしてください。 AWS SCT でドライバを設定するには、設定、グローバル設定、ドライバを選択します。	

タスク	説明	必要なスキル
AWS SCT プロジェクトを作成します。	Oracle をソース DB エンジンとして、Amazon RDS for PostgreSQL をターゲット DB エンジンとして使用して、AWS SCT プロジェクトとレポートを作成します。 1. 接続の詳細を指定して、ソース Oracle データベースとターゲット Amazon RDS for PostgreSQL データベースへの接続をテストします。 ソース Oracle データベースには、以下の権限または権限が必要です。 • `CONNECT` • `SELECT_CATALOG_ROLE` • `SELECT ANY DICTIONARY` • `SELECT on SYS.USER$ TO <sct_user>` 詳細については、「AWS SCTのソースとしてのOracle データベースの使用」を参照してください。 AWS SCT が移行レポートを開始する前に、ソース接続とターゲット接続の両方に成功する必要があります。	AWS 全般

タスク	説明	必要なスキル
	2. レポートが表示されたら、変換するスキーマを入力し、Finish を選択します。	
データベースオブジェクトを検証します。	1. スキーマのロードを選択します。 AWS SCT には、エラーのあるオブジェクトを含め、ソースオブジェクトと変換されたターゲットオブジェクトが表示されます。ターゲットデータベース上の不正なオブジェクトをすべて更新します。 2. エラーを確認し、手動操作でクリアします。 3. すべてのエラーがクリアされたら、スキーマの読み込みをもう一度選択します。 4. データベースに適用を選択します。 5. pgAdmin または PostgreSQL DB Connect をサポートする任意のツールに接続し、スキーマとオブジェクトを確認します。	AWS 全般、DBA

AWS DMS の設定と実行

タスク	説明	必要なスキル
レプリケーションインスタンスを作成します。	1. アカウントにサインインして AWS マネジメントコン	AWS 全般

タスク	説明	必要なスキル
	ソールを開いて AWS DMS コンソールに移動します。 2. VPC、セキュリティグループ、アベイラビリティーゾーン、その他の接続属性の有効な設定を使用してレプリケーションインスタンスを作成します。	
証明書をインポートします。	AWS リージョンの証明書バンドル (PEM) をダウンロードします。 バンドルには、`rds-ca-2019` 中間証明書とルート証明書の両方が含まれています。またバンドルには、`rds-ca-rsa2048-g1` 、`rds-ca-rsa4096-g1` 、および `rds-ca-ecc384-g1` ルート CA 証明書も含まれています。アプリケーショントラストストアは、ルート CA 証明書のみを登録する必要があります。	AWS 全般

タスク	説明	必要なスキル
ソースエンドポイントを作成します。	1. RDS DB インスタンスを選択を選択し、作成した RDS for Oracle DB インスタンスを選択して、Amazon RDS for Oracle のソースエンドポイントを作成します。エンドポイント設定の詳細が自動的に入力されます。 2. アクセス情報を手動で提供を選択します。ポートには、必ず 2484 と入力してください。 3. Secure Socket Layer (SSL) モードで、 を選択しverify-ca 、前に作成した CA 証明書を選択します。 4. エンドポイント設定で、サイズなしで NUMBER データ型をサポートする接続属性 NumberDataTypeScale=-2 を追加します。 詳細については、「AWS データベース移行サービスに対して、 Oracle のデータベースをソースとして使用する」を参照してください。	AWS 全般

タスク	説明	必要なスキル
ターゲットエンドポイントを作成します。	1. RDS DB インスタンスを選択を選択することで、Amazon RDS for PostgreSQL のターゲットエンドポイントを作成して、PostgreSQL DB インスタンスの RDSを選択します。エンドポイント設定の詳細が自動的に入力されます。 2. アクセス情報を手動で提供を選択します。ポートには、必ず 2484 と入力してください。 詳細については、「AWS データベース移行サービスのターゲットとして PostgreSQL データベースを使用する」を参照してください。	AWS 全般
エンドポイントをテストします。	1. ソースとターゲットのエンドポイントをテストして、両方が正常に動作し、使用可能であることを確認します。 2. テストに失敗した場合は、セキュリティグループのインバウンドルールが有効であることを確認します。	AWS 全般

タスク	説明	必要なスキル
移行タスクを作成します。	フルロードと変更データキャプチャ (CDC) またはデータ検証用の移行タスクを作成するには、次の手順を実行します。 1. データベース移行タスクを作成するには、レプリケーションインスタンス、ソースデータベースエンドポイント、ターゲットデータベースエンドポイントを選択します。移行タイプを次のいずれかに指定します。 • 既存データを移行 (全ロード) • データ変更のみをレプリケートする • 既存のデータを移行し、継続的変更をレプリケーション (フルロードと CDC ） 2. テーブルマッピングでは、GUI または JSON 形式の選択ルールと変換ルールを設定できます。 • 選択ルールで、スキーマを選択し、テーブル名を入力し、設定するアクション (「含む」または「除外」) を選択します。たとえば、「スキーマ ORCL」、「テーブル	AWS 全般

タスク	説明	必要なスキル
	名%」、「アクションを含む」などです。 • 変換ルールで、以下の一つの作業を実施します。 • スキーマを選択し、アクション (大文字と小文字、接頭辞、接尾辞) を選択します。たとえば、「ターゲットスキーマ ORCL」、「アクションは小文字にする」などです。 • スキーマを選択し、テーブル名を入力して、アクション (大文字と小文字、接頭辞、接尾辞) を選択します。たとえば、ターゲットスキーマ ORCL、テーブル % 、アクションを小文字にするなどです。 3. Amazon CloudWatch Logs モニタリングを有効にします。 4. マッピングルールには、次の JSON コードを追加します。 `{ "rules": [ { "rule-type": "transformation",`	

タスク	説明	必要なスキル
	"rule-id" : "1", "rule-name": "1", "rule-target": "table", "object-locator": { "schema-name": "%", "table-name": "%" }, "rule-action": "convert-lowercase", "value": null, "old-value": null }, { "rule-type": "transformation", "rule-id" : "2", "rule-name": "2", "rule-target": "schema", "object-locator": { "schema-name": "ORCL", "table-name": "%" }, "rule-action": "convert-lowercase",	

タスク	説明	必要なスキル
	"value": null, "old-value": null }, { "rule-type": "selection", "rule-id": "3", "rule-name": "3", "object-locator": { "schema-name": "ORCL", "table-name": "DEPT" }, "rule-action": "include", "filters": [] }] }	
本番稼働を計画する。	アプリケーション所有者などの利害関係者とダウンタイムを確認し、本番システムで AWS DMS を実行してください。	移行リード

タスク	説明	必要なスキル

移行タスクを実行します。	1. ステータスが準備完了の AWS DMS タスクを開始し、Amazon CloudWatch の移行タスクログにエラーがないか監視します。
	
	移行タイプとして 既存のデータを移行し、継続的な変更を複製するを選択し、ステータスが 継続的なレプリケーションのロード完了の場合、CDC データ移行による全ロードが完了し、検証が進行中です。
	2. 移行を開始すると、CloudWatch で追加の SSL 接続情報を取得できます。Oracle に対して、CloudWatch でコネクタの文字列を示しています。
	
	`2019-12-17T09:15:11 [SOURCE_UNLOAD ]I: Connecting to Oracle: Beginning session (oracle_endpoint_conn.c:834)`
	
	PostgreSQL 接続文字列は、次の例のようになります。
	
	`2019-12-17T09:15:11 [TARGET_LOAD ]I:`
		AWS 全般

タスク	説明	必要なスキル
	`Going to connect to ODBC connection string: PROTOCOL=7.4-0;DRIVER={PostgreSQL};SERVER=mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com;DATABASE=pgdb;PORT=5432;sslmode=require;UID=pguser; (odbc_endpoint_imp.c:2218)`	
データを検証します。	移行タスクの結果と、移行元の Oracle データベースと移行先の PostgreSQL データベースのデータを確認します。 1. pgAdmin に接続し、スキーマ `ORCL` を使用して PostgreSQL データベース内のデータを確認します。 2. CDC の場合は、ソース Oracle データベースにデータを挿入または更新して、進行中の変更を確認します。	DBA
移行タスクを停止します。	データ検証が正常に完了したら、移行タスクを停止します。	AWS 全般

リソースをクリーンアップします。

タスク	説明	必要なスキル
AWS DMS タスクを削除します。	1. AWS DMS コンソールでデータベース移行タスクに移動し、進行中または実行中の AWS DMS タスクを停止します。 2. タスク、または複数のタスクを選択し、アクションを選択して、削除を選択します。	AWS 全般
AWS DMS エンドポイントを削除します。	作成したソースエンドポイントとターゲットエンドポイントを選択し、[Actions] を選択し、[Delete] を選択します。	AWS 全般
AWS DMS レプリケーションインスタンスを削除します。	レプリケーションインスタンスを選択し、Actions を選択し、Delete を選択します。	AWS 全般
PostgreSQL データベースを削除します。	1. Amazon RDS コンソールで、データベースを選択します。 2. 作成した PostgreSQL データベースインスタンスを選択し、アクションを選択してから削除を選択します。	AWS 全般
Oracle データベースを削除します。	Amazon RDS コンソールで Oracle データベースインスタンスを選択し、アクションを選択し、削除を選択します。	AWS 全般

トラブルシューティング

問題	ソリューション
AWS SCT のソースとターゲットのテスト接続が失敗しています。	受信トラフィックを受け入れるように JDBC ドライバーバージョンと VPC セキュリティグループのインバウンドルールを設定します。
ソースまエンドポイントのテスト実行が失敗しました。	エンドポイントの設定と、レプリケーションインスタンスが使用可能かどうかを確認します。
AWS DMS タスクの全ロード実行が失敗します。	ソースデータベースとターゲットデータベースに、一致しているデータ型とサイズがあるかを確認します。
AWS DMS 検証移行タスクがエラーを返します。	1. テーブルにプライマリキーがあるかどうかを確認します。プライマリキーのないテーブルは検証されません。 2. テーブルには、プライマリキーがありますが、エラーが返された場合、ソースエンドポイントの追加の接続属性を確認します。追加の接続属性は、テーブル内のデータに基づいて動的にサイズなしで NUMBER データ型をサポートするために `numberDataTypeScale=-2` を持っている必要があります。

関連リソース

データベース

- 「Amazon RDS for Oracle」
- Amazon RDS for PostgreSQL

SSL DB 接続

- 「SSL/TLS を使用して、 DB インスタンスへの接続を暗号化」

  - 「RDS for Oracle DB インスタンスでの SSL の使用」
  - 「SSL/TLS を使用して RDS for PostgreSQL への接続を保護する」
  - 特定の AWS リージョンの証明書バンドルをダウンロードする
    - CA-2019 ルート証明書をダウンロードする (2024 年 8 月に期限切れ)
- 「オプショングループを使用する」
  - 「Oracle DB インスタンスへのオプションの追加」
  - 「Oracle Secure Sockets Layer」
- 「パラメータグループを使用する」
- 「PostgreSQL SSL モード接続パラメーター」
- 「JDBC からの SSL の使用」
- SSL/TLS 証明書の更新
  - DB インスタンスまたはクラスターを変更して CA 証明書を更新する
  - メンテナンスを適用して CA 証明書を更新する

AWS SCT

- 「AWS スキーマ変換ツール」
- 「AWS スキーマ変換ツールユーザーガイド」
- 「AWS SCT ユーザーインターフェースの使用」
- 「AWS SCT のソースとして、Oracle Database の使用」

AWS DMS

- AWS Database Migration Service
- 「AWS Database Migration Service ユーザーガイド」
  - 「AWS DMSのソースとして Oracle データベースを使用」
  - 「PostgreSQL データベースを AWS DMS ターゲットとして使用する」
- 「AWS データベース移行サービスを組み合わせて SSL を使用する」
- 「リレーショナルデータベースを実行するアプリケーションの AWS への移行」

追加情報

Amazon RDS 認証機関証明書は 2024 年 8 月に`rds-ca-2019`期限切れになりました。証明書の検証で SSL または TLS を使用して RDS DB インスタンスまたはマルチ AZ DB クラスターに接続する場合、新しい CA 証明書の 1 つである `rds-ca-rsa2048-g1`、`rds-ca-rsa4096-g1`、または の使用を検討してください`rds-ca-ecc384-g1`。

Oracle SERIALLY_REUSABLE プラグマパッケージを PostgreSQL に移行

作成者:Vinay Paladi (AWS)

概要

このパターンは、SERIALLY_REUSABLE プラグマとして定義されている Oracle パッケージを Amazon Web Services (AWS) 上の PostgreSQL に移行するための段階的なアプローチを提供します。このアプローチでは、SERIALLY_REUSABLE プラグマの機能が維持されます。

PostgreSQL はパッケージの概念と SERIALLY_REUSABLE プラグマをサポートしていません。PostgreSQL でも同様の機能を実現するには、パッケージ用のスキーマを作成し、関連するすべてのオブジェクト (関数、プロシージャ、タイプなど) をスキーマ内にデプロイできます。SERIALLY_REUSABLE プラグマの機能を実現するために、このパターンで提供されるラッパー関数スクリプトの例では、「AWS Schema Conversion Tool (AWS SCT) 拡張パック」を使用しています。

詳細については、Oracle ドキュメントの「SERIALLY_REUSABLE Pragma」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS SCT の最新バージョンと必要なドライバー
- Amazon Aurora PostgreSQL 互換エディションデータベースまたは PostgreSQL データベース用の Amazon Relational Database Service (Amazon RDS)

製品バージョン

- Oracle データベースバージョン 10g 以降

アーキテクチャ

ソーステクノロジースタック

- Oracle Database のオンプレミス

ターゲットテクノロジースタック

- 「Aurora PostgreSQL 互換」または Amazon RDS for PostgreSQL
- AWS SCT

移行アーキテクチャ

ツール

AWS サービス

- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。
- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。
- 「Amazon Relational Database Service (Amazon RDS)」を使用して、AWS クラウドでの PostgreSQL リレーショナルデータベースをセットアップ、運用、スケーリングできます。

その他のツール

- 「pgAdmin」は PostgreSQL 用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。

エピック

AWS SCT を使用して Oracle パッケージを移行する

タスク	説明	必要なスキル
AWS SCT を設定します。	ソースデータベースへの AWS SCT 接続を設定します。詳細については、「AWS SCT のソースとしての Oracle データベースの使用」を参照してください。	DBA、開発者

タスク	説明	必要なスキル
スクリプトを変換します。	AWS SCT を使用して Oracle パッケージを変換します。そのためには、ターゲットデータベースを Aurora PostgreSQL 互換として選択します。	DBA、開発者
.sql ファイルを保存します。	.sql ファイルを保存する前に、AWS SCT の [プロジェクト設定] オプションを [ステージごとの単一ファイル] に変更します。AWS SCT は、.sql ファイルをオブジェクトタイプに基づいて複数の.sql ファイルに分割します。	DBA、開発者
コードを変更してください。	AWS SCT によって生成された `init` 関数を開き、「追加情報」セクションの例に示すように変更します。機能 `pg_serialize = 0` を実現するための変数が追加されます。	DBA、開発者
変換をテストします。	`init` 関数を Aurora PostgreSQL 互換データベースにデプロイし、結果をテストします。	DBA、開発者

関連リソース

- 「AWS スキーマ変換ツール」
- 「Amazon RDS」
- 「Amazon Aurora 機能」
- 「SERIALLY_REUSABLE Pragma プラグマ」

追加情報

```
Source Oracle Code:

CREATE OR REPLACE PACKAGE test_pkg_var
IS
PRAGMA SERIALLY_REUSABLE;
PROCEDURE function_1
 (test_id number);
PROCEDURE function_2
 (test_id number
 );
END;

CREATE OR REPLACE PACKAGE BODY test_pkg_var
IS
PRAGMA SERIALLY_REUSABLE;
v_char VARCHAR2(20) := 'shared.airline';
v_num number := 123;

PROCEDURE function_1(test_id number)
IS
begin
dbms_output.put_line( 'v_char-'|| v_char);
dbms_output.put_line( 'v_num-'||v_num);
v_char:='test1';
function_2(0);
END;

PROCEDURE function_2(test_id number)
is
begin
dbms_output.put_line( 'v_char-'|| v_char);
dbms_output.put_line( 'v_num-'||v_num);
END;
END test_pkg_var;

Calling the above functions

set serveroutput on


EXEC test_pkg_var.function_1(1);
```

```
EXEC test_pkg_var.function_2(1);


Target Postgresql Code:


CREATE SCHEMA test_pkg_var;

CREATE OR REPLACE FUNCTION test_pkg_var.init(pg_serialize IN INTEGER DEFAULT 0)

RETURNS void
AS
$BODY$

DECLARE

BEGIN

if aws_oracle_ext.is_package_initialized( 'test_pkg_var' ) AND pg_serialize = 0

then

return;

end if;

PERFORM aws_oracle_ext.set_package_initialized( 'test_pkg_var' );

PERFORM aws_oracle_ext.set_package_variable( 'test_pkg_var', 'v_char',
 'shared.airline.basecurrency'::CHARACTER

VARYING(100));

PERFORM aws_oracle_ext.set_package_variable('test_pkg_var', 'v_num', 123::integer);

END;

$BODY$

LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION test_pkg_var.function_1(pg_serialize int default 1)
```

```
RETURNS void
AS

$BODY$
DECLARE

BEGIN

PERFORM test_pkg_var.init(pg_serialize);

raise notice 'v_char%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_char');

raise notice 'v_num%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_num');

PERFORM aws_oracle_ext.set_package_variable( 'test_pkg_var', 'v_char',
 'test1'::varchar);

PERFORM test_pkg_var.function_2(0);
END;

$BODY$
LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION test_pkg_var.function_2(IN pg_serialize integer default 1)

RETURNS void

AS

$BODY$

DECLARE

BEGIN

PERFORM test_pkg_var.init(pg_serialize);

raise notice 'v_char%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_char');

raise notice 'v_num%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_num');

END;
```

```
$BODY$
LANGUAGE plpgsql;


Calling the above functions

select test_pkg_var.function_1()

 select test_pkg_var.function_2()
```

Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行

作成者: anuradha chintha (AWS) と Rakesh Raghav (AWS)

概要

外部テーブルにより、Oracle がデータベースの外部にあるフラットファイルに保存されているデータをクエリできます。ORACLE_LOADER ドライバーを使用して、SQL*Loader ユーティリティでロードできるあらゆる形式で保存されているデータにアクセスできます。外部テーブルではデータ操作言語 (DML) を使用できませんが、クエリ、結合、ソート操作には外部テーブルを使用できます。

Amazon Aurora PostgreSQL 互換エディションには、Oracle の外部テーブルと類似する機能はありません。代わりに、モダナイゼーションを使用して、機能要件を満たし、かつ質素なスケーラブルなソリューションを開発する必要があります。

このパターンでは、aws_s3 拡張を使用して、さまざまなタイプの Oracle 外部テーブルをAmazon Web Services (AWS) クラウド上の Aurora PostgreSQL 互換エディションに移行する手順を示しています。

実稼働環境に実装する前に、このソリューションを徹底的にテストすることを推奨します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI)
- 使用可能な Aurora PostgreSQL 互換データベースインスタンス。
- 外部テーブルのあるオンプレミスの Oracle データベース
- PG. クライアント API
- データファイル

機能制限

- このパターンには Oracle 外部テーブルの置き換えの機能がありません。ただし、手順とサンプルコードをさらに拡張して、データベースのモダナイゼーション目標を達成することができます。
- ファイルには、aws_s3 エクスポート関数とインポート関数で区切り文字として渡される文字を含んではなりません。

製品バージョン

- Amazon S3 から PostgreSQL の RDS にインポートするには、データベースで PostgreSQL バージョン 10.7 以降を実行する必要があります。

アーキテクチャ

ソーステクノロジースタック

- Oracle

ソースアーキテクチャ

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL- 互換
- Amazon CloudWatch
- AWS Lambda
- AWS Secrets Manager
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)

ターゲットアーキテクチャ

以下の図表は、このソリューションの概要を示しています。

1. ファイルが S3 バケットにアップロードされます。
2. Lambda 関数が初期化されます。
3. Lambda 関数は DB 関数呼び出しを開始します。
4. Secrets Manager は、データベースにアクセスするための認証情報を提供します。
5. DB 関数によって、SNS アラームが作成されます。

自動化とスケール

外部テーブルへの追加や変更は、メタデータのメンテナンスで処理できます。

ツール

- 「Amazon Aurora PostgreSQL-互換」 — Amazon Aurora PostgreSQL 互換エディションは、フルマネージド型で PostgreSQL 互換で、ACID 準拠のリレーショナルデータベースエンジンです。ハイエンドの商用データベースのスピードと信頼性を、オープンソースデータベースの高いコスト効率を経験できます。
- 「AWS CLI」 — AWS コマンドラインインターフェイス (AWS CLI)は、AWS のサービスを管理するための統合ツールです。ダウンロードおよび設定用の 1 つのツールのみを使用して、コマンドラインから複数の AWS サービスを制御し、スクリプトを使用してこれらを自動化することができます。
- 「Amazon CloudWatch」 — Amazon CloudWatch は Amazon S3 のリソースと利用状況を監視します。
- 「AWS Lambda」 — AWS Lambdaは、サーバーのプロビジョニングや管理、ワークロードに対応したクラスタースケーリングロジックの作成、イベント統合の維持、あるいはランタイムの管理などを行うことなくコードを実行でき、サーバーレスコンピューティングサービスです。このパターンでは、ファイルが Amazon S3 にアップロードされるたびに、Lambda がデータベース関数を実行します。
- 「AWS Secrets Manager」 — AWS Secrets Manager は、認証情報を保存および取得するためのサービスです。Secrets Manager を使用して、コードにハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得できます。
- 「Amazon S3」 — Amazon Simple Storage Service (Amazon S3) は、Aurora PostgreSQL 互換クラスターとの間で消費および送信するファイルを受信および保存するためのストレージレイヤーを提供します。
- 「aws_s3」 — aws_s3 の拡張は Amazon S3 と Aurora PostgreSQL 互換を統合します。
- 「Amazon SNS」 — Amazon Simple Notification Service (Amazon SNS)は、パブリッシャーやクライアントの間のメッセージ配信や送信を調整および管理します。このパターンでは、Amazon SNS を使用して通知を送信します。

Code

ファイルを S3 バケットに配置するたびに、DB 関数を作成して処理アプリケーションまたは Lambda 関数から呼び出す必要があります。詳細については、コード (添付) を参照してください。

エピック

外部ファイルの作成

タスク	説明	必要なスキル
外部ファイルをソースデータベースに追加します。	外部ファイルを作成し、`oracle` ディレクトリに移動します。	DBA

ターゲットの設定 (Aurora PostgreSQL 互換の) 設定

タスク	説明	必要なスキル
Aurora PostgreSQL データベースを作成します。	Amazon Aurora PostgreSQL 互換クラスターに DB インスタンスを作成します。	DBA
スキーマ、aws_s3 エクステンション、テーブルを作成します。	「追加情報」セクションの `ext_tbl_scripts` にあるコードを使用します。テーブルには、実際のテーブル、ステージングテーブル、エラーとログテーブル、およびメタテーブルが含まれます。	DBA、開発者
DB関数を作成します。	DB 関数を作成するには、追加情報セクションの `load_external_table_latest` 関数の下のコードを使用します。	DBA、開発者

Lambda 関数の作成と設定

タスク	説明	必要なスキル
ロールを作成します。	Amazon S3 と Amazon Relational Database Service	DBA

タスク	説明	必要なスキル
	(Amazon RDS) にアクセスする権限を持つロールを作成します。このロールは、パターンを実行するための Lambda に割り当てられます。	
Lambda 関数を作成します。	Amazon S3 (例: `file_key = info.get('object', {}).get('key')`)からファイル名を読み取り、そのファイル名を入力パラメータとして DB 関数 (例、`curs.callproc("load_external_tables", [file_key])`) を呼び出す Lambda 関数を作成します。 関数呼び出しの結果に応じて、SNS 通知が開始されます (例、`client.publish(TopicArn='arn:',Message='fileloadsuccess',Subject='fileloadsuccess')`)。 ビジネスニーズに基づいて、必要に応じて追加のコードを使用して Lambda 関数を作成できます。詳細については、Lambda の「ドキュメント」を参照してください。	DBA

タスク	説明	必要なスキル
S3 バケットイベントトリガーを設定します。	S3 バケットのすべてのオブジェクト作成イベントで Lambda 関数を呼び出すメカニズムを設定します。	DBA
シークレットを作成します。	Secrets Manager を使用して、データベース認証情報のシークレット名を作成します。Lambda 関数にシークレットを渡します。	DBA
Lambda サポートファイルをアップロードします。	Lambda サポートパッケージを Aurora PostgreSQL 互換に接続するための添付の Python スクリプトを含む.zip ファイルをアップロードします。Python コードは、データベースで作成した関数を呼び出します。	DBA
SNS トピックを作成します。	SNS トピックを作成して、データロードの成功または失敗のメールを送信します。	DBA

Amazon S3 との統合を追加する

タスク	説明	必要なスキル
S3 バケットを作成する。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを作成します。S3 バケット名はグローバルに一意であり、名前	DBA

タスク	説明	必要なスキル
	空間はすべての AWS アカウントによって共有されます。	
IAM ポリシーを作成します。	AWS 識別とアクセス管理 (IAM) ポリシーを作成するには、追加情報セクションの `s3bucketpolicy_for_import` の配下のコードを使用します。	DBA
ロールを作成します。	Aurora PostgreSQL 互換用に 2 つのロールを作成します。1 つはインポート用で、もう 1 つはエクスポート用です。対応するポリシーをロールに割り当てます。	DBA
ロールを Aurora PostgreSQL 互換クラスターにアタッチします。	ロールの管理で、Aurora PostgreSQL クラスターにインポートロールとエクスポートロールをアタッチします。	DBA
Aurora PostgreSQL 互換のサポートオブジェクトを作成します。	テーブルスクリプトについては、追加情報セクション `ext_tbl_scripts` の配下にあるコードを使用します。 カスタム関数については、追加情報セクションの `load_external_Table_latest` の配下にあるコードを使用します。	DBA

テストファイルを処理

タスク	説明	必要なスキル
S3 バケットにファイルをアップロードします。	テストファイルを S3 バケットにアップロードするには、コンソールまたは AWS CLIにある以下のコマンドを使用します。 `aws s3 cp /Users/Desktop/ukpost/exttbl/"testing files"/aps s3://s3importtest/inputext/aps` ファイルがアップロードされるとすぐに、バケットイベントによって Lambda 関数が開始されますが、それによりAurora PostgreSQL 互換関数が実行されます。	DBA
データ、ログ、エラーファイルを確認してください。	Aurora PostgreSQL 互換関数はファイルをメインテーブルに読み込み、S3 バケットに `.log` と `.bad` のファイルを作成します。	DBA
ソリューションを監視します。	Amazon CloudWatch コンソールで、Lambda 関数をモニタリングします。	DBA

関連リソース

- 「Amazon S3 統合」
- 「Amazon S3」

- 「Amazon Aurora PostgreSQL-互換エディションと連携」
- 「AWS Lambda」
- 「Amazon CloudWatch」
- 「AWS Secrets Manager」
- 「Amazon SNS 通知のセットアップ」

追加情報

ext_table_scripts

```
CREATE EXTENSION aws_s3 CASCADE;
CREATE TABLE IF NOT EXISTS meta_EXTERNAL_TABLE
(
    table_name_stg character varying(100) ,
    table_name character varying(100)  ,
    col_list character varying(1000)  ,
    data_type character varying(100)  ,
    col_order numeric,
    start_pos numeric,
    end_pos numeric,
    no_position character varying(100)  ,
    date_mask character varying(100)  ,
    delimeter character(1)  ,
    directory character varying(100)  ,
    file_name character varying(100)  ,
    header_exist character varying(5)
);
CREATE TABLE IF NOT EXISTS ext_tbl_stg
(
    col1 text
);
CREATE TABLE IF NOT EXISTS error_table
(
    error_details text,
    file_name character varying(100),
    processed_time timestamp without time zone
);
CREATE TABLE IF NOT EXISTS log_table
(
    file_name character varying(50) COLLATE pg_catalog."default",
    processed_date timestamp without time zone,
```

```
    tot_rec_count numeric,
    proc_rec_count numeric,
    error_rec_count numeric
);
sample insert scripts of meta data:
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'source_filename', 'character varying', 2, 8, 27, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'record_type_identifier', 'character varying', 3, 28, 30, NULL, NULL, NULL,
 'databasedev', 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'fad_code', 'numeric', 4, 31, 36, NULL, NULL, NULL, 'databasedev', 'externalinterface/
loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'session_sequence_number', 'numeric', 5, 37, 42, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'transaction_sequence_number', 'numeric', 6, 43, 48, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
```

s3bucketpolicy_for import

```
---Import role policy
--Create an IAM policy to allow, Get,  and list actions on S3 bucket
 {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "s3import",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
```

```
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::s3importtest",
                "arn:aws:s3:::s3importtest/*"
            ]
        }
    ]
}
--Export Role policy
--Create an IAM policy to allow, put,  and list actions on S3 bucket
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "s3export",
            "Action": [
                "S3:PutObject",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::s3importtest/*"
            ]
        }
    ]
}
```

Sample DB function load_external_tables_latest

```
CREATE OR REPLACE FUNCTION public.load_external_tables(pi_filename text)
 RETURNS character varying
 LANGUAGE plpgsql
AS $function$
/* Loading data from S3 bucket into a APG table */
DECLARE
 v_final_sql TEXT;
 pi_ext_table TEXT;
 r refCURSOR;
 v_sqlerrm text;
 v_chunk numeric;
 i integer;
 v_col_list TEXT;
```

```
v_postion_list CHARACTER VARYING(1000);
v_len  integer;
v_delim varchar;
v_file_name CHARACTER VARYING(1000);
v_directory CHARACTER VARYING(1000);
v_table_name_stg CHARACTER VARYING(1000);
v_sql_col TEXT;
v_sql TEXT;
v_sql1 TEXT;
v_sql2 TEXT;
v_sql3 TEXT;
v_cnt integer;
v_sql_dynamic TEXT;
v_sql_ins TEXT;
proc_rec_COUNT integer;
error_rec_COUNT integer;
tot_rec_COUNT integer;
v_rec_val integer;
rec record;
v_col_cnt integer;
kv record;
v_val text;
v_header text;
j integer;
ERCODE VARCHAR(5);
v_region text;
cr CURSOR FOR
SELECT distinct DELIMETER,
  FILE_NAME,
  DIRECTORY
FROM  meta_EXTERNAL_TABLE
WHERE table_name = pi_ext_table
  AND DELIMETER IS NOT NULL;


cr1 CURSOR FOR
  SELECT   col_list,
  data_type,
  start_pos,
  END_pos,
  concat_ws('',' ',TABLE_NAME_STG) as TABLE_NAME_STG,
  no_position,date_mask
FROM  meta_EXTERNAL_TABLE
WHERE table_name = pi_ext_table
```

```
 order by col_order asc;
cr2 cursor FOR
SELECT  distinct table_name,table_name_stg
   FROM  meta_EXTERNAL_TABLE
   WHERE upper(file_name) = upper(pi_filename);


BEGIN
 -- PERFORM utl_file_utility.init();
   v_region := 'us-east-1';
   /* find tab details from file name */


   --DELETE FROM  ERROR_TABLE WHERE file_name= pi_filename;
  -- DELETE FROM  log_table WHERE file_name= pi_filename;


 BEGIN


   SELECT distinct table_name,table_name_stg INTO strict pi_ext_table,v_table_name_stg
   FROM  meta_EXTERNAL_TABLE
   WHERE upper(file_name) = upper(pi_filename);
 EXCEPTION
   WHEN NO_DATA_FOUND THEN
    raise notice 'error 1,%',sqlerrm;
    pi_ext_table := null;
    v_table_name_stg := null;
      RAISE USING errcode = 'NTFIP' ;
    when others then
        raise notice 'error others,%',sqlerrm;
 END;
 j :=1 ;

for rec in  cr2
 LOOP




  pi_ext_table     := rec.table_name;
  v_table_name_stg := rec.table_name_stg;
  v_col_list := null;
```

```
IF pi_ext_table IS NOT NULL
 THEN
   --EXECUTE concat_ws('','truncate table  ' ,pi_ext_table) ;
  EXECUTE concat_ws('','truncate table  ' ,v_table_name_stg) ;




      SELECT distinct DELIMETER INTO STRICT v_delim
      FROM  meta_EXTERNAL_TABLE
      WHERE table_name = pi_ext_table;


      IF v_delim IS NOT NULL THEN
    SELECT distinct DELIMETER,
      FILE_NAME,
      DIRECTORY ,
      concat_ws('',' ',table_name_stg),
      case  header_exist when 'YES' then 'CSV HEADER' else 'CSV' end as header_exist
    INTO STRICT v_delim,v_file_name,v_directory,v_table_name_stg,v_header
    FROM  meta_EXTERNAL_TABLE
    WHERE table_name = pi_ext_table
      AND DELIMETER IS NOT NULL;


    IF    upper(v_delim) = 'CSV'
    THEN
      v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3 ( ''',
      v_table_name_stg,''','''',
      ''DELIMITER '''','''' CSV HEADER QUOTE '''''''''''', aws_commons.create_s3_uri
( ''',
      v_directory,''','''',v_file_name,''', ''',v_region,'''))');
      ELSE
      v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3(''',
          v_table_name_stg, ''','''', ''DELIMITER AS ''''^''''',''',','
         aws_commons.create_s3_uri
          ( ''',v_directory, ''','''',
          v_file_name, ''',',
           '''',v_region,''')
         )');
         raise notice 'v_sql , %',v_sql;
      begin
       EXECUTE  v_sql;
```

```
      EXCEPTION
        WHEN OTHERS THEN
          raise notice 'error 1';
        RAISE USING errcode = 'S3IMP' ;
      END;


      select count(col_list) INTO v_col_cnt
      from  meta_EXTERNAL_TABLE where table_name = pi_ext_table;







       -- raise notice 'v_sql 2, %',concat_ws('','update ',v_table_name_stg, ' set
col1 = col1||''',v_delim,''''');


      execute concat_ws('','update ',v_table_name_stg, ' set col1 =
col1||''',v_delim,''''');




      i :=1;
      FOR rec in cr1
      loop
      v_sql1 := concat_ws('',v_sql1,'split_part(col1,''',v_delim,''',', i,')',' as
',rec.col_list,',');
      v_sql2 := concat_ws('',v_sql2,rec.col_list,',');
  --    v_sql3 := concat_ws('',v_sql3,'rec.',rec.col_list,'::',rec.data_type,',');


      case
        WHEN upper(rec.data_type) = 'NUMERIC'
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                coalesce((trim(split_part(col1,''',v_delim,''',',
i,')))::NUMERIC,0)::',rec.data_type,' END as ',rec.col_list,',') ;
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDD'
```

```
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''99990101''),''YYYYMMDD''):',rec.data_type,' END as ',rec.col_list,',');
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
 'MM/DD/YYYY hh24:mi:ss'
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''01/01/9999 0024:00:00''),''MM/DD/YYYY hh24:mi:ss''):',rec.data_type,' END as
',rec.col_list,',');
         ELSE
       v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                 coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''''):',rec.data_type,' END as ',rec.col_list,',') ;
      END case;


      i :=i+1;
      end loop;


        -- raise notice 'v_sql 3, %',v_sql3;


      SELECT trim(trailing ' ' FROM v_sql1) INTO v_sql1;
      SELECT trim(trailing ',' FROM v_sql1) INTO v_sql1;


      SELECT trim(trailing ' ' FROM v_sql2) INTO v_sql2;
      SELECT trim(trailing ',' FROM v_sql2) INTO v_sql2;


      SELECT trim(trailing ' ' FROM v_sql3) INTO v_sql3;
      SELECT trim(trailing ',' FROM v_sql3) INTO v_sql3;
```

```
       END IF;
      raise notice 'v_delim , %',v_delim;


     EXECUTE concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg)  INTO v_cnt;


    raise notice 'stg cnt , %',v_cnt;


    /* if upper(v_delim) = 'CSV' then
       v_sql_ins := concat_ws('', ' SELECT * from ' ,v_table_name_stg );
     else
      -- v_sql_ins := concat_ws('',' SELECT ',v_sql1,'  from (select col1 from
 ' ,v_table_name_stg , ')sub ');
       v_sql_ins := concat_ws('',' SELECT ',v_sql3,'  from (select col1 from
 ' ,v_table_name_stg , ')sub ');
       END IF;*/


v_chunk := v_cnt/100;




for i in 1..101
loop
     BEGIN
    -- raise notice 'v_sql , %',v_sql;
       -- raise notice 'Chunk number , %',i;
       v_sql_ins := concat_ws('',' SELECT ',v_sql3,'  from (select col1 from
 ' ,v_table_name_stg , ' offset ',v_chunk*(i-1), ' limit ',v_chunk,') sub ');


     v_sql := concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins);
     -- raise notice 'select statement , %',v_sql_ins;
          -- v_sql := null;
     -- EXECUTE concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins, 'offset
 ',v_chunk*(i-1), ' limit ',v_chunk );
     --v_sql := concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins );


     -- raise notice 'insert statement , %',v_sql;
```

```
   raise NOTICE 'CHUNK START %',v_chunk*(i-1);
  raise NOTICE 'CHUNK END %',v_chunk;


    EXECUTE v_sql;


 EXCEPTION
      WHEN OTHERS THEN
      -- v_sql_ins := concat_ws('',' SELECT ',v_sql1, '  from (select col1 from
' ,v_table_name_stg , ' )sub ');
        -- raise notice 'Chunk number for cursor , %',i;


   raise NOTICE 'Cursor - CHUNK START %',v_chunk*(i-1);
  raise NOTICE 'Cursor -  CHUNK END %',v_chunk;
        v_sql_ins := concat_ws('',' SELECT ',v_sql3, '  from (select col1 from
' ,v_table_name_stg , ' )sub ');


        v_final_sql := REPLACE (v_sql_ins, ''''::text, ''''''::text);
       -- raise notice 'v_final_sql %',v_final_sql;
        v_sql :=concat_ws('','do $a$ declare  r refcursor;v_sql text; i
numeric;v_conname text;  v_typ  ',pi_ext_table,'[]; v_rec  ','record',';
          begin







          open r for execute ''select col1 from ',v_table_name_stg ,'  offset
',v_chunk*(i-1), ' limit ',v_chunk,''';
          loop
          begin
          fetch r into v_rec;
          EXIT WHEN NOT FOUND;
```

```
          v_sql := concat_ws('''','insert into  ',pi_ext_table,' SELECT ',REPLACE
 (v_sql3, ''''::text, ''''''::text) , '  from ( select ''''''',v_rec.col1,''''''' as
 col1) v'');
           execute v_sql;


          exception
           when others then
         v_sql := ''INSERT INTO  ERROR_TABLE VALUES (concat_ws(''''''''','''''Error
 Name: '''',$$''||SQLERRM||''$$,'''''Error State: '''','''''''||
SQLSTATE||''''''','''''record : '''',$$''||v_rec.col1||''$$),''''''||
pi_filename||''''''',now())'';


             execute v_sql;
            continue;
          end ;
          end loop;
          close r;
          exception
          when others then
        raise;
          end ; $a$');
     -- raise notice ' inside excp v_sql %',v_sql;
         execute v_sql;
     --  raise notice 'v_sql %',v_sql;
       END;
  END LOOP;
     ELSE


     SELECT distinct DELIMETER,FILE_NAME,DIRECTORY ,concat_ws('',' ',table_name_stg),
       case  header_exist when 'YES' then 'CSV HEADER' else 'CSV' end as header_exist
       INTO STRICT v_delim,v_file_name,v_directory,v_table_name_stg,v_header
     FROM  meta_EXTERNAL_TABLE
     WHERE table_name = pi_ext_table                  ;
     v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3(''',
       v_table_name_stg, ''','''', ''DELIMITER AS ''''#'''' ',v_header,' '','','
      aws_commons.create_s3_uri
       ( ''',v_directory, ''','''',
       v_file_name, ''',',
        '''',v_region,''')
      )');
         EXECUTE  v_sql;
```

```
    FOR rec in cr1
    LOOP


     IF rec.start_pos IS NULL AND rec.END_pos IS NULL AND rec.no_position = 'recnum'
     THEN
       v_rec_val := 1;
     ELSE


      case
        WHEN upper(rec.data_type) = 'NUMERIC'
        THEN v_sql1 := concat_ws('',' case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1)))::NUMERIC,0)::',rec.data_type,' END as
',rec.col_list,',') ;
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDD'
        THEN v_sql1 := concat_ws('','case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),''99990101''),''YYYYMMDD''):',rec.data_type,'
END as ',rec.col_list,',');
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDDHH24MISS'
        THEN v_sql1 := concat_ws('','case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),''9999010100240000''),''YYYYMMDDHH24MISS''):',rec.data_
END as ',rec.col_list,',');
         ELSE
       v_sql1 := concat_ws('',' case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
```

```
                 coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),''''):',rec.data_type,' END as
',rec.col_list,',') ;
      END case;


     END IF;
     v_col_list := concat_ws('',v_col_list ,v_sql1);
    END LOOP;




          SELECT trim(trailing ' ' FROM v_col_list) INTO v_col_list;
          SELECT trim(trailing ',' FROM v_col_list) INTO v_col_list;


          v_sql_col   :=  concat_ws('',trim(trailing ',' FROM v_col_list) , ' FROM
 ',v_table_name_stg,' WHERE col1 IS NOT NULL AND length(col1)>0 ');




          v_sql_dynamic := v_sql_col;


          EXECUTE  concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg) INTO v_cnt;




        IF v_rec_val = 1 THEN
            v_sql_ins := concat_ws('',' select row_number() over(order by ctid) as
line_number ,' ,v_sql_dynamic) ;


        ELSE
              v_sql_ins := concat_ws('',' SELECT' ,v_sql_dynamic) ;
          END IF;


    BEGIN
      EXECUTE concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins);
          EXCEPTION
```

```
              WHEN OTHERS THEN
          IF v_rec_val = 1 THEN
                  v_final_sql := ' select row_number() over(order by ctid) as
line_number ,col1 from ';
               ELSE
                v_final_sql := ' SELECT col1 from';
              END IF;
      v_sql :=concat_ws('','do $a$ declare  r refcursor;v_rec_val numeric :=
',coalesce(v_rec_val,0),';line_number numeric; col1 text; v_typ  ',pi_ext_table,'[];
v_rec  ',pi_ext_table,';
            begin
            open r for execute ''',v_final_sql, ' ',v_table_name_stg,' WHERE col1 IS
NOT NULL AND length(col1)>0 '' ;
            loop
            begin
            if   v_rec_val = 1 then
            fetch r into line_number,col1;
            else
            fetch r into col1;
            end if;


            EXIT WHEN NOT FOUND;
             if v_rec_val = 1 then
             select line_number,',trim(trailing ',' FROM v_col_list) ,' into v_rec;
             else
               select ',trim(trailing ',' FROM v_col_list) ,' into v_rec;
             end if;


            insert into  ',pi_ext_table,' select v_rec.*;
             exception
             when others then
              INSERT INTO  ERROR_TABLE VALUES (concat_ws('''',''Error Name:
'',SQLERRM,''Error State: '',SQLSTATE,''record : '',v_rec),''',pi_filename,''',now());
              continue;
             end ;
              end loop;
            close r;
             exception
             when others then
             raise;
             end ; $a$');
        execute v_sql;
```

```
    END;


        END IF;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM  ' ,pi_ext_table)   INTO proc_rec_COUNT;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM  error_table WHERE file_name
='''',pi_filename,''' and processed_time::date = clock_timestamp()::date')  INTO
error_rec_COUNT;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg)   INTO tot_rec_COUNT;


  INSERT INTO  log_table values(pi_filename,now(),tot_rec_COUNT,proc_rec_COUNT,
error_rec_COUNT);


  raise notice 'v_directory, %',v_directory;


  raise notice 'pi_filename, %',pi_filename;


  raise notice 'v_region, %',v_region;


 perform aws_s3.query_export_to_s3('SELECT
replace(trim(substring(error_details,position(''('' in
error_details)+1),'')''),'','','';''),file_name,processed_time FROM  error_table WHERE
file_name = '''||pi_filename||'''',
  aws_commons.create_s3_uri(v_directory, pi_filename||'.bad', v_region),
  options :='FORmat csv, header, delimiter $$,$$'
  );


raise notice 'v_directory, %',v_directory;
```

```
   raise notice 'pi_filename, %',pi_filename;


   raise notice 'v_region, %',v_region;


  perform aws_s3.query_export_to_s3('SELECT * FROM  log_table WHERE file_name = '''||
pi_filename||''''',
   aws_commons.create_s3_uri(v_directory, pi_filename||'.log', v_region),
   options :='FORmat csv, header, delimiter $$,$$'
   );




   END IF;
 j := j+1;
 END LOOP;


       RETURN 'OK';
EXCEPTION
    WHEN  OTHERS THEN
  raise notice 'error %',sqlerrm;
   ERCODE=SQLSTATE;
   IF ERCODE = 'NTFIP' THEN
     v_sqlerrm := concat_Ws('',sqlerrm,'No data for the filename');
   ELSIF ERCODE = 'S3IMP' THEN
    v_sqlerrm := concat_Ws('',sqlerrm,'Error While exporting the file from S3');
   ELSE
      v_sqlerrm := sqlerrm;
   END IF;


 select distinct directory into v_directory from  meta_EXTERNAL_TABLE;




 raise notice 'exc v_directory, %',v_directory;


   raise notice 'exc pi_filename, %',pi_filename;
```

```
   raise notice 'exc v_region, %',v_region;


  perform aws_s3.query_export_to_s3('SELECT * FROM  error_table WHERE file_name = '''||
pi_filename||'''',
   aws_commons.create_s3_uri(v_directory, pi_filename||'.bad', v_region),
   options :='FORmat csv, header, delimiter $$,$$'
   );
    RETURN null;
END;
$function$
```

関数ベースのインデックスを Oracle から PostgreSQL に移行する

作成者：Veeranjaneyulu Grandhi () とNavakanth Talluri ()

概要

インデックスはデータベースのパフォーマンスを向上させる一般的な方法です。インデックスを使用すると、データベースサーバーはインデックスがない場合よりもずっと速く特定の行を検索して取得できます。ただし、インデックスはデータベースシステム全体にオーバーヘッドも発生させるため、適切に使用する必要があります。関数または式に基づく関数ベースのインデックスには、複数の列や数式が含まれる場合があります。関数ベースのインデックスは、インデックス式を使用するクエリのパフォーマンスを向上させます。

PostgreSQL はもともと、安定と定義されているボラティリティを持つ関数を使った関数ベースのインデックスの作成をサポートしていません。ただし、IMMUTABLE としてのボラティリティを持つ関数を作成して、インデックスの作成に使用することはできます。

IMMUTABLE 関数はデータベースを変更することはできず、同じ引数が与えられてもずっと同じ結果を返すことが保証されています。このカテゴリにより、クエリが定数引数で関数を呼び出したときに、オプティマイザーは関数を事前評価できます。

このパターンは、to_char、to_date、to_numberなどの関数と一緒に使用すると、Oracle の関数ベースのインデックスを同等の PostgreSQL に移行するのに役立ちます。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント
- リスナーサービスがセットアップされて実行されているソース Oracle データベースインスタンス
- PostgreSQL データベースに関する知識

制約事項

- データベースのサイズ制限は 64 TB です。
- インデックス作成に使用される関数は IMMUTABLE でなければなりません。

製品バージョン

- バージョン 11g (バージョン 11.2.0.3.v1 以降) および、12.2 および 18c までのすべての Oracle データベースエディション
- PostgreSQL バージョン 9.6 以降

アーキテクチャ

ソーステクノロジースタック

- オンプレミスまたはAmazon Elastic Compute Cloud (Amazon EC2) インスタンス上の Oracle データベース、または Amazon RDS for Oracle DB インスタンス

ターゲットテクノロジースタック

- PostgreSQL エンジンのすべて

ツール

- pgAdmin 4 は Postgres 向けのオープンソース管理ツールです。pgAdmin 4 ツールは、データベースオブジェクトを作成、管理、使用するためのグラフィカルインターフェイスを提供します。
- Oracle SQL Developer は、従来のデプロイとクラウドデプロイの両方で Oracle データベースを開発および管理するための統合開発環境 (IDE) です。

エピック

規定の関数を使用して関数ベースのインデックスを作成する

タスク	説明	必要なスキル

to_char 関数を使用して列に関数ベースのインデックスを作成する。	次のコードを使用して、関数を作成します。
	`postgres=# create table funcindex( col1 timestamp without time zone);`
	`CREATE TABLE`
		DBA、アプリ開発者

タスク	説明	必要なスキル
	postgres=# insert into funcindex values (now()); INSERT 0 1 postgres=# select * from funcindex; col1 ---------- 2022-08-09 16:00:57. 77414 (1 rows) postgres=# create index funcindex_idx on funcindex(to_char(col1,'DD-MM-YYYY HH24:MI:SS')); ERROR: functions in index expression must be marked IMMUTABLE **Note** PostgreSQL では、IMMUTABLE 句なしで関数ベースのインデックスを作成することはできません。	
関数のボラティリティをチェックします。	関数のボラティリティをチェックするには、「追加情報」セクションのコードを使用してください。	DBA

ラッパー関数を使用して関数ベースのインデックスを作成する

タスク	説明	必要なスキル
ラッパー関数を作成する。	ラッパー関数を作成するには、「追加情報」セクションのコードを使用してください。	PostgreSQL 開発者
ラッパー関数を使用してインデックスを作成します。	「追加情報」セクションのコードを使用して、アプリケーションと同じスキーマにある `IMMUTABLE` キーワードを含むユーザー定義関数を作成し、索引作成スクリプトで参照します。 ユーザー定義の関数が (前の例の) 共通のスキーマで作成されている場合は、`search_path` を次のように更新します。 `ALTER ROLE <ROLENAME> set search_path=$user, COMMON;`	DBA、PostgreSQL 開発者

インデックス作成の検証

タスク	説明	必要なスキル
インデックス作成についての検証を行います。	クエリのアクセスパターンに基づき、インデックスの作成が必要であることを検証します。	DBA

タスク	説明	必要なスキル
インデックスが使用可能であることを検証します。	PostgreSQL オプティマイザーが関数ベースのインデックスを取得しているか確認するには、「explain」または「explain analyze」で SQL ステートメントを実行します。「追加情報」セクションのコードを使用してください。可能であれば、テーブル統計情報も収集します。 **Note** 説明プランに気付いた場合、PostgreSQL オプティマイザは述語条件のために関数ベースのインデックスを選択しました。	DBA

関連リソース

- 関数ベースのインデックス (Oracle ドキュメント)
- 式のインデックス (PostgreSQL ドキュメント)
- PostgreSQL のボラティリティ (PostgreSQL ドキュメント)
- PostgreSQL search_path (PostgreSQL ドキュメント)
- Oracle Database 19c から Amazon Aurora PostgreSQL への移行プレイブック

追加情報

ラッパー関数を作成する

```
CREATE OR REPLACE FUNCTION myschema.to_char(var1 timestamp without time zone, var2
 varchar) RETURNS varchar AS $BODY$ select to_char(var1, 'YYYYMMDD'); $BODY$ LANGUAGE
 sql IMMUTABLE;
```

ラッパー関数を使用してインデックスを作成する

```
postgres=# create function common.to_char(var1 timestamp without time zone, var2
 varchar) RETURNS varchar AS $BODY$ select to_char(var1, 'YYYYMMDD'); $BODY$ LANGUAGE
 sql IMMUTABLE;
CREATE FUNCTION
postgres=# create index funcindex_idx on funcindex(common.to_char(col1,'DD-MM-YYYY
 HH24:MI:SS'));
CREATE INDEX
```

関数のボラティリティをチェックする

```
SELECT DISTINCT p.proname as "Name",p.provolatile as "volatility" FROM
 pg_catalog.pg_proc p
 LEFT JOIN pg_catalog.pg_namespace n ON n.oid = p.pronamespace
 LEFT JOIN pg_catalog.pg_language l ON l.oid = p.prolang
 WHERE n.nspname OPERATOR(pg_catalog.~) '^(pg_catalog)$' COLLATE pg_catalog.default AND
 p.proname='to_char'GROUP BY p.proname,p.provolatile
ORDER BY 1;
```

インデックスが使用可能であることを検証する

```
explain analyze <SQL>


postgres=# explain select col1 from funcindex where common.to_char(col1,'DD-MM-YYYY
 HH24:MI:SS') = '09-08-2022 16:00:57';
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------
 Index Scan using funcindex_idx on funcindex  (cost=0.42..8.44 rows=1 width=8)
   Index Cond: ((common.to_char(col1, 'DD-MM-YYYY HH24:MI:SS'::character
 varying))::text = '09-08-2022 16:00:57'::text)
(2 rows)
```

エクステンションを使用して Oracle のネイティブ関数を PostgreSQL に移行

作成者: Pinesh Singal (AWS)

概要

この移行パターンでは、aws_oracle_ext および orafce の拡張を PostgreSQL (psql) ネイティブの組み込みコードに変更することにより、Oracle データベースインスタンスの Amazon Relational Database Service (Amazon RDS)をAmazon RDS for PostgreSQL、または Amazon Aurora PostgreSQL 互換バージョンに移行するためのステップバイステップのガイダンスを提供します。これにより、処理時間を節約します。

このパターンでは、トランザクション数が多いマルチテラバイトの Oracle ソースデータベースを、ダウンタイムなしでオフラインで手動で移行する方法を示しています。

移行プロセスでは、aws_oracle_ext と orafce 拡張付けの AWS Schema Conversion Tool (AWS SCT)を使用して、 Amazon RDS for Oracle データベーススキーマを Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換のデータベーススキーマに変換します。次に、コードを PostgreSQL でサポートされるネイティブ psql ビルトインコードに手動で変更します。理由は、拡張呼び出しが PostgreSQL データベースサーバーのコード処理に影響し、すべての拡張コードが PostgreSQL コードに完全に準拠する、または互換性があるわけではないからです。

このパターンは、主に AWS SCT と拡張 aws_oracle_ext と orafce を使用して、 SQL コードを手動で移行することにフォーカスしています。すでに使用されている拡張をネイティブの PostgreSQL (psql) ビルトインに変換します。次に、拡張の参照をすべて削除し、それに応じてコードを変換します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オペレーティングシステム (Windows または Mac) または Amazon EC2 インスタンス (稼働中)
- Orafce

機能制限

aws_oracle_ext または orafceの拡張を使用するすべての Oracle 関数が、ネイティブ PostgreSQL 関数に変換できるわけではありません。PostgreSQL ライブラリーでコンパイルするには、手動でやり直す必要があるかもしれません。

AWS SCT 拡張を使用する場合の欠点は、実行しその結果を得られるまでが遅いことです。そのコストは、添付文書のパフォーマンス比較チェックセクションで説明されているように、3つのコードすべて (aws_oracle_ext、orafce、psql およびデフォルト) 間で、Oracle SYSDATE 関数の単純な「PostgreSQL EXPLAIN 計画」 (ステートメントの実行計画) からPostgreSQL NOW( ) 関数に移行することを理解できます。

製品バージョン

- ソース: Amazon RDS for Oracle データベース 10.2 以降 (10.x 用)、11g (11.2.0.3.v1 以降) および 12.2、18c、19c (およびそれ以降) まで (エンタープライズエディション、スタンダードエディション、スタンダードエディション 1 、およびスタンダードエディション 2 用)
- ターゲット : Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換データベース 9.4 以降 (9.x用)、10.x、11.x、12.x、13.x、14.x (およびそれ以降)
- AWS SCT: 最新バージョン (このパターンは1.0.632でテスト済です)
- Orafce: 最新バージョン (このパターンは 3.9.0 でテスト済です)

アーキテクチャ

ソーステクノロジースタック

- バージョン 12.1.0.2.v18 の Amazon RDS for Oracle データベースインスタンス

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換データベースインスタンス (バージョン 11.5)

データベース移行アーキテクチャ

次の図表では、ソース Oracle とターゲット PostgreSQL のデータベース間のデータベース移行アーキテクチャを示しています。アーキテクチャには、AWS クラウド、仮想プライベートクラウド (VPC)、アベイラビリティーゾーン、プライベートサブネット、Amazon RDS for Oracle データベース、AWS SCT、Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換データベース、Oracle (aws_oracle_ext and orafce) の拡張、および構造化言語 (SQL) ファイルが含まれます。

1. Amazon RDS for Oracle DB インスタンス (ソース DB) を起動します。
2. aws_oracle_ext と orafce の拡張パックAWS SCT を使用して、ソースコードを Oracle から PostreSQL に変換します。
3. この変換により、PostgreSQL 適用の移行済みの .sql ファイルが生成されます。
4. 変換されない Oracle 拡張コードを手動で PostgreSQL (psql) コードに変換します。
5. 手動で変換すると、PostgreSQL 適用の変換済みの .sql ファイルが生成されます。
6. これらの .sql ファイルを、Amazon RDS for PostgreSQL DB インスタンス (ターゲット DB) で実行します。

ツール

ツール

AWS サービス

- 「AWS SCT」 - AWS Schema Conversion Tool (AWS SCT) は、既存のデータベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換します。リレーショナルオンライントランザクショナル処理 (OLTP)スキーマ、またはデータウェアハウススキーマンを変換できます。変換されたスキーマは、Amazon RDS for MySQL DB インスタンス、Amazon Aurora DB クラスター、Amazon RDS for PostgreSQL DB インスタンス、または Amazon Redshift クラスター に適しています。また変換されたスキーマは、Amazon EC2 インスタンスでデータベースと共に使用することも、または Amazon S3 バケットにデータとして保存することもできます。

  AWS SCT には、ソースデータベースのデータベーススキーマをターゲット Amazon RDS インスタンスと互換性のある形式に自動変換するための、プロジェクトベースのユーザーインターフェイスが用意されています。

  AWS SCT を使用して、Oracle ソースデータベースから前述のターゲットのいずれかに移行します。AWS SCT を使用して、スキーマ、ビュー、ストアドプロシージャ、関数などのソースデータベースオブジェクト定義をエクスポートできます。

  AWS SCTを使用して、Oracle からのデータをAmazon RDS for PostgreSQL または Amazon Aurora PostgreSQL 互換エディションに変換できます。

  このパターンでは、AWS SCT を使用して aws_oracle_ext と orafce 拡張とを使用して Oracle コードを PostgreSQL に変AWS換および移行し、拡張コードを psql デフォルトまたはネイティブの組み込みコードに手動で移行します。

- AWS SCT 拡張パックは、オブジェクトをターゲットデータベースに変換する際に必要な、ソースデータベースの関数をエミュレートするアドオンモジュールです。AWS SCT 拡張パックをインストールできるようになるには、データベーススキーマを変換する必要があります。

  データベースまたはデータウェアハウスのスキーマを変換すると、AWS SCT によってターゲットデータベースに別のスキーマが追加されます。この別のスキーマは、ソースデータベースの SQL システム関数を実装します。これらの関数により、変換したスキーマがターゲットデータベースに書き込まれます。この別のスキーマは、拡張パックスキーマと呼ばれます。

  OLTP データベースの拡張パックスキーマは、ソースデータベースに従って名前がつけられます。Oracle データベースの場合、拡張パックスキーマは `AWS_ORACLE_EXT` です。

その他のツール

- 「Oracle」 — Oracle は Oracle 互換の関数、データ型、およびパッケージを実装するモジュールです。Berkeley Source Distribution (BSD) ライセンスを持つオープンソースのツールで、誰でも使用できます。`orafce` モジュールは、多くの Oracle 関数が PostgreSQL に実装されているため、Oracle から PostgreSQL への移行に役立ちます。

コード

AWS SCT 拡張コードの使用を避けるために、一般的に使用され、Oracle から PostgreSQL に移行されたすべてのコードのリストについては、添付のドキュメントを参照してください。

エピック

Amazon RDS for Oracle ソースデータベースを設定

タスク	説明	必要なスキル
Oracle データベースインスタンスを作成します。	Amazon RDS コンソールから Amazon RDS for Oracle または Aurora PostgreSQL 互換のデータベースインスタンスを作成します。	AWS 全般、DBA

タスク	説明	必要なスキル
セキュリティグループを設定します。	インバウンドとアウトバウンドのセキュリティグループを設定します。	AWS 全般
データベースを作成します。	必要なユーザーとスキーマを使用して、Oracle データベースを作成します。	AWS 全般、DBA
オブジェクトを作成します。	オブジェクトを作成し、スキーマにデータを挿入します。	DBA

Amazon RDS for PostgreSQL ターゲットデータベースを設定

タスク	説明	必要なスキル
PostgreSQL データベース インスタンスを作成します。	Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL データベースインスタンスを、Amazon RDS コンソールから作成します。	AWS 全般、DBA
セキュリティグループを設定します。	インバウンドとアウトバウンドのセキュリティグループを設定します。	AWS 全般
データベースを作成します。	必要なユーザーとスキーマを使用して PostgreSQL データベースを作成します。	AWS 全般、DBA
エクステンションを確認します。	`aws_oracle_ext` と `orafce` がPostgreSQL データベースに正しくインストールされ、設定されていることを確保します。	DBA

タスク	説明	必要なスキル
PostgreSQL データベースが使用可能であることを確認します。	PostgreSQL データベースが稼働していることを確認します。	DBA

AWS SCT とエクステンションを使用して Oracle スキーマを PostgreSQL に移行します

タスク	説明	必要なスキル
AWS SCT をインストールします。	AWS SCTの最新バージョンをインストールします。	DBA
AWS SCT を設定します。	Oracle (`ojdbc8.jar`) と PostgreSQL (`postgresql-42.2.5.jar`) の Java データベース接続 (JDBC) ドライバを使用して AWS SCT を設定します。	DBA
AWS SCT 拡張パックまたはテンプレートを有効にします。	AWS SCT プロジェクト設定で、Oracle データベーススキーマの `aws_oracle_ext` 、および `orafce` 拡張によるビルドイン関数の実装を有効にします。	DBA
スキーマを変換します。	AWS SCT で、スキーマの変換を選択して、スキーマを Oracle から PostgreSQL に変換し、 .sql ファイルを生成します。	DBA

AWS SCT 拡張コードを psql コードに変換

タスク	説明	必要なスキル
コードを手動で変換します。	添付ドキュメントに詳述されているように、拡張適用のコードの各行を psql デフォルトのビルトインコードに手動で変換します。例えば、AWS_ORACLE_EXT.SYSDATE() または ORACLE.SYSDATE() を NOW() に変更します。	DBA
コードを検証	(オプション) PostgreSQL データベースでコードの各行を一時的に実行して検証します。	DBA
PostgreSQL データベースにオブジェクトを作成します。	PostgreSQL データベースにオブジェクトを作成するには、AWS SCT によって生成され、前の 2 つのステップで変更されたv.sql ファイルを実行します。	DBA

関連リソース

- データベース
  - Amazon RDS 上の Oracle
  - Amazon RDS 上の PostgreSQL
  - Amazon Aurora PostgreSQL の操作
  - 「PostgreSQL EXPLAIN 計画」
- AWS SCT
  - AWS スキーマ変換ツールの概要
  - 「AWS SCT ユーザーガイド」

  - AWS SCT ユーザーインターフェイスの使用
  - AWS SCTのソースとしての Oracle Database の使用
- AWS SCT の拡張
  - AWS SCT 拡張パックの使用
  - 「Oracleの機能 (en)」
  - 「PGX Oracle」
  - 「GitHub Oracle」

追加情報

詳細については、添付文書のコードを手動で変換するための構文および例が付いた詳細なコマンドを参照してください。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS DMS を使用して Db2 データベースを Amazon EC2 から Aurora MySQL 互換のデータベースに移行する

作成者: Pinesh Singal (AWS)

概要

「IBM Db2 for LUW データベース」を「Amazon Elastic Compute Cloud (Amazon EC2)」に移行したら、Amazon Web Services (AWS) のクラウドネイティブデータベースに移行してデータベースを再設計することを検討してください。このパターンは、「Amazon」EC2 インスタンスで実行されている IBM「Db2」for LUW データベースを、AWS 上の「Amazon Aurora MySQL 互換エディション」データベースに移行することを対象としています。

このパターンは、トランザクション数が多い数テラバイトの Db2 ソースデータベースを、ダウンタイムを最小限に抑えるオンライン移行戦略を示しています。

このパターンでは、「AWS Schema Conversion Tool (AWS SCT)」を使用して Db2 データベーススキーマを Aurora MySQL 互換スキーマに変換します。次に、パターンは「AWS Database Migration Service (AWS DMS)」を使用して、Db2 データベースから Aurora MySQL 互換データベースにデータを移行します。AWS SCT によって変換されないコードには、手動での変換が必要になります。

前提条件と制限

前提条件

- 仮想プライベートクラウド(VPC) を使用するアクティブ的な AWS アカウント
- AWS SCT
- AWS DMS

製品バージョン

- AWS SCT 最新バージョン
- Linux バージョン 11.1.4.4 以降の Db2

アーキテクチャ

ソーステクノロジースタック

- EC2 インスタンスにマウントされた DB2/Linux x86-64 ビット

ターゲットテクノロジースタック

- Amazon Aurora MySQL 互換エディションデータベースインスタンス

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、ソース Db2 とターゲット Aurora MySQL 互換データベース間のデータ移行アーキテクチャを示しています。AWS クラウドのアーキテクチャには、仮想プライベートクラウド (VPC) (仮想プライベートクラウド)、アベイラビリティーゾーン、Db2 インスタンスと AWS DMS レプリケーションインスタンスのパブリックサブネット、Aurora MySQL 互換データベースのプライベートサブネットが含まれます。

ツール

AWS サービス

- 「Amazon Aurora」はクラウド用に構築されたフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。AWS SCT は、IBM Db2 for LUW バージョン 9.1、9.5、9.7、10.1、10.5、11.1、11.5 をソースとしてサポートしています。

ベストプラクティス

ベストプラクティスについては、「AWS Database Migration Service のベストプラクティス」を参照してください。

エピック

ソース IBM Db2 データベースを設定します。

タスク	説明	必要なスキル
Amazon EC2 に IBM Db2 データベースを作成します。	EC2 インスタンスに IBM Db2 データベースを作成するには、AWS Marketplace Amazon マシンイメージ (AMI) を使用するか、EC2 インスタンスに Db2 ソフトウェアをインストールします。 IBM Db2 用の AMI (たとえば、「IBM Db2 v11.5.7 RHEL 7.9」) を選択して EC2 インスタンスを起動します。これはオンプレミスデータベースに似ています。	DBA、AWS 全般
セキュリティグループを設定する	SSH (セキュアシェル) と TCP の VPC セキュリティグループのインバウンドルールを、それぞれポート 22 と 50000 で設定します。	AWS 全般
データベースインスタンスの作成	新しいインスタンス (ユーザー) とデータベース (スキーマ) を作成するか、デフォルトの `db2inst1` インスタンスとサンプルデータベースを使用してください。 1. ターミナルを使用して EC2 インスタンスに接続し、Db2 データベースに接続します。または、Db2	DBA

タスク	説明	必要なスキル
	データベースに接続する任意の DB クライアントソフトウェアをインストールすることもできます。 2. db2inst1 ユーザーのパスワードを設定するには、`sudo passwd db2inst1` コマンドを実行します。 3. db2inst1 インスタンスに接続するには、`sudo su - db2inst1` コマンドを実行します。 4. Db2 データベースに接続するには、db2 コマンドを実行します。 5. サンプルデータベースに接続するには、`connect to sample` コマンドを使用します。または、作成したデータベースに接続します。 6. データベースインスタンスに接続したら、Db2 SQL ステートメントを使用してオブジェクトを作成し、そのオブジェクトにデータを挿入します。	
Db2 DB インスタンスが利用可能であることを確認します。	Db2 データベースインスタンスが稼働中であることを確認するには、`Db2pd -` コマンドを使用します。	DBA

ターゲットの Aurora MySQL-Compatible Database を設定する

タスク	説明	必要なスキル
Aurora MySQL-Compatible Database を作成します。	AWS RDS サービスから MySQL 対応 Amazon Aurora Database の作成 • MySQL との互換性と任意のバージョンを使用して Amazon Aurora にデータベースを作成します。例:Aurora (MySQL) — 5.6.10a • MySQL Workbench アプリケーションまたは MySQL データベースへの接続を可能にする希望する DB クライアントソフトウェアをインストールします	DBA、AWS 全般
セキュリティグループを設定する	SSH 接続と TCP 接続の VPC セキュリティグループのインバウンドルールを設定します。	AWS 全般
Aurora データベースが使用可能であることを確認します。	Aurora MySQL 互換データベースが稼働していることを確認するには、以下を実行します。 1. SSH を使用して EC2 インスタンスに接続します。 2. MySQL ワークベンチから Aurora MySQL 互換インスタンスを設定して接続します。次の例に示すように、エンドポイントをホスト名として使用します。	DBA

タスク	説明	必要なスキル
	`mysql-cluster-instance-1.cokmvis0v46q.us-east-1.rds.amazonaws.com` 3. 新しいスキーマ (例:`mysql-sample-db2`) を作成して接続します。 4. MySQL ステートメントを実行して、データベース内のスキーマとオブジェクトを確認します。	

AWS SCT の設定と実行

タスク	説明	必要なスキル
AWS SCT をインストールします。	「AWS SCT」の最新バージョン (現在の最新バージョン 1.0.628) をダウンロードしてインストールします。	AWS 全般
AWS SCT を設定します。	1. IBM Db2 (4.22.X バージョン) と MySQL (8.x) 用の Java データベース接続 (JDBC) ドライバーをダウンロードします。 2. AWS SCT でドライバを設定するには、[設定]、[グローバル設定]、[ドライバー] を選択します。	AWS 全般
AWS SCT プロジェクトを作成します。	ソース DB エンジンとして Db2 for LUW を使用し、ター	AWS 全般

タスク	説明	必要なスキル
	ゲット DB エンジンとして Aurora MySQL 互換を使用する AWS SCT プロジェクトとレポートを作成します。 Db2 for LUW データベースへの接続に必要な権限を確認するには、「Db2 LUW を AWS SCT のソースとして使用する」を参照してください。	

タスク	説明	必要なスキル
オブジェクトを検証します。	[スキーマのダウンロード] を選択し、オブジェクトを検証します。ターゲットデータベース上の不正なオブジェクトをすべて更新します。 1. 接続の詳細を入力して Amazon Aurora MySQL 互換サーバーに接続し、[テスト接続] を選択します。 AWS SCT が移行レポートを開始する前に、ソース接続とターゲット接続の両方に成功する必要があります。 2. レポートが完成したら、変換するスキーマを入力し、[完了] を選択します。 AWS SCT は、変換され エラーのあるソースオブジェクトとターゲットオブジェクトをすべて一覧表示します。 3. エラーを確認し、手動でクリアします。 4. エラーをすべてクリアしたら、スキーマのコンテキスト (右クリック) メニューを開き、[スキーマのダウンロード] を選択します。 5. [データベースに適用] を選択します。	DBA、AWS 全般

タスク	説明	必要なスキル
	6. MySQL ワークベンチで、Aurora MySQL 互換データベースに接続し、スキーマとオブジェクトを確認します。	

AWS DMS の設定と実行

タスク	説明	必要なスキル
レプリケーションインスタンスを作成します。	AWS マネジメントコンソールにサインインし、AWS DMS サービスに移動し、ソースデータベースとターゲットデータベースに設定した VPC セキュリティグループの有効な設定を使用してレプリケーションインスタンスを作成します。	AWS 全般
エンドポイントを作成します。	Db2 データベースのソースエンドポイントを作成し、Aurora MySQL 互換データベースのターゲットエンドポイントを作成します。 1. [RDS DB インスタンスを選択] を選択し、作成した Db2 インスタンスを選択して、ソースとして IBM Db2 のエンドポイントを作成します。エンドポイント設定の詳細は自動的に入力されます。	AWS 全般

タスク	説明	必要なスキル
	2. エンドポイント固有の設定で、次の追加の接続属性を指定します。 `CurrentLSN=<scan>; MaxKBytesPerRead=64; SetDataCaptureChanges=true` これらの属性を指定しないと、ソースエンドポイントのテスト接続は成功しません。詳細については、「AWS DMS のソースとして IBM Db2 LUW の使用」を参照してください。 3. [RDS DB インスタンスを選択] を選択し、作成した Aurora MySQL 互換インスタンスを選択して、ターゲットとして Aurora MySQL 互換のエンドポイントを作成します。エンドポイント設定の詳細は自動的に入力されます。詳細については、「AWS Database Migration Service のターゲットとして MySQL 互換データベースの使用」を参照してください。 4. ソースエンドポイントとターゲットエンドポイントをテストします。両方とも	

タスク	説明	必要なスキル
	成功し、使用可能であることを確認します。 5. テストに失敗した場合は、セキュリティグループのインバウンドルールが有効であることを確認してください。	

タスク	説明	必要なスキル
移行タスクの作成	全負荷と CDC またはデータ検証用の 1 つの移行タスクまたは複数の移行タスクを作成します。 1. データベース移行タスクを作成するには、レプリケーションインスタンス、ソースデータベースエンドポイント、ターゲットデータベースエンドポイントを選択します。移行タイプを [既存データを移行 (全ロード)]、[データ変更のみを複製 (CDC)]、または [既存データを移行して継続的な変更を複製 (全ロードと CDC)] に指定します。 2. 「テーブルマッピング」では、選択ルールと変換ルールを GUI または JSON 形式で設定できます。 3. [選択ルール] で、スキーマを選択し、テーブル名を入力し、設定する「アクション (含める/除外)」を選択します (例:スキーマ:サンプル、テーブル名: %、アクション:インクルード)。 4. 「変換ルール」で、ターゲット (スキーマ、テーブル、または列) を選択します。スキーマ名を選択し、アクション (大文字	AWS 全般

タスク	説明	必要なスキル
	と小文字、プレフィックス、サフィックス) を選択します。例:ターゲット:スキーマ、mysql-sample-db ; アクション:小文字にする。 5. Amazon CloudWatch Logs モニタリングをオンにします。	
本番稼働を計画する。	アプリケーション所有者などの利害関係者とダウンタイムを確認し、本番システムで AWS DMS を実行してください。	移行リード
移行タスクを実行します。	1. ステータスが「準備完了」の AWS DMS タスクを開始します。 2. Amazon CloudWatch Logs での移行タスクログにエラーがないかモニタリングします。	AWS 全般

タスク	説明	必要なスキル
データを検証します。	移行タスクの結果と、移行元の Db2 データベースと移行先の MySQL データベースのデータを確認します。 1. ステータスが [継続的なレプリケーションのロード完了] の場合、CDC データ移行による全ロードが完了し、検証が進行中です。 2. Aurora MySQL 互換データベースに接続し、データを確認します。 3. Db2 データベースにデータを挿入または更新して、進行中の変更を確認します。	DBA
移行タスクを停止します。	データ検証が正常に完了したら、検証移行タスクを停止します。	AWS 全般

トラブルシューティング

問題	ソリューション
AWS SCT のソースとターゲットのテスト接続が失敗しています。	受信トラフィックを受け入れるように JDBC ドライバーバージョンと VPC セキュリティグループのインバウンドルールを設定します。
Db2 ソースエンドポイントのテスト実行は失敗します。	追加接続設定の `CurrentLSN=<scan>;` を行います。
AWSDMS タスクが Db2 ソースに接続できず、次のエラーが返されます。	エラーを回避するには、次のコマンドを実行します。

問題	ソリューション

database is recoverable if either or both of the database configuration parameters LOGARCHMETH1 and LOGARCHMETH2 are set to ON	1. $ db2 update db cfg for sample using LOGARCHMETH1 DISK:/home/db2inst1/logs
	2. $ db2stop
	3. $ db2start
	4. $ db2 connect to sample
	`SQL1116N A connection to or activation of database "SAMPLE" cannot be made because of BACKUP PENDING. SQLSTATE=57019`
	5. $ db2 backup database sample to ../logs
	`SQL2036N The path for the file or device "../logs" is not valid`
	6. $ cd
	7. $ pwd
	`/home/db2inst1`
	8. $ mkdir /tmp/backup
	9. $ db2 backup database sample to /tmp/backup
	`Backup successful. The timestamp for this backup image is : 20190530084921`
	10$ db2 connect to sample
	`Database Connection Information`
	`Database server = DB2/LINUX 9.7.1`
	`SQL authorization ID = DB2INST1`

問題	ソリューション
	`Local database alias = SAMPLE`

関連リソース

Amazon EC2

- Amazon EC2
- 「Amazon S3 ユーザーガイド」

データベース

- 「IBM Db2 データベース」
- Amazon Aurora
- 「Amazon Aurora MySQL の操作」

AWS SCT

- 「AWS DMS スキーマ変換」
- 「AWS スキーマ変換ツールユーザーガイド」
- 「AWS SCT ユーザーインターフェースの使用」
- 「IBM Db2 LUW を AWS SCT のソースとして使用する」

AWS DMS

- AWS Database Migration Service
- 「AWS Database Migration Service ユーザーガイド」
- 「データ移行のソース」
- 「データ移行のターゲット」
- 「AWS Database Migration Service と AWS Schema Conversion Tool がソースとして IBM Db2 LUW をサポートするようになりました」(ブログ投稿)
- 「リレーショナルデータベースを実行するアプリケーションの AWS への移行」

AWS DMS を使用して Microsoft SQL Server データベースを Amazon EC2 から Amazon DocumentDB に移行します

作成者: Umamaheswara Nolocated (AWS)

概要

このパターンでは、AWS Database Migration Service (AWS DMS) を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされている Microsoft SQL Server データベースを Amazon DocumentDB (MongoDB 互換の) データベースに移行する方法を説明します。

AWS DMS レプリケーションタスクは SQL Server データベースのテーブル構造を読み取り、Amazon DocumentDB に対応するコレクションを作成し、全負荷移行を実行します。

このパターンを使用して、SQL Server DB インスタンス用のオンプレミスの SQL Server または Amazon Relational Database Service (Amazon RDS) を Amazon DocumentDB に移行することもできます。詳細については、AWS 規範ガイダンスウェブサイトにある「Microsoft SQL Server データベースの AWS クラウドへの移行」ガイドを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- EC2 インスタンス上の既存の SQL Server データベース。
- SQL Server データベースの AWS DMS に割り当てられたデータベース (db_owner) ロールを修正しました。詳細については、SQL Server のドキュメントの「データベースレベルのロール」を参照してください。
- `mongodump`、`mongorestore`、`mongoexport`、および `mongoimport` の各ユーティリティを使用して、Amazon DocumentDB クラスターの内外にデータを移動できます。
- 「Microsoft SQL Server マネージメントスタジオ」のインストールと設定

機能制限

- Amazon DocumentDB のクラスターサイズ制限は 64 TB です。詳細については、Amazon DocumentDB ドキュメントの「クラスター制限」を参照してください。
- AWS DMS で複数のソーステーブルを 1 つの Amazon DocumentDB, コレクションにマージすることはできません。

- AWS DMS がプライマリキーのないソーステーブルからの変更を処理する場合、ソーステーブルのラージオブジェクト (LOB) 列は無視されます。

アーキテクチャ

ソーステクノロジースタック

- Amazon EC2

ターゲットアーキテクチャ

ターゲットテクノロジースタック

- Amazon DocumentDB

ツール

- AWS DMS — AWS Database Migration Service (AWS DMS) は、データベースを簡単かつ安全に移行するのに役立ちます。
- Amazon DocumentDB – Amazon DocumentDB は、MongoDB と互換性のある、高速で信頼性が高く、フルマネージドのドキュメントデータベースサービスです。
- 「Amazon EC2」— Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。
- Microsoft SQL サーバー — SQL サーバーはリレーショナルデータベース管理システムです。
- 「Microsoft SQL Server Management Studio (SSMS)」 は、SQL Server コンポーネントへのアクセス、設定、管理など、SQL Server を管理するためのツールです。

エピック

VPC の作成と設定

タスク	説明	必要なスキル
VPC を作成します。	AWS マネジメントコンソールにサインインし、「Amazon	システム管理者

タスク	説明	必要なスキル
	VPC コンソール」 を開きます。IPv4 CIDR ブロック範囲を使用して仮想プライベートクラウド (VPC) を作成します。	
セキュリティグループとネットワーク ACL を作成します。	Amazon VPC コンソールで、要件に応じて VPC のセキュリティグループとネットワークアクセスコントロールリスト (ネットワーク ACL) を作成します。これらの設定のデフォルト設定を使用することもできます。このストーリーやその他のストーリーについて詳しくは、「関連リソース」セクションを参照してください。	システム管理者

Amazon DocumentDB クラスターの作成と設定

タスク	説明	必要なスキル
Amazon DocumentDB クラスターを作成します。	Amazon DocumentDB コンソールを開き、クラスターを選択します。[作成] を選択し、1 つのインスタンスを含む Amazon DocumentDB クラスターを作成します。重要：必ず VPC のセキュリティグループを使用してこのクラスターを設定してください。	システム管理者
mongo シェルをインストールします。	mongo シェルは、Amazon DocumentDB クラスターを接	システム管理者

タスク	説明	必要なスキル
	続してクエリするために使用するコマンドラインユーティリティです。インストールするには、「/etc/yum.repos.d/mongodb-org-3.6.repo」コマンドを実行してリポジトリファイルを作成します。「sudo yum install-y mongodb-org-shell」コマンドを実行してmongo シェルをインストールします。転送中のデータを暗号化するには、Amazon DocumentDB のパブリックキーをダウンロードしてからAmazon DocumentDB インスタンスに接続します。このステップの詳細については、「関連リソース」セクションを参照してください。	
Amazon DocumentDB クラスターでデータベースを作成します。	データベースの名前を指定して「use」コマンドを実行し、Amazon DocumentDB クラスターにデータベースを作成します。	システム管理者

AWS DMS レプリケーションインスタンスを作成して設定します。

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	AWS DMS コンソールを開き、[レプリケーションインスタンスを作成] を選択します。レプリケーションタスクの名前と説明 (オプション) を	システム管理者

タスク	説明	必要なスキル
	入力します。インスタンスクラス、エンジンバージョン、ストレージ、VPC、マルチ AZ を選択し、パブリックにアクセスできるようにします。[Advanced (アドバンスト)] タブを選択して、ネットワークおよび暗号化設定の値を設定します。メンテナンス設定を指定し、「レプリケーションインスタンスを作成」を選択します。	
SQL Server データベースの設定	Microsoft SQL Server にログインし、ソースエンドポイントと AWS DMS レプリケーションインスタンス間の通信に関するインバウンドルールを追加します。レプリケーションインスタンスの IP アドレスをソースとして使用します。重要：レプリケーションインスタンスとターゲットエンドポイントは同じ VPC 上にある必要があります。ソースインスタンスとレプリケーションインスタンスで VPC が異なる場合は、セキュリティグループ内の代替ソースを使用してください。	システム管理者

AWS DMS でソースエンドポイントとターゲットエンドポイントを作成してテストする

タスク	説明	必要なスキル
ソースデータベースとターゲットデータベースを作成します。	AWS DMS コンソールを開き、[ソースとターゲットのデータベースエンドポイントをConnect] を選択します。ソースデータベースとターゲットデータベースの接続情報を指定します。必要に応じて [詳細設定] タブを選択し、[その他の接続属性] の値を設定します。証明書バンドルをダウンロードしてエンドポイント設定で使用します。	システム管理者
エンドポイント接続をテストします。	接続をテストするには、[テストの実行] を選択します。セキュリティグループの設定と、ソースとターゲットの両方のデータベースインスタンスから AWS DMS レプリケーションインスタンスへの接続を確認して、エラーメッセージのトラブルシューティングを行います。	システム管理者

データを移行する

タスク	説明	必要なスキル
AWS DMS 移行タスクを作成します。	AWS DMS コンソールで、[タスク]、[タスクの作成] を選択します。ソースとターゲットのエンドポイント名、レ	システム管理者

タスク	説明	必要なスキル
	プリケーションインスタンス名などのタスクオプションを指定します。「移行タイプ」で「既存データを移行」と「データ変更のみを複製」を選択します。[タスクを開始] を選択します。	
AWS DMS 移行タスクを実行します。	「タスク設定」で、「何もしない」、「テーブルをターゲットにドロップ」、「切り捨て」、「LOB 列をレプリケーションに含める」など、テーブル準備モードの設定を指定します。AWS DMS が受け付ける最大 LOB サイズを設定し、[ログ記録を有効にする] を選択します。[詳細設定] はデフォルト値のままにして、[タスクを作成] を選択します。	システム管理者
移行を監視します。	AWS DMS コンソールで「Tasks」を選択し、移行タスクを選択します。「タスクモニタリング」を選択してタスクをモニタリングします。全ロード移行が完了しキャッシュされた変更が適用されると、タスクは停止します。	システム管理者

移行のテストと検証

タスク	説明	必要なスキル
モンゴシェルを使用して Amazon DocumentDB クラスターConnect。	Amazon DocumentDBコンソールを開き、[クラスター] で、クラスターを見つけます。[接続性とセキュリティ] タブの、[mongo シェルを使用してこのクラスターに接続] を選択します。	システム管理者
移行の結果を確認します。	データベースの名前を指定して「使用」コマンドを実行し、次に「コレクションを表示」コマンドを実行します。使用しているデータベースの名前を指定して「db. .count ();」コマンドを実行します。結果がソースデータベースと一致すれば、移行は成功です。	システム管理者

関連リソース

VPC の作成と設定

- VPC 用のセキュリティグループを作成するには
- ネットワーク ACL の作成

Amazon DocumentDB クラスターの作成と設定

- Amazon DocumentDB クラスターの作成
- Amazon DocumentDB の mongo シェルをインストールします
- ステップ 5： Amazon DocumentDB クラスターに接続する

AWS DMS レプリケーションインスタンスを作成して設定

- パブリックおよびプライベートレプリケーション インスタンス

AWS DMS でソースエンドポイントとターゲットエンドポイントを作成してテストする

- Amazon DocumentDB を AWS DMS のターゲットとして使用します
- Microsoft SQL Server データベースを &DMS; のソースとして使用
- AWS DMS エンドポイントを使用する

データを移行

- Amazon DocumentDB への移行

その他のリソース

- SQL Server を AWS DMS のソースとして使用する場合の制限
- Amazon DocumentDB を使用してアプリケーションを大規模に構築および管理する方法

オンプレミスの ThoughtSpot Falcon データベースを Amazon Redshift に移行する

作成者: Battulga Purevragchaa (AWS)、Antony Prasad Thevaraj (AWS)

概要

オンプレミスのデータウェアハウスは、特に大規模なデータセットの場合、管理に多大な時間とリソースを必要とします。これらのウェアハウスの構築、維持、拡張にかかる財務コストも非常に高くなります。コストを管理し、抽出、変換、ロード (ETL) の複雑さを低く抑え、データの増加に応じてパフォーマンスを向上させるには、どのデータを読み込み、どのデータをアーカイブするかを常に選択する必要があります。

オンプレミスの ThoughtSpot Falcon データベースをAmazon Web Services (AWS) クラウドに移行することで、クラウドベースのデータレイクとデータウェアハウスにアクセスできるようになり、ビジネスの俊敏性、セキュリティ、アプリケーションの信頼性が向上し、インフラストラクチャ全体のコストが削減されます。Amazon Redshift は、データウェアハウスのコストと運用上のオーバーヘッドを大幅に削減するのに役立ちます。Amazon Redshift Spectrum を使用して、データをロードせずに大量のデータをネイティブ形式で分析することもできます。

このパターンでは、ThoughtSpot Falcon データベースをオンプレミスのデータセンターから AWS クラウド上の Amazon Redshift データベースに移行する手順とプロセスを説明しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスのデータセンターでホストされている ThoughtSpot Falcon データベース

製品バージョン

- ThoughtSpot バージョン 7.0.1

アーキテクチャ

この図表は、次のワークフローを示しています：

1. データはオンプレミスのリレーショナルデータベースでホストされます。

2. AWS Schema Conversion Tool (AWS SCT) は、Amazon Redshift と互換性のあるデータ定義言語 (DDL) を変換します。
3. テーブルを作成したら、AWS Database Migration Service (AWS DMS) を使用してデータを移行できます。
4. データは Amazon Redshift に読み込まれます。
5. Redshift Spectrum を使用しているか、既に Amazon S3 でデータをホストしている場合、データは Amazon Simple Storage Service (Amazon S3) に保存されます。

ツール

- AWS DMS – AWS データ移行サービス (AWS DMS) は、データベースを迅速かつ安全に AWS に移行するのに役立ちます。
- Amazon Redshift – Amazon Redshift は、高速でフルマネージドのペタバイト規模のデータウェアハウスサービスで、既存のビジネスインテリジェンスツールを使用してすべてのデータを簡単かつコスト効率よく効率的に分析できます。
- AWS SCT – AWS Schema Conversion Tool (AWS SCT) は、既存のデータベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換します。

エピック

移行の準備をする

タスク	説明	必要なスキル
適切な Amazon Redshift 設定を特定する。	要件とデータ量に基づいて、適切な Amazon Redshift クラスター設定を特定します。 詳細については、Amazon Redshift ドキュメントの「Amazon Redshift クラスター」を参照してください。	DBA
Amazon Redshift を調べて、要件を満たしているかどうかを評価する。	Amazon Redshift に関するよくある質問を参照して、Amazon Redshift が要件を満たし	DBA

タスク	説明	必要なスキル
	ているかどうかを理解し、評価します。	

ターゲットの Amazon Redshift クラスターを準備する

タスク	説明	必要なスキル
Amazon Redshift クラスターを作成します。	AWS マネジメントコンソールにサインインし、Amazon Redshift コンソールを開いて、仮想プライベートクラウド (VPC) で Amazon Redshift クラスターを作成します。 詳細については、Amazon Redshift のドキュメントの「VPC でクラスターを作成する」を参照してください。	DBA
Amazon Redshift データベース設計の PoC を実施する。	Amazon Redshift のベストプラクティスに従い、データベース設計の概念実証 (PoC) を実施します。 詳細については、Amazon Redshift のドキュメントの Amazon Redshift の概念実証の実施を参照してください。	DBA
データベースユーザーを作成する。	Amazon Redshift データベースにユーザーを作成し、スキーマとテーブルにアクセスするための適切なロールを付与します。	DBA

タスク	説明	必要なスキル
	詳細については、Amazon Redshift のドキュメントの「GRANT」を参照してください。	
ターゲットデータベースに構成設定を適用する。	要件に従って、Amazon Redshift データベースに構成設定を適用します。 データベース、セッション、およびサーバーレベルのパラメータを有効にする方法の詳細については、Amazon Redshift ドキュメントの「設定リファレンス」を参照してください。	DBA

Amazon Redshift クラスターでオブジェクトを作成する

タスク	説明	必要なスキル
Amazon Redshift で DDL を使用してテーブルを手動で作成する。	(オプション) AWS SCT を使用する場合、テーブルは自動的に作成されます。ただし、DDL の複製時に障害が発生した場合は、テーブルを手動で作成する必要があります。	DBA
Redshift Spectrum の外部テーブルを作成する。	Amazon Redshift Spectrum の外部スキーマを使用して外部テーブルを作成します。外部テーブルを作成するには、外部スキーマの所有者またはデータベーススーパーユーザーである必要があります。	DBA

タスク	説明	必要なスキル
	詳細については、Amazon Redshift のドキュメントの「Amazon Redshift Spectrum の外部テーブルの作成」を参照してください。	

AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
AWS DMS を使用してデータを移行する。	Amazon Redshift データベースにテーブルの DDL を作成したら、AWS DMS を使用してデータを Amazon Redshift に移行します。 詳細な手順については、AWS DMS のドキュメントの「Using an Amazon Redshift database as a target for AWS DMS」を参照してください。	DBA
COPY コマンドを使用してデータをロードする。	Amazon S3 から Amazon Redshift COPY コマンドを使用して Amazon Redshift にデータをロードします。 詳細については、Amazon Redshift ドキュメントの「COPY コマンドを使用し、Amazon S3 からロードする」を参照してください。	DBA

Amazon Redshift クラスターを確認する

タスク	説明	必要なスキル
ソースレコードとターゲットレコードを検証する。	ソースシステムからロードされたソースレコードとターゲットレコードのテーブル数を検証します。	DBA
Amazon Redshift のパフォーマンスチューニングのベストプラクティスを実装します。	Amazon Redshift テーブル設計のベストプラクティスを実装する。 詳細については、ブログ記事「Top 10 performance tuning techniques for Amazon Redshift」を参照してください。	DBA
クエリのパフォーマンスを最適化する。	Amazon Redshift は、SQL ベースのクエリを使用してシステム内のデータとオブジェクトを操作します。データ操作言語 (DML) は、データの表示、追加、変更、削除に使用できる SQL のサブセットです。DDL は、テーブルやビューなどのデータベースオブジェクトを追加、変更、削除するために使用する SQL のサブセットです。 詳細については、Amazon Redshift のドキュメントの「クエリパフォーマンスのチューニング」を参照してください。	DBA

タスク	説明	必要なスキル
WLM を実装する。	ワークロード管理 (WLM) を使用して複数のクエリキューを定義し、ランタイムにクエリを適切なキューに配信することができます。 詳細については、Amazon Redshift ドキュメントの「ワークロード管理の実装」を参照してください。	DBA
同時実行スケーリングを使用する。	同時実行スケーリング機能を使用すると、実質的に無制限の同時ユーザーと同時クエリをサポートし、一貫して高速なクエリパフォーマンスを実現できます。 詳細については、Amazon Redshift のドキュメントの「同時実行スケーリングを使用する」を参照してください。	DBA
テーブル設計に Amazon Redshift のベストプラクティスを使用する。	データベースをプランニングする際、テーブル設計に関して、全体的なクエリパフォーマンスに多大な影響を与える重要な決定があります。 最適なテーブル設計のオプションを選択する方法については、Amazon Redshift のドキュメントの「Amazon Redshift テーブル設計のベストプラクティス」を参照してください。	DBA

タスク	説明	必要なスキル
Amazon Redshift でマテリアライズドビューを作成する。	マテリアライズドビューには、1 つ以上のベーステーブルで実行された SQL クエリに基づいて事前計算された結果が含まれています。SELECT ステートメントを使用すれば、データベースで他のテーブルやビューをクエリするのと同じ方法でマテリアライズドビューをクエリすることができます。 詳細については、Amazon Redshift のドキュメントの「Amazon Redshift でのマテリアライズドビューの作成」を参照してください。	DBA

タスク	説明	必要なスキル
テーブル間の結合を定義する。	ThoughtSpot で複数のテーブルを同時に検索するには、2 つのテーブル間で一致するデータを含む列を指定して、テーブル間の結合を定義する必要があります。これらの列は、結合の `primary key` および `foreign key` を表します。 Amazon Redshift または ThoughtSpot の `ALTER TABLE` コマンドを使用してこれらを定義できます。詳細については、Amazon Redshift ドキュメントの「ALTER TABLE」を参照してください。	DBA

Amazon Redshift への ThoughtSpot 接続のセットアップ

タスク	説明	必要なスキル
Amazon Redshift 接続を追加する。	Amazon Redshift 接続をオンプレミスの ThoughtSpot Falcon データベースに追加します。 詳細については、ThoughtSpot ドキュメントの「Add a Redshift connection」を参照してください。	DBA

タスク	説明	必要なスキル
Amazon Redshift 接続を編集する。	Amazon Redshift 接続を編集してテーブルと列を追加できます。 詳細については、ThoughtSpot ドキュメントの「Edit a Redshift connection」を参照してください。	DBA
Amazon Redshift 接続を再マッピングする。	Amazon Redshift 接続を追加したときに作成されたソースマッピングの .yaml ファイルを編集して、接続パラメータを変更します。 例えば、既存のテーブルまたは列を、既存のデータベース接続の別のテーブルまたは列に再マップできます。ThoughtSpot では、接続内のテーブルまたは列を再マップする前後に依存関係を確認して、必要に応じて表示されることを確認することを推奨しています。 詳細については、ThoughtSpot ドキュメントの「Remap a Redshift connection」を参照してください。	DBA

タスク	説明	必要なスキル
Amazon Redshift 接続からのテーブルを削除する。	(オプション) Amazon Redshift 接続内のテーブルを削除しようとすると、ThoughtSpot は依存関係をチェックし、依存オブジェクトのリストを表示します。リストされたオブジェクトを選択して削除するか、依存関係を削除できます。その後、テーブルを削除できます。 詳細については、ThoughtSpot のドキュメントの「Delete a table from a Redshift connection」を参照してください。	DBA
Amazon Redshift 接続から依存オブジェクトを含むテーブルを削除する。	(オプション) 依存オブジェクトを含むテーブルを削除しようとすると、操作はブロックされます。`Cannot delete` ウィンドウが開き、依存オブジェクトへのリンクのリストが表示されます。依存関係がすべて削除されると、テーブルを削除できます。 詳細については、ThoughtSpot ドキュメントの「Delete a table with dependent objects from a Redshift connection」を参照してください。	DBA

タスク	説明	必要なスキル
Amazon Redshift 接続を削除する。	(オプション) 接続は複数のデータソースまたはビジュアライゼーションで使用できるので、Amazon Redshift 接続を削除する前に、その接続を使用するすべてのソースとタスクを削除する必要があります。 詳細については、ThoughtSpot ドキュメントの「Delete a Redshift connection」を参照してください。	DBA
Amazon Redshift の接続に関するリファレンスを確認する。	ThoughtSpot ドキュメントの接続リファレンスを使用して、Amazon Redshift 接続に必要な情報を必ず提供してください。	DBA

追加情報

- ThoughtSpot と Amazon Redshift によるあらゆる規模の AI 主導型アナリティクス
- Amazon Redshift の価格設定
- Getting started with AWS SCT
- Amazon Redshift の使用開始
- Using data extraction agents
- Chick-fil-A improves speed to insight with ThoughtSpot and AWS

AWS DMS を使用して Oracle データベースを Amazon DynamoDB に移行する

作成者: Rambabu Karnena (AWS)

概要

このパターンでは、AWS Database Migration Service (AWS DMS) を使用して Oracle データベースを Amazon DynamoDB に移行する手順を、順を追って説明します。対象は次の 3 種類のソースデータベースです。

- オンプレミスの Oracle データベース
- Amazon Elastic Compute Cloud (Amazon EC2) 上の Oracle Database
- Amazon DB インスタンス用 Amazon Relational Database Service (Amazon RDS)

この概念実証では、このパターンは Amazon RDS for Oracle DB インスタンスからの移行に焦点を当てています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon RDS for Oracle データベースに接続するアプリケーション
- ソース Amazon RDS for Oracle データベースにプライマリキーとサンプルデータを使用して作成されたテーブル

制限

- プロシージャ、関数、パッケージ、トリガーなどの Oracle データベースオブジェクトは、Amazon DynamoDB ではこれらのデータベースオブジェクトをサポートしていないため、移行の対象にはなりません。

製品バージョン

- このパターンは、AWS DMS でサポートされている Oracle データベースのすべてのエディションとバージョンに適用されます。詳細については、「Using an Oracle database as a source for AWS DMS」および「Using an Amazon DynamoDB database as a target for AWS Database Migration

Service」を参照してください。最も包括的なバージョンと機能サポートするため、最新バージョンを使用することをお勧めします。

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for Oracle DB インスタンス、Amazon EC2 上の Oracle、またはオンプレミス Oracle データベース

ターゲットテクノロジースタック

- Amazon DynamoDB

AWS データ移行アーキテクチャ

ツール

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。このパターンでは Amazon RDS for Oracle を使用します。

エピック

移行を計画する

タスク	説明	必要なスキル
VPC を作成します。	AWS アカウントで、仮想プライベートクラウド (VPC) とプ	システム管理者

タスク	説明	必要なスキル
	ライベートサブネットを作成します。	
セキュリティグループとネットワークアクセス制御リストを作成します。	詳細については、AWS ドキュメントを参照してください。	システム管理者
Amazon RDS for Oracle DB インスタンスを設定して起動します。	詳細については、AWS ドキュメントを参照してください。	DBA、システム管理者

データを移行する

タスク	説明	必要なスキル
DynamoDB にアクセスするための IAM ロールを作成します。	AWS Identity and Access Management (IAM) コンソールで、ロールを作成し、ポリシー `AmazonDynamoDBFullAccess to it` をアタッチして、AWS DMS をサービスとして選択します。	システム管理者
移行用の AWS DMS レプリケーションインスタンスを作成します。	レプリケーションインスタンスは、ソースデータベースと同じアベイラビリティゾーンおよび VPC に存在する必要があります。	システム管理者
AWS DMS でソースエンドポイントとターゲットエンドポイントを作成します。	ソースデータベースのエンドポイントを作成するには、次の 2 つのオプションがあります。 • Amazon RDS コンソールで、[データベース]、[DB 識	システム管理者

タスク	説明	必要なスキル
	別子]、[接続とセキュリティ] を選択し、エンドポイントを選択します。 • AWS DMS コンソールで [RDS DB インスタンスを選択] を選択します。 ターゲットデータベースのエンドポイントを作成するには、前のタスクの Amazon リソースネーム (ARN) を選択して DynamoDB にアクセスします。	
AWS DMS タスクを作成して、ソース Oracle データベーステーブルを DynamoDB にロードします。	ソースとターゲットのエンドポイント名、および前のステップのレプリケーションインスタンスを選択します。タイプは全負荷でもかまいません。Oracle スキーマを選択し、% を指定してすべてのテーブルを選択します。	システム管理者
DynamoDB の表を検証します。	移行結果を表示するには、DynamoDB コンソールの左側のナビゲーションペインから [テーブル] を選択します。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーションコードを変更します。	DynamoDB に接続してデータを取得するには、アプリケーションコードを更新します。	アプリ所有者、DBA、システム管理者

カットオーバー

タスク	説明	必要なスキル
DynamoDB を使用するようにアプリケーションクライアントを切り替えます。		DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
AWS リソースをシャットダウンします。	例えば、Amazon RDS for Oracle インスタンス、DynamoDB、および AWS DMS レプリケーションインスタンスをシャットダウンします。	DBA、システム管理者
メトリクスを収集します。	指標には、移行にかかる時間、手作業とツールが実行した作業の割合、コスト削減などが含まれます。	DBA、アプリ所有者、システム管理者

関連リソース

- AWS Database Migration Service and Amazon DynamoDB: What You Need to Know (ブログ投稿)
- 「AWS DMSのソースとして Oracle データベースを使用」

- Using an Amazon DynamoDB database as a target for AWS Database Migration Service
- Best Practices for Migrating from RDBMS to Amazon DynamoDB (ホワイトペーパー)

AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行

サウラヴ・ミシュラ（AWS）とエドゥアルド・ヴァレンティム（AWS）によって作成されました

概要

このパターンは、ネイティブパーティショニングをサポートしていない AWS Database Migration Service (AWS DMS) を使用して Oracle から PostgreSQL へのパーティションテーブルのロードを高速化する方法を示しています。ターゲット PostgreSQL データベースは Amazon Elastic Compute Cloud (Amazon EC2) にインストールすることも、Amazon Relational Database Service (Amazon RDS) for PostgreSQL または Amazon Aurora PostgreSQL-Compatible Edition DB インスタンスにすることもできます。

分割テーブルをアップロードするには、次の手順が含まれます。

1. Oracle パーティションテーブルと同様の親テーブルを作成しますが、パーティションは含めないでください。
2. ステップ 1 で作成した親テーブルを継承する子テーブルを作成します。
3. 親テーブルへの挿入を処理するプロシージャ関数とトリガーを作成します。

ただし、トリガーは挿入のたびに起動されるため、AWS DMS を使用した初期ロードは非常に遅くなる可能性があります。

Oracle から PostgreSQL 9.0 への初期ロードを高速化するために、このパターンはパーティションごとに個別の AWS DMS タスクを作成し、対応する子テーブルをロードします。次に、カットオーバー中にトリガーを作成します。

PostgreSQL バージョン 10 は、ネイティブパーティションをサポートしています。ただし、場合によっては継承されたパーティショニングを使用することもできます。詳細については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 分割テーブルを含むソース Oracle データベース
- AWS 上の PostgreSQL データベース

製品バージョン

- PostgreSQL 9.6

アーキテクチャ

ソーステクノロジースタック

- Oracle のパーティションテーブル。

ターゲットテクノロジースタック

- PostgreSQL のパーティションテーブル (Amazon EC2、Amazon RDS for PostgreSQL、または Aurora PostgreSQL 上)

ターゲット アーキテクチャ

ツール

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。

エピック

AWS DMS のセットアップ

タスク	説明	必要なスキル
PostgreSQL でテーブルを作成します。	パーティションに必要なチェック条件を使用して、PostgreSQLに親テーブルと対応する子テーブルを作成します。	DBA
パーティションごとに AWS DMS タスクを作成します。	AWS DMS タスクにパーティションのフィルタ条件を含め	DBA

タスク	説明	必要なスキル
	ます。パーティションを対応する PostgreSQL 子テーブルにマッピングします。	
フルロードおよび変更データキャプチャ (CDC) を使用して AWS DMS タスクを実行します。	`StopTaskCachedChangesApplied` パラメータが `true` に設定され、`StopTaskCachedChangesNotApplied` パラメータが `false` に設定されていることを確認します。	DBA

カットオーバー

タスク	説明	必要なスキル
レプリケーションタスクを停止します。	タスクを停止する前に、ソースとターゲットが同期されていることを確認します。	DBA
親テーブルにトリガーを作成します。	親テーブルはすべての挿入コマンドと更新コマンドを受け取るため、パーティショニング条件に基づいてこれらのコマンドをそれぞれの子テーブルにルーティングするトリガーを作成します。	DBA

関連リソース

- AWS DMS
- 「テーブルパーティショニング (PostgreSQL ドキュメント)」

追加情報

PostgreSQL バージョン 10 はネイティブパーティショニングをサポートしていますが、以下のユースケースでは継承パーティショニングを使用することもできます。

- パーティショニングでは、すべてのパーティションが親パーティションと同じ列セットを持つ必要があるというルールが適用されますが、テーブル継承では子が追加の列を持つこともサポートされます。
- テーブル継承は多重継承をサポートします。
- 宣言型パーティショニングは、リストパーティショニングとレンジパーティショニングのみをサポートします。テーブル継承では、データを必要に応じて分割できます。ただし、制約除外によってパーティションを効果的にプルーニングできなければ、クエリのパフォーマンスが低下します。
- 一部の操作では、宣言型パーティショニングを使用する方が、テーブル継承を使用する場合よりも強力なロックが必要になります。たとえば、`ACCESS EXCLUSIVE` パーティション化されたテーブルにパーティションを追加または削除するには親テーブルをロックする必要がありますが、`SHARE UPDATE EXCLUSIVE` 通常の継承にはロックで十分です。

個別のジョブパーティションを使用する場合、AWS DMS の検証に問題がある場合はパーティションをリロードすることもできます。パフォーマンスとレプリケーション制御を向上させるには、個別のレプリケーションインスタンスでタスクを実行します。

Amazon RDS for Oracle から Amazon RDS for MySQL に移行する

作成者: Jitender Kumar (AWS)、Neha Sharma (AWS)、Srini Ramaswamy (AWS)

概要

このパターンは、Oracle DB インスタンス用の Amazon Relational Database Service (Amazon RDS) を Amazon Web Services (AWS) 上の Amazon RDS for MySQL DB インスタンスに移行するためのガイダンスを提供します。このパターンでは、AWS Database Migration Service (AWS DMS) と AWS Schema Conversion Tool (AWS SCT) を使用します。

このパターンは、ストアドプロシージャの移行を処理するためのベストプラクティスを提供します。また、アプリケーションレイヤーをサポートする およびコード変更についても説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon RDS for Oracle ソースデータベース
- Amazon RDS for MySQL ターゲットデータベース。ソースデータベースとターゲットデータベースは、同じ Virtual Private Cloud (VPC) に存在する必要があります。複数の VPCs を使用している場合、または必要なアクセス許可が必要です。
- ソースとターゲットのデータベース、AWS SCT、アプリケーションサーバー、および AWS DMS 間の接続を許可するセキュリティグループ。
- ソースデータベースで AWS SCT を実行するために必要な権限があるユーザーアカウント。
- ソースデータベースで AWS DMS を実行するためのサプリメンタルロギングが有効になりました。

制約事項

- ソースとターゲットの Amazon RDS データベースのサイズ制限は 64 TB です。Amazon RDS のサイズ情報については、AWS ドキュメントを参照してください。
- Oracle ではデータベースオブジェクトの大文字と小文字は区別されますが、MySQL では区別されません。AWS SCT は、オブジェクトの作成中にこの問題を処理できます。ただし、完全な大文字と小文字の区別をサポートするためには、いくつかの手動作業が必要です。
- この移行では、Oracle ネイティブ機能を有効にするために MySQL 拡張機能は使用されません。AWS SCT がほとんどの変換を処理しますが、コードを手動で変更するには作業が必要です。

- Java データベース接続 (JDBC) ドライバーの変更がアプリケーションで必要です。

製品バージョン

- Amazon RDS for Oracle 12.2.0.1 以降。現在サポートされている RDS for Oracle のバージョンについては、AWS のドキュメントを参照してください。
- Amazon RDS for MySQL 8.0.15 以降。現在サポートされている RDS for MySQL のバージョンについては、AWS ドキュメントを参照してください。
- AWS DMS バージョン 3.3.0 以降。AWS DMS がサポートするソースエンドポイントとターゲットエンドポイントの詳細については、AWS ドキュメントを参照してください。
- AWS SCT バージョン 1.0.628 以降。 AWS ドキュメントの「AWS SCT ソースとターゲットエンドポイントのサポートマトリックス」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for Oracle 詳細については、「AWS DMS のソースとして Oracle データベースを使用する」を参照してください。

ターゲットテクノロジースタック

- Amazon RDS for MySQL 詳細については、「AWS DMS のターゲットとして MySQL 互換データベースを使用する」を参照してください。

移行アーキテクチャ

次の図では、AWS SCT は Amazon RDS for Oracle ソースデータベースからスキーマオブジェクトをコピーして変換し、そのオブジェクトを Amazon RDS for MySQL ターゲットデータベースに送信します。AWS DMS はソースデータベースからデータをレプリケートし、Amazon RDS for MySQL インスタンスに送信します。

ツール

- AWS Data Migration Service は、データストアを AWS クラウドに移行するか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役立ちます。

- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。このパターンでは、Amazon RDS for Oracle と Amazon RDS for MySQL を使用します。
- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマとカスタムコードの大部分を、ターゲットデータベースと互換性のある形式に自動的に変換することにより、異種データベース移行をサポートします。

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
適切なインスタンスタイプ（容量、ストレージ機能、ネットワーク機能）を選択します。		DBA、SysAdmin
ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、SysAdmin
アプリケーション移行戦略を選択します。	カットオーバー作業のために完全なダウンタイムと部分的なダウンタイムのどちらが必要かを検討してください。	DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
VPC とサブネットを作成します。		SysAdmin
セキュリティグループとネットワークアクセスコントロールリスト (ACL) を作成します。		SysAdmin
Amazon RDS for Oracle インスタンスの設定と起動を行います。		DBA、SysAdmin
Amazon RDS for MySQL インスタンスの設定と起動を行います。		DBA、SysAdmin
コード変換を検証するためのテストケースを準備します。	これは、変換されたコードのユニットテストに役立ちます。	DBA、開発者
AWS DMS インスタンスを設定します。		
AWS DMS でソースとターゲットのエンドポイントを設定します。		

データを移行する

タスク	説明	必要なスキル
AWS SCT を使用してターゲットデータベースのスクリプトを生成します。	AWS SCT によって変換されたコードの正確性を確認して	DBA、開発者

タスク	説明	必要なスキル
	ください。　手作業が必要です。	
AWS SCT で、[大文字と小文字を区別しない] 設定を選択します。	AWS SCT で、[プロジェクト設定]、[ターゲット大文字と小文字の区別]、[大文字と小文字を区別しない] を選択します。	DBA、開発者
AWS SCT では、Oracle ネイティブ関数を使用しないことを選択します。	プロジェクト設定で、TO_CHAR/TO_NUMBER/TO_DATE 関数をチェックします。	DBA、開発者
"sql%notfound" コードに変更を追加します。	コードを手動で変換する必要があります。	
ストアドプロシージャ内のテーブルとオブジェクトに対するクエリ (小文字のクエリを使用)。		DBA、開発者
すべての変更が行われた後、プライマリスクリプトを作成し、ターゲットデータベースにデプロイします。		DBA、開発者
サンプルデータを使用してストアドプロシージャとアプリケーションコールをユニットテストします。		
ユニットテスト中に作成されたデータをクリーンアップします。		DBA、開発者

タスク	説明	必要なスキル
外部キーの制約をターゲットデータベースにドロップします。	この手順は、初期データをロードするために必要です。外部キー制約をドロップしたくない場合は、プライマリテーブルとセカンダリテーブルに固有データ用の移行タスクを作成する必要があります。	DBA、開発者
プライマリキーとユニークキーをターゲットデータベースにドロップします。	このステップにより、初期ロードのパフォーマンスが向上します。	DBA、開発者
ソースデータベースでの補足的なログ記録を有効にします。		DBA
AWS DMS で初期ロード用の移行タスクを作成します。	[既存データを移行する] オプションを選択します。	DBA
プライマリキーと外部キーをターゲットデータベースに追加します。	初回ロード後に、制約を追加する必要があります。	DBA、開発者
継続的なレプリケーション用の移行タスクを作成します。	継続的なレプリケーションにより、ターゲットデータベースとソースデータベースの同期が維持されます。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
Oracle のネイティブ関数を MySQL のネイティブ関数に置き換えます。		アプリ所有者
SQL クエリのデータベースオブジェクトには、必ず小文字の名前のみを使用してください。		DBA、SysAdmin、アプリ所有者

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションサーバーをシャットダウンします。		アプリ所有者
ソースデータベースとターゲットデータベースが同期していることを検証します。		DBA、アプリ所有者
Amazon RDS for Oracle DB インスタンスを停止します。		DBA
移行タスクを停止します。	前のステップを完了すると、自動的に停止します。	DBA
JDBC 接続を Oracle から MySQL に変更します。		アプリ所有者、DBA
アプリケーションを起動します		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
プロジェクト文書を確認して検証する。		DBA、SysAdmin
移行の所要時間、手動タスクとツールタスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin
AWS DMS インスタンスを停止して削除します。		DBA
ソースおよびターゲットのエンドポイントを削除します。		DBA
移行タスクを削除します。		DBA
Amazon RDS for Oracle DB インスタンスのスナップショットを作成します。		DBA
Amazon RDS for Oracle DB インスタンスを削除します。		DBA
使用したその他の一時的な AWS リソースをシャットダウンして削除します。		DBA、SysAdmin
プロジェクトを閉じて、フィードバックします。		DBA

関連リソース

- AWS DMS
- AWS SCT

- Amazon RDS の価格設定
- AWS DMS の使用開始
- 「Amazon RDS の開始方法」

AWS DMS と AWS SCT を使用して IBM Db2 on Amazon EC2 から Aurora PostgreSQL 互換に移行する

作成者: Sirsendu Halder (AWS) と Abhimanyu Chhabra (AWS)

概要

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2)インスタンス上の IBM Db2 データベースを Amazon Aurora PostgreSQL 互換エディション DB インスタンスに移行するためのガイダンスが提供されます。このパターンでは、AWS Database Migration Service (AWS DMS) と AWS Schema Conversion Tool (AWS SCT) を使用して、データ移行とスキーマ変換を行います。

このパターンでは、トランザクション数の多いテラバイト規模の IBM Db2 ソースデータベースに対して、ダウンタイムをほとんどまたはまったく発生させないオンライン移行戦略を目指しています。パフォーマンスを向上させるには、PostgreSQL で `NUMERIC` データ型のプライマリキー (PK) と外部キー (FK) の列を `INT` または `BIGINT` に変換することをお勧めします。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- EC2 インスタンス上のソース IBM Db2 データベース

製品バージョン

- DB2/LINUXX8664 バージョン 11.1.4.4 以降

アーキテクチャ

ソーステクノロジースタック

- EC2 インスタンス上の Db2 データベース

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換バージョン 10.18 以降の DB インスタンス

データベース移行アーキテクチャ

ツール

- AWS Database Migration Service (AWS DMS) を使用すると、AWS クラウドに、またはクラウドとオンプレミスを組み合わせたセットアップの間にデータストアを移行できます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS は、広く普及しているほとんどの商用データベースとオープンソースデータベース間のデータ移行にご利用いただけます。AWS DMS は、IBM Db2 から Aurora PostgreSQL 互換バージョン 10.18 以降への移行など、異なるデータベースプラットフォーム間での異種間の移行をサポートしています。詳細については、AWS DMS ドキュメントの「データ移行のソース」と「データ移行のターゲット」を参照してください。
- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のデータベースコードオブジェクト (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを簡単に移行できるようにします。自動的に変換されないオブジェクトにはわかりやすいマークが付いており、手動で変換して移行を完了できます。AWS SCT では、アプリケーションのソースコードに埋め込まれた SQL ステートメントをスキャンして変換することもできます。

エピック

環境をセットアップする

タスク	説明	必要なスキル
Aurora PostgreSQL と互換性のある DB インスタンスを作成します。	DB インスタンスを作成するには、「AWS ドキュメント」の指示に従ってください。[エンジンのタイプ] には、[Amazon Aurora] を選択します。[エディション] には、[Amazon Aurora PostgreSQL 互換エディション] を選択します。 Aurora PostgreSQL 互換バージョン 10.18 以降の DB インスタンスは、ソース IBM Db2 データベースと同じ仮想プラ	Amazon RDS

タスク	説明	必要なスキル
	イベートクラウド (VPC) 内に配置する必要があります。	

データベーススキーマを変換する

タスク	説明	必要なスキル
AWS SCT をインストールして検証します。	1. 「AWS SCT ドキュメント」の指示に従って AWS SCT をインストールします。 2. 「AWS SCT ドキュメント」の手順に従ってインストールを検証します。	AWS 管理者、DBA、移行エンジニア
AWS SCT を起動してプロジェクトを開始します。	AWS SCT ツールを起動し、データベース移行評価レポートを実行する新しいプロジェクトを作成するには、AWS SCT ドキュメントの指示に従ってください。	移行エンジニア
データベースサーバーを追加し、マッピングルールを追加します。	1. AWS SCT ドキュメントの指示に従ってソースとターゲットのデータベースサーバーを追加します。 2. ソースデータベースのターゲットデータベースプラットフォームを定義するマッピングルールを作成します。 手順については、「AWS SCTドキュメント」を参照してください。	移行エンジニア

タスク	説明	必要なスキル
データベース移行評価レポートを作成します。	「AWS SCT ドキュメント」の手順に従って、データベース移行評価レポートを作成します。	移行エンジニア
評価レポートを表示します。	データベース移行評価レポートの [概要] タブを使用してレポートを表示し、データを分析します。この分析は、移行の複雑さを判断するのに役立ちます。詳細については、「AWS SCT ドキュメント」を参照してください。	移行エンジニア
スキーマを変換します。	データベーススキーマを変換するには: 1. AWS SCT コンソールで、[ビュー] を選択してから、[メインビュー] を選択します。 2. ソーススキーマからオブジェクトまたは親ノードを選択し、コンテキスト (右クリック) メニューを開いてから、[スキーマの変換] を選択します。 詳細については、「AWS SCT ドキュメント」を参照してください。	移行エンジニア

タスク	説明	必要なスキル
変換されたデータベーススキーマをターゲット DB インスタンスに適用します。	1. ターゲット DB インスタンスに計画したスキーマを表示するスキーマ要素を、プロジェクトの右パネルで選択します。 2. スキーマ要素のコンテキスト (右クリック) メニューを開き、[Apply to database] (データベースに適用) を選択します。 詳細については、「AWS SCT ドキュメント」を参照してください。	移行エンジニア

データを移行する

タスク	説明	必要なスキル
VPC と DB パラメータグループを設定します。	VPC と DB のパラメータグループを設定し、移行に必要なインバウンドルールとパラメータを設定します。手順については、「AWS DMS ドキュメント」を参照してください。 VPC セキュリティグループには、Db2 の EC2 インスタンスと Aurora PostgreSQL 互換の DB インスタンスの両方を選択します。 このレプリケーションインスタンスは、ソースおよびターゲットの DB イ	移行エンジニア

タスク	説明	必要なスキル
	ンスタンスと同じ VPC 内に配置する必要があります。	
ソースとターゲットの DB インスタンスを準備します。	移行に向けてソースとターゲットの DB インスタンスを準備します。　ソースデータベースは、すでに本番環境で存在しています。 ソースデータベースの場合、サーバー名は Db2 が実行されている EC2 インスタンスのパブリックドメインネームシステム (DNS) にする必要があります。ユーザー名には、`db2inst1` コードとポートを使用できます。IBM Db2 の場合は 5000 になります。	移行エンジニア

タスク	説明	必要なスキル
Amazon EC2 クライアントとエンドポイントを作成します。	1. Amazon EC2 クライアントを作成します。このクライアントを使用して、レプリケートするデータをソースデータベースに入力します。また、このクライアントを使用して、ターゲットデータベースでクエリを実行してレプリケーションを検証します。 2. 次の手順で使用するソースデータベースとターゲット DB インスタンスのエンドポイントを作成します。手順については、「AWS DMS ドキュメント」を参照してください。ソースデータベースとターゲットデータベースに個別のエンドポイントを作成します。Aurora PostgreSQL 互換バージョン 10.18 以降の場合、ポートは 5432 になり、DB インスタンスのエンドポイントからサーバー名を取得できます。	移行エンジニア

タスク	説明	必要なスキル
レプリケーションインスタンスを作成します。	AWS DMS コンソールを使用してレプリケーションインスタンスを作成し、ソースやターゲットのエンドポイントを指定します。レプリケーション インスタンスは、エンドポイント間でデータ移行を行います。詳細については、AWS DMS のドキュメントを参照してください。	移行エンジニア
AWS DMS タスクを作成して、データを移行します。	「AWS DMS ドキュメント」の手順に従って、ソース IBM Db2 テーブルをターゲット PostgreSQL DB インスタンスにロードするタスクを作成します。 • ソースとターゲットには、ソースとターゲットのエンドポイント名を使用してください。 • 移行タイプは全ロードで問題ありません。 • スキーマルールには、Db2 データベースの `inst1` スキーマを使用できます。 • テーブル名に % を指定して、すべてのテーブルを移行します。ロードが完了すると、`inst1` スキーマの Db2 テーブルが Aurora PostgreSQL 互換データベースに表示されます。	移行エンジニア

関連リソース

リファレンス

- 「Amazon Aurora ドキュメント」
- 「PostgreSQL 外部データラッパー (FDW) に関するドキュメント」
- 「PostgreSQL 外部スキーマのインポートに関するドキュメント」
- 「AWS DMS のドキュメント」
- AWS SCT のドキュメント

チュートリアルと動画

- AWS DMS の使用開始 (チュートリアル)
- Amazon EC2 のご紹介 - Elastic クラウドサーバーと AWS でのホスティング (動画)

SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行

作成者: Kumar Babu P G (AWS)

概要

このパターンでは、オンプレミスの Oracle 8i または 9i データベースを、Amazon Relational Database Service (Amazon RDS) for PostgreSQL または Amazon Aurora に移行する方法を説明しています。AWS Database Migration Service (AWS DMS) は Oracle 8i または 9i をソースとしてサポートしていないため、Quest SharePlex はオンプレミスの 8i または 9i データベースから、AWS DMS と互換性のある中間の Oracle データベース (Oracle 10g または 11g) にデータを複製します。

中間の Oracle インスタンスから、スキーマとデータは AWS Schema Conversion Tool (AWS SCT) と AWS DMS を使用して AWS 上の PostgreSQL データベースに移行されます。この方法では、レプリケーションの遅延を最小限に抑えながら、ソース Oracle データベースからターゲット PostgreSQL DB インスタンスにデータを継続的にストリーミングできます。この実装では、ダウンタイムは、ターゲット PostgreSQL データベース上のすべての外部キー、トリガー、およびシーケンスを作成または検証するのにかかる時間に制限されます。

移行では、Oracle 10g または 11g がインストールされた Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用して、ソース Oracle データベースからの変更内容をホストします。AWS DMS は、この中間 Oracle インスタンスをソースとして使用し、Amazon RDS for PostgreSQL または Aurora PostgreSQL にデータをストリーミングします。オンプレミスの Oracle データベースから中間 Oracle インスタンスへのデータ複製を一時停止して再開できます。また、中間 Oracle インスタンスからターゲット PostgreSQL データベースに一時停止して再開できるため、AWS DMS データ検証ツールまたはカスタムデータ検証ツールのいずれかを使用してデータを検証できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスデータセンターにあるソース Oracle 8i または 9i データベース
- オンプレミスデータセンターと AWS の間に AWS Direct Connect を設定
- AWS SCT がインストールされている、ローカルマシンまたは EC2 インスタンスのどちらかにインストールされている AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバー
- 「Oracle データベースを AWS DMS ソースとして使用」 に精通している
- 「Oracle データベースを AWS DMS ソースとしての使用」に精通している

- Quest SharePlex データレプリケーションに精通している

制約事項

- データベースのサイズ制限は 64 TB です
- オンプレミスの Oracle データベースはエンタープライズエディションである必要があります

製品バージョン

- ソースデータベース用の Oracle 8i または 9i
- 中間データベースの Oracle 10g または 11g
- PostgreSQL 9.6 以降

アーキテクチャ

ソーステクノロジースタック

- Oracle 8i または 9i データベース
- クエスト シェアプレックス

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL または Aurora PostgreSQL

ソースアーキテクチャとターゲットアーキテクチャ

ツール

- AWS DMS — 「AWS Database Migration Service (AWS DMS)」 は、データベースを迅速かつ安全に移行するのに役立ちます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS により、広く普及しているほとんどの商用データベースとオープンソースデータベース間で、データを移行することができます。

- AWS SCT — AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコード (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを簡単に移行できるようにします。自動的に変換できないオブジェクトにはわかりやすいマークが付いていて、手動で変換して移行を完了できます。AWS SCT では、アプリケーションのソースコードをスキャンして、埋め込み SQL ステートメントを探し、データベーススキーマ変換プロジェクトの一部として変換することもできます。このプロセスでは、AWS SCT は Oracle と SQL Server のレガシー関数を AWS の同等の機能に変換することでクラウドネイティブなコードの最適化を行います。これにより、データベースを移行しながらアプリケーションを最新化できます。スキーマの変換が完了すると、AWS SCT は組み込みのデータ移行エージェントを使用して、さまざまなデータウェアハウスから Amazon Redshift へのデータ移行を支援します。
- Quest SharePlex — Quest SharePlex は Oracle 間のデータレプリケーションツールです。ダウンタイムを最小限に抑え、データ損失なしでデータを移動できます。

エピック

EC2 インスタンスを作成して Oracle をインストール

タスク	説明	必要なスキル
Amazon EC2 のネットワークをセットアップします。	仮想プライベートクラウド (VPC)、サブネット、インターネットゲートウェイ、ルートテーブル、およびセキュリティグループを作成します。	AWS SysAdmin
新しい EC2 インスタンスを作成します。	EC2インスタンスの Amazon マシンイメージ (AMI)を選択します。インスタンスサイズを選択し、インスタンスの詳細 (インスタンス数 (1)、前のステップの VPC とサブネット、パブリック IP の自動割り当て、その他のオプション)を設定します。ストレージを追加し、セキュリティグループ	AWS SysAdmin

タスク	説明	必要なスキル
	を設定し、インスタンスを起動します。プロンプトが表示されたら、次のステップのためにキーペアを作成して保存します。	
EC2 インスタンスに Oracle をインストールします。	ライセンスと必要な Oracle バイナリを取得し、EC2 インスタンスに Oracle 10g または 11g をインストールします。	DBA

EC2 インスタンスに SharePlex をセットアップし、データレプリケーションを設定します

タスク	説明	必要なスキル
SharePlex をセットアップします。	Amazon EC2 インスタンスを作成して、Oracle 8i または 9i と互換性のある SharePlex バイナリをインストールします。	AWS SysAdmin、DBA
データレプリケーションを設定します。	SharePlex のベストプラクティスに従って、オンプレミスの Oracle 8i/9i データベースから Oracle 10g/11g インスタンスへのデータレプリケーションを設定します。	DBA

Oracle データベーススキーマを PostgreSQL に変換

タスク	説明	必要なスキル
AWS SCT をセットアップします。	新しいレポートを作成し、ソースとして Oracle に、ター	DBA

タスク	説明	必要なスキル
	ゲットとして PostgreSQL に接続します。プロジェクト設定で SQL スクリプトタブを開き、ターゲット SQL スクリプトを「複数ファイル」に変更します。	
Oracle データベーススキーマを変換します。	アクションタブで、レポートを生成、スキーマを変換、SQL として保存の順に選択します。	DBA
AWS SCT によって生成された SQL スクリプトを変更します。		DBA

Amazon RDS DB インスタンスを作成して設定する

タスク	説明	必要なスキル
Amazon RDS DB インスタンスを作成します。	Amazon RDS コンソールで、新しい PostgreSQL DB インスタンスを作成します。	AWS SysAdmin、DBA
DB インスタンスを設定します。	DB エンジンのバージョン、DB インスタンスクラス、マルチ AZ 配置、ストレージタイプ、ストレージの割り当てを指定します。DB インスタンス識別子、マスターユーザー名、マスターパスワードを入力します。	AWS SysAdmin、DBA
ネットワークとセキュリティを設定します。	VPC、サブネットグループ、パブリックアクセシビリティ	AWS SysAdmin、DBA

タスク	説明	必要なスキル
	、アベイラビリティーゾーンプリファレンス、セキュリティグループを指定します。	
データベースオプションを設定します。	データベース名、ポート、パラメータグループ、暗号化、マスターキーを指定します。	AWS SysAdmin、DBA
バックアップを設定します。	バックアップ保持期間、バックアップウィンドウ、開始時刻、期間、およびタグをスナップショットにコピーするかどうかを指定します。	SysAdmin、DBA
モニタリングオプションを設定します。	拡張モニタリングとパフォーマンスインサイトの有効または無効にします。	AWS SysAdmin、DBA
メンテナンスオプションを設定します。	マイナーバージョンの自動マイナーアップグレード、メンテナンスウィンドウ、および開始日、時刻、および期間を指定します。	AWS SysAdmin、DBA
AWS SCT から移行前スクリプトを実行します。	Amazon RDS インスタンスで、create_database.sql、create_sequence.sql、create_table.sql、create_view.sql、create_function.sql のスクリプトを実行します。	AWS SysAdmin、DBA

AWS DMS を使用してデータを移行

タスク	説明	必要なスキル
AWS DMSでレプリケーションインスタンスを作成します。	名前、インスタンスクラス、VPC (EC2 インスタンスと同じ)、マルチ AZ、パブリックアクセシビリティのフィールドを入力します。詳細設定セクションで、割り当てられたストレージ、サブネットグループ、アベイラビリティーゾーン、VPC セキュリティグループ、および AWS Key Management Service (AWS KMS) ルートキーを指定します。	AWS SysAdmin、DBA
ソースデータベースのエンドポイントを作成します。	エンドポイント名、タイプ、ソースエンジン (Oracle)、サーバー名 (Amazon EC2 プライベート DNS 名)、ポート、SSL モード、ユーザー名、パスワード、SID、VPC (レプリケーションインスタンスのある VPC を指定)、およびレプリケーションインスタンスを指定します。接続をテストするには、[テストを実行] を選択し、エンドポイントを作成します。maxFileSize と numberDataTypeScale の詳細設定も行えます。	SysAdmin、DBA
AWS DMS レプリケーションタスクを作成します。	タスク名、レプリケーションインスタンス、ソースとターゲットのエンドポイント、レ	AWS SysAdmin、DBA

タスク	説明	必要なスキル
	プリケーションインスタンスを指定します。移行タイプについては、[既存のデータを移行し、実行中の変更をリプリケートする] を選択します。「作成時にタスクを開始」チェックボックスをクリアします。	
AWS DMS レプリケーションタスク設定を構成します。	ターゲットテーブル作成モードには、 [何もしない] を選択します。全ロードが完了したらタスクを停止して、プライマリキーを作成します。制限付き LOB モードまたはフル LOB モードを指定し、制御テーブルを有効にします。オプションで CommitRate の詳細設定を行えます。	DBA
テーブルマッピングを設定します。	テーブルマッピングセクションで、移行に含まれるすべてのスキーマのすべてのテーブルを対象とするインクルードルールを作成し、除外ルールを作成します。スキーマ、テーブル、列の名前を小文字に変換する 3 つの変換ルールを追加し、特定の移行に必要なその他のルールを追加します。	DBA

タスク	説明	必要なスキル
タスクの開始	レプリケーションタスクを開始します。全ロードが実行されていることを確認します。プライマリ Oracle データベースで ALTER SYSTEM SWITCH LOGFILE を実行してタスクを開始します。	DBA
AWS SCT から移行中のスクリプトを実行します。	Amazon RDS for PostgreSQL では、create_index.sql と create_constraint.sql というスクリプトを実行します。	DBA
タスクを再開して、変更データキャプチャ (CDC) を続行します。	Amazon RDS for PostgreSQL DB インスタンスで VACUUM を実行し、AWS DMS タスクを再起動して、キャッシュされた CDC の変更を適用します。	DBA

PostgreSQL データベースへのカットオーバー

タスク	説明	必要なスキル
AWS DMS のログとメタデータテーブルを確認します。	エラーを検証し、必要に応じて修正します。	DBA
Oracle の依存関係をすべて停止します。	Oracle データベースのリスナーをシャットダウンし、ALTER SYSTEM SWITCH ログファイルを実行します。アクティビティが表示されない場合は、AWS DMS タスクを停止します。	DBA

タスク	説明	必要なスキル
AWS SCT から移行後のスクリプトを実行します。	Amazon RDS for PostgreSQL では、以下のスクリプトを実行します： create_foreign_key_constraint.sql と create_triggers.sql 。	DBA
Amazon RDS for PostgreSQL のその他のステップをすべて完了します。	必要に応じて Oracle と一致するようにシーケンスをインクリメントし、VACUUM と ANALYZE を実行し、コンプライアンスのためのスナップショットを撮ります。	DBA
[Amazon RDS for PostgreSQL に接続] を選択します。	Amazon RDS for PostgreSQL から AWS DMS セキュリティグループを削除し、実稼働セキュリティグループを追加して、アプリケーションを新しいデータベースに接続します。	DBA
AWS リソースをクリーンアップします。	エンドポイント、レプリケーションタスク、レプリケーションインスタンス、EC2 インスタンスを削除します。	SysAdmin、DBA

関連リソース

- AWS DMS のドキュメント
- AWS SCT のドキュメント
- 「Amazon RDS for PostgreSQL 料金設定」
- 「AWS DMSのソースとして Oracle データベースを使用」
- AWS DMSのターゲットとして PostgreSQL データベースを使用
- Quest SharePlex ドキュメント

マテリアライズドビューと AWS DMS を使用して Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行

作成者: Kumar Babu P G (AWS) と Pragnesh Patel (AWS)

概要

このパターンでは、オンプレミスの Oracle 8i または 9i データベースを、Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL 互換エディションに移行する方法について説明しています。

AWS Database Migration Service (AWS DMS) は Oracle 8i または 9i をソースとしてサポートしていないため、このパターンでは、Oracle 10g や 11g などの AWS DMS と互換性のある中間 Oracle データベースインスタンスを使用します。また、マテリアライズドビュー特徴量を使用して、ソース Oracle 8i/9i インスタンスから中間の Oracle 10g/11g インスタンスにデータを移行します。

AWS Schema Conversion Tool (AWS SCT) はデータベーススキーマを変換し、AWS DMS はデータをターゲット PostgreSQL データベースに移行します。

このパターンでは、データベースのダウンタイムを最小限に抑えてレガシー Oracle データベースから移行したいユーザーに役立ちます。この実装では、ダウンタイムはターゲットデータベース上のすべての外部キー、トリガー、シーケンスを作成または検証するのにかかる時間に限定されます。

このパターンでは、AWS DMS によるデータのストリーミングを支援するのに、Oracle 10g/11g データベースをインストールした Amazon Elastic Compute Cloud (Amazon EC2) インスタンスが使用されています。オンプレミスの Oracle データベースから中間 Oracle インスタンスへのストリーミングレプリケーションを一時停止して、AWS DMS がデータ検証で確認ができるようにしたり、または別のデータ検証ツールを使用したりできます。AWS DMS が現在の変更の移行を完了すると、PostgreSQL DB インスタンスと中間 Oracle データベースに同じデータが含まれます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスデータセンターにあるソース Oracle 8i または 9i データベース
- オンプレミスデータセンターと AWS の間に AWS Direct Connect を設定
- AWS SCT がインストールされている、ローカルマシンまたは EC2 インスタンスのどちらかにインストールされている AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバー
- 「Oracle データベースを AWS DMS ソースとして使用」 に精通している

- 「AWS DMS ターゲットとして PostgreSQL データベースを使用」 に精通している

機能制限

- データベースのサイズ制限は 64 TB です

製品バージョン

- ソースデータベース用の Oracle 8i または 9i
- 中間データベースの Oracle 10g または 11g
- PostgreSQL 10.17 以降

アーキテクチャ

ソーステクノロジースタック

- Oracle 8i または 9i データベース

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL 互換

ターゲットアーキテクチャ

ツール

- 「AWS DMS」 は、データベースを迅速かつ安全に移行するのに役立ちます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS により、広く普及しているほとんどの商用データベースとオープンソースデータベース間で、データを移行することができます。
- AWS SCT は、ソースデータベーススキーマと大部分のカスタムコード (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的にk変換します。自動的に変換できないオブジェクトにはわかりやすいマークが付いていて、手動で変換して移行を完了できます。AWS SCT では、アプリケーションのソースコードをスキャンして、埋め込み SQL ステートメントを探し、データベーススキーマ変換プロジェクトの一部として変換することもで

きます。このプロセスでは、AWS SCT は Oracle と SQL Server のレガシー関数を AWS の同等の機能に変換することでクラウドネイティブなコードの最適化を行います。これにより、データベースを移行しながらアプリケーションを最新化できます。スキーマの変換が完了すると、AWS SCT は組み込みのデータ移行エージェントを使用して、さまざまなデータウェアハウスから Amazon Redshift へのデータ移行を支援します。

ベストプラクティス

マテリアライズドビューを更新するベストプラクティスについては、以下の Oracle ドキュメントを参照してください：

- 「マテリアライズドビューの更新」
- 「マテリアライズドビューの高速リフレッシュ」

エピック

EC2 インスタンスに Oracle をインストールし、マテリアライズドビューを作成

タスク	説明	必要なスキル
EC2 インスタンスのネットワークをセットアップします。	仮想プライベートクラウド (VPC)、サブネット、インターネットゲートウェイ、ルートテーブル、およびセキュリティグループを作成します。	AWS SysAdmin
EC2 インスタンスを作成します。	EC2 インスタンスの Amazon マシンイメージ (AMI)を選択します。インスタンスサイズを選択し、インスタンスの詳細 (インスタンス数 (1)、前のステップの VPC とサブネット、パブリック IP の自動割り当て、その他のオプション)を設定します。ストレージを追加し、セキュリティグループ	AWS SysAdmin

タスク	説明	必要なスキル
	を設定し、インスタンスを起動します。プロンプトが表示されたら、次のステップのためにキーペアを作成して保存します。	
EC2 インスタンスに Oracle をインストールします。	ライセンスと必要な Oracle バイナリを取得し、EC2 インスタンスに Oracle 10g または 11g をインストールします。	DBA
Oracle ネットワークを設定します。	`listener.ora` のエントリを変更または追加してオンプレミスのソース Oracle 8i/9i データベースに接続し、データベースリンクを作成します。	DBA
マテリアライズドビューを作成します。	ソース Oracle 8i/9i データベースで複製するためのデータベースオブジェクトを特定し、データベースリンクを使用してすべてのオブジェクトのマテリアライズドビューを作成します。	DBA
スクリプトをデプロイして、必要な間隔でマテリアライズドビューを更新します。	Amazon EC2 Oracle 10g/11g インスタンスで、必要な間隔でマテリアライズドビューを更新するスクリプトを開発してデプロイします。増分更新オプションを使用してマテリアライズドビューを更新します。	DBA

Oracle データベーススキーマを PostgreSQL に変換

タスク	説明	必要なスキル
AWS SCT をセットアップします。	新しいレポートを作成し、次にソースとして Oracle に、ターゲットとして PostgreSQL に接続します。プロジェクト設定で、[SQL スクリプト] タブを開きます。ターゲット SQL スクリプトを複数ファイルに変更します。(AWS SCT が Oracle 8i/9i データベースに適用されないため、中間の Oracle 10g/11g インスタンスでスキーマのみのダンプを復元し、それを AWS SCT のソースとして使用する必要があります)。	DBA
Oracle データベーススキーマを変換します。	[アクション] タブで、[レポートを生成]、[スキーマを変換]、[SQL として保存] の順に選択します。	DBA
SQL スクリプトを変更します。	ベストプラクティスに基づいて変更を加えます。たとえば、適切なデータ型に切り替えて、Oracle 固有の関数に対応する PostgreSQL を開発します。	DBA、DevDBA

Amazon RDS DB インスタンスを作成・設定して、変換されたデータベースをホストする

タスク	説明	必要なスキル
Amazon RDS DB インスタンスを作成します。	Amazon RDS コンソールで、新しい PostgreSQL DB インスタンスを作成します。	AWS SysAdmin、DBA
DB インスタンスを設定します。	DB エンジンのバージョン、DB インスタンスクラス、マルチ AZ 配置、ストレージタイプ、ストレージの割り当てを指定します。DB インスタンス識別子、マスターユーザー名、マスターパスワードを入力します。	AWS SysAdmin、DBA
ネットワークとセキュリティを設定します。	VPC、サブネットグループ、パブリックアクセシビリティ、アベイラビリティーゾーンプリファレンス、セキュリティグループを指定します。	DBA、SysAdmin
データベースオプションを設定します。	データベース名、ポート、パラメータグループ、暗号化、マスターキーを指定します。	DBA, AWS SysAdmin
バックアップを設定します。	バックアップ保持期間、バックアップウィンドウ、開始時刻、期間、およびタグをスナップショットにコピーするかどうかを指定します。	SysAdmin、DBA
モニタリングオプションを設定します。	拡張モニタリングとパフォーマンスインサイトの有効または無効にします。	AWS SysAdmin、DBA

タスク	説明	必要なスキル
メンテナンスオプションを設定します。	マイナーバージョンの自動マイナーアップグレード、メンテナンスウィンドウ、および開始日、時刻、および期間を指定します。	AWS SysAdmin、DBA
AWS SCT から移行前スクリプトを実行します。	ターゲット Amazon RDS for PostgreSQL インスタンスで、AWS SCT の SQL スクリプトとその他の変更を加えてデータベーススキーマを作成します。これには、ユーザー作成、データベース作成、スキーマ作成、テーブル、ビュー、関数、その他のコードオブジェクトを含む複数のスクリプトの実行が含まれる場合があります。	AWS SysAdmin、DBA

AWS DMS を使用してデータを移行

タスク	説明	必要なスキル
AWS DMSでレプリケーションインスタンスを作成します。	名前、インスタンスクラス、VPC (EC2 インスタンスと同じ)、マルチ AZ、パブリックアクセシビリティのフィールドを入力します。詳細設定セクションで、割り当てられたストレージ、サブネットグループ、アベイラビリティーゾーン、VPC セキュリティグループ、および AWS Key Management	AWS SysAdmin、DBA

タスク	説明	必要なスキル
	Service (AWS KMS) キーを指定します。	
ソースデータベースのエンドポイントを作成します。	エンドポイント名、タイプ、ソースエンジン (Oracle)、サーバー名 (EC2 インスタンスのプライベート DNS 名)、ポート、SSL モード、ユーザー名、パスワード、SID、VPC (レプリケーションインスタンス付けの VPC を指定)、およびレプリケーションインスタンスを指定します。接続をテストするには、[テストを実行] を選択し、エンドポイントを作成します。maxFileSize と numberDataTypeScale の詳細設定も行えます。	SysAdmin、DBA
AWS DMS を Amazon RDS for PostgreSQLに接続します。	PostgreSQL データベースが別の VPC にある場合は、VPC 間の接続用の移行セキュリティグループを作成します。	AWS SysAdmin、DBA

タスク	説明	必要なスキル
ターゲットデータベースエンドポイントを作成します。	エンドポイント名、タイプ、ソースエンジン (PostgreSQL)、サーバー名 (Amazon RDS エンドポイント)、ポート、SSL モード、ユーザー名、パスワード、データベース名、VPC (レプリケーションインスタンスがあり VPC を指定)、およびレプリケーションインスタンスを指定します。接続をテストするには、[テストを実行] を選択し、エンドポイントを作成します。maxFileSize と numberDataTypeScale の詳細設定も行えます。	SysAdmin、DBA
AWS DMS レプリケーションタスクを作成します。	タスク名、レプリケーションインスタンス、ソースとターゲットのエンドポイント、レプリケーションインスタンスを指定します。移行タイプには、[既存のデータを移行し、現在進行中の変更のレプリケーションを移行] をします。作成のタスクを開始のチェックボックスをクリアします。	AWS SysAdmin、DBA

タスク	説明	必要なスキル
AWS DMS レプリケーションタスク設定を構成します。	ターゲットテーブル作成モードについては、何もしないを選択します。フルロードが完了したら、タスクを停止します (プライマリキーを作成するため)。制限付き LOB モードまたはフル LOB モードを指定し、制御テーブルを有効にします。オプションで CommitRate の詳細設定を行えます。	DBA
テーブルマッピングを設定します。	テーブルマッピングセクションで、移行に含まれるすべてのスキーマのすべてのテーブルを対象にインクルードルールを作成し、次に除外ルールを作成します。スキーマ、テーブル、列の名前を小文字に変換する 3 つの変換ルールを追加し、この特定の移行に必要なその他のルールを追加します。	DBA
タスクの開始	レプリケーションタスクを開始します。全ロードが実行されていることを確認します。プライマリ Oracle データベースで、`ALTER SYSTEM SWITCH LOGFILE` を実行してタスクを開始します。	DBA

タスク	説明	必要なスキル
AWS SCT から移行中のスクリプトを実行します。	Amazon RDS for PostgreSQL では、次の `create_index.sql` および `create_constraint.sql` のスクリプトを実行します(完全なスキーマが最初に作成されていない場合)。	DBA
タスクを再開して、変更データキャプチャ (CDC) を継続します。	Amazon RDS for PostgreSQL DB で `VACUUM` インスタンスで実行し、AWS DMS タスクを再起動して、キャッシュされた CDC の変更を適用します。	DBA

PostgreSQL データベースへのカットオーバー

タスク	説明	必要なスキル
AWS DMS のログと検証テーブルを確認します。	レプリケーションエラーまたは検証エラーを確認して修正します。	DBA
オンプレミスの Oracle データベースとその依存関係の使用を停止します。	Oracle の依存関係をすべて停止し、Oracle データベースのリスナーをシャットダウンして `ALTER SYSTEM SWITCH LOGFILE` を実行します。アクティビティが表示されない場合は、AWS DMS タスクを停止します。	DBA
AWS SCT から移行後のスクリプトを実行します。	Amazon RDS for PostgreSQL では、次のスクリプトを実行します: `create_fo`	DBA

タスク	説明	必要なスキル
	reign_key_constrai nt.sql and create_tr iggers.sql シーケンス が最新であることを確認しま す。	
Amazon RDS for PostgreSQL のその他のステップを完了しま す。	必要に応じて Oracle と一致するようにシーケンスをイ ンクリメントし、VACUUM と ANALYZE を実行し、コンプ ライアンスのためのスナップ ショットを作成します。	DBA
[Amazon RDS for PostgreSQL に接続] を選択します。	Amazon RDS for PostgreSQ L から AWS DMS セキュリ ティグループを削除し、実稼 働セキュリティグループを追 加して、アプリケーションを 新しいデータベースに接続し ます。	DBA
AWS DMS オブジェクトをク リーンアップします。	エンドポイント、レプリケー ションタスク、レプリケー ションインスタンス、EC2 イ ンスタンスを削除します。	SysAdmin、DBA

関連リソース

- AWS DMS のドキュメント
- AWS SCT のドキュメント
- 「Amazon RDS for PostgreSQL 料金設定」
- 「AWS DMSのソースとして Oracle データベースを使用」
- 「AWS DMSのターゲットとして PostgreSQL データベースを使用する」

AWS DMS と AWS SCT を使用して Oracle on Amazon EC2 から Amazon RDS for MySQL に移行する

作成者: Anil Kunapareddy (AWS)、Harshad Gohil

概要

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで Oracle データベースを管理するには、リソースが必要でコストがかかります。これらのデータベースを MySQL データベースインスタンスの Amazon Relational Database Service (Amazon RDS) に移行することで、IT 予算全体を最適化して作業を軽減できます。Amazon RDS for MySQL には、マルチ AZ、スケーラビリティ、自動バックアップなどの機能も用意されています。

このパターンでは、Amazon EC2 上で実行するソース Oracle データベースをターゲット Amazon RDS for MySQL DB インスタンスに移行する手順を順番に説明します。AWS Database Migration Service (AWS DMS) を使用してデータを移行し、AWS Schema Conversion Tool (AWS SCT) を使用してソースデータベース スキーマとオブジェクトを Amazon RDS for MySQL と互換性のあるフォーマットに変換します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- インスタンスサービスとリスナーサービスが ARCHIVELOG モードで実行されているソースデータベース
- データ移行のための十分なストレージを備えた、ターゲット Amazon RDS for MySQL データベース

機能制限

- AWS DMS はターゲットデータベースにスキーマを作成しないので、ユーザーが作成する必要があります。スキーマ名がすでに存在している必要があります。ソーススキーマのテーブルは、AWS DMS がターゲットインスタンスへの接続に使用するユーザーまたはスキーマにインポートされます。複数のスキーマに移行する必要がある場合、複数のレプリケーションタスクを作成する必要があります。

製品バージョン

- バージョン 10.2 以降の11g から 12.2 まで、および 18c のすべてのエディションの Oracle データベース。サポートされているバージョンの最新リストについては、「Oracle データベースを AWS DMS のソースとして使用する」および「AWS DMS のターゲットとして MySQL 互換データベースを使用する」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。AWS SCT でサポートされている Oracle データベースのバージョンについては、「AWS SCT のドキュメント」を参照してください。
- AWS DMS は MySQL のバージョン 5.5、5.6、5.7 をサポートしています。

アーキテクチャ

ソーステクノロジースタック

- EC2 インスタンス上の Oracle データベース

ターゲットテクノロジースタック

- Amazon RDS for MySQL DB インスタンス

データ移行アーキテクチャ

ソースアーキテクチャとターゲットアーキテクチャ

ツール

- AWS DMS - AWS Database Migration Service (AWS DMS) は、オンプレミス、Amazon RDS DB インスタンス、または EC2 インスタンス上のデータベースから、Amazon RDS for MySQL や EC2 インスタンスなどの AWS サービス上のデータベースにデータを移行するために使用できるウェブサービスです。データベースを AWS Service からオンプレミスのデータベースに移行することもできます。異種または同種のデータベースエンジン間でデータを移行できます。
- AWS SCT - AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコードオブジェクト (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを簡単に移行できるようにします。AWS SCT を使用してデータベーススキーマとコードオブジェクトを変換した後、AWS DMS

を使用してソースデータベースからターゲットデータベースにデータを移行し、移行プロジェクトを完了できます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースデータベースとターゲットデータベースについて、バージョンとエンジンを特定します。		DBA/開発者
DMS レプリケーションインスタンスを特定します。		DBA/開発者
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA/開発者
レイテンシーや帯域幅などのネットワーク要件を特定します。		DBA/開発者
Oracle 互換性リストと容量要件に基づいて、ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA/開発者
ソースおよびターゲットのデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA/開発者
AWS SCT と Oracle ドライバーをインストールします。		DBA/開発者
バックアップ戦略を決定します。		DBA/開発者

タスク	説明	必要なスキル
可用性の要件を決定します。		DBA/開発者
アプリケーションの移行/切り替え戦略を特定します。		DBA/開発者
容量、ストレージ、ネットワーク機能に基づいて、適切な DB インスタンスタイプを選択します。		DBA/開発者

環境を設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。ソース、ターゲット、レプリケーションインスタンスは、同じ VPC 上に存在する必要があります。これらを同じアベイラビリティゾーンに配置するのも良いでしょう。		開発者
データベースアクセスに必要なセキュリティグループを作成します。		開発者
キーペアを生成して設定します。		開発者
サブネット、アベイラビリティゾーン、CIDR ブロックを設定します。		開発者

ソースの設定: EC2 インスタンス上の Oracle データベース

タスク	説明	必要なスキル
必要なユーザーとロールを使用して Amazon EC2 に Oracle データベースをインストールします。		DBA
EC2 インスタンスの外部から Oracle にアクセスするには、次の列で 3 つの手順を実行します。	1. `tnsnames` のローカルホストを Amazon EC2 パブリック DNS に変更します。 2. `listener` のローカルホストを Amazon EC2 パブリック DNS に変更します。 3. リスナーを停止して再起動します。	DBA
Amazon EC2 を再起動すると、パブリック DNS が変更されます。'tnsnames' と 'listener' の Amazon EC2 パブリック DNS を更新するか、Elastic IP アドレスを使用してください。		DBA/開発者
レプリケーションインスタンスと必要なクライアントがソースデータベースにアクセスできるように EC2 インスタンスのセキュリティグループを設定します。		DBA/開発者

ターゲットの設定: Amazon RDS for MySQL

タスク	説明	必要なスキル
Amazon RDS for MySQL DB インスタンスの設定と起動を行います。		開発者
Amazon RDS for MySQL DB インスタンスに必要なテーブルスペースを作成します。		DBA
レプリケーションインスタンスと必要なクライアントがターゲットデータベースにアクセスできるようにセキュリティグループを設定します。		開発者

AWS SCT を設定し、ターゲットデータベースにスキーマを作成します。

タスク	説明	必要なスキル
AWS SCT と Oracle ドライバーをインストールします。		開発者
適切なパラメータを入力して、ソースとターゲットに接続します。		開発者
スキーマ変換レポートを生成します。		開発者
コードとスキーマ、特にテーブルスペースと引用符を必要に応じて修正し、ターゲットデータベースで実行します。		開発者

タスク	説明	必要なスキル
データを移行する前に、ソースとターゲットでスキーマを検証します。		開発者

AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
フルロードおよび変更データキャプチャ (CDC) または CDC のみの場合は、追加の接続属性を設定する必要があります。		開発者
AWS DMS ソース Oracle データベース定義で指定されたユーザーには、必要なすべての権限を付与する必要があります。詳細については、https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Self-Managed を参照してください。		DBA/開発者
ソース DB での補足的なログ記録を有効にします。		DBA/開発者
フルロードおよび変更データキャプチャ (CDC) または CDC のみの場合は、ソースデータベースで ARCHIVELOG モードを有効にします。		DBA

タスク	説明	必要なスキル
ソースエンドポイントとターゲットエンドポイントを作成してテストします。		開発者
エンドポイントが正常に接続された後、レプリケーションタスクを作成します。		開発者
タスクで CDC のみ（または）フルロード + CDC を選択すると、継続的なレプリケーションのみの変更（または）フルロード + 継続的な変更がそれぞれキャプチャされます。		開発者
レプリケーションタスクを実行し、Amazon CloudWatch Logs をモニタリングします。		開発者
ソースデータベースとターゲットデータベースのデータを検証します。		開発者

アプリケーションの移行とカットオーバー

タスク	説明	必要なスキル
アプリケーション移行戦略の手順に従ってください。		DBA、開発者、アプリ所有者
アプリケーションのカットオーバー/スイッチオーバー戦略に従ってください。		DBA、開発者、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
ソースデータベースとターゲットデータベースにあるスキーマとデータを検証します。		DBA/開発者
移行の所要時間、手動とツールの比率、コスト削減などのメトリクスを収集します。		DBA/開発者/アプリ所有者
プロジェクト文書とアーティファクトをレビューして検証します。		DBA/開発者/アプリ所有者
一時的な AWS リソースをシャットダウンします。		DBA/開発者
プロジェクトを終了し、フィードバックを提供します。		DBA/開発者/アプリ所有者

関連リソース

- AWS DMS のドキュメント
- AWS DMS ウェブサイト
- AWS DMS に関するブログ投稿
- 「Oracle データベースを AWS に移行するための戦略」
- 「Amazon RDS for Oracle よくある質問」
- 「Oracle よくある質問」
- Amazon EC2
- 「Amazon EC2 よくある質問」
- 「クラウドコンピューティング環境における Oracle ソフトウェアのライセンス」

AWS DMS を使用して Oracle から Amazon DocumentDB に移行する

作成者: Sashikanta Pattanayak (AWS) と Munesh Siddappa (AWS)

概要

このパターンは、AWS Database Migration Service (AWS DMS) を使用して Oracle データベースを Amazon DocumentDB (MongoDB 互換) データベースに移行する場合のガイダンスを提供します。このアプローチは、オンプレミスの Oracle ソースデータベースだけでなく、Oracle DB インスタンスの Amazon Relational Database Service (Amazon RDS) にも適用できます。このパターンでは、例として Amazon RDS Oracle DB ソースインスタンスを使用しています。

Amazon DocumentDB (MongoDB 互換) は、フルマネージドの MongoDB 互換ドキュメントデータベースサービスで、JSON データの保存、クエリ実行、インデックス作成を容易にします。

このパターンのユースケースは、Amazon DocumentDB コレクションへの Oracle データベーステーブルの 1 対 1 リプリケーションです。このパターンでは、AWS DMS レプリケーションタスクを使用して Oracle データベースのテーブル構造を読み取り、Amazon DocumentDB に対応するコレクションを作成し、全ロード移行を実行します。MongoDB の場合と同様に、Amazon DocumentDB でもデータを表示したりクエリしたりできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Oracle データベースの使用に関する知識
- Amazon DocumentDB の使用に関する知識
- Oracle ユーザーの場合、 `SELECT ANY TABLE` 権限
- Amazon DocumentDB を使用する場合、データをダンプするために必要な権限

制約事項

AWS DMS のターゲットとして Amazon DocumentDB を使用する場合、以下の制限が適用されます：

- Amazon DocumentDB では、コレクション名にドル記号 ($) を含めることはできません。また、データベース名に Unicode 文字を含めることもできません。

- AWS DMS で複数のソーステーブルを 1 つの Amazon DocumentDB, コレクションにマージすることはできません。
- AWS DMS は、プライマリキーがないソーステーブルの変更を処理する際に、そのテーブルのラージバイナリオブジェクト (LOB) 列を無視します。
- [Change table] (変更テーブル) オプションが有効で、AWS DMS で「_id」という名前のソース列が検出されると、その列が変更テーブルに「__id」(2 つのアンダースコア) として表示されます。
- ソースエンドポイントとして Oracle を選択する場合は、Oracle ソースで完全なサプリメンタルロギングが有効になっている必要があります。有効になっていないと、ソースに変更されていない列がある場合にデータが null 値として Amazon DocumentDB にロードされます。

製品バージョン

- Amazon RDS for Oracle バージョン 12.1.0.2 以降
- AWS DMS バージョン 3.1.3 以降 (最新バージョン情報については、AWS DMS ドキュメントの「Amazon DocumentDB を DMS のターゲットとして使用する」を参照してください)

アーキテクチャ

ソーステクノロジースタック

- Amazon RDS for Oracle DB インスタンス。

ターゲットテクノロジースタック

- Amazon DocumentDB

ソースアーキテクチャとターゲットアーキテクチャ

ツール

- DMS – AWS Database Migration Service (AWS DMS) は、ソースデータストアからターゲットデータストアへのデータの移行に使用できるウェブサービスです。「DMS ユーザーガイド」には、AWS DMS での使用がサポートされている Oracle ソースデータベースのバージョンとエディションが記載されています。このパターンに関連する追加情報については、「Amazon DocumentDB を DMS のターゲットとして使用する」を参照してください。

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon DocumentDB クラスターは、デフォルトの仮想プライベートクラウド (VPC) で実行されています。Amazon DocumentDB クラスターを操作するには、Amazon DocumentDB クラスターを作成したのと同じ AWS リージョンで、EC2 インスタンスをデフォルトの VPC に起動する必要があります。詳細については、Amazon DocumentDB ドキュメントの「Amazon EC2 インスタンスを起動する」を参照してください。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		AWS 管理者
適切なインスタンスタイプ（容量、ストレージ機能、ネットワーク機能）を選択します。		AWS 管理者
ソースデータベースとターゲットデータベースのネットワーク / ホストアクセスセキュリティ要件を特定する。		AWS 管理者
ソースデータベースとターゲットデータベースへのアウトバウンドトラフィック用のセキュリティグループを作成する。		AWS 管理者
Amazon DocumentDB の EC2 インスタンスを作成して構成する。		AWS 管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
VPC とサブネットを作成する。		AWS 管理者
セキュリティグループとネットワークアクセスコントロールリスト (ACL) を作成します。		AWS 管理者
Amazon RDS for Oracle のソースインスタンスを構成して起動する。		AWS 管理者
Amazon DocumentDB インスタンスを構成して起動する。		AWS 管理者

ソースデータベースの準備

タスク	説明	必要なスキル
接続の詳細を使用して Oracle データベースに接続できることを確認する。		AWS 管理者
Oracle ユーザーに SELECT ANY TABLE 権限があることを確認する。		AWS 管理者

ターゲットデータベースの準備

タスク	説明	必要なスキル
適切なインスタンスクラスとインスタンス数を選択して		AWS 管理者

タスク	説明	必要なスキル
Amazon DocumentDB クラスターを作成する。		

Amazon EC2 を構成

タスク	説明	必要なスキル
EC2 インスタンスを構成する。	Amazon DocumentDB クラスターを操作するには、Amazon DocumentDB クラスターを作成したのと同じ AWS リージョンで、EC2 インスタンスをデフォルトの VPC に起動する必要があります。EC2 インスタンスの AWS リージョン、VPC、アベイラビリティーゾーン、サブネットを構成します。	AWS 管理者
key pair を構成します。	起動後に EC2 インスタンスに安全に接続できるようにするパブリック/プライベートキーペア。	AWS 管理者
踏み台ホストの CIDR 範囲を設定する (オプション)。	踏み台ホストインスタンスへの外部 Secure Shell (SSH) へのアクセスが許可されている CIDR IP 範囲を設定します。	AWS 管理者

データを移行する — フルロード

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。		AWS 管理者
ソースおよびターゲットエンドポイントを作成します。		AWS 管理者
全ロードの AWS DMS レプリケーションタスクを作成する。		AWS 管理者

移行のテスト

タスク	説明	必要なスキル
EC2 インスタンスを通じて Amazon DocumentDB クラスターに接続します。		AWS 管理者
mongo シェルを使用してクラスターに接続します。	手順については、「参照およびヘルプ」セクションの Amazon DocumentDB リンクを参照してください。	AWS 管理者
移行の結果を確認します。		AWS 管理者

関連リソース

- DMS の仕組み
- Amazon DocumentDB への移行
- Amazon DocumentDB を DMS のターゲットとして使用する
- Amazon DocumentDB の概要
- mongo シェルを使用して Amazon DocumentDB クラスターにアクセスして使用する

- オフライン方式を使用して MongoDB から Amazon DocumentDB に移行する (ブログ記事)
- Amazon DocumentDB (MongoDB 互換) を使用して、大規模なアプリケーションの構築と管理を行う方法 (ブログ記事)

AWS DMS と AWS SCT を使用して、Oracle データベースを Amazon EC2 から Amazon RDS for MariaDB に移行する

作成者: Veeranjaneyulu Grandhi (AWS)、vinod kumar (AWS)

概要

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の Oracle データベースを、Amazon DB インスタンスの Amazon Relational Database Service (Amazon RDS) に移行する手順を、順を追って説明します。このパターンでは、データ移行に AWS Data Migration Service (AWS DMS) を使用し、スキーマ変換に AWS Schema Conversion Tool (AWS SCT) を使用します。

EC2 インスタンスで Oracle データベースを管理することは、Amazon RDS 上のデータベースを使用するよりも多くのリソースを必要とし、コストもかかります。Amazon RDS を使用すると、クラウドでリレーショナルデータベースを簡単に設定、運用、拡張できます。Amazon RDS は、ハードウェアのプロビジョニング、データベースのセットアップ、パッチ適用、バックアップなどの時間のかかる管理タスクを自動化しながら、コスト効率が高くサイズ変更可能な容量を提供します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- インスタンスサービスとリスナーサービスが稼働しているソース Oracle データベース。このデータベースは ARCHIVELOG モードになっているはずです。
- AWS DMS のソースとしての Oracle データベースの使用に関する知識。
- AWS SCT のソースとして Oracle を使用することについての知識。

制限

- データベースサイズの上限: 64 TB

製品バージョン

- バージョン 10.2 以降の 11g から 12.2 まで、および 18c のすべての工ディションの Oracle データベース。サポートされているバージョンの最新リストについては、AWS ドキュメントの「Using an Oracle Database as a Source for AWS DMS」と「AWS SCT version table」を参照してください。

- Amazon RDS は、MariaDB サーバーコミュニティサーバーのバージョン 10.3、10.4、10.5、および 10.6 をサポートしています。サポートされているバージョンの最新リストについては、Amazon RDS ドキュメントを参照してください。

アーキテクチャ

ソーステクノロジースタック

- EC2 インスタンス上の Oracle データベース

ターゲットテクノロジースタック

- Amazon RDS for MariaDB

データ移行アーキテクチャ

ターゲット アーキテクチャ

ツール

- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベースのスキーマと大部分のデータベースコードオブジェクト (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行を予測可能にします。AWS SCT を使用してデータベーススキーマとコードオブジェクトを変換した後、AWS DMS を使用してソースデータベースからターゲットデータベースにデータを移行し、移行プロジェクトを完了できます。詳細については、AWS SCT ドキュメントの「Using Oracle as a Source for AWS SCT」を参照してください。
- AWS Database Migration Service (AWS DMS) – データベースを迅速かつ安全に AWS に移行するのに役立ちます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS は、広く普及しているほとんどの商用データベースとオープンソースデータベース間のデータ移行にご利用いただけます。AWS DMS は、Oracle から Oracle への同種移行だけでなく、Oracle や Microsoft SQL Server から Amazon Aurora へなど、異なるデータベースプラットフォーム間の異種移行もサポートしています。Oracle データベースの移行の詳細については、AWS DMS のドキュメントの「Using an Oracle Database as a Source for AWS DMS」を参照してください。

エピック

移行計画

タスク	説明	必要なスキル
バージョンとデータベースエンジンを特定します。	ソースデータベースとターゲットデータベースについて、バージョンとエンジンを特定します。	DBA、開発者
レプリケーションインスタンスを特定します。	AWS DMS レプリケーションインスタンスを特定します。	DBA、開発者
ストレージ要件を特定します。	ストレージの種類と容量を特定します。	DBA、開発者
ネットワーク要件を特定します。	ネットワークの遅延と帯域幅を特定します。	DBA、開発者
ハードウェア要件を特定します。	ソースサーバーインスタンスとターゲットサーバーインスタンスのハードウェア要件を (Oracle 互換性リストと容量要件に基づいて) 特定します。	DBA、開発者
セキュリティ要件を特定します。	ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。	DBA、開発者
ドライバーをインストールします。	最新の AWS SCT および Oracle ドライバーをインストールします。	DBA、開発者
バックアップ戦略を決定します。		DBA、開発者
可用性の要件を決定します。		DBA、開発者

タスク	説明	必要なスキル
アプリケーション移行/スイッチオーバー戦略を選択してください。		DBA、開発者
インスタンスタイプを選択します。	容量、ストレージ、ネットワーク機能に基づいて適切なインスタンスタイプを選択します。	DBA、開発者

環境を設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。	ソース、ターゲット、レプリケーションインスタンスは、同じ VPC と同じアベイラビリティゾーンにある必要があります (推奨)。	開発者
セキュリティグループを作成します。	データベースアクセスに必要なセキュリティグループを作成します。	開発者
キーペアを生成します。	キーペアを生成して設定します。	開発者
その他のリソースを設定します。	サブネット、アベイラビリティゾーン、CIDR ブロックを設定します。	開発者

ソースを設定

タスク	説明	必要なスキル
EC2 インスタンスを起動します。	手順については、Amazon EC2 のドキュメントを参照してください。	開発者
Oracle データベースをインストールします。	EC2 インスタンスに Oracle データベースを、必要なユーザーとロールとともにインストールします。	DBA
タスクの説明にあるステップに従って、EC2 インスタンスの外部から Oracle にアクセスします。	1. `tnsnames` でローカルホストを Amazon EC2 パブリック DNS に変更します。 2. `listener` のローカルホストを Amazon EC2 パブリック DNS に変更します。 3. リスナーを停止して再起動します。	DBA
Amazon EC2 パブリック DNS を更新します。	EC2 インスタンスが再起動すると、パブリック DNS が変更されます。必ず `tnsnames` および `listener` で Amazon EC2 パブリック DNS を更新するか、Elastic IP アドレスを使用してください。	DBA、開発者
EC2 インスタンスのセキュリティグループを設定します。	EC2 インスタンスのセキュリティグループを設定して、レプリケーションインスタンスと必要なクライアントがソースデータベースにアクセスできるようにします。	DBA、開発者

ターゲット Amazon RDS for MariaDB 環境を設定します

タスク	説明	必要なスキル
RDS DB インスタンスを起動します。	Amazon RDS for MariaDB インスタンスを設定し、起動します。	開発者
テーブルスペースを作成します。	Amazon RDS MariaDB データベースに必要なテーブルスペースをすべて作成します。	DBA
セキュリティグループを設定します。	レプリケーションインスタンスと必要なクライアントがターゲットデータベースにアクセスできるようにセキュリティグループを設定します。	開発者

AWS SCT を設定

タスク	説明	必要なスキル
ドライバーをインストールします。	最新の AWS SCT および Oracle ドライバーをインストールします。	開発者
接続。	適切なパラメータを入力し、ソースとターゲットに接続します。	開発者
スキーマ変換レポートを生成します。	AWS SCT スキーマ変換レポートを生成します。	開発者
必要に応じてコードとスキーマを修正してください。	コードとスキーマ (特にテーブルスペースと引用符) を必要に応じて修正します。	DBA、開発者

タスク	説明	必要なスキル
スキーマを検証します。	データをロードする前に、ソースとターゲットのスキーマを検証します。	開発者

AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
接続属性を設定します。	フルロードおよび変更データキャプチャ (CDC) の場合、または CDC のみの場合は、追加の接続属性を設定します。詳細については、「Amazon RDS ドキュメント」を参照してください。	開発者
補足的なログを有効にします。	ソースデータベースでの補足的なログ記録を有効にします。	DBA、開発者
アーカイブログモードを有効にします。	フルロードと CDC (または CDC のみ) の場合は、ソースデータベースのアーカイブログモードを有効にします。	DBA
エンドポイントを作成してテストします。	ソースエンドポイントとターゲットエンドポイントを作成し、接続をテストします。詳細については、Amazon DMS のドキュメントを参照してください。	開発者
レプリケーションタスクを作成します。	エンドポイントが正常に接続された後、レプリケーションタスクを作成します。詳細に	開発者

タスク	説明	必要なスキル
	ついては、Amazon DMS のドキュメントを参照してください。	
[レプリケーション] タイプを選択します。	タスクで [CDC のみ] または [フルロードと CDC] を選択して、それぞれ連続レプリケーションのみ、またはフルロードと継続的な変更の変更をキャプチャします。	開発者
タスクを開始して監視します。	レプリケーションタスクを開始し、Amazon CloudWatch Logs をモニタリングします。詳細については、Amazon DMS のドキュメントを参照してください。	開発者
データを検証します。	ソースデータベースとターゲットデータベースのデータを検証します。	開発者

アプリケーションを移行し、ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
選択したアプリケーション移行戦略に従ってください。		DBA、アプリ所有者、デベロッパー
選択したアプリケーションのカットオーバー/スイッチオーバー戦略に従ってください。		DBA、アプリ所有者、デベロッパー

プロジェクトを閉じる

タスク	説明	必要なスキル
スキーマとデータを検証します。	プロジェクトを終了する前に、スキーマとデータがソースとターゲットで正常に検証されていることを確認します。	DBA、開発者
メトリクスを収集します。	移行時間、手動タスクとツールタスクの比率、コスト削減などの基準に関する指標を収集します。	DBA、アプリ所有者、デベロッパー
ドキュメントを確認します。	プロジェクト文書とアーティファクトをレビューして検証します。	DBA、アプリ所有者、デベロッパー
リソースをシャットダウンします。	一時的な AWS リソースをシャットダウンします。	DBA、開発者
プロジェクトを閉じます。	移行プロジェクトを閉じて、フィードバックを送ってください。	DBA、アプリ所有者、デベロッパー

関連リソース

- MariaDB Amazon RDS の概要
- Amazon RDS for MariaDB の特徴
- 「AWS DMSのソースとして Oracle データベースを使用」
- Oracle データベースを AWS に移行するための戦略
- Licensing Oracle Software in the Cloud Computing Environment
- Amazon RDS for Oracle FAQs
- AWS DMS の概要
- AWS DMS に関するブログ投稿

- Amazon EC2 の概要
- Amazon EC2 よくある質問
- AWS SCT のドキュメント

AWS DMS と AWS SCT を使用してオンプレミスの Oracle データベースを Amazon RDS for MySQL に移行する

作成者: Sergey Dmitriev (AWS) と Naresh Damera (AWS)

概要

このパターンでは、オンプレミスの Oracle データベースを Amazon Relational Database Service (Amazon RDS) for MySQL DB インスタンスに移行する手順を説明します。AWS Database Migration Service (AWS DMS) を使用してデータを移行し、AWS Schema Conversion Tool (AWS SCT) を使用してソースデータベーススキーマとオブジェクトを Amazon RDS for MySQL と互換性のあるフォーマットに変換します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターにあるソース Oracle データベース

制限

- データベースサイズの上限: 64 TB

製品バージョン

- バージョン 11g (バージョン 11.2.0.3.v1 以降) から 12.2 まで、および 18c のすべてのエディションの Oracle データベース。サポートされているバージョンの最新リストについては、「Using an Oracle Database as a Source for AWS DMS」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。AWS SCT でサポートされている Oracle データベースのバージョンについては、AWS SCT のドキュメントを参照してください。
- AWS DMS は現在、MySQL のバージョン 5.5、5.6、5.7 をサポートしています。サポートされているバージョンの最新リストについては、AWS ドキュメントの「Using a MySQL-Compatible Database as a Target for AWS DMS」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース

ターゲットテクノロジースタック

- Amazon RDS for MySQL DB インスタンス

データ移行アーキテクチャ

ツール

- AWS DMS - AWS Database Migration Service (AWS DMS) は、リレーショナルデータベース、データウェアハウス、NoSQL データベース、その他データストアの移行を促します。AWS DMS を使用して、オンプレミスのインスタンス間 (AWS クラウドセットアップを使用)、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間で、AWS クラウドにデータを移行できます。
- AWS SCT - AWS Schema Conversion Tool (AWS SCT) は、データベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換するために使用されます。ツールによって変換されるカスタムコードには、ビュー、ストアドプロシージャ、関数が含まれます。ツールで自動的に変換されないコードは明確にマークされるので、ユーザーが手動で変換できます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を特定する。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定する。		DBA、SysAdmin

タスク	説明	必要なスキル
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、SysAdmin
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。		DBA、SysAdmin
アプリケーション移行戦略を特定します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		SysAdmin
セキュリティグループとネットワークアクセスコントロールリスト (ACL)を作成する。		SysAdmin
Amazon RDS DB インスタンスを設定し、起動する。		DBA、SysAdmin

データを移行する

タスク	説明	必要なスキル
AWS SCT を使用してデータベーススキーマを移行する。		DBA

タスク	説明	必要なスキル
AWS DMS を使用してデータを移行する。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
AWS SCT を使用して、アプリケーションコード内の SQL コードを分析して変換する。	詳細については、https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Converting.App.html を参照してください。	アプリ所有者
アプリケーション移行戦略に従う。		DBA、SysAdmin、アプリ所有者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin

タスク	説明	必要なスキル
プロジェクト文書を確認して検証する。		DBA、SysAdmin
移行までの時間、手動とツールの比率、コスト削減などの指標を収集する。		DBA、SysAdmin
プロジェクトを終了し、フィードバックを提供します。		

関連リソース

リファレンス

- 「AWS DMS のドキュメント」
- AWS SCT のドキュメント
- Amazon RDS の料金

チュートリアルと動画

- AWS DMS の使用開始
- Amazon RDS の使用開始
- DMS (ビデオ)
- Amazon RDS (ビデオ)

Oracle バイスタンダーと AWS DMS を使用して、オンプレミスの Oracle データベースを Amazon RDS for PostgreSQL に移行する

作成者: Cady Motyka (AWS)

概要

このパターンでは、最小限のダウンタイムでオンプレミスの Oracle データベースを以下に示す PostgreSQL 互換の AWS データベースサービスのいずれかに移行する方法を説明します。

- Amazon Relational Database Service (Amazon RDS) for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション

このソリューションでは、AWS Database Migration Service (AWS DMS) を使用してデータを移行し、AWS Schema Conversion Tool (AWS SCT) を使用してデータベーススキーマを変換し、Oracle バイスタンダーデータベースを使用して移行管理を支援します。この実装では、ダウンタイムはデータベース上のすべての外部キーを作成または検証するのに必要な時間に制限されます。

このソリューションでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスと Oracle バイスタンダーデータベースを使用すると、AWS DMS 経由のデータストリームを制御できます。オンプレミスの Oracle データベースから Oracle バイスタンダーへのストリーミングレプリケーションを一時的に停止し、AWS DMS をアクティブにしてデータ検証を把握したり、別のデータ検証ツールを使用したりできます。Amazon RDS for PostgreSQL DB インスタンスまたは Aurora PostgreSQL 互換 DB インスタンス、およびバイスタンダーデータベースは、AWS DMS が現在の変更移行を完了すると同じデータを保持します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミス型データセンターのソース Oracle データベースと Active Data Guard スタンバイデータベースが設定済み
- データベースシークレットを保存するオンプレミス型データセンターと AWS Secrets Manager の間で AWS Direct Connect が設定済み
- AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバーが、ローカルマシンまたは AWS SCT がインストールされている EC2 インスタンスにインストール済み

- Oracle データベースを AWS DMS のソースとして使用することに精通している
- Oracle データベースを AWS DMS のターゲットとして使用することに精通している

機能制限

- データベースサイズの上限: 64 TB

製品バージョン

- AWS DMS は、バージョン 10.2 以降 (バージョン 10.x の場合)、11g から 12.2、18c、19c までのすべての Oracle データベースエディションに対応しています。サポートされているバージョンの最新リストについては、「Oracle データベースを AWS DMS のソースとして使用する」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。AWS SCT でサポートされている Oracle データベースのバージョンについては、「AWS SCT のドキュメント」を参照してください。
- AWS DMS は、PostgreSQL バージョン 9.4 以降 (バージョン 9.x の場合)、10.x、11.x、12.x、13.x をサポートしています。最新情報については、AWS のドキュメント「AWS DMS のターゲットとして PostgreSQLデータベースを使用する」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース
- Oracle データベースのバイスタンダーを保持する EC2 インスタンス

ターゲットテクノロジースタック

- Amazon RDS for PostgreSQL または Aurora PostgreSQL インスタンス、PostgreSQL 9.3 以降

ターゲット アーキテクチャ

次の図は、AWS DMS および Oracle オブザーバを使用して Oracle バイスタンダ–を PostgreSQL 互換の AWS データベースに移行するワークフローの例を示しています。

ツール

- 「AWS Database Migration Service (AWS DMS)」を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベースのスキーマおよびカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。

エピック

Oracle データベースのスキーマを PostgreSQL に変換する

タスク	説明	必要なスキル
AWS SCT を設定する	新しいレポートを作成し、ソースとして Oracle に接続し、ターゲットとして PostgreSQL に接続します。[プロジェクト設定] で [SQL スクリプト作成] タブに移動します。ターゲット SQL スクリプトを複数のファイルに変更します。これらのファイルは後で使用され、次のような名前が付けられます。 • create_database.sql • create_sequence.sql • create_table.sql • create_view.sql • create_function.sql	DBA
Oracle データベースのスキーマを変換する	[アクション] タブで [レポートを生成] を選択します。次に、[スキーマの変換] を選択し、	DBA

タスク	説明	必要なスキル
	[SQL として保存] を選択します。	
スクリプトの変更	たとえば、ソーススキーマの数値が PostgreSQL で数値形式に変換されたが、パフォーマンスを向上させるために代わりに BIGINT を使用したい場合は、スクリプトを変更することができます。	DBA

Amazon RDS DB インスタンスを作成して設定する

タスク	説明	必要なスキル
Amazon RDS DB インスタンスを作成します。	適切な AWS リージョンで、新しい PostgreSQL DB インスタンスを作成します。詳細については、Amazon RDS ドキュメントの「PostgreSQL DB インスタンスを作成して PostgreSQL インスタンス上のデータベースに接続する」を参照してください。	SysAdmin、DBA
DB インスタンスの仕様を設定します。	DB エンジンのバージョン、DB インスタンスクラス、マルチ AZ 配置、ストレージタイプ、ストレージの割り当てを指定します。DB インスタンス識別子、プライマリ ユーザー名、プライマリ パスワードを入力します。	SysAdmin、DBA

タスク	説明	必要なスキル
ネットワークとセキュリティを設定します。	仮想プライベートクラウド (VPC)、サブネットグループ、パブリックアクセシビリティ、アベイラビリティーゾーンの設定、およびセキュリティグループを指定します。	DBA、SysAdmin
データベースオプションを設定します。	データベース名、ポート、パラメータグループ、暗号化、KMS キーを指定します。	SysAdmin、DBA
バックアップを設定します。	バックアップ保持期間、バックアップウィンドウ、開始時刻、期間、およびタグをスナップショットにコピーするかどうかを指定します。	SysAdmin、DBA
モニタリングオプションを設定します。	強化されたモニタリングとパフォーマンスインサイトを有効または無効にします。	SysAdmin、DBA
メンテナンスオプションを設定します。	マイナーバージョンの自動アップグレード、メンテナンスウィンドウ、開始日、時刻、期間を指定します。	SysAdmin、DBA

タスク	説明	必要なスキル
AWS SCT から移行前スクリプトを実行します。	Amazon RDS インスタンスで、AWS SCT が生成した以下のスクリプトを実行します。 • create_database.sql • create_sequence.sql • create_table.sql • create_view.sql • create_function.sql	SysAdmin、DBA

Amazon EC2 で Oracle バイスタンダーを設定する

タスク	説明	必要なスキル
Amazon EC2 のネットワークを設定します。	新しい VPC、サブネット、インターネットゲートウェイ、ルートテーブル、およびセキュリティグループを作成します。	AWS SysAdmin
EC2 インスタンスを作成します。	適切な AWS リージョンで、新しい EC2 インスタンスを作成します。Amazon マシンイメージ (AMI) を選択し、インスタンスサイズを選択し、インスタンスの詳細 (インスタンス数 (1)、前のタスクで作成した VPC とサブネット、パブリック IP の自動割り当て、その他のオプションを設定します。ストレージを追加し、セキュリティグループを設定して起動します。プロンプトが	AWS SysAdmin

タスク	説明	必要なスキル
	表示された後、次のステップのためにキーペアを作成して保存します。	
Oracle ソースデータベースを EC2 インスタンスに接続します。	IPv4 パブリック IP アドレスと DNS をテキストファイルにコピーし、次のように SSH コマンドを使用して接続します。ssh -i "your_file.pem" ec2-user@<your-IP-address-or-public-DNS>	AWS SysAdmin
Amazon EC2 でバイスタンダー向けの初期ホストを設定します。	SSH キー、Bash プロファイル、ORATAB、シンボリックリンクを設定します。Oracle ディレクトリを作成します。	AWS SysAdmin、Linux 管理者
Amazon EC2 でバイスタンダー向けのデータベースを設定します。	RMAN を使用してデータベースコピーを作成し、補足ロギングを有効にして、スタンバイコントロール ファイルを作成します。コピーの完了後、データベースを復旧モードに移行します。	SysAdmin、DBA

タスク	説明	必要なスキル
Oracle Data Guard を設定します。	listener.ora ファイルを変更し、リスナーを起動します。新規アーカイブの宛先を設定します。バイスタンダ–を復旧モードにし、一時ファイルを置き換えて将来の破損を防ぎ、必要に応じてアーカイブディレクトリの空き容量が不足しないように crontab をインストールし、ソースとスタンバイの manage-trclog-files-oracle.cfg ファイルを編集します。	SysAdmin、DBA
配送を同期するため、Oracle データベースの準備を整えます。	スタンバイ ログファイルを追加し、復旧モードを変更します。ソースプライマリとソーススタンバイの両方で、ログ配信を SYNC AFFIRM に変更します。ログをプライマリに切り替え、Amazon EC2 バイスタンダ–のアラートログで、スタンバイのログファイルを使用していることを確認し、REDO ストリームの配信が同期されていることを確認します。	SysAdmin、DBA

AWS DMS でデータを移行する

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	名前、インスタンスクラス、VPC (Amazon EC2 イ	SysAdmin、DBA

タスク	説明	必要なスキル
	ンスタンスと同様)、マルチ AZ、パブリックアクセシビリティのフィールドに値を入力します。[詳細設定] で割り当てられたストレージ、サブネットグループ、アベイラビリティーゾーン、VPC セキュリティグループ、および AWS Key Management Service (AWS KMS) キーを指定します。	
ソースデータベースのエンドポイントを作成します。	エンドポイント名、タイプ、ソースエンジン (Oracle)、サーバー名 (Amazon EC2 プライベート DNS 名)、ポート、SSL モード、ユーザー名、パスワード、SID、VPC (レプリケーションインスタンスのある VPC を指定)、およびレプリケーションインスタンスを指定します。接続をテストするには、 [テストを実行] を選択し、エンドポイントを作成します。maxFileSize と numberDataTypeScale の詳細設定も行えます。	SysAdmin、DBA
[Amazon RDS for PostgreSQL に接続] を選択します。	VPC 間で接続するための移行用セキュリティグループを作成します。	SysAdmin、DBA

タスク	説明	必要なスキル
ターゲットデータベースエンドポイントを作成します。	エンドポイント名、タイプ、ソースエンジン (PostgreSQL)、サーバー名 (Amazon RDS エンドポイント)、ポート、SSL モード、ユーザー名、パスワード、データベース名、VPC (レプリケーションインスタンスがあり VPC を指定)、およびレプリケーションインスタンスを指定します。接続をテストするには、 [テストを実行] を選択し、エンドポイントを作成します。maxFileSize と numberDataTypeScale の詳細設定も行えます。	SysAdmin、DBA
AWS DMS レプリケーションタスクを作成します。	タスク名、レプリケーションインスタンス、ソースとターゲットのエンドポイント、レプリケーションインスタンスを指定します。移行タイプで、[既存のデータを移行し、継続的な変更をレプリケートする] を選択します。[作成時にタスクを開始] チェックボックスをオフにします。	SysAdmin、DBA

タスク	説明	必要なスキル
AWS DMS レプリケーションタスク設定を構成します。	ターゲットテーブル作成モードでは、何もしないを選択します。フルロードの完了後、(プライマリキーを作成するため) タスクを停止します。制限付き LOB モードまたはフル LOB モードを指定し、コントロールテーブルを有効にします。オプションで、[CommitRate] の詳細設定を行えます。	DBA
テーブルマッピングを設定します。	テーブルマッピングセクションで、移行に含まれる全スキーマのテーブルを対象とする 含める ルールを作成してから、除外 ルールを作成します。スキーマ、テーブル、列の名前を小文字に変換する 3 つの変換ルールを追加し、特定の移行に必要なその他のルールを追加します。	DBA
タスクの開始	レプリケーションタスクを開始します。全ロードが実行されていることを確認します。プライマリ Oracle データベースで、ALTER SYSTEM SWITCH LOGFILE を実行してタスクを開始します。	DBA

タスク	説明	必要なスキル
AWS SCT から移行中のスクリプトを実行します。	Amazon RDS for PostgreSQL で、AWS SCT が生成した以下のスクリプトを実行します。 • create_index.sql • create_constraint.sql	DBA
タスクを再開して、変更データキャプチャ (CDC) を継続します。	Amazon RDS for PostgreSQL DB インスタンスで VACUUM を実行し、AWS DMS タスクを再起動して、キャッシュされた CDC の変更を適用します。	DBA

PostgreSQL データベースへのカットオーバー

タスク	説明	必要なスキル
AWS DMS のログと検証テーブルにエラーがないことを確認してください。	レプリケーションエラーまたは検証エラーを確認して修正します。	DBA
Oracle の依存関係をすべて停止します。	Oracle の依存関係をすべて停止し、Oracle データベースのリスナーをシャットダウンして、ALTER SYSTEM SWITCH LOGFILE を実行します。アクティビティが表示されない場合は、AWS DMS タスクを停止します。	DBA
AWS SCT から移行後のスクリプトを実行します。	Amazon RDS for PostgreSQL で、AWS SCT が生成した	DBA

タスク	説明	必要なスキル
	以下のスクリプトを実行します。 • create_foreign_key_constraint.sql • create_triggers.sql	
Amazon RDS for PostgreSQL の追加手順を完了します。	必要に応じて、Oracle と一致するようにシーケンスをインクリメントし、VACUUM と ANALYZE を実行して、コンプライアンスのためのスナップショットを作成します。	DBA
[Amazon RDS for PostgreSQL に接続] を選択します。	Amazon RDS for PostgreSQL から AWS DMS セキュリティグループを削除し、実稼働セキュリティグループを追加して、アプリケーションを新しいデータベースに接続します。	DBA
AWS DMS オブジェクトをクリーンアップします。	エンドポイント、レプリケーションタスク、レプリケーションインスタンス、EC2 インスタンスを削除します。	SysAdmin、DBA

関連リソース

- AWS DMS のドキュメント
- AWS SCT のドキュメント
- 「Amazon RDS for PostgreSQL の料金」

Oracle GoldenGate を使用して Oracle Database から Amazon RDS for PostgreSQL に移行

作成者: Dhairya Jindani (AWS)、Rajeshkumar Sabankar (AWS)、及び Sindhusha Paturu (AWS)

概要

このパターンでは、Oracle Cloud Infrastructure (OCI) GoldenGate を使って、 Oracle データベースを PostgreSQL の Amazon Relational Database Service (Amazon RDS) に移行する方法を示します。

Oracle GoldenGate を使用して、最小限のダウンタイムで、ソースデータベースと 1 つ以上のデスティネーションデータベース間でデータを複製できます。

> **Note**
>
> ソース Oracle データベースは、オンプレミスでも Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでもかまいません。オンプレミスレプリケーションツールを使用する場合も、同様の手順を使用できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Oracle GoldenGate ライセンス
- PostgreSQL データベースに接続するための Java データベース接続 (JDBC) ドライバー
- ターゲットの Amazon RDS for PostgreSQL データベースで、「AWS Schema Conversion Tool (AWS SCT)」 を使用して作成されたスキーマとテーブル

制約事項

- Oracle GoldenGate では、既存のテーブルデータ (初期ロード) と継続的な変更 (変更データキャプチャ) のみのレプリケートが可能

製品バージョン

- Oracle Database Enterprise Edition 10g またはそれ以降のバージョン

- Oracleまたはそれ以降のバージョン用のOracle GoldenGate 12.2.0.1.1
- PostgreSQL またはより新しいバージョン用の Oracle GoldenGate 12.2.0.1.1

アーキテクチャ

次の図表は、Oracle GoldenGate を使用して Oracle データベースを Amazon RDS for PostgreSQL に移行するためのワークフローの例を示しています：

この図表は、次のワークフローを示しています：

1. Oracle GoldenGate 「抽出プロセス」 はソースデータベースに対して実行され、データを抽出します。
2. Oracle GoldenGate 「Replicat プロセス」は、抽出されたデータをターゲットの Amazon RDS for PostgreSQL データベースに送信します。

ツール

- 「Oracle GoldenGate」は、Oracle Cloud Infrastructureにおけるデータレプリケーションおよびストリームデータ処理ソリューションの設計、実行、調整、監視を支援します。
- PostrgreSQLのAmazon Relational Database Service (Amazon RDS) は、AWS クラウドの リレーショナルデータベース (DB) をセットアップ、運用、スケーリングを行います。

エピック

Oracle GoldenGate をダウンロードしてインストール

タスク	説明	必要なスキル
Oracle GoldenGate をダウンロードします。	Oracle GoldenGate の次のバージョンをダウンロードします： • Oracle またはより新しいバージョン用 Oracle GoldenGate 12.2.0.1.1	DBA

タスク	説明	必要なスキル
	• PostgreSQL またはより新しいバージョン用 Oracle GoldenGate 12.2.0.1.1 ソフトウェアをダウンロードするには、Oracleのウェブサイトの「Oracle GoldenGate ダウンロード」 を参照してください。	
Oracle GoldenGate for Oracle を、ソースのOracle データベースサーバーにインストールします。	手順については、「Oracle ドキュメント」を参照してください。	DBA
Amazon EC2 インスタンスに Oracle GoldenGate for PostgreSQL データベースをインストールします。	手順については、「Oracle ドキュメント」を参照してください。	DBA

ソースおよびターゲットのデータベースで Oracle GoldenGate を設定

タスク	説明	必要なスキル
ソースデータベースに Oracle データベース用 Oracle GoldenGate を設定します。	手順については、「Oracleドキュメント」を参照してください。 次のことを必ず設定してください： • サプリメンタルロギング • Oracle GoldenGate ユーザー • 必要な権限と権限	DBA

タスク	説明	必要なスキル
	• パラメータファイル • マネージャープロセス • ディレクトリ • グローバル ファイル • Oracle ウォレット	
ターゲットデータベースに Oracle GoldenGate for PostgreSQL を設定します。	手順については、Oracle のウェブサイトの「Part VI Using Oracle GoldenGate for PostgreSQL」を参照してください。 次のことを必ず設定してください： • マネージャープロセス • グローバル ファイル • Oracle ウォレット	DBA

データキャプチャの設定

タスク	説明	必要なスキル
ソースデータベースに Extract プロセスを設定します。	ソース Oracle データベースで、データを抽出するための抽出ファイルを作成します。 手順については、Oracle ドキュメンテーションの「抽出の追加」を参照してください。	DBA

タスク	説明	必要なスキル
	Note 抽出ファイルには、抽出パラメータファイルと証跡ファイルディレクトリの作成が含まれます。	
データポンプを設定して、トレイルファイルをソースからターゲットデータベースに転送します。	Oracle ウェブサイトのデータベースユーティリティで「PARFILE」の手順に従って、EXTRACT パラメータファイルとトレイルファイルディレクトリを作成します。 詳細については、Oracleのウェブサイトの「Oracle GoldenGateを理解するフュージョンミドルウェア」の「トレイルとは？」を参照してください。	DBA

タスク	説明	必要なスキル
Amazon EC2 インスタンスでレプリケーションを設定します。	レプリケーションパラメータファイルとトレイルファイルディレクトリを作成します。 レプリケーションパラメータファイルの作成についての詳細は、Oracle Databaseドキュメントのセクション「3.5 パラメータファイルの検証」を参照してください。 証跡ファイルディレクトリの作成についての詳細は、Oracle クラウド ドキュメントの「証跡の作成」を参照してください。 **Important** ターゲットの GLOBALS ファイルにチェックポイントテーブルエントリを追加してください。 詳細については、Oracle ウェブサイトの「フュージョンミドルウェア Oracle GoldenGateを理解する」にある「レプリケーションとは何か？」を参照してください。	DBA

データレプリケーションを設定

タスク	説明	必要なスキル
ソースデータベースで、初期ロード用のデータを抽出するパラメータファイルを作成します。	Oracle Cloud ドキュメントの「GGSCI でのパラメータファイルの作成」の手順に従います。 **⚠ Important** Manager がターゲットで実行されていることを確認します。	DBA
ターゲットデータベースにパラメータファイルを作成し、初期ロードのデータをレプリケーションします。	Oracle Cloud ドキュメントの「GGSCI でのパラメータファイルの作成」の手順に従います。 **⚠ Important** レプリカプロセスを追加および開始することを確認します。	DBA

Amazon RDS for PostgreSQL データベースへのへのカットオーバー

タスク	説明	必要なスキル
レプリケーションプロセスを停止し、ソースデータベースとターゲットデータベースが同期していることを確認します。	ソースデータベースとターゲットデータベースの行数を比較して、データ複製が成功したことを確認します。	DBA

タスク	説明	必要なスキル
データ定義言語 (DDL) サポートを設定します。	DDL スクリプトを実行して、PostgreSQL でトリガー、シーケンス、シノニム、参照キーを作成します。 **Note** 任意の標準 SQL クライアントアプリケーションを使用して、DB クラスター内のデータベースに接続できます。たとえば、「pgAdmin」 を使用して DB インスタンスに接続できます。	DBA

関連リソース

- 「Amazon RDS for PostgreSQL」(Amazon RDS ユーザーガイド)
- 「Amazon EC2 ドキュメント」
- 「Oracle GoldenGate に適用される処理メソッドとデータベース」(Oracle ドキュメント)

AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する

作成者: Piyush Goyal (AWS) と Brian motzer (AWS)

概要

このパターンは、AWS Database Migration Service (AWS DMS) (AWS DMS) と AWS Schema Conversion Tool (AWS SCT) を使用して Oracle データベースをAmazon Web Services (AWS) クラウド内の Amazon Redshift クラウドデータウェアハウスに移行するためのガイダンスを提供します。このパターンは、オンプレミスまたは Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにインストールされているソース Oracle データベースを対象としています。Oracle Amazon Relational Database Service (Amazon RDS) についても説明します。

前提条件と制限

前提条件

- オンプレミスのデータセンターまたは AWS クラウドで実行されている Oracle データベース
- アクティブな AWS アカウント
- Oracle データベースを 「AWS DMS のソースとして使用する」ことに精通していること
- 「Amazon Redshift データベースを AWS DMS のターゲット」として使用することについての知識
- Amazon RDS、Amazon Redshift、該当するデータベーステクノロジー、および SQL に関する知識
- AWS SCT がインストールされている AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバー

製品バージョン

- セルフ マネージド 型 Oracle データベースの場合、AWS DMS ではバージョン 10.2 以降(バージョン 10.x の場合)、11g から 12.2、18c、19c までのすべての Oracle データベースエディションに対応しています。 AWS が 管理する Amazon RDS for Oracle Database の場合、 AWS DMS はバージョン 11g (バージョン 11.2.0.4 以降) および 12.2、18c、19c までのすべての Oracle データベースエディションに対応しています。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。

アーキテクチャ

ソーステクノロジースタック

次のいずれかです:

- オンプレミスの Oracle データベース
- EC2 インスタンス上の Oracle データベース
- Amazon RDS for Oracle DB インスタンス。

ターゲットテクノロジースタック

- Amazon Redshift

ターゲットアーキテクチャ

AWS クラウドで実行されている Oracle データベースから Amazon Redshift へ:

オンプレミスのデータセンターで実行されている Oracle データベースから Amazon Redshift へ:

ツール

- 「AWS DMS」- AWS データ移行サービス (AWS DMS) は、データベースを AWS に迅速かつ安全に移行するのに役立ちます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。AWS DMS は、広く普及しているほとんどの商用データベースとオープンソースデータベース間のデータ移行にご利用いただけます。
- 「AWS SCT」- AWS Schema Conversion Tool (AWS SCT) を使用して、既存のデータベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換できます。Oracle、SQL Server、PostgresSQL など、さまざまなデータベースエンジンをソースとしてサポートしています。

エピック

移行の準備をする

タスク	説明	必要なスキル
データベースのバージョンを検証します。	ソースとターゲットのデータベースバージョンを検証し、それらが AWS DMS でサポートされていることを確認します。サポートされている Oracle データベースのバージョンについては、「Oracle データベースを AWS DMS のソースとしての使用」を参照してください。ターゲットとして Amazon Redshift を使用する方法については、「AWS DMS のターゲットとしての Amazon Redshift データベースの使用」を参照してください。	DBA
VPC とセキュリティグループを作成します。	AWS アカウントに存在しない場合、仮想プライベートクラウド (VPC) を作成します。ソースデータベースとターゲットデータベースへのアウトバウンドトラフィック用のセキュリティグループを作成します。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメント を参照してください。	システム管理者
AWS SCT をインストールします。	AWS SCT の最新バージョンと対応するドライバーをダウンロードしてインストールし	DBA

タスク	説明	必要なスキル
	ます。詳細については、「AWS SCTのインストール、検証、及び更新」を参照してください。	
AWS DMS タスクのユーザーを作成します。	ソースデータベースに AWS DMS ユーザーを作成し、それに READ 権限を付与します。このユーザーは AWS SCT と AWS DMS の両方で使用されます。	DBA
DB 接続をテストします。	Oracle DB インスタンスへの接続をテストします。	DBA
AWS SCT で新しいプロジェクトを作成します。	AWS SCT ツールを開き、新しいプロジェクトを作成します。	DBA
移行する Oracle スキーマを分析します。	AWS SCT を使用して移行するスキーマを分析し、データベース移行評価レポートを生成します。詳細については、AWS SCT ドキュメントの「データベース移行評価レポートの作成」を参照してください。	DBA
評価レポートをレビューします。	移行の可否についてレポートを確認します。一部の DB オブジェクトが手動の変換が必要です。レポートの詳細については、AWS SCT ドキュメントの「評価レポートの表示」を参照してください。	DBA

ターゲットデータベースの準備

タスク	説明	必要なスキル
Amazon Redshift クラスターを作成します。	先程作成した VPC 内に Amazon Redshift クラスターを作成します。詳細については、Amazon Redshift ドキュメントの「Amazon Redshift クラスター」を参照してください。	DBA
データベースユーザーを作成する。	Oracle ソースデータベースからユーザ、ロール、権限のリストを抽出します。ターゲット Amazon Redshift データベースにユーザーを作成し、前のステップのロールを適用します。	DBA
データベースのパラメータを評価します。	Oracle ソースデータベースのデータベースオプション、パラメータ、ネットワークファイル、およびデータベースリンクを確認し、ターゲットへの適用性を評価します。	DBA
関連する設定をターゲットに適用します。	このステップの詳細については、Amazon Redshift ドキュメントの「設定リファレンス」を参照してください。	DBA

ターゲットデータベースにオブジェクトを作成する

タスク	説明	必要なスキル
ターゲットデータベースに AWS DMS ユーザーを作成します。	ターゲットデータベースに AWS DMS ユーザーを作成し、読み取り権限と書き込み権限を付与します。AWS SCT からの接続を検証します。	DBA
スキーマを変換し、SQL レポートを確認して、エラーや警告を保存します。	詳細については、AWS SCT ドキュメントの「AWS SCT を使用したデータベーススキーマの変換」を参照してください。	DBA
スキーマの変更をターゲットデータベースに適用するか、.sql ファイルとして保存します。	手順については、AWS SCT ドキュメントの「AWS SCT の「変換されたスキーマの保存と適用」を参照してください。	DBA
ターゲットデータベースのオブジェクトを検証します。	前のステップで作成されたオブジェクトをターゲットデータベースで検証します。正常に変換されなかったオブジェクトはすべて書き直すか、再設計します。	DBA
外部キーとトリガーを無効にします。	外部キーとトリガーをすべて無効にします。これにより、AWS DMS の実行時に全ロードプロセス中にデータロードの問題が発生する可能性があります。	DBA

AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	AWS マネジメントコンソールにサインインし、AWS DMS コンソールを開きます。ナビゲーションペインで、レプリケーション インスタンスを選択し、次にレプリケーションインスタンスを作成します。詳細な手順については、AWS DMS ドキュメントの AWS DMS の使用開始の「ステップ 1」を参照してください。	DBA
ソースおよびターゲットエンドポイントを作成します。	ソースエンドポイントとターゲットエンドポイントを作成し、レプリケーションインスタンスからソースエンドポイントとターゲットエンドポイントの両方への接続をテストします。詳細な手順については、AWS DMS ドキュメントの AWS DMS の使用開始の「ステップ 2」 を参照してください。	DBA
レプリケーションタスクを作成します。	レプリケーションタスクを作成し、適切な移行方法を選択します。詳細な手順については、AWS DMS ドキュメントの AWS DMS の使用開始の「ステップ 3」を参照してください。	DBA

タスク	説明	必要なスキル
データレプリケーションを開始する。	レプリケーションタスクを開始し、ログにエラーがないか監視します。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーションサーバーを作成します。	AWS に新しいアプリケーションサーバーを作成します。	アプリ所有者
アプリケーションコードを移行します。	アプリケーションコードを新しいサーバーに移行します。	アプリ所有者
アプリケーションサーバーを設定します。	ターゲットデータベースとドライバー用にアプリケーションサーバーを設定します。	アプリ所有者
アプリケーションコードを最適化します。	ターゲットエンジンに合わせてアプリケーションコードを最適化します。	アプリ所有者

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
ユーザーを検証。	ターゲット Amazon Redshift データベースで、ユーザーを検証し、ロールと権限を付与します。	DBA
アプリケーションがロックされていることを確認します。	今後変更されないように、アプリケーションがロックされていることを確認します。	アプリ所有者

タスク	説明	必要なスキル
データを検証します。	ターゲットの Amazon Redshift データベース内のデータを検証します。	DBA
外部キーとトリガーを有効にします。	ターゲット Amazon Redshift データベースで外部キーとトリガーを有効にします。	DBA
新しいデータベースに接続します。	新しい Amazon Redshift データベースに接続するようにアプリケーションを設定します。	アプリ所有者
最終チェックを行います。	本番稼働前に、最終的かつ包括的なシステムチェックを行います。	DBA、アプリ所有者
本番稼働。	ターゲットの Amazon Redshift データベースで稼働を開始します。	DBA

移行プロジェクトを閉じます

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。	AWS DMS レプリケーションインスタンスや AWS SCT に使用される EC2 インスタンスなどの一時的な AWS リソースをシャットダウンします。	DBA、システム管理者
文書を確認する。	移行プロジェクトドキュメントを確認して検証します。	DBA、システム管理者
メトリクスを収集します。	移行にかかる時間、手動タスクとツールタスクの比率、総	DBA、システム管理者

タスク	説明	必要なスキル
	コスト削減額など、移行プロジェクトに関する情報を収集します。	
プロジェクトを終了します。	プロジェクトを終了し、フィードバックを提供します。	DBA、システム管理者

関連リソース

リファレンス

- 「AWS DMS ユーザーガイド」
- 「AWS SCT ユーザーガイド」
- 「Amazon Redshift 入門ガイド」

チュートリアルと動画

- 「AWS SCT と AWS DMS について深く掘り下げてみよう」 (AWS re: Invent 2019 からのプレゼンテーション)
- AWS Database Migration Service の使用開始

AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行

センティル・ラマサミー (AWS) によって作成されました

概要

このパターンでは、AWS データ移行サービス (AWS DMS) と AWS Schema Conversion Tool (AWS SCT) を使用して Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する方法を説明します。

このパターンは、オンプレミスの Oracle ソース Oracle データベース、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにをインストールする Oracle データベース、および Oracle データベースの Amazon Relational Database Service (Amazon RDS) を対象としています。このパターンは、これらのデータベースを Aurora PostgreSQL 互換に変換します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスデータセンターまたは AWS クラウドにある Oracle データベース。
- ローカルマシンまたは EC2 インスタンスにインストールされた SQL クライアント。
- AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバー。ローカルマシンまたは AWS SCT がインストールされている EC2 インスタンスにインストールされます。

制約事項

- データベースサイズの上限:128 TB
- ソースデータベースが市販の (COTS) アプリケーションをサポートしているか、ベンダー固有の場合、別のデータベースエンジンに変換できない場合があります。このパターンを使用する前に、アプリケーションが Aurora PostgreSQL 互換をサポートしていることを確認します。

製品バージョン

- セルフ マネージド 型 Oracle データベースの場合、AWS DMS ではバージョン 10.2 以降(バージョン 10.x の場合)、11g から 12.2、18c、19c までのすべての Oracle データベースエディションに対応しています。サポートされている Oracle データベースバージョン (自己管理型と Oracle

Amazon RDS for Oracle の両方) の最新リストについては、「Oracle データベースを AWS DMS のソースとして使用する」と「AWS DMS のターゲットとしての PostgreSQL データベースの使用」を参照してください。

- 最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。AWS SCT でサポートされている Oracle データベースのバージョンについては、「AWS SCT のドキュメント」を参照してください。
- 「Aurora PostgreSQL リリースおよびエンジンのバージョン」は Amazon Aurora PostgreSQL リリースおよびエンジンのバージョンで管理されています。

アーキテクチャ

ソーステクノロジースタック

次のいずれかです:

- オンプレミスの Oracle データベース
- EC2 インスタンス上の Oracle データベース
- Amazon RDS for Oracle DB インスタンス。

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換

ターゲットアーキテクチャ

データ移行アーキテクチャ

- AWS クラウドで実行する Oracle データベースから

- オンプレミスデータセンターで実行する Oracle データベースから

ツール

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマと大部分のカスタムコード (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを簡単に移行できるようにします。

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースデータベースを準備します。	ソースデータベースを準備するには、AWS SCT ドキュメントの「Oracle データベースを AWS SCT のソースとして使用する」を参照してください。	DBA
AWS SCT の EC2 インスタンスを作成します。	必要に応じて AWS SCT の EC2 インスタンスを作成して設定します。	DBA
AWS SCT をダウンロードします。	AWS SCT および関連ドライバーの最新バージョンをダウンロードします。詳細については、AWS SCT ドキュメントの「AWS SCT のインストール、検証、更新」を参照してください。	DBA
ユーザーおよびアクセス許可を追加します。	ソースデータベースに前提条件となるユーザーと権限を追加して検証します。	DBA

タスク	説明	必要なスキル
AWS SCT プロジェクトを作成します。	ワークロード用の AWS SCT プロジェクトを作成し、ソースデータベースに接続します。手順については、AWS SCT ドキュメントの「AWS SCT プロジェクトの作成」と「データベースサーバーの追加」を参照してください。	DBA
実現可能性を評価します。	自動的に変換できないスキーマのアクション項目を要約し、手作業による変換作業の見積もりを記載した評価レポートを作成します。詳細については、AWS SCT ドキュメントの「データベース移行評価レポートの作成とレビュー」を参照してください。	DBA

ターゲットデータベースの準備

タスク	説明	必要なスキル
Amazon RDS DB インスタンスを作成します。	Amazon Aurora をデータベースエンジンとして使用して、ターゲット Amazon RDS DB インスタンスを作成します。手順については、Amazon RDS ドキュメントの「Amazon RDS DB インスタンスの作成」を参照してください。	DBA

タスク	説明	必要なスキル
ユーザ、ロール、権限を抽出します。	ユーザー、ロール、権限のリストをソースデータベースから抽出します。	DBA
ユーザーをマップします。	既存のデータベースユーザーを新しいデータベースユーザーにマッピングします。	アプリ所有者
ユーザーを作成します。	ターゲットデータベースにユーザーを作成します。	DBA、アプリ所有者
ロールを適用します。	前のステップのロールをターゲットデータベースに適用します。	DBA
オプション、パラメータ、ネットワークファイル、データベースリンクをチェックします。	ソースデータベースでオプション、パラメーター、ネットワークファイル、データベースリンクを確認し、ターゲットデータベースへの適用性を評価します。	DBA
設定を適用します。	ターゲットデータベースに関連する設定を適用します。	DBA

オブジェクトを転送

タスク	説明	必要なスキル
AWS SCT 接続を設定します。	ターゲットデータベースへの AWS SCT 接続を設定します。	DBA
AWS SCT を使用してスキーマを変換します。	AWS SCT は、ソースデータベーススキーマとほとんどのカスタムコードをターゲット	DBA

タスク	説明	必要なスキル
	データベースと互換性のある形式に自動的に変換します。ツールで自動的に変換されないコードが明確にマークされ、手動で変換できます。	
レポートを確認します。	生成された SQL レポートを確認し、エラーと警告をすべて保存します。	DBA
自動スキーマ変更を適用します。	自動スキーマ変更をターゲットデータベースに適用するか、sql ファイルとして保存します。	DBA
オブジェクトを検証します。	AWS SCT がターゲット上にオブジェクトを作成したことを確認します。	DBA
変換されなかった項目を処理します。	自動的に変換できなかった項目は手動で書き換え、却下、または再設計します。	DBA、アプリ所有者
ロールとユーザー権限を適用します。	生成されたロールとユーザー権限を適用し、例外がないか確認します。	DBA

データを移行する

タスク	説明	必要なスキル
方法を決めます。	データを移行する方法を決めます。	DBA
レプリケーションインスタンスを作成します。	AWS DMS コンソールからレプリケーションインスタンス	DBA

タスク	説明	必要なスキル
	を作成します。詳細については、AWS DMS ドキュメントの「AWS DMS レプリケーションインスタンスの使用」を参照してください。	
ソースおよびターゲットエンドポイントを作成します。	エンドポイントを作成するには、AWS DMS ドキュメントの「ソースエンドポイントとターゲットエンドポイントの作成」の手順に従います。	DBA
レプリケーションタスクを作成します。	タスクを作成するには、AWS DMS ドキュメントの「AWS DMS タスクの操作」を参照してください。	DBA
レプリケーションタスクを開始し、ログを監視します。	この手順の詳細については、「AWS DMS タスクの監視」を参照してください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーションコード内の SQL 項目を分析して変換します。	AWS SCT を使用して、アプリケーションコード内の SQL 項目を分析および変換します。エンジン間でデータベーススキーマを変換するときは、古いデータベースエンジンの代わりに新しいデータベースエンジンとやり取りするように、アプリケーションの SQL コードを更新する必要が	アプリ所有者

タスク	説明	必要なスキル
	あります。変換された SQL コードは表示、分析、編集、保存できます。	
アプリケーションサーバーを作成します。	AWS に新しいアプリケーションサーバーを作成します。	アプリ所有者
アプリケーションコードを移行します。	アプリケーションコードを新しいサーバーに移行します。	アプリ所有者
アプリケーションサーバーを設定します。	ターゲットデータベースとドライバーにアプリケーションサーバーを設定します。	アプリ所有者
コードを修正します。	アプリケーションのソースデータベースエンジンに固有のコードを修正します。	アプリ所有者
コードを最適化します。	ターゲットデータベースエンジンに合わせてアプリケーションコードを最適化します。	アプリ所有者

カットオーバー

タスク	説明	必要なスキル
ターゲットデータベースにカットオーバーする	新しいデータベースへのカットオーバーを実行します。	DBA
アプリケーションをロックします。	これ以上変更されないようにアプリケーションをロックします。	アプリ所有者

タスク	説明	必要なスキル
変更を検証します。	すべての変更がターゲットデータベースに反映されたことを検証します。	DBA
ターゲットデータベースにリダイレクトします。	新しいアプリケーションサーバーをターゲットデータベースにポイントします。	アプリ所有者
すべてチェックします。	最終的かつ包括的なシステムチェックを行います。	アプリ所有者
本番稼働。	最終的なカットオーバータスクを完了します。	アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的なリソースをシャットダウンします。	AWS DMS レプリケーションインスタンスや AWS SCT に使用される EC2 インスタンスなどの一時的な AWS リソースをシャットダウンします。	DBA、アプリ所有者
フィードバックを更新します。	社内チーム向けの AWS DMS プロセスに関するフィードバックを更新します。	DBA、アプリ所有者
プロセスとテンプレートを改訂します。	AWS DMS プロセスを改訂し、必要に応じてテンプレートを改善します。	DBA、アプリ所有者
ドキュメントを検証します。	プロジェクト文書を確認して検証する。	DBA、アプリ所有者

タスク	説明	必要なスキル
メトリクスを収集します。	メトリクスを収集して、移行に要する時間、手動によるコスト削減とツールのコスト削減率などを評価します。	DBA、アプリ所有者
プロジェクトを閉じます。	移行プロジェクトを終了し、利害関係者にフィードバックを提供します。	DBA、アプリ所有者

関連リソース

リファレンス

- 「AWS DMS のソースとして Oracle データベースを使用する」
- 「PostgreSQL データベースを AWS Database Migration Service のターゲットとして使用する」
- 「Oracle Database 11g/12c から PostgreSQL 互換の Amazon Aurora (9.6.x) への移行プレイブック」
- 「Oracle Database 19c から PostgreSQL 互換 (12.4) 対応 Amazon Aurora への移行プレイブック」
- 「Amazon RDS for Oracle データベースから Amazon Aurora PostgreSQL 互換エディションへの移行」
- 「AWS Database Migration Service」
- 「AWS Schema Conversion Tool」
- 「Oracle から Amazon Aurora への移行」
- 「Amazon RDS の価格設定」

チュートリアルと動画

- 「データベース移行の段階的説明」
- AWS DMS の使用開始
- 「Amazon RDS の開始方法」
- 「AWS Data Migration Service (動画)」

- 「Oracle データベースの PostgreSQL への移行」 (動画)

追加情報

.

オンプレミスの Oracle データベースから Aurora PostgreSQL にデータを移行する

作成者: Michelle Deng (AWS)、Shunan Xiang (AWS)

概要

このパターンは、オンプレミスの Oracle データベースから Amazon Aurora PostgreSQL 互換エディションにデータを移行する方法について説明しています。これは、高度なデータ操作言語 (DML) アクティビティが多い大規模なテーブルを含む、マルチテラバイトの Oracle データベースのダウンタイムを最小限に抑えたオンライン データ移行戦略を対象としています。Oracle Active Data Guard スタンバイデータベースをソースとして使用し、プライマリデータベースからのデータ移行の負荷を軽減します。Oracle プライマリデータベースからスタンバイデータベースへのレプリケーションは、ORA-01555 エラーを回避するため、フルロード中は中断できます。

通常、NUMBER データ型のプライマリキー (PK) または外部キー (FK) のテーブル列は、Oracle で整数を格納するために使用されます。パフォーマンスを向上させるために、PostgreSQL ではこれらを INT または BIGINT に変換することをお勧めします。AWS Schema Conversion Tool (AWS SCT) を使用して、PK 列と FK 列のデフォルトのデータ型マッピングを変更できます。(詳細については、AWS ブログ記事「NUMBER データ型を Oracle から PostgreSQL に変換する」を参照してください)。このパターンのデータ移行では、フルロードと変更データキャプチャ (CDC) の両方に AWS Database Migration Service (AWS DMS) を使用します。

また、このモードを使用して、オンプレミス Oracle データベースを Postgre SQL 用のAmazon Relational Database Service (Amazon RDS) に移行したり、Amazon Elastic Compute Cloud (Amazon EC2) 上でホスティングされている Oracle データベースを Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換に移行することもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Active Data Guard スタンバイが構成されたオンプレミス データセンター内の Oracle ソース データベース
- オンプレミスデータセンターと AWS クラウドの間に AWS Direct Connect が構成されています
- Oracle データベースを AWS DMS のソースとして使用することに精通している
- Oracle データベースを AWS DMS のターゲットとして使用することに精通している

機能制限

- Amazon Aurora データベースクラスターは、最大 128 TiB のストレージを作成できます。Amazon RDS for PostgreSQL データベースインスタンスは、最大 64 TiB のストレージを作成できます。最新のストレージ情報については、AWS ドキュメントの「Amazon Aurora のストレージと信頼性」および「Amazon RDS DB インスタンスストレージ」を参照してください。

製品バージョン

- AWS DMS は、バージョン 10.2 以降 (バージョン 10.x の場合)、11g から 12.2、18c、19c までのすべての Oracle データベースエディションに対応しています。サポートされているバージョンの最新リストについては、AWS ドキュメントの「AWS DMS のソースとして Oracle データベースを使用する」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Oracle Active Data Guard スタンバイが構成されたオンプレミスの Oracle データベース

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換

データ移行アーキテクチャ

ツール

- AWS DMS — 「AWS Database Migration Service」(AWS DMS) は、複数のソースデータベースとターゲットデータベースをサポートします。サポートされている Oracle のソースデータベースおよびターゲットデータベースのバージョンとエディションのリストについては、AWS DMS ドキュメントの「AWS DMS のソースとして Oracle データベースを使用する」を参照してください。ソースデータベースが AWS DMS でサポートされていない場合は、フェーズ 6 (エピックセクション) でデータを移行する別の方法を選択する必要があります。重要な注意: これは異機種間の移行であるため、まず、データベースが商用オフザシェルフ (COTS) アプリケーションをサポートしているか確認する必要があります。アプリケーションが COTS の場合は、続行する前に、ベンダーに問い合わせて Aurora PostgreSQL 互換がサポートされていることを確認してください。詳

細については、AWS ドキュメントの[AWS DMS の段階的な移行チュートリアル]を参照してください。

- AWS SCT - AWS Schema Conversion Tool」(AWS SCT)　は、ソースデータベーススキーマと大部分のカスタムコード をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースの移行を処理します。ツールが変換されるカスタムコードには、ビュー、ストアドプロシージャ、関数が含まれます。ツールで自動的に変換されないコードは明確にマークされるので、ユーザーが手動で変換できます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。		DBA
AWS SCT とドライバーをインストールします。		DBA
AWS SCT の前提条件であるユーザーとグラントソースデータベースを追加して検証します。		DBA
ワークロード用の AWS SCT プロジェクトを作成し、ソースデータベースに接続します。		DBA
評価レポートを生成し、実現可能性を評価します。		DBA、アプリ所有者

ターゲットデータベースの準備

タスク	説明	必要なスキル
Aurora PostgreSQL 互換ターゲットデータベースを作成します。		DBA
ソースデータベースからユーザー、ロール、権限リストを抽出します。		DBA
既存のデータベースユーザーを新しいデータベースユーザーにマッピングします。		アプリ所有者
ターゲットデータベースにユーザーを作成します。		DBA
前のステップのロールをターゲット Aurora PostgreSQL 互換データベースに適用します。		DBA
ソースデータベースのデータベースオプション、パラメータ、ネットワークファイル、およびデータベースリンクを確認し、ターゲットデータベースへの適用性を評価します。		DBA、アプリ所有者
ターゲットデータベースに関連する設定を適用します。		DBA

データベースオブジェクトコード変換の準備を整えます

タスク	説明	必要なスキル
ターゲットデータベースへの AWS SCT 接続を設定します。		DBA
AWS SCT でスキーマを変換されたコードを.sql ファイルとして保存します。		DBA、アプリ所有者
自動変換に失敗したデータベースオブジェクトをすべて手動で変換します。		DBA、アプリ所有者
データベースコード変換を最適化します。		DBA、アプリ所有者
.sql ファイルをオブジェクトタイプに基づいて複数の.sql ファイルに分割します。		DBA、アプリ所有者
ターゲットデータベースの SQL スクリプトを確認します。		DBA、アプリ所有者

移行の準備をする

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。		DBA
ソースおよびターゲットエンドポイントを作成します。	PK と FK のデータ型が Oracle の NUMBER から PostgreSQL の BIGINT に変換されている場合は、ソースエンド	DBA

タスク	説明	必要なスキル
	ポイントを作成するときに接続プロパティ `numberDataTypeScale=-2` を指定することを検討してください。	

データを移行する — フルロード

タスク	説明	必要なスキル
ターゲットデータベースでスキーマとテーブルを作成します。		DBA
テーブルをグループ化するか、テーブルサイズに基づいて大きなテーブルを分割することで、AWS DMS のフルロードタスクを作成します。		DBA
ソース Oracle データベース上のアプリケーションを短時間にわたり停止します。		アプリ所有者
Oracle スタンバイデータベースがプライマリデータベースと同期していることを確認し、プライマリデータベースからスタンバイデータベースへのレプリケーションを停止します。		DBA、アプリ所有者
アプリケーションをソース Oracle データベースで起動します。		アプリ所有者

タスク	説明	必要なスキル
Oracle スタンバイデータベースから Aurora PostgreSQL 互換データベースへの AWS DMS フルロード タスクを並行して開始します。		DBA
フルロードが完了後、PK とセカンダリインデックスを作成します。		DBA
データを検証します。		DBA

データを移行する — CDC

タスク	説明	必要なスキル
Oracle スタンバイデータベースがプライマリデータベースと同期され、前のタスクでアプリケーションが再起動される前に、カスタム CDC 開始時間またはシステム変更番号 (SCN) を指定して、AWS DMS の進行中のレプリケーションタスクを作成します。		DBA
AWS DMS タスクを並行して開始し、進行中の変更を Oracle スタンバイデータベースから Aurora PostgreSQL 互換データベースに複製します。		DBA
Oracle プライマリデータベースからスタンバイデータベー		DBA

タスク	説明	必要なスキル
スへのレプリケーションを再確立します。		
ターゲット Aurora PostgreSQL 互換データベースがソース Oracle データベースとほぼ同期した後、ログをモニタリングし、Oracle データベース上のアプリケーションを停止します。		DBA、アプリ所有者
ターゲットがソース Oracle データベースと完全に同期されたら、AWS DMS タスクを停止します。		DBA
FK を作成し、ターゲットデータベースのデータを検証します。		DBA
ターゲットデータベースに関数、ビュー、トリガー、シーケンス、その他のオブジェクトタイプを作成します。		DBA
ターゲットデータベースにロールの権限を適用します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
AWS SCT を使用して、アプリケーションコード内の SQL		アプリ所有者

タスク	説明	必要なスキル
ステートメントを分析および変換します。		
AWS に新しいアプリケーションサーバーを作成します。		アプリ所有者
アプリケーションコードを新しいサーバーに移行します。		アプリ所有者
ターゲットデータベースとドライバー用にアプリケーションサーバーを設定します。		アプリ所有者
アプリケーションのソースデータベースエンジンに固有のコードを修正します。		アプリ所有者
ターゲットデータベースに合わせてアプリケーションコードを最適化します。		アプリ所有者

カットオーバー

タスク	説明	必要なスキル
新しいアプリケーションサーバーをターゲットデータベースにポイントします。		DBA、アプリ所有者
健全性チェックを実行します。		DBA、アプリ所有者
本番稼働。		DBA、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、システム管理者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者
移行の所要時間、手動タスクとツールによるタスクの割合、コスト削減などのメトリクスを収集します。		DBA、アプリ所有者
プロジェクトを終了し、フィードバックを提供します。		DBA、アプリ所有者

関連リソース

リファレンス

- 「Oracle Database から Aurora PostgreSQL 互換への移行プレイブック」
- 「Amazon RDS for Oracle Database から Amazon Aurora MySQLへの移行」
- AWS DMS ウェブサイト
- AWS DMS のドキュメント
- 「AWS SCT ウェブサイト」
- AWS SCT のドキュメント
- 「Oracle から Amazon Aurora への移行」

チュートリアル

- AWS DMS の使用開始
- 「Amazon RDS の開始方法」
- 「AWS Database Migration Service のステップバイステップチュートリアル」

AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する

作成者: Amit Kumar (AWS)

概要

このパターンは、SAP Adaptive Server Enterprise (ASE) データベースを、Microsoft SQL サーバーを実行している Amazon Relational Database Service (Amazon RDS) DB インスタンスに移行させるためのガイダンスを提供します。ソースデータベースは、オンプレミスのデータセンターに配置することも、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに配置することもできます。このパターンでは、AWS Database Migration Service (AWS DMS) を使用してデータ移行を行い、(オプションで) コンピューター支援ソフトウェアエンジニアリング (CASE) ツールを使用してデータベーススキーマを変換します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスデータセンターまたは EC2 インスタンスにある SAP ASE データベース
- 稼働中のターゲット Amazon RDS for SQL Server データベース

制限

- データベースサイズの上限: 64 TB

製品バージョン

- SAP ASE バージョン 15.7 または 16.x のみ。最新情報については、「SAP データベースを DMS のソースとして使用する」を参照してください。
- Amazon RDS ターゲットデータベースの場合、AWS DMS は Enterprise、Standard、Web、および Express の各エディションでAmazon RDS の Microsoft SQL Server バージョンをサポートします。サポートされているバージョンの最新情報については、 DMS ドキュメントを参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスまたは Amazon EC2 インスタンスにある SAP ASE データベース

ターゲットテクノロジースタック

- Amazon RDS for SQL Server の DB インスタンス

ソースアーキテクチャとターゲットアーキテクチャ

Amazon EC2 上の SAP ASE データベースから Amazon RDS for SQL Server DB インスタンスへ

オンプレミスの SAP ASE Database から Amazon RDS for SQL Server) DB インスタンスへ

ツール

- AWS Database Migration Service (AWS DMS) は、オンプレミスであるか、Amazon RDS DB インスタンス上にある、または EC2 インスタンス内のデータベースから、Amazon RDS for SQL Server や EC2 インスタンスなどの AWS サービス上のデータベースにデータを移行するために使用できるウェブサービスです。データベースを AWS Service からオンプレミスのデータベースに移行することもできます。異種または同種のデータベースエンジン間でデータを移行できます。
- スキーマの変換には、オプションで erwin Data Modeler または SAP PowerDesigner を使用できます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。		DBA
ストレージ要件 (ストレージタイプと容量) を識別します。		DBA、SysAdmin

タスク	説明	必要なスキル
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、SysAdmin
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。		DBA、SysAdmin
アプリケーション移行戦略を特定します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		SysAdmin
セキュリティグループとネットワークアクセスコントロールリスト (ACL) を作成します。		SysAdmin
Amazon RDS DB インスタンスを構成して起動します。		SysAdmin

データ移行 - オプション 1

タスク	説明	必要なスキル
データベーススキーマを手動で移行するか、erwin Data		DBA

タスク	説明	必要なスキル
Modeler や SAP PowerDesigner などの CASE ツールを使用します。		

データ移行 — オプション 2

タスク	説明	必要なスキル
AWS DMS を使用してデータを移行します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、SysAdmin、アプリ所有者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin

タスク	説明	必要なスキル
プロジェクト文書を確認して検証する。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動タスクと自動タスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを終了し、フィードバックを提供します。		DBA、SysAdmin、アプリ所有者

関連リソース

リファレンス

- AWS DMS ウェブサイト
- Amazon SNS の料金
- SAP ASE データベースの DMS のソースとしての使用
- RDS Custom for SQL Server の制限

チュートリアルと動画

- AWS DMS の使用開始
- Amazon RDS の使用開始
- DMS (ビデオ)
- Amazon RDS (ビデオ)

AWS DMS を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する

作成者: Marcelo Fernandes (AWS)

概要

このパターンは、AWS Data Migration Service (AWS DMS) を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行するためのガイダンスを提供します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターにあるソース Microsoft SQL Server データベース
- AWS DMS ドキュメントで説明されている Amazon Redshift データベースを AWS DMS のターゲットとして使用するための前提条件を完了

製品バージョン

- SQL Server 2005 ~ 2019、Enterprise、Standard、Workgroup、Developer、および Web エディション サポートされているバージョンの最新リストについては、AWS ドキュメントの AWS DMS のソースとして Microsoft SQL Server データベースを使用するを参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス Microsoft SQL Server データベース

ターゲットテクノロジースタック

- Amazon Redshift

データ移行アーキテクチャ

ツール

- AWS DMS は、複数のタイプのソースデータベースとターゲットデータベースをサポートするデータ移行サービスです。AWS DMS での使用がサポートされている Microsoft SQL Server データベースのバージョンとエディションについては、AWS DMS ドキュメントの AWS DMS のソースとして Microsoft SQL Server データベースを使用するを参照してください。AWS DMS がソースデータベースをサポートしていない場合は、データ移行の代替方法を選択する必要があります。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を識別します。		DBA、システム管理者
ストレージ要件 (ストレージタイプと容量) を識別します。		DBA、システム管理者
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、システム管理者
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。		DBA、システム管理者
アプリケーション移行戦略を特定します。		DBA、アプリ所有者、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。	詳細については、VPC での DB インスタンスの操作を参照してください。	システム管理者
セキュリティグループを作成します。		システム管理者
Amazon Redshift クラスターを設定して起動します。	詳細については、Amazon Redshift ドキュメントの Amazon Redshift クラスターのサンプルを作成するを参照してください。	DBA、システム管理者

データを移行する

タスク	説明	必要なスキル
AWS DMS を使用して Microsoft SQL Server データベースからデータを移行します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、アプリ所有者、システム管理者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、システム管理者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者、システム管理者
移行の所要時間、手動タスクと自動タスクの割合、コスト削減などのメトリクスを収集します。		DBA、アプリ所有者、システム管理者
プロジェクトを終了し、フィードバックを提供します。		DBA、アプリ所有者、システム管理者

関連リソース

リファレンス

- AWS DMS ドキュメント
- Amazon Redshift ドキュメント
- Amazon Redshift の価格設定

チュートリアルと動画

- AWS DMS の使用開始
- Amazon Redshift の使用開始
- AWS Database Migration Service のターゲットとして Amazon Redshift データベースを使用する
- AWS DMS (動画)

AWS SCT データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する

作成者: Neha Thakur (AWS)

概要

このパターンは、AWS Schema Conversion Tool (AWS SCT) データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server ソースデータベースを Amazon Redshift ターゲットデータベースに移行する手順の概要を示しています。エージェントは AWS SCT と統合されていますが、データ変換は他の場所で実行し、ユーザーに代わって他の AWS サービスとインタラクトする外部プログラムです。

前提条件と制限

前提条件

- オンプレミスデータセンターのデータウェアハウスワークロードに使用される Microsoft SQL Server ソースデータベース
- アクティブな AWS アカウント

製品バージョン

- Microsoft SQL Server バージョン 2008 以降 サポートされているバージョンの最新一覧については、 AWS SCT ドキュメントを参照してください。

アーキテクチャ

テクノロジースタックソース

- オンプレミス Microsoft SQL Server データベース

テクノロジースタックターゲット

- Amazon Redshift

データ移行アーキテクチャ

ツール

- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコードをターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースの移行を処理します。ソースデータベースとターゲットデータベースが大きく異なる場合は、AWS SCT エージェントを使用して追加のデータ変換を実行できます。詳細については、AWS ドキュメントのオンプレミスデータウェアハウスから Amazon Redshift へのデータの移行を参照してください。

ベストプラクティス

- AWS SCT のベストプラクティス
- Amazon Redshift のベストプラクティス

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
適切なインスタンスタイプ（容量、ストレージ機能、ネットワーク機能）を選択します。		DBA、SysAdmin
ソースとターゲットデータベースのネットワークアクセ		DBA、SysAdmin

タスク	説明	必要なスキル
スセキュリティ要件を特定します。		
アプリケーション移行戦略を選択します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		SysAdmin
セキュリティグループを作成します。		SysAdmin
Amazon Redshift クラスターを設定して、起動します。		SysAdmin

データを移行する

タスク	説明	必要なスキル
AWS SCT データ抽出エージェントを使用してデータを移行します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
選択したアプリケーション移行戦略に従ってください。		DBA、SysAdmin、アプリ所有者

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin
プロジェクト文書を確認して検証する。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動タスクと自動タスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを閉じて、フィードバックします。		DBA、SysAdmin、アプリ所有者

関連リソース

リファレンス

- 「AWS SCT ユーザーガイド」
- Using data extraction agents
- Amazon Redshift の価格設定

チュートリアルと動画

- AWS Schema 変換ツールの使用開始

- Amazon Redshift の使用開始

AWS SCT データ抽出エージェントを使用して Teradata データベースを Amazon Redshift に移行

作成者: Sergey Dmitriev (AWS)

概要

このパターンでは、オンプレミスデータセンターのデータウェアハウスとして使用されている Teradata データベースを Amazon Redshift データベースに移行する手順を順に説明します。このパターンは、AWS Schema Conversion Tool (AWS SCT) データ抽出エージェントを使用します。エージェントは AWS SCT と統合されていますが、他の場所でデータ変換を行い、ユーザーに代わって他の AWS サービスとやり取りする外部プログラムです。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの Teradata ソースデータベース

製品バージョン

- Teradata バージョン 13 以降。サポートされているバージョンの最新リストについては、「AWS SCT のドキュメント」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス Teradata データベース

ターゲットテクノロジースタック

- Amazon Redshift クラスター

データ移行アーキテクチャ

ツール

- AWS SCT —「AWS Schema Conversion Tool」(AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコード (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを移行できます。ソースデータベースとターゲットデータベースは相互に大きく異なる場合は、AWS SCT エージェントにより、追加のデータ変換を実行できます。詳細については、AWS ドキュメントの「Migrating Data from an On-Premises Data Warehouse to Amazon Redshift」を参照してください。

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
適切なインスタンスタイプ（容量、ストレージ機能、ネットワーク機能）を選択します。		DBA、SysAdmin
ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、SysAdmin
アプリケーション移行戦略を選択します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		SysAdmin
セキュリティグループを作成します。		SysAdmin
Amazon Redshift クラスターを設定して、起動します。		SysAdmin

データを移行する

タスク	説明	必要なスキル
AWS SCT データ抽出エージェントにより、データを移行します。	AWS SCT データ抽出エージェントの使用方法の詳細については、「参考資料とヘルプ」セクションのリンクを参照してください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
選択したアプリケーション移行戦略に従ってください。		DBA、SysAdmin、アプリ所有者

ターゲット Amazon Redshift データベースへの切り取り

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin
プロジェクト文書を確認して検証する。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動タスクとツールタスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを閉じて、フィードバックします。		

関連リソース

リファレンス

- 「AWS SCT ユーザーガイド」
- Using data extraction agents
- Amazon Redshift の価格設定
- 「Teradata の RESET WHEN 特徴量を Amazon Redshift SQL に変換」(AWS 規範ガイダンス)
- 「Teradata のNORMALIZE の時系列特徴量を Amazon Redshift SQL」(AWS 規範ガイダンス)

チュートリアル

- 「AWS Schema Conversion Tool の開始方法」
- Amazon Redshift の使用開始

AWS SCT データ抽出エージェントを使用してオンプレミスの Vertica データベースを Amazon Redshift に移行する

作成者: Sergey Dmitriev (AWS)

概要

このパターンは、AWS Schema Conversion Tool (AWS SCT) データ抽出エージェントを使用してオンプレミスの Vertica データベースを Amazon Redshift クラスターに移行するためのガイダンスを提供します。エージェントは AWS SCT と統合されているが、他の場所でデータ変換を行い、ユーザーに代わって他の AWS サービスとやり取りする外部プログラムです。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターのデータウェアハウスワークロードに使用される Vertica ソースデータベース
- Amazon Redshift ターゲットクラスター

製品バージョン

- Vertica (バージョン 7.2.2 以降)。サポートされているバージョンの最新リストについては、AWS SCT のドキュメントを参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Vertica データベース

ターゲットテクノロジースタック

- Amazon Redshift クラスター

データ移行アーキテクチャ

ツール

- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコードをターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースの移行を処理します。ソースデータベースとターゲットデータベースは相互に大きく異なる場合は、AWS SCT エージェントを使用して追加のデータ変換を行うことができます。詳細については、AWS ドキュメントの「Migrating Data from an On-Premises Data Warehouse to Amazon Redshift」を参照してください。

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。		DBA
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
適切なインスタンスタイプ（容量、ストレージ機能、ネットワーク機能）を選択します。		DBA、SysAdmin
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、SysAdmin
アプリケーション移行戦略を選択します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		SysAdmin
セキュリティグループを作成します。		SysAdmin
Amazon Redshift クラスターをセットアップして起動します。		SysAdmin

データを移行する

タスク	説明	必要なスキル
AWS SCT データ抽出エージェントを使用してデータを移行します。	AWS SCT データ抽出エージェントの使用方法の詳細については、「リファレンスとヘルプ」セクションのリンクを参照してください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
選択したアプリケーション移行戦略に従ってください。		DBA、SysAdmin、アプリ所有者

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin
プロジェクト文書を確認して検証する。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動タスクとツールタスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを閉じて、フィードバックします。		

関連リソース

リファレンス

- 「AWS SCT ユーザーガイド」
- Using data extraction agents
- Amazon Redshift の価格設定

チュートリアルと動画

- AWS Schema 変換ツールの使用開始

- Amazon Redshift の使用開始

レガシーアプリケーションを Oracle Pro*C から ECPG に移行する

作成者: Sai Parthasaradhi (AWS) と Mahesh Balumuri (AWS)

概要

SQL コードが埋め込まれているほとんどのレガシーアプリケーションは、Oracle Pro*C プリコンパイラを使用してデータベースにアクセスします。これらの Oracle データベースを PostgreSQL 用の Amazon Relational Database Service (Amazon RDS) または Amazon Aurora PostgreSQL-Compatible エディションに移行する場合、アプリケーションコードを PostgreSQL のプリコンパイラと互換性のある形式 (ECPG と呼ばれる) に変換する必要があります。このパターンは、Oracle Pro*C のコードを PostgreSQL ECPG の同等のコードに変換する方法を説明しています。

Pro*C の詳細については、Oracle のドキュメントを参照してください。ECPG の簡単な紹介については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon RDS for PostgreSQL または Aurora PostgreSQL-Compatible データベース
- オンプレミスで稼働している Oracle データベース

ツール

- 次のセクションにリストされている PostgreSQL パッケージ。
- CLI - AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールで、コマンドラインシェルのコマンドで AWS サービスとインタラクトします。最小限の構成で、コマンドプロンプトからブラウザベースの AWS マネジメントコンソールで提供される機能と同等の機能を実装する AWS CLI コマンドを実行できます。

エピック

CentOS または RHEL でビルド環境を設定する

タスク	説明	必要なスキル

PostgreSQL パッケージをインストールします。	次のコマンドを使用して必要な PostgreSQL パッケージをインストールします。
	
	`yum update -y`
	`yum install -y yum-utils`
	`rpm -ivh https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm`
	`dnf -qy module disable postgresql`
		アプリ開発者、DevOps エンジニア

ヘッダーファイルとライブラリをインストールします。	以下のコマンドを使用して、ヘッダーファイルとライブラリを含む `postgresql12-devel` パッケージをインストールします。ランタイム環境でのエラーを避けるため、開発環境とランタイム環境の両方にパッケージをインストールします。
	
	`dnf -y install postgresql12-devel`
	`yum install ncompress zip ghostscript jq unzip wget git -y`
		アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル

	開発環境でのみ、次のコマンドも実行します。
	
	`yum install zlib-devel make -y`
	`ln -s /usr/pgsql-12/bin/ecpg /usr/bin/`	

環境変数のパスを構成します。	PostgreSQL クライアントライブラリの環境パスを設定します。 `export PATH=$PATH:/usr/pgsql-12/bin`	アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル

必要に応じて追加のソフトウェアをインストールします。	必要に応じて、Oracle の SQL*Loader の代わりに pgLoader をインストールしてください。
	
	`wget -O /etc/yum.repos.d/pgloader-ccl.repo https://dl.packager.io/srv/opf/pgloader-ccl/master/installer/el/7.repo`
	`yum install pgloader-ccl -y`
	`ln -s /opt/pgloader-ccl/bin/pgloader /usr/bin/`
	
	Pro*C モジュールから Java アプリケーションを呼び出す場合は、Java をインストールしてください。
	
	`yum install java -y`
	
	ant をインストールして Java コードをコンパイルします。
	
	`yum install ant -y`
		アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル

AWS CLI をインストールします。	AWS CLI をインストールして、アプリケーションから AWS Secrets Manager や Amazon Simple Storage Service (Amazon S3) などの AWS のサービスと対話するコマンドを実行します。
	`cd /tmp/`
	`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`./aws/install -i /usr/local/aws-cli -b /usr/local/bin --update`
		アプリ開発者、DevOps エンジニア

変換するプログラムを特定します。	Pro*C から ECPG に変換するアプリケーションを特定します。	アプリ開発者、アプリオーナー

Pro*C コードを ECPG に変換する

タスク	説明	必要なスキル
不要なヘッダーを削除する。	PostgreSQL では不要な `include` ヘッダーを削除します。たとえば、`oci.h`、`oratypes`、および `sqlda` など。	アプリ所有者、アプリ開発者
変数宣言を更新します。	ホスト変数として使用されるすべての変数宣言に `EXEC`	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	SQL ステートメントを追加します。 次のような EXEC SQL VAR 宣言をアプリケーションから削除します。 `EXEC SQL VAR query IS STRING(2048);`	

タスク	説明	必要なスキル

ROWNUM 機能を更新します。	ROWNUM 関数は PostgreSQL では使用できません。これを SQL クエリの ROW_NUMBER ウィンドウ関数に置き換えてください。
	
	Pro*C コード:
	
	`SELECT SUBSTR(RT RIM(FILE_NAME,'.tx t'),12) INTO :gcpclFil eseq FROM (SELECT FILE_NAME FROM DEMO_FILES_TABLE WHERE FILE_NAME LIKE '%POC%' ORDER BY FILE_NAME DESC) FL2 WHERE ROWNUM <=1 ORDER BY ROWNUM;`
	
	ECPG コード:
	
	`SELECT SUBSTR(RT RIM(FILE_NAME,'.tx t'),12) INTO :gcpclFil eseq FROM (SELECT FILE_NAME , ROW_NUMBE R() OVER (ORDER BY FILE_NAME DESC) AS ROWNUM FROM demo_sche ma.DEMO_FILES_TABLE WHERE FILE_NAME LIKE '%POC%' ORDER BY FILE_NAME DESC) FL2`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	`WHERE ROWNUM <=1 ORDER BY ROWNUM;`	

エイリアス変数を使用するように関数パラメータを更新します。	PostgreSQL では、関数パラメータをホスト変数として使用することはできません。エイリアス変数を使用してそれらを上書きします。
	
	Pro*C コード:
	
	`int processData(int referenceId){ EXEC SQL char col_val[100]; EXEC SQL select column_name INTO :col_val from table_name where col=:referenceId; }`
	
	ECPG コード:
	
	`int processData(int referenceIdParam){ EXEC SQL int referenceId = referenceIdParam; EXEC SQL char col_val[100]; EXEC SQL select column_name INTO :col_val from table_name where col=:referenceId; }`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル

構造体タイプを更新します。	struct 型変数がホスト変数として使用される場合は、EXEC SQL BEGIN および END ブロックの struct 型を typedef で定義します。struct 型がヘッダー (.h) ファイルで定義されている場合は、EXEC SQL include ステートメントでファイルをインクルードします。
	
	Pro*C コード:
	
	ヘッダーファイル (demo.h)
	
	``
	struct s_partiti
	on_ranges
	{
	 char sc_table_
	group[31];
	 char sc_table_
	name[31];
	 char sc_range_
	value[10];
	};
	struct s_partiti
	on_ranges_ind
	{
	  short ss_table_
	group;
	  short ss_table_
	name;
	  short ss_range_
	value;
	};
	``
	
	ECPG コード:
	
	ヘッダーファイル (demo.h)
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル

	`EXEC SQL BEGIN DECLARE SECTION;`
	`typedef struct`
	`{`
	`  char  sc_table_group[31];`
	`  char  sc_table_name[31];`
	`  char  sc_range_value[10];`
	`} s_partition_ranges;`
	`typedef struct`
	`{`
	`  short  ss_table_group;`
	`  short  ss_table_name;`
	`  short  ss_range_value;`
	`} s_partition_ranges_ind;`
	`EXEC SQL END DECLARE SECTION;`
	
	Pro*C ファイル (demo.pc)
	
	`#include "demo.h"`
	`struct s_partition_ranges gc_partition_data[MAX_PART_TABLE] ;`
	`struct s_partition_ranges_ind gc_partition_data_ind[MAX_PART_TABLE] ;`
	
	ECPG ファイル (demo.pc)
	
	`exec sql include "demo.h"`	

タスク	説明	必要なスキル
	`EXEC SQL BEGIN DECLARE SECTION; s_partition_ranges gc_partition_data[ MAX_PART_TABLE] ; s_partition_ranges_ind gc_partition_data_ ind[MAX_PART_TABLE] ; EXEC SQL END DECLARE SECTION;`	

カーソルから取得するようにロジックを変更します。	配列変数を使用してカーソルから複数の行を取得するには、FETCH FORWARD を使用するようコードを変更します。
	
	Pro*C コード:
	
	`EXEC SQL char aPoeFiles [MAX_FILES][FILENA ME_LENGTH]; EXEC SQL FETCH filename_ cursor into :aPoeFile s;`
	
	ECPG コード:
	
	`EXEC SQL char aPoeFiles [MAX_FILES][FILENA ME_LENGTH]; EXEC SQL int fetchSize = MAX_FILES; EXEC SQL FETCH FORWARD :fetchSiz e filename_cursor into :aPoeFiles;`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル

戻り値がないパッケージコールを修正します。	戻り値のない Oracle パッケージ関数は、指標変数を使用して呼び出す必要があります。アプリケーションに同じ名前の関数が複数含まれている場合や、型が不明な関数がランタイムエラーを生成する場合は、値をデータ型に typecast します。
	
	Pro*C コード:
	
	`void ProcessData (char *data , int id) { EXEC SQL EXECUTE BEGIN pkg_demo. process_data (:data, :id); END; END-EXEC; }`
	
	ECPG コード:
	
	`void ProcessData (char *dataParam, int idParam ) { EXEC SQL char *data = dataParam; EXEC SQL int id = idParam; EXEC SQL short rowInd;`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	EXEC SQL short rowInd = 0; EXEC SQL SELECT pkg_demo.process_data (inp_data => :data::text, inp_id => :id) INTO :rowInd; }	

タスク	説明	必要なスキル

SQL_CURSOR 変数を書き換える。	SQL_CURSOR 変数とその実装を書き直します。
	
	Pro*C コード:
	
	`/* SQL Cursor */`
	`SQL_CU`
	`RSOR demo_cursor;`
	`EXEC SQL`
	` ALLOCATE :demo_cursor;`
	`EXEC SQL EXECUTE`
	`  BEGIN`
	`  pkg_demo.`
	`get_cursor(`
	`  demo_cur=`
	`>:demo_cursor`
	`  );`
	`  END;`
	`END-EXEC;`
	
	ECPG コード:
	
	`EXEC SQL DECLARE`
	` demo_cursor CURSOR FOR`
	` SELECT`
	`  * from`
	`  pkg_demo.open_file`
	`name_rc(`
	`  demo_cur=`
	`>refcursor`
	`  ) ;`
	`EXEC SQL char open_file`
	`name_rcInd[100];`
	`# As the below function`
	` returns cursor_name`
	` as`
	`# return we need to`
	` use char[] type as`
	` indicator.`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	`EXEC SQL SELECT pkg_demo.get_cursor ( demo_cur=>'demo_cursor' ) INTO :open_filename_rcInd;`	
一般的な移行パターンを適用します。	• PostgreSQL と互換性を持つように SQL クエリを変更します。 • 匿名ブロックを、ECPG でサポートされていないときにデータベースに移動します。 • PostgreSQL ではサポートされていない `dbms_application_info` ロジックを削除します。 • カーソルを閉じた後に `EXEC SQL COMMIT` ステートメントを移動します。ループ内でカーソルからレコードを取得しているときにクエリをコミットすると、カーソルが閉じて、「カーソルが存在しません」というエラーが表示されます。 • ECPG での例外処理とエラーコードについては、PostgreSQL ドキュメントの「エラー処理」を参照してください。	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
必要に応じてデバッグを有効にします。	ECPG プログラムをデバッグモードで実行するには、メイン関数ブロック内に以下のコマンドを追加します。 `ECPGdebug(1, stderr);`	アプリ開発者、アプリオーナー

ECPG プログラムをコンパイルする

タスク	説明	必要なスキル

ECPG 用の実行ファイルを作成します。	`prog1.pgc` という名前の埋め込み SQL C ソースファイルがある場合は、以下のコマンドシーケンスを使用して実行プログラムを作成できます。
	`ecpg prog1.pgc`
	`cc -I/usr/local/pgsql/include -c prog1.c`
	`cc -o prog1 prog1.o -L/usr/local/pgsql/lib -lecpg`
		アプリ開発者、アプリオーナー

コンパイル用の Make ファイルを作成します。	次のサンプルファイルに示されているように、ECPG プログラムをコンパイルする Make ファイルを作成します。
	`CFLAGS ::= $(CFLAGS) -I/usr/pgsql-12/include -g -Wall`
	`LDFLAGS ::= $(LDFLAGS) -L/usr/pgsql-12/li`
		アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	b -Wl,-rpath,/usr/pg sql-12/lib LDLIBS ::= $(LDLIBS) - lecpg PROGRAMS = test .PHONY: all clean %.c: %.pgc ecpg $< all: $(PROGRAMS) clean: rm -f $(PROGRAM S) $(PROGRAMS:%=%.c) $(PROGRAMS:%=%.o)	

アプリケーションをテストする

タスク	説明	必要なスキル
コードをテストします。	変換したアプリケーションコードをテストして、正しく機能することを確認します。	アプリ開発者、アプリオーナー、テストエンジニア

関連リソース

- ECPG - C で記述された埋め込み SQL (PostgreSQL ドキュメント)
- エラーの処理 (PostgreSQL ドキュメント)
- Oracle Pro*C/C++ プリコンパイラを使用する理由 (Oracle ドキュメント)

追加情報

PostgreSQL には、Oracle Pro*C プリコンパイラと同等の埋め込み SQL プリコンパイラ ECPG があります。ECPG は、SQL コールを特殊な関数の呼び出しに置き換えることで、埋め込み SQL 文を含むC プログラムを標準 C コードに変換します。その後、出力ファイルは任意の C コンパイラツールチェーンで処理できます。

入力ファイルと出力ファイル

ECPG は、コマンドラインで指定した各入力ファイルを対応する C 出力ファイルに変換します。入力ファイル名にファイル拡張子が付いていない場合は、.pgc と見なされます。ファイルの拡張子は .c に置き換えられ、出力ファイル名が作成されます。ただし、-o オプションでデフォルトの出力ファイル名をオーバーライドできます。

入力ファイル名としてダッシュ (-) を使用すると、ECPG は -o オプションを使用してプログラムをオーバーライドしない限り、標準入力からプログラムを読み取り、標準出力に書き込みます。

ヘッダーファイル

PostgreSQL コンパイラは、前処理された C コードファイルをコンパイルするときに、PostgreSQL includeディレクトリ内の ECPG ヘッダーファイルを探します。そのため、-I オプションを使用してコンパイラに正しいディレクトリ (例: -I/usr/local/pgsql/include) を指定しなければならない場合があります。

[Libraries] (ライブラリ)

埋め込み SQL で C コードを使用するプログラムは、libecpg ライブラリにリンクする必要があります。たとえば、リンカーオプション -L/usr/local/pgsql/lib -lecpg を使用できます。

変換されたECPGアプリケーションは、埋め込み SQL ライブラリ (ecpglib) を介して libpq ライブラリ内の関数を呼び出し、標準のフロントエンド/バックエンドプロトコルを使用してPostgreSQLサーバーと通信します。

仮想生成列をOracleから PostgreSQL に移行

Veeranjaneyulu Grandhi (AWS)、Rajesh Madiwale (AWS)、Ramesh Pathuri (AWS) によって作成された

概要

バージョン11以前では、PostgreSQLはOracle仮想列と直接同等の機能を提供していません。Oracle Database から PostgreSQL バージョン 11 以前への移行中に仮想生成された列を処理するのは、次の 2 つの理由で困難です。

- 仮想列は移行中は表示されません。
- PostgreSQLはバージョン12より前の`generate`式をサポートしていません。

ただし、同様の機能をエミュレートする回避策があります。AWS Database Migration Service (AWS DMS) を使用して、Oracle Database から PostgreSQL バージョン 11 以前にデータを移行する場合、トリガー関数を使用して仮想生成列に値を入力できます。このパターンは、この目的に使用できる Oracle データベースと PostgreSQL コードの例を示しています。AWS では、PostgreSQL データベースには Amazon Relational Database Service (Amazon RDS) または Amazon Aurora PostgreSQL 互換エディションを使用できます。

PostgreSQL バージョン 12 以降では、生成された列がサポートされています。生成された列は、他の列値からその場で計算することも、計算して保存することもできます。「PostgreSQL で生成されたカラム」はOracleの仮想カラムに似ています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ソース: Oracle データベース
- PostgreSQL データベースをターゲットとします (Amazon RDS for PostgreSQL または Aurora PostgreSQL との互換性あり)
- PL/PgSQL コーディングの専門知識

制約事項

- バージョン 12 より前のバージョンの PostgreSQL にのみ適用されます。

- Oracle データベースバージョン 11g 以降に適用されます。
- 仮想列はデータ移行ツールではサポートされていません。
- 同じテーブルで定義されている列にのみ適用されます。
- 仮想生成列が決定論的ユーザー定義関数を参照している場合、その列をパーティショニングキー列として使用することはできません。
- 式の出力はスカラー値でなければなりません。Oracle が提供するデータ型、ユーザー定義型、LOBまたはLONG RAWを返すことはできません。
- 仮想列に対して定義されるインデックスは、PostgreSQLの関数ベースのインデックスと同等です。
- テーブル統計を収集する必要があります。

ツール

- 「pgAdmin 4」は PostgreSQL 用のオープンソース管理ツールです。このツールは、データベースオブジェクトの作成、保守、使用を簡素化するグラフィカルインターフェースを提供します。
- 「Oracle SQL Developer」は、従来のデプロイメントとクラウドデプロイメントの両方で Oracle データベース内の SQL を操作できる、無料の統合開発環境です。

エピック

ソースデータベースとターゲットデータベーステーブルの作成

タスク	説明	必要なスキル
ソース Oracle データベーステーブルを作成します。	Oracle Database で、次の文を使用して仮想生成列を含むテーブルを作成します。 `CREATE TABLE test.generated_column ( CODE NUMBER, STATUS VARCHAR2(12) DEFAULT 'PreOpen', FLAG CHAR(1) GENERATED ALWAYS AS (CASE UPPER(STATUS) WHEN 'OPEN' THEN 'N' ELSE`	DBA、アプリ開発者

タスク	説明	必要なスキル

	<pre>'Y' END) VIRTUAL
	VISIBLE
	);</pre>
	このソーステーブルでは、STATUS列のデータがAWS DMS を介してターゲットデータベースに移行されます。ただし、このFLAG列にはgenerate by機能を使用してデータが入力されるため、移行中は AWS DMSにこの列は表示されません。generated byの機能を実装するには、次の工ピックに示すように、ターゲットデータベースのトリガーと関数を使用してFLAG列の値を入力する必要があります。	

タスク	説明	必要なスキル
AWS にターゲット PostgreSQL テーブルを作成します。	次のステートメントを使用して AWS のPostgreSQLテーブルを作成します。 `CREATE TABLE test.generated_column ( code integer not null, status character varying(12) not null , flag character(1) );` このテーブルでは、status列は標準列です。このflag列は、status列内のデータに基づいて生成された列になります。	DBA、アプリ開発者

PostgreSQL で仮想列を処理するトリガー関数を作成する

タスク	説明	必要なスキル
PostgreSQL トリガーを作成します。	PostgreSQL で、トリガーを作成します。 `CREATE TRIGGER tgr_gen_column AFTER INSERT OR UPDATE OF status ON test.generated_column FOR EACH ROW EXECUTE FUNCTION test.tgf_gen_column();`	DBA、アプリ開発者

タスク	説明	必要なスキル
PostgreSQL トリガー関数を作成します。	PostgreSQL で、トリガー用の関数を作成します。この関数は、アプリケーションまたは AWS DMS によって挿入または更新された仮想列を入力し、データを検証します。 `CREATE OR REPLACE FUNCTION test.tgf_gen_column() RETURNS trigger AS $VIRTUAL_COL$ BEGIN IF (TG_OP = 'INSERT') THEN IF (NEW.flag IS NOT NULL) THEN RAISE EXCEPTION 'ERROR: cannot insert into column "flag"' USING DETAIL = 'Column "flag" is a generated column.'; END IF; END IF; IF (TG_OP = 'UPDATE') THEN IF (NEW.flag::VARCHAR != OLD.flag::varchar) THEN RAISE EXCEPTION 'ERROR: cannot update column "flag"' USING DETAIL = 'Column "flag" is a generated column.'; END IF; END IF; IF TG_OP IN ('INSERT','UPDATE') THEN`	DBA、アプリ開発者

タスク	説明	必要なスキル

	IF (old.flag is NULL)
	OR (coalesce(old.stat
	us,'') != coalesce(
	new.status,'')) THEN
	UPDATE test.gene
	rated_column
	SET flag = (CASE
	UPPER(status) WHEN
	'OPEN' THEN 'N' ELSE
	'Y' END)
	WHERE code = new.code;
	END IF;
	END IF;
	RETURN NEW;
	END
	$VIRTUAL_COL$ LANGUAGE
	plpgsql;	

AWS DMS を使用してデータ移行をテストする

タスク	説明	必要なスキル
レプリケーションインスタンスを作成します。	レプリケーションインスタンスを作成するには、AWS DMS ドキュメントの「指示」に従います。レプリケーションインスタンスは、ソースデータベースとターゲットデータベースと同じ 仮想プライベートクラウド (VPC) 内に存在する必要があります。	DBA、アプリ開発者
ソースおよびターゲットエンドポイントを作成します。	エンドポイントを作成するには、AWS DMS のドキュメントの手順に従います。	DBA、アプリ開発者

タスク	説明	必要なスキル
エンドポイント接続をテストします。	VPC とレプリケーションインスタンスを指定し、「テストを実行」を選択すると、エンドポイント接続をテストできます。	DBA、アプリ開発者
フルロードタスクを作成して開始します。	手順については、AWS DMS ドキュメントの「タスクの作成」と「フルロードタスク設定」を参照してください。	DBA、アプリ開発者
仮想列のデータを検証します。	ソースデータベースとターゲットデータベース内の仮想列のデータを比較します。データを手動で検証することも、このステップのスクリプトを記述することもできます。	DBA、アプリ開発者

関連リソース

- AWS Database Migration Service の使用開始 (AWS DMSドキュメント)
- 「AWS DMSのソースとして Oracle データベースを使用」 (AWS DMS ドキュメント)
- PostgreSQL データベースを AWS DMS のターゲットとして使用する (AWS DMS ドキュメント)
- PostgreSQL で生成されたカラム (PostgreSQL ドキュメント)
- トリガー関数 (PostgreSQL ドキュメンテーション)
- Oracle データベースの仮想カラム (Oracle ドキュメント)

Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする

作成者: Rakesh Raghav (AWS) と anuradha chintha (AWS)

概要

Oracle から Amazon Web Services (AWS) クラウド上の Amazon Aurora PostgreSQL-Compatible エディションへの移行の一環として、複数の課題に直面する場合があります。たとえば、Oracle の UTL_FILE ユーティリティに依存するコードの移行は常に課題です。Oracle PL/SQL では、UTL_FILE パッケージは基盤となるオペレーティングシステムと連携して、読み取りや書き込みなどのファイル操作に使用されます。この UTL_FILE ユーティリティは、サーバーマシンシステムとクライアントマシンシステムの両方で動作します。

Amazon Aurora PostgreSQL-Compatible は、マネージドデータベースサービスです。このため、データベースサーバー上のファイルにアクセスすることはできません。このパターンでは、Amazon Simple Storage Service (Amazon S3) と Amazon Aurora PostgreSQL-Compatible を統合して、UTL_FILE 機能のサブセットを実現する手順を示しています。この統合により、サードパーティの抽出、変換、ロード (ETL) ツールやサービスを使用せずにファイルを作成して利用できます。

オプションで Amazon CloudWatch によるモニタリングおよび Amazon SNS 通知を設定できます。

本稼働環境に実装する前に、このソリューションを徹底的にテストすることをお勧めします。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Database Migration Service (AWS DMS) 専門知識
- PL/pgSQL コーディングに関する専門知識
- Amazon Aurora PostgreSQL-Compatible クラスター
- S3 バケット

機能制限

このパターンには Oracle UTL_FILE ユーティリティの代わりとなる機能はありません。ただし、手順とサンプルコードをさらに拡張して、データベースのモダナイゼーション目標を達成することはできます。

製品バージョン

- Amazon Aurora PostgreSQL-Compatible エディション 11.9

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL-Compatible
- Amazon CloudWatch
- Amazon Simple Notification Service (Amazon SNS)
- Amazon S3

ターゲットアーキテクチャ

次の図はソリューションの概要を示しています。

1. ファイルはアプリケーションから S3 バケットにアップロードされます。
2. aws_s3 拡張機能は PL/pgSQL を使用してデータにアクセスし、そのデータを Aurora PostgreSQL-Compatible にアップロードします。

ツール

- Amazon Aurora PostgreSQL-Compatible – Amazon Aurora PostgreSQL-Compatible エディションは、フルマネージド型で PostgreSQL 互換の ACID 準拠リレーショナルデータベースエンジンです。ハイエンドの商用データベースのスピードおよび信頼性と、オープンソースデータベースのシンプルさとコスト効率を併せ持っています。
- CLI – AWS コマンドラインインターフェイス (AWS CLI) は、AWS のサービスを管理するための統合ツールです。ダウンロードおよび構成用の単一のツールのみを使用して、コマンドラインから複数の AWS サービスを制御し、スクリプトを使用してこれらを自動化することができます。
- Amazon CloudWatch — Amazon CloudWatch は Amazon S3 のリソースと使用状況をモニタリングします。
- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。このパターンでは、Amazon S3 は Aurora PostgreSQL-Compatible クラスターとの間で使用および送信するファイルを受信および保存するためのストレージレイヤーを提供します。

- _s3 — aws_s3 拡張機能は Amazon S3 と Aurora PostgreSQL-Compatible を統合します。
- 「Amazon SNS」 — Amazon Simple Notification Service (Amazon SNS)は、パブリッシャーやクライアントの間のメッセージ配信や送信を調整および管理します。このパターンでは、Amazon SNS を使用して通知を送信します。
- pgAdmin — pgAdmin は Postgres 用のオープンソース管理ツールです。pgAdmin 4 は、データベースオブジェクトを作成、管理、および使用するためのグラフィカルインターフェイスを提供します。

コード

必要な機能を実現するために、このパターンは UTL_FILE に類似した名前の関数を複数作成します。「追加情報」セクションには、これらの関数のコードベースが含まれています。

コードでは、testaurorabucket をテストの S3 バケットの名前に置き換えます。us-east-1 については、テストの S3 バケットがある AWS リージョンに置き換えます。

エピック

Amazon S3 と Aurora PostgreSQL-Compatible を統合する

タスク	説明	必要なスキル
IAM ポリシーを設定する。	S3 バケットとその中のオブジェクトへのアクセス権を付与する AWS Identity and Access Management (IAM) ポリシーを作成します。コードについては、「追加情報」セクションを参照してください。	AWS 管理者、DBA
Amazon S3 アクセスロールを Aurora PostgreSQL に追加します。	2 つの IAM ロールを作成します。1 つは Amazon S3 への読み取りアクセス用で、もう 1 つは書き込みアクセス用です。2 つのロールを Aurora PostgreSQL-Compatible クラスターにアタッチします。	AWS 管理者、DBA

タスク	説明	必要なスキル
	• S3Export 機能用のロールが 1 つ • S3Import 機能用のロールが 1 つ 詳細については、Amazon S3 へのデータのインポートとエクスポートに関する「Aurora PostgreSQL-Compatible ドキュメント」を参照してください。	

Aurora PostgreSQL-Compatible の拡張機能をセットアップする

タスク	説明	必要なスキル
aws_commons 拡張機能を作成します。	aws_commons 拡張機能は、aws_s3 拡張機能の依存関係です。	DBA、開発者
aws_s3 拡張機能を作成します。	aws_s3 拡張機能は Amazon S3 と相互作用します。	DBA、開発者

Amazon S3 と Aurora PostgreSQL-Compatible の統合を検証する

タスク	説明	必要なスキル
Amazon S3 から Aurora PostgreSQL へのデータインポートをテストする	Aurora PostgreSQL-Compatible へのファイルのインポートをテストするには、サンプル CSV ファイルを作成して S3 バケットにアップロードします。CSV ファイルに	DBA、開発者

タスク	説明	必要なスキル
	基づいてテーブル定義を作成し、aws_s3.table_import_from_s3 関数を使用してファイルをテーブルに読み込みます。	
Aurora PostgreSQL から Amazon S3 へのファイルのエクスポートをテストする。	Aurora PostgreSQL-Compatible からのファイルのエクスポートをテストするには、テストテーブルを作成してデータを入力し、aws_s3.query_export_to_s3 関数を使用してデータをエクスポートします。	DBA、開発者

UTL_FILE ユーティリティを模倣するためにラッパー関数を作成するには

タスク	説明	必要なスキル
utl_file_utility スキーマを作成します。	このスキーマはラッパー関数をまとめて維持します。スキーマを作成するには、次のコマンドを実行します。 `CREATE SCHEMA utl_file_utility;`	DBA、開発者
file_type タイプを作成します。	file_type タイプを作成数には、以下のコードを使用します。 `CREATE TYPE utl_file_utility.file_type AS ( p_path character varying(30),`	DBA/開発者

タスク	説明	必要なスキル
	`p_file_name character varying );`	
init 関数を作成します。	`init` 関数は、`bucket` や `region` などの共通変数を初期化します。コードについては、「追加情報」セクションを参照してください。	DBA/開発者
ラッパー関数を作成する。	ラッパー関数 `fopen`、`put_line`、および `fclose` を作成します。コードについては、「追加情報」セクションを参照してください。	DBA、開発者

ラッパー関数をテストする

タスク	説明	必要なスキル
ラッパー関数を書き込みモードでテストします。	ラッパー関数を書き込みモードでテストするには、「追加情報」セクションに記載されているコードを使用してください。	DBA、開発者
アペンドモードでラッパー関数をテストします。	アペンドモードでラッパー関数をテストするには、「追加情報」セクションに記載されているコードを使用してください。	DBA、開発者

関連リソース

- 「Amazon S3 統合」
- 「Amazon S3」
- Aurora
- Amazon CloudWatch
- Amazon SNS

追加情報

IAM ポリシーを設定する

次のポリシーを作成します。

ポリシー名	JSON
S3IntRead	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "S3integrationtest", "Effect": "Allow", "Action": [ "s3:GetObject", "s3:ListBucket" ], "Resource": [ "arn:aws:s3:::testaurorabucket/*", "arn:aws:s3:::testaurorabucket" ] } ] }`
S3IntWrite	`{ "Version": "2012-10-17",`

```
    "Statement": [
        {
            "Sid": "S3integrationtest
",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",

                "s3:ListBucket"
            ],
            "Resource": [
    "arn:aws:s3:::testaurorabucket/
*",                "arn:aws:s3:::test
aurorabucket"
            ]
        }
    ]
}
```

init 関数を作成する

bucket や region などの共通変数数を初期化するには、次のコードを使用して init 関数を作成します。

```
CREATE OR REPLACE FUNCTION utl_file_utility.init(
    )
    RETURNS void
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
BEGIN
      perform set_config
      ( format( '%s.%s','UTL_FILE_UTILITY', 'region' )
      , 'us-east-1'::text
      , false );

      perform set_config
      ( format( '%s.%s','UTL_FILE_UTILITY', 's3bucket' )
      , 'testaurorabucket'::text
      , false );
```

```
END;
$BODY$;
```

ラッパー関数を作成する

ラッパー関数 fopen、put_line、fclose を作成します。

fopen

```
CREATE OR REPLACE FUNCTION utl_file_utility.fopen(
    p_file_name character varying,
    p_path character varying,
    p_mode character DEFAULT 'W'::bpchar,
    OUT p_file_type utl_file_utility.file_type)
    RETURNS utl_file_utility.file_type
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
declare
    v_sql character varying;
    v_cnt_stat integer;
    v_cnt integer;
    v_tabname character varying;
    v_filewithpath character varying;
    v_region character varying;
    v_bucket character varying;

BEGIN
    /*initialize common variable */
    PERFORM utl_file_utility.init();
    v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
    v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

    /* set tabname*/
    v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );
    v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;
    raise notice 'v_bucket %, v_filewithpath % , v_region %', v_bucket,v_filewithpath,
 v_region;
```

```
   /* APPEND MODE HANDLING; RETURN EXISTING FILE DETAILS IF PRESENT ELSE CREATE AN
EMPTY FILE */
   IF p_mode = 'A' THEN
       v_sql := concat_ws('','create temp table if not exists ', v_tabname,' (col1
text)');
       execute v_sql;

       begin
       PERFORM aws_s3.table_import_from_s3
           ( v_tabname,
           '',
           'DELIMITER AS ''#''',
           aws_commons.create_s3_uri
           (     v_bucket,
               v_filewithpath ,
               v_region)
           );
       exception
           when others then
            raise notice 'File load issue ,%',sqlerrm;
            raise;
       end;
       execute concat_ws('','select count(*) from ',v_tabname) into v_cnt;

       IF v_cnt > 0
       then
           p_file_type.p_path := p_path;
           p_file_type.p_file_name := p_file_name;
       else
           PERFORM aws_s3.query_export_to_s3('select ''''',
                           aws_commons.create_s3_uri(v_bucket, v_filewithpath,
v_region)
                             );

           p_file_type.p_path := p_path;
           p_file_type.p_file_name := p_file_name;
       end if;
       v_sql := concat_ws('','drop table ', v_tabname);
       execute v_sql;
   ELSEIF p_mode = 'W' THEN
           PERFORM aws_s3.query_export_to_s3('select ''''',
                           aws_commons.create_s3_uri(v_bucket, v_filewithpath,
v_region)
                             );
```

```
            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
    END IF;

EXCEPTION
        when others then
            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
            raise notice 'fopenerror,%',sqlerrm;
            raise;
END;
$BODY$;
```

put_line

```
CREATE OR REPLACE FUNCTION utl_file_utility.put_line(
    p_file_name character varying,
    p_path character varying,
    p_line text,
    p_flag character DEFAULT 'W'::bpchar)
    RETURNS boolean
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
/*****************************************************************************
* Write line, p_line in windows format to file, p_fp - with carriage return
* added before new line.
*****************************************************************************/
declare
    v_sql varchar;
    v_ins_sql varchar;
    v_cnt INTEGER;
    v_filewithpath character varying;
    v_tabname  character varying;
    v_bucket character varying;
    v_region character varying;

BEGIN
 PERFORM utl_file_utility.init();

/* check if temp table already exist */
```

```
 v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );

 v_sql := concat_ws('','select count(1) FROM pg_catalog.pg_class c LEFT JOIN
 pg_catalog.pg_namespace n ON n.oid = c.relnamespace where n.nspname like ''pg_temp_
%'''
                        ,' AND pg_catalog.pg_table_is_visible(c.oid) AND
 Upper(relname) = Upper( '''
                        ,  v_tabname ,''' ) ');

 execute v_sql into v_cnt;

  IF v_cnt = 0 THEN
         v_sql := concat_ws('','create temp table ',v_tabname,' (col text)');
        execute v_sql;
        /* CHECK IF APPEND MODE */
        IF upper(p_flag) = 'A' THEN
            PERFORM utl_file_utility.init();
            v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY',
 'region' ) );
            v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY',
 's3bucket' ) );

            /* set tabname*/
            v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;

            begin
               PERFORM aws_s3.table_import_from_s3
                     ( v_tabname,
                          '',
                       'DELIMITER AS ''#''',
                        aws_commons.create_s3_uri
                           ( v_bucket,
                               v_filewithpath,
                               v_region    )
                    );
            exception
                when others then
                    raise notice  'Error Message : %',sqlerrm;
                    raise;
            end;
        END IF;
```

```
    END IF;
    /* INSERT INTO TEMP TABLE */
    v_ins_sql := concat_ws('','insert into ',v_tabname,' values(''',p_line,''')');
    execute v_ins_sql;
    RETURN TRUE;
    exception
            when others then
                raise notice  'Error Message : %',sqlerrm;
                raise;
END;
$BODY$;
```

fclose

```
CREATE OR REPLACE FUNCTION utl_file_utility.fclose(
    p_file_name character varying,
    p_path character varying)
    RETURNS boolean
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
DECLARE
    v_filewithpath character varying;
    v_bucket character varying;
    v_region character varying;
    v_tabname character varying;
    v_sql character varying;
BEGIN
      PERFORM utl_file_utility.init();

    v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
    v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

    v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );
    v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;

    raise notice 'v_bucket %, v_filewithpath % , v_region %', v_bucket,v_filewithpath,
 v_region ;
```

```
    /* exporting to s3 */
    perform aws_s3.query_export_to_s3
        (concat_ws('','select * from ',v_tabname,'  order by ctid asc'),
            aws_commons.create_s3_uri(v_bucket, v_filewithpath, v_region)
        );
    v_sql := concat_ws('','drop table ', v_tabname);
    execute v_sql;
    RETURN TRUE;
EXCEPTION
       when others then
     raise notice 'error fclose %',sqlerrm;
     RAISE;
END;
$BODY$;
```

設定とラッパー関数をテストする

次の匿名コードブロックを使用して、設定をテストします。

書き込みモードをテストする

次のコードは、S3 バケットの s3inttest という名前のファイルを記述します。

```
do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
l_mode char(1) := 'W';
l_fs utl_file_utility.file_type ;
l_status boolean;

begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;

select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for
 test purpose', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from utl_file_utility.fclose( l_file_name , l_path ) into l_status ;
raise notice 'fclose : l_status %', l_status;
```

```
end;
$$
```

アペンドモードをテストする

次のコードは、前のテストで作成した s3inttest ファイルに行を追加します。

```
do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
l_mode char(1) := 'A';
l_fs utl_file_utility.file_type ;
l_status boolean;

begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;


select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for
 test purpose : append 1', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket : for
 test purpose : append 2', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from utl_file_utility.fclose( l_file_name , l_path ) into l_status ;
raise notice 'fclose : l_status %', l_status;

end;
$$
```

Amazon SNSの通知

オプションで Amazon CloudWatch によるモニタリングと Amazon SNS 通知を S3 バケットに設定できます。詳細については、「Amazon S3 をモニタリングする」と「Amazon SNS 通知のセットアップ」を参照してください。

Oracle から Amazon Aurora PostgreSQL への移行後にデータベースオブジェクトを検証する

ベンカトラマナ・チンサ（AWS）とエドゥアルド・ヴァレンティム（AWS）によって作成された

概要

このパターンは、Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行した後にオブジェクトを検証する段階的なアプローチを示しています。

このパターンは、データベースオブジェクト検証の使用シナリオと手順の概要を示しています。詳細については、「AWS Database ブログ」の「AWS SCT と AWS DMS を使用した移行後のデータベースオブジェクトの検証」を参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Aurora PostgreSQL 互換データベースに移行された、オンプレミスの Oracle データベース。
- Aurora PostgreSQL 互換データベースに「AmazonRDSDataFullAccess」ポリシーが適用されているサインイン認証情報。
- このパターンでは、Amazon Relational Database Service (Amazon RDS) コンソールにある「Aurora サーバーレス DB クラスターのクエリエディター」を使用します。ただし、このパターンは他のクエリエディタでも使用可能です。

制約事項

- Oracle SYNONYM オブジェクトは PostgreSQL では使用できませんが、「ビュー」または SET search_path クエリによって部分的に検証できます。
- Amazon RDS クエリエディタは、「特定の AWS リージョン、特定の MySQL および PostgreSQL バージョン」でのみ使用できます

アーキテクチャ

ツール

ツール

- Amazon Aurora PostgreSQL-Compatible Edition – Amazon Aurora PostgreSQL は、フルマネージド型で PostgreSQL 互換の、ACID 準拠のリレーショナルデータベースエンジンです。ハイエンドの商用データベースのスピードと信頼性を、オープンソースデータベースのシンプルさとコスト効率でご利用いただけます。
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベースをセットアップ、運用、スケーリングできます。業界スタンダードのリレーショナルデータベース向けに、費用対効果に優れたエクステンションを備え、一般的なデータベース管理タスクを管理します。
- Aurora Severless 用クエリエディタ – クエリエディタを使用すると、Amazon RDS コンソールで SQL クエリを実行できます。Auroraサーバーレス DB クラスターでは、データ操作やデータ定義のステートメントも含めて、任意の有効な SQL ステートメントを実行できます。

オブジェクトを検証するには、「添付ファイル」セクションの「オブジェクト検証スクリプト」ファイルにあるフルスクリプトを使用します。次の表を参照してください。

Oracle オブジェクト	使用するスクリプト
パッケージ	Query 1
テーブル	Query 3
ビュー	クエリ 5
シーケンス	クエリ 7
トリガー	クエリ 9
プライマリキー	クエリ 11
インデックス	クエリ 13
検査制約	クエリ 15
外部キー	クエリ 17

PostgreSQL オブジェクト	使用するスクリプト
パッケージ	Query 2
テーブル	クエリ 4
ビュー	クエリ 6
シーケンス	クエリ 8
トリガー	クエリ 10
プライマリキー	クエリ 12
インデックス	クエリ 14
検査制約	クエリ 16
外部キー	クエリ 18

エピック

ソース Oracle データベース内のオブジェクトを検証します

タスク	説明	必要なスキル
ソース Oracle データベースで「パッケージ」検証クエリを実行します。	「添付ファイル」セクションから「オブジェクト検証スクリプト」ファイルをダウンロードして開きます。クライアントプログラムを使用してソース Oracle データベース Connect。「オブジェクト検証スクリプト」ファイルから「Query 1」検証スクリプトを実行します。重要：クエリには「your_schema」の代わりに Oracle ユーザー名を入力し	開発者、DBA

タスク	説明	必要なスキル
	てください。クエリの結果は必ず記録してください。	
「テーブル」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 3」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「ビュー」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 5」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「シーケンス」カウント検証を実行します。	「オブジェクト検証スクリプト」ファイルから「Query 7」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「トリガー」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 9」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「プライマリーキー」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 11」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA

タスク	説明	必要なスキル
「インデックス」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 13」検証スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「チェック制約」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 15」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「外部キー」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 17」検証スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA

ターゲット Aurora PostgreSQL 互換データベース内のオブジェクトを検証します

タスク	説明	必要なスキル
クエリエディタを使用して、対象のAurora PostgreSQL互換データベースに接続します。	AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。右上で、Aurora PostgreSQL 互換データベースを作成した AWS リージョンを選択します。ナビゲーションペインで、「データベース」を選択して、ターゲットの Aurora PostgreSQL 互換データベースを選択します。[アクション]、[クエリ] の順に選択します。重要：まだデータベー	開発者、DBA

タスク	説明	必要なスキル
	スに接続していない場合は、[Connect to database (データベースに接続)] ページが開きます。次に、ユーザー名やパスワードなどのデータベース情報を入力する必要があります。	
「パッケージ」検証クエリを実行します。	「添付ファイル」セクションの「オブジェクト検証スクリプト」ファイルから「Query 2」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「テーブル」検証クエリを実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 4」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「ビュー」検証クエリを実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 6」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA

タスク	説明	必要なスキル
「シーケンス」カウント検証を実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 8」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「トリガー」検証クエリを実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 10」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「プライマリーキー」検証クエリを実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 12」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「インデックス」検証クエリを実行します。	Aurora PostgreSQL 互換データベースのクエリエディタに戻り、「オブジェクト検証スクリプト」ファイルから「Query 14」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA

タスク	説明	必要なスキル
「チェック制約」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 16」スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA
「外部キー」検証クエリを実行します。	「オブジェクト検証スクリプト」ファイルから「Query 18」検証スクリプトを実行します。クエリの結果は必ず記録してください。	開発者、DBA

ソースとターゲットの検証レコードを比較

タスク	説明	必要なスキル
両方のクエリ結果を比較して検証します。	Oracle と Aurora PostgreSQL 互換データベースのクエリ結果を比較して、すべてのオブジェクトを検証します。すべてが一致すれば、すべてのオブジェクトは正常に検証されたことになります。	開発者、DBA

関連リソース

- AWS SCT と AWS DMS を使用した移行後のデータベースオブジェクトの検証
- Amazon Aurora Features: PostgreSQL-Compatible Edition

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

リホスト

トピック

- Microsoft ワークロードの検出と AWS への移行を加速する
- Windows 上の AWS Managed Services の事前ワークロード取り込みアクティビティを自動化する
- へのリホスト移行中にファイアウォールリクエストの承認プロセスを作成する AWS
- EC2 Windows インスタンスを AWS Managed Services アカウントに取り込み、移行
- Couchbase Server データベースを Amazon EC2 に移行する
- ログ配信を使用して Db2 for LUW を Amazon EC2 に移行することで、システム停止時間を短縮する
- 高可用性ディザスタリカバリ機能を備えた Db2 for LUW を Amazon EC2 に移行する
- PowerCLI を使用して HCX オートメーションで VMware 仮想マシンを移行
- F5 BIG-IPワークロードをAWS クラウド上のF5 BIG-IP VEに移行する
- バイナリメソッドを使用してオンプレミスの Go ウェブアプリケーションを AWS Elastic Beanstalk に移行します
- AWS を使用してオンプレミス SFTP サーバーを に移行する AWS Transfer for SFTP
- AWS アプリケーション移行サービスを使用してオンプレミス VM を Amazon EC2 に移行する
- AWS SFTP を使用して小規模なデータセットをオンプレミスから Amazon S3 に移行する
- Oracle GlassFish から AWS Elastic Beanstalk へ移行する
- オンプレミスの Oracle データベースを Oracle Amazon EC2 に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon EC2 に移行する
- AWS MGN を使用して RHEL BYOL システムを AWS ライセンス込みのインスタンスに移行する
- オンプレミスの SAP ASE データベースを Amazon EC2 に移行
- オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する
- オンプレミス MySQL データベースを Amazon EC2 に移行する
- アプリケーション移行サービスを使用することで、同種の SAP 移行のカットオーバー時間を短縮する
- AWS Cloud でオンプレミスワークロードをリホストする: 移行チェックリスト
- Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする
- BMC ディスカバリークエリを使用して移行計画のために移行データを抽出

Microsoft ワークロードの検出と AWS への移行を加速する

作成者: Ali Alzand

概要

このパターンは、Migration Validator Toolkit PowerShell モジュールを使用して Microsoft ワークロードを検出し、AWS に移行する方法を示しています。このモジュールは、あらゆる Microsoft ワークロードに関連する一般的なタスクに対して複数のチェックと検証を実行することで機能します。例えば、モジュールは複数のディスクがアタッチされている可能性のあるインスタンスや、多くの IP アドレスを使用するインスタンスをチェックします。モジュールが実行できるチェックの完全なリストについては、モジュールの GitHub ページの「チェック」セクションを参照してください。

移行検証ツールキット PowerShell モジュールを使用すると、組織は Microsoft ワークロードで実行されているアプリケーションやサービスの検出にかかる時間と労力を削減できます。このモジュールは、ワークロードの設定を特定するのにも役立ちます。これにより、その設定が AWS でサポートされているかどうかを確認できます。また、このモジュールでは、移行前、移行中、移行後に設定ミスを回避できるよう、次のステップや緩和策に関する推奨事項も提供しています。

前提条件と制限

前提条件

- ローカル管理者アカウント
- PowerShell 4.0

制約事項

- Microsoft Windows Server 2012 R2 以降でのみ動作

ツール

ツール

- PowerShell 4.0

コードリポジトリ

このパターンの Migration Validator Toolkit PowerShell モジュールは、GitHubmigration-validator-toolkit-for-microsoft-workloads で入手できます。

エピック

1 つのターゲットで Migration Validator Toolkit PowerShell モジュールを実行する

タスク	説明	必要なスキル

モジュールをダウンロード、抽出、インポート、および呼び出します。	モジュールをダウンロードしてデプロイするには、次のいずれかの方法を選択します。
	• PowerShell スクリプトを実行する
	• .zip ファイルをダウンロードして抽出する
	• GitHub リポジトリのクローンを作成する
	
	PowerShell スクリプトを実行する
	
	PowerShell で、次のサンプルコードを実行します。
	
	`#MigrationValidatorToolkit`
	`$uri = 'https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads/archive/refs/heads/main.zip'`
	`$destination = (Get-Location).Path`
	`if ((Test-Path -Path "$destination\MigrationValidatorToolkit.zip" -PathType Leaf) -or (Test-Path -Path "$destination\Migr`
		システム管理者

タスク	説明	必要なスキル

	ationValidatorTool
	kit")) {
	    write-host "File
	 $destination\Migra
	tionValidatorToolk
	it.zip or folder
	 $destination\Migra
	tionValidatorToolkit
	 found, exiting"
	}else {
	    Write-host "Enable
	 TLS 1.2 for this
	 PowerShell session
	 only."
	    [Net.ServicePointM
	anager]::SecurityP
	rotocol = [Net.Secu
	rityProtocolType]:
	:Tls12
	    $webClient =
	 New-Object System.Ne
	t.WebClient
	    Write-host
	 "Downloading Migration
	ValidatorToolkit.zip"
	    $webClient.Downloa
	dFile($uri, "$destina
	tion\MigrationVali
	datorToolkit.zip")
	    Write-host
	 "MigrationValidato
	rToolkit.zip download
	 successfully"
	    Add-Type -Assembly
	 "system.io.compres
	sion.filesystem"
	    [System.IO.Compres
	sion.ZipFile]::Ext
	ractToDirectory("$
	destination\Migrat
	ionValidatorToolki
	t.zip","$destinati	

タスク	説明	必要なスキル

	<pre>on\MigrationValida
	torToolkit")
	    Write-host
	 "Extracting Migration
	ValidatorToolkit.zip
	 complete successfully"
	    Import-Module
	 "$destination\Migr
	ationValidatorToolkit
	\migration-validator-
	toolkit-for-microsoft
	-workloads-main\Mi
	grationValidatorTo
	olkit.psm1"; Invoke-
	MigrationValidatorTo
	olkit
	}</pre>
	このコードは、.zip ファイルからモジュールをダウンロードします。次に、コードはモジュールを抽出、インポート、呼び出します。
	
	.zip ファイルをダウンロードして抽出する
	
	1. .zip ファイルをダウンロードします (ダウンロード) 。
	2. .zip ファイルを展開します。
	3. このガイドの「モジュールを手動で呼び出す」のステップに従います。	

タスク	説明	必要なスキル
	GitHub リポジトリのクローンを作成する 1. GitHubmigration-validator-toolkit-for-microsoft-workloads のクローンを作成するには、ターミナルウィンドウで次の Git コマンドを実行します。 ``` git clone https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads.git ``` 2. このガイドの「モジュールを手動で呼び出す」のステップに従います。	

タスク	説明	必要なスキル

モジュールを手動で呼び出します。	1. ダウンロードしたモジュールが保存されているディレクトリに移動します。
	2. 選択した出力を生成するには、PowerShell で管理者として次のいずれかのコマンドを実行します。
	
	形式 - テーブル形式：
	
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit`
	
	Format-Listformat:
	
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit -List`
	
	Out-GridView 形式：
	
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit -GridView`
	
	ConvertTo-Csvformat：
	
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-Migra`
		システム管理者

タスク	説明	必要なスキル
	`tionValidatorToolkit -csv`	

複数のターゲットで Migration Validator Toolkit PowerShell モジュールを実行する

タスク	説明	必要なスキル
.zip ファイルをダウンロードするか、GitHub リポジトリのクローンを作成します。	以下のオプションのいずれかを選択してください： • zip ファイルをダウンロードします (ダウンロード) 。 • GitHubmigration-validator-toolkit-for-microsoft-workloads のクローンを作成するには、ターミナルウィンドウで次の Git コマンドを実行します。 `git clone https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads.git`	システム管理者
server.csv リストを更新します。	.zip ファイルをダウンロードした場合は、次の手順に従います。 1. .zip ファイルを展開します。	システム管理者

タスク	説明	必要なスキル
	2. MigrationValidator Toolkit\Inputs\ ディレクトリに移動します。 3. ターゲットコンピュータのホスト名serverlist.csv で を更新します。	
モジュールを呼び出します。	ターゲットコンピュータへの管理者アクセス権を持つドメインユーザーを使用するドメイン内の任意のコンピュータを使用できます。 1. ソースコードを .zip ファイルとしてダウンロードし、ファイルを抽出します。 2. PowerShell の管理者として、次のコマンドを実行します。 `Import-Module .\MigrationValidatorToolkit.psm1;Invoke-DomainComputers` 出力 .csv ファイルは、プレフィックス名で にMigrationValidatorToolkit\Outputs\folder 保存されますDomainComputers_MigrationAutomations_YYYY-MM-DDTHH-MM-SS 。	システム管理者

トラブルシューティング

問題	ソリューション
`MigrationValidatorToolkit` は、実行、コマンド、エラーに関する情報を実行中のホストのログファイルに書き込みます。	ログファイルは、次の場所で手動で表示できます。 1. `MigrationValidatorToolkit\logs\` ディレクトリに移動します。 2. ログファイルを見つけます。ログファイル名の形式は次のとおりです。 `ComputerName_MigrationValidatorToolkit_YYYY-MM-SSTHH-MM-SS.log`

関連リソース

- Microsoft ワークロードを AWS に移行するためのオプション、ツール、ベストプラクティス (AWS 規範ガイダンス)
- Microsoft 移行パターン (AWS 規範ガイダンス)
- AWS での無料のクラウド移行サービス (AWS ドキュメント)
- 事前定義された起動後のアクション (アプリケーションマーケティングドキュメント)

追加情報

よくある質問

Migration Validator Toolkit PowerShell モジュールはどこで実行できますか？

モジュールは、Microsoft Windows Server 2012 R2 以降で実行できます。

このモジュールはいつ実行すればよいですか？

移行ジャーニーの評価段階でモジュールを実行することをお勧めします。

モジュールは既存のサーバーを変更しますか？

いいえ。このモジュールのすべてのアクションは読み取り専用です。

モジュールの実行にはどのくらいの時間がかかりますか？

通常、モジュールの実行には 1～5 分かかりますが、サーバーのリソース割り当てによって異なります。

モジュールを実行するにはどのようなアクセス許可が必要ですか？

ローカル管理者アカウントからモジュールを実行する必要があります。

モジュールを物理サーバーで実行できますか？

はい。オペレーティングシステムが Microsoft Windows Server 2012 R2 以降である限り、可能です。

複数のサーバーでモジュールを大規模に実行するにはどうすればよいですか？

複数のドメインに参加しているコンピュータでモジュールを大規模に実行するには、このガイドの「Run the Migration Validator Toolkit PowerShell module on multiple targets epic」のステップに従います。ドメインに参加していないコンピュータの場合は、リモート呼び出しを使用するか、このガイドの 1 つのターゲットエピックの「移行検証ツールキット PowerShell モジュールを実行する」の手順に従ってモジュールをローカルで実行します。

Windows 上の AWS Managed Services の事前ワークロード取り込みアクティビティを自動化する

作成者: Jacob Zhang (AWS)、Calvin Yeh (AWS)、Dwayne Bordelon (AWS)

概要

Amazon Web Services（AWS）クラウドでは、AWS Managed Services（AMS）は AMS ワークロードインジェスト（WIGS）を使用して既存のワークロードを AMS マネージド VPC に移動します。このパターンは、.NET と Windows PowerShell のアップグレード、AMS が管理する Windows WIGS の取り込み前検証の実行など、一般的な事前ワークロード取り込みアクティビティを自動化するソリューションを説明しています。このパターンでは、実行結果の統一ユーザーインターフェイスも提供されます。事前取り込みアクティビティを実行する AWS Systems Manager コマンドドキュメントを、AWS CloudFormation テンプレートにパッケージ化します。テンプレートは、Systems Manager 自体にアクセスする必要なく、または AMS からの自動化と競合することなく、繰り返しデプロイできます。

ビジネスバックグラウンド

AMS への移行では、AMS コンポーネントを含む AMS マネージド Amazon マシンイメージ (AMI) を使用して、新しい Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをプロビジョニングする必要があります。既存のデータセンターで実行中のワークロードまたはアプリケーションはすべて、これらの AMS AMI から起動したフレッシュな EC2 インスタンスに再デプロイする必要があります。プロセス中に大量の手作業が発生する可能性を避けるため、AMS チームはカスタムイメージを AMS にオンボードする AMS ワークロード取り込み (WIGS) ワークフローを構築しました。

Windows インスタンスは、WIGS プロセスを実行する前に、いくつかの前提条件を満たす必要があります。通常、Windows PowerShell スクリプトを使用して、必要な準備 (WIGS 準備) を実行し、インスタンスが WIG の準備ができているかどうかを確認します (WIGS の取り込み前検証)。準備と検証のプロセスでは、エンジニアは各サーバーで 15～30 分かけて手動でログインし、スクリプトを 1 つずつ実行する必要があります。

ビジネスドライバー

従来は、Systems Manager を使用すると、Windows PowerShell スクリプトの実行などの運用タスクを自動化できます。ただし、リスクが高く、AMS の自動化とユーザーの自動化との間で頻繁に競合が発生するため、通常、AMS はユーザーに Systems Manager へのアクセス権を付与しません。

AWS Application Migration Service (AWS MGN) を使用した一括移行の場合、`C:\Program Files (x86)\AWS Replication Agent\post_launch folder` の Windows PowerShell スクリプトは

通常、テストインスタンスまたはカットオーバーインスタンスの起動時に自動的に実行されます。ただし、これらのスクリプトをインスタンスの起動時にすぐに実行すると、AMS の自動化と競合することがよくあります。その結果、障害のトラブルシューティングに必要な実行結果が得られずに起動が失敗する可能性があります。

このパターンはこれらの問題に対処し、実用的な自動化ソリューションを提供します。

前提条件と制限

前提条件

- AMS オンボーディングのアクティブな AWS アカウントが完了している
- AWS アカウントの Amazon Simple Storage Service (Amazon S3) バケット アカウントに管理する S3 バケットがない場合は、変更要求 (RFC) を使用して作成してください。
- Prewigs_CFN.json テンプレートが ams-auto-prewigs-windows リポジトリからダウンロードされている
- このパターンを適用するサーバーは、以下の要件を満たしている必要があります。
  - Windows Server 2012 以降を実行
  - サンドボックス VPC 移行サブネットで起動、または起動する準備ができている
  - AWS Systems Manager Agent (SSM Agent) がインストールされている
  - AWS Identity and Access Management (IAM) インスタンスプロファイルがアタッチされているインスタンスプロファイルには、同じ AWS アカウントの S3 バケットからファイルをダウンロードする権限がある必要があります。上記の要件を満たすインスタンスプロファイルは、通常、移行の初期設定時にすでに確立されています。
  - AWS Systems Manager Fleet Manager から閲覧できる

制限

- 事前 WIGS アクティビティは、環境やビジネス要件に応じて異なります。このパターンを若干変更して、特定のニーズに合わせる必要がある場合があります。

製品バージョン

- パターンは、Windows Server 2012、2012 R2、2016、2019 でテストされます。理論的には、新しい Windows バージョンでも動作します。これより前のバージョンの Windows では動作しません。

アーキテクチャ

以下は、アーキテクチャ図を示しています。

1. 準備されていないサーバーを含む移行サブネットがあるサンドボックス VPC。
2. CloudFormation テンプレートで使用されるスクリプトを格納する S3 バケット。
3. CloudFormation テンプレートは、Systems Manager コマンドドキュメントをデプロイします。このプロセスは手順が完了するまで繰り返されます。
4. インスタンスが準備され、WIGS の RFC が作成されます。
5. AMS マネージド VPC では、AMS マネージドサブネットにはワークロード取り込み後のサーバーが含まれます。

仕組み

- このパターンは、AWS CloudFormation テンプレートにパッケージ化されており、Infrastructure as Code (IaC) の繰り返し可能なデプロイを可能になります。このテンプレートは、この自動化が必要な各 AWS アカウントに 1 回だけデプロイする必要があります。
- 自動化は、このパターンがデプロイされている AWS アカウントの AutoPreWIGs というタグキーがあるすべての EC2 インスタンスに適用されます。タグキー AutoPreWIGs がある Amazon EC2 Windows インスタンスを初めて起動すると、自動化で次のタスクが実行されます。
  1. Windows PowerShell をバージョン 5.1 に、.NET をバージョン 4.5.2 にアップグレード。インスタンスは、既存の Windows PowerShell と.NET のバージョンに応じて、複数回再起動する場合があります。各再起動後に、アップグレードは完了するまで続行されます。この手順では、Windows PowerShell スクリプトから変更された CloudFormation テンプレートの埋め込みコードと、サーバーの再起動に関する特定のSystems Manager ガイダンスを使用します。
  2. Amazon S3 からダウンロードし、カスタマイズした Windows PowerShell スクリプトを実行して、WIGS 用の Amazon EC2 Windows インスタンスを準備します。詳細については、エピックセクションを参照してください。
  3. Windows WIGS の取り込み前検証 PowerShell モジュールを AWS からインストールします。
  4. Windows WIGS の取り込み前検証を実行し、その結果を Systems Manager の State Manager で表示できるようにします。

ツール

- AWS CloudFormation — AWS CloudFormation は、AWS リソースをモデル化してセットアップするサービスです。必要なすべての AWS リソースとその依存関係を説明したものを使用して、それらのリソースをスタックとして起動して設定できます。このパターンでは、CloudFormation テンプレートを使用して、このパターンのリソースのデプロイを自動化します。
- AWS Managed Services — AWS Managed Services (AMS) は、AWS インフラストラクチャの継続的な管理を提供するエンタープライズサービスです。AMS 環境でのインフラストラクチャの変更は、RFC で行う必要があります。
- AWS Systems Manager – AWS Systems Manager (旧称 SSM) は、 AWS でインフラストラクチャの表示と制御に使用できる AWS サービスです。Systems Manager コンソールを使用すると、複数の AWS サービスからの運用データを表示し、AWS リソース全体の運用タスクを自動化できます。このパターンでは、Systems Manager を使用して WIGS 前のアクティビティを実行および実行結果を表示します。
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、およびパフォーマンスを提供するオブジェクトストレージサービスです。このパターンでは、Amazon S3 を使用して CloudFormation テンプレートとダウンロードした Windows PowerShell スクリプトを保存します。

エピック

カスタム Windows PowerShell スクリプトを作成して、追加タスクを自動化する

タスク	説明	必要なスキル
ビジネスニーズに基づきサーバーに必要な変更をします。	取り込まれる前に変更をサーバーに自動的に適用する必要がある場合は、`ingestion-prep.ps1` という Windows PowerShell スクリプトを作成します。 **Important** スクリプトには、サーバーを再起動する手順を含めることはできま	PowerShell スクリプト

タスク	説明	必要なスキル
	せん。また、管理者権限も必要ありません。	
AMS でサポートされていないソフトウェアは削除してください。	AMS では、WIGS を実行する前に、ウイルス対策アプリケーションや VMware Tools などの特定のソフトウェアを削除する必要があります。アンインストールを `ingestion-prep.ps1` スクリプトに含めてください。サポートされていないソフトウェアの詳細については、AWS ドキュメントを参照してください。	PowerShell スクリプト

CloudFormation テンプレートとオプションの Windows PowerShell スクリプトを Amazon S3 にアップロードする

タスク	説明	必要なスキル
S3 にフォルダを作成します。	このパターンをデプロイする場所と同じ AWS アカウントの S3 バケットに、フォルダを作成します。	AWS 全般
スクリプトをアップロードします。	前のエピックで作成した `PreWIGs_CFN.json` CloudFormation テンプレートと `ingestion-prep.ps1` Windows PowerShell スクリプトを Amazon S3 フォルダにアップロードします。	AWS 全般

CloudFormation スタックをデプロイする

タスク	説明	必要なスキル
変更タイプを選択します。	AMS コンソールにナビゲートして、 RFC を作成します。CloudFormation (CFN) テンプレートからスタックを作成変更タイプを使用します。	AWS 全般
CloudFormation テンプレートへのパスの実行パラメータを設定します。	実行設定 セクションで、追加設定 を展開します。CloudFormation テンプレート S3 エンドポイントボックスで、URL を CloudFormation テンプレートに貼り付けます。	AWS 全般
Amazon S3 フォルダへのパスを指定します。	パラメータで、ScriptSource を名前として使用します。値の場合は、Windows PowerShell スクリプトを含む S3 フォルダへのパスを入力します。必ず `s3://xxx` URI の代わりに `https://xxx` URL を使用し、末尾には / を含めてください。	AWS 全般
スタックをデプロイします。	スタックをデプロイするには、[Create (作成)] を選択します。	AWS 全般
RFC を AMS オペレーションにエスカレーションします。	RFC は、Systems Manager を使用してリソースをデプロイし、セキュリティレビューをする必要があるため、AMS Ops チームが手動で実装する必要があります。RFC を	AWS 全般

タスク	説明	必要なスキル
	作成するとすぐに、システムによって自動的に拒否されます。RFC を選択し、手動で実行してくださいという対応を RFC に追加します。RFC ID を書き留め、サービスリクエストとともにエスカレーションします。	

自動化をインスタンスに適用する

タスク	説明	必要なスキル
AutoPrewigs タグをインスタンスに追加します。	この自動化を適用するすべてのインスタンスの ID を書き留め、AMS が実装した自動化がインスタンスを終了するまで少なくとも 30 分待機します。自動 RFC を送信して、キーとして AutoPreWIGs でタグを追加し、値として 1 などの文字列を追加します。 自動化は、タグを追加してから数分後に適用されます。	AWS 全般
自動化の結果を検証します。	AWS Systems Manager コンソールを開き、[Fleet Manager] を選択します。AMS-PreWIG-Prep-and-Validation-Association という名前の [Association ID (アソシエーション ID)] を選択します。[Execution history (実	AWS 全般

タスク	説明	必要なスキル
	行履歴)] タブでは、自動化の結果を確認できます。	
エラーを修正します。	自動化が失敗した場合は、[Execution ID (実行 ID)] を選択します。各 EC2 インスタンスの実行結果を確認できます。自動化の各ステップの詳細を表示するには、[Output (出力)] を選択します。特定の手順に失敗する場合は、[Output (出力)] セクションと [Error (エラー)] セクションの情報を使用して問題を診断します。	移行エンジニア
AutoPrewigs タグを削除します。	**Important** エラーを修正したら、自動 RFC を送信して AutoPreWIGs タグを削除します。タグを削除しないと、WIGS は失敗します。	AWS 全般

準備したインスタンスを取り込む

タスク	説明	必要なスキル
WIGS のRFCを送信します。	インスタンスがワークロードを取り込む準備ができたため、WIGS の RFC を送信します。	AWS 全般

関連リソース

- AMS ワークロード取り込み (WIGS)
- ワークロードの移行: Windows の取り込み前検証
- AWS アプリケーション移行サービスのクイックスタートガイド
- AWS CloudFormation の使用開始
- AWS Systems Manager のセットアップ

へのリホスト移行中にファイアウォールリクエストの承認プロセスを作成する AWS

作成者：Srikanth Rangavajhala (AWS)

概要

へのリホスト移行に で AWS Application Migration Serviceまたは Cloud Migration Factory AWS を使用する場合は AWS クラウド、TCP ポート 443 と 1500 を開いたままにしておくことが前提条件の 1 つです。通常、これらのファイアウォールポートを開くには、情報セキュリティ (InfoSec) チームによる承認が必要です。

このパターンは、 へのリホスト移行中に InfoSec チームからファイアウォールリクエストの承認を取得するプロセスの概要を示しています AWS クラウド。このプロセスにより、コストと時間がかかる可能性のある InfoSec チームによるファイアウォールリクエストの拒否を回避できます。ファイアウォールリクエストプロセスには、 AWS 移行コンサルタントと、InfoSec およびアプリケーションチームと協力してファイアウォールポートを開くリーダーの間で、2 つのレビューと承認のステップがあります。

このパターンは、組織の AWS コンサルタントまたは移行スペシャリストによるリホスト移行を計画していることを前提としています。このパターンは、組織にファイアウォール承認プロセスやファイアウォールリクエストの一括承認フォームがない場合に使用できます。詳細については、このパターンの [制限事項] セクションを参照してください。Application Migration Service のネットワーク要件の詳細については、「Application Migration Service のドキュメント」の「ネットワークの要件」を参照してください。

前提条件と制限

前提条件

- 組織の AWS コンサルタントまたは移行スペシャリストによる計画的なリホスト移行
- スタックの移行に必要なポートと IP 情報
- 現在と未来の状態のアーキテクチャ図
- オンプレミスと宛先のインフラストラクチャ、ポート、ゾーン間のトラフィックフローに関するファイアウォール情報
- ファイアウォールリクエストのレビューチェックリスト (添付)
- 組織の要件により構成されたファイアウォール申請書類
- 以下のロールを含むファイアウォールのレビュー担当者と承認者の連絡先リスト。
  - ファイアウォールリクエスト送信者 – AWS 移行スペシャリストまたはコンサルタント。ファイアウォールリクエスト送信者は、組織の移行スペシャリストでもかまいません。

- ファイアウォールリクエストレビューワー – 通常、これは単一連絡先 (SPOC) の発信元です AWS。
- ファイアウォールリクエスト承認者 — InfoSec チームメンバー。

機能制限

- このパターンは、一般的なファイアウォールリクエスト承認プロセスを表しています。要件は組織によって異なる場合があります。
- ファイアウォールリクエストのドキュメントの変更を必ず追跡してください。

次の表に、このパターンの使用例を示します。

あなたの組織には既存のファイアウォール承認プロセスがありますか?	あなたの組織には既存のファイアウォール申請書がありますか?	推奨されるアクション
あり	あり	AWS コンサルタントまたは移行スペシャリストと協力して、組織のプロセスを実装します。
いいえ	はい	このパターンのファイアウォール承認プロセスを使用します。組織の AWS コンサルタントまたは移行スペシャリストを使用して、ファイアウォールリクエストの一括承認フォームを送信します。
いいえ	いいえ	このパターンのファイアウォール承認プロセスを使用します。組織の AWS コンサルタントまたは移行スペシャリストを使用して、ファイアウォールリクエストの一括承認フォームを送信します。

アーキテクチャ

次の表は、ファイアウォールリクエスト承認プロセスの手順を示しています。

ツール

「Palo Alto Networks」または「SolarWinds」などのスキャナーツールを使用して、ファイアウォールと IP アドレスを分析し、検証できます。

エピック

ファイアウォールリクエストを分析

タスク	説明	必要なスキル
ポートと IP アドレスを分析します。	ファイアウォールリクエストの送信者は、必要なファイアウォールポートと IP アドレスを把握するための初期分析を行います。これが完了すると、InfoSec チームが必要なポートを開き、IP アドレスをマッピングするようリクエストします。	AWS クラウドエンジニア、移行スペシャリスト

ファイアウォールリクエストを検証します。

タスク	説明	必要なスキル
ファイアウォール情報を検証します。	AWS クラウド エンジニアは、InfoSec チームとの会議をスケジュールします。この会議中、エンジニアはファイアウォールリクエスト情報を調べて検証します。	AWS クラウドエンジニア、移行スペシャリスト

タスク	説明	必要なスキル
	通常、ファイアウォールリクエスト送信者は、ファイアウォールリクエスターと同じものです。この検証フェーズは、何か確認され、推奨された場合、承認者からのフィードバックに基づいて反復的に行うことができます。	
ファイアウォールリクエストのドキュメントを更新します。	InfoSec チームがフィードバックを共有すると、ファイアウォールリクエストのドキュメントが編集、保存、再アップロードされます。この文書は繰り返しのたびに更新されます。 このドキュメントはバージョン管理下の保存フォルダに保存することをお勧めします。つまり、すべての変更が追跡され、正しく適用されます。	AWS クラウドエンジニア、移行スペシャリスト

ファイアウォールリクエストを送信

タスク	説明	必要なスキル
ファイアウォールリクエストを送信します。	ファイアウォールリクエスト承認者がファイアウォールの一括承認リクエストを承認すると、 AWS クラウド エンジニアはファイアウォールリクエストを送信します。リクエストは、開いている必要があるポートと、 のマッピングと	AWS クラウドエンジニア、移行スペシャリスト

タスク	説明	必要なスキル
	更新に必要な IP アドレスを指定します AWS アカウント。 ファイアウォールリクエストが提出されたら、提案やフィードバックを行うことができます。このフィードバックプロセスを自動化し、定義済みのワークフローメカニズムを通じて編集内容を提出することをお勧めします。	

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

EC2 Windows インスタンスを AWS Managed Services アカウントに取り込み、移行

作成者：Anil Kunapareddy (AWS) と Venkatramana Chintha (AWS)

概要

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) Windows インスタンスを Amazon Web Services (AWS) Managed Services (AMS) アカウントに移行して取り込むプロセスを段階的に説明しています。AMS は、インスタンスをより効率的かつ安全に管理するのに役立ちます。AMS は運用上の柔軟性を実現し、セキュリティとコンプライアンスを強化し、キャパシティの最適化とコストの削減を支援します。

このパターンでは、AMS アカウントのステージングサブネットに移行した EC2 Windows インスタンスから始まります。このタスクを実行するために、AWS アプリケーション移行サービスなど、さまざまな移行サービスとツールを利用できます。

AMS が管理する環境を変更するには、特定の操作またはアクションの変更要求 (RFC) を作成して送信します。AMS ワークロードインジェスト (WIGS) RFC を使用して、インスタンスを AMS アカウントに取り込み、カスタム Amazon マシンイメージ (AMI) を作成します。次に、別の RFC を送信して EC2 スタックを作成し、AMS が管理する EC2 インスタンスを作成します。詳細については、Kubernetes ドキュメントのAMS ワークロード取り込みを参照してください。

前提条件と制限

前提条件

- AMS が管理するアクティブな AWS アカウント
- 既存のランディングゾーン
- AMS が管理する VPC に変更を加える権限
- AMS アカウントのステージングサブネットの Amazon EC2 Windows インスタンス
- AMS WIGS を使用してワークロードを移行するための一般的な前提条件の完了
- AMS WIGS を使用してワークロードを移行するためのWindows 前提条件の完了

制約事項

- このパターンでは Windows サーバーを稼働させる EC2 インスタンス用です。このパターンでは、 Linux などの他のオペレーティングシステムを実行しているインスタンスに適用されません。

アーキテクチャ

ソーステクノロジースタック

AMS アカウントのステージングサブネットにある Amazon EC2 Windows インスタンス

ターゲットテクノロジースタック

AWS Managed Services (AMS) によって管理されるAmazon EC2 Windows インスタンス

ターゲット アーキテクチャ

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動できます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Managed Services (AMS) は、モニタリング、インシデント管理、セキュリティガイダンス、パッチサポート、AWS ワークロードのバックアップなど、AWS インフラストラクチャの継続的な管理を提供することで、より効率的で安全な運用を支援します。

その他のサービス

- PowerShell は Windows、Linux、および macOS で動作するMicrosoft の自動化および構成管理プログラムです。

エピック

インスタンスの設定を行います。

タスク	説明	必要なスキル
DNS クライアント設定を変更します。	1. ソースEC2インスタンスで、管理者としてコ	移行エンジニア

タスク	説明	必要なスキル
	マンドプロンプトを開き、gpedit.msc と入力し、Enter キーを押します。 2. ローカルグループポリシーエディターで、コンピューターの構成、管理用テンプレート、ネットワーク、DNS クライアントに移動します。 3. プライマリ DNS サフィックスの場合、未設定を選択します。 4. プライマリ DNS サフィックスの権限委譲の場合、未設定を選択します。	
Windows 更新の設定を変更します。	1. gpedit.msc で、コンピュータの構成 > 管理用テンプレート > Windows コンポーネント > Windows Updateの順に移動します。 2. イントラネットの Microsoft 更新サービスの場所を指定で、未設定を選択します。 3. 自動更新の設定では、未設定を選択します。 4. 自動更新の検出頻度には、未設定を選択します。 5. ローカルグループポリシーエディタを閉じます。	移行エンジニア

タスク	説明	必要なスキル
ファイアウォールの有効化	1. ソースEC2インスタンスで、管理者としてコマンドプロンプトを開き、`services.msc` と入力し、Enter キーを押します。 2. Windows サービスで、ファイアウォールを有効にします。 3. サービスウィンドウを閉じます。	移行エンジニア

AMS WIGS のインスタンスを準備します。

タスク	説明	必要なスキル
インスタンスをクリーンアップして準備します。	1. Bastion ホストとローカル認証情報を使用して、ステージングサブネット内の EC2 インスタンスへのリモートデスクトッププロトコル (RDP) 接続を作成します。 2. AMS に必要のないレガシーソフトウェア、ウイルス対策ソフトウェア、バックアップソリューションをすべて削除します。	移行エンジニア
sppnp.dll ファイルを修復します。	1. `C:\Windows\System32\sppnp.dll` に移動します。	移行エンジニア

タスク	説明	必要なスキル

	2. sppnp.dll を sppnp_old.dll に名前変更します。
	3. PowerShell と管理者の認証情報を使用して、次のコマンドを入力します。
	`dism /online /cleanup-image /restorehealth`
	`sfc /scannnow`
	4. EC2 Windows インスタンスを再起動します。	

WIG 以前の検証スクリプトを実行します。	1. AMS ドキュメントのワークロードの移行：Windows の取り込み前検証 から Windows WIGS の取り込み前検証 zip ファイル (windows-prewings-valication.zip) をダウンロードします。 2. Windows の WIG 以前の検証スクリプトを実行して、結果を確認します。 3. 検証に失敗した場合、問題を修正し、検証が成功するまで検証スクリプトを再実行します。	移行エンジニア

タスク	説明	必要なスキル
フェイルセーフ AMI を作成します。	WIG 前の検証に成功したら、次のように取り込み前 AMI を作成します。 1. デプロイ、高度なスタックコンポーネント、AMI、作成を選択します。 2. 作成中に `Key=Name, Value=APPLICATION-ID_IngestReady` タグを追加します。 3. 次に進む前に、AMI が作成されるまで待っています。 詳細については、AMS ドキュメントの AMI | Createを参照してください。	移行エンジニア

インスタンスの取り込みと検証

タスク	説明	必要なスキル
RFC を提出して、ワークロードの取り込みスタックを作成します。	AMS WIGS を起動するための変更要求 (RFC) を提出します。手順については、AMS ドキュメントのワークロードインジェストスタック：作成 を参照してください。これによりワークロードの取り込みが開始され、バックアップツール、Amazon EC2 管理ソフトウェア、ウイルス対策ソフトウェアなど、AMS に必要なす	移行エンジニア

タスク	説明	必要なスキル
	べてのソフトウェアがインストールされます。	
移行が成功に完了したことを検証します。	ワークロードの取り込みが完了すると、AMS が管理するインスタンスと AMS に取り込まれた AMI が表示されます。 1. AMS が管理するインスタンスにドメイン認証情報を使用してログインします。 2. ドメイン参加を以下のように検証します。 a. ファイルエクスプローラーで、この PC を右クリックし、プロパティを選択します。 b. デバイス仕様セクションで、ドメインがフルデバイス名に表示されていることを確認します。 3. ソースとターゲットのディスクドライブを検証します。	移行エンジニア

ターゲットの AMS アカウントでインスタンスを起動します。

タスク	説明	必要なスキル
RFC を送信して EC2 スタックを作成します。	1. Windows インスタンスの AMS で取り込まれた AMI を使用して、AMS ドキュメントのEC2 スタックインスタンスの作成の手順に従っ	移行エンジニア

タスク	説明	必要なスキル
	て EC2 スタックの RFC を準備します。EC2 スタック RFC で、サーバー名、タグ、ターゲット VPC、ターゲットサブネット、インスタンスタイプ、ターゲットセキュリティグループ、取り込み AMI、ロールを含むすべてのパラメータを指定します。 2. EC2 スタックの RFC を送信し、インスタンスが正常に作成されるまで待ちます。	

関連リソース

AWS 規範ガイダンス

- Windows 上の AWS Managed Services 事前ワークロード取り込みアクティビティを自動化
- Python を使用して AMS で RFC を自動的に作成する

AMS ドキュメント

- AMS ワークロードインジェスト
- 移行でリソースを変更する方法
- ワークロードの移行：標準プロセス

マーケティングリソース

- AWS マネージドサービス
- AWS Managed Services よくある質問
- AWS Managed Services リソース

- AWS Managed Services 機能

Couchbase Server データベースを Amazon EC2 に移行する

作成者: Subhani Shaik (AWS)

概要

このパターンでは、Couchbase Server をオンプレミス環境から Amazon Elastic Compute Cloud (Amazon EC2) に移行する方法について説明します AWS。

Couchbase Server は、リレーショナルデータベース機能を提供する分散 NoSQL (JSON ドキュメント) データベースです。Couchbase Server データベースを に移行すると AWS 、スケーラビリティの向上、パフォーマンスの向上、コスト効率の向上、セキュリティの強化、管理の簡素化、グローバルリーチを実現でき、高可用性と低レイテンシーのデータアクセスを必要とするアプリケーションにメリットをもたらします。また、 AWS マネージドサービスを通じて高度な機能にアクセスすることもできます。

の Couchbase Server AWS には、次の主要な機能があります。

- メモリファーストアーキテクチャ
- 高可用性、ディザスタリカバリ、ロードバランシング
- 最適なパフォーマンスを実現するマルチマスター、マルチリージョンのデプロイ

主な利点の詳細については、「追加情報」セクションと「Couchbase ウェブサイト」を参照してください。

前提条件と制限

前提条件

- Virtual Private Cloud (VPC)、2 つのアベイラビリティーゾーン、プライベートサブネット、セキュリティグループ AWS アカウント を持つアクティブな 。手順については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「VPC の作成」を参照してください。
- ソース環境とターゲット環境間の接続が有効になります。Couchbase Server で使用される TCX ポートの詳細については、Couchbase ドキュメントを参照してください。

アーキテクチャ

次の図は、Couchbase Server を に移行するための大まかなアーキテクチャを示しています AWS。

オンプレミスの Couchbase クラスターから、データは を使用してカスタマーゲートウェイを移動しますAWS Direct Connect。データはルーターと AWS Direct Connect ルートを通過し、 AWS Virtual Private Network （ AWS VPN ） ゲートウェイを介して VPC に到達します。VPC には、Couchbase Server を実行している EC2 インスタンスが含まれています。この AWS インフラストラクチャには、アクセスコントロール用の AWS Identity and Access Management (IAM)、データ暗号化用の AWS Key Management Service （ AWS KMS ）、ブロックストレージ用の Amazon Elastic Block Store (Amazon EBS)、データストレージ用の Amazon Simple Storage Service (Amazon S3) も含まれています。

ツール

AWS のサービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Direct Connect は、標準のイーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、ネットワークパスでインターネットサービスプロバイダーをバイパス AWS のサービス しながら、仮想インターフェイスをパブリックに直接作成できます。

ベストプラクティス

- さまざまな運用プラットフォームでの Couchbase のインストールと設定
- Couchbase Server を にデプロイするためのベストプラクティス AWS
- Couchbase クラスターの作成
- Couchbase アプリケーションのパフォーマンスのベストプラクティス
- Couchbase Server のセキュリティのベストプラクティス
- Couchbase Server データベースのストレージのベストプラクティス

エピック

Couchbase Server 用の Amazon EC2 インスタンスをデプロイする

タスク	説明	必要なスキル
Amazon EC2 コンソールを開きます。	AWS Management Console にサインインし Amazon EC2 コンソールを開きます。	DevOps エンジニア、Couchbase 管理者
Amazon EC2 インスタンスをデプロイします。	オンプレミスの Couchbase Server 設定に一致する EC2 インスタンスを起動します。EC2 インスタンスをデプロイする方法の詳細については、Amazon EC2 ドキュメントの「Amazon EC2 インスタンスを起動する」を参照してください。 Amazon EC2	DevOps エンジニア、Couchbase 管理者

Amazon EC2 での Couchbase Server のインストールと設定

タスク	説明	必要なスキル
Couchbase クラスターをインストールします。	Amazon EC2 に Couchbase Server をインストールする前に、Couchbase Server のデプロイガイドラインを確認してください。 Couchbase Server をインストールするには、Couchbase Server のドキュメントを参照してください。	Couchbase 管理者
クラスターを設定します。	クラスターを設定するには、Couchbase ドキュメント	Couchbase 管理者

タスク	説明	必要なスキル
	の「クラスター設定オプション」を参照してください。	

新しいノードを追加し、Couchbase クラスターを再調整する

タスク	説明	必要なスキル
EC2 インスタンスのノードを追加します。	Couchbase がインストールされている新しくデプロイされた EC2 インスタンスを既存のオンプレミスクラスターに追加します。手順については、Couchbase Server ドキュメントの「ノードの追加と再調整」を参照してください。	Couchbase 管理者
クラスターを再調整します。	再調整プロセスにより、EC2 インスタンスで新しく追加されたノードが Couchbase クラスターのアクティブなメンバーになります。手順については、Couchbase Server ドキュメントの「ノードの追加と再調整」を参照してください。	Couchbase 管理者

接続の再設定

タスク	説明	必要なスキル
オンプレミスノードを削除して再調整します。	クラスターからオンプレミスノードを削除できるようになりました。ノードを削除したら、再調整プロセスに従っ	Couchbase 管理者

タスク	説明	必要なスキル
	て、クラスター内の使用可能なノード間でデータ、インデックス、イベント処理、クエリ処理を再分散します。手順については、Couchbase Server ドキュメントの「ノードの削除と再調整」を参照してください。	
接続パラメータを更新します。	アプリケーションが新しいノードに接続できるように、アプリケーションの接続パラメータを更新して新しい Amazon EC2 IP アドレスを使用します。	Couchbase アプリケーション開発者

関連リソース

- Couchbase サーバーサービス
- を使用して Couchbase サーバーをデプロイする AWS Marketplace
- Couchbase サーバーに接続する
- バケットの管理
- クロスデータセンターレプリケーション (XDCR)
- Couchbase Inc. ライセンス契約

追加情報

主な利点

Couchbase データベースを に移行する AWS と、次の利点があります。

スケーラビリティ。Couchbase クラスターは、物理ハードウェアを管理することなく、需要に応じてスケールアップまたはスケールダウンできるため、変動するデータボリュームやアプリケーションの使用状況に簡単に対応できます。 AWS が提供するもの：

- 垂直スケーリングオプションと水平スケーリングオプション
- グローバルデプロイ機能
- 間の負荷分散 AWS リージョン
- データベーススケーリングソリューション
- コンテンツ配信の最適化

パフォーマンス最適化. は、Couchbase データベースの高速なデータアクセスと低レイテンシーを確保するために、高性能なネットワークインフラストラクチャと最適化されたインスタンスタイプ AWS を提供します。

- ハイパフォーマンスコンピューティング (HPC) オプション
- Amazon CloudFront によるグローバルコンテンツ配信
- 複数のストレージオプション
- Amazon Relational Database Service (Amazon RDS) や Amazon DynamoDB などの高度なデータベースサービス
- を使用した低レイテンシーの接続 AWS Direct Connect

コスト最適化。適切なインスタンスタイプと設定を選択して、ワークロードに基づいてパフォーマンスとコストのバランスを取ります。使用したリソースに対してのみ支払います。これにより、オンプレミスハードウェアを管理し、規模の AWS クラウド 経済を活用する必要がなくなるため、運用コストを削減できる可能性があります。

- リザーブドインスタンスは、Couchbase を で使用するときに、事前に計画し、コストを大幅に削減するのに役立ちます AWS。
- 自動スケーリングはオーバープロビジョニングを防ぎ、使用率とコスト効率を最適化するのに役立ちます。

セキュリティを強化しました。データ暗号化 AWS、アクセスコントロール、セキュリティグループなど、 の堅牢なセキュリティ機能を活用して、Couchbase に保存する機密データを保護します。その他の利点：

- 責任AWS 共有モデルは、クラウドのセキュリティ (AWS 責任) とクラウド内のセキュリティ (お客様の責任) を明確に区別します。
- AWS コンプライアンスは、主要なセキュリティ標準をサポートしています。

- AWS には、高度な暗号化オプションが用意されています。
- AWS Identity and Access Management (IAM) は、 リソースへの安全なアクセスを管理するのに役立ちます。

簡素化された management. は Couchbase のマネージドサービス AWS を提供するため、基盤となるインフラストラクチャを管理するのではなく、アプリケーション開発に集中できます。

グローバルリーチ。Couchbase クラスターを複数の にデプロイ AWS リージョン して、世界中のユーザーに低レイテンシーを実現できます。データベースは、クラウドまたはハイブリッド環境に完全にデプロイできます。組み込みのエンタープライズグレードのセキュリティと、エッジからクラウドへの高速で効率的なデータの双方向同期により、データを保護します。同時に、ウェブおよびモバイルアプリを構築するための一貫したプログラミングモデルを使用して開発を簡素化できます。

ビジネス継続性：

- データのバックアップと復旧。問題が発生した場合は、 AWS Backupを使用してデータの耐障害性と簡単な復旧を確保できます。ディザスタリカバリオプションについては、 AWS Well-Architected Framework ドキュメントを参照してください。
- Couchbase マルチリージョンデプロイ: Couchbase データベースをマルチリージョン AWS 環境にデプロイするには、 で Couchbase Server にサブスクライブしAWS Marketplace、 AWS CloudFormationテンプレートを使用して各リージョンに個別の Couchbase クラスターを作成し、リージョン間でデータを同期するようにクロスリージョンレプリケーションを設定できます。この設定により、複数のリージョンにわたって高可用性と地理的冗長性が確保されます。詳細については、Couchbase ドキュメントの「 を使用して Couchbase サーバーをデプロイ AWS Marketplaceする」を参照してください。

インフラストラクチャの俊敏性：

- 迅速なリソースのプロビジョニングとプロビジョニング解除
- グローバルインフラストラクチャのリーチ
- 需要に基づく自動スケーリング
- 一貫性のあるデプロイのためのコードとしてのインフラストラクチャ (IaC)
- 異なるワークロードに最適化された複数のインスタンスタイプ

イノベーションの有効化：

- AI/ML、IoT、分析などの最新のテクノロジーへのアクセス
- 運用オーバーヘッドを削減するマネージドサービス
- 最新のアプリケーション開発プラクティス
- サーバーレスコンピューティングオプション

運用上の優秀性：

- 一元的なモニタリングとログ記録
- 自動リソース管理
- 予知保全機能
- リソース使用状況の可視性を強化
- 効率的なデプロイプロセス

モダナイゼーションの機会：

- マイクロサービスアーキテクチャ
- DevOps プラクティスの実装
- クラウドネイティブアプリケーション開発
- レガシーアプリケーションのモダナイゼーション

競争上の利点：

- 市場投入までの時間を短縮
- カスタマーエクスペリエンスの向上
- データ駆動型意思決定
- 強化されたビジネスインテリジェンス

ログ配信を使用して Db2 for LUW を Amazon EC2 に移行することで、システム停止時間を短縮する

作成者: Feng Cai (AWS)、Ambarish Satarkar (AWS)、Saurabh Sharma (AWS)

概要

お客様が IBM Db2 for LUW (Linux、UNIX、Windows) ワークロードを Amazon Web Services (AWS) に移行する場合、Amazon Elastic Compute Cloud (Amazon EC2) を Bring Your Own License (BYOL) モデルとともに使用するのが最も速い方法です。ただし、オンプレミス Db2 から AWS に大量のデータを移行することは、特に停止期間が短い場合、課題となる可能性があります。多くのお客様は、停止時間を 30 分未満に設定しようとしています。これにより、データベース自体に費やす時間が少なくなります。

このパターンでは、トランザクションログ配信を使用して、停止時間を短くして Db2 の移行を実現する方法を扱っています。このアプローチは、リトルエンディアン Linux プラットフォームで実行される Db2 にも運用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスファイルシステムのレイアウトと一致する EC2 インスタンス上で実行される Db2 インスタンス
- EC2 インスタンスにアクセス可能な Amazon Simple Storage Service (Amazon S3) バケット
- Amazon S3 をプログラムで呼び出すための AWS Identity and Access Management (IAM) ポリシーとロール Amazon S3
- Amazon EC2 とオンプレミスサーバーのタイムゾーンとシステムクロックの同期
- AWS Site-to-Site VPN または AWS Direct Connect を介して AWS に接続されたオンプレミスのネットワーク

機能制限

- Db2 オンプレミスインスタンスと Amazon EC2 は同じプラットフォームファミリーに配置する必要があります。
- Db2 のオンプレミスワークロードをログ記録する必要があります。ログに記録されていないトランザクションをブロックするには、データベース設定で `blocknonlogged=yes` を設定します。

製品バージョン

- Db2 for LUW バージョン 11.5.9 以降

アーキテクチャ

ソーステクノロジースタック

- Db2 on Linux x86_64

ターゲットテクノロジースタック

- Amazon EBS
- Amazon EC2
- AWS Identity and Access Management (IAM)
- Amazon S3
- AWS Site-to-Site VPN または Direct Connect

ターゲット アーキテクチャ

次の図は、Amazon EC2 上の Db2 への仮想プライベートネットワーク (VPN) 接続を使用してオンプレミスで実行されている 1 つの Db2 インスタンスを示しています。点線は、データセンターと AWS クラウド間の VPN トンネルを表しています。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- AWS Direct Connect は、標準のイーサネット光ファイバーケーブルを介して内部ネットワークを Direct Connect の場所にリンクします。この接続を使用すると、Amazon S3 などの のパブリッククサービス、または Amazon VPC に対する仮想インターフェイスを直接作成できるため、ネットワークパスのインターネットサービスプロバイダーを回避できます。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Site-to-Site VPN は、AWS で起動するインスタンスと独自のリモートネットワーク間でトラフィックを渡すのに役立ちます。

その他のツール

- db2cli は、 Db2 のインタラクティブ CLI コマンドです。

ベストプラクティス

- ターゲットデータベースで、Amazon S3 のゲートウェイエンドポイントを使用して、Amazon S3 のデータベースバックアップイメージとログファイルにアクセスします。
- ソースデータベースで、AWS PrivateLink for Amazon S3 を使用して、データベースのバックアップイメージとログファイルを Amazon S3 に送信します。

エピック

環境変数を設定する

タスク	説明	必要なスキル
環境変数を設定する。	このパターンは、次の名前を使用しています。 • インスタンス名: `db2inst1` • データベース名: `SAMPLE` 環境に合わせて変更することができます。	DBA

オンプレミスの Db2 サーバーを設定する

タスク	説明	必要なスキル

AWS CLI をセットアップします。	最新バージョンの AWS CLI をダウンロードしてインストールするには、次のコマンドを実行します。
	
	`$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`sudo ./aws/install`
		Linux 管理者

Db2 アーカイブログのローカル送信先を設定します。	Amazon EC2 上のターゲットデータベースをオンプレミスのソースデータベースと同期させるには、ソースデータベースから最新のトランザクションログを取得する必要があります。 このセットアップでは、`/db2logs` はソース上の `LOGARCHMETH2` によってステージング領域として設定されます。このディレクトリにアーカイブされたログは Amazon S3 に同期され、Db2 on Amazon EC2 によってアクセスされます。AWS CLI コマンドではアクセスできないサードパーティベンダーツールを使用するように `LOGARCHMETH2` が設定さ	DBA

タスク	説明	必要なスキル

	れている可能性があるため、このモードでは LOGARCHMETH1 が使用されます。ログを取得するには、次のコマンドを実行します。
	`db2 connect to sample`
	`db2 update db cfg for SAMPLE using LOGARCHMETH2 disk:/db2logs`	

オンラインデータベースのバックアップを実行します。	オンラインデータベースバックアップを実行し、ローカルバックアップファイルシステムに保存します。 `db2 backup db sample online to /backup`	DBA

S3 バケットおよび IAM ポリシーをセットアップする

タスク	説明	必要なスキル
S3 バケットを作成する。	オンプレミスサーバー用の S3 バケットを作成して、バックアップ Db2 イメージとログファイルを AWS に送信します。 バケットは Amazon EC2 からもアクセスされます。 `aws s3api create-bucket --bucket logshipmig-db2 --region us-east-1`	AWS システム管理者

タスク	説明	必要なスキル
IAM ポリシーを作成します。	db2bucket.json ファイルには、Amazon S3 バケットにアクセスするための IAM ポリシーが含まれています。 { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["kms:GenerateDataKey", "kms:Decrypt", "s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload", "s3:ListBucket", "s3:DeleteObject", "s3:GetObjectVersion", "s3:ListMultipartUploadParts"], "Resource": [	AWS 管理者、AWS システム管理者

タスク	説明	必要なスキル

	<pre>"arn:aws:s3:::logshipmig-db2/*",
	
	"arn:aws:s3:::logshipmig-db2"
	]
	}
	]
	}</pre>
	ポリシーを作成するには、次の AWS CLI コマンドを使用します。
	
	<pre>aws iam create-policy \
	--policy-name db2s3policy \
	--policy-document file://db2bucket.json</pre>
	JSON 出力には、ポリシーの Amazon リソースネーム (ARN) が表示されます。 は アカウント ID aws_account_id を表します。
	
	<pre>"Arn": "arn:aws:iam::aws_account_id:policy/db2s3policy"</pre>	

タスク	説明	必要なスキル
EC2 インスタンスで使用される IAM ロールに IAM ポリシーをアタッチします。	ほとんどの AWS 環境では、実行中の EC2 インスタンスには、システム管理者によって設定された IAM ロールがあります。IAM ロールが設定されていない場合は、ロールを作成し、EC2 コンソールで IAM ロールの変更 を選択して、Db2 データベースをホストする EC2 インスタンスにロールを関連付けます。ポリシー ARN を使用して IAM ポリシーを IAM ロールにアタッチします。 `aws iam attach-role-policy \ --policy-arn "arn:aws:iam::aws_account_id:policy/db2s3policy" \ --role-name db2s3role` ポリシーがアタッチされると、IAM ロールに関連付けられた EC2 インスタンスは S3 バケットにアクセスできます。	AWS 管理者、AWS システム管理者

ソースデータベースのバックアップイメージとログファイルを Amazon S3 に送信します。

タスク	説明	必要なスキル

オンプレミスの Db2 サーバーで AWS CLI を設定します。	AWS CLI を で設定Access Key IDし、前のステップで Secret Access Key生成します。
	`$ aws configure`
	`AWS Access Key ID [None]: **********`
	`AWS Secret Access Key [None]: ********** **********`
	`Default region name [None]: us-east-1`
	`Default output format [None]: json`
		AWS 管理者、AWS システム管理者

Amazon S3 にバックアップイメージを送信します。	以前は、オンラインデータベースのバックアップが /backup ローカルディレクトリに保存されていました。バックアップイメージを S3 バケットに送信するには、次のコマンドを実行します。 `aws s3 sync /backup s3://logshipmig-db2/SAMPLE_backup`	AWS 管理者、移行エンジニア
Amazon S3 に Db2 アーカイブログを送信します。	オンプレミスの Db2 アーカイブログを、Amazon EC2 のターゲット Db2 インスタンスがアクセスできる S3 バケットと同期します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	`aws s3 sync /db2logs s3://logshipmig-db2/SAMPLE_LOG` cron または他のスケジューリングツールを使用して、このコマンドを定期的に実行します。この頻度は、ソースデータベースがトランザクションログファイルをアーカイブする頻度に応じて異なります。	

Db2 on Amazon EC2 を Amazon S3 に接続し、データベース同期を開始します。

タスク	説明	必要なスキル

PKCS12 キーストアを作成します。	Db2 は、公開鍵暗号規格 (PKCS) 暗号化キーストアを使用して AWS アクセスキーをセキュアに保護します。キーストアを作成し、それを使用するようにソース Db2 インスタンスを設定します。
	
	`gsk8capicmd_64 -keydb -create -db "/home/db2inst1/.keystore/db2s3.p12" -pw "<password>" -type pkcs12 -stash`
	
	`db2 "update dbm cfg using keystore_location /home/db2inst1/.keystore/db`
		DBA

タスク	説明	必要なスキル
	`2s3.p12 keystore_type pkcs12"`	

Db2 ストレージアクセスエイリアスを作成します。	ストレージアクセスエイリアスを作成するには、次のスクリプト構文を使用します。
	
	`db2 "catalog storage access alias <alias_name> vendor S3 server <S3 endpoint> container '<bucket_name>'"`
	
	たとえば、スクリプトは次のようになります。
	
	`db2 "catalog storage access alias DB2AWSS3 vendor S3 server s3.us-east-1.amazonaws.com container 'logshipmig-db2'"`
		DBA

タスク	説明	必要なスキル
ステージング領域を設定します。	デフォルトでは、Db2 は Amazon S3 でファイルをアップロードおよびダウンロードするためのステージング領域として DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH を使用します。デフォルトのパスは、インスタンスのホームディレクトリの下にある sqllib/tmp/RemoteStorage.xxxx で、xxxx は Db2 パーティション番号を参照します。ステージング領域には、バックアップイメージとログファイルを保存するために十分な容量が必要であることに注意してください。レジストリを使用して、ステージング領域を別のディレクトリにポイントできます。 またDB2_ENABLE_COS_SDK=ON 、、DB2_OBJECT_STORAGE_SETTINGS=EnableStreamingRestore 、およびawssdkライブラリへのリンクを使用して、データベースのバックアップと復元のために Amazon S3 ステージングエリアをバイパスすることをお勧めします。 `#By root:`	DBA

タスク	説明	必要なスキル

	`cp -rp /home/db2inst1/sqllib/lib64/awssdk/RHEL/7.6/* /home/db2inst1/sqllib/lib64/`
	
	`#By db2 instance owner:`
	`db2set DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH=/db2stage`
	`db2set DB2_ENABLE_COS_SDK=ON`
	`Db2set DB2_OBJECT_STORAGE_SETTINGS=EnableStreamingRestore`
	`db2stop`
	`db2start`	

バックアップイメージからデータベースを復元します。	S3 バケットのバックアップイメージから Amazon EC2 のターゲットデータベースを復元します。 `db2 restore db sample from DB2REMOTE://DB2AWSS3/logshipmig-db2/SAMPLE_backup replace existing`	DBA

タスク	説明	必要なスキル

データベースをロールフォワードします。	復元が完了すると、ターゲットデータベースはロールフォワード保留状態になります。トランザクションログファイルを取得する場所を Db2 が認識LOGARCHMETH2 するように LOGARCHMETH1 と を設定します。
	
	`db2 update db cfg for SAMPLE using LOGARCHMETH1 'DB2REMOTE://DB2AWSS3//SAMPLE_LOGS/'`
	`db2 update db cfg for SAMPLE using LOGARCHMETH2 OFF`
	
	データベースのロールフォワードを開始します。
	
	`db2 ROLLFORWARD DATABASE sample to END OF LOGS`
	
	このコマンドは S3 バケットに転送されたすべてのログファイルを処理します。s3 sync コマンドは、オンプレミス Db2 サーバー上で定期的に実行されます。たとえば、s3 sync を 1 時間ごとに実行し、すべてのログファイルを同期するのに 10分 かかる場合は、コマンドを 1 時間お
		DBA

タスク	説明	必要なスキル
	きに 10 分間実行するように設定します。	

カットオーバー期間中に Db2 on Amazon EC2 をオンラインにします

タスク	説明	必要なスキル

ターゲットデータベースをオンラインにします。	カットオーバー期間中、以下のいずれかの操作を行います。
	• オンプレミスデータベースを ADMIN MODE に配置し、s3 sync コマンドを実行して最後のトランザクションログを強制的にアーカイブします。
	• データベースをシャットダウンします。
	
	最後のトランザクションログが Amazon S3 に同期されたら、最後に ROLLFORWARD コマンドを実行します。
	
	`db2 rollforward DB sample to END OF LOGS`
	`db2 rollforward DB sample complete`
	
	`Rollforward Status`
	`....`
		DBA

タスク	説明	必要なスキル

	`Rollforward status = not pending`
	`....`
	`DB20000I The ROLLFORWARD command completed successfully.`
	
	`db2 activate db sample`
	`DB20000I The ACTIVATE DATABASE command completed successfully.`
	
	ターゲットデータベースをオンラインにし、アプリケーション接続を Db2 on Amazon EC2 にポイントします。	

トラブルシューティング

問題	ソリューション
複数のデータベースが異なるホスト (DEV、QA、PROD) で同じインスタンス名とデータベース名を持つ場合、バックアップとログは同じサブディレクトリに移動することがあります。	DEV、QA、PROD に異なる S3 バケットを使用し、混乱を避けるためにホスト名をサブディレクトリプレフィックスとして追加します。
同じ場所に複数のバックアップイメージがある場合、復元時に次のエラーが発生します。 `SQL2522N More than one backup file matches the time stamp value provided for the backed up database image.`	`restore` コマンドで、バックアップのタイムスタンプを追加します。 `db2 restore db sample from DB2REMOTE://DB2AWSS3/logshipmig-db2/SAMPLE_backup taken at 20230628164042 replace existing`

関連リソース

- 「異なるオペレーティングシステムとハードウェアプラットフォーム間の Db2 バックアップおよびリストア操作」
- 「Db2 ストレージアクセスエイリアスと DB2-REMOTE の設定」
- 「Db2 ロールフォワードコマンド」
- 「　Db2 セカンダリログアーカイブ方式」

高可用性ディザスタリカバリ機能を備えた Db2 for LUW を Amazon EC2 に移行する

作成者: Feng Cai (AWS)、Aruna Gangireddy (AWS)、および Venkatesan Govindan (AWS)

概要

お客様が IBM Db2 LUW (Linux、UNIX、Windows) ワークロードを Amazon Web Services (AWS) に移行する場合、Amazon Elastic Compute Cloud (Amazon EC2) を Bring Your Own License (BYOL) モデルとともに使用するのが最も速い方法です。ただし、オンプレミス Db2 から AWS に大量のデータを移行することは、特に停止期間が短い場合、課題となる可能性があります。多くのお客様は、停止時間を 30 分未満に設定しようとしています。これにより、データベース自体に費やす時間が少なくなります。

このパターンでは、Db2 の高可用性ディザスタリカバリ (HADR) を使用して、短期間の停止で Db2 を移行する方法を扱っています。　このアプローチは、リトルエンディアン Linux プラットフォーム上にあり、データパーティショニング機能 (DPF) を使用していない Db2 データベースに適用されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスファイルシステムのレイアウトと一致する Amazon EC2 インスタンス上で実行される Db2 インスタンス
- EC2 インスタンスにアクセス可能な Amazon Simple Storage Service (Amazon S3) バケット
- Amazon S3 をプログラムで呼び出すための AWS Identity and Access Management (IAM) ポリシーとロール Amazon S3
- Amazon EC2 とオンプレミスサーバーのタイムゾーンとシステムクロックの同期
- AWS Site-to-Site VPN または AWS Direct Connect を介して AWS に接続されたオンプレミスのネットワーク
- HADR ポートでのオンプレミスサーバーと Amazon EC2 間の通信

機能制限

- Db2 オンプレミスインスタンスと Amazon EC2 は同じプラットフォームファミリーに配置する必要があります。
- HADR はパーティション化されたデータベース環境ではサポートされません。

- HADR は、データベースログファイルの RAW I/O (ディスクへの直接アクセス) の使用をサポートしていません。
- HADR は、無限ログ記録をサポートしていません。
- LOGINDEXBUILD は YES に設定する必要があります。これにより、インデックスを再構築するためのログの使用量が増加します。
- Db2 のオンプレミスワークロードをログ記録する必要があります。ログに記録されていないトランザクションをブロックするには、データベース設定で blocknonlogged=yes を設定します。

製品バージョン

- Db2 for LUW バージョン 11.5.9 以降

アーキテクチャ

ソーステクノロジースタック

- Db2 on Linux x86_64

ターゲットテクノロジースタック

- Amazon EC2
- AWS Identity and Access Management (IAM)
- Amazon S3
- AWS Site-to-Site VPN

ターゲット アーキテクチャ

以下の図では、オンプレミスの Db2 が db2-server1 上でプライマリとして稼働しています 2 つの HADR スタンバイターゲットがあります。1 つのスタンバイターゲットはオプションとしてオンプレミスに配置されています。もう 1 つのスタンバイターゲット db2-ec2 は Amazon EC2 に配置されています。データベースが AWS にカットオーバーされると、 がプライマリdb2-ec2になります。

1. ログはプライマリオンプレミスデータベースからスタンバイオンプレミスデータベースにストリーミングされます。

2. Db2 HADR を使用すると、ログはオンプレミスのプライマリデータベースから Site-to-Site VPN 経由で Db2 on Amazon EC2 にストリーミングされます。
3. Db2 のバックアップとアーカイブのログは、オンプレミスのプライマリデータベースから AWS の S3 バケットに送信されます。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- AWS Direct Connect は、標準のイーサネット光ファイバーケーブルを介して内部ネットワークを Direct Connect の場所にリンクします。この接続を使用すると、Amazon S3 などの のパブリックサービス、または Amazon VPC に対する仮想インターフェイスを直接作成できるため、ネットワークパスのインターネットサービスプロバイダーを回避できます。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Site-to-Site VPN は、AWS で起動するインスタンスと独自のリモートネットワーク間でトラフィックを渡すのに役立ちます。

その他のツール

- db2cli は、 Db2 のインタラクティブ CLI コマンドです。

ベストプラクティス

- ターゲットデータベースで、Amazon S3 のゲートウェイエンドポイントを使用して、Amazon S3 のデータベースバックアップイメージとログファイルにアクセスします。
- ソースデータベースで、AWS PrivateLink for Amazon S3 を使用して、データベースのバックアップイメージとログファイルを Amazon S3 に送信します。

エピック

環境変数を設定する

タスク	説明	必要なスキル
環境変数を設定する。	このパターンでは、以下の名前とポートを使用します。 1. Db2 オンプレミスホスト名: `db2-server1` 2. HADR スタンバイホスト名: `db2-server2` (HADR がオンプレミスで実行されている場合) 3. Amazon EC2 ホスト名: `db2-ec2` 4. インスタンス名: `db2inst1` 5. データベース名: `SAMPLE` 6. ハードポート: • `db2-server1: 50010` • `db2-server2: 50011` • `db2-ec2: 50012` 環境に合わせて変更することができます。	DBA

オンプレミスの Db2 サーバーを設定する

タスク	説明	必要なスキル
AMS CLI をセットアップします。	最新バージョンの AWS CLI をダウンロードしてインストールするには、次のコマンドを実行します。	Linux 管理者

タスク	説明	必要なスキル

	`$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`sudo ./aws/install`	

タスク	説明	必要なスキル
Db2 アーカイブログのローカル送信先を設定します。	大量の更新バッチジョブやネットワークの速度低下などの状況により、HADR スタンバイサーバーに遅延が生じる可能性があります。スタンバイサーバーは、進行状況に間に合うようにプライマリサーバーのトランザクションログを必要とします。次の順序でログをリクエストします。 • プライマリサーバーのアクティブログディレクトリ。 • スタンバイサーバー上の LOGARCHMETH1 または LOGARCHMETH2 の場所。 • プライマリサーバー上の LOGARCHMETH1 または LOGARCHMETH2 の場所。 このセットアップでは、/db2logs はソース上の LOGARCHMETH2 によってステージング領域として設定されます。このディレクトリにアーカイブされたログは Amazon S3 に同期され、Db2 on Amazon EC2 によってアクセスされます。このパターンでは、AWS CLI コマンドがアクセスできないサードパーティーベンダーツールを使用するように が設定されている場合LOGARCHME	DBA

タスク	説明	必要なスキル

	TH2 LOGARCHMETH1 があるため、 が を使用します。
	
	`db2 connect to sample`
	`db2 update db cfg for`
	`SAMPLE using LOGARCHME`
	`TH2 disk:/db2logs`	

オンラインデータベースのバックアップを実行します。	オンラインデータベースバックアップを実行し、ローカルバックアップファイルシステムに保存します。
	
	`db2 backup db sample`
	`online to /backup`
		DBA

S3 バケットおよび IAM ポリシーをセットアップする

タスク	説明	必要なスキル

S3 バケットを作成する。	オンプレミスサーバー用の S3 バケットを作成して、バックアップ Db2 イメージとログファイルを AWS に送信します。 バケットは Amazon EC2 によってアクセスされます。
	
	`aws s3api create-bucket`
	`--bucket hadrmig-db2`
	`--region us-east-1`
		AWS 管理者

IAM ポリシーを作成します。	`db2bucket.json` ファイルには、S3 バケットにアクセス	AWS 管理者、AWS システム管理者

タスク	説明	必要なスキル
	するための IAM ポリシーが含まれています。 `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["kms:GenerateDataKey", "kms:Decrypt", "s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload", "s3:ListBucket", "s3:DeleteObject", "s3:GetObjectVersion", "s3:ListMultipartUploadParts"], "Resource": ["arn:aws:s3:::hadrmig-db2/*",`	

タスク	説明	必要なスキル

	<pre>"arn:aws:s3:::hadrmig-db2"
	            ]
	        }
	    ]
	}</pre>
	ポリシーを作成するには、次の AWS CLI コマンドを使用します。
	<pre>aws iam create-policy \
	      --policy-name db2s3hapolicy \
	      --policy-document file://db2bucket.json</pre>
	JSON 出力には、ポリシーの Amazon リソースネーム (ARN) が表示されます。 は アカウント ID aws_account_id を表します。
	<pre>"Arn": "arn:aws:iam::aws_account_id:policy/db2s3hapolicy"</pre>	

タスク	説明	必要なスキル
IAM ロールに IAM ポリシーをアタッチします。	通常、Db2 が実行されている EC2 インスタンスには、システム管理者によって IAM ロールが割り当てられます。 Db2 IAM ロールが割り当てられていない場合は、Amazon EC2 コンソールで IAM ロールの変更 を選択できます。 EC2 インスタンスに関連付けられた IAM ロールに IAM ポリシーをアタッチします。ポリシーがアタッチされると、EC2 インスタンスは S3 バケットにアクセスできます。 `aws iam attach-role-policy --policy-arn "arn:aws:iam::aws_account_id:policy/db2s3hapolicy" --role-name db2s3harole`	

ソースデータベースのバックアップイメージとログファイルを Amazon S3 に送信します。

タスク	説明	必要なスキル

オンプレミスの Db2 サーバーで AWS CLI を設定します。	前に生成`Secret Access Key`した `Access Key ID`とを使用して AWS CLI を設定します。
	
	`$ aws configure`
	`AWS Access Key ID [None]: **********`
		AWS 管理者、AWS システム管理者

タスク	説明	必要なスキル
	`AWS Secret Access Key [None]: ********** ********** Default region name [None]: us-east-1 Default output format [None]: json`	
Amazon S3 にバックアップイメージを送信します。	以前は、オンラインデータベースのバックアップが /backup ローカルディレクトリに保存されていました。バックアップイメージを S3 バケットに送信するには、次のコマンドを実行します。 `aws s3 sync /backup s3://hadrmig-db2/SAMPLE_backup`	AWS 管理者、AWS システム管理者

タスク	説明	必要なスキル
Amazon S3 に Db2 アーカイブログを送信します。	オンプレミスの Db2 アーカイブログを、Amazon EC2 のターゲット Db2 インスタンスからアクセスできる Amazon S3 バケットと同期します。Db2 Amazon EC2 `aws s3 sync /db2logs s3://hadrmig-db2/SAMPLE_LOGS` cron または他のスケジューリングツールを使用して、このコマンドを定期的に実行します。この頻度は、ソースデータベースがトランザクションログファイルをアーカイブする頻度に応じて異なります。	

Db2 on Amazon EC2 を Amazon S3 に接続し、データベースの同期を開始します。

タスク	説明	必要なスキル
PKCS12 キーストアを作成します。	Db2 は、公開鍵暗号規格 (PKCS) 暗号化キーストアを使用して AWS アクセスキーをセキュアに保護します。キーストアを作成し、それを使用するようにソース Db2 を設定します。 `gsk8capicmd_64 -keydb -create -db "/home/db2inst1/.keystore/db2s3.p12" -pw "<passwor`	DBA

タスク	説明	必要なスキル
	d>" -type pkcs12 - stash db2 "update dbm cfg using keystore_ location /home/db2 inst1/.keystore/db 2s3.p12 keystore_type pkcs12"	

タスク	説明	必要なスキル

Db2 ストレージアクセスエイリアスを作成します。	Db2 はストレージアクセスエイリアスを使用して INGEST、LOAD、BACKUP DATABASE、または RESTORE DATABASE コマンドで Amazon S3 に直接アクセスします。
	
	EC2 インスタンスに IAM ロールを割り当てたため、 USERおよび PASSWORD は必須ではありません。
	
	`db2 "catalog storage access alias <alias_name> vendor S3 server <S3 endpoint> container '<bucket_name>'"`
	
	たとえば、スクリプトは次のようになります。
	
	`db2 "catalog storage access alias DB2AWSS3 vendor S3 server s3.us-east-1.amazonaws.com container 'hadrmig-db2'"`
		DBA

タスク	説明	必要なスキル

ステージング領域を設定します。	DB2_ENABLE_COS_SDK =ON 、DB2_OBJEC T_STORAGE_SETTINGS =EnableStreamingRe store 、および awssdkライブラリへのリンクを使用して、データベースのバックアップと復元のために Amazon S3 ステージングエリアをバイパスすることをお勧めします。
	
	`#By root:`
	`cp -rp /home/db2inst1/ sqllib/lib64/awssdk/ RHEL/7.6/* /home/db2 inst1/sqllib/lib64/`
	
	`#By db2 instance owner:`
	`db2set DB2_OBJEC T_STORAGE_LOCAL_ST AGING_PATH=/db2stage`
	`db2set DB2_ENABL E_COS_SDK=ON`
	`db2set DB2_OBJEC T_STORAGE_LOCAL_ST AGING_PATH=/db2stage`
	`db2stop`
	`db2start`
		DBA

タスク	説明	必要なスキル

バックアップイメージからデータベースを復元します。	S3 バケットのバックアップイメージから Amazon EC2 のターゲットデータベースを復元します。
	
	`db2 create db sample on /data1`
	`db2 restore db sample from DB2REMOTE://DB2AWSS3/hadrmig-db2/SAMPLE_backup replace existing`
		DBA

オンプレミスで HADR を使用せずに HADR をセットアップする

タスク	説明	必要なスキル
オンプレミスの Db2 サーバーをプライマリとして設定します。	db2-server1 (オンプレミス ソース)上にある HADR のデータベース構成設定をプライマリとして更新します。トランザクション応答時間が最短の SUPERASYNC モードHADR_SYNCMODE に設定します。 `db2 update db cfg for sample using HADR_LOCAL_HOST db2-server1 HADR_LOCAL_SVC 50010 HADR_REMOTE_HOST db2-ec2 HADR_REMOTE_SVC 50012 HADR_REMOTE_INST db2inst1 HADR_SYNCMODE`	DBA

タスク	説明	必要なスキル
	SUPERASYNC DB20000 I The UPDATE DATABASE CONFIGURATION command completed successfully オンプレミスデータセンターと AWS 間で、ネットワークに多少の遅延が発生することが予想されます。(ネットワークの信頼性に基づいて異なる HADR_SYNCMODE 値を設定できます。詳細については、「関連リソース」セクションを参照してください。	
ターゲットデータベースログのアーカイブ先を変更します。	Amazon EC2 環境と一致するようにターゲットデータベースログのアーカイブ先を変更します。 `db2 update db cfg for SAMPLE using LOGARCHMETH1 'DB2REMOTE://DB2AWSS3//SAMPLE_LOGS/' LOGARCHMETH2 OFF DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully`	DBA

タスク	説明	必要なスキル
Amazon EC2 サーバーで Db2 用の HADR を設定します。	スタンバイdb2-ec2として の HADR のデータベース設定を更新します。 `db2 update db cfg for sample using HADR_LOCAL_HOST db2-ec2 HADR_LOCAL_SVC 50012 HADR_REMOTE_HOST db2-server1 HADR_REMOTE_SVC 50010 HADR_REMOTE_INST db2inst1 HADR_SYNCMODE SUPERASYNC DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully`	DBA

タスク	説明	必要なスキル

HADR 設定を検証します。	ソースとターゲットの Db2 サーバーで、HADR パラメーターを検証します。
	
	のセットアップを確認するにはdb2-server1 、次のコマンドを実行します。
	
	`db2 get db cfg for sample|grep HADR`
	`HADR database role = PRIMARY`
	`HADR local host name (HADR_LOCAL_HOST) = db2-server1`
	`HADR local service name (HADR_LOCAL_SVC) = 50010`
	`HADR remote host name (HADR_REMOTE_HOST) = db2-ec2`
	`HADR remote service name (HADR_REMOTE_SVC) = 50012`
	`HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1`
	`HADR timeout value (HADR_TIMEOUT) = 120`
	`HADR target list (HADR_TARGET_LIST) =`
	`HADR log write synchronization mode`
		DBA

タスク	説明	必要なスキル
	(HADR_SYNCMODE) = NEARSYNC HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = AUTOMATIC(52000) HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0 HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0 HADR SSL certificate label (HADR_SSL_LABEL) = HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF のセットアップを確認するにはdb2-ec2、次のコマンドを実行します。 db2 get db cfg for sample|grep HADR HADR database role = STANDBY HADR local host name (HADR_LOCAL_HOST) = db2-ec2 HADR local service name (HADR_LOCAL_SVC) = 50012 HADR remote host name (HADR_REMOTE_HOST) = db2-server1	

タスク	説明	必要なスキル
	HADR remote service name (HADR_REMOTE_SVC) = 50010 HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1 HADR timeout value (HADR_TIMEOUT) = 120 HADR target list (HADR_TARGET_LIST) = HADR log write synchronization mode (HADR_SYNCMODE) = SUPERASYNC HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = AUTOMATIC(52000) HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0 HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0 HADR SSL certificate label (HADR_SSL_LABEL) = HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF HADR_LOCAL_HOST 、HADR_LOCAL_SVC 、HADR_REMOTE_HOST 、および	

タスク	説明	必要なスキル
	HADR_REMOTE_SVC のパラメータは、1 つのプライマリと 1 つのスタンバイの HADR 設定を示します。	

Db2 HADR インスタンスを起動します。	db2-ec2 まずスタンバイサーバーで Db2 HADR インスタンスを起動します。
	`db2 start hadr on db sample as standby`
	`DB20000I The START HADR ON DATABASE command completed successfully.`
	プライマリ (ソース) サーバーで Db2 HADR を起動しますdb2-server1 。
	`db2 start hadr on db sample as primary`
	`DB20000I The START HADR ON DATABASE command completed successfully.`
	これで、オンプレミスの Db2 と Amazon EC2 間の HADR 接続が正常に確立されました。 Db2 プライマリサーバー db2-server1 は、トランザクションログの db2-ec2 へのリアルタイムストリーミングを開始します。
		DBA

HADR がオンプレミスで存在する場合に HADR を設定する

タスク	説明	必要なスキル
Db2 on Amazon EC2 を補助スタンバイとして追加します。	HADR がオンプレミス Db2 インスタンスで実行されている場合、 で次のコマンドHADR_TARGET_LIST を実行して、 を使用して Amazon EC2 の Db2 を補助スタンバイとして追加できますdb2-ec2。 Amazon EC2 db2 update db cfg for sample using HADR_LOCAL_HOST db2-ec2 HADR_LOCAL_SVC 50012 HADR_REMOTE_HOST db2-server1 HADR_REMOTE_SVC 50010 HADR_REMOTE_INST db2inst1 HADR_SYNCMODE SUPERASYNC DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully. db2 update db cfg for sample using HADR_TARGET_LIST "db2-server1:50010|db2-server2:50011" DB20000I The UPDATE DATABASE CONFIGURATION command	DBA

タスク	説明	必要なスキル
	`completed successfully.`	

タスク	説明	必要なスキル
オンプレミスサーバーに補助スタンバイ情報を追加します。	2 台のオンプレミスサーバ (プライマリサーバとスタンバイサーバ)の HADR_TARGET_LIST を更新します。 でdb2-server1 、次のコードを実行します。 `db2 update db cfg for sample using HADR_TARGET_LIST "db2-server2:50011|db2-ec2:50012" DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully. SQL1363W One or more of the parameters submitted for immediate modification were not changed dynamically. For these configuration parameters, the database must be shutdown and reactivated before the configuration parameter changes become effective.`  でdb2-server2 、次のコードを実行します。  `db2 update db cfg for sample using HADR_TARG`	DBA

タスク	説明	必要なスキル
	ET_LIST "db2-serv er1:50010|db2-ec2: 50012" DB20000I The UPDATE DATABASE CONFIGURATION command completed successfu lly. SQL1363W One or more of the parameter s submitted for immediate modificat ion were not changed dynamically. For these configura tion parameters, the database must be shutdown and reactivated before the configuration parameter changes become effective.	

タスク	説明	必要なスキル

HADR 設定を検証します。	ソースとターゲットの Db2 サーバーで、HADR パラメーターを検証します。
	
	でdb2-server1 、次のコードを実行します。
	
	`db2 get db cfg for sample|grep HADR`
	`HADR database role = PRIMARY`
	`HADR local host name (HADR_LOCAL_HOST) = db2-server1`
	`HADR local service name (HADR_LOCAL_SVC) = 50010`
	`HADR remote host name (HADR_REMOTE_HOST) = db2-server2`
	`HADR remote service name (HADR_REMOTE_SVC) = 50011`
	`HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1`
	`HADR timeout value (HADR_TIMEOUT) = 120`
	`HADR target list (HADR_TARGET_LIST) = db2-server2:50011|db2-ec2:50012`	

タスク	説明	必要なスキル
	HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = AUTOMATIC(52000) HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0 HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0 HADR SSL certificate label (HADR_SSL_LABEL) = HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF でdb2-server2 、次のコードを実行します。 db2 get db cfg for sample|grep HADR HADR database role = STANDBY HADR local host name (HADR_LOCAL_HOST) = db2-server2 HADR local service name (HADR_LOCAL_SVC) = 50011 HADR remote host name (HADR_REMOTE_HOST) = db2-server1	

タスク	説明	必要なスキル
	HADR remote service name (HADR_REMOTE_SVC) = 50010 HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1 HADR timeout value (HADR_TIMEOUT) = 120 HADR target list (HADR_TARGET_LIST) = db2-server1:50010|db2-ec2:50012 HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = AUTOMATIC(52000) HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0 HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0 HADR SSL certificate label (HADR_SSL_LABEL) = HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF でdb2-ec2、次のコードを実行します。	

タスク	説明	必要なスキル

	`db2 get db cfg for sample|grep HADR`
	`HADR database role = STANDBY`
	`HADR local host name (HADR_LOCAL_HOST) = db2-ec2`
	`HADR local service name (HADR_LOCAL_SVC) = 50012`
	`HADR remote host name (HADR_REMOTE_HOST) = db2-server1`
	`HADR remote service name (HADR_REMOTE_SVC) = 50010`
	`HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1`
	`HADR timeout value (HADR_TIMEOUT) = 120`
	`HADR target list (HADR_TARGET_LIST) = db2-server1:50010|db2-server2:50011`
	`HADR log write synchronization mode (HADR_SYNCMODE) = SUPERASYNC`
	`HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = AUTOMATIC(52000)`	

タスク	説明	必要なスキル
	HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0 HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0 HADR SSL certificate label (HADR_SSL_LABEL) = HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF HADR_LOCAL_HOST 、HADR_LOCAL_SVC 、HADR_REMOTE_HOST 、HADR_REMOTE_SVC 、および HADR_TARGET_LIST のパラメータは、1 つのプライマリ HADR 設定と 2 つのスタンバイ HADR 設定を示します。	

タスク	説明	必要なスキル

Db2 HADR を停止して起動します。	これで、3 台のサーバーすべてに HADR_TARGET_LIST が設定されました。各 Db2 サーバーは他の 2 つのサーバーを認識しています。新しい設定を利用するには、HADR を停止して再起動します (短時間の停止)。
	
	でdb2-server1 、次のコマンドを実行します。
	
	`db2 stop hadr on db sample`
	`db2 deactivate db sample`
	`db2 activate db sample`
	
	でdb2-server2 、次のコマンドを実行します。
	
	`db2 deactivate db sample`
	`db2 start hadr on db sample as standby`
	`SQL1766W  The command completed successfully`
	
	でdb2-ec2、次のコマンドを実行します。
	
	`db2 start hadr on db sample as standby`
	`SQL1766W  The command completed successfully`
		DBA

タスク	説明	必要なスキル

	でdb2-server1 、次のコマンドを実行します。
	
	`db2 start hadr on db sample as primary`
	`SQL1766W The command completed successfully`
	
	これで、オンプレミスの Db2 と Amazon EC2 間の HADR 接続が正常に確立されました。 Db2 プライマリサーバー db2-server1 は、db2-server2 と db2-ec2 の両方へのトランザクションログレコードのリアルタイムストリーミングを開始します。	

カットオーバー期間中は Db2 on Amazon EC2 をプライマリとして設定します。

タスク	説明	必要なスキル
スタンバイサーバーに HADR 遅延がないことを確認してください。	プライマリサーバー db2-server1 から HADR ステータスを確認します。HADR_STATE が REMOTE_CATCHUP 状態になっても、心配する必要はありません。HADR_SYNCMODE が SUPERASYNC に設定されている場合は正常です。PRIMARY_LOG_TIME とは、同期しているSTANDBY_R	DBA

タスク	説明	必要なスキル

	EPLAY_LOG_TIME ことを示します。
	
	`db2pd -hadr -db sample`
	
	`HADR_ROLE = PRIMARY`
	
	`REPLAY_TYPE = PHYSICAL`
	
	`HADR_SYNCMODE = SUPERASYNC`
	
	`STANDBY_ID = 2`
	
	`LOG_STREAM_ID = 0`
	
	`HADR_STATE = REMOTE_CATCHUP`
	`.....`
	
	`PRIMARY_LOG_TIME = 10/26/2022 02:11:32.000000 (1666750292)`
	
	`STANDBY_LOG_TIME = 10/26/2022 02:11:32.000000 (1666750292)`
	`STANDBY_REPLAY_LOG_TIME = 10/26/2022 02:11:32.000000 (1666750292)`	

タスク	説明	必要なスキル

HADR テイクオーバーを実行します。	移行を完了するには、HADR テイクオーバーのコマンドを実行して db2-ec2 をプライマリデータベースとして使用します。コマンドを使用して HADR_ROLE 値db2pdを確認します。
	
	`db2 TAKEOVER HADR ON DATABASE sample`
	`DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.`
	
	`db2pd -hadr -db sample`
	`Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:03:25 -- Date 2022-10-26-02.46.45.048988`
	
	`HADR_ROLE = PRIMARY`
	`REPLAY_TYPE = PHYSICAL`
	
	AWS への移行を完了するには、Db2 on Amazon EC2 にアプリケーション接続を設定します。	

トラブルシューティング

問題	ソリューション
ファイアウォールとセキュリティ上の理由から NAT を使用する場合、ホストには 2 つの IP アドレス (1 つは内部、もう 1 つは外部) が割り当てられる可能性があるので、HADR IP アドレスのチェックが失敗する可能性があります。START HADR ON DATABASE コマンドは、次のメッセージを返します。 `HADR_LOCAL_HOST:HADR_LOCAL_SVC (-xx-xx-xx-xx.:50011 (xx.xx.xx.xx:50011)) on remote database is different from HADR_REMOTE_HOST:HADR_REMOTE_SVC (xx-xx-xx-xx.:50011 (x.x.x.x:50011)) on local database.`	NAT 環境で HADR をサポートするには、内部アドレスと外部アドレスの両方を使用して HADR_LOCAL_HOST を設定できます。例えば、Db2 サーバーの内部名が host1 で外部名が host1E の場合、HADR_LOCAL_HOST は HADR_LOCAL_HOST: "host1 | host1E" になります

関連リソース

- 「異なるオペレーティングシステムとハードウェアプラットフォーム間の Db2 バックアップおよびリストア操作」
- 「Db2 ストレージアクセスエイリアスと DB2-REMOTE の設定」
- 「Db2 高可用性ディザスタリカバリ」
- 「hadr_syncmode - ピアステート設定パラメータへのログ書き込み用の HADR 同期モード」

PowerCLI を使用して HCX オートメーションで VMware 仮想マシンを移行

作成者：Giri Nadiminty (AWS), Hassan Adekoya (AWS), と Naveen Deshwal

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

このパターンでは、VMware PowerCLI スクリプトによる VMware ハイブリッドクラウドエクステンション (HCX) オートメーションを使用して VMware オンプレミス仮想マシン (VM) を VMware Cloud on AWS に移行する方法について説明します。PowerCLI は Windows PowerShell に構築されたコマンドラインツールです。VMware ソフトウェアの管理に役立ち、インフラストラクチャと移行のタスクを自動化します。

このパターンでは、vCenter、ソフトウェアデファインドデータセンター (SDDC)、およびクラウド環境を自由に組み合わせて移行できます。このパターンに含まれている PowerCLI スクリプトは、すべての VM 構成タスクとスケジュールタスクをマウスクリックの代わりに自動化するため、移行作業にかかる時間を節約でき、人為的ミスのリスクを軽減できます。

前提条件と制限

前提条件

- SDDC 付けの アカウントのVMware Cloud on AWS
- 既存のオンプレミスまたはクラウドベースの vCenter または SDDC
- ソースとデスティネーションの vCenter または SDDC に必要な権限を持つユーザーアカウント
- ソースと目的の vCenter または SDDC 間で設定された HCX ネットワーク拡張 (HCX-NE) による HCX サイトペアリング
- 選択したサーバに VMware PowerCLI がインストールされています

制約事項

- ソースのvCenterがクロスvCenter NSXを使用している場合、PowerCLIモジュールが機能しません。PowerCLI の代わりに HCX API でスクリプトメソッド (Python など) を使用します。

- 移行した VM に新しい名前または IP アドレスが必要な場合、HCX API のスクリプトメソッド (Python など) を使用します。
- このパターンでは、必須.csv ファイルに入力されません。このファイルには、VMware vRealize ネットワークインサイト (vRNI) またはその他の方法を使用してデータを入力できます。

製品バージョン

- VMware vSphere バージョン 5 以降
- VMware HCX バージョン 4.4 以降
- VMware PowerCLI バージョン 12.7 以降

アーキテクチャ

ソーステクノロジースタック

- オンプレミスまたはクラウドベースの VMware

ターゲットテクノロジースタック

- VMware Cloud on AWS

ターゲット アーキテクチャ

ツール

AWS サービス

- VMware Cloud on AWS は、オンプレミスの VMware vSphere ベースの環境をAWS Cloudに移行して拡張することができる、と VMware が共同で設計したサービスです。

その他のツール

- VMware ハイブリッドクラウドエクステンション (HCX) は、基盤となるプラットフォームを変更せずに、オンプレミスの VMware 環境から VMware Cloud on AWS にワークロードを移行するためのユーティリティです。注：この製品は、以前はハイブリッドクラウドエクステンションおよび

NSX ハイブリッド接続と呼ばれていました。このパターンでは、仮想マシンの移行に HCX を使用します。

- VMware PowerCLI は、VMware vSphere と vCloud の管理を自動化するためのコマンドラインツールです。PowerCLI コマンドは、PowerShell コマンドレットを使用して Windows PowerShell で実行します。このパターンでは、PowerCLI を使用して移行コマンドを実行します。

コード

シンプルで自己完結型のスクリプト

この単一マシンのスクリプトを初期テストに使用して、設定オプションが受け入れられ、期待どおりに動作することを確認することを推薦します。手順については、エピックセクションを参照してください。

```
<# Manual Variables #>
$HcxServer = "[enterValue]"
$SrcNetworkName = "[enterValue]"
$DstNetworkName = "[enterValue]"
$DstComputeName = "[enterValue]"
$DstDSName = "[enterValue]"
$DstFolderName = "[enterValue]"
$vmName = "[enterValue]"

<# Environment Setup #>
Connect-HCXServer -Server $HcxServer
$HcxDstSite = Get-HCXSite -Destination
$HcxSrcSite = Get-HCXSite -Source
$SrcNetwork = Get-HCXNetwork -Name $SrcNetworkName -Type VirtualWire -Site $HcxSrcSite
$DstNetwork = Get-HCXNetwork -Name $DstNetworkName -Type NsxtSegment -Site $HcxDstSite
$DstCompute = Get-HCXContainer -Name $DstComputeName -Site $HcxDstSite
$DstDS = Get-HCXDatastore -Name $DstDSName -Site $HcxDstSite
$DstFolder = Get-HCXContainer -name $DstFolderName -Site $HcxDstSite
$vm = Get-HCXVM -Name $vmName

<# Migration #>
$NetworkMapping = New-HCXNetworkMapping -SourceNetwork $SrcNetwork -DestinationNetwork
 $DstNetwork
$NewMigration = New-HCXMigration -VM $vm -MigrationType vMotion -SourceSite $HcxSrcSite
 -DestinationSite $HcxDstSite -Folder $DstFolder -TargetComputeContainer $DstCompute
 -TargetDatastore $DstDS -NetworkMapping $NetworkMapping -DiskProvisionType Thin
```

```
 -UpgradeVMTools $True -RemoveISOs $True -ForcePowerOffVm $True -RetainMac $True -
UpgradeHardware $True -RemoveSnapshots $True
```

フル機能の.csv ベースのスクリプト

テストが完了した後、以下のスクリプトを実稼働環境で使用できます。手順については、エピックセクションを参照してください。

```
<# Schedule #>
write-host("Getting Time for Scheduling")
$startTime = [DateTime]::Now.AddDays(12)
$endTime = [DateTime]::Now.AddDays(15)

<# Migration #>
Connect-HCXServer -Server [enterValue]
write-host("Getting Source Site")
$HcxSrcSite = Get-HCXSite
write-host("Getting Target Site")
$HcxDstSite = Get-HCXSite -Destination
$HCXVMS = Import-CSV .\Import_VM_list.csv
ForEach ($HCXVM in $HCXVMS) {
    $DstFolder = Get-HCXContainer $HCXVM.DESTINATION_VM_FOLDER -Site $HcxDstSite
    $DstCompute = Get-HCXContainer $HCXVM.DESTINATION_COMPUTE  -Site $HcxDstSite
    $DstDatastore = Get-HCXDatastore $HCXVM.DESTINATION_DATASTORE -Site $HcxDstSite
    $SrcNetwork = Get-HCXNetwork $HCXVM.SOURCE_NETWORK -Type VirtualWire -Site
 $HcxSrcSite
    $DstNetwork = Get-HCXNetwork $HCXVM.DESTINATION_NETWORK -Type NsxtSegment -Site
 $HcxDstSite
    $NetworkMapping = New-HCXNetworkMapping -SourceNetwork $SrcNetwork -
DestinationNetwork $DstNetwork
    $NewMigration = New-HCXMigration -VM (Get-HCXVM $HCXVM.VM_NAME) -MigrationType
 Bulk -SourceSite $HcxSrcSite -DestinationSite $HcxDstSite -Folder $DstFolder -
TargetComputeContainer $DstCompute -TargetDatastore $DstDatastore -NetworkMapping
 $NetworkMapping -DiskProvisionType Thin -UpgradeVMTools $True -RemoveISOs $True -
ForcePowerOffVm $True -RetainMac $True -UpgradeHardware $True -RemoveSnapshots $True -
ScheduleStartTime $startTime -ScheduleEndTime $endTime
    Start-HCXMigration -Migration $NewMigration -Confirm:$false
}
```

エピック

手動変数の情報を収集

タスク	説明	必要なスキル
ソースとデスティネーションの vCenter と SDDC サーバの名前を検索します。	PowerCLI スクリプトには、このエピックで説明されている変数が必要です。この情報は、スクリプトを使いやすくするために事前に収集できます。 vSphere コンソールの HCX セクションで、インフラストラクチャ、サイトペアリングを選択します。表示されている移行元と移行先のサーバー名をメモします。	クラウドアーキテクト
移行元と移行先の HCX 名を見つけます。	vSphere コンソールの HCX セクションで、システム、管理を選択します。表示されている移行元と移行先の HCX 名をメモします。	クラウドアーキテクト
移行元と移行先のネットワーク名を見つけます。	vSphere コンソールの HCX セクションで、システム、ネットワーク拡張を選択します。移行元と移行先のネットワーク名をメモします。 **Note** 別の方法として、HCX サーバーに接続した後、PowerCLI Get-HCXNetwork コマンド	クラウドアーキテクト

タスク	説明	必要なスキル
	と Get-HCXNetwork-Destination コマンドを使用して、送信元と送信先のネットワーク名を取得することもできます。	
vSphere コンソールから追加情報を収集します。	vSphere コンソールで、次の情報を収集します。 • 移行する VM の名前 • 移行先コンピューティング環境 (クラスタ/ホスト) • 移行先データストア • デスティネーション VM フォルダ名	クラウドアーキテクト

移行判断を行う

タスク	説明	必要なスキル
移行オプションを決定します。	以下を決定します。 • `MigrationType` — HCX アシストの移行タイプには、vMotion、バルク、コールド、RAV があります。どちらを選択するかは、ダウンタイム要件、ネットワーク帯域幅、移行期間、およびワークロードの種類によって異なります。詳細については、AWS ブログ記事ハイブリッドクラウド	クラウドアーキテクト

タスク	説明	必要なスキル
	拡張 (HCX) による VMware Cloud on AWS へのワークロードの移行 を参照してください。 • `DiskProvisionType (Thin, Thick)` • `UpgradeVMTools ($True, $False)` • `RemoveISOs ($True, $False)` • `ForcePowerOffVm ($True, $False)` • `RetainMac ($True, $False)` • `UpgradeHardware ($True, $False)` • `RemoveSnapshots ($True, $False)` 以下のオプションについて詳しくは、VMware の開発者向けドキュメントを参照してください。	

初期テスト用の簡単なスクリプトを実行します。

タスク	説明	必要なスキル
スクリプトをコピーします。	簡易版のスクリプトは 1 つのファイルにまとめられています。これを使用して 1 台の	クラウドアーキテクト

タスク	説明	必要なスキル
	マシンの移行をテストできます。 このパターンのコードセクションから最初のスクリプトをコピーし、VMware PowerCLI モジュールがインストールされているコンピュータに保存します。MXNet をインストールするには、VMware のドキュメントの指示に従います。	
スクリプト変数を設定します。	`Manual Variables` スクリプトのセクションにすべての変数を設定します。	クラウドアーキテクト
移行変数を設定します。	スクリプトの `Migration` セクションですべての `New-HCXMigration` 設定を行います。	クラウドアーキテクト
サイトを指定します。	(オプション) 移行元、または移行先に複数のサイトがある場合、スクリプトの `Environment Setup` セクションでサイトを手動で指定します。 ソースとターゲットのサイトが 1 つの場合、スクリプトは自動的に情報を検索します。	クラウドアーキテクト

タスク	説明	必要なスキル
スクリプトを実行します。	PowerCLI がインストールされているサーバーで、管理者の PowerShell ウィンドウからスクリプトを実行し、プロンプトが表示されたら認証情報を入力します。	クラウドアーキテクト
スクリプトを検証します。	VM の移行が開始されたことを確認します。	クラウドアーキテクト

フル機能のスクリプトを実行して、複数の VM を移行

タスク	説明	必要なスキル

.csv ファイルを作成して入力します。	コンピューター上で `Import_VM_list.csv` という.csv ファイルを作成し、次のサンプル内容を入力します。
	
	`VM_NAME,DESTINATION_VM_FOLDER,DESTINATION_COMPUTE,DESTINATION_DATASTORE,SOURCE_NETWORK,DESTINATION_NETWORK`
	`[enterValue],[enterValue],[enterValue],[enterValue],[enterValue],[enterValue]`
	
	.csv ファイルの各 `[enterValue]` ファイルを、以前に収集した情報に置き換えます。
		クラウドアーキテクト

タスク	説明	必要なスキル
	Note .csv ファイルは、VMware vRealize Network Insight (vRNI) または他の方法を使用して入力できます。	
スクリプトをコピーします。	フル機能版のスクリプトでは、外部の.csv ファイルの情報を使用して複数の VM を自動的に移行します。 このパターンのコードセクションから 2 番目のスクリプトをコピーし、VMware PowerCLI モジュールがインストールされているコンピューターの.csv ファイルと同じフォルダに保存します。	クラウドアーキテクト

タスク	説明	必要なスキル
スクリプトを修正します。	スクリプトを編集し、以下の変更を行います。 • 7 行目：HCX サーバ変数 (`Connect-HCXServer `) を設定します。 • 12 行目：(オプション) .csv ファイル名を異なる方法で設定した場合、更新します。 • 3～4行目：(オプション) スケジュールを設定します。 • 20行目：(オプション) `Migration` セクションで `New-HCXMigration` 設定を指定します。 • 9 行目と 11 行目：(オプション) 移行元または移行先に複数のサイトが含まれている場合、目的のサイトを手動で指定します。	クラウドアーキテクト
スクリプトを実行します。	PowerCLI がインストールされているサーバーで、管理者の PowerShell ウィンドウからスクリプトを実行し、プロンプトが表示されたら認証情報を入力します。	クラウドアーキテクト
スクリプトを検証します。	VM の移行が開始されたことを確認します。	クラウドアーキテクト

トラブルシューティング

問題	ソリューション
スクリプトが失敗し、次のエラーメッセージが表示されます。 「すべてのソースネットワークがターゲットにマップされていません!」	ソースの vCenter がクロス vCenter NSXを使用する場合、PowerCLIモジュールが機能しません。PowerCLI の代わりに HCX API でスクリプトメソッド (Python など) を使用します。これは、PowerCLI スクリプトの既知の制限です。
スクリプトが失敗し、次のエラーメッセージが表示されます。 「接続-HCX サーバエラー：無許可」	入力された認証情報には必要な権限がありません。

関連リソース

- ハイブリッドクラウド拡張(HCX) 付けの VMware Cloud on AWS へのワークロードの移行 (AWS ブログ記事)
- VMware のアプリケーションとワークロードをAWS クラウドに再配置する移行アプローチの選択 (AWS 規範ガイダンス)
- HCXを使用するAWS上でVMware SDDCをVMware Cloud on AWSへ移行する (AWS 規範ガイダンス)
- HCX モジュールの使用を開始 (VMware ブログ記事)

F5 BIG-IPワークロードをAWS クラウド上のF5 BIG-IP VEに移行する

ウィル・バウアー (AWS) によって作成されました

概要

Organizations は、アジリティとレジリエンスを高めるために、Amazon Web Services (AWS) クラウドへの移行を検討しています。F5 BIG-IP のセキュリティおよびトラフィック管理ソリューションをAWS クラウドに移行した後は、エンタープライズアーキテクチャ全体にわたる俊敏性と価値の高い運用モデルの採用に集中できます。

このパターンでは、F5 BIG-IPワークロードをAWS クラウド上の F5 BIG-IP仮想エディション (VE) ワークロードに移行する方法を説明します。ワークロードは、既存の環境をリホストし、サービス検出や API 統合などのリプラットフォームの側面をデプロイすることによって移行されます。AWS CloudFormation テンプレート は、ワークロードの AWS クラウドへの移行を加速します。

このパターンは、F5のセキュリティおよびトラフィック管理ソリューションを移行しようとしている技術エンジニアリングチームと建築チームを対象としており、AWS AWS 規範ガイダンス ワェブサイトの AWSクラウド上の F5 BIG-IP から F5 BIG-IP VE への移行 ガイドに付属しています。

前提条件と制限

前提条件

- 既存のオンプレミスの F5 BIG-IP ワークロード。
- BIG-IP VE バージョン用の既存の F5 ライセンス。
- アクティブなAWS アカウント
- NATゲートウェイまたはElastic IPアドレスを介してイグジットが構成され、以下のエンドポイントにアクセスできるように構成された既存の仮想プライベートクラウド（VPC）：Amazon Simple Storage Service (Amazon S3)、Amazon Elastic Compute Cloud (Amazon EC2)、AWS Security Token Service (AWS STS)、および Amazon CloudWatch。また、モジュール式でスケーラブルな VPC アーキテクチャ のクイックスタートをデプロイの構成要素として変更することもできます。
- 要件に応じて、1 つまたは 2 つの既存のアベイラビリティーゾーン。
- 各アベイラビリティーゾーンに 3 つの既存のプライベートサブネット
- AWS CloudFormation テンプレート。F5 GitHub リポジトリで入手できます。

移行の際、要件に応じて、次のものを使用することもあります。

- F5 クラウドフェイルオーバーエクステンション は、Elastic IP アドレスマッピング、セカンダリ IP マッピング、ルートテーブルの変更を管理します。
- 複数のアベイラビリティーゾーンを使用する場合は、F5 クラウドフェイルオーバーエクステンションを使用して仮想サーバーへのElastic IPマッピングを処理する必要があります。
- 設定を管理するには、F5 アプリケーションサービス 3 (AS3)、F5 アプリケーションサービステンプレート (FAST)、またはその他のInfrastructure as Code (IaC) モデルを使用することを検討してください。IaC モデルで構成を準備し、コードリポジトリを使用すると、移行や継続的な管理作業に役立ちます。

専門知識

- このパターンでは、1 つ以上の VPC を既存のデータセンターに接続する方法に精通している必要があります。詳細については、Amazon VPC ドキュメントの ネットワークから Amazon VPC への接続オプション をご参照ください。
- トラフィック管理オペレーティングシステム(TMOS)、ローカルトラフィックマネージャ(LTM)、グローバルトラフィックマネージャ(GTM)、 アクセスポリシーマネージャ(APM)、アプリケーションセキュリティマネージャ(ASM)、アドバンストFirewall Manager(AFM)、BIG-IQ などのF5の製品とモジュールにも精通している必要があります。

製品バージョン

- このパターンはF5 BIG-IP バージョン12.1 以降をサポートしていますが、F5 BIG-IP バージョン13.1 以降を使用することを推奨します。

アーキテクチャ

ソーステクノロジースタック

- F5 BIG-IP ワークロード

ターゲットテクノロジースタック

- Amazon CloudFront
- Amazon CloudWatch
- Amazon EC2
- Amazon S3

- Amazon VPC
- AWS Global Accelerator
- AWS STS
- AWS Transit Gateway
- F5 BIG-IP VE

ターゲット アーキテクチャ

ツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- Amazon CloudFront は、世界中のデータセンターネットワークを通じて配信することで、ウェブコンテンツの配信を高速化します。これにより、レイテンシーが減少し、パフォーマンスが向上します。
- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Security Token Service (AWS STS) を使用すると、ユーザーに権限が制限された一時的な認証情報をリクエストできます。
- AWS Transit Gateway は、仮想プライベートクラウド (VPC) とオンプレミスネットワークを接続する中央ハブです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

発見と評価

タスク	説明	必要なスキル
F5 BIG-IPのパフォーマンスを評価してください。	仮想サーバー上のアプリケーションのパフォーマンスメトリックと、移行されるシステムのメトリックを収集して記録します。これにより、ターゲット AWS インフラストラクチャのサイズを正しく設定し、コストを最適化しやすくなります。	F5 アーキテクト、エンジニア、ネットワークアーキテクト、エンジニア
F5 BIG-IPのオペレーティングシステムと構成を評価してください。	どのオブジェクトを移行するか、またVLANなどのネットワーク構造を維持する必要があるかどうかを評価してください。	F5 アーキテクト、エンジニア
F5 ライセンスオプションを評価してください。	どのライセンスと消費モデルが必要かを評価してください。この評価は、F5 BIG-IP のオペレーティングシステムと構成に対するお客様の評価に基づいて行う必要があります。	F5 アーキテクト、エンジニア
公開アプリケーションを評価してください。	どのアプリケーションがパブリック IP アドレスを必要とするかを決定します。パフォーマンスとサービスレベルアグリーメント (SLA) の要件を満たすように、それらのアプリケーションを必要なインスタ	F5 アーキテクト、クラウドアーキテクト、ネットワークアーキテクト、エンジニア、アプリケーションチーム

タスク	説明	必要なスキル
	ンスとクラスターに合わせてください。	
内部アプリケーションを評価します。	内部ユーザーがどのアプリケーションを使用するかを評価してください。それらの内部ユーザーが組織のどこにいるのか、そしてそれらの環境がどのように AWS クラウドに接続しているのかを確認してください。また、それらのアプリケーションがデフォルトドメインの一部としてドメインネームシステム (DNS) を使用できることも確認する必要があります。	F5 アーキテクト、クラウドアーキテクト、ネットワークアーキテクト、エンジニア、アプリケーションチーム
AMI を確定します。	すべての F5 BIG-IP バージョンが Amazon マシンイメージ (AMI) として作成されるわけではありません。特定の必須クイックフィックスエンジニアリング(QFE)バージョンがある場合は、F5 BIG-IPイメージジェネレータツールを使用できます。このストーリーやその他のストーリーについて詳しくは、「関連リソース」セクションを参照してください。	F5 アーキテクト、クラウドアーキテクト、エンジニア
インスタンスタイプとアーキテクチャを確定します。	インスタンスタイプ、VPC アーキテクチャ、相互接続アーキテクチャを決定します。	F5 アーキテクト、クラウドアーキテクト、ネットワークアーキテクト、エンジニア、アプリケーションチーム

セキュリティとコンプライアンス関連のアクティビティを完了する

タスク	説明	必要なスキル
既存のF5セキュリティポリシーを文書化してください。	既存のF5セキュリティポリシーを文書化してください。必ず安全なコードリポジトリにコピーを作成してください。	F5 アーキテクト、エンジニア
AMI を暗号化します。	(オプション) 組織によっては、保管中のデータの暗号化が必要な場合があります。カスタム Bring-Your-Own-License (BYOL) イメージの作成の詳細については、「関連リソース」セクションを参照してください。	F5 アーキテクト、エンジニア、ネットワークアーキテクト、エンジニア
デバイスを強化します。	これにより、潜在的な脆弱性からの保護に役立ちます。	F5 アーキテクト、エンジニア

新しい AWS 環境を設定する

タスク	説明	必要なスキル
エッジアカウントとセキュリティアカウントを作成します。	AWS マネジメントコンソールにサインインし、エッジサービスとセキュリティサービスを提供し、運用する AWS アカウントを作成します。これらのアカウントは、共有サービスやアプリケーションの VPC を運用するアカウントとは異なる可能性があります。このステップはランディング	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	ゾーン一部として完了できます。	
エッジ VPC とセキュリティ VPC をデプロイします。	エッジサービスとセキュリティサービスの提供に必要な VPC をセットアップして設定します。	クラウドアーキテクト、エンジニア
ソースデータセンターに接続します。	F5 BIG-IPワークロードをホストするソースデータセンター Connect。	クラウドアーキテクト、ネットワークアーキテクト、エンジニア
VPC 接続をデプロイします。	エッジ VPC とセキュリティサービス VPC をアプリケーション VPC Connec \t。	ネットワークアーキテクト、エンジニア
インスタンスをデプロイします。	「関連リソース」セクションの AWS CloudFormation テンプレートを使用してインスタンスをデプロイします。	F5 アーキテクト、エンジニア
インスタンスフェイルオーバーのテストと設定。	AWS Advanced HA iApp テンプレートまたは F5 クラウドフェイルオーバーエクステンションが正しく設定され、動作していることを確認してください。	F5 アーキテクト、エンジニア

ネットワークを設定する

タスク	説明	必要なスキル
VPC トポロジを準備します。	Amazon VPC コンソールを開き、F5 BIG-IP VEのデプロイに必要なすべてのサブネット	ネットワークアーキテクト、F5 アーキテクト、クラウドアーキテクト、エンジニア

タスク	説明	必要なスキル
	と保護がVPC にあることを確認します。	
VPC エンドポイントを準備します。	F5 BIG-IPワークロードがTMMインターフェイス上のNATゲートウェイまたはElastic IPアドレスにアクセスできない場合は、Amazon EC2、Amazon S3、およびAWS STS用のVPC エンドポイントを準備します。	クラウドアーキテクト、エンジニア

データを移行する

タスク	説明	必要なスキル
設定を移行します。	F5 BIG-IP 設定を AWS クラウド上の F5 BIG-IP VE に移行します。	F5 アーキテクト、エンジニア
セカンダリ IP を関連付けます。	仮想サーバーの IP アドレスは、インスタンスに割り当てられたセカンダリ IP アドレスと関係があります。セカンダリ IP アドレスを割り当て、「再割り当て/再割り当てを許可」が選択されていることを確認します。	F5 アーキテクト、エンジニア

テスト設定

タスク	説明	必要なスキル
仮想サーバーの設定を検証します。	仮想サーバーをテストします。	F5 アーキテクト、アプリケーションチーム

運用の最終決定

タスク	説明	必要なスキル
バックアップ戦略を作成する。	フルスナップショットを作成するには、システムをシャットダウンする必要があります。詳細については、「関連リソース」セクションの「F5 BIG-IP 仮想マシンの更新」を参照してください。	F5 アーキテクト、クラウドアーキテクト、エンジニア
クラスタフェイルオーバーランブックを作成する。	フェイルオーバーランブックプロセスが完了していることを確認します。	F5 アーキテクト、エンジニア
ロギングの設定と検証。	必要な宛先にログを送信するように F5 テレメトリストリーミングを設定します。	F5 アーキテクト、エンジニア

カットオーバーの完了

タスク	説明	必要なスキル
新しいデプロイメントに切り替えましょう。		F5 アーキテクト、クラウドアーキテクト、ネットワークアーキテクト、エンジニア、アプリケーションチーム

関連リソース

移行ガイド

- AWS クラウドでのF5 BIG-IP から F5 BIG-IP VE への移行

F5 リソース

- AWS CloudFormation テンプレート。F5 GitHub リポジトリで入手できます。
- AWS Marketplace F5
- F5 ビッグIP VEの概要
- クイックスタートの例-WAF (LTM+ ASM) 搭載のBIG-IP バーチャルエディション
- AWS でのF5アプリケーションサービス：概要(ビデオ)
- F5 アプリケーションサービス 3 拡張ユーザーガイド
- F5 クラウドドキュメンテーション
- F5 iControl REST ウィキ
- F5 単一設定ファイルの概要 (11.x-15.x)
- F5 トポロジーラボ
- F5 ホワイトペーパー
- F5 BIG-IP イメージジェネレータツール
- F5 BIG-IP VE 仮想マシンのアップデート
- UCS アーカイブの「プラットフォーム移行」オプションの概要

バイナリメソッドを使用してオンプレミスの Go ウェブアプリケーションを AWS Elastic Beanstalk に移行します

Suhas Basavaraj (AWS) とShumaz Mukhtar Kazi (AWS) によって作成されました

概要

このパターンでは、オンプレミスの Go ウェブアプリケーションを AWS Elastic Beanstalk に移行する方法について説明します。アプリケーションの移行後、Elastic Beanstalk はソースバンドルのバイナリを構築し、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにデプロイします。

リホスト移行戦略としては、このパターンのアプローチは高速で、コードを変更する必要がないため、テストや移行にかかる時間を短縮できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- オンプレミスから Web アプリケーションへ
- ソースコードを含む または GitHub リポジトリ。GitHub を使用しない場合は、Elastic Beanstalk のアプリケーションソースバンドルを作成する他の方法があります。

製品バージョン

- Elastic Beanstalk でサポートされている最新の Go バージョンです。Elastic Beanstalk の詳細については、Elastic Beanstalk のドキュメントを参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスから Web アプリケーションへ

ターゲットテクノロジースタック

- AWS Elastic Beanstalk
- Amazon CloudWatch

ターゲット アーキテクチャ

ツール

- AWS Elastic Beanstalk を使用すると、アプリケーションを実行しているインフラストラクチャについて知識を得なくても、AWS クラウドでアプリケーションのデプロイと管理を簡単に行うことができます。Elastic Beanstalk は、選択肢を狭めたり制御を制限したりすることなく、管理の複雑さを軽減します。
- GitHub はオープンソースの分散型バージョン管理システムです。

エピック

Go Web アプリケーションソースバンドル (.zip ファイル) を作成します。

タスク	説明	必要なスキル
Go Web アプリケーションソースバンドル (.zip ファイル) を作成します。	Go アプリケーションのソースコードを含む GitHub リポジトリを開き、ソースバンドルを準備します。ソースバンドルには、Go `application.go` アプリケーションのメインパッケージをホストするルートディレクトリにソースファイルが含まれています。GitHub を使用しない場合は、アプリケーションソースバンドルを作成する他の方法について、このパターンの前半の前提条件 セクションを参照してください。	システム管理者、アプリケーション開発者
設定ファイルを作成します。	ソースバンドルに `.ebextensions` フォルダを作成し、そのフォルダ内に `options.config` ファイルを作成しま	システム管理者、アプリケーション開発者

タスク	説明	必要なスキル
	す。Elastic Beanstalk の詳細については、Elastic Beanstalk のドキュメントを参照してください。	
ソースバンドルの.zip ファイルを作成します。	以下のコマンドを実行してください。 `git archive -o ../godemoapp.zip HEAD` ソースバンドルの.zip ファイルを作成します。.zip ファイルをローカルファイルとしてダウンロードして保存します。 ⚠ Important .zip ファイルは 512 MB を超えることはできません。また、親フォルダまたは最上位ディレクトリを含めることはできません。	システム管理者、アプリケーション開発者

Elastic Beanstalk に Go ウェブアプリケーションを移行する

タスク	説明	必要なスキル
Elastic Beanstalk アプリケーションを選択します。	1. AWS マネジメントコンソールにサインインし、Elastic Beanstalk コンソールを開きます。	システム管理者、アプリケーション開発者

タスク	説明	必要なスキル
	2. AWS Region： リストからリージョン を選択します。 3. ナビゲーションペインで、アプリケーション を選択し、リストから既存のアプリケーション名を選択するか、またはアプリケーションを作成します。 Elastic Beanstalk アプリケーションを作成する方法については、Elastic Beanstalk のドキュメント を参照してください。	
Elastic Beanstalk ウェブサーバー環境を開始します。	1. アプリケーションの概要ページで、新しい環境の作成を選択し、Web サーバー環境を選択します。 2. 環境名とドメイン名フィールドに入力します。 3. プラットフォームバージョンを選択し、プラットフォームとして Go を選択します。	システム管理者、アプリケーション開発者

タスク	説明	必要なスキル
ソースバンドルの.zip ファイルを Elastic Beanstalk にアップロードします	1. アプリケーションコード では、コードをアップロードを選択して、ローカルファイル を選択します。 2. ソースバンドルを含む.zip ファイルを選択します。 3. バージョンラベルで、ファイルに一意の名前を付けて、環境を作成.を選択します。	システム管理者、アプリケーション開発者
デプロイした Go Web アプリケーションをテストします。	Elastic Beanstalk アプリケーションの概要ページにリダイレクトされます。概要の上部にある環境 ID の横で、`elasticbeanstalk.com` アプリケーションに移動するための末尾の URL を選択します。アプリケーションでは、設定ファイル内でこの名前を環境変数として使用し、Web ページに表示する必要があります。	システム管理者、アプリケーション開発者

トラブルシューティング

問題	ソリューション
Application Load Balancer を介してアプリケーションにアクセスできない	Elastic Beanstalk アプリケーションを含むターゲットグループを確認します。異常がある場合は、Elastic Beanstalk インスタンスにログインし、`nginx.conf` ファイル設定をチェックして、正しいヘルスステータス URL にルーティングされていることを確認します。ターゲット

問題	ソリューション
	グループのヘルスチェック URL を変更する必要がある場合があります。

関連リソース

- 古い Elastic Beanstalk プラットフォームバージョンは IMDSv1 をサポートしていました。
- Elastic Beanstalk での設定ファイルの使用
- Elastic Beanstalk でのサンプルアプリケーションの作成

AWS を使用してオンプレミス SFTP サーバーを に移行する AWS Transfer for SFTP

作成者：Akash Kumar (AWS)

概要

このパターンでは、 AWS クラウド AWS Transfer for SFTP サービスを使用して、Secure Shell (SSH) ファイル転送プロトコル (SFTP) を使用するオンプレミスのファイル転送ソリューションを に移行する方法について説明します。ユーザーは通常、ドメイン名または固定 IP を使用して SFTP サーバーに接続します。このパターンはどちらの場合にも当てはまります。

AWS Transfer for SFTP は のメンバーです AWS Transfer Family。これは、SFTP 経由で AWS ストレージサービスとの間でファイルを転送するために使用できる安全な転送サービスです。Amazon Simple Storage Service (Amazon S3) または Amazon Elastic File System (Amazon EFS) AWS Transfer for SFTP で を使用できます。このパターンは、Amazon S3 をストレージに使用します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 既存の SFTP ドメイン名または固定 SFTP IP アドレス。

制限

- 1 回のリクエストで転送できる最大オブジェクトは、現在 5 GiB です。100 MiB を超えるファイルの場合は、Amazon S3 マルチパートアップロードの使用を検討してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスのフラットファイルまたはデータベースダンプファイル。

ターゲットテクノロジースタック

- AWS Transfer for SFTP
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)

- AWS Identity and Access Management (IAM) ロールとポリシー
- Elastic IP アドレス
- セキュリティグループ
- Amazon CloudWatch Logs (オプション)

ターゲット アーキテクチャ

自動化とスケール

このパターンのターゲットアーキテクチャを自動化するには、アタッチされた AWS CloudFormation テンプレートを使用します。

- `amazon-vpc-subnets.yml` は、2 つのパブリックサブネットと 2 つのプライベートサブネットを使用して仮想プライベートクラウド (VPC) をプロビジョニングします。
- `amazon-sftp-server.yml` は、SFTP サーバーをプロビジョニングします。
- `amazon-sftp-customer.yml` は、ユーザーを追加します。

ツール

AWS のサービス

- Amazon CloudWatch Logs は、すべてのシステム、アプリケーション、および からのログを一元化 AWS のサービス し、ログをモニタリングして安全にアーカイブするのに役立ちます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。このパターンでは、Amazon S3 をファイル転送のストレージシステムとして使用します。
- AWS Transfer for SFTP は、SFTP プロトコルを介して AWS ストレージサービスとの間でファイルを転送するのに役立ちます。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

エピック

「VPC を作成する」

タスク	説明	必要なスキル
サブネットを持つ VPC を作成します。	Amazon VPC コンソールを開きます。2 つのパブリックサブネットを持つ仮想プライベートクラウド (VPC) を作成します。(2 番目のサブネットは高可用性を実現します。) -または- 添付された CloudFormation テンプレート `amazon-vpc-subnets.yml` を CloudFormation コンソールにデプロイして、このエピックのタスクを自動化できます。	デベロッパー、システム管理者
インターネットゲートウェイを追加します。	インターネットゲートウェイをプロビジョニングし、VPC に接続します。	デベロッパー、システム管理者
既存の IP を移行します。	Elastic IP アドレスに既存の IP アドレスをアタッチします。アドレスプールから Elastic IP アドレスを作成して使用できます。	デベロッパー、システム管理者

SFTP サーバーをプロビジョニングする

タスク	説明	必要なスキル
SFTP サーバーの作成	AWS Transfer Family コンソールを開きます。 AWS	デベロッパー、システム管理者

タスク	説明	必要なスキル
	Transfer Family ドキュメントの「サーバーのインターネット向けエンドポイントを作成する」の手順に従って、インターネット向けエンドポイントを持つ SFTP サーバーを作成します。[エンドポイントタイプ] で、[ホストされた VPC] を選択します。[アクセス] には、[インターネット接続] を選択します。VPC には、前のエピックで作成した VPC を選択します。 -または- 添付された CloudFormation テンプレート `amazon-sftp-server.yml` を CloudFormation コンソールにデプロイして、このエピックのタスクを自動化できます。	
ドメイン名を移行する。	既存のドメイン名をカスタムホスト名にアタッチします。新しいドメイン名を使用している場合は、Amazon Route 53 DNS エイリアスを使用してください。既存のドメイン名には、[その他 DNS] を選択します。詳細については、AWS Transfer Family ドキュメントの「カスタムホスト名の使用」を参照してください。	デベロッパー、システム管理者

タスク	説明	必要なスキル
CloudWatch ログ記録ロールを追加します。	(オプション) CloudWatch ロギングを有効にする場合は、CloudWatch Logs API オペレーション `logs:CreateLogGroup` 、`logs:CreateLogStream` 、 `logs:DescribeLogStreams` 、および `logs:PutLogEvents` を使用して Transfer ロールを作成します。詳細については、AWS Transfer Family ドキュメントのCloudWatch でのアクティビティのログ記録」を参照してください。	開発者、システム管理者
保存して送信する。	[保存] を選択します。[アクション] で [開始] を選択し、SFTP サーバーが [オンライン] というステータスで作成されるのを待ちます。	デベロッパー、システム管理者

Elastic IP アドレスをSFTP サーバーにマッピングする

タスク	説明	必要なスキル
設定を変更できるようにサーバーを停止する。	AWS Transfer Family コンソールで、サーバーを選択し、作成した SFTP サーバーを選択します。[アクション] で [停止] を選択します。サーバーがオフラインの場合は、[編集] を選択して設定を変更します。	開発者、システム管理者

タスク	説明	必要なスキル
アベイラビリティゾーンとサブネットを選択する。	[アベイラビリティゾーン] セクションで、VPC のアベイラビリティゾーンとサブネットを選択します。	デベロッパー、システム管理者
Elastic IP アドレスを追加する。	IPv4 アドレスの場合、各サブネットの Elastic IP アドレスを選択し、[保存] を選択します。	デベロッパー、システム管理者

ユーザーの追加

タスク	説明	必要なスキル
S3 バケットにアクセスするユーザーの IAM ロールを作成する。	`Transfer` の IAM ロールを作成し、S3 バケット名をリソースとして、 `s3:ListBucket` 、 `s3:GetBucketLocation` 、および `s3:PutObject` を追加します。詳細については、 AWS Transfer Family ドキュメントの「IAM ロールとポリシーの作成」を参照してください。 —または— 添付された CloudFormation テンプレート `amazon-sftp-customer.yml` を CloudFormation コンソールにデプロイして、このエピックのタスクを自動化できます。	デベロッパー、システム管理者

タスク	説明	必要なスキル
S3 バケットを作成する。	アプリケーション用の S3 バケット環境を作成します。	デベロッパー、システム管理者
オプションのフォルダを作成する。	(オプション) ユーザー用のファイルを特定の Amazon S3 フォルダに個別に保存する場合は、必要に応じてフォルダを追加します。	デベロッパー、システム管理者
SSH パブリックキーを作成する。	SSH キーペアを作成するには、 AWS Transfer Family ドキュメントの「SSH キーの生成」を参照してください。	デベロッパー、システム管理者
ユーザーを追加します。	AWS Transfer Family コンソールで、サーバーを選択し、作成した SFTP サーバーを選択し、ユーザーの追加を選択します。[ホームディレクトリ] には、作成した S3 バケットを選択します。[SSH 公開キー] の場合は、SSH キーペアの公開キー部分を指定します。SFTP サーバーのユーザーを追加し、[追加] を選択します。	デベロッパー、システム管理者

SFTP サーバーをテストする

タスク	説明	必要なスキル
セキュリティグループを作成する。	SFTP サーバーの [セキュリティグループ] セクションで、テストマシンの IP を追加し	開発者

タスク	説明	必要なスキル
	て SFTP アクセスを取得します。	
SFTP クライアントユーティリティを使用してサーバーをテストする。	任意の SFTP クライアントユーティリティを使用してファイル転送をテストします。クライアントのリストと手順については、 AWS Transfer Family ドキュメントの「クライアントを使用したファイルの転送」を参照してください。	開発者

関連リソース

- AWS Transfer Family ユーザーガイド
- Amazon S3 ユーザーガイド
- Amazon EC2 ドキュメントの「Elastic IP アドレス」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS アプリケーション移行サービスを使用してオンプレミス VM を Amazon EC2 に移行する

作成者: Thanh Nguyen (AWS)

概要

アプリケーションの移行に関しては、組織はさまざまな方法でアプリケーションのサーバーをオンプレミス環境から Amazon Web Services (AWS) クラウドにリホスト (リフトアンドシフト) できます。1 つの方法は、新しい Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをプロビジョニングし、アプリケーションを最初からインストールして設定することです。もう 1 つの方法は、サードパーティまたは AWS ネイティブの移行サービスを使用して、複数のサーバーを同時に移行することです。

このパターンは、AWS アプリケーション移行サービスを使用して、サポートされている仮想マシン (VM) を AWS クラウド上の Amazon EC2 インスタンスに移行する手順の概要を示しています。このパターンの方法を使用して、1 台または複数の仮想マシンを手動で 1 台ずつ移行することも、説明されている手順に基づいて適切な自動化スクリプトを作成して自動的に移行することもできます。

前提条件と制限

前提条件

- アプリケーション移行サービスをサポートするいずれかの AWS リージョンのアクティブな AWS アカウント
- AWS Direct Connect または仮想プライベートネットワーク (VPN) を使用するプライベートネットワーク、またはインターネットを介した、ソースサーバーとターゲット EC2 サーバー間のネットワーク接続

制限

- サポートされているリージョンの最新リストについては、「Supported AWS Regions」を参照してください。
- サポートされているオペレーティングシステムのリストについては、「Supported operating systems」と「Amazon EC2 よくある質問」の「全般」セクションを参照してください。

アーキテクチャ

ソーステクノロジースタック

- Amazon EC2 がサポートするオペレーティングシステムを実行する、物理、仮想、またはクラウドでホストされるサーバー

ターゲットテクノロジースタック

- ソース VM と同じオペレーティングシステムを実行している Amazon EC2 インスタンス
- Amazon Elastic Block Store (Amazon EBS)

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、ソリューションの高レベルアーキテクチャと主要コンポーネントを示しています。オンプレミスデータセンターには、ローカルディスクを備えた仮想マシンがあります。AWS には、レプリケーションサーバーを含むステージングエリアと、テストとカットオーバー用の EC2 インスタンスを含む移行済みリソースエリアがあります。どちらのサブネットにも EBS ボリュームが含まれています。

1. AWS アプリケーション移行サービスを初期化します。
2. ステージングエリアサーバーの設定とレポート (ステージングエリアリソースを含む) を設定します。
3. ソースサーバーにエージェントをインストールし、ブロックレベルの継続的なデータ複製 (圧縮および暗号化) を行います。
4. オーケストレーションとシステム変換を自動化して、カットオーバー時間を短縮します。

ネットワークアーキテクチャ

次の図は、オンプレミスデータセンターと AWS の主要コンポーネント間の通信に必要なプロトコルとポートなど、ネットワークの観点から見たソリューションの大まかなアーキテクチャと主要コンポーネントを示しています。

ツール

- AWS Application Migration Service を使用すると、変更を加えることなく、最小限のダウンタイムで、アプリケーションを AWS クラウドにリホスト (リフトアンドシフト) できます。

ベストプラクティス

- ターゲット EC2 インスタンスへのカットオーバーが完了するまで、ソースサーバーをオフラインにしたり、再起動したりしないでください。
- ユーザーがターゲットサーバーでユーザー承認テスト (UAT) を実施して、問題を特定して解決するための十分な機会を提供してください。理想的には、このテストはカットオーバーの少なくとも 2 週間前に開始する必要があります。
- Application Migration Service コンソールでサーバーのレプリケーションステータスを頻繁に監視して、問題を早期に特定してください。
- エージェントのインストールには、永続的な IAM ユーザー認証情報の代わりに、一時的な AWS Identity and Access Management (IAM) 認証情報を使用します。

エピック

AWS 認証情報を生成する

タスク	説明	必要なスキル
AWS レプリケーションエージェント IAM ロールを作成する。	管理者権限を使用して AWS アカウントにサインインします。 AWS Identity and Access Management (IAM) コンソールで、IAM ロールを作成します。 1. IAM コンソールで、[ロール] を選択します。 2. [ロールの作成] を選択します。 3. [信頼できるエンティティの選択] ページの [信頼できるエンティティタイプ] セクションで、[AWS アカウント] を選択します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	4. [AWS アカウント] セクションで、[このアカウント (<account-id>)] を選択します。 5. [次へ] を選択します。 6. [アクセス権限の追加] ページで `AWSApplicationMigrationAgentInstallationPolicy` ポリシーを検索し、ポリシー名の横にあるチェックボックスをオンにします。 7. [次へ] を選択します。 8. ロールの詳細ページで、ロール名として MGN_Agent_installation_Role と入力します。 9. フィールドが正しいことを確認し、[ロールの作成] を選択します。	

タスク	説明	必要なスキル
一時的なセキュリティ認証情報を生成する。	AWS コマンドラインインターフェイス (AWS CLI) がインストールされているマシンで、管理者権限でサインインします。または (サポートされている AWS リージョン内で) AWS マネジメントコンソールで、管理者権限で AWS アカウントにサインインし、AWS CloudShell を開きます。 次のコマンドで一時的な認証情報を生成し、AWS アカウント ID に `<account-id>` を置き換えます。 `aws sts assume-role --role-arn arn:aws:iam::<account-id>:role/MGN_Agent_Installation_Role --role-session-name mgn_installation_session_role` コマンドの出力から、`AccessKeyId` 、`SecretAccessKey` 、および `SessionToken` の値をコピーします。後で使用できるように安全な場所に保管してください。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	Important これらの一時的な認証情報は 1 時間後に期限切れになります。1 時間後に認証情報が必要な場合は、前のステップを繰り返します。	

アプリケーション移行サービスを初期化し、レプリケーション設定テンプレートを作成します。

タスク	説明	必要なスキル
サービスを初期化する。	コンソールで、管理者権限を使用して AWS アカウントにサインインします。 [アプリケーション移行サービス] を選択し、[はじめに] を選択します。	AWS 管理者、移行エンジニア
レプリケーション設定テンプレートを作成して設定する。	1. 次の設定の詳細を入力します。 a. ステージングエリアのサブネットを選択します。 b. レプリケーションサーバーのインスタンスタイプを選択します (デフォルトでは `t3.small`)。 c. EBS ボリュームタイプ (デフォルトでは gp3) を選択します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	d. EBS 暗号化オプションを選択します。 e. [常にアプリケーション移行サービスのセキュリティグループを使用する] チェックボックスがオンになっていることを確認します。 f. オンプレミス環境と AWS 間でプライベートネットワーク接続を使用している場合は、[データ複製 (VPN、DirectConnect、VPC ピアリング) にプライベート IP を使用する] チェックボックスを選択します。 g. Application Migration Service のネットワーク帯域幅を制限する場合は、[ネットワーク帯域幅を調整する (サーバーごと - Mbps 単位)] チェックボックスをオンにします。 2. [テンプレートの作成] を選択します。 アプリケーション移行サービスは、データの複製と移行されたサーバーの起動を円滑に進めるために必要なすべての	

タスク	説明	必要なスキル
	IAM ロールを自動的に作成します。	

ソースマシンに AWS Replication Agent をインストールする

タスク	説明	必要なスキル
必要な AWS 認証情報を用意する。	ソースサーバーでインストーラファイルを実行するときは、`AccessKeyId` 、`SecretAccessKey` 、`SessionToken` など、以前に生成した一時的な認証情報を入力する必要があります。	移行エンジニア、AWS 管理者
Linux サーバー用のエージェントをインストールする。	インストーラコマンドをコピーし、ソースサーバーにログインし、インストーラを実行します。詳細な手順については、AWS のドキュメントを参照してください。	AWS 管理者、移行エンジニア
Windows サーバー用のエージェントをインストールする。	インストーラファイルを各サーバーにダウンロードし、インストーラコマンドを実行します。詳細な手順については、AWS のドキュメントを参照してください。	AWS 管理者、移行エンジニア
初期データ複製が完了するまで待つ。	エージェントがインストールされると、ソースサーバーがアプリケーション移行サービスコンソールの [ソースサーバー] セクションに表示されま	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	す。サーバーが初期データ複製を行うまで待ちます。	

起動設定の構成

タスク	説明	必要なスキル
サーバーの詳細を指定する。	アプリケーション移行サービスコンソールで、「ソースサーバー」セクションを選択し、一覧からサーバー名を選択してサーバーの詳細にアクセスします。	AWS 管理者、移行エンジニア
起動設定を設定する。	[起動設定] タブを選択します。一般的な起動設定や EC2 起動テンプレート設定など、さまざまな設定を行うことができます。詳細な手順については、AWS のドキュメントを参照してください。	AWS 管理者、移行エンジニア

テストを実行

タスク	説明	必要なスキル
ソースサーバーをテストする。	1. Application Migration Service コンソールの [ソースサーバー] セクションで、ソースサーバーの [移行ライフサイクル] が [テスト準備完了] で、[データ複製ステータス] が [正常] であることを確認します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	2. 各ソースサーバーの左にあるチェックボックスを選択します。 3. [テストとカットオーバー] を選択し、[テストインスタンスを起動] を選択します。 4. プロンプトが表示されたら、[起動] を選択します。 サーバーが起動します。	
テストが正常に完了したことを確認します。	テストサーバーが完全に起動すると、ページ上の [アラート] ステータスに各サーバーが [起動済み] と表示されます。	AWS 管理者、移行エンジニア
サーバーをテストする。	テストサーバーに対してテストを行い、期待どおりに機能することを確認します。	AWS 管理者、移行エンジニア

カットオーバーのスケジュール設定と実行

タスク	説明	必要なスキル
カットオーバーウィンドウのスケジュールを設定する。	関連チームと適切なカットオーバー期間のスケジュールを設定してください。	AWS 管理者、移行エンジニア
カットオーバーを実行します。	1. アプリケーション移行コンソールの [ソースサーバー] ページで、各ソースサーバーの左側にあるチェックボックスを選択します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
	2. [テストとカットオーバー] を選択し、[カットオーバーの準備ができているとしてマークする] を選択します。 3. 各ソースサーバーの [移行ライフサイクル] が [カットオーバーの準備ができている] であることを確認します。 4. [テストとカットオーバー] を選択し、[カットオーバーインスタンスを起動] を選択します。 5. プロンプトが表示されたら、[起動] を選択します。サーバーが起動します。 ソースサーバーの [移行ライフサイクル] が [カットオーバー中] に変わります。	
カットオーバーが正常に完了したことを確認します。	カットオーバーサーバーが完全に起動すると、各サーバーの [ソースサーバー] ページの [アラート] ステータスに [起動済み] と表示されます。	AWS 管理者、移行エンジニア
サーバーをテストする。	カットオーバーサーバーに対してテストを行い、想定どおりに機能することを確認します。	AWS 管理者、移行エンジニア

タスク	説明	必要なスキル
カットオーバーを確定する。	[テストとカットオーバー] を選択し、[カットオーバーの完了] を選択して移行プロセスを終了します。	AWS 管理者、移行エンジニア

関連リソース

- AWS Application Migration Service
- AWS Application Migration Service ユーザーガイド

AWS SFTP を使用して小規模なデータセットをオンプレミスから Amazon S3 に移行する

作成者: Charles Gibson (AWS) と Sergiy Shevchenko (AWS)

概要

このパターンでは、AWS Transfer for SFTP (AWS SFTP) を使用して、小規模なデータセット (5 TB 以下) をオンプレミスのデータセンターから Amazon Simple Storage Service (Amazon S3) に移行する方法について説明します。 データはデータベースダンプでもフラットファイルでもかまいません。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- データセンターと AWS の間に AWS Direct Connect リンクが確立されている

機能制限

- データファイルは 5 TB 未満である必要があります。5 TB を超えるファイルの場合は、Amazon S3 へのマルチパートアップロードを実行するか、別のデータ転送方法を選択できます。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスのフラットファイルまたはデータベースダンプ

ターゲットテクノロジースタック

- Amazon S3

ソースアーキテクチャとターゲットアーキテクチャ

ツール

- AWS SFTP — セキュアファイル転送プロトコル (SFTP) を使用して、Amazon S3 との間で直接ファイルを転送できます。
- AWS Direct Connect — オンプレミスのデータセンターから AWS への専用ネットワーク接続を確立します。
- VPC エンドポイント — AWS PrivateLink を使用してサポートされている AWS のサービスや VPC エンドポイントサービスに VPC をプライベートに接続できます。インターネットゲートウェイ、ネットワークアドレス変換 (NAT)デバイス、VPN 接続、または AWS Direct Connect 接続は必要ありません。VPC のインスタンスでは、サービスのリソースと通信するためにパブリック IP アドレスを必要としません。

エピック

移行の準備をする

タスク	説明	必要なスキル
現在の SFTP 要件を文書化してください。		アプリ所有者、SA
認証要件を特定します。	要件には、キーベースの認証、ユーザー名またはパスワード、または ID プロバイダー (IdP) が含まれる場合があります。	アプリ所有者、SA
アプリケーション統合要件を特定する。		アプリ所有者
サービスを必要とするユーザーを特定します。		アプリ所有者
SFTP サーバーエンドポイントの DNS 名を決定します。		ネットワーク
バックアップ戦略を決定します。		SA、DBA (データが転送される場合)

タスク	説明	必要なスキル
アプリケーションの移行またはカットオーバー戦略を特定します。		アプリ所有者、SA、DBA

インフラストラクチャを設定する

タスク	説明	必要なスキル
AWS アカウントに 1 つ以上の仮想プライベートクラウド (VPC) とサブネットを作成します。		AMS のアプリ所有者
セキュリティグループとネットワークアクセスコントロールリスト (ACL)		ネットワークとセキュリティ
S3 バケットを作成する。		AMS のアプリ所有者
AWS Identity and Access Management (IAM) ロールの ARN。	S3 バケットにアクセスするためのアクセス許可が含まれるIAM ポリシーを作成します。この IAM ポリシーは、ユーザーに提供するアクセスのレベルを決定します。他の IAM ポリシーを作成し、AWS SFTP と信頼関係を構築します。	セキュリティ、AMS
登録済みドメインを関連付けます (オプション)。	独自の登録済みドメインを所有している場合は、そのドメインを SFTP サーバーに関連付けることができます。SFTP トラフィックを といったドメインから、または といったサ	ネットワーク、AMS

タスク	説明	必要なスキル
	ブドメインから、SFTP サーバーエンドポイントにルーティングできます。	
SFTP サーバーの作成	SFTP サーバーを作成し、サービスが使用する ID プロバイダータイプを指定して、ユーザーを認証します。	AMS のアプリ所有者
SFTP クライアントを開きます。	SFTP クライアントを開いて接続を設定し、使用する SFTP サーバーの SFTP エンドポイントホスト名を使用します。 はすべての標準 SFTP クライアントをサポートしています。一般的に使用されているSFTPクライアントには、OpenSSH、WinSCP、サイバーダック、FileZilla などがあります。SFTP サーバーのホスト名は AWS SFTP コンソールから取得できます。	AMS のアプリ所有者

プランニングとテスト

タスク	説明	必要なスキル
アプリケーションの移行を計画する。	必要なアプリケーション構成の変更を計画し、移行日を設定し、テストスケジュールを決定します。	AMS のアプリ所有者
インフラストラクチャをテストする	本番稼働用環境以外でテストする	AMS のアプリ所有者

関連リソース

リファレンス

- SFTP 用 AWS Transfer ユーザーガイド
- AWS Direct Connect のリソース
- VPC エンドポイント

チュートリアルと動画

- AWS Transfer for SFTP
- SFTP 用 AWS Transfer ユーザーガイド
- AWS SA ホワイトボード — Direct Connect (ビデオ)

Oracle GlassFish から AWS Elastic Beanstalk へ移行する

作成者： Saandeep Bondugula (AWS)

概要

このパターンでは、オンプレミスの Oracle GlassFish サーバー上で実行される Java アプリケーションから、AWS クラウド上の AWS Elastic Beanstalk に移行する方法について説明します。

AWS では、Amazon Elastic Compute Cloud (Amazon EC2) のAuto Scaling グループで動作する AWS Elastic Beanstalk を搭載した Docker GlassFish サーバー上に Java アプリケーションがデプロイされます。

その他の機能

- Amazon Elastic Beanstalk は、基盤となる複数のリソースのラッパーとして機能します。Amazon Route 53 からの着信トラフィックを処理する耐障害性ロードバランサーを設定し、そのトラフィックを 1 つ以上の EC2 インスタンスに分散することで、デプロイツールとしても機能します。
- オンプレミスデータベースを Amazon Relational Database Service (Amazon RDS) に移行するには、データベース接続の詳細を更新します。バックエンドデータベースでは、Amazon RDS マルチ AZ 配置を設定して、データベースエンジンタイプを選択できます。
- マルチ AZ 配置を導入することで高可用性を実現し、Auto Scaling グループと拡張ポリシーを使用して耐障害性を向上させることができます。
- CPU 使用率などの Amazon CloudWatch メトリクスに基づき、スケーリングポリシーを設定できます。
- AWS Elastic Beanstalk では、基盤となる Elastic Load Balancing の設定と Amazon EC2 Auto Scaling を設定できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- GlassFish 上で動作するオンプレミスの Java アプリケーション
- Java ウェブアプリケーションリソース (WAR) ファイル

製品バージョン

- Oracle Glassfish 4.1.2 と 5.0
- Java 7 GlassFish 4.0
- Java 8 GlassFish 4.1 以降

アーキテクチャ

ソーステクノロジースタック

- GlassFish で開発されたアプリケーション

ターゲットテクノロジースタック

- Elastic Beanstalk

ターゲット アーキテクチャ

デプロイのワークフロー

ツール

- Amazon Elastic Beanstalk は、Java、.NET、PHP、Node.js、Python、Ruby、Go、Docker で開発されたウェブアプリケーションとサービスを、Apache、Nginx、Passenger、IIS などの一般的なサーバーにデプロイしてスケーリングするサービスです。
- Amazon CloudWatch では、データと実用的なインサイトを利用して、アプリケーションのモニタリング、システム全体のパフォーマンスの変化に関する理解と対応、リソース使用率の最適化、および運用状態の統合的な確認を行うことができます。
- Docker — ソフトウェアを標準化されたユニットにパッケージ化して、アプリケーションを迅速に構築、テスト、およびデプロイするプラットフォーム。
- Java — 汎用プログラミング言語。Java はクラスベースのオブジェクト指向であり、実装の依存性が少なくなるように設計されています。

エピック

VPC をセットアップする

タスク	説明	必要なスキル
必要な情報を使用して、仮想プライベートクラウド (VPC) インスタンスを作成します。		SysAdmin
VPC 内に最低 2 つのサブネットを作成します。		SysAdmin
要件ごとにルートテーブルを作成します。		SysAdmin

Amazon SNS のセットアップ

タスク	説明	必要なスキル
Amazon Simple Storage Service (Amazon S3) バケットを作成します。		SysAdmin
WAR ファイルを S3 バケットにコピーし、アプリケーションコードをアップロードします。		SysAdmin

IAM ロールを作成する

タスク	説明	必要なスキル
(オプション) AWS Identity and Access Management (IAM) ロールを作成します。	デフォルトの "aws-elasticbeanstalk-ec2-role" プロファイルを使用するか、Elast	SysAdmin

タスク	説明	必要なスキル
	ic Beanstalk に自動的に作成させることができます。	

Elastic Beanstalk について

タスク	説明	必要なスキル
Elastic Beanstalk ダッシュボードを開きます。		SysAdmin
新しいアプリケーションを作成し、ウェブサーバー環境を選択します。		SysAdmin
事前設定されたプラットフォームで、GlassFish Docker を選択します。		SysAdmin
コードをアップロードする	S3 バケットファイルの URL またはローカルシステムファイルから ZIP ファイルを指定します。	SysAdmin
[環境タイプ] を選択します。	[設定容量] で、単一インスタンスまたはロードバランサーを選択します。	SysAdmin
ロードバランサーを設定します。	前のステップでロードバランサーを選択した場合は、マルチ AZ 配置を設定します。	SysAdmin
[設定セキュリティ] で、以前に作成した IAM ロールを選択します。		SysAdmin

タスク	説明	必要なスキル
[設定セキュリティ] で、既存のキーペアがある場合は、直接使用するか、新しい Amazon EC2 キーペアを作成します。		SysAdmin
[設定モニタリング] で、Amazon CloudWatch を設定します。		SysAdmin
[設定セキュリティ] で、以前に作成した VPC を選択します。		SysAdmin
[環境の作成] を選択します。		SysAdmin

アプリケーションをテストする

タスク	説明	必要なスキル
作成された環境で提供される URL を使用して、アプリケーションをテストします。		
Amazon Route 53 でドメインネームサービス (DNS) の変更を適用します。		

関連リソース

- Oracle GlassFish のドキュメント
- 「GlassFish オープンソース Java EE リファレンスの実装」」
- AWS Elastic Beanstalk のドキュメント
- 「Amazon CloudWatch で Elastic Beanstalk を使用する」

- 「AWS Elastic Beanstalk」
- 「Auto Scaling グループ」
- 「Auto Scaling グループのサイズのスケーリング」
- 「Amazon RDS マルチ AZ 配置」

オンプレミスの Oracle データベースを Oracle Amazon EC2 に移行する

作成者: Baji Shaik (AWS)、Pankaj Choudhary (AWS)

概要

このパターンでは、オンプレミスの Oracle データベースを Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の Oracle に移行する手順について説明します。AWS Data Migration Services (AWS DMS) を使用するか、RMAN、Data Pump インポート/エクスポート、 トランスポータブル表領域、Oracle GoldenGate などのネイティブ Oracle ツールを使用するかの 2 つの移行オプションについて説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターにあるソース Oracle データベース

機能制限

- ターゲットオペレーティングシステム (OS) は Amazon EC2 でサポートされている必要があります。サポートされているシステムの完全なリストについては、「Amazon EC2 のよくある質問」を参照してください。

製品バージョン

- Enterprise、Standard、Standard One、および Standard Two エディションの Oracle バージョン 10.2 以降 (バージョン 10.x)、11g、12.2 および 18c まで。AWS DMS でサポートされているバージョンの最新リストについては、AWS DMS ドキュメントの「データ移行ソース」にある「オンプレミスと Amazon EC2 インスタンスデータベース」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース

ターゲットテクノロジースタック

- Amazon EC2 上の Oracle データベースインスタンス

ターゲット アーキテクチャ

データ移行アーキテクチャ

AWS DMS を使用する:

ネイティブ Oracle ツールの使用:

ツール

- AWS DMS — AWS Database Migration Service (AWS DMS) は、複数のソースデータベースとターゲットデータベースをサポートします。サポートされているデータベースのバージョンとエディションについては、「AWS DMS のソースとして Oracle データベースを使用する」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。
- ネイティブ Oracle ツール - RMAN、Data Pump インポート/エクスポート、トランスポータブル表領域、Oracle GoldenGate

エピック

移行を計画する

タスク	説明	必要なスキル
ソースデータベースとターゲットデータベースのバージョンを検証します。		DBA
ターゲット OS のバージョンを特定します。		DBA、SysAdmin

タスク	説明	必要なスキル
Oracle 互換性リストと容量要件に基づいて、ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
ネットワーク要件 (レイテンシーと帯域幅) を特定します。		DBA、SysAdmin
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、SysAdmin
ソースデータベースおよびターゲットデータベースのネットワークアクセスのセキュリティ要件を特定します。		DBA、SysAdmin
Oracleソフトウェアのインストールに必要な OS ユーザーのリストを決定します。		DBA、SysAdmin
AWS Schema Conversion Tool (AWS SCT) とドライバーをダウンロードします。		DBA
ワークロード用の AWS SCT プロジェクトを作成し、ソースデータベースに接続します。		DBA

タスク	説明	必要なスキル
オブジェクト (テーブル、インデックス、シーケンスなど) を作成する SQL ファイルを生成します。		DBA
バックアップ戦略を決定します。		DBA、SysAdmin
可用性の要件を決定します。		DBA
アプリケーションの移行/切り替え戦略を特定します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
AWS アカウントに仮想プライベートクラウド (VPC) を作成します。		SysAdmin
セキュリティグループとネットワークアクセスコントロールリスト (ACL) を作成します。		SysAdmin
EC2 インスタンスを設定して起動します。		SysAdmin

Oracle ソフトウェアをインストールします。

タスク	説明	必要なスキル
Oracle ソフトウェアに必要な OS ユーザーとグループを作成します。		DBA、SysAdmin
Oracle ソフトウェアの必要なバージョンをダウンロードします。		
EC2 インスタンスに Oracle ソフトウェアをインストールします。		DBA、SysAdmin
AWS SCT で生成されたスクリプトを使用して、テーブル、プライマリキー、ビュー、シーケンスなどのオブジェクトを作成します。		DBA

データ移行 — オプション 1

タスク	説明	必要なスキル
ネイティブ Oracle ツールまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。	Oracle ツールには、Data Pump インポート/エクスポート、RMAN、トランスポータブル表領域、および GoldenGate が含まれます。	DBA

データ移行 — オプション 2

タスク	説明	必要なスキル
移行方法を決定します。		DBA
AWS DMS コンソールからレプリケーションインスタンスを作成します。		DBA
ソースおよびターゲットエンドポイントを作成します。		DBA
レプリケーションタスクを作成します。		DBA
変更データキャプチャ (CDC) を有効にして、継続的なレプリケーションの変更をキャプチャします。		DBA
レプリケーションタスクを実行し、ログをモニタリングします。		DBA
完全なロードが完了したら、インデックスや外部キーなどのセカンダリオブジェクトを作成します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、SysAdmin、アプリ所有者

カットオーバー

タスク	説明	必要なスキル
アプリケーションのカットオーバー/スイッチオーバー戦略に従ってください。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS Secrets Manager リソースをシャットダウンします。		DBA、SysAdmin
プロジェクト文書を確認して検証する。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動タスクとツールによるタスクの割合、コスト削減などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを終了し、フィードバックを提供します。		

関連リソース

リファレンス

- 「Oracle データベースを AWS に移行するための戦略」
- 「AWS クラウドへの Oracle データベースの移行」
- 「Amazon EC2 ウェブサイト」
- AWS DMS ウェブサイト

- AWS DMS に関するブログ投稿
- 「Amazon EC2 の料金」
- 「クラウドコンピューティング環境における Oracle ソフトウェアのライセンス」

チュートリアルと動画

- 「Amazon EC2 の開始方法」
- AWS DMS の使用開始
- Amazon EC2 のご紹介 - Elastic クラウドサーバーと AWS でのホスティング (動画)

Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon EC2 に移行する

ナバカンス・タルリ (AWS) によって作成されました

概要

データベースを移行するときは、ソースとターゲットのデータベースエンジンとバージョン、移行ツールとサービス、許容されるダウンタイム期間などの要素を考慮する必要があります。オンプレミスの Oracle データベースを Amazon Elastic Compute Cloud (Amazon EC2) に移行する場合は、Oracle Data Pump や Oracle Recovery Manager (RMAN) などのOracleのツールを使用できます。詳細については、ガイド AWS クラウドへの Oracle データベースの移行を参照してください。

Oracle Data Pump は、データベースの論理的で一貫性のあるバックアップを抽出し、ターゲット EC2 インスタンスに復元するのに役立ちます。このパターンでは、Oracle Data Pump `NETWORK_LINK` とパラメータを使用して、オンプレミスの Oracle データベースを EC2 インスタンスに移行する方法を、最小限のダウンタイムで説明します。この `NETWORK_LINK` パラメータは、データベースリンクを通じてインポートを開始します。ターゲット EC2 インスタンスの Oracle Data Pump Import (impdp) クライアントは、ソースデータベースに接続し、そこからデータを取得して、ターゲットインスタンスのデータベースに直接データを書き込みます。このソリューションではバックアップファイルやダンプファイルは使用されません。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 以下の機能を備えたオンプレミスの Oracle データベース
  - Oracle・リアル・アプリケーション・クラスター (RAC) データベースではない
  - Oracle 自動ストレージ管理 (Oracle ASM) データベースではない
  - 読み書きモードです。
- オンプレミスデータセンターと AWS の間に AWS Direct Connect リンクが作成されました。詳細については、接続の作成 (Direct Connect ドキュメント)を参照してください。

製品バージョン

- Oracle Database 12c リリース 1 (12.1.0.2.v14) 以降

アーキテクチャ

ソーステクノロジースタック

- オンプレミス・データセンター内のスタンドアロン (非RACおよび非ASM) Oracleデータベース・サーバー

ターゲットテクノロジースタック

- Amazon EC2 上で実行されている Oracle データベース

ターゲット アーキテクチャ

AWS Well-Architected フレームワーク の 信頼性の柱 は、高い可用性と耐障害性を提供するためにデータのバックアップを作成することを推奨しています。詳細については、AWS で Oracle Database を実行するためのベストプラクティスの 高可用性を実現するアーキテクチャ を参照してください。このパターンでは、Oracle Active Data Guard を使用して EC2 インスタンスにプライマリデータベースとスタンバイデータベースを設定します。高可用性を実現するには、EC2 インスタンスを異なるアベイラビリティーゾーンに配置する必要があります。ただし、アベイラビリティーゾーンは同じ AWS リージョン内に存在することも、異なる AWS リージョン内に存在することもできます。

Oracle アクティブデータガードは、クエリ、ソート、レポート、その他の読み取り操作を行う際に、フィジカル・スタンバイ・データベースへの読み取り専用アクセスを提供しますが、その間、プライマリ・データベースからREDOの変更を継続的に適用します。目標復旧時点 (RPO) と目標復旧時間 (RTO) に基づいて、同期と非同期のREDO転送オプションを選択できます。

以下の画像は、プライマリ EC2 インスタンスとスタンバイ EC 2 インスタンスが異なる AWS リージョンにある場合のターゲットアーキテクチャを示しています。

データ移行アーキテクチャ

ターゲットアーキテクチャの設定が完了したら、Oracle Data Pump を使用してオンプレミスのデータとスキー マをプライマリ EC2 インスタンスに移行します。カットオーバー中、アプリケーションはオンプレミスデータベースまたはターゲットデータベースにアクセスできません。これらのアプリケーションは、プライマリ EC2 インスタンスの新しいターゲットデータベースに接続できるよう になるまでシャットダウンします。

以下の画像は、データ移行中のアーキテクチャを示しています。このサンプルアーキテクチャでは、プライマリ EC2 インスタンスとスタンバイ EC2 インスタンスは異なる AWS リージョンにあります。

ツール

サービス

- AWS Direct Connect は、標準のイーサネット光ファイバーケーブルを介して内部ネットワークを Direct Connect の場所にリンクします。この接続を使用すると、Amazon S3 などの のパブリッククサービス、または Amazon VPC に対する仮想インターフェイスを直接作成できるため、ネットワークパスのインターネットサービスプロバイダーを回避できます。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

その他のツールとサービス

- Oracle Active Data Guard は、スタンバイ・データベースの作成、保守、管理、監視に役立ちます。
- Oracle Data Pump を使用すると、データやメタデータをあるデータベースから別のデータベースに高速に移動できます。

ベストプラクティス

- AWS で Oracle Database を実行するためのベストプラクティス
- NETWORK_LINK を使用してデータをインポートします

エピック

AWS で EC2 インスタンスをセットアップする

タスク	説明	必要なスキル
オンプレミスホストのソースハードウェア構成とカーネルパラメータを 特定します。	ストレージサイズ、1 秒あたりの入出力オペレーション (IOPS)、CPU など、オンプレミス設定を検証します。これは CPU コアに基づく Oracle ライセンスにとって重要です。	DBA、SysAdmin
AWS でインフラストラクチャを作成します。	仮想プライベートクラウド (VPC)、プライベートサブネット、セキュリテ ィグループ、ネットワークアクセスコントロールリスト (ACL)、ルートテーブル、インターネットゲートウェイを作成します。詳細については次を参照してください: • VPC とサブネット • チュートリアル： DB インスタンス用の Amazon VPC を作成する	AWS システム管理者
アクティブデータガードを使用して EC2 インスタンスをセットアップします。	AWS Well-Architected フレームワーク で説明されているように、アクティブデータガード設定を使用して AWS EC2 インスタンスを設定します。EC2 インスタンス上の Oracle Database のバージョンは、オンプレミスバージョンと異なる場合があります。	AWS システム管理者

タスク	説明	必要なスキル
	これは、このパターンでは論理バックアップが使用されるためです。次の点に注意してください: • ターゲットデータベースを読み書きモードにします。 • ターゲットデータベースで、ソースデータベースのトランスペアレント ネットワーク基板 (TNS) の詳細を指定します。 詳細については、以下を参照してください。 • データベースの起動 (Oracle ドキュメント) • Oracle データベースの作成と設定 (Oracle ドキュメント)	

データベースを Amazon EC2 に移行する

タスク	説明	必要なスキル
EC2 インスタンスからオンプレミスデータベースへの dblink を作成します。	EC2 インスタンスの Oracle データベースとオンプレミスの Oracle データベースとの間にデータベースリンク (dblink) を作成します。詳細については、ネットワークリンクインポートによるデータの移	DBA

タスク	説明	必要なスキル
	動 (Oracle ドキュメント) を参照してください。	
EC2 インスタンスとオンプレミスホスト間の接続を確認します。	dblink を使用して、EC2 インスタンスとオンプレミスデータベース間の接続が機能していることを確認します。手順については、データベースリンクの作成 (Oracle ドキュメント) を参照してください。	DBA
オンプレミスデータベースに接続されているすべてのアプリケーション を停止します。	データベースのダウンタイムが承認されたら、オンプレミスデータベースに接続しているすべてのアプリケーションと依存ジョブを停止します。これは、アプリケーションから直接実行することも、cron を使用してデータベースから実行することもできます。詳細については、Oracle Linux で Crontab ユーティリティを使用してタスクをスケジュールするを参照してください。	DBA、アプリケーション開発者

タスク	説明	必要なスキル
データ移行ジョブをスケジュールします。	ターゲットホストで、impdb コマンドを使用して Data Pump のインポートをスケジュールします。これにより、ターゲットデータベースがオンプレミスホストに接続され、データ移行が開始されます。詳細については、Data Pump Import と NETWORK_LINK(Oracle ドキュメント) を参照してください。	DBA
データ移行を検証します。	データ検証は重要なステップです。データ検証には、dblink と SQL クエリの組み合わせなど、カスタムツールや Oracle ツールを使用できます。	DBA

カットオーバー

タスク	説明	必要なスキル
移行元データベースは、特別な読み取り専用モードとなります。	アプリケーションがシャットダウンされ、ソースデータベースに、変更が加えられていないことを確認します。ソースデータベースを読み取り専用モードで開きます。これにより、未処理のトランザクションを回避できます。詳細については、SQL ステートメントの ALTER DATABASE を参照してください。	DevOps エンジニア、開発者

タスク	説明	必要なスキル
オブジェクト数とデータを検証します。	データ検証には、dblink と SQL クエリの組み合わせなど、カスタムツールや Oracle ツールを使用できます。	DBA、アプリケーション開発者
プライマリ EC2 インスタンス上の新しいデータベースに接続するアプリケーション。	プライマリ EC2 インスタンスに作成した新しいデータベースを指すようにアプリケーションの接続属性を変更します。	DBA、アプリケーション開発者
アプリケーションのパフォーマンスを検証します。	アプリケーションの起動 Automated Workload Repository (Oracle ドキュメント) を使用して、アプリケーションの機能とパフォーマンスを検証します。	アプリ開発者、DevOps エンジニア

関連リソース

AWS リファレンス

- 「AWS クラウドへの Oracle データベースの移行」
- Amazon EC2 for Oracle
- クロスプラットフォーム環境の場合にバルク Oracle データベースを AWS に移行します
- VPC とサブネット
- チュートリアル：DB インスタンス用の Amazon VPC を作成する

Oracleのリファレンス

- Oracle Data Guard 設定
- データポンプインポート

AWS MGN を使用して RHEL BYOL システムを AWS ライセンス込みのインスタンスに移行する

作成者:Mike Kuznetsov (AWS)

概要

AWS Application Migration Service (AWS MGN) を使用してワークロードを AWS に移行する場合、移行中に Red Hat Enterprise Linux (RHEL) インスタンスをリフトアンドシフト (リホスト) し、ライセンスをデフォルトの Bring Your Own License (BYOL) モデルから AWS ライセンス付属 (LI) モデルに変更しなければならない場合があります。AWS MGN は Amazon マシンイメージ (AMI) ID を使用するスケーラブルなアプローチをサポートしています。このパターンは、大規模なリホスト移行中に RHEL サーバーのライセンス変更を行う方法を説明しています。また、Amazon Elastic Compute Cloud (Amazon EC2) ですでに実行している Red System のライセンスを変更する方法についても説明します。

前提条件と制限

前提条件

- ターゲット AWS アカウントへのアクセス
- 移行対象の AWS アカウントとリージョンで AWS MGN を初期化しました (オンプレミスシステムから AWS にすでに移行している場合は不要)
- 有効な RHEL ライセンスを持つソース RHEL サーバー

アーキテクチャ

このパターンは 2 つのシナリオをカバーしています。

- AWS MGN を使用して、システムをオンプレミスから AWS LI インスタンスに直接移行します。このシナリオでは、最初のエピック (LI インスタンスに移行 — オプション 1) と 3 番目のエピックの指示に従います。
- Amazon EC2 ですでに稼働している RHEL システムのライセンスモデルを BYOL から LI に変更します。このシナリオでは、2 番目のエピック (LI インスタンスへの移行 — オプション 2) と 3 番目のエピックの指示に従います。

> **Note**
>
> 3 番目のエピックでは、AWS が提供する Red Hat Update Infrastructure (RHUI) サーバーを使用するように新しい RHEL インスタンスを再設定します。このプロセスはどちらのシナリオでも同じです。

ツール

AWS サービス

- 「AWS Application Migration Service (AWS MGN)」を使用すると、変更を加えることなく、ダウンタイムを最小限に抑えながら、アプリケーションを AWS クラウドにリホスト (リフトアンドシフト) できます。

エピック

LI インスタンスへの移行 — オプション 1 (オンプレミスの RHEL システム用)

タスク	説明	必要なスキル
ターゲットリージョンの RHEL AWS LI インスタンスの AMI ID を検索します。	「AWS Marketplace」にアクセスするか、「Amazon EC2 コンソール」を使用して RHEL ソースシステムのバージョン (RHEL-7.7 など) と一致する RHEL AMI ID を探し、その AMI ID を書き留めます。Amazon EC2 コンソールでは、次の検索用語のいずれかを使用して AMI をフィルタリングできます。 • 説明 = Red Hat, Inc. 提供 • AMI 名 = RHEL-7.7	クラウド管理者
AWS MGN の起動設定を行います。	1. 「AWS MGN コンソール」で、ソース RHEL システ	クラウド管理者

タスク	説明	必要なスキル
	ムを追加します。AWS Replication Agent をインストールし、「AWS MGN ドキュメント」の指示に従ってソースサーバーを追加します。 2. [ソースサーバー] ページで、ソース RHEL システムを選択し、[起動設定] タブを選択します。 3. [一般的な起動設定] セクションで、[編集] を選択します。自動選択を無効にしてターゲットインスタンスタイプを手動で指定するには、[インスタンスタイプ適正サイズ] を [なし] に変更し、[設定の保存] を選択します。これにより、Amazon EC2 起動テンプレートで設定したインスタンスタイプを使用できます。詳細については、「AWS MGN ドキュメント」を参照してください。 4. 「EC2 起動テンプレート」セクションで、[変更] を選択します。「EC2 起動テンプレートの変更について」ダイアログボックスで、もう一度 [変更] を選択します。Amazon EC2 コンソールが開き、このインスタン	

タスク	説明	必要なスキル
	スのテンプレートを変更できます。 5. 「AWS MGN ドキュメント」の主な考慮事項を確認してください。 **Note** 独自の AMI の選択に対する警告を無視できます。 6. 「Amazon EC2 コンソール」の新しい起動テンプレートで、以下を変更します。 • [AMI] の場合は、以前に特定した AMI ID を指定するか、RHEL-x を検索して必要なバージョン (RHEL-7.7 など) を指定します。 • [インスタンスタイプ] には、目的のターゲットインスタンスタイプを設定します。 • キーペア (ログイン)、ネットワーク設定 (ターゲットサブネットとセキュリティグループを指定する場合を除く)、ストレージ、リソースタグ (タグを追加または変更する場合を除く) の各セク	

タスク	説明	必要なスキル
	ションはそのままにしておきます。 • (オプション) AWS Systems Manager による future 管理に必要な場合は、「詳細」セクションで IAM インスタンスプロファイルのロールを指定します。 7. [テンプレートバージョンの作成] を選択し、成功メッセージ内のリンクを選択して起動テンプレートを表示します。 8. [アクション] を選択し、[デフォルトバージョンを設定] を選択します。[テンプレートバージョン] では、最新バージョン (新しいシステムの場合はバージョン 2) を選択し、[デフォルトバージョンとして設定] を選択します。 これで、AWS MGN はこのバージョンの起動テンプレートを使用してテストインスタンスまたはカットオーバーインスタンスを起動します。詳細については、「AWS MGN ドキュメント」を参照してください。	

タスク	説明	必要なスキル
設定を検証します。	1. 「AWS MGN コンソール」のソースサーバーページで、ソースサーバーを選択し、[起動設定] タブを選択します。 2. 「EC2 起動テンプレート」セクションで、インスタンスタイプ、サブネット、セキュリティグループのパラメータが正しく設定されていることを確認します。 **Note** このセクションには、選択した AMI ID は表示されません。ID を確認するには、「Amazon EC2 コンソール」を開き、起動テンプレートを表示し、このセクションに表示されているテンプレート ID を検索します。	クラウド管理者

タスク	説明	必要なスキル
新しい LI インスタンスを起動します。	1. 初期同期が完了すると、AWS MGN コンソールの [ソースサーバー] ページのサーバーの [移行ライフサイクル] 列が [テスト準備完了] に変わります。新しいテストインスタンスを起動するには、ソースサーバーを選択し、テストとカットオーバーメニューを開いて [テストインスタンスを起動] を選択します。[ジョブの詳細を表示] を選択して、起動ジョブのステータスを監視します。詳細については、「AWS MGN ドキュメント」を参照してください。 2. 起動ジョブが完了するのを待ってから、起動した EC2 インスタンスの詳細ページを開きます。[詳細] タブを選択し、[インスタンスの詳細] セクションに以下が含まれていることを確認します。 • プラットフォーム詳細:「Red Hat Enterprise Linux」 • AMI 名:EC2 起動テンプレートで指定した AMI の名前 3. 「AWS MGN ドキュメントの指示」に従って、新しい	クラウド管理者

タスク	説明	必要なスキル
	LI インスタンスに切り替えます。 4. 前回のエピックの手順に従って、AWS が提供する RHUI サーバーを使用するように新しいインスタンスを再設定します。	

LI インスタンスへの移行-オプション 2 (RHEL BYOL EC2 インスタンスの場合)

タスク	説明	必要なスキル
RHEL BYOL EC2 インスタンスを AWS LI インスタンスに移行します。	以前に BYOL として AWS に移行した RHEL システムを AWS LI インスタンスに切り替えることができます。そのためには、ディスク (Amazon Elastic Block Store ボリューム) を移動して新しい LI インスタンスにアタッチします。これらを切り替えるには、以下の手順で行います。 1. RHEL LI AMI から新しいターゲット RHEL インスタンスを起動します。選択した AMI が以下であることを確認します。 • 現在の RHEL インスタンスと同じ RHEL バージョンを使用します。 • 現在の RHEL インスタンスと同じ起動プロセス (BIOS または UEFI)	クラウド管理者

タスク	説明	必要なスキル
	があります。たとえば、ソースサーバーが BIOS ベースの場合は、BIOS ベースでもある AWS Marketplace RHEL AMI を使用します。UEFI ベースのシステムでは、UEFI ベースの AMI を選択します。 2. 新しい LI インスタンスと元のソースインスタンスの両方のインスタンスを停止します。 3. 新しい LI インスタンスからすべての EBS ボリューム (ルートディスクを含む) をデタッチして削除します。 4. すべての EBS ボリューム (ルートディスクを含む) を古いソースインスタンスからデタッチし、新しい LI インスタンスにアタッチします。デバイスへのボリュームのマッピングは同じにしてください。(たとえば、以前に /dev/sda ドライブにアタッチされていた EBS ボリュームは、新しいインスタンスに /dev/sda としてアタッチされなければなりません)。 5. ソース (現在はディスクレス) のインスタンスを削除します。	

タスク	説明	必要なスキル
	6. 新しい LI インスタンスを起動します。インスタンスにログインし、次のエピックの手順に従って AWS が提供する RHUI サーバーを使用するように再設定します。	

AWS が提供する RHUI を使用するように RHEL OS を再設定する (どちらのオプションも選択可能)

タスク	説明	必要なスキル
Red Hat のサブスクリプションとライセンスから OS の登録を解除します。	移行してカットオーバーが成功したら、Red Hat ライセンスの使用を停止し、二重請求を防ぐため、RHEL システムを Red Hat サブスクリプションから削除する必要があります。 Red Hat サブスクリプションから RHEL OS を削除するには、Red Hat サブスクリプション管理 (RHSM) ドキュメントに記載されているプロセスに従います。 CLI コマンドを使用します。 `subscription-manager unregister` また、サブスクリプションマネージャープラグインを無効にして、[yum] が呼び出されるたびにサブスクリプショ	Linux またはシステム管理者ユーザー

タスク	説明	必要なスキル
	ンの状態を確認しないようにすることもできます。これを行うには、設定ファイル /etc/yum/pluginconf.d/subscription-manager.conf を編集し、パラメーター enabled=1 を enabled=0 に変更します。	

タスク	説明	必要なスキル
古い更新設定 (RHUI、Red Hat Satellite ネットワーク、yum リポジトリ) を AWS が提供する RHUI に置き換えます。	移行した RHEL システムを、AWS が提供する RHUI サーバーを使用するように再設定する必要があります。これにより、外部の更新インフラストラクチャを必要とせずに、AWS リージョン内の RHUI サーバーにアクセスできます。この変更には以下のプロセスが含まれます。 1. 既存の yum 設定をバックアップします。 2. 古い RHUI ([yum] リポジトリ) の設定とパッケージを削除します。 3. AWS が提供する新しい RHUI 設定と証明書パッケージを追加します。これらの設定パッケージは AWS が提供する RHUI サーバーでのみ使用できるため、これらを AWS 上の別の RHEL インスタンスから取得する必要があります。 詳細な手順とコマンドは次のとおりです。 1. すべての `/etc/yum*` と `/etc/pki/*` フォルダをバックアップ場所にコピーして、既存の [yum] 設定と	Linux またはシステム管理者ユーザー

タスク	説明	必要なスキル

	証明書をバックアップします。以下に例を示します。
	`mkdir yum-backup`
	`cp -ra /etc/yum* /etc/pki ./yum-backup`
	`tar czf yum-backup.tgz ./yum-backup`
	2. 古い RHUI 設定とパッケージを削除します。
	a. インストールされているすべての RHUI パッケージを検索:
	`sudo rpm -qa | grep rhui`
	b. 以下のパッケージを削除:
	`sudo yum remove $(rpm -qa | grep rhui)`
	c. `/etc/yum/vars/releasever` ファイルが存在する場合は、それを削除します。
	3. AWS が提供する新しい RHUI と証明書パッケージを追加します。これらは AWS 上の別の RHEL インスタンスから取得する必要があります。この方法には、いくつかあります。たとえば、「Red Hat ナレッジベースの記事」に記載さ	

タスク	説明	必要なスキル

	れている指示に従うことができます。
	a. 「AWS Marketplace」から別の RHEL (RHEL-EC2) インスタンスを起動します。
	b. このインスタンスから、最新の RHUI クライアント設定パッケージと認証局 (CA) 証明書の 2 つのパッケージをダウンロードします。たとえば、以下のコマンドをデスクトップから実行します。
	`ssh RHEL-EC2 "sudo yumdownloader ca-certificates rh-amazon-rhui-client"`
	c. RHEL-EC2 インスタンスから移行した新しいシステムにパッケージをコピーします。以下に例を示します。
	`scp RHEL-EC2:rh-amazon-rhui-client\* RHEL-EC2:ca-certificates\* .`
	`ssh <migrated-instance> "mkdir /tmp/amazon"`
	`scp rh-amazon-rhui-client* ca-certificates* <migrated`	

タスク	説明	必要なスキル
	-instance>:/tmp/am azon d. 移行したインスタンスに新しい RHUI と CA の設定パッケージをインストールします。 `ssh <migrated-instance> "sudo rpm -Uhv /tmp/amazon/*"`	

設定を検証します。	移行したターゲットインスタンスで、新しい設定が正しいことを確認します。
	`sudo yum clean all`
	`sudo yum repolist`
		Linux またはシステム管理者ユーザー

関連リソース

- 「AWS Application Migration Service (AWS MGN) ユーザーガイド」
- 「IMDSv2 をサポートする AWS RHUI クライアントパッケージを入手する」(Red Hat ナレッジベース記事)
- 「Amazon EC2 起動テンプレート」(Amazon EC2 ドキュメント)

オンプレミスの SAP ASE データベースを Amazon EC2 に移行

作成者: Sergey Dmitriev (AWS) と Gergely Cserdi (AWS)

概要

このパターンは、 SAP Adaptive Server Enterprise (ASE) データベースを オンプレミスホストから Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに移行する方法を説明します。このパターンでは、AWS Database Migration Service (AWS DMS) または ASE コックピット、ASE 用の Sybase Central、移行用の DBA コックピットなどの SAP ASE ネイティブツールの使用を取り上げます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの SAP ASE ソースデータベース

制限

- ソースデータベースは 64 TB 未満であること

製品バージョン

- SAP ASE バージョン 15.x および 16.x またはそれ以降

アーキテクチャ

ソーステクノロジースタック

- オンプレミス SAP ASE データベース

ターゲットテクノロジースタック

- EC2 インスタンスの SAP ASE データベース

データベース移行アーキテクチャ

AWS DMS を使用する:

ネイティブ SAP ASE ツールの使用:

ツール

- AWS DMS - 「AWS データ移行サービス」(AWS DMS) は、複数の異なるソースデータベースとターゲットデータベースをサポートしています。詳細については、「データ移行のソース」と「データ移行のターゲット」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。
- SAP ASE — ネイティブツールには、ASE コックピット、ASE 用 Sybase Central、DBA コックピットなどがあります。

エピック

移行を分析する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。		DBA
ターゲット OS バージョンを特定します。		DBA、SysAdmin
SAP ASE 互換性リストと容量要件に基づいて、ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、SysAdmin
ストレージタイプと容量の要件を特定します。		DBA、SysAdmin

タスク	説明	必要なスキル
レイテンシーや帯域幅などのネットワーク要件を特定します。		DBA、SysAdmin
適切なインスタンスタイプ、容量、ストレージ機能、ネットワーク機能を選択します。		DBA、SysAdmin
ソースおよびターゲットのデータベースのネットワークおよびホストアクセスのセキュリティ要件を特定します。		DBA、SysAdmin
SAP ASE ソフトウェアのインストールに必要なオペレーティングシステムユーザーのリストを特定します。		DBA、SysAdmin
バックアップ戦略を決定します。		DBA
可用性要件を決定します。		DBA
アプリケーションの移行とスイッチオーバー戦略を特定します。		DBA、SysAdmin、アプリ所有者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		システム管理者

タスク	説明	必要なスキル
セキュリティグループとネットワークアクセスコントロールリスト (ACL)を作成します。		システム管理者
EC2 インスタンスを設定して起動します。		システム管理者

ソフトウェアをインストールします。

タスク	説明	必要なスキル
SAP ASE ソフトウェアの動作に必要な OS ユーザーとグループを作成します。		DBA、SysAdmin
SAP ASE ソフトウェアの必要なバージョンをダウンロードします。		DBA、SysAdmin
SAP ASE データベース、バックアップサーバーソフトウェア、レプリケーションサーバーソフトウェアを EC2 インスタンスにインストールしてから、サーバーを設定します。		DBA、SysAdmin

データを移行する — オプション 1

タスク	説明	必要なスキル
ネイティブ SAP ASE ツールまたはサードパーティツール	SAP ASE またはサードパーティツールのドキュメントを	DBA

タスク	説明	必要なスキル
を使用して、データベースオブジェクトとデータを移行します。	参照してください。これらには、ASE コックピット、ASE 用 Sybase Central、DBA コックピットが含まれます。	

データを移行する — オプション 2

タスク	説明	必要なスキル
AWS DMS を使用してデータを移行します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、SysAdmin、アプリ所有者

カットオーバー

タスク	説明	必要なスキル
アプリケーションのカットオーバーまたはスイッチオーバー戦略に従います。		DBA、SysAdmin、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、SysAdmin
プロジェクト文書を検証し、レビューします。		DBA、SysAdmin、アプリ所有者
移行の所要時間、手動とツールによるコスト削減の割合などのメトリクスを収集します。		DBA、SysAdmin、アプリ所有者
プロジェクトを閉じて、フィードバックします。		DBA、SysAdmin、アプリ所有者

関連リソース

リファレンス

- Amazon EC2
- AWS DMS
- Amazon EC2 の価格設定

チュートリアルと動画

- Amazon EC2 入門
- AWS Database Migration Service の使用開始
- AWS Data Migration Service (動画)
- Amazon EC2 のご紹介 - Elastic クラウドサーバーと AWS でのホスティング (動画)

オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する

センティル・ラマサミー (AWS) によって作成されました

概要

このパターンでは、オンプレミスの Microsoft SQL Server データベースを Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の Microsoft SQL Server に移行する方法を説明します。移行には AWS Database Migration Service 、 (AWS DMS) を使用するか、バックアップと復元などのネイティブ Microsoft SQL Server ツールを使用するか、データベースをコピーウィザード、データベースのコピーとアタッチの 2 つのオプションがあります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon EC2 がサポートするオペレーティングシステム (サポートされているオペレーティングシステムバージョンの全リストについては、Amazon EC2 のよくある質問 を参照してください)
- オンプレミスデータセンターにある Microsoft SQL Server ソースデータベース

製品バージョン

- オンプレミスおよび Amazon EC2 インスタンスデータベースの場合、 は以下 AWS DMS をサポートします。
  - SQL Server バージョン 2005、2008、2008R2、2012、2014、2016、2017、および 2019
  - Enterprise、Standard、Workgroup、Developer、および Web エディション
- サポートされているバージョンの最新リストについては、「ターゲットとしての Microsoft SQL Server データベースの使用 AWS DMS」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Microsoft SQL Server データベースのオンプレミス版

ターゲットテクノロジースタック

- Microsoft SQL Server データベースをホストする EC2 インスタンス

ターゲット アーキテクチャ

データ移行アーキテクチャ

- の使用 AWS DMS

- ネイティブ SQL Server ツールの使用

ツール

- AWS Database Migration Service （ AWS DMS ） は、Oracle、SQL Server、MySQL、PostgreSQL など、広く使用されている商用データベースやオープンソースデータベースとの間でデータを移行するのに役立ちます。 AWS DMS を使用して、オンプレミスインスタンス間 (AWS クラウド セットアップ経由 ） AWS クラウド、またはクラウドとオンプレミスのセットアップの組み合わせ間でデータを に移行できます。
- AWS Schema Conversion Tool （ AWS SCT ） は、ソースデータベーススキーマとカスタムコードの大部分をターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベースの移行をサポートします。
- ネイティブの Microsoft SQL Server ツールには、バックアップと復元、データベースのコピーウィザード、データベースのコピーとアタッチが含まれます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。		DBA
対象オペレーティングシステムのバージョンを特定します。		DBA、システム管理者
Microsoft SQL Server 互換性リストと容量要件に基づいて、ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、システム管理者
ストレージタイプと容量要件を特定します。		DBA、システム管理者
レイテンシーや帯域幅などのネットワーク要件を特定します。		DBA、システム管理者
容量、ストレージ機能、ネットワーク機能に基づいてEC2インスタンスタイプを選択してください。		DBA、システム管理者
ソースおよびターゲットのデータベースのネットワークおよびホストアクセスのセキュリティ要件を特定します。		DBA、システム管理者

タスク	説明	必要なスキル
Microsoft SQL Server ソフトウェアのインストールに必要なユーザーのリストを特定してください。		DBA、システム管理者
バックアップ戦略を決定します。		DBA
可用性要件を決定します。		DBA
アプリケーションの移行とカットオーバー戦略を特定する。		DBA、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。		システム管理者
セキュリティグループとネットワークアクセスコントロールリスト (ACL)		システム管理者
EC2 インスタンスを設定して起動します。		システム管理者

ソフトウェアをインストールします。

タスク	説明	必要なスキル
Microsoft SQL Server ソフトウェアの動作に必要なユー		DBA、システム管理者

タスク	説明	必要なスキル
ザーとグループを作成します。		
Microsoft SQL Server ソフトウェアをダウンロードします。		DBA、システム管理者
Microsoft SQL Server ソフトウェアを EC2 インスタンスにインストールし、サーバーを設定します。		DBA、システム管理者

データを移行する — オプション 1

タスク	説明	必要なスキル
ネイティブ SQL Server ツールまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。	Microsoft SQL Serverのネイティブツールには、データベースコピーウィザード、データベースのコピーとアタッチが含まれます。詳細については、「Microsoft SQL Server データベースの への移行 AWS クラウド」ガイドを参照してください。	DBA

データを移行する — オプション 2

タスク	説明	必要なスキル
AWS DMS を使用してデータを移行します。	の使用の詳細については AWS DMS、関連リソースセクションのリンクを参照してください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。	AWS Schema Conversion Tool （ AWS SCT) を使用して、アプリケーションのソースコードに埋め込まれた SQL コードを分析および変更します。	DBA、アプリ所有者

カットオーバー

タスク	説明	必要なスキル
アプリケーションのスイッチオーバー戦略に従ってください。		DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
すべての一時 AWS リソースをシャットダウンします。	一時的なリソースには、 AWS DMS レプリケーションインスタンスと EC2 インスタンスが含まれます AWS SCT。	DBA、システム管理者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者、システム管理者
移行の所要時間、手動とツールによるコスト削減の割合などのメトリクスを収集します。		DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
プロジェクトを閉じて、フィードバックを提供します。		DBA、アプリ所有者、システム管理者

関連リソース

リファレンス

- Microsoft SQL Server データベースの への移行 AWS クラウド
- Amazon EC2
- Amazon EC2 よくある質問
- Amazon EC2 の料金
- AWS Database Migration Service
- での Microsoft 製品 AWS
- での Microsoft ライセンス AWS
- での Microsoft SQL Server AWS

チュートリアルと動画

- 「Amazon EC2 の開始方法」
- https://aws.amazon.com/dms/getting-started/ の開始方法AWS Database Migration Service
- Amazon EC2 インスタンスを Simple AD Active Directory に結合する
- Amazon EC2 インスタンスを AWS Managed Microsoft AD Active Directory に結合する
- AWS Database Migration Service (ビデオ)
- Amazon EC2 の概要 – を使用した Elastic Cloud Server とホスティング AWS (ビデオ)

オンプレミス MySQL データベースを Amazon EC2 に移行する

作成者: Lorenzo Mota (AWS)

概要

このパターンは、オンプレミス MySQL データベースを Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上の MySQL データベースに移行するガイダンスを提供します。このパターンでは、移行に AWS Database Migration Service （ AWS DMS) または mysqldump などのネイティブ MySQL ツールを使用する方法について説明します。MySQL DB インスタンスへの完全なデータベース移行に焦点を当てています。

このパターンは主に DBAs とソリューションアーキテクトを対象としています。これは、小規模または大規模なプロジェクト、テストフェーズ、または最終移行フェーズで使用できます。このパターンを本番環境で使用する前に、少なくとも 1 つのテストサイクルを実行することをお勧めします。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- オンプレミスデータセンターの MySQL ソースデータベース

製品バージョン

- MySQL バージョン 5.5 以降
- Amazon EC2 でサポートされているターゲットオペレーティングシステム。Amazon EC2 に関するFAQs」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス MySQL データベース

ターゲットテクノロジースタック

- Amazon EC2 の MySQL データベースインスタンス

AWS データ移行方法

- AWS DMS
- mysqldump などのネイティブ MySQL ツール、または Percona XtraBackup などのサードパーティーツール

ターゲット アーキテクチャ

次の図は、カットオーバー後のターゲット Amazon EC2 実装を示しています。

AWS データ移行アーキテクチャ

AWS DMS を使用する:

次の図は、カットオーバーまでターゲット MySQL データベースに完全および増分の変更を送信 AWS DMS するための に基づくデータ移行ワークフローを示しています。オンプレミスから へのネットワーク接続は、SQL クライアントの要件 AWS に依存し、このパターンの範囲外です。

他の MySQL ツールの使用：

次の図は、MySQL ツールを使用してオンプレミスデータベースからエクスポートダンプファイルを生成することに基づくデータ移行ワークフローを示しています。これらのファイルは Amazon Simple Storage Service (Amazon S3) に移動され、カットオーバー前にターゲット MySQL データベースにインポートされます。オンプレミスから へのネットワーク接続は、SQL クライアントの要件 AWS に依存し、このパターンの範囲外です。

注記:

- ダウンタイムに関する考慮事項と最終カットオーバーのデータベースのサイズに応じて、 AWS DMS または別の変更データキャプチャ (CDC) ツールを使用してカットオーバー時間を最小限に抑えることができます。などの CDC ツールを使用すると AWS DMS、数分でターゲットデータベースに移行できます。
- mysqldump を使用したオフライン戦略では、データベースのサイズとネットワークレイテンシーが短いカットオーバー移行ウィンドウを許容すれば十分です。（おおよその時間を取得するためにテストを実行することをお勧めします）。

- 通常、 経由の CDC 戦略では、オフラインオプションよりも多くのモニタリングと複雑さ AWS DMS が必要です。

ツール

AWS サービス

- AWS Database Migration Service （ AWS DMS ） は、複数のソースデータベースとターゲットデータベースをサポートしています。でサポートされている MySQL ソースデータベースとターゲットデータベースの詳細については AWS DMS、「 のソースとして MySQL 互換データベース AWS DMSを使用する」および「 のターゲットとして MySQL 互換データベースを使用する AWS DMS」を参照してください。ソースデータベースが でサポートされていない場合は AWS DMS、別の方法を選択してデータを移行する必要があります。

その他のツール

- mysqldump MySQL は、バックアップまたは移行の目的で MySQL データベースからダンプファイルを作成する MySQL ユーティリティです。
- Percona XtraBackup は、MySQL データベースでノンブロッキングバックアップを実行するためのオープンソースユーティリティです。

エピック

移行を計画する

タスク	説明	必要なスキル
データベースのバージョンを検証します。	ソースデータベースとターゲットデータベースのバージョンを検証します。でサポートされている MySQL バージョンの詳細については AWS DMS、 AWS DMS ドキュメントの「 のソース AWS DMS」と「 のターゲット AWS DMS」を参照してください。	DBA

タスク	説明	必要なスキル
ターゲットオペレーティングシステムを特定します。	ターゲットオペレーティングシステムのバージョンを決定します。Amazon EC2 でサポートされているターゲットオペレーティングシステムのリストについては、Amazon EC2 のFAQs」を参照してください。	DBA、システム管理者
ハードウェア要件を特定します。	MySQL 互換性リストと容量要件に基づいて、ターゲットサーバーインスタンスのハードウェア要件を決定します。	DBA、システム管理者
ストレージ要件を特定します。	ターゲットデータベースのストレージタイプと容量を決定します。	DBA、システム管理者
ネットワーク要件を特定します。	レイテンシーや帯域幅などのネットワーク要件を決定します。	DBA、システム管理者
ターゲットインスタンスタイプを選択します。	容量、ストレージ機能、ネットワーク機能に基づいて、ターゲットインスタンスタイプを選択します。	DBA、システム管理者
セキュリティ要件を特定します。	ソースデータベースとターゲットデータベースのネットワークまたはホストアクセスのセキュリティ要件を決定します。	DBA、システム管理者

タスク	説明	必要なスキル
ユーザーを特定します。	MySQL ソフトウェアインストール用のオペレーティングシステムユーザーのリストを決定します。詳細については、MySQL ドキュメントを参照してください。	DBA、システム管理者
バックアップ戦略を決定します。		DBA
可用性の要件を決定します。		DBA
アプリケーションの移行またはスイッチオーバー戦略を特定します。		DBA、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) とサブネットを作成する。	ルートテーブル、インターネットゲートウェイ、NAT ゲートウェイ、サブネットを設定します。詳細については、Amazon VPC ドキュメントの「VPC 設定オプション」を参照してください。	システム管理者
セキュリティグループとネットワークアクセスコントロールリスト (ACL) を作成します。	ポート (MySQL のデフォルトは 3306) と CIDR 範囲、または要件に応じて特定の IPsを設定します。	システム管理者
EC2 インスタンスを設定して起動します。	手順については、Amazon EC2 ドキュメントの「EC2	システム管理者

タスク	説明	必要なスキル
	インスタンスを起動する」を参照してください。 Amazon EC2	

MySQL ソフトウェアをインストールする

タスク	説明	必要なスキル
ユーザーとグループを作成します。	サーバーとデータベースにアクセスする必要があるオペレーティングシステムのユーザーとグループを作成します。詳細は、MySQL ドキュメントの「アクセスコントロールおよびアカウントマネジメント」を参照してください。	DBA、システム管理者
MySQL をダウンロードします。	MySQL ソフトウェアをダウンロードします。手順とバイナリについては、MySQL ドキュメントの「MySQL のインストール」を参照してください。	DBA、システム管理者
EC2 インスタンスに MySQL をインストールし、サーバーを設定します。	EC2 インスタンスに接続し、MySQL ソフトウェアをインストールします。詳細については、Amazon EC2 ドキュメントの「EC2 インスタンスに接続する」を参照してください。 Amazon EC2	DBA、システム管理者

データの移行 – オプション 1

タスク	説明	必要なスキル
ネイティブ MySQL またはサードパーティーのツールを使用してデータを移行します。	このオプションは、ネイティブ MySQL ツールまたはサードパーティーツールを使用してデータベースオブジェクトとデータを移行します。手順については、mysqldump または Percona XtraBackup (物理的な移行用) のドキュメントを参照してください。これらのツールの使用の詳細については、 AWS ブログ記事MySQL から Amazon RDS for MySQL または Amazon Aurora MySQL への移行オプション」を参照してください。	DBA

データの移行 – オプション 2

タスク	説明	必要なスキル
でデータを移行します AWS DMS。	詳細については、 AWS DMS ドキュメントの「 の高レベルビュー AWS DMS」を参照してください。	DBA

カットオーバーの準備

タスク	説明	必要なスキル
オブジェクト数を収集します。	ソースデータベースと新しいターゲットデータベースか	DBA

タスク	説明	必要なスキル
	らオブジェクト数を収集します。ターゲットデータベースの不一致を修正します。	
依存関係を確認します。	他のデータベースとの間の依存関係 (リンク) がまだ有効であり、正しく動作することを確認します。	DBA
テスト。	これがテストサイクルの場合は、クエリテストを実行し、メトリクスを収集して、問題を修正します。	DBA

カットオーバー

タスク	説明	必要なスキル
クライアントを移動します。	アプリケーションクライアントを新しいインフラストラクチャに切り替えます。	DBA、アプリ所有者、システム管理者
サポートを提供します。	機能アプリケーションテスト中にサポートを提供します。	DBA

プロジェクトを閉じる

タスク	説明	必要なスキル
リソースをシャットダウンします。	AWS DMS レプリケーションインスタンスおよびその他の一時 AWS リソースをシャットダウンします。	DBA、システム管理者

タスク	説明	必要なスキル
ドキュメントを確認してプロジェクトします。	プロジェクト文書を確認して検証する。	DBA、アプリ所有者、システム管理者
メトリクスを収集します。	移行時間、ツール支援の変更に対する手動変更の割合、コスト削減などのメトリクスを収集します。	DBA、アプリ所有者、システム管理者
プロジェクトを終了します。	移行プロジェクトを終了し、フィードバックを提供します。	DBA、アプリ所有者、システム管理者
ソースデータベースを廃止します。	オンプレミス MySQL データベースを廃止します。	DBA、システム管理者

関連リソース

リファレンス

- 「Amazon EC2 ドキュメント」
- AWS DMS ドキュメント
- 「Amazon EC2 の料金」
- AWS DMS Step-by-Stepのチュートリアル
- mysqldump
- Percona XtraBackup

チュートリアルと動画

- の開始方法 AWS DMS
- Amazon EC2 の概要 – を使用した Elastic Cloud Server とホスティング AWS (ビデオ)

アプリケーション移行サービスを使用することで、同種の SAP 移行のカットオーバー時間を短縮する

作成者: Pavel Rubin (AWS)、Diego Valverde (AWS)、Sunil Yadav (AWS)

概要

このパターンは、AWS アプリケーション移行サービスを使用して SAP ワークロードを移行する手順の概要を示します。アプリケーション移行サービスは、ブロックレベルのレプリケーションを使用してソースと継続的に同期されるレプリケーションボリュームを維持し、カットオーバーを容易にします。

SAP ワークロードには、SAP Customer Relationship Management (SAP CRM)、SAP Enterprise Resource Planning (ERP)、SAP Business Warehouse (SAP BW) のアプリケーションが含まれます。

前提条件と制限

前提条件

- ソース SAP サーバーと AWS 上の送信先の仮想プライベートクラウド (VPC) 間の安定したネットワーク接続を備えたアクティブな AWS アカウント
- オンプレミスデータセンターにある Linux または Windows 用の SAP Adaptive Server Enterprise (ASE) ソースデータベース

制限

- ターゲットオペレーティングシステムは、Amazon Elastic Compute Cloud (Amazon EC2) でサポートされている必要があります。詳細については、Amazon SNS のよくある質問を参照してください。

アーキテクチャ

ソーステクノロジースタック

- SAP ASE データベース

ターゲットテクノロジースタック

- Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、オンプレミスサーバーからレプリケーションエージェントを経由してアプリケーション移行サービスのエンドポイントへの移行を示しています。Amazon Simple Storage Service (Amazon S3)のエンドポイントを使用して、インストールファイルと設定ファイルにアクセスします。ステージングエリアのサブネットと移行されたリソースには EC2 インスタンスが含まれ、EBS ボリュームにデータストレージが格納されます。ポート TCP 443 は、ソースマシンネットワークをアプリケーション移行サービスに接続し、ステージングエリアのサブネットをアプリケーション移行サービス、Amazon EC2、および Amazon S3 リージョナルエンドポイントに接続するために使用されます。ポート TCP 1500 は、ローカルネットワークとステージングエリア間のデータ複製に使用されます。

ツール

- AWS Application Migration Service を使用すると、変更を加えることなく、最小限のダウンタイムで、アプリケーションを AWS クラウドにリホスト (リフトアンドシフト) できます。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。仮想サーバーを必要な数だけ起動して、迅速にスケールアップまたはスケールダウンできます。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、任意のデータ量を保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Security Token Service (AWS STS)を使用すると、ユーザーに一時的、限定された特権認証情報をリクエストする上で役立ちます。

エピック

AWS Application Migration Service

タスク	説明	必要なスキル
アプリケーション移行サービスを初期化します。	SAP ASE データベースをデプロイする AWS リージョンで、アプリケーション移行サービスを初期化します。AWS では、各リージョンのアプリケーション移行サービスページに初めてナビゲートしたときに、自動セットアップが提供されます。	AWS 管理者
サービスロールを手動で作成します。	(オプション) 自動化 (AWS Control Tower など) を使用してアカウントを設定する場合は、インストール、レプリケーション、起動に必要な 6 つの AWS ID およびアクセス管理 (IAM) ロールを手動で作成できます。手順については、AWS ドキュメントを参照してください。	AWS 管理者
レプリケーション設定テンプレートを作成します。	レプリケーション設定テンプレートは、サブネット、インスタンスタイプ、Amazon EBS 暗号化、およびデータのルーティング方法を定義します。詳細な設定情報については、AWS ドキュメントを参照してください。	AWS 全般

エージェントインストール用の認証情報を生成する

タスク	説明	必要なスキル
新しい IAM ロールを作成します。	IAM コンソールで、[Roles (ロール)] にナビゲートし、[Create Role (ロールの作成)] を選択します。 信頼されたエンティティタイプの場合は、[AWS account (AWS アカウント)] を選択してから、[Next (次へ)] を選択します。	AWS システム管理者
AWSApplicationMigrationAgentPolicy を IAM ロールにアタッチします。	AWS マネージド`AWSApplicationMigrationAgentPolicy` ポリシーには、アプリケーション移行サービスエージェントのインストールを実行するために必要な権限が含まれています。 ポリシーをアタッチしたら、[Next (次へ)] を選択します。	AWS システム管理者
ロールの作成を完了します。	わかりやすい名前を付け、[Create role (ロールを作成)] を選択します。	AWS システム管理者
一時認証情報を生成します。	アクセスキー ID、シークレットアクセスキー、セッショントークンを生成するには、AWS STS ドキュメントの指示に従います。これらの認証情報はエージェントを	AWS システム管理者

タスク	説明	必要なスキル
	インストール中に使用されます。	

SAP ソースマシンにアプリケーション移行サービスエージェントをインストールします。

タスク	説明	必要なスキル
SAP ソースマシンにエージェントインストーラーをダウンロードします。	ソースオペレーティングシステム (「Windows」または「Linux」) に適したエージェントインストーラーをダウンロードします。	アプリ所有者
AWS レプリケーションエージェントをインストールします。	ソースマシンでエージェントインストーラーファイルを実行すると、まず、アクセスキー、シークレットアクセスキー、セッショントークン、レプリケーション先のリージョンの入力を求められます。以前に作成した IAM ロールの一時認証情報と、初期化中に設定したものと同じリージョンを使用します。	アプリ所有者
最初のデータレプリケーションを待機します。	エージェントをインストールすると、ソースマシンがアプリケーション移行サービスコンソールの [Machines (マシン)] タブに表示されます。	アプリ所有者

ターゲットマシンの起動テンプレートを設定します。

タスク	説明	必要なスキル
ソースサーバーの起動テンプレートを更新します。	各ソースサーバーは、ターゲット EC2 サーバーの設定を通知する一意の EC2 起動テンプレートを使用します。移行したサーバーの Amazon EC2 設定をカスタマイズする場合は、このテンプレートを編集できます。	AWS 全般
デフォルトの起動テンプレートのバージョンを設定します。	起動テンプレートに必要な変更をしたら、この更新バージョンをデフォルトの起動 テンプレートとして使用することを指定します。詳細については、AWS ドキュメントを参照してください。	AWS 全般
インスタンスタイプの適切なサイジングをオフにします。	(オプション)インスタンスタイプの適切なサイジングにより、ソース SAP サーバーの構成に基づき、インスタンスタイプの自動推奨が提供されます。起動テンプレートでカスタマイズしたインスタンスタイプを指定できるように、この設定をオフにすることをお勧めします。	AWS 全般

テストを実行

タスク	説明	必要なスキル
テスト起動を開始します。	アプリケーション移行サービスコンソールで 1 台以上のサーバーを選択してから、テストとカットオーバーで [Launch test instances (テストインスタンスを起動)] を選択します。	AWS 全般、移行エンジニア、移行リード
変換と起動プロセスが完了するまで待機します。	[Launch history(起動履歴)] タブで起動プロセスを確認できます。マシンが EC2 インスタンスとして正常に起動すると、[Alerts (アラート)] タブは [Launched (起動済み)] に更新されます。	
テストが正常に完了したことを確認します。	起動したインスタンスにリモートデスクトッププロトコル (RDP) または SSH (セキュアなシェル) で接続し、適切なアプリケーションチェックを実行します。たとえば、SAP インターフェースにログインして機能を検証します。	移行エンジニア、アプリ所有者
ソースライフサイクルを更新します。	テストが成功したら、[Test and Cutover (テストとカットオーバー)] タブで、[Mark as “Ready for cutover” (「カットオーバー準備完了」としてマーク)] に更新します。	移行エンジニア、移行リード

Amazon EC2 ターゲットへのカットオーバーをスケジュールして実行する

タスク	説明	必要なスキル
カットオーバーウィンドウのスケジュールを設定する。		カットオーバーリーダー、移行リーダー、アプリ所有者
カットオーバー起動を開始します。	1 つ以上のサーバーを選択します。[Test and Cutover (テストとカットオーバー)] タブで、アプリケーション移行サービスコンソールのテストとカットオーバーにある [Launch cutover instances (カットオーバーインスタンスを起動)] を選択します。	移行エンジニア
変換と起動プロセスが完了するまで待機します。	[Launch history(起動履歴)] タブで起動プロセスを確認できます。マシンが EC2 インスタンスとして正常に起動すると、[Alerts (アラート)] タブは [Launched (起動済み)] に更新されます。	
カットオーバーが正常に完了したことを確認します。	RDP または SSH で起動したインスタンスに接続し、適切なアプリケーションチェックを実行します。	アプリ所有者、移行エンジニア
ソースライフサイクルを更新します。	カットオーバーが成功したら、[Test and Cutover (テストとカットオーバー)] タブの [Finalize cutover (カットオーバーを終了)] を選択して、ソースマシンのライフサイクルを更新します。	移行エンジニア

関連リソース

リファレンス

- AWS Application Migration Service
- 「AWS アプリケーション移行のよくある質問」

動画

- AWS Application Migration Service

AWS Cloud でオンプレミスワークロードをリホストする: 移行チェックリスト

作成者：Srikanth Rangavajhala (AWS)

概要

Amazon Web Services (AWS) クラウドでのオンプレミスワークロードのリホストには、計画、事前検出、検出、構築、テスト、カットオーバーという移行フェーズが含まれます。このパターンは、フェーズとその関連タスクの概要を示しています。タスクは高レベルで説明され、すべてのアプリケーションワークロードの約 75% をサポートしています。これらのタスクは、アジャイルスプリントサイクルで 2～3 週間かけて実装できます。

これらのタスクは、移行チームとコンサルタントと確認して検証する必要があります。確認後は、入力を収集し、要件を満たすための必要に応じてタスクを削除または再評価し、ポートフォリオ内のアプリケーションワークロードの少なくとも 75% をサポートするように他のタスクを変更できます。その後、Atlassian Jira または Rally Software などのアジャイルプロジェクト管理ツールを使用して、タスクをインポートし、リソースに割り当て、移行アクティビティを追跡できます。

このパターンでは、AWS クラウド移行ファクトリーを使用してワークロードをリホストすることを前提としていますが、任意の移行ツールを使用できます。

Amazon Macie は、Amazon Simple Storage Service (Amazon S3) バケットにデータソース、モデル呼び出しログ、プロンプトストアとして保存されているナレッジベース内の機密データを識別するのに役立ちます。詳細については、Macie ドキュメントを参照してください。

前提条件と制限

前提条件

- 移行タスクを追跡するプロジェクト管理ツール (例、Atlassian Jira または Rally ソフトウェア)
- AWS でワークロードをリホストする移行ツール (例、クラウド移行ファクトリ)

アーキテクチャ

ソースプラットフォーム

- オンプレミスソーススタック (テクノロジー、アプリケーション、データベース、インフラストラクチャなど)

ターゲットプラットフォーム

- AWS クラウドターゲットスタック (テクノロジー、アプリケーション、データベース、インフラストラクチャなど)

アーキテクチャ

次の図は、クラウド移行ファクトリーと AWS Application Migration Service を使用したリホスティング (オンプレミスのソース環境から AWS へのサーバーの検出と移行) を示しています。

ツール

- 任意の移行とプロジェクト管理ツールを使用できます。

エピック

計画フェーズ

タスク	説明	必要なスキル
事前検出のバックログを整理します。	部門長やアプリケーション所有者とともに事前検出のバックログ整理作業セッションを実施します。	プロジェクトマネージャー、アジャイルスクラムリーダー
スプリント計画作業セッションを実施します。	範囲確定演習として、移行するアプリケーションをスプリントとウェーブに分散します。	プロジェクトマネージャー、アジャイルスクラムリーダー

事前検出フェーズ

タスク	説明	必要なスキル
アプリケーションに関する知識を確認します。	アプリケーション所有者とそのアプリケーションに関する知識を確認し、文書化します。技術的な質問に対応する	移行スペシャリスト (インタビュアー)

タスク	説明	必要なスキル
	担当者が他にいるかどうかを確認します。	
アプリケーションのコンプライアンス要件を決定します。	アプリケーションが Payment Card Industry Data Security Standard (PCI DSS)、Sarbanes-Oxley Act (SOX)、個人を特定できる情報 (PII)、またはその他の基準の要件に準拠する必要がないことをアプリケーション所有者に確認します。コンプライアンス要件が存在する場合、チームは移行するサーバーでコンプライアンスチェックを終了する必要があります。	移行スペシャリスト (インタビュアー)
製品リリース要件を確認します。	移行したアプリケーションを本番環境にリリースするための要件 (リリース日やダウンタイム期間など) を、アプリケーション所有者または技術担当者に確認します。	移行スペシャリスト (インタビュアー)
サーバーリストを取得します。	ターゲットのアプリケーションに関連付けられているサーバーのリストを取得します。	移行スペシャリスト (インタビュアー)
現在の状態を示す論理ダイアグラムを取得します。	アプリケーションの現在の状態図は、エンタープライズアーキテクトまたはアプリケーション所有者から入手します。	移行スペシャリスト (インタビュアー)

タスク	説明	必要なスキル
ターゲットの状態を示す論理ダイアグラムを作成します。	AWS 上のターゲットアーキテクチャを示すアプリケーションの論理ダイアグラムを作成します。このダイアグラムは、サーバー、接続性、マッピング要素を示す必要がありますす。	エンタープライズアーキテクト、ビジネスオーナー
サーバー情報を取得します	設定詳細など、アプリケーションに関連付けられたサーバーに関する情報を収集します。	移行スペシャリスト (インタビュアー)
検出テンプレートにサーバー情報を追加します。	アプリケーション検出テンプレートに詳細なサーバー情報を追加します (このパターンについては添付の `mobilize-application-questionnaire.xlsx` を参照)。このテンプレートには、アプリケーション関連のセキュリティ、インフラストラクチャ、オペレーティングシステム、ネットワークに関する詳細がすべて含まれます。	移行スペシャリスト (インタビュアー)
アプリケーション検出テンプレートを公開します。	アプリケーション検出テンプレートをアプリケーション所有者および移行チームと共有して、共通のアクセスと使用を可能にします。	移行スペシャリスト (インタビュアー)

発見フェーズ

タスク	説明	必要なスキル
サーバーリストを確認します。	サーバーのリストと各サーバーの目的を、アプリケーション所有者またはテクニカルリードに確認します。	移行スペシャリスト
サーバーグループを特定して追加します。	Web サーバーまたはアプリケーションサーバーなどのサーバーグループを特定し、この情報をアプリケーション検出テンプレートに追加します。各サーバーが属するアプリケーションの階層 (Web、アプリケーション、データベース) を選択します。	移行スペシャリスト
アプリケーション検出テンプレートに記入します。	移行チーム、アプリケーションチーム、AWS の協力を得て、アプリケーション検出テンプレートの詳細を完了します。	移行スペシャリスト
足りないサーバーの詳細 (ミドルウェアチームと OS チーム) を追加します。	ミドルウェアチームとオペレーティングシステム (OS) チームに、アプリケーション検出テンプレートを確認し、不足しているサーバー情報 (データベース情報など) を追加するよう依頼します。	移行スペシャリスト
インバウンド/アウトバウンドのトラフィックルールを取得します (ネットワークチーム)。	ネットワークチームに、送信元サーバーと送信先サーバーのインバウンド/アウトバウンドのトラフィックルールを取得するよう依頼します。ま	移行スペシャリスト

タスク	説明	必要なスキル
	た、ネットワークチームは既存のファイアウォールルールを追加し、セキュリティグループ形式にエクスポートし、既存のロードバランサーをアプリケーション検出テンプレートに追加する必要があります。	
必要なタグ付けを特定します。	アプリケーションのタグ付け要件を決定します。	移行スペシャリスト
ファイアウォールリクエストの詳細を作成します。	アプリケーションとの通信に必要なファイアウォールルールをキャプチャしてフィルタリングします。	移行スペシャリスト、ソリューションアーキテクト、ネットワークリード
EC2 インスタンスタイプを更新します。	インフラストラクチャとサーバーの要件に基づいて、ターゲット環境で使用する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスタイプを更新します。	移行スペシャリスト、ソリューションアーキテクト、ネットワークリード
現在の状態ダイアログを特定します。	アプリケーションの現在の状態を示すダイアログを特定または作成します。このダイアログは情報セキュリティ (InfoSec) リクエストで使用されます。	移行スペシャリスト、ソリューションアーキテクト
将来の状態ダイアログを完成させます。	アプリケーションの将来の (ターゲット) 状態を示すダイアログを完成させます。このダイアログも InfoSec リクエストで使用されます。	移行スペシャリスト、ソリューションアーキテクト

タスク	説明	必要なスキル
ファイアウォールまたはセキュリティグループのサービスリクエストを作成します。	ファイアウォールまたはセキュリティグループのサービスリクエスト (開発/QA、実稼働前、実稼働用) を作成します。クラウド移行ファクトリーを使用中の場合は、レプリケーション固有のポート (まだ開いていない場合) を含めます。	移行スペシャリスト、ソリューションアーキテクト、ネットワークリード
ファイアウォールまたはセキュリティグループのリクエストを確認します (InfoSec チーム)。	この手順では、InfoSecチームが前の手順で作成したファイアウォールまたはセキュリティグループのリクエストを確認して承認します。	InfoSec エンジニア、移行スペシャリスト
ファイアウォールセキュリティグループのリクエストを実装します (ネットワークチーム)。	InfoSec チームがファイアウォールリクエストを承認すると、ネットワークチームは必要なインバウンド/アウトバウンドのファイアウォールルールを実装します。	移行スペシャリスト、ソリューションアーキテクト、ネットワークリード

ビルドフェーズ (開発/QA、実稼働前環境、本番環境の場合は繰り返し)

タスク	説明	必要なスキル
アプリケーションとサーバーのデータをインポートします。	1. 対象範囲内のソースサーバーのローカル管理者権限があるドメインユーザーとして移行実行サーバーにログインしていることを確認します。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
	2. 移行インテークフォームを使用して、対象範囲内のソースサーバーの属性をインポートします。追加情報については、クラウド移行ファクトリー実装ガイドを参照してください。 クラウド移行ファクトリーを使用していない場合は、移行ツールの設定手順に従います。	
ソースサーバーの前提条件を確認します。	対象範囲内のソースサーバーと接続し、TCP 1500、TCP 443、ルートボリュームの空き容量、.Net Framework バージョン、その他のパラメータなど、前提条件を確認します。これらはレプリケーションに必要です。追加情報については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者
レプリケーションエージェントをインストールするサービスリクエストを作成します。	開発/QA、実稼働前、または実稼働用の範囲内対象サーバーにレプリケーションエージェントをインストールするサービスリクエストを作成します。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
レプリケーションエージェントをインストールします。	開発/QA、実稼働前、または実稼働マシンの範囲内対象ソースサーバーにレプリケーションエージェントをインストールします。追加情報については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者
起動後スクリプトをプッシュします。	AWS Application Migration Service は起動後のスクリプトをサポートしており、ターゲットインスタンス起動後のソフトウェアのインストール/アンインストールなど、OS レベルのアクティビティの自動化に役立ちます。この手順は、移行対象として特定されたサーバーに応じて、起動後のスクリプトを Windows マシンまたは Linux マシンにプッシュします。手順については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
レプリケーションステータスを確認します。	提供されたスクリプトを使用して、対象範囲内のソースサーバーのレプリケーションステータスを自動的に確認します。このスクリプトは、指定したウェーブ内のすべてのソースサーバーのステータスが健全ステータスに変わるまで 5 分ごとに繰り返します。手順については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者
管理者ユーザーを作成します。	対象範囲内のソースサーバーから AWS への移行後の問題のトラブルシューティングには、ソースマシンのローカル管理者または sudo ユーザーが必要になる場合があります。認証サーバー (DC または LDAP サーバーなど) にアクセスできない場合は、移行チームはこのユーザーを使用してターゲットサーバーにログインします。この手順の指示については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
起動テンプレートを検証します。	サーバーのメタデータを検証して、正常に動作し、無効なデータがないことを確認します。この手順はテストメタデータとカットオーバーメタデータの両方を検証します。手順については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

テストフェーズ (開発/QA、実稼働前環境、本番環境の場合は繰り返し)

タスク	説明	必要なスキル
サービスリクエストを作成します。	インフラストラクチャチームと他のチームが、開発/QA、実稼働前、または実稼働インスタンスへのアプリケーションのカットオーバーを実行するサービスリクエストを作成します。	移行スペシャリスト、クラウド管理者
ロードバランサーを設定します (オプション)。	Application Load Balancer または iRules を搭載した F5 ロードバランサーなど、必要なロードバランサーを設定します。	移行スペシャリスト、クラウド管理者
テスト用のインスタンスを起動します。	テストモードの AWS Application Migration Service で、指定したウェーブのすべてのターゲットマシンを起動します。追加情報については、クラウ	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
	ド移行ファクトリー実装ガイドを参照してください。	
ターゲットインスタンスのステータスを確認します。	同じウェーブ内のすべての対象範囲内のソースサーバーの起動プロセスをチェックして、ターゲットインスタンスのステータスを確認します。ターゲットインスタンスが起動するまでに最大 30 分かかることがあります。Amazon EC2 コンソールにログインし、ソースサーバー名を検索し、[Status check (ステータスチェック)] 列を確認することで、ステータスを手動で確認できます。ステータス 2/2 チェック合格は、インフラストラクチャの観点からインスタンスが健全であることを示します。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
DNS エントリを変更します。	ドメインネームシステム (DNS) エントリを変更します。(Microsoft Windows 環境では `resolv.conf` または `host.conf` を使用) 各 EC2 インスタンスが、このホストの新しい IP アドレスを指すように設定します。 **Note** オンプレミスサーバーと AWS クラウドサーバーの間に DNS 競合がないことを確認します。サーバーがホストされている環境に応じて、この手順と以下の手順はオプションです。	移行スペシャリスト、クラウド管理者
EC2 インスタンスからバックエンドホストへの接続性をテストします。	移行したサーバーのドメイン認証情報を使用してログインを確認します。	移行スペシャリスト、クラウド管理者
DNS A レコードを更新します。	各ホストの DNS A レコードが新しい Amazon EC2 プライベート IP アドレスを指すように更新します。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
DNS CNAME レコードを更新します。	仮想 IP (ロードバランサー名) の DNS CNAME レコードが、Web サーバーとアプリケーションサーバーのクラスターを指すように更新します。	移行スペシャリスト、クラウド管理者
該当する環境でアプリケーションをテストします。	新しい EC2 インスタンスにログインし、開発/QA、実稼働前環境、本番環境でアプリケーションをテストします。	移行スペシャリスト、クラウド管理者
カットオーバーの準備完了とマークします。	テストが完了したら、ソースサーバーのステータスがカットオーバーの準備完了を示し、ユーザーがカットオーバーインスタンスを起動できるように変更します。手順については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

カットオーバーフェーズ

タスク	説明	必要なスキル
実稼働環境デプロイ計画を作成します。	実稼働環境デプロイ計画 (バックアウト計画を含む) を作成します。	移行スペシャリスト、クラウド管理者
オペレーションチームにダウンタイムを通知します。	オペレーションチームにサーバーのダウンタイムスケジュールを通知します。チームによっては、この通知の変更リクエストまたはサービス	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
	リクエスト (CR/SR) チケットが必要な場合があります。	
実稼働マシンを複製します。	アプリケーション移行サービスまたは別の移行ツールを使用して、実稼働マシンを複製します。	移行スペシャリスト、クラウド管理者
対象範囲内のソースサーバーをシャットダウンします。	ソースサーバーのレプリケーションステータスを確認したら、ソースサーバーをシャットダウンして、クライアントアプリケーションからサーバーへのトランザクションを停止できます。ソースサーバーはカットオーバーウィンドウでシャットダウンできます。詳細については、クラウド移行ファクトリー実装ガイドを参照してください。	クラウド管理者
カットオーバー用のインスタンスを起動します。	カットオーバーモードの Application Migration Service で、指定したウェーブのすべてのターゲットマシンを起動します。詳細については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

タスク	説明	必要なスキル
ターゲットインスタンス IP を取得します。	ターゲットインスタンスの IP を取得します。DNS 更新が環境で手動処理の場合、すべてのターゲットインスタンスの新しい IP アドレスを取得する必要があります。詳細については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者
ターゲットサーバーの接続を確認します。	DNS レコードを更新したら、ホスト名でターゲットインスタンスに接続して接続を確認します。詳細については、クラウド移行ファクトリー実装ガイドを参照してください。	移行スペシャリスト、クラウド管理者

関連リソース

- 移行方法
- AWS クラウド移行ファクトリー実装ガイド
- クラウド移行ファクトリーによる大規模なサーバー移行の自動化
- AWS Application Migration Service ユーザーガイド
- AWS Migration Acceleration Program

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする

作成者: Manish Garg (AWS)、T.V.R.L.Phani Kumar Dadi (AWS)、Nishad Mankar (AWS)、RAJNEESH TYAGI (AWS)

概要

多数の Microsoft SQL Server Always On フェイルオーバークラスターインスタンス (FCI) を迅速に移行する必要がある場合、このパターンを使用するとプロビジョニング時間を最小限に抑えることができます。自動化と Amazon FSx for Windows File Server を使用することで、手作業、人為的ミス、および多数のクラスターをデプロイするのに必要な時間を削減できます。

このパターンでは、Amazon Web Services (AWS) のマルチアベイラビリティーゾーン (マルチ AZ) 配置で SQL Server FCI のインフラストラクチャを設定します。このインフラストラクチャに必要な AWS サービスのプロビジョニングは、AWS CloudFormationテンプレートを使用して自動化されます。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでの SQL Server のインストールとクラスターノードの作成は、PowerShell コマンドで実行されます。

このソリューションでは、可用性の高いマルチ AZ Amazon FSx for Windows ファイルシステムが、SQL Server データベースファイルを保存するための共有モニタリングシステムとして使用されます。SQL Server をホストする Amazon FSx ファイルシステムと EC2 Windows インスタンスは、AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) ドメインに参加します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS CloudFormation テンプレートを使用してリソースをプロビジョニングするために十分な権限を持つ AWS ユーザー
- AWS Directory Service for Microsoft Active Directory
- キーと値のペアで AWS Managed Microsoft AD を認証するための AWS Secrets Manager の認証情報:
  - `ADDomainName`: <Domain Name>
  - `ADDomainJoinUserName`: <Domain Username>
  - `ADDomainJoinPassword`:<Domain User Password>

- `Target0U` : <Target OU Value>

> **Note**
>
> AWS Systems Manager オートメーションで同じキー名を使用します。

- SQL Server のインストールおよび Windows サービスまたはドメインアカウントを作成するための SQL Server メディアファイル。これらはクラスター作成時に使用されます。
- 別々のアベイラビリティーゾーンに 2 つのパブリックサブネット、アベイラビリティーゾーンに 2 つのプライベートサブネット、インターネットゲートウェイ、NAT ゲートウェイ、ルートテーブルの関連付け、ジャンプサーバーを備えた仮想プライベートクラウド (VPC)

製品バージョン

- Windows Server 2012 R2 および Microsoft SQL Server 2016

アーキテクチャ

ソーステクノロジースタック

- 共有ドライブを使用する FCI を備えたオンプレミスの SQL Server

ターゲットテクノロジースタック

- AWS EC2 インスタンス
- Amazon FSx for Windows File Server
- AWS Systems Manager Automation ランブック
- ネットワーク設定 (VPC、サブネット、インターネットゲートウェイ、NAT ゲートウェイ、ジャンプサーバー、セキュリティグループ)
- AWS Secrets Manager
- AWS Managed Microsoft AD
- Amazon EventBridge
- AWS Identity and Access Management (IAM)

ターゲット アーキテクチャ

次の図は、1 つの AWS リージョン内の AWS アカウントを示しています。その VPC には 2 つのアベイラビリティーゾーン、2 つのパブリックサブネット (NAT ゲートウェイ付き)、1 つ目のパブリックサブネットに 1 つのジャンプサーバー、2 つのプライベートサブネット (それぞれにノードセキュリティグループ内の SQL Server ノードの EC2 インスタンスがあります)、および 各 SQL Server ノードに接続する Amazon FSx ファイルシステムが含まれます。AWS Directory Service、Amazon EventBridge、AWS Secrets Manager、AWS Systems Manager も含まれます。

自動化とスケール

- AWS Systems Manager を使用して AWS Managed Microsoft AD に参加し、SQL Server のインストールを実行します。

ツール

AWS サービス

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- AWS Directory Service は、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Relational Database Service (Amazon RDS) for SQL Server、Amazon FSx for Windows File Server などの他の AWS サービスで Microsoft Active Directory (AD) を使用するための複数の方法を提供します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。たとえば、AWS Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。

- 「AWS Systems Manager」は、AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理が簡略化され、オペレーション上の問題の検出と解決時間が短縮され、AWS リソースを大規模かつセキュアに管理できるようになります。

その他のツール

- PowerShell は Windows、Linux、および macOS で動作する Microsoft の自動化および構成管理プログラムです。このパターンでは PowerShell スクリプトを使用します。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-windows-failover-cluster-automation」リポジトリで利用できます。

ベストプラクティス

- このソリューションのデプロイに使用する IAM ロールは、最小特権の原則に従う必要があります。詳細については、「IAM ドキュメント」を参照してください。
- 「AWS CloudFormation ベストプラクティス」に従ってください。

エピック

インフラストラクチャをデプロイする

タスク	説明	必要なスキル
Systems Manager CloudFormation スタックをデプロイします。	1. AWS アカウントにサインインして、AWS マネジメントコンソール を開きます。 2. CloudFormation コンソールに移動し、`ssm.yaml` テンプレートをアップロードして Systems Manager CloudFormation スタックを	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	作成します。次のパラメータの値を指定します。 • StateUnjoinAssociationLoggingBucketName — テンプレートがログ記録用に作成する S3 バケットの名前を指定します。 • SSMAssociationADUnjoinName — `AWS::SSM::Association` リソースの名前を入力します。 • SSMAutomationDocumentName — Systems Manager Automation ランブックの名前を入力します。 • EventBridgeName — EventBridge イベントバスの名前を入力します。 3. `ssm.yaml` CloudFormation テンプレートを起動して、Systems Manager CloudFormation スタックをデプロイします。このテンプレートは Systems Manager Automation ランブックを作成します。このランブックは、`ADJoined:FSXADD` タグの付いた新しい EC2 インスタンスが起動するときに開始されます。Automation ラン	

タスク	説明	必要なスキル
	ブックは、AWS Managed Microsoft AD ディレクトリにインスタンスを追加します。	

タスク	説明	必要なスキル
インフラストラクチャをデプロイします。	Systems Manager スタックのデプロイが正常に完了したら、EC2 インスタンスノード、セキュリティグループ、Amazon FSx for Windows File Server ファイルシステム、および IAM ロールを含む `infra` スタックを作成します。 1. CloudFormation コンソールに移動し、`infra-cf.yaml` テンプレートを起動します。このスタックのデプロイには、以下のパラメータが必要です。 • `ActiveDirectoryId` – AWS Managed Microsoft AD の ID • `ADDnsIpAddresses1` — AWS Managed Microsoft AD のプライマリ DNS IP アドレス • `ADDnsIpAddresses2` — AWS Managed Microsoft AD のセカンダリ DNS IP アドレス • `FSxSecurityGroupName` — Amazon FSx セキュリティグループの名前 • `FSxWindowsFileSystemName` — Amazon FSx ドライブの名前	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	• `ImageID` — SQL Server インスタンスノードの作成に使用されたベース Windows 2012 R2 のイメージまたは Amazon マシンイメージ (AMI) の ID • `KeyPairName` — EC2 インスタンスノードにアクセスするためにアタッチするキーと値のペア • `Node1SecurityGroupName` — 最初のノードのセキュリティグループの名前。 • `Node2SecurityGroupName` — 2 番目のノードのセキュリティグループの名前。 • `OUSecretName` — AWS Managed Microsoft AD 情報を含むシークレットの名前 • `PrivateSubnet1` — 最初のプライベートサブネットの ID • `PrivateSubnet2` — 2 番目のプライベートサブネットの ID • `SqlFSxFCIName` — プライマリノード、セカンダリノード、および Amazon FSx に適用されるタグの名前。	

タスク	説明	必要なスキル
	• `SqlFSxServerNetBIOSName1` — プライマリ EC2 インスタンスノードの名前 (最大 15 文字) • `SqlFSxServerNetBIOSName2` — セカンダリ EC2 インスタンスノードの名前 (最大 15 文字) • `VPC` – VPC ID • `WorkloadInstanceType` — EC2 インスタンスのタイプ `infra` スタックをデプロイします。このスタックは、Windows SQL Server FCI の設定に必要なすべてのインフラストラクチャコンポーネントを作成します。 2. EC2 インスタンスノードが起動すると、Systems Manager Automation ドキュメントが呼び出され、インスタンスが AWS Managed Microsoft AD に結合されます。Systems Manager コンソールの [オートメーション] ページで進行状況を追跡できます。	

Windows SQL Server Always On FCI をセットアップします。

タスク	説明	必要なスキル

Windows ツールのインストール	1. プライマリ EC2 インスタンス (ノード 1) にログインします。Windows の機能 (アクティブディレクトリおよび FCI ツール) をインストールするには、次の PowerShell スクリプトを実行します。
	`Install-WindowsFeature -Name RSAT-AD-Powershell,Failover-Clustering -IncludeManagementTools`
	`Install-WindowsFeature -Name RSAT-Clustering,RSAT-ADDS-Tools,RSAT-AD-Powershell,RSAT-DHCP,RSAT-DNS-Server`
	2. ノード 2 のセカンダリ EC2 インスタンスにログインし、同じスクリプトを実行してノード 2 の機能を有効にします。
		AWS DevOps、DevOps エンジニア、DBA

Active Directory ドメインサービスでクラスター化されたコンピュータオブジェクトをプレステージします。	Active Directory ドメインサービス (AD DS) のクラスター名オブジェクト (CNO) をプレステージし、クラスター化されたロールの仮想コンピューターオブジェクト (VCO) をプレステージするには、Windows Server ドキュ	AWS DevOps、DBA、DevOps エンジニア

タスク	説明	必要なスキル
	メントの指示に従ってください。	

タスク	説明	必要なスキル
WSFC を作成します。	次の手順に従って、Windows Server Failover Clustering (WSFC) クラスターを作成できます。 1. プライマリ EC2 インスタンス (ノード 1) にログインします。Amazon FSx ファイル共有を作成し、一覧表示された AD サービスアカウントへのフルアクセスを許可するには、次のコードを実行します。 `Invoke-Command -ComputerName "<FSx Windows Remote PowerShell Endpoint>" -ConfigurationName FSxRemoteAdmin -scriptblock { New-FSxSmbShare -Name "SQLDB" -Path "D:\share" -Description "SQL Databases Share" -ContinuouslyAvailable $true -FolderEnumerationMode AccessBased -EncryptData $true grant-fsxsmbshareaccess -name SQLDB -AccountName "<domain\user>" -accessRight Full }`	AWS DevOps、DBA、DevOps エンジニア

タスク	説明	必要なスキル
	このコマンドは、Microsoft SQL Server での使用に最適化され、継続的に使用可能な (CA) ファイル共有も作成します。 2. プライマリインスタンス (ノード 1) にフェイルオーバークラスターを作成するには、次のコマンドを実行します。 `New-Cluster -Name <CNO Name> -Node <Node1 Name>, <Node2 Name> -StaticAddress <Node1 Secondary Private IP>, <Node2 Secondary Private IP>` このコマンドには、以下のパラメータが必要となります。 • Name - クラスターの名前 (CNO) • Node — プライマリノードとセカンダリノードの名前 • StaticAddress — プライマリノードとセカンダリノードのセカンダリ IP アドレス	

タスク	説明	必要なスキル
	Important Windows Server フェイルオーバークラスタリング (WSFC) クラスターを作成するには、ドメイン管理者または通常のユーザーに両方のノードに対する管理者権限が必要です。それ以外の場合、前回のコマンドは失敗し、メッセージ `You do not have administrator privilege on servers` が返されます。 3. クラスターを作成したら、次のコマンドを実行してファイル共有モニタリングを追加します。 `Set-ClusterQuorum -FileShareWitness \ \<FSx Windows Remote PowerShell Endpoint> \share\witness`	

タスク	説明	必要なスキル
SQL Server フェイルオーバークラスターをインストールします。	WSFC クラスターを設定したら、SQL Server クラスターをプライマリインスタンス (ノード 1) にインストールします。 1. 両方のノードの T ドライブに、tempdb フォルダと log フォルダを作成します。これらのフォルダは PowerShell コマンドで使用されます。 2. SQL Server をインストールするための SQL Server メディアファイルを両方のノードにコピーした後、ノード 1 で次の PowerShell コマンドを実行して SQL Server をノード 1 にインストールします。 `D:\setup.exe /Q ` ` /ACTION=InstallFailoverCluster ` /IACCEPTSQLSERVERLICENSETERMS ` /FEATURES="SQL,IS,BC,Conn" ` /INSTALLSHAREDDIR="C:\Program Files\Microsoft SQL Server” ` /INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server" ` /RSINSTALLMODE="FilesOnlyMode" ` `	AWS DevOps、DBA、DevOps エンジニア

タスク	説明	必要なスキル
	/INSTANCEID="MSS QLSERVER" ` /INSTANCENAME="M SSQLSERVER" ` /FAILOVERCLUSTER GROUP="SQL Server (MSSQLSERVER)" ` /FAILOVERCLUSTER IPADDRESSES="IPv4; <2nd Sec Private Ip node1>;Cluster Network 1;<subnet mask>" ` /FAILOVERCLUSTER NETWORKNAME="<Fail over cluster Network Name>" ` /INSTANCEDIR="C: \Program Files\Mic rosoft SQL Server" ` /ENU="True" ` /ERRORREPORTING=0 ` /SQMREPORTING=0 ` /SAPWD="<Domain User password>" ` /SQLCOLLATION="S QL_Latin1_General_ CP1_CI_AS" ` /SQLSYSADMINACCO UNTS="<domain\user name>" ` /SQLSVCACCOUNT=" <domain\username>" /SQLSVCPASSWORD="< Domain User password>" ` /AGTSVCACCOUNT=" <domain\username>" /AGTSVCPASSWORD="< Domain User password>" ` /ISSVCACCOUNT="<domain \username>" /ISSVCPAS	

タスク	説明	必要なスキル
	SWORD="<Domain User password>" ` /FTSVCACCOUNT="NT Service\MSSQLFDLauncher" ` /INSTALLSQLDATADIR="\\<FSX DNS name>\share\Program Files\Microsoft SQL Server" ` /SQLUSERDBDIR="\\<FSX DNS name>\share\data" ` /SQLUSERDBLOGDIR="\\<FSX DNS name>\share\log" ` /SQLTEMPDBDIR="T:\tempdb" ` /SQLTEMPDBLOGDIR="T:\log" ` /SQLBACKUPDIR="\\<FSX DNS name>\share\SQLBackup" ` /SkipRules=Cluster_VerifyForErrors ` /INDICATEPROGRESS	

タスク	説明	必要なスキル
セカンダリノードをクラスターに追加します。	SQL Server をセカンダリノード (ノード 2) に追加するには、次の PowerShell コマンドを実行します。 `D:\setup.exe /Q ` `/ACTION=AddNode ` `/IACCEPTSQLSERVERLICENSETERMS ` `/INSTANCENAME="MSSQLSERVER" ` `/FAILOVERCLUSTERGROUP="SQL Server (MSSQLSERVER)" ` `/FAILOVERCLUSTERIPADDRESSES="IPv4;<2nd Sec Private Ip node2>;Cluster Network 2;<subnet mask>" ` `/FAILOVERCLUSTERNETWORKNAME="<Fail over cluster Network Name>" ` `/CONFIRMIPDEPENDENCYCHANGE=1 ` `/SQLSVCACCOUNT="<domain\username>" /SQLSVCPASSWORD="<Domain User password>" ` `/AGTSVCACCOUNT="domain\username>" /AGTSVCPASSWORD="<Domain User password>" ` `/FTSVCACCOUNT="NT Service\MSSQLFDLauncher" ` `/SkipRules=Cluster_VerifyForErrors `	AWS DevOps、DBA、DevOps エンジニア

タスク	説明	必要なスキル
	`/INDICATEPROGRESS`	
SQL Server FCI をテストします。	1. いずれかのノードの Windows インスタンスの管理ツールで、Failover Cluster Manager を起動します。 2. [ノード] に移動し、ノードのステータスが実行中になっていることを確認します。 3. [ロール] を選択し、SQL Server (MSSQLSERVER) のコンテキスト (右クリック) メニューを開き、[移動してノードを選択] を選択します。 4. ノードを選択したら、SQL Server を別のノードで実行する必要があります。	DBA、DevOps エンジニア

リソースをクリーンアップする

タスク	説明	必要なスキル
リソースをクリーンアップします。	リソースをクリーンアップするには、AWS CloudFormation スタックの削除プロセスを実行します。 1. AWS CloudFormation コンソールを開きます。 2. [スタック] ページで、`infra` スタックを選	AWS DevOps、DBA、DevOps エンジニア

タスク	説明	必要なスキル
	択します。スタックは現在実行中である必要があります。 3. [スタックの詳細] ペインで、[削除] を選択します。 4. プロンプトが表示されたら、[スタックの削除] を選択します。 5. ssm スタックについては、ステップ 2 ~ 4 を繰り返します。 スタックの削除が完了すると、スタックの状態が DELETE_COMPLETE になります。デフォルトでは、DELETE_COMPLETE 状態のスタックは CloudFormation コンソールに表示されません。削除したスタックを表示するには、「削除済みのスタックを AWS CloudFormation コンソールで確認する」で説明されているようにスタックの表示フィルターを変更する必要があります。 削除が失敗した場合、スタックは DELETE_FAILED 状態になります。解決策については、CloudFormation のドキュメントの「スタックの削除の失敗」を参照してください。	

トラブルシューティング

問題	ソリューション
AWS CloudFormation テンプレートの失敗	CloudFormation テンプレートがデプロイ中に失敗した場合は、次の操作を行います。 1. AWS CloudFormation コンソールを開きます。 2. CloudFormation コンソールの [スタック] ページで、スタックを選択します。 3. [イベント] を選択し、スタックのステータスを確認します。
AWS Managed Microsoft AD への参加に失敗	参加に関する問題のトラブルシューティングには、以下の手順を実行します。 1. Systems Manager コンソールを開きます。 2. デプロイのリージョンを選択します。 3. 左側のペインで [オートメーション] を選択し、失敗したオートメーションランブックを探します。 4. オートメーションランブックを開き、実行ステータスと実行ステップを確認します。 5. 失敗したステップの詳細を調べて、正確なエラーまたは失敗を確認します。

関連リソース

- 「Amazon FSx for Windows File Server を使用して、Microsoft SQL Server の高可用性デプロイメントを簡素化する」
- 「Microsoft SQL Server で FSx for Windows File Server を使用する」

BMC ディスカバリークエリを使用して移行計画のために移行データを抽出

作成者: Ben Tailor-Hamblin (AWS), Simon Cunningham (AWS), Emma Baldry (AWS), と Shabnam Khan (AWS)

概要

このガイドには、BMC ディスカバリーを使用してオンプレミスのインフラストラクチャとアプリケーションからデータを抽出するためのクエリの例と手順が記載されています。このパターンは、BMC ディスカバリークエリを使用して、インフラストラクチャをスキャンし、ソフトウェア、サービス、依存関係の情報を抽出する方法を示します。抽出されたデータは、Amazon Web Services (AWS) クラウドへの大規模な移行の評価段階と準備段階に必要です。このデータを使用して、移行計画の一環としてどのアプリケーションをまとめて移行するかという重要な決定を下すことができます。

前提条件と制限

前提条件

- BMC ディスカバリー (旧 BMC ADDM) または BMC ヘリックスディスカバリーのsoftware as a service (SaaS)バージョンのライセンス
- オンプレミスまたは SaaS バージョンの BMC Discovery がインストールされている

> **Note**
>
> BMC Discovery のオンプレミスバージョンでは、複数のデータセンター間での移行の対象となるすべてのネットワークおよびサーバーデバイスにアクセスできるクライアントネットワークにアプリケーションをインストールする必要があります。クライアントネットワークへのアクセスは、アプリケーションのインストール手順に従って提供する必要があります。Windows Server 情報のスキャンが必要な場合は、ネットワークで Windows プロキシマネージャーデバイスをセットアップする必要があります。

- BMC ヘリックスディスカバリーを使用する場合、「ネットワークアクセス」により、データセンターを渡ってデバイスをスキャンできるようにします

製品バージョン

- BMC ディスカバリー 22.2 (12.5)
- BMC ディスカバリー 22.1 (12.4)

- BMC ディスカバリー 21.3 (12.3)
- BMC ディスカバリー 21.05 (12.2)
- BMC ディスカバリー 20.08 (12.1)
- BMC ディスカバリー 20.02 (12.0)
- BMC ディスカバリー 11.3
- BMC ディスカバリー 11.2
- BMC ディスカバリー 11.1
- BMC ディスカバリー 11.0
- BMC アトリウムディスカバリー 10.2
- BMC アトリウムディスカバリー 10.1
- BMC アトリウムディスカバリー 10.0

アーキテクチャ

次の図表は、資産管理者が BMC ディスカバリークエリを使用して、SaaS 環境とオンプレミス 環境の両方でBMCモデル化されたアプリケーションをスキャンする方法を示します。

この図表は次のワークフローを示しています: 資産管理者は BMC ディスカバリー または BMC ヘリックスディスカバリーを使用して、複数の物理サーバーでホストされている仮想サーバーで実行されているデータベースとソフトウェアインスタンスをスキャンします。このツールは、複数の仮想サーバーと物理サーバーにまたがるコンポーネントを含むアプリケーションをモデル化できます。

テクノロジースタック

- BMC ディスカバリー
- BMC ヘリックスディスカバリー

ツール

- 「BMC ディスカバリー」 は、データセンターを自動的に検出するためのデータセンター検出ツールです。
- 「BMC ヘリックスディスカバリー」 は、 SaaS ベースの検出および依存関係モデリングシステムで、データ資産とその依存関係を動的にモデル化することを支援します。

ベストプラクティス

ベストプラクティスは、クラウドに移行する場合、アプリケーション、依存関係、インフラストラクチャのデータをマッピングすることです。マッピングは、現在の環境の複雑さや、さまざまなコンポーネント間の依存関係の把握に役立ちます。

これらのクエリより提供される資産情報は、いくつかの理由で重要です：

1. 「計画 」— コンポーネント間の依存関係を理解することで、移行プロセスをより効果的に計画することを支援します。たとえば、他のコンポーネントを確実に正常に移行するためには、まず特定のコンポーネントを移行する必要がある場合があります。
2. 「リスク評価 」— コンポーネント間の依存関係をマッピングすることは、移行プロセス中に発生する可能性のある潜在的なリスクや問題を特定することに役立ちます。たとえば、特定のコンポーネントが、クラウドで問題を引き起こす可能性のある古いテクノロジーや適用されないテクノロジーに依存していることに気付くかもしれません。
3. 「クラウドアーキテクチャ 」— アプリケーションとインフラストラクチャのデータをマッピングすることで、組織のニーズを満たす適切なクラウドアーキテクチャを設計することを支援します。たとえば、高い可用性やスケーラビリティの要件をサポートする多層アーキテクチャを設計する必要があるかもしれません。

全体として、アプリケーション、依存関係、インフラストラクチャのデータをマッピングすることは、クラウド移行プロセスの重要なステップです。マッピングの実践は、現在の環境をよりよく理解し、潜在的な問題やリスクを特定し、適切なクラウドアーキテクチャを設計することを支援します。

エピック

ディスカバリーツールの特定と評価

タスク	説明	必要なスキル
ITSM 所有者を特定します。	IT サービス管理 (ITSM) のオーナーを特定します (通常はオペレーションサポートチームに連絡)。	移行リード
CMDB を確認します。	資産情報を含む構成管理データベース (CMDB) の数を特定	移行リード

タスク	説明	必要なスキル
	し、次にその情報のソースを特定します。	
検出ツールを特定し、BMC ディスカバリーの使用を確認します。	組織が BMC ディスカバリーを使用して、環境に関するデータを CMDB ツールに送信している場合、そのスキャンの範囲と対象範囲を確認します。たとえば、BMC ディスカバリーがすべてのデータセンターをスキャンしているかどうか、アクセスサーバーが境界ゾーンにあるかどうかを確認します。	移行リード
アプリケーションモデリングのレベルをチェックします。	アプリケーションが BMC ディスカバリーでモデル化されているかどうかを確認します。そうでない場合は、BMC ディスカバリーツールを使用して、実行中のどのソフトウェアインスタンスがアプリケーションとビジネスサービスを提供するかをモデル化することを推奨します。	移行エンジニア、移行リード

インフラストラクチャデータの抽出

タスク	説明	必要なスキル
物理サーバーと仮想サーバーでデータを抽出します。	BMC ディスカバリー によってスキャンされた物理サーバーと仮想サーバーのデータを抽出するには、「クエリビ	移行エンジニア、移行リード

タスク	説明	必要なスキル
	ルダー」 を使用して次のクエリを実行します： `search Host show key as 'Serverid', virtual, name as 'HOSTNAME', os_type as 'osName', os_version as 'OS Version', num_logical_processors as 'Logical Processor Counts', cores_per_processor as 'Cores per Processor', logical_ram as 'Logical RAM', #Consumer:StorageUse:Provider:DiskDrive.size as 'Size'` **Note** 抽出されたデータを使用して、移行に適したインスタンスサイズを決定できます。	

タスク	説明	必要なスキル
モデル化されたアプリケーションのデータを抽出します。	アプリケーションが BMCディスカバリーでモデル化されている場合、アプリケーションソフトウェアを実行するサーバーに関するデータを抽出できます。サーバー名を取得するには、「クエリビルダー」を使用して次のクエリを実行します： `search SoftwareInstance show key as 'ApplicationID', #RunningSoftware:HostedSoftware:Host:Host.key as 'ReferenceID', type, name` **Note** アプリケーションは、実行中のソフトウェアインスタンスのコレクションによって BMC Discovery でモデル化されます。アプリケーションは、アプリケーションソフトウェアを実行するすべてのサーバーに依存します。	BMC ディスカバリのアプリ所有者

タスク	説明	必要なスキル
データベースからデータを抽出します。	スキャンされたすべてのデータベースと、それらのデータベースが実行されているサーバーのリストを取得するには、「クエリビルダー」を使用して次のクエリを実行します： `search Database show key as 'Key', name, type as 'Source Engine Type', #Detail:Detail:ElementWithDetail:SoftwareInstance.name as 'Software Instance', #Detail:Detail:ElementWithDetail:SoftwareInstance.product_version as 'Product Version', #Detail:Detail:ElementWithDetail:SoftwareInstance.edition as 'Edition', #Detail:Detail:ElementWithDetail:SoftwareInstance.#RunningSoftware:HostedSoftware:Host:Host.key as 'ServerID'`	アプリ所有者

タスク	説明	必要なスキル
サーバー通信のデータを抽出します。	BMC ディスカバリによって収集されたサーバー間のすべてのネットワーク通信に関する情報を過去のネットワーク通信ログから取得するには、「クエリビルダー」を使用して次のクエリを実行します `search Host TRAVERSE InferredElement:Inference:Associate:Discovery Access TRAVERSE Discovery Access:DiscoveryAccessResult:DiscoveryResult:NetworkConnectionList TRAVERSE List:List:Member:Discovered NetworkConnection PROCESS WITH networkConnectionInfo`	BMC ディスカバリのアプリ所有者
アプリケーションディスカバリーのデータを抽出します。	アプリケーションの依存関係の情報を取得するには、「クエリビルダー」 を使用して次のクエリを実行します： `search SoftwareInstance show key as 'SRC App ID', #Dependant:Dependency:DependedUpon:SoftwareInstance.key as 'DEST App ID'`	BMC ディスカバリのアプリ所有者

タスク	説明	必要なスキル
ビジネスサービスのデータを抽出します。	ホストが提供するビジネスサービスのデータを抽出するには、「クエリビルダー」を使用して次のクエリを実行します： `search Host show name, #Host:HostedSoftware:AggregateSoftware:BusinessService.name as 'Name'`	BMC ディスカバリのアプリ所有者

トラブルシューティング

問題	ソリューション
クエリの実行がされないか、入力されていない列が含まれています。	BMC ディスカバリのアセットレコードを確認し、どのフィールドが必要かを判断します。次に、「クエリビルダー」を使用してクエリ内のこれらのフィールドを置き換えます。
依存資産の詳細は入力されません。	これは、おそらくアクセス権限またはネットワーク接続によるものと考えられます。ディスカバリーツールでは、特に、異なるネットワークや環境にある時、特定の資産へのアクセスに必要な権限がない場合があります。 ディスカバリ主題専門家と緊密に連携して、関連するすべての資産が確実に特定されるようにすることを推奨します。

関連リソース

リファレンス

- 「BMC ディスカバリー・ ライセンス資格」 (BMC 文書)
- 「BMC ディスカバリーの特徴量とコンポーネント」 (BMC ドキュメント)
- 「BMC ディスカバリーユーザーガイド」 (BMC ドキュメント)
- 「データの検索 (BMC ディスカバリーについて)」 (BMC ドキュメント)
- 「移行のためのポートフォリオの発見と分析 (AWS 規範ガイダンス)

チュートリアルと動画

- 「BMC ディスカバリー:ウェビナー-レポーティングクエリのベストプラクティス (第1部分)」 (YouTube)

リロケート

トピック

- Amazon RDS for Oracle データベースを別の に移行 AWS アカウント し、 AWS リージョン を使用して継続的なレプリケーション AWS DMS を行う
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX
- Amazon RDS DB インスタンスを別の VPC またはアカウントに移行する
- Amazon RDS for Oracle DB インスタンスを別の VPC へ移行する
- Amazon Redshift クラスターを中国の AWS リージョンに移行する
- VMware HCX を使用して VMware Cloud on AWS ワークロードを移行
- pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する

Amazon RDS for Oracle データベースを別の に移行 AWS アカウント し、 AWS リージョン を使用して継続的なレプリケーション AWS DMS を行う

作成者: Durga Prasad Cheepuri (AWS) と Eduardo Valentim (AWS)

概要

> **Warning**
>
> IAM ユーザーは長期認証情報を保有するため、セキュリティ上のリスクが生じます。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

このパターンでは、Oracle ソースデータベース用の Amazon Relational Database Service (Amazon RDS) を別の AWS アカウント および に移行する手順について説明します AWS リージョン。このパターンでは、DB スナップショットを 1 回限りのフルデータロードに使用し、継続的なレプリケーションに enable AWS Database Migration Service （ AWS DMS) を使用します。

前提条件と制限

前提条件

- デフォルト以外の AWS Key Management Service （ AWS KMS) キーを使用して暗号化されたソース Amazon RDS for Oracle データベース AWS アカウント を含むアクティブな 。
- ターゲット Amazon RDS for Oracle データベースに使用する、ソースデータベース AWS リージョン とは異なる AWS アカウント でアクティブな 。
- ソース VPC とターゲット VPC 間の仮想プライベートクラウド (VPC) ピアリング
- Oracle データベースを のソースとして使用する AWS DMS方法に関する知識
- のターゲットとして Oracle データベースを使用することに精通している AWS DMS

製品バージョン

- Oracle バージョン 11g (バージョン 11.2.0.3.v1 以降) ~ 12.2、および 18c。サポートされているバージョンとエディションの最新リストについては、 AWS ドキュメントの「 のソースとしての Oracle データベースの使用 AWS DMS」および「 のターゲットとしての Oracle データベースの

使用 AWS DMS」を参照してください。Amazon RDS でサポートされている Oracle バージョンについては、Oracle on Amazon RDS を参照してください。

アーキテクチャ

ソースとターゲットテクノロジースタック

- Amazon RDS for Oracle DB インスタンス

継続的なレプリケーションアーキテクチャ

ツール

1 回限りのフルデータロードに使用されるツール

- Amazon Relational Database Service (Amazon RDS) は、DB インスタンスのストレージボリュームスナップショットを作成し、個々のデータベースだけでなく、DB インスタンス全体をバックアップします。DB スナップショットを作成したら、バックアップする DB インスタンスを識別した後、DB スナップショットに名前を付けて後で復元できるようにする必要があります。スナップショットの作成にかかる時間は、データベースのサイズによって異なります。スナップショットにはストレージボリューム全体が含まれているため、一時ファイルなどのファイルのサイズも、スナップショットを作成する時間に影響します。DB スナップショットを作成する方法については、Amazon RDS ドキュメントの DB スナップショットを作成するを参照してください。
- AWS Key Management Service （ AWS KMS ） は Amazon RDS 暗号化用の キーを作成します。暗号化された DB インスタンスを作成するときに、暗号化AWS KMSキーのキー識別子を指定することもできます。AWS KMS キー識別子を指定しない場合、Amazon RDS は新しい DB インスタンスにデフォルトの暗号化キーを使用します。 AWS KMSは のデフォルトの暗号化キーを作成します AWS アカウント。 AWS アカウント には、それぞれ異なるデフォルトの暗号化キーがあります AWS リージョン。このパターンでは、Amazon RDS DB インスタンスはデフォルト以外のAWS KMSキーを使用して暗号化する必要があります。Amazon RDS 暗号化のAWS KMSキーの使用の詳細については、Amazon RDS ドキュメントの「Amazon RDS リソースの暗号化」を参照してください。

継続的なレプリケーションに使用されるツール

- AWS Database Migration Service （ AWS DMS ） は、進行中の変更をレプリケートし、ソースデータベースとターゲットデータベースを同期させるために使用されます。継続的なレプリケーション AWS DMS に を使用する方法の詳細については、 AWS DMS ドキュメントの「レAWS DMS プリケーションインスタンスの使用」を参照してください。

エピック

ソースを設定する AWS アカウント

タスク	説明	必要なスキル
ソース Oracle DB インスタンスを準備します。	Amazon RDS for Oracle DB インスタンスをアーカイブログモードで実行し、保持期間を設定します。詳細については、「 のソースとしてのAWS マネージド Oracle データベースの使用 AWS DMS」を参照してください。	DBA
ソース Oracle DB インスタンスのサプリメンタルロギングを設定します。	Amazon RDS for Oracle DB インスタンスのデータベースレベルとテーブルレベルのサプリメンタルロギングを設定します。詳細については、「 のソースとしての AWS マネージド Oracle データベースの使用 AWS DMS」を参照してください。	DBA
ソースアカウントの AWS KMS キーポリシーを更新します。	ソースの AWS KMS キーポリシーを更新 AWS アカウントして、ターゲットが暗号化された Amazon RDS AWS KMS キー AWS アカウント を使用できるようにします。詳細に	SysAdmin

タスク	説明	必要なスキル
	ついては、 AWS KMS ドキュメントを参照してください。	
ソース DB インスタンスの手動 Amazon RDS DB スナップショットを作成します。		AWS IAM ユーザー
手動の暗号化された Amazon RDS スナップショットをターゲットと共有します AWS アカウント。	詳細については、「DB スナップショットの共有」を参照してください。	AWS IAM ユーザー

ターゲットを設定する AWS アカウント

タスク	説明	必要なスキル
ポリシーをアタッチします。	ターゲットで AWS アカウント、 AWS Identity and Access Management (IAM) ポリシーをルート IAM ユーザーにアタッチして、IAM ユーザーが共有 AWS KMS キーを使用して暗号化された DB スナップショットをコピーできるようにします。	SysAdmin
ソースに切り替えます AWS リージョン。		AWS IAM ユーザー
共有したスナップショットをコピーします。	Amazon RDS コンソールのスナップショットペインで、自分と共有を選択し、共有スナップショットを選択します。ソースデータベースで使用される AWS KMS キー	AWS IAM ユーザー

タスク	説明	必要なスキル
	の Amazon リソースネーム (ARN) を使用して、ソースデータベース AWS リージョン と同じ にスナップショットをコピーします。詳細については、「DB スナップショットのコピー」を参照してください。	
ターゲットに切り替え AWS リージョン、新しい AWS KMS キーを作成します。		AWS IAM ユーザー
スナップショットをコピーします。	ソースに切り替えます AWS リージョン。Amazon RDS コンソールのスナップショットペインで、自分が所有するを選択し、コピーしたスナップショットを選択します。新しいターゲットの AWS KMS キー AWS リージョン を使用して、スナップショットをターゲットにコピーします AWS リージョン。	AWS IAM ユーザー

タスク	説明	必要なスキル
スナップショットを復元します。	ターゲットに切り替えます AWS リージョン。Amazon RDS コンソールのスナップショットペインで、自分が所有する を選択します。コピーしたスナップショットを選択し、Amazon RDS for Oracle DB インスタンスに復元します。詳細については、「DB スナップショットからの復元」を参照してください。	AWS IAM ユーザー

継続的なレプリケーションに備えてソースデータベースを準備する

タスク	説明	必要なスキル
適切な権限がある Oracle ユーザーを作成します。	ソースとして Oracle に必要な権限を持つ Oracle ユーザーを作成します AWS DMS。詳細については、 AWS DMS ドキュメントを参照してください。	DBA
Oracle LogMiner または Oracle Binary Reader のソースデータベースを設定します。		DBA

継続的なレプリケーションに備えてターゲットデータベースを準備する

タスク	説明	必要なスキル
適切な権限がある Oracle ユーザーを作成します。	ターゲットとして Oracle に必要な権限を持つ Oracle	DBA

タスク	説明	必要なスキル
	ユーザーを作成します AWS DMS。詳細については、AWS DMS ドキュメントを参照してください。	

AWS DMS コンポーネントの作成

タスク	説明	必要なスキル
ターゲットにレプリケーションインスタンスを作成します AWS リージョン。	ターゲットの VPC にレプリケーションインスタンスを作成します AWS リージョン。詳細については、AWS DMS ドキュメントを参照してください。	AWS IAM ユーザー
必要な暗号化でソースエンドポイントとターゲットエンドポイントを作成し、接続をテストします。	詳細については、AWS DMS ドキュメントを参照してください。	DBA
レプリケーションタスクを作成します。	1. 移行タイプには、継続的レプリケーションを選択します。 2. 変更データキャプチャ (CDC) の開始点には、Amazon RDS スナップショットが全ロード用に撮られたときの Oracle システム変更番号 (SCN)、またはすべてがロードされたときのタイムスタンプを使用します。 3. で`TargetTablePrepMode` 、DO_NOTHING を選択	IAM ユーザー

タスク	説明	必要なスキル
	します。タスクにラージバイナリオブジェクト (LOB) データテーブルがある場合は、制限付き LOB モードを選択し、最大 LOB サイズをテーブル内の LOB データの最大サイズに設定します。 4. ログ作成を有効化します。 5. キーで関連付けられているテーブルを 1 つのタスクにグループ化します。大量の LOB データを含むテーブルがあり、そのテーブルが他のテーブルと関係がない場合は、前述の LOB 設定を使用してそのテーブル用に別のタスクを作成します。 詳細については、AWS DMS ドキュメントを参照してください。	
タスクを開始して監視します。	詳細については、AWS DMS ドキュメントを参照してください。	AWS IAM ユーザー
必要に応じて、タスクの検証を有効化します。	検証を有効化すると、レプリケーションのパフォーマンスに影響することに注意してください。詳細については、AWS DMS ドキュメントを参照してください。	AWS IAM ユーザー

関連リソース

- キーポリシーの変更
- 手動 Amazon RDS DB スナップショットを作成する
- 手動 Amazon RDS DB スナップショットを共有する
- スナップショットをコピーする
- Amazon RDS Custom DB スナップショットから復元する
- の開始方法 AWS DMS
- のソースとしての Oracle データベースの使用 AWS DMS
- のターゲットとしての Oracle データベースの使用 AWS DMS
- AWS DMS VPC ピアリングを使用した のセットアップ
- 手動 Amazon RDS DB スナップショットまたは DB クラススナップショットを別の と共有するにはどうすればよいですか AWS アカウント？ AWS ナレッジセンターの記事)

Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX

作成者: Deepak Kumar (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 の AWS 担当者にお問い合わせください。

このパターンでは、VMware Hybrid Cloud Extension（HCX）を使用して、オンプレミスの仮想マシン（VM）とアプリケーションをAmazon Web Services（AWS）上のVMware Cloudに移行する方法について説明します。移行では、AWS クラウド上の VMware エンタープライズクラスのソフトウェア定義データセンター (SDDC) ソフトウェアを使用して、AWS サービスへの最適なアクセスを提供します。

VMware Cloud on AWS は、VMware の主力製品であるコンピューティング、ストレージ、ネットワーク仮想化製品 (vSphere、vSAN、NSX) と vCenter 管理を統合し、これらのサービスを柔軟性のあるベアメタル インフラストラクチャ上で実行するように最適化します。その結果生まれたインフラストラクチャは、メンテナンスが少なく、シンプルで、ハイパーコンバージド型になっています。

このサービスにより、IT チームは使い慣れた VMware ツールを使用してクラウドベースのリソースを管理できます。詳細については、VMware ドキュメントの「VMware Cloud on AWS」を参照してください。

VMware HCXは3種類のクラウド移行をサポートしています。

- ハイブリディティ (データセンターの拡張) — 既存のオンプレミスの VMware SDDC を クラウドに拡張して、設置面積の拡大、オンデマンド容量、テスト/開発環境、仮想デスクトップを提供している。
- クラウドからの排除 (データセンターのインフラストラクチャの更新) — データセンターを統合し、AWS クラウドに完全に移行している。これにはデータセンターのコロケーションやリース終了のシナリオの処理を含む。
- アプリケーション固有移行 — 個々のアプリケーションを AWS クラウドに移動して、特定のビジネスニーズを満たしている。

前提条件と制限

前提条件

- AWSアカウントにサインアップします（VMware Cloud SDDCの作成に必要である）。
- My VMware アカウントにサインアップします。「https：//my.vmware.com/web/vmware/」で登録し、すべてのフィールドに入力してください。
- vCenter とホストのバージョンを確認し、仮想マシンの数を収集します。可能であれば、仮想環境に関する情報を表示するために「RVTools」のエクスポートを依頼してください。バージョン 6.0 以降をお勧めします。
- データセンターネットワーク (L2) を拡張したり、HCX を使用して vMotion をテストしたり、vRealize Network Insight を使用してアプリケーションの依存関係を分析したりする場合は、分散仮想スイッチを導入する必要があります。
- 競合のないオンプレミスの現在の管理サブネットネットワークを選択して、VMware Cloud on AWS に SDDC を作成します。
- 「VMware HCX ユーザーガイド」に記載されている前提条件を確認して HCX 要件を検証してください。
- 移行の波に備えて VM を特定してグループ化します。テストに使用できる VM がないか確認する。
- 相対的な帯域幅使用量、WAN 圧縮、データ転送速度に関するデータを収集します。

Notes (メモ)

- VMware NSX-V や NSX-T をオンプレミスで使用する必要はありません。
- HCX には追加コストはかかりません (VMware Cloud on AWS に含まれています)。

アーキテクチャ

次の図は、複数のコンポーネントサービス上に構築された HCX ソリューションを示します。各コンポーネントは HCX ソリューションの特定の機能をサポートします。各 HCX コンポーネントの詳細については、ブログ記事「ハイブリッドクラウドエクステンション (HCX) による VMware Cloud on AWS へのワークロードの移行」を参照してください。

ソーステクノロジースタック

- VMware vSphere によって管理されるオンプレミスの仮想マシンとアプリケーション

ターゲットテクノロジースタック

- VMware Cloud on AWS

ツール

- VMware HCX — VMware HCX は、データセンターやクラウド環境間でアプリケーションやワークロードを移行するために使用できるツールです。VMware Cloud on AWS Sに含まれています。

エピック

移行を計画する

タスク	説明	必要なスキル
移行戦略を選択します。	データセンターを拡張 (ハイブリディティ) するか、すべてのデータセンターを移転 (クラウド退避) するか、特定のアプリケーションを AWS に移行するかを決めてください。	SysAdmin、アプリオーナー
HCX の要件を検証します。	移行情報については、「VMware HCX ユーザーガイド」を参照してください。	SysAdmin、アプリオーナー

VMware Cloud on AWS

タスク	説明	必要なスキル
VM またはアプリケーションを移動します。	詳細については、VMware ドキュメントの「Hybrid Migration with VMware HCXn」を参照してください。	SysAdmin、アプリオーナー

関連リソース

- VMware Cloud on AWS： 入門
- VMware HCX によるハイブリッド移行
- 「VMware HCX User Guide」
- VMware Cloud on AWS 価格
- VMware Cloud on AWS ロードマップ

Amazon RDS DB インスタンスを別の VPC またはアカウントに移行する

作成者:Dhrubajyoti Mukherjee (AWS)

概要

このパターンは、Amazon Relational Database Service (Amazon RDS) DB インスタンスを、ある仮想プライベートクラウド (VPC) から同じ AWS アカウント内の別の仮想プライベートクラウド (VPC) に、またはある AWS アカウントから別の AWS アカウントに移行するためのガイダンスを提供します。

このパターンは、分離またはセキュリティ上の理由から Amazon RDS DB インスタンスを別の VPC またはアカウントに移行する場合 (たとえば、アプリケーションスタックとデータベースを別の VPC に配置する場合など) に役立ちます。

DB インスタンスを別の AWS アカウントに移行するには、手動スナップショットの作成、共有、ターゲットアカウントでのスナップショットの復元などの手順が必要です。この処理は、データベースの変更やトランザクションレートによっては時間がかかる場合があります。また、データベースのダウンタイムも発生するため、事前に移行計画を立ててください。ダウンタイムを最小限に抑えるため、ブルー/グリーンデプロイ戦略を検討してください。 または、AWS データ移行サービス (AWS DMS) を評価して、変更によるダウンタイムを最小限に抑えることもできます。ただし、このパターンではこのオプションは対象外です。詳細については、「メトリクスのドキュメント」 を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- VPC、サブネット、Amazon RDS コンソールに必要な AWS Identity and Access Management (IAM) のアクセス許可

制約事項

- VPC を変更するとデータベースが再起動し、アプリケーションが停止します。移行はピークの少ない時間に移行することをお勧めします。
- Amazon RDS を別の VPC に移行する場合の制限事項:
  - 移行する DB インスタンスは、スタンバイのない 1 つのインスタンスである必要があります。クラスターのメンバーであってはなりません。

  - Amazon RDS が複数のアベイラビリティーゾーンに置かれていてはなりません。
  - Amazon RDS にはリードレプリカがあってはなりません。
  - ターゲット VPC で作成されたサブネットグループには、ソースデータベースが実行されているアベイラビリティーゾーンのサブネットが必要です。
- Amazon RDS を別の RDS に移行する場合の制限事項:
  - Amazon RDS のデフォルトサービスキーで暗号化されたスナップショットの共有は、現在サポートされていません。

アーキテクチャ

同じ AWS アカウントの VPC への移行

次の図は、Amazon RDS DB インスタンスを同じ AWS アカウントの別の VPC に移行するためのワークフローを示しています。

ステップには、以下があります。詳細な手順については、「エピック」 セクションを参照してください。

1. ターゲット VPC に DB サブネットグループを作成します。DB サブネットグループは DB インスタンスを作成する場合に特定の VPC を指定するサブネットのコレクションです。
2. 新しい DB サブネットグループを使用するように、ソース VPC の Amazon RDS DB インスタンスを設定します。
3. 変更を適用して Amazon RDS DB をターゲット VPC に移行します。

別の AWS アカウントに移行

次の図は、Amazon RDS DB インスタンスを別の AWS アカウントに移行するワークフローを示しています。

ステップには、以下があります。詳細な手順については、「エピック」 セクションを参照してください。

1. 作成元の AWS アカウントで Amazon RDS DB インスタンスにアクセスします。
2. 作成元の AWS アカウントに Amazon RDS スナップショットを作成します。

3. Amazon RDS スナップショットをターゲット AWS アカウントと共有します。
4. ターゲット AWS アカウントの Amazon RDS スナップショットにアクセスします。
5. ターゲット AWS アカウントに Amazon RDS DB インスタンスを作成します。

ツール

AWS サービス

- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

ベストプラクティス

- Amazon RDS DB インスタンスを別のアカウントに移行する際にデータベースのダウンタイムが懸念される場合は、「AWS DMS」を使用することをお勧めします。このサービスではデータ複製が可能で、停止時間は 5 分未満です。

エピック

同じ AWS アカウントの別の VPC に移行する

タスク	説明	必要なスキル
新しい VPC を作成します。	「Amazon VPC コンソール」で、必要なプロパティと IP アドレス範囲を使用して新しい VPC とサブネットを作成します。詳細な手順については、「Amazon VPC のドキュメント」 を参照してください。	管理者
DB サブネットグループを作成します。	「Amazon RDS コンソール」で:	管理者

タスク	説明	必要なスキル
	1. サブネットグループを選択して、DB サブネットグループを作成します。 2. サブネットグループ名、説明、VPC ID を入力します。 3. サブネットグループに属するサブネットを追加します。サブネットを追加して、少なくとも 2 つのアベイラビリティーゾーンをカバーします。 4. [Create] (作成) を選択します。 詳細については、「Amazon EC2 ドキュメント」を参照してください。	

タスク	説明	必要なスキル
新しいサブネットグループを使用するように Amazon RDS DB インスタンスを変更します。	Amazon RDS コンソールで: 1. ナビゲーションペインで、データベースを選択し、移行する Amazon RDS DB インスタンスを選択します。 2. 接続セクションで、ターゲット VPC に関連付けられているサブネットグループを選択します。 3. 変更をスケジュールセクションで、すぐに適用を選択します。 ターゲット VPC への移行が完了すると、ターゲット VPC のデフォルトのセキュリティグループが Amazon RDS DB インスタンスに割り当てられます。DB インスタンスに必要なインバウンドルールとアウトバウンドルールを使用して、その VPC の新しいセキュリティグループを設定できます。 または、AWS コマンドラインインターフェイス (AWS CLI) を使用して、新しい VPC セキュリティグループ ID を明示的に指定して、ターゲット VPC に移行します。以下に例を示します。 `aws rds modify-db-instance \`	管理者

タスク	説明	必要なスキル

	`--db-instance-identifier testrds \`
	`--db-subnet-group-name new-vpc-subnet-group \`
	`--vpc-security-group-ids sg-idxxxx \`
	`--apply-immediately`	

別の AWS アカウントに移行する

タスク	説明	必要なスキル
ターゲット AWS アカウントに新しい VPC とサブネットグループを作成します。	1. 「Amazon VPC コンソール」で、必要なプロパティと IP アドレス範囲を使用して新しい VPC を作成します。詳細な手順については、「Amazon VPC のドキュメント」 を参照してください。 2. 「Amazon VPC ドキュメント」の指示に従って、新しい VPC のサブネットを作成します。 3. 「Amazon RDS コンソール」で DB サブネットグループを作成します。手順については、「Amazon RDS ドキュメント」 を参照してください。	管理者

タスク	説明	必要なスキル
データベースの手動スナップショットを共有し、ターゲットアカウントと共有します。	1. 「Amazon RDS ドキュメント」の指示に従って、ソースデータベースの手動スナップショットを作成します。 2. ターゲットアカウント ID を指定して、スナップショットをターゲット AWS アカウントと共有します。手順については、DB スナップショットを他のアカウントと共有することに関する「re: POST の記事」を参照してください。	管理者
新しいAmazon RDS DB インスタンスを起動します。	ターゲット AWS アカウントの共有スナップショットから新しい Amazon RDS DB インスタンスを起動します。手順については、「Amazon RDS ドキュメント」を参照してください。	管理者

関連リソース

- 「Amazon VPCドキュメント」
- 「Amazon RDS ドキュメント」
- 「VPC を Amazon RDS DB インスタンスに変更する方法」 (AWS re: POST の記事)
- 「Amazon RDS リソースの所有権を別の AWS アカウントに移すにはどうすればよいですか?」 (AWS re: POST の記事)
- 「手動の Amazon RDS DB スナップショットまたは Aurora DB クラスタースナップショットを別の AWS アカウントと共有する方法を教えてください。」 (AWS re: POST の記事)
- AWS DMS のドキュメント

Amazon RDS for Oracle DB インスタンスを別の VPC へ移行する

作成者: Pinesh Singal (AWS)

概要

この移行パターンは、Amazon Relational Database Service (Amazon RDS) for Oracle データベース (DB) インスタンスを、ある仮想プライベートクラウド (VPC) から同じ Amazon Web Services (AWS) アカウントにある別の VPC に移行するステップバイステップのガイダンスを提供します。たとえば、データベースと Amazon Elastic Compute Cloud (Amazon EC2) アプリケーションサーバーを同じ VPC に配置する必要がある場合は、このパターンを使用できます。

このパターンでは、トランザクション数が多い数テラバイトの Oracle ソースデータベースにダウンタイムをほとんど発生させないオンライン移行戦略を説明します。

Amazon RDS for Oracle DB インスタンスを別の VPC に移動するには、Amazon RDS サブネットグループを変更する必要があります。このサブネットグループは、新しい VPC と必要なサブネットで事前設定する必要があります。あるネットワークから別のネットワークへ VPC を変更中、Amazon RDS インスタンスを再起動するため、進行中はデータベースにアクセスできなくなります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- プライベートサブネットがある 2 つの VPC
- インバウンドとアウトバウンドのセキュリティグループが設定された Amazon RDS for Oracle データベースインスタンス (稼働中)

制限

- 複数のアベイラビリティーゾーン (マルチ AZ) にまたがる DB インスタンスはサポートされていません。ただし、このパターンはこの制限を回避する方法を提供します。
- リードレプリカがオンになっている間は DB インスタンスは移行できません。
- 新しい VPC のサブネットグループは、データベースと同じアベイラビリティーゾーンにある必要があります。
- DB を別の VPC に移動するとデータベースが再起動し、アプリケーションが数分間停止するため、移行は予定されているメンテナンス期間またはトラフィックが少ない時間帯に行う必要があります。

製品バージョン

- Amazon RDS for Oracle DB インスタンス、12.1.0.2 以降

アーキテクチャ

ソーステクノロジースタック

- VPC の Amazon RDS for Oracle 12.1.0.2.v22 DB インスタンス
- 別のルートテーブルに設定された VPC
- VPC に設定された Amazon RDS サブネットグループ
- Amazon RDS オプショングループ (必要な場合)

ターゲットテクノロジースタック

- 別の VPC でバージョン 12.1.0.2.v22 の Amazon RDS for Oracle データベースインスタンス
- 別のルートに設定された Amazon VPC
- 新しい VPC に設定された Amazon RDS サブネットグループ
- Amazon RDS オプショングループ (必要な場合)

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、コンソールを使用して Amazon RDS for Oracle DB を、ある VPC のプライベートサブネットから別の VPC のプライベートサブネットに移動する方法を示しています。

1. コンソールを使用して、ソース Amazon RDS for Oracle DB インスタンスを変更します。
2. ターゲット VPC で、サブネットグループを変更し、使用している場合はオプショングループを変更します。

ツール

- 「Amazon RDS」— Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケールをより簡単にするウェブサービスです。リレーショナルデータベース向けに、費用対効果に優れスケーラブルな容量を提供し、一般的

なデータベース管理タスクを管理します。このパターンでは Amazon RDS for Oracle を使用します。

エピック

既存の VPC の Amazon RDS for Oracle データベースの設定を変更する

タスク	説明	必要なスキル
サブネットグループを作成します。	Amazon RDS でサブネットグループを設定します。	AWS 全般
オプショングループを作成します。	(オプション) Amazon RDS でオプショングループを設定します。	AWS 全般
Amazon RDS for Oracle DB インスタンスを変更します。	サブネットグループとオプショングループでデータベースを変更します。	AWS 全般、DBA
必要に応じて Oracle データベースを更新します。	ソース Amazon RDS for Oracle データベースを移行するには、以下の変更をします。 • リードレプリカがある場合は削除します。 • マルチ AZ 機能がオンになっている場合は、オフにします。	AWS 全般

ターゲット VPC で Amazon RDS for Oracle データベースを設定する

タスク	説明	必要なスキル
サブネットグループを作成します。	Amazon RDS で、新しい VPC のサブネットとデータベース	AWS 全般

タスク	説明	必要なスキル
	のアベイラビリティーゾーンを使用してサブネットグループを設定します。	
オプショングループを作成します。	(オプション) Amazon RDS でオプショングループを設定します。	AWS 全般
Amazon RDS for Oracle データベースを変更します。	新しい VPC の新しいサブネットグループとオプショングループでデータベースを変更します。これらの変更はすぐに適用することも、メンテナンス期間中に適用することもできます。 変更は完了までに数分かかることがあります。変更中は、次のようなステータスの変化が見られます。 • vpc に移動中 • 拡張モニタリングを設定中 • 変更中 • 利用可能 変更により、新しい VPC のデフォルトのセキュリティグループがアタッチされます。Amazon RDS for Oracle の必要に応じて新しいセキュリティグループをアタッチします。	AWS 全般、DBA

タスク	説明	必要なスキル
必要に応じて、Amazon RDS for Oracle データベースを更新します。	新しい VPC 内のターゲット Amazon RDS for Oracle データベースに移行後、必要に応じて、以下の変更をします。 • リードレプリカがソースデータベースに存在する場合は有効にします。 • ソースデータベースでマルチ AZ 機能が有効になっていた場合は有効にします。	AWS 全般
アプリケーションの接続性をテストします。	任意のアプリケーションからデータベースの接続性テストを実行します。新しい VPC の変更された Amazon RDS for Oracle DB が接続され、アプリケーションからアクセスできることを確認します。	アプリ所有者

関連リソース

- Amazon VPC ドキュメント
- VPC とサブネット
- VPC 内の DB インスタンスを使用する
- 「Amazon RDS ドキュメント」
- Oracle on Amazon RDS
- Amazon RDS コンソール
- VPC を Amazon RDS DB インスタンス用に変更する方法

Amazon Redshift クラスターを中国の AWS リージョンに移行する

作成者: Jing Yan (AWS)

概要

このパターンでは、Amazon Redshift クラスターを別の AWS リージョンから中国の AWS リージョンに移行するステップバイステップのアプローチを提供します。

このパターンは、SQL コマンドを使用してすべてのデータベースオブジェクトを再作成し、UNLOAD コマンドを使用してこのデータを Amazon Redshift からソースリージョンの Amazon Simple Storage Service (Amazon S3) バケットに移動します。その後、データは中国の AWS リージョンの S3 バケットに移行されます。COPY コマンドは、S3 バケットからデータをロードし、ターゲットの Amazon Redshift クラスターに転送します。

Amazon Redshift は現在、中国の AWS リージョンへのスナップショットコピーなどのクロスリージョン機能をサポートしていません。このパターンはその制限を回避する方法を提供します。このパターンの手順を逆にして、中国の AWS リージョンから別の AWS リージョンにデータを移行することもできます。

前提条件と制限

前提条件

- 中国リージョンと中国以外の AWS リージョンの両方のアクティブな AWS アカウント
- 中国リージョンと中国以外の AWS リージョンの両方にある既存の Amazon Redshift クラスター

制限

- これはオフライン移行です。つまり、ソース Amazon Redshift クラスターは移行中に書き込み操作を実行できません。

アーキテクチャ

ソーステクノロジースタック

- 中国以外の AWS リージョンにある Amazon Redshift クラスター

ターゲットテクノロジースタック

- 中国の AWS リージョンにある Amazon Redshift クラスター

ターゲットアーキテクチャ

ツール

ツール

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) は、スケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。Amazon S3 を使用して、Amazon Redshift からのデータを保存できます。また、S3 バケットから Amazon Redshift にデータをコピーできます。
- Amazon Redshift - Amazon Redshift は、クラウド内のフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。
- psql — psql は PostgreSQL へのターミナルベースのフロントエンドです。

エピック

ソースリージョンで移行の準備をする

タスク	説明	必要なスキル
ソースリージョンで EC2 インスタンスを起動して設定します。	AWS マネジメントコンソールにサインインし、Amazon Elastic Compute Cloud (Amazon EC2) コンソールを開きます。現在のリージョンは、画面トップのナビゲーションバーに表示されます。このリージョンは中国の AWS リージョンにはできません。Amazon EC2 コンソールダッシュボードから、[Launch instance (インスタンスを起動)] を選択し、EC2 イ	DBA、開発者

タスク	説明	必要なスキル
	ンスタンスを作成して設定します。重要: インバウンドルールの EC2 セキュリティグループが、ソースマシンから TCP ポート 22 への無制限のアクセスを許可していることを確認します。EC2 インスタンスを起動して設定する方法については、「関連リソース」セクションを参照してください。	
psql ツールをインストールします。	PostgreSQL をダウンロードしてインストールします。psql ツールは、Amazon Redshift から提供されるのではなく PostgreSQL とともにインストールされます。psql の使用と PostgreSQL ツールのインストールの詳細については、「関連リソース」セクションを参照してください。	DBA

タスク	説明	必要なスキル
Amazon Redshift クラスターの詳細を記録します。	Amazon Redshift コンソールを開き、ナビゲーションペインの [Clusters (クラスター)] を選択します。次に、リストから Amazon Redshift クラスター名を選択します。[Database configurations (データベース設定)] セクションの [Properties (プロパティ)] タブで、「データベース名」と 「ポート」を記録します。[Connection details (接続詳細)] セクションを開き、「endpoint:<port>/<databasename>」形式の「エンドポイント」を記録します。重要: インバウンドルールの Amazon Redshift セキュリティグループが EC2 インスタンスから TCP ポート 5439 への無制限のアクセスを許可していることを確認します。	DBA

タスク	説明	必要なスキル
psql を Amazon Redshift クラスターに接続します。	コマンドプロンプトで、「psql -h <endpoint> -U <userid> -d <databasename> -p <port>」コマンドを実行して接続情報を指定します。psql パスワードプロンプトで、「<userid>」ユーザーのパスワードを入力します。これで Amazon Redshift クラスターに接続されるため、コマンドをインタラクティブに入力できます。	DBA
S3 バケットを作成する。	Amazon S3 コンソールを開き、S3 バケットを作成して Amazon Redshift からエクスポートされたファイルを保持します。S3 バケットを作成する手順については、「関連リソース」セクションを参照してください。	DBA、AWS 全般

タスク	説明	必要なスキル
データのアンロードをサポートする IAM ポリシーを作成します。	AWS Identity and Access Management (IAM) コンソールを開き、[Policies (ポリシー)] を選択します。[Create policy (ポリシーの作成)] を選択し、[JSON] タブを選択します。「追加情報」セクションからデータをアンロードする IAM ポリシーをコピーして貼り付けます。重要:「s3_bucket_name」は、お使いの S3 バケット名に置き換えてください。[Review policy (ポリシーの確認)] を選択し、ポリシーの名前と説明を入力します。[Create policy (ポリシーの作成)] を選択します。	DBA

タスク	説明	必要なスキル
IAM ロールを作成し、Amazon Redshift の UNLOAD 操作を許可します。	IAM コンソールを開き、[Roles（ロール）] を選択します。[Create role（ロールの作成）] を選択し、[Select type of trusted entity（信頼できるエンティティのタイプの選択）] で [AWS service（AWS サービス）] を選択します。サービスの [Redshift] を選択し、[Redshift – Customizable（Redshift — カスタマイズ可能）] を選択してから、[Next（次へ）] を選択します。前に作成した [Unload policy（ポリシーをアンロード）] を選択し、[Next（次へ）] を選択します。「ロール名」を入力し、[Create role（ロールの作成）] を選択します。	DBA

タスク	説明	必要なスキル
IAM ロールを Amazon Redshift クラスターに関連付けます。	Amazon Redshift コンソールを開き、[Manage IAM roles (IAM ロールの管理)] を選択します。ドロップダウンメニューから、[Available roles (利用可能なロール)] を選択し、前に作成したロールを選択します。[Apply changes (変更の適用)] を選択します。[Manage IAM roles (IAM ロールの管理)] の IAM ロールの [Status (ステータス)] が [In-sync (同期中)] と表示されたら、UNLOAD コマンドを実行できます。	DBA
Amazon Redshift クラスターへの書き込み操作を停止します。	移行が完了するまで、移行元の Amazon Redshift クラスターへのすべての書き込み操作を停止することを忘れないでください。	DBA

ターゲットリージョンの移行を準備する

タスク	説明	必要なスキル
ターゲットリージョンで EC2 インスタンスを起動して設定します。	中国のリージョン (北京または寧夏) の AWS マネジメントコンソールにサインインします。Amazon EC2 コンソールで、[Launch instance (インスタンスを起動)] を選択し、EC2 インスタンスを作成	DBA

タスク	説明	必要なスキル
	して設定します。重要: インバウンドルールのAmazon EC2 セキュリティグループが、ソースマシンから TCP ポート 22 への無制限のアクセスを許可していることを確認します。EC2 インスタンスを起動して設定する方法の詳細については、「関連リソース」セクションを参照してください。	
Amazon Redshift クラスターの詳細を記録します。	Amazon Redshift コンソールを開き、ナビゲーションペインの [Clusters (クラスター)] を選択します。次に、リストから Amazon Redshift クラスター名を選択します。[Database configurations (データベース設定)] セクションの [Properties (プロパティ)] タブで、「データベース名」と 「ポート」を記録します。[Connection details (接続詳細)] セクションを開き、「endpoint:<port>/<databasename>」形式の「エンドポイント」を記録します。重要:インバウンドルールの Amazon Redshift セキュリティグループが EC2 インスタンスから TCP ポート 5439 への無制限のアクセスを許可していることを確認します。	DBA

タスク	説明	必要なスキル
psql を Amazon Redshift クラスターに接続します。	コマンドプロンプトで、「psql -h <endpoint> -U <userid> -d <databasename> -p <port>」コマンドを実行して接続情報を指定します。psql パスワードプロンプトで、「<userid>」ユーザーのパスワードを入力します。これで Amazon Redshift クラスターに接続されるため、コマンドをインタラクティブに入力できます。	DBA
S3 バケットを作成する。	Amazon S3 コンソールを開き、S3 バケットを作成して Amazon Redshift からエクスポートされたファイルを保持します。このストーリーやその他のストーリーに関するヘルプは、「関連リソース」セクションを参照してください。	DBA

タスク	説明	必要なスキル
データのコピーをサポートする IAM ポリシーを作成します。	IAM コンソールを開き、[Policies（ポリシー）] を選択します。[Create policy（ポリシーの作成)] を選択し、[JSON] タブを選択します。「追加情報」セクションからデータをコピーする IAM ポリシーをコピーして貼り付けます。重要:「s3_bucket_name」は、お使いの S3 バケット名に置き換えてください。[Review policy（ポリシーの確認）] を選択し、ポリシーの名前と説明を入力します。[Create policy（ポリシーの作成）] を選択します。	DBA

タスク	説明	必要なスキル
IAM ロールを作成し、Amazon Redshift の COPY 操作を許可します。	IAM コンソールを開き、[Roles（ロール）] を選択します。[Create role（ロールの作成）] を選択し、[Select type of trusted entity（信頼できるエンティティのタイプの選択）] で [AWS service（AWS サービス）] を選択します。サービスの [Redshift] を選択し、[Redshift – Customizable（Redshift — カスタマイズ可能）] を選択してから、[Next（次へ）] を選択します。前に作成した「コピー」ポリシーを選択し、[Next（次へ）] を選択します。「ロール名」を入力し、[Create role（ロールの作成）] を選択します。	DBA

タスク	説明	必要なスキル
IAM ロールを Amazon Redshift クラスターに関連付けます。	Amazon Redshift コンソールを開き、[Manage IAM roles (IAM ロールの管理)] を選択します。ドロップダウンメニューから、[Available roles (利用可能なロール)] を選択し、前に作成したロールを選択します。[Apply changes (変更の適用)] を選択します。[Manage IAM roles (IAM ロールの管理)] の IAM ロールの [Status (ステータス)] が [In-sync (同期中)] と表示されたら、「COPY」コマンドを実行できます。	DBA

移行を開始する前に、ソースデータとオブジェクト情報を確認する

タスク	説明	必要なスキル
ソース Amazon Redshift テーブルの行を確認します。	「追加情報」セクションのスクリプトを使用して、ソース Amazon Redshift テーブルの行数を確認して記録します。UNLOAD スクリプトと COPY スクリプトでは、データを均等に分割することを忘れないでください。これにより、各スクリプトでカバーされるデータ量のバランスが取れるため、データのアンロードとロードの効率が向上します。	DBA

タスク	説明	必要なスキル
ソース Amazon Redshift クラスター内のデータベースオブジェクトの数を確認します。	「追加情報」セクションのスクリプトを使用して、ソース Amazon Redshift クラスター内のデータベース、ユーザー、スキーマ、テーブル、ビュー、およびユーザー定義関数 (UDF) の数を確認および記録します。	DBA
移行前に SQL ステートメントの結果を検証します。	データ検証用の SQL ステートメントの中には、実際のビジネス状況やデータ状況に応じてソートする必要があるものがあります。これは、インポートされたデータが一貫していて正しく表示されているかを検証するためです。	DBA

データとオブジェクトをターゲットリージョンに移行する

タスク	説明	必要なスキル
Amazon Redshift DDL スクリプトを生成します。	「追加情報」セクションの「Amazon Redshift をクエリする SQL ステートメント」セクションのリンクを使用して、データ定義言語 (DDL) スクリプトを生成します。これらの DDL スクリプトには、「ユーザー作成」、「スキーマの作成」、「ユーザーへのスキーマの特権」、「テーブル/ビューの作成」、「ユーザーへのオブジェクトの特権」、	DBA

タスク	説明	必要なスキル
	および「関数の作成」クエリが含まれている必要があります。	
ターゲットリージョンの Amazon Redshift クラスターにオブジェクトを作成します。	中国の AWS リージョンで AWS コマンドラインインターフェイス (AWS CLI) を使用して DDL スクリプトを実行します。これらのスクリプトは、ターゲットリージョンの Amazon Redshift クラスターにオブジェクトを作成します。	DBA
Amazon Redshift クラスターのソースデータを S3 バケットにアンロードします。	UNLOAD コマンドを実行して、ソースリージョンの Amazon Redshift クラスターから S3 バケットにデータをアンロードします。	DBA、開発者
ソースリージョン S3 バケットデータをターゲットリージョン S3 バケットに転送します。	ソースリージョン S3 バケットからターゲットの S3 バケットにデータを転送します。「$ aws s3 sync」コマンドは使用できないため、「関連リソース」セクションの「AWS リージョンから中国の AWS リージョンへ Amazon S3 データを転送する」記事で説明されているプロセスを使用してください。	開発者

タスク	説明	必要なスキル
ターゲット Amazon Redshift クラスターにデータをロードします。	ターゲットリージョンの psql ツールで、COPY コマンドを実行して、S3 バケットからターゲット Amazon Redshift クラスターにデータをロードします。	DBA

移行後にソースリージョンとターゲットリージョンのデータを検証する

タスク	説明	必要なスキル
ソーステーブルとターゲットテーブルの行数を検証して比較します。	ソースリージョンとターゲットリージョンのテーブル行数を検証して比較し、すべてが移行されていることを確認します。	DBA
ソースデータベースオブジェクトとターゲットデータベースオブジェクトの数を検証して比較します。	ソースリージョンとターゲットリージョン内のすべてのデータベースオブジェクトを検証して比較し、すべてが移行されていることを確認します。	DBA
ソースリージョンとターゲットリージョンの SQL スクリプトの結果を検証して比較します。	移行前に準備した SQL スクリプトを実行します。データを検証して比較し、SQL の結果が正しいことを確認します。	DBA
ターゲット Amazon Redshift クラスターのすべてのユーザーのパスワードをリセットします。	移行が完了し、すべてのデータが確認されたら、中国の AWS リージョンにある Amazon Redshift クラスターのすべてのユーザーパスワー	DBA

タスク	説明	必要なスキル
	ドをリセットする必要があります。	

関連リソース

- Amazon S3 データを AWS リージョンから中国の AWS リージョンに転送する
- S3 バケットを作成する
- Amazon Redshift ユーザーパスワードをリセットする
- psql ドキュメント

追加情報

データをアンロードする IAM ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::s3_bucket_name"]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:DeleteObject"],
      "Resource": ["arn:aws:s3:::s3_bucket_name/*"]
    }
  ]
}
```

データをコピーする IAM ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::s3_bucket_name"]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::s3_bucket_name/*"]
    }
  ]
}
```

Amazon Redshift をクエリする SQL ステートメント

```
##Database

select * from pg_database where datdba>1;

##User

select * from pg_user where usesysid>1;

##Schema

SELECT n.nspname AS "Name",

  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"

FROM pg_catalog.pg_namespace n

WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'

ORDER BY 1;

##Table

select count(*) from pg_tables where schemaname not in
 ('pg_catalog','information_schema');

select schemaname,count(*) from pg_tables where schemaname not in
 ('pg_catalog','information_schema') group by schemaname order by 1;

##View
```

```
SELECT

    n.nspname AS schemaname,c.relname AS
 viewname,pg_catalog.pg_get_userbyid(c.relowner) as "Owner"

FROM

    pg_catalog.pg_class AS c

INNER JOIN

    pg_catalog.pg_namespace AS n

    ON c.relnamespace = n.oid

WHERE relkind = 'v' and n.nspname not in ('information_schema','pg_catalog');

##UDF

SELECT

   n.nspname AS schemaname,

   p.proname AS proname,

   pg_catalog.pg_get_userbyid(p.proowner) as "Owner"

FROM pg_proc p

LEFT JOIN pg_namespace n on n.oid = p.pronamespace

WHERE p.proowner != 1;
```

DDL ステートメントを生成する SQL スクリプト

- Get_schema_priv_by_user スクリプト
- Generate_tbl_ddl スクリプト
- Generate_view_ddl
- Generate_user_grant_revoke_ddl
- Generate_udf_ddl

VMware HCX を使用して VMware Cloud on AWS ワークロードを移行

作成者: Deepak Kumar (AWS)、Derek Cox (AWS)、Himanshu Gupta (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

このパターンでは、VMware ハイブリッドクラウド拡張 (HCX) を使用して、基盤となるプラットフォームを変更せずに、オンプレミスの VMware 環境から VMware Cloud on AWS にワークロードを移行する方法を説明しています。VMware HCX は、オンプレミスデータセンターとクラウドサーバーの両方で、移行を効率化し、ワークロードのバランスを調整し、データを保護し、ディザスタリカバリプロセスを最適化します。このパターンでは、HCX のインストール、設定、アップグレード、アンインストールの手順について説明しています。

以下をサポートしています。

- 古いバージョンの VMware vSphere — HCX は、仮想マシン (VM) を古いバージョンの vSphere から VMware Cloud on AWS に移行することを支援します。ホストは自動的に更新および修復されるため、移行に準備する時間のかかる更新を行う必要がありません。
- 一括移行 — HCX を WAN 最適化サービスとともに使用すると、ダウンタイムなしで多数の VM を 1 ステップで移行し、オンプレミスネットワークをクラウドに拡張できます。
- 異種ネットワーク環境 — 現在のネットワーク (vSphere、NSX、VXLAN、NSX-T など) によって、移行の複雑さが決まります。HCX は、複雑な手順を必要とせずに、ネットワークアプリケーションの基礎を抽出し、現在のネットワークをクラウドに拡張します。
- 遅いネットワーク速度 — 移行には通常 250 Mbps を超える接続速度が必要です。HCX では、100 Mbps 程度というはるかに低い速度でワークロードを移行できます。

HCX には、3 種類のクラウド移行をサポートします。

- ハイブリディティ (データセンターの拡張) — 既存のオンプレミスの VMware SDDC を クラウドに拡張して、設置面積の拡大、オンデマンド容量、テスト/開発環境、仮想デスクトップを提供している。

- クラウドからの排除 (データセンターのインフラストラクチャの更新) — データセンターを統合し、 クラウドに完全に移行している。これにはデータセンターのコロケーションやリース終了のシナリオの処理を含む。
- アプリケーション固有 — 個々のアプリケーションを クラウドに移動して、特定のビジネスニーズを満たしている。

HCX を使用して、オンプレミス環境と VMware Cloud on AWS の間でワークロードを双方向に移行できます。HCX には、ソースロケーションとターゲットロケーションの間でワークロードを移行する方法が複数用意されています。

- HCX コールド移行は、オフラインの VM を移行します。この方法は、かなりのダウンタイムが必要なため、パワーオフ状態の VM に適しています。
- HCX VMotion は VMware vMotion プロトコルを使用して仮想マシンを移動します。HCX vMotion ではダウンタイムなしの移行が可能ですが、一度に移行できる VM は 1 台だけです。
- HCX 一括移行では、VMware vSphere のレプリケーションプロトコルを使用して仮想マシンを移行先に移動します。複数の仮想マシンを並行で移行し、スイッチオーバーをスケジュールできます。ダウンタイムはサーバーの再起動と同等で、すべての仮想マシンのスイッチオーバーが並行で行われます。
- HCX レプリケーションアシスト vMotion (RAV) は HCX 一括移行と HCX vMotion を組み合わせたものです。並行移行、スケジューリング、ゼロダウンタイムを実現します。
- HCX OS Assisted Migration は、オンプレミスで複数のハイパーバイザーと非 vSphere VM を使用する場合、複数の VM をまとめて移行することを支援します。HCX OS Assisted Migration は、オンプレミスから VMware Cloud on AWS への移行には無料ですが、2 つのオンプレミス環境間、またはオンプレミスから他のクラウドプロバイダーに移行する場合、追加のライセンスが必要です。

前提条件と制限

前提条件

- vmware.com の VMware コンソールにアクセスするための VMware アカウント。
  - HCX に対して、次のファイアウォールポートが必要です。

ソース	デスティネーション	ポート

オンプレミスのHCX マネージャーとアプライアンス IP	VMware Cloud on AWS の HCX マネージャーとアプライアンス IP	UDP 500、UDP 4500、ICMP
オンプレミスのHCX マネージャーとアプライアンス IP	connect.hcx.vmware.com | hybridity-depot.vmware.com	TCP 443
オンプレミスのHCX マネージャーとアプライアンス IP	HCX クラウド URL	TCP 443

オンプレミスネットワークに内部ファイアウォールがある場合、データセンター内でローカルにいくつかのポートを許可する必要があります。HCX のポート要件のフルリストについては、VMware HCX のドキュメント を参照してください。

- HCX を設定するには、ドメインネームシステム (DNS) IP、vCenter 完全修飾ドメイン名 (FQDN)、NTP サーバの FQDN、シングルサインオン (SSO) ユーザー、および同様の情報が必要です。導入に遅れが発生しないように、これらの詳細を事前に収集します。

制約事項

ネットワーク拡張アプライアンスを使用して、オンプレミス環境と VMware Cloud on AWS の間で最大 8 つのネットワークを拡張できます。HCX サービスの制限の詳細なリストについては、VMware HCX のドキュメント を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス VMware ワークロード

ターゲットテクノロジースタック

- VMware Cloud on AWS

ツール

ツール

- VMware Cloud on AWS は、オンプレミスの VMware vSphere ベースの環境を に移行して拡張することができる、 と VMware が共同で設計したサービスです。
- VMware ハイブリッドクラウドエクステンション (HCX) は、基盤となるプラットフォームを変更せずに、オンプレミスの VMware 環境から VMware Cloud on AWS にワークロードを移行するための VMware ユーティリティです。

エピック

HCX をデプロイ

タスク	説明	必要なスキル
VMware Cloud on AWS で HCX サービスを有効に	1. VMware Cloud on AWS コンソール にログインします。 2. SDCC に移動し、詳細を表示を選択します。 3. アドオン タブを選択します。 4. IDE を開くを選択します。 5. HCX をデプロイ を選択して確認します。HCX のデプロイが開始します。	ネットワーク管理者、システム管理者
HCX アクティベーションキーを生成します。	1. VMware Cloud on AWS コンソールで。 2. SDCC に移動し、詳細を表示を選択します。 3. アドオン タブを選択します。 4. HCXを開くを選択し、アクティベーションキーを選択します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	5. アクティベーションキーの作成を選択し、キーをコピーします。	

タスク	説明	必要なスキル
クラウド SDDC の HCX のファイアウォールルールを追加します。	HCXマネージャーをデプロイした後、ファイアウォールルールを設定してオンプレミス環境と SDDC 間の通信を有効にする必要があります。2 つのファイアウォールルールを作成する必要があります。1 つはインバウンド通信用、もう 1 つはアウトバウンド通信用です。 1. VMware Cloud on AWS コンソール で SDDC を選択し、ネットワークとセキュリティにナビゲーションします。 2. ゲートウェイファイアウォールを選択し、管理ゲートウェイタブを選択します。 3. ルールを追加 を選択し、アウトバウンドルールを作成します。 a. ルール名を入力します。 b. ソースを編集し、HCX. を選択します。 c. 宛先を編集し、HCX にアクセスできるオンプレミス IP とサブネットを指定します。 d. サービスの場合、すべてを選択します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	e. アクション で、インストール を選択します。 f. 発行 を選択します。 4. ルールを追加を選択し、インバウンドルールを作成します。 a. ルール名を入力します。 b. ソースを編集し、HCX にアクセスできるオンプレミス IP とサブネットを指定します。 c. 宛先を編集し、HCX を選択します。 d. サービスには、SSH、HTTPS、TCP (9443)、および ICMP を選択します。 e. アクション で、インストール を選択します。 f. 発行 を選択します。	

タスク	説明	必要なスキル
HCX マネージャーをオンプレミスにインストールします。	1. クラウド vCenter にログインし、メニューから HCX に移動します。 2. HCX ダッシュボードで、管理、システムアップデートを選択します。 3. VMware HCX Connector のダウンロードリンクをリクエストし、オンプレミスの OVA ファイルをダウンロードします。 4. オンプレミスの vCenter にログインし、ダウンロードした OVA ファイルを使用して OVF テンプレートをデプロイします。 5. テンプレートのデプロイ時に、プロンプトが表示された後、固定 IP、NTP、DNS、DNS 検索リスト、およびその他の詳細を指定します。 6. すべての詳細を確認して HCX マネージャーのデプロイを完了します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
HCX マネージャーをオンプレミスで設定します。	1. HCX マネージャーを https://<HCX_Manager_IP>:9433 ブラウザーで開きます。 2. デプロイ時に指定したユーザー名とパスワードを使用してログインします。 3. 以前に作成したアクティベーションキーを入力し、アクティブにするを選択して HCX インスタンスをアクティブにします。 4. 確認を選択して、次のステップに進みます。 5. オンプレミスデータセンターの位置を選択し、続行を選択します。 6. システム名にホスト名を入力し、続行を選択して、アクティブにすることを完了します。 7. 情報を入力して vCenter 接続を設定します。 8. SSO/PSC の詳細を設定するための情報を入力します。 9. 再起動を選択して、変更を有効にします。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
サイトペアリングを設定します。	クラウドとオンプレミスで HCX を設定した後、次の手順に従って両者間のサイトペアリングを設定します。 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. 左側のナビゲーションペインで、サイトペアリングを選択し、リモートサイトに接続を選択します。 3. リモートサイトに接続ダイアログボックスで、HCX クラウド URL と認証情報を追加し、接続を選択します。 サイトのペアリングが完了する場合、サイトペアリングダッシュボードには、接続されたオンプレミスとクラウド SDDC が表示されます。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
ネットワークプロファイルの作成	ネットワークプロファイルは、ネットワークのレイヤー 3 コンポーネントを抽象化したものです。このプロファイルでは、コンピュートプロファイルを作成するための前提条件です。 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. インターコネクトを選択し、ネットワークプロファイルタブを選択し、ネットワークプロファイルの作成を選択します。 3. ネットワークプロファイルを設定します。 a. vCenter サーバを選択します。 b. 「ネットワーク」を選択します。 c. プロファイル名を追加します。 d. IP プール、プレフィックス長、ゲートウェイ、DND、MTU を指定します。 e. 作成を選択します。 4. オンプレミスでネットワークプロファイルを作成するプロセスに従います。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
コンピュートプロファイルを作成します。	コンピュートプロファイルでは HCX のネットワーク、ストレージ、コンピュートの詳細で構成されています。 HCX は、サービスメッシュの作成中に HCX アプライアンスを作成する際にこれらの設定を使用します。 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. インターコネクトを選択し、コンピュートプロファイルタブを選択し、コンピュートプロファイルの作成を選択します。 3. コンピュートプロファイルの名前を指定します。 4. 有効にする HCX サービスを選択し、続行を選択します。 5. サービスリソースを選択します。クラスターが複数ある場合、HCX サービスを有効にするクラスターをそれぞれ選択し、続行を選択します。 6. HCX アプライアンスをデプロイするためのコンピュートリソースとストレージリソースを選択し、続行を選択します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	7. vCenter ホストと ESXi ホストの管理インターフェイスへのアクセスに使用できる管理ネットワークプロファイルを選択し、続行を選択します。 8. リモートサイトのインターコネクトアプライアンスへのアクセスに使用できるアップリンクネットワークプロファイルと、リモートサイトアプライアンスがローカルインターコネクトアプライアンスへの接続に使用できるアップリンクネットワークプロファイルを選択し、続行を選択します。 9. vMotion ネットワークプロファイルを選択し、続行を選択します。 10.vSphere レプリケーションネットワークプロファイルを選択し、続行を選択します。 11.ネットワーク拡張に適した分散スイッチを選択し、続行を選択します。 12.WAN 接続と LAN 接続で開く必要があるポートをすべて確認し、続行を選択します。 13.完了を選択してプロジェクトを作成します。	

タスク	説明	必要なスキル
	14.クラウドサイトでコンピュートプロファイルを作成するステップに従います。	

タスク	説明	必要なスキル
サービスメッシュの作成	サービスメッシュは、オンプレミスサイトとクラウドサイトの両方に HCX サービス設定を提供します。サービスメッシュを作成すると、両方のサイトに HCX インターコネクト仮想アプライアンスの導入が開始されます。ソースサイトでインターコネクトサービスを作成する必要があります。 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. インターコネクトを選択し、サービスメッシュタブを選択し、サービスメッシュの作成を選択します。 3. サービスメッシュを作成するソースサイトとターゲットサイトを選択し、続行を選択します。 4. 前に作成したソースサイトとターゲットサイトのコンピュートプロファイルを選択し、続行を選択します。 5. 有効にする HCX サービスを選択し、続行を選択します。 6. ソースサイトとターゲットサイトの両方のアップリンクプロファイルを選択し、続行を選択します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	7. リソースとネットワークを確認し、続行を選択します。 8. サービスメッシュの名前を入力し、完了を選択します。 サービスメッシュのデプロイが開始されます。サービスメッシュのタスクタブで進行状況を確認できます。デプロイが完了すると、サービスメッシュに対して有効にしたすべての HCX サービスのステータスが表示されます。	

HCX を使用してネットワークを拡張

タスク	説明	必要なスキル
ネットワークエクステンションを作成します。	HCX ネットワーク拡張機能を使用して、クラウド SDDC HCX サイトに L2 ネットワーク拡張を作成し、リモートネットワークとソースネットワークをブリッジできます。 これにより、同じ IP アドレスを維持したまま、サーバーをオンプレミスから VMware Cloud on AWS に移行できます。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. サービス、ネットワーク拡張を選択します。 3. ネットワークの拡張、またはネットワーク拡張の作成を選択します。 4. 適切なサービスメッシュ、分散ポートグループ、または NSX 論理スイッチを選択します。 5. ゲートウェイ IP アドレスを指定し、提出を選択します。 ネットワーク拡張が完了すると、システムに拡張完了と表示されます。	

HCX を使用してレプリケーションジョブを設定します。

タスク	説明	必要なスキル
レプリケーションの設定	HCX を使用して VM を複製するには： 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	2. 移行を選択し、移行タブを選択します。 3. モビリティグループ名を入力し、移行する VM を選択して、追加を選択します。 4. ターゲットのコンピュートコンテナ、ストレージフォルダー、移行タイプ (コールド、バルク、RAV、VMotion)、および切り替えスケジュールを選択します。 5. 検証を選択し、検証が完了するのを待ってから、移動を選択してレプリケーションを開始します。	

HCX のアップグレード

タスク	説明	必要なスキル
推奨事項と手順をレビューします。	大規模な移行プロジェクトは 6 か月から 8 か月、場合によってはそれ以上かかることもあります。VMware は、ソフトウェア修正、セキュリティ更新、バグ修正で構成される HCX アップデートを定期的に公開しています。セキュリティの脆弱性を排除し、新しい機能を活用するために、HCX とアプライアンスを最新の状態に保つことをお勧めします。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	Note 現在の HCX バージョンが最新リリースより 3 バージョン遅れている場合は、HCX をアップグレードできず、再デプロイする必要があります。 HCX のアップグレードには、次の 3 つのステップで構成されます。 1. オンプレミスとクラウドで HCX マネージャーをバックアップします。 2. オンプレミスとクラウドで HCX マネージャーをアップグレードします。 3. オンプレミスとクラウドでサービスメッシュアプライアンスをアップグレードします。 以下のストーリーで、これらのステップについて詳しく説明します。	

タスク	説明	必要なスキル
HCX クラウドマネージャーをバックアップします。	VMware Cloud on AWS の HCX クラウドマネージャーは VMware によって管理されているため、スナップショットを取ることはできません。HCX クラウドマネージャーをバックアップするには、HCX コンソールからバックアップをダウンロードし、アップグレードが失敗したり、前の段階にロールバックする必要がある場合に備えて、このバックアップを使用して HCX 設定を復元する必要があります。 1. `https://<HCX_cloud manager_ip_or_fqdn >:9433` で HCX クラウドマネージャーにログインします。 2. 管理、トラブルシューティング、バックアップと復元に移動します。 3. バックアップ セクションで、生成 を選択して、バックアップファイルを作成します。 4. 音声ファイルを保存するには、ダウンロード を選択します。 HCX-IX、HCX-NE、HCX-WO などの HCX サービスアプライ	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	アンスは、個別のバックアップを必要としません。	

タスク	説明	必要なスキル
HCX マネージャーをオンプレミスでバックアップします。	オンプレミスの HCX マネージャーをバックアップするには、VM のスナップショットを取る方法と、設定ファイルをバックアップする方法の 2 つがあります。 VM スナップショットを取るには： 1. オンプレミスの vCenter にログインします。 2. VM とテンプレートに移動し、HCX マネージャー VM にナビゲーションします。 3. アクション、スナップショット、スナップショットの作成を選択します。 構成ファイルをバックアップするには： 1. `https://<HCX_cloud manager_ip_or_fqdn >:9433` で HCX クラウドマネージャーにログインします。 2. 管理、トラブルシューティング、バックアップと復元に移動します。 3. バックアップ セクションで、生成 を選択して、バックアップファイルを作成します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	4. 音声ファイルを保存するには、ダウンロード を選択します。 HCX-IX、HCX-NE、HCX-WO などの HCX サービスアプライアンスは、個別のバックアップを必要としません。	

タスク	説明	必要なスキル
オンプレミスとクラウドで HCX マネージャーをアップグレードします。	まず HCX マネージャーをオンプレミスでアップグレードし、次に HCX クラウドマネージャーをアップグレードする必要があります。 HCX マネージャーをオンプレミスでアップグレードするには： 1. vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. システム、管理を選択します。 3. 管理ページでは、システムアップデートタブを選択します。利用可能なサービスアップデートバージョンの列には、保留中のアップデートが表示されます。 4. サービス更新を選択、ダウンロードを選択して、後でアップグレードするアップデートをダウンロードするか、ダウンロード & アップグレードを選択して、アップデートをすぐにダウンロードしてデプロイします。ダウンロードを選択した場合、アップグレードを選択し、準備ができた時、アップグレードを開始することを確認します。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	5. アップグレードが完了すると、以下のようになります。 • HCX マネージャーの管理ページで、最新の HCX バージョンが表示されていることを確認します。 • HCX ダッシュボードで、サイトのペアリングがアップであることを確認します。 • インフラ、サービスメッシュを選択して、すべての HCX サービスが正常であることを確認します。 HCX クラウドマネージャーをアップグレードするステップに従います。	

タスク	説明	必要なスキル
サービスメッシュアプライアンスをアップグレードします。	サービスメッシュが、ソースサイトの HCX マネージャーと別に更新されます。ターゲットサイトのサービスメッシュアプライアンスが自動的に更新されます。 ソースサイトのサービスメッシュアプライアンスをアップグレードするには： 1. vCenter にログインし、HCX ダッシュボードに移動します。 2. インフラストラクチャを選択し、サービスメッシュタブを選択します。 3. 「サービスメッシュアプライアンスの新バージョンが利用可能です」というバナーが表示された場合 「アプライアンスを更新して最新にアップグレード」をクリックし、アプライアンスの更新を選択します。 4. アプライアンスを表示するダイアログボックスでは、1 つ以上のアプライアンスを選択し、OK を選択してアップグレードプロセスを開始します。(サービスメッシュアプライアンスをすべて更新することを推薦します。)	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	5. 各サービスメッシュのタスクを表示を選択してアップグレードを監視します。 6. アップグレードが完了すると、アプライアンスとサービスごとにバナーが表示され、正常に完了したことを確認できます。 7. アップグレード後にトンネルの状態を確認します。 • インフラストラクチャ、サービスメッシュ、アプライアンスを表示を選択します。 • トンネルステータス列にはアップと表示され、画面にはアプライアンスの他の使用可能なバージョンが表示されません。	

HCX ネットワーク拡張機能の削除

タスク	説明	必要なスキル
ネットワークの拡張解除。	前のステップでは、HCX ネットワーク拡張機能を使用して L2 ネットワーク拡張を作成し、オンプレミスから VMware cloud on AWS への移行中に既存の IP を維持する方法を説明します。特定の VLAN のすべての VM を VMware Cloud on AWS に移	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	動した後、オンプレミスサイトとクラウド SDDC 間のネットワークの拡張を解除し、そのネットワークを SDDC でルーティング可能にする必要があります。 遅延を避けるため、すべての VM をオンプレミスから VMware Cloud on AWS に移行したらすぐに、拡張ネットワークを削除することを推薦します。 1. オンプレミスの vCenter にログインし、HCX ダッシュボードにナビゲーションします。 2. HCX ダッシュボードで、サービス、ネットワーク拡張を選択します。 3. 拡張を解除するネットワークを選択し、ネットワークの拡張解除を選択します。 4. 拡張解除後にクラウドネットワークをクラウドエッジゲートウェイに接続を選択します。これにより、クラウド側のネットワークがアクティブになります。	

タスク	説明	必要なスキル
移動したネットワークをクラウド SDDC にルーティングします。	1. VMC ポータル にログインします。 2. SDCC に移動し、詳細を表示を選択します。 3. ネットワークとセキュリティタブを選択します。 4. ネットワークとセキュリティページでは： • ネットワーク、セグメントを選択し、最近拡張解除のサブネットがルーティング可能と表示されていることを確認します。 • インベントリ、グループ,を選択し、そのサブネットをグループに追加します。 • セキュリティ、分散ファイアウォールを選択し、そのグループが目的のファイアウォールルールに含まれていることを確認します。	ネットワーク管理者、システム管理者

WSL のアンインストール

タスク	説明	必要なスキル
前提条件をチェックする	データセンターが終了した場合、移行プロジェクトの終了時に HCX をアンインストールし、そのコンポーネントを削	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	除することを推薦します。ただし、オンプレミスのフットプリントがまだ残っている場合、HCX を稼働させ続けることを推薦します。 HCX をアンインストールする前に、次のことを確認します。 • アクティブな移行がありません。 • ネットワーク拡張がすべて削除されました。	

タスク	説明	必要なスキル
HCX をオンプレミスでアンインストールします。	1. オンプレミスの vCenter にログインし、HCX コンソールにナビゲーションします。 2. サービス、移行を選択して、アクティブな移行がないことを確認します。 3. サービス、ネットワーク拡張を選択し、拡張ネットワークがないことを確認します。 4. インフラ、サイトペアリング、サービスメッシュを選択します。 5. サービスメッシュを特定し、削除 を選択します。 6. 確認プロンプトで、削除を選択します。「サービスメッシュの削除」というバナーがサービスメッシュ画面に表示されます。 7. 他のサービスメッシュについてもステップ 5 ~ 6 を繰り返します。 8. サイトペアリングを解除するには、インフラ、サイトペアリングを選択し、ペアリングされているサイトをすべて切断します。 9. HCX マネージャーアプライアンスを削除します。 a. オンプレミスの vCenter にログインし、HCX	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
	Manager アプライアンスに移動します。 b. アクション、パワー、パワーオフを選択します。 c. アクション、ディスクから削除を選択します。	

タスク	説明	必要なスキル
オンプレミスの vCenter サーバから HCX プラグインを登録解除します。	1. `https://<vc_fqdn>/mob` の vCenter MOB ユーザーインターフェイスにログインします。 2. プロパティセクションの値列の内容を選択します。 3. コンテンツページで ExtensionManager を選択して、登録されているすべてのプラグインが表示されます。 4. `com.vmware.hybridity` 、`com.vmware.hcsp.alarm` 、及び `com.vmware.vca.marketing.ngc.ui` で始まる拡張に注意します。 5. 拡張機能を削除します。 • メソッドセクションで、UnregisterExtension を選択します。 • ステップ 4 で書き留めた拡張キーを入力し、呼び出しのメソッドを選択して拡張機能を削除します。 すべての拡張機能が削除される場合、HCX プラグインが vSphere Web Client から削除されます。	ネットワーク管理者、システム管理者

タスク	説明	必要なスキル
HCX をクラウドからアンインストールします。	クラウドの HCX サービスメッシュとサイトペアリングを削除するには、前述のオンプレミスの HCX のアンインストールで説明した手順を繰り返します。VMware Cloud on AWS では、HCXマネージャーが Vmare で管理されています。vCenter から削除することはできませんが、VMC 管理インターフェイスからアンデプロイすることはできます。 HCX マネージャーをアンデプロイするには： 1. VMC 管理インターフェースにログインします。 2. 組織と SDDC を選択します。 3. アドオンを選択して、HCX がデプロイされたすべての SDDC が表示されます。 4. HCXのデプロイ解除を選択します。	ネットワーク管理者、システム管理者

トラブルシューティング

問題	ソリューション
HCX 一括移行を設定する場合、移行するサーバーを選択できません。	原因：これらのサーバーの移行はキャンセルされましたが、クリーンアップ中に HCX データベースは更新されませんでした。HCX はデータベースの移行がまだ進行中であると見なして

問題	ソリューション
	いるため、ステータスを「切り替え中」に口ックしています。 解決策：VMware サポートチームに連絡して HCX データベースをクリーンアップします。
スイッチオーバーが失敗しますが、強制電源オフオプションで機能します。	原因：VMware ツールのバージョンが HCX 一括移行の前提条件を満たしていなかったため、HCX はソース VM をシャットダウンできませんでした。 解決策：VMware ツールを移行タイプに適した推奨バージョンに更新します。
移行中に「継続的な一括移行には操作が許可されていません」というエラーが表示され、HCX サイトペアリングアプライアンスのアップグレードが失敗します。	原因：スイッチオーバー後に HCX データベースが更新されません。 解決策：移行が進行中でないことを保証します。サイトペアリングアプライアンスをアップグレードする時、強制アップグレードを選択します。
「リソースの可用性が低い」というエラーでカットオーバーが失敗しました。	原因：ホスト VM のストレージが不足しています。 解決策：移行前に、ストレージとリソースを確認します。

関連リソース

リファレンス

- VMware Cloud on AWS 機能
- VMware Cloud on AWS の概要と運用モデル (AWS 規範ガイダンス)
- HCXを使用するAWS上でVMware SDDCをVMware Cloud on AWSへ移行する (AWS 規範ガイダンス)
- VMware VMware Cloud on AWS HCX (ヴイエムウェアのドキュメント)

- HCX HCX リリースノート (ヴイエムウェアドキュメント)
- AWS の SDDC デプロイおよびベストプラクティスガイド (AWS ホワイトペーパー)

ツール

- PowerCLI を使用した VMware Cloud on AWS オートメーション (VMware クラウドテックゾーン)

パートナー

- VMware Cloud on AWS パートナーイニシアティブ

動画

- VMware Cloud on AWS (YouTube ビデオ)

pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する

作成者:Raunak Rishabh (AWS) and Jitender Kumar (AWS)

概要

このパターンは、[pg_transport] エクステンションを使用して PostgreSQL DB インスタンス用 2 つの Amazon Relational Database Service (Amazon RDS) 間で非常に大規模なデータベースを移行する手順を示しています。このエクステンションは、物理的な移行メカニズムで各データベースを移行させます。最小限のプロセスでデータベースファイルをストリーミングすることで、最小限のダウンタイムで DB インスタンス間で大規模なデータベースを移行できます。この拡張機能では、ターゲット DB インスタンスがソース DB インスタンスからデータベースをインポートするプルモデルを使用します。

前提条件と制限

前提条件

- 両方の DB インスタンスが同じメジャーバージョンの PostgreSQL を実行している必要があります。
- データベースはターゲットに存在してはいけません。そうしない場合、移行は失敗します。
- [pg_transport] 以外の拡張機能をソースデータベースで有効にしないでください。
- すべてのソースデータベースオブジェクトはデフォルトの pg_default テーブルスペースになければなりません。
- ソース DB インスタンスのセキュリティグループは、ターゲット DB インスタンスからのトラフィックを許可する必要があります。
- 「psql」や「PgAdmin」などの PostgreSQL クライアントをインストールして、Amazon RDS PostgreSQL DB インスタンスと連携させます。クライアントは、ローカルシステムにインストールすることも、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用することもできます。このパターンでは、EC2 インスタンスで psql を使用します。

機能制限

- Amazon RDS for PostgreSQL の異なるメジャーバージョン間でデータベースを転送することはできません。
- ソースデータベースからのアクセス許可と所有権は、ターゲットデータベースに移行されません。

- リードレプリカやリードレプリカの親インスタンス上でデータベースを転送することはできません。
- このメソッドで転送する予定のデータベーステーブルでは、reg データタイプを使用することはできません。
- DB インスタンス上で、合計 32 トランスポー (インポートとエクスポートの両方を含む) まで同時に実行できる。
- テーブルの名前を変更したり、テーブルを含めたり除外したりすることはできません。すべてはそのまま移行されます。

注意

- 拡張機能を削除すると依存オブジェクトやデータベースの操作に重要なデータも削除されるため、拡張機能を削除する前にバックアップを作成してください。
- [pg_transport] のワーカー数と `work_mem` 値を決定するときは、ソースインスタンス上の他のデータベースで実行されているインスタンスクラスとプロセスを考慮してください。
- トランスポートが開始されると、ソースデータベースのすべての接続が終了し、データベースは読み取り専用モードになります。

> **Note**
>
> トランスポートが 1 つのデータベースで実行されている場合、同じサーバー上の他のデータベースには影響しません。

製品バージョン

- Amazon RDS for PostgreSQL 10.10 以降、および Amazon RDS for PostgreSQL 11.5 以降、および Amazon RDS for PostgreSQL 11.5 以降。最新バージョン情報については、Amazon RDS ドキュメントの「DB インスタンス間の PostgreSQL データベースの転送」を参照してください。

アーキテクチャ

ツール

- pg_transport は各データベースを移動するための物理的なトランスポートメカニズムを提供します。最小限のプロセスでデータベースファイルをストリーミングすることで、物理的な移行は、従来のダンプおよびロードプロセスよりもはるかに早くデータを移動でき、ダウンタイムを最小限に抑えます。PostgreSQL トランスポータブルデータベースは、移行先 DB インスタンスが移行元 DB インスタンスからデータベースをインポートするプルモデルを使用します。このパターンで説明されているように、ソース環境とターゲット環境を準備するときに DB インスタンスにこのエクステンションをインストールします。
- 「psql」を使用すると、PostgreSQL DB インスタンスに接続して操作することができます。[psql] をシステムにインストールするには、「PostgreSQL のダウンロード」ページを参照してください。

エピック

ターゲットパラメータグループを作成する

タスク	説明	必要なスキル
ターゲットシステムのパラメータグループを作成します。	ターゲットパラメータグループとして識別できるグループ名 (例: `pgtarget-param-group`) を指定します。手順については、「Amazon RDS ドキュメント」を参照してください。	DBA
パラメータグループのパラメータを変更します。	以下のパラメータを設定します。 1. `shared_preload_libraries` パラメーターに `pg_transport` を追加します。 `shared_preload_libraries = pg_stat_s`	DBA

タスク	説明	必要なスキル
	tatements, pg_transport 2. pg_transport.num_workers パラメータを設定します。トランスポートを実行したいワーカーの数を選択します。設定した値によって、ソースで作成される transport.send_file ワーカーの数が決まります。 3. max_worker_processes の値を pg_transport.num_workers の値の 3 倍以上に増やします。たとえば、pg_transport.num_workers の値を 4 に設定した場合、max_worker_processes 値は 13 以上になるはずです。これが失敗した場合、[pg_transport] は最小値を推奨します。 4. pg_transport.timing を 1 に設定します。この設定により、移行中にタイミング情報を報告できます。 5. pg_transport.work_mem パラメータを設定します。このパラメータは、各ワーカーに割り当てる最大メモリを指定します。	

タスク	説明	必要なスキル
	デフォルト値は 128 MB です。 パラメータグループの詳細については、「Amazon RDS ドキュメント」を参照してください。	

ソースパラメーターグループの作成

タスク	説明	必要なスキル
ソースシステムのパラメータグループを作成します。	ソースパラメータグループであることがわかるグループ名 (例: `pgsource-param-group`) を指定します。手順については、「Amazon RDS ドキュメント」を参照してください。	DBA
パラメータグループのパラメータを変更します。	以下のパラメータを設定します。 1. `shared_preload_libraries` パラメーターに `pg_transport` を追加します。 ``` shared_preload_libraries = pg_stat_statements, pg_transport ``` 2. `pg_transport.num_workers` パラメータを設	DBA

タスク	説明	必要なスキル
	定します。ターゲットに定義されているこのパラメータの値によって、使用する transport.send_file ワーカーの数が決まります。このインスタンスでインポートを実行している場合は、この値を増やしてください。ただし、すでに実行中のワーカーの数を考慮してください。 3. ターゲット上でmax_worker_processes の値を pg_transport.num_workers の値の 3 倍以上に増やします。たとえば、pg_transport.num_workers ターゲットの値を 4 に設定した場合、max_worker_processes ソースの値は 13 以上でなければなりません。これが失敗した場合、[pg_transport] は最小値を推奨します。 4. pg_transport.work_mem パラメータを設定します。このパラメータは、各ワーカーに割り当てる最大メモリを指定します。デフォルト値は 128 MB です。	

タスク	説明	必要なスキル
	パラメータグループの詳細については、「Amazon RDS ドキュメント」を参照してください。	

ターゲット環境を準備する

タスク	説明	必要なスキル
ソースデータベースの転送先となる新しい Amazon RDS for PostgreSQL DB インスタンスを作成します。	ビジネス要件に基づいてインスタンスクラスと PostgreSQL バージョンを決定してください。	DBA、システム管理者、データベースアーキテクト
EC2 インスタンスから DB インスタンスポートに接続できるように、ターゲットのセキュリティグループを変更します。	デフォルトでは、PostgreSQL インスタンスのポートは 5432 です。別のポートを使用している場合は、そのポートへの接続を EC2 インスタンスで開いておく必要があります。	DBA、システム管理者
インスタンスを変更し、新しいターゲットパラメータグループを割り当てます。	例えば、`pgtarget-param-group` と指定します。	DBA
ターゲットの Amazon RDS DB インスタンスを再起動します。	パラメータ `shared_preload_libraries` とパラメータ `max_worker_processes` は静的パラメータで、インスタンスの再起動が必要です。	DBA、システム管理者
psql を使用して EC2 インスタンスからデータベースに接続します。	コマンドを使用します。	DBA

タスク	説明	必要なスキル
	`psql -h <rds_end_point> -p PORT -U username -d database -W`	
pg_transport 拡張機能を作成します。	`rds_superuser` ロールを持つユーザーとして次のクエリを実行します。 `create extension pg_transport;`	DBA

ソース環境の準備

タスク	説明	必要なスキル
Amazon EC2 インスタンスとターゲット DB インスタンスからの DB インスタンスポートへの接続を許可するように、ソースのセキュリティグループを変更します。	デフォルトでは、PostgreSQL インスタンスのポートは 5432 です。別のポートを使用している場合は、そのポートへの接続を EC2 インスタンスで開いておく必要があります。	DBA、システム管理者
インスタンスを変更し、新しいソースパラメータグループを割り当てます。	例えば、`pgsource-param-group` と指定します。	DBA
ソース Amazon RDS DB インスタンスを再起動します。	パラメータ `shared_preload_libraries` とパラメータ `max_worker_processes` は静的パラメータで、インスタンスの再起動が必要です。	DBA

タスク	説明	必要なスキル
psql を使用して EC2 インスタンスからデータベースに接続します。	コマンドを使用します。 `psql -h <rds_end_point> -p PORT -U username -d database -W`	DBA
pg_transport エクステンションを作成し、転送するデータベースから他のすべてのエクステンションを削除します。	ソースデータベースに [pg_transport] 以外の拡張機能がインストールされていると、転送は失敗します。このコマンドは、`rds_superuser` ロールを持つユーザーが実行する必要があります。	DBA

トランスポートの実行

タスク	説明	必要なスキル
ドライランをを実行します。	`transport.import_from_server` 関数を使用して、最初にリハーサルを実行します。 `SELECT transport.import_from_server( 'source-db-instance-endpoint', source-db-instance-port, 'source-db-instance-user', 'source-user-password', 'source-database-name', 'destination-user-password', 'true');`	DBA

タスク	説明	必要なスキル
	この関数の最後のパラメーター (true に設定) はドライランを定義します。 この関数は、メイントランスポートを実行したときに表示されるエラーをすべて表示します。メインストランスポートを実行する前に、エラーを解決してください。	
ドライランが成功したら、データベーストランスポートを開始します。	transport.import_from_server 関数を実行してトランスポートを実行します。ソースに接続し、データをインポートします。 `SELECT transport.import_from_server( 'source-db-instance-endpoint', source-db-instance-port, 'source-db-instance-user', 'source-user-password', 'source-database-name', 'destination-user-password', false);` この関数の最後のパラメーター (false に設定) は、これがドライランではないことを示しています。	DBA

タスク	説明	必要なスキル
ポストトランスポートステップを実行する。	データベーストランスポートが完了したら: • ターゲット環境のデータを検証します。 • すべてのロールと権限をターゲットに追加します。 • 必要に応じて、ターゲットとソースで必要な拡張機能をすべて有効にします。 • `max_worker_processes` パラメータの値を戻します。	DBA

関連リソース

- 「Amazon RDS ドキュメント」
- 「pg_transport ドキュメンテーション」
- 「RDS PostgreSQL トランスポータブルデータベースを使用したデータベースの移行」(ブログ投稿)
- 「PostgreSQL のダウンロード」
- 「psql ユーティリティ」
- DB パラメータグループを作成する
- 「DB パラメータグループ内のパラメータを変更する」
- 「PostgreSQL のダウンロード」

リプラットフォーム

トピック

- AWS DMS を使用して Microsoft SQL Server データベースを Amazon S3 にエクスポートする
- AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する
- OpenText TeamSite ワークロードの AWS クラウド への移行
- Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行
- データベースリンクを経由した直接 Oracle Data Pump Import を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle E-Business Suite を Amazon RDS Custom に移行
- Oracle PeopleSoft を Amazon RDS Custom に移行
- Oracle ROWID 機能を AWS の PostgreSQL に移行
- Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する
- Redis ワークロードを AWS 上の Redis Enterprise Cloud に移行
- AWS SCT と AWS DMS を使用して Amazon EC2 上の SAP ASE を Amazon Aurora PostgreSQL 互換の Amazon Aurora PostgreSQL 互換に移行します
- ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行
- メッセージキューを Microsoft Azure Service Bus から Amazon SQS に移行
- でリレーショナルデータベースを MongoDB Atlas に移行する AWS
- でセルフホスト型 MongoDB 環境を MongoDB Atlas に移行する AWS
- Oracle Data Pump と AWS DMS を使用して Oracle JD Edwards EnterpriseOne データベースを AWS に移行します
- AWS DMS を使用して Oracle PeopleSoft データベースを AWS に移行する
- オンプレミス MySQL データベースを Amazon RDS for MySQL に移行する
- オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- Rclone を使用して Microsoft Azure Blob から Amazon S3 にデータを移行する
- カウチベースサーバーから AWS 上のカウチベースカペラへの移行
- IBM WebSphere アプリケーションサーバーから Amazon EC2 の Apache Tomcat に移行
- IBM WebSphere アプリケーションサーバーから Amazon EC2 上の Apache Tomcat への自動スケーリングによる移行

- .NET アプリケーションを Microsoft Azure App Serviceから AWS Elastic Beanstalk に移行
- Amazon ECS で Oracle WebLogic から Apache Tomcat (ToMee) に移行する
- AWS DMS を使用して Oracle データベースを Amazon EC2 から Amazon RDS for Oracle に移行する
- Logstash を使用して、オンプレミス Oracle データベースを Amazon OpenSearch Service へ移行する
- オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- pglogic を使用して Amazon EC2 上の PostgreSQL から Amazon RDS for PostgreSQL に移行する
- オンプレミス PostgreSQL データベースを Aurora PostgreSQL に移行する
- オンプレミス Microsoft SQL Server データベースを、Linux を実行中の Amazon EC2 上の Microsoft SQL Server に移行する
- リンクされたサーバーを使用して、オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- ネイティブバックアップと復元メソッドを使用して、オンプレミスの Microsoft SQL サーバーデータベースを Amazon RDS for SQL Server に移行
- AWS DMS と AWS SCT を使用して Microsoft SQL Server データベースを Aurora MySQL に移行
- ネイティブツールを使用して オンプレミスの MariaDB Amazon RDS for MariaDB に移行する
- オンプレミス MySQL データベースを Aurora MySQL に移行する
- Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミスの MySQL データベースを Aurora MySQL に移行する
- AWS App2Container を使用したオンプレミスの Java プリケーションの AWS への移行
- AWS の大規模移行における共有ファイルシステムの移行
- Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する
- Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションへのデータベース移行をサポートするように Python と Perl アプリケーションを変更する
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

AWS DMS を使用して Microsoft SQL Server データベースを Amazon S3 にエクスポートする

作成者: Sweta Krishna (AWS)

概要

多くの場合、組織はデータベースの移行、バックアップと復元、データアーカイブ、データ分析のために、データベースを Amazon Simple Storage Service (Amazon S3) にコピーする必要があります。このパターンは、Microsoft SQL Server データベースを Amazon S3 にエクスポートする方法を説明しています。ソースデータベースは、オンプレミスでホストすることも、Amazon Web Services (AWS) クラウドの Microsoft SQL Server 用の Amazon Elastic Compute Cloud (Amazon EC2) または Amazon Relational Database Service (Amazon RDS) でホストすることもできます。

データは AWS Database Migration Service (AWS DMS) を使用してエクスポートされます。デフォルトでは、AWS DMS は全ロードデータおよび変更データキャプチャ (CDC) データをカンマ区切り値 (.csv) 形式で書き込みます。よりコンパクトなストレージとより高速なクエリオプションを実現するため、このパターンでは Apache Parquet (.parquet) 形式オプションを使用します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ターゲット S3 バケットへの書き込み、削除、タグ付けのアクセス権を持つアカウントの AWS Identity and Access Management (IAM) ロール、およびこの IAM ロールに信頼できるエンティティとして追加された AWS DMS (`dms.amazonaws.com`)
- オンプレミスの Microsoft SQL Server データベース (または EC2 インスタンスもしくは Amazon RDS for SQL Server 上の Microsoft SQL Serverデータベース)
- AWS 上の仮想プライベートクラウド (VPC) と、AWS Direct Connect または仮想プライベートネットワーク (VPN) が提供するオンプレミスネットワークとの間のネットワーク接続

制約事項

- VPC 対応 (ゲートウェイ VPC) S3 バケットは、現在、3.4.7 より前の AWS DMS バージョンではサポートされていません。
- 全ロード時のソーステーブル構造に対する変更はサポートされていません。

- AWS DMS フルラージバイナリオブジェクト (LOB) モードはサポートされていません。

製品バージョン

- Enterprise、Standard、Workgroup、および Developer エディションの Microsoft SQL Server バージョン 2005 以降。
- ソースとしての Microsoft SQL Server バージョン 2019 のサポートは、AWS DMS バージョン 3.3.2 以降で利用可能です。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Microsoft SQL Server データベース (または EC2 インスタンスまたは Amazon RDS for SQL Server 上の Microsoft SQL Server データベース)

ターゲットテクノロジースタック

- AWS Direct Connect
- AWS DMS
- Amazon S3

ターゲット アーキテクチャ

ツール

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- AWS Direct Connect は、標準のイーサネット光ファイバーケーブルを介して内部ネットワークを Direct Connect の場所にリンクします。この接続を使用すると、ネットワークパスのインターネットサービスプロバイダーを回避してパブリック AWS サービスに対する仮想インターフェイスを直接作成できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

エピック

移行の準備をする

タスク	説明	必要なスキル
データベースのバージョンを検証します。	ソースデータベースのバージョンを検証し、AWS DMS でサポートされていることを確認します。サポートされている SQL Server データベースのバージョンについては、「Microsoft SQL Server データベースを AWS DMS のソースとして使用する」を参照してください。	DBA
VPC とセキュリティグループを作成する	AWS アカウントで、VPC とセキュリティグループを作成します。詳細については、Amazon VPC のドキュメント参照してください。	システム管理者
AWS DMS タスクのユーザーを作成します。	ソースデータベースに AWS DMS ユーザーを作成し、読み取り権限を付与します。このユーザーは AWS DMS で使用します。	DBA
DB 接続をテストします。	AWS DMS ユーザーから SQL Server DB インスタンスへの接続をテストします。	DBA
S3 バケットを作成する。	ターゲットの S3 バケットを作成します。このバケットには、移行されたテーブルデータが格納されます。	システム管理者

タスク	説明	必要なスキル
IAM ポリシーとロールを作成する。	1. バケット権限のある IAM ポリシーを作成するには、「追加情報」セクションのコードを使用してください。 2. AWS DMS のロールのポリシーを作成して、ロールにポリシーをアタッチします。	システム管理者

AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	AWS マネジメントコンソールにサインインし、AWS DMS コンソールを開きます。ナビゲーションペインで、[レプリケーション インスタンス] を選択し、[レプリケーション インスタンスの作成] を選択します。手順については、「AWS DMS ドキュメント」のステップ 1 を参照してください。	DBA
ソースおよびターゲットエンドポイントを作成します。	ソースおよびターゲットエンドポイントを作成します。レプリケーション インスタンスからソースおよびターゲットエンドポイントの両方への接続をテストします。手順については、「AWS DMS ドキュメント」のステップ 2 を参照してください。	DBA

タスク	説明	必要なスキル
レプリケーションタスクを作成します。	レプリケーションタスクを作成し、[全ロード] または [変更データキャプチャ (CDC) による全ロード] を選択して SQL Server から S3 バケットにデータを移行します。手順については、「AWS DMS ドキュメント」のステップ 3 を参照してください。	DBA
データレプリケーションを開始する。	レプリケーションタスクを開始し、ログにエラーがないかモニタリングします。	DBA

データを検証する

タスク	説明	必要なスキル
移行したデータを検証します。	Amazon S3 コンソールで、ターゲット S3 バケットに移動します。ソースデータベースと同じ名前のサブフォルダを開きます。ソースデータベースから移行したすべてのテーブルがフォルダに含まれていることを確認します。	DBA

リソースをクリーンアップする

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンして削除する。	データ移行用に作成した一時的な AWS リソース (AWS DMS レプリケーションインス	DBA

タスク	説明	必要なスキル
	タンスなど) をシャットダウンし、エクスポートを検証した後に削除します。	

関連リソース

- Database Migration Service User Guide
- Microsoft SQL Server データベースの DMS のソースとしての使用
- Database Migration Service のターゲットとしての Amazon S3 の使用
- S3 バケットを DMS ターゲットとして使用する (AWS re:Post)

追加情報

次のコードを使用して、S3 バケット権限を持つ AWS DMS ロールが含まれる IAM ポリシーを追加します。`bucketname` をバケットの名前に置き換えます。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject",
                 "s3:DeleteObject"
             ],
             "Resource": [
                 "arn:aws:s3:::bucketname*"
             ]
         },
         {
             "Effect": "Allow",
             "Action": [
                 "s3:ListBucket"
             ],
             "Resource": [
                 "arn:aws:s3:::bucketname*"
             ]
         }
```

```
    ]
}
```

AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する

作成者: Mustafa Waheed (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、Unix または Linux サーバーで実行されているオンプレミスの機械学習 (ML) アプリケーションを、Amazon SageMaker を使用して AWS でトレーニングおよびデプロイするように移行するためのガイダンスを提供します。たとえば、継続的な統合や継続的なデプロイ (CI/CD) パイプラインを使用している場合です。移行パターンは AWS CloudFormation スタックを使用してデプロイします。

前提条件と制限

前提条件

- AWS Landing Zone を使用するアクティブな AWS アカウント
- Unix または Linux サーバーにインストールして構成された コマンドラインインターフェイス (CLI)
- AWS CodeCommit でプロビジョニングされた ML ソースコードリポジトリ

機能制限

- 1 つの AWS リージョンにデプロイできる個別のパイプラインは 300 だけです。
- このパターンは、Python でコードをトレーニングしてデプロイする、監督付きの ML ワークロードを対象としています。

製品バージョン

- Python 3.6x を使用する Docker バージョン 19.03.5、ビルド 633a0ea

アーキテクチャ

ソーステクノロジースタック

- ローカルファイルシステムまたはリレーショナルデータベース内のデータを含むオンプレミスの Linux コンピュートインスタンス

ソースアーキテクチャ

ターゲットテクノロジースタック

- データストレージ用に Amazon S3 を使用し、パイプライン実行の追跡または口ギング用のメタデータストアとして使用する Amazon DynamoDB とともにデプロイされる AWS CodePipeline

ターゲット アーキテクチャ

アプリケーション移行アーキテクチャ

- ネイティブ Python パッケージと AWS CodeCommit リポジトリ (データベースインスタンス上のオンプレミスデータセットの場合は SQL クライアント)

ツール

- Python3
- Git
- AWS CLI — CLI は AWS CloudFormation スタックをデプロイし、データを S3 バケットに移動します。それを受けてS3 バケットはターゲットに移動します。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースコードとデータセットを検証します。		データサイエンティスト

タスク	説明	必要なスキル
ターゲットのビルド、トレーニング、デプロイのインスタンスタイプとサイズを特定します。		データエンジニア、データサイエンティスト
機能リストと容量要件を作成します。		
ネットワーク要件を特定します。		DBA、システム管理者
ソースアプリケーションとターゲットアプリケーションのネットワークまたはホストアクセスセキュリティ要件を特定します。		データエンジニア、ML エンジニア、システム管理者
バックアップ戦略を決定します。		ML エンジニア、システム管理者
可用性の要件を決定します。		ML エンジニア、システム管理者
アプリケーションの移行またはスイッチオーバー戦略を特定します。		データサイエンティスト、ML エンジニア

インフラストラクチャを構成する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成		ML エンジニア、システム管理者
セキュリティグループを作成します。		ML エンジニア、システム管理者

タスク	説明	必要なスキル
ML コード用に Amazon S3 バケットと AWS CodeCommit リポジトリブランチを設定する。		ML エンジニア

データとコードをアップロードする

タスク	説明	必要なスキル
ネイティブ MySQL ツールまたはサードパーティツールを使用して、データセットをプロビジョニングされた S3 バケットに移行、トレーニング、検証、テストします。	これは AWS CloudFormation スタックのデプロイに必要です。	データエンジニア、ML エンジニア
ML トレインとホスティングコードを Python パッケージとしてパッケージ化し、AWS CodeCommit または GitHub のプロビジョニングされたリポジトリにプッシュします。	移行用に AWS CloudFormation テンプレートをデプロイするには、リポジトリのブランチ名が必要です。	データサイエンティスト、ML エンジニア

アプリケーションを移行する

タスク	説明	必要なスキル
ML ワークロードの移行戦略に従います。		アプリ所有者、ML エンジニア
AWS CloudFormation スタックをデプロイします。	AWS CLI を使用して、このソリューションで提供される YAML テンプレートで宣言されたスタックを作成します。	データサイエンティスト、ML エンジニア

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		アプリ所有者、データサイエンティスト、ML エンジニア

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。	AWS CloudFormation テンプレートからすべてのカスタムリソース (たとえば、使用されていない AWS Lambda 関数) をシャットダウンします。	データサイエンティスト、ML エンジニア
プロジェクト文書を確認して検証する。		アプリ所有者、データサイエンティスト
結果と ML モデルの評価指標をオペレーターと検証します。	モデルのパフォーマンスがアプリケーションユーザーの期待と一致し、オンプレミスの状態と同等であることを確認します。	アプリ所有者、データサイエンティスト
プロジェクトを終了し、フィードバックを提供します。		アプリ所有者、ML エンジニア

関連リソース

- CodePipeline
- AWS CodeCommit
- CodeBuild

- Amazon SageMaker
- Amazon S3
- Amazon DynamoDB
- Lambda

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

OpenText TeamSite ワークロードの AWS クラウド への移行

作成者:Battulga Purevragchaa (AWS), Michael Stewart, and Carlos Marruenda Molina

概要

> **Warning**
>
> このシナリオでは、プログラムによるアクセスと長期的な認証情報を持つ IAM ユーザーが必要です。これはセキュリティ上のリスクをもたらします。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。アクセスキーは、必要に応じて更新できます。詳細については、IAM ユーザーガイドの「アクセスキーの更新」を参照してください。

多数の「OpenText Experience Platform」インスタンスは、固定容量またはレガシーホスティングコストモデルを備えたオンプレミスまたは従来のホスティングソリューションでホストされます。OpenText Experience Platform ワークロードを Amazon Web Services (AWS) クラウドに移行することで、ビジネスの俊敏性と統合機会が増加し、全体的な所有コストが削減されるため、さらなる機能と価値が提供されます。

このパターンは、「OpenText TeamSite」ワークロードを AWS クラウドに移行するためのステップとテンプレートを提供します。このパターンは、OpenText TeamSite の移行プロセスをガイドする詳細なエピックセクションを提供することで、移行プロジェクトの範囲と予算の決め方を理解するのに役立ちます。

このパターンは AWS と AWS パートナーの「TBSCG」が開発したもので、AWS 規範ガイダンスのウェブサイトにある「OpenText TeamSite とメディア管理のワークロードを AWS クラウドへの移行」というガイドに添付されています。

前提条件と制限

前提条件

- 少なくとも 1 つのアクティブな AWS アカウント
- オンプレミスのデータセンターまたは別のクラウドプロバイダーでホストされている OpenText ワークロード
- OpenText アクティブな OpenText ライセンス

移行プロセスには、次の表で説明する役割と責任も必要です。

ロール	責任
スポンサー	内部スポンサー
デリバリーマネージャー	移行デリバリー
ソリューションアーキテクチャ	現在のアーキテクチャと新しいアーキテクチャを定義
DevOps エンジニア	DevOps アクティビティ
QA テスター	システムレベルのテスト
プロダクトオーナー	ビジネス要件に基づくタスクの優先順位付け
チーム/サイト作成者	移行ユーザー受け入れテスト (UAT)
チームサイト管理者	移行 UAT
OpenText リード	OpenText プロダクトスペシャリスト
OpenText 開発者	OpenText プロダクトスペシャリスト
価格設定スペシャリスト	AWS と OpenText のライセンス
IT セキュリティ	IT セキュリティベースライン
サードパーティー統合デベロッパー	既存のインテグレーションをやり直す
フロントエンドデベロッパー	移行したフロントエンドコードに変更を加える
データベース管理者	データベース設定

制限事項

- ターゲットオペレーティングシステム (OS) との互換性を確認する。移行する OpenText 製品バージョンの製品リリースノートにある互換性マトリックスを使用できます。

アーキテクチャ

ソーステクノロジースタック

- OpenText カスタマーエクスペリエンスソリューションは、オンプレミスまたは別のクラウドプロバイダーでホストされています。
  - OpenText TeamSite
  - OpenText LiveSite
  - OpenText メディア管理
  - OpenText MediaBin

ターゲットテクノロジースタック

- AWS クラウド上でホストされる OpenText カスタマーエクスペリエンスプラットフォームで、次の AWS サービスを使用します。
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - Amazon Elastic Container Service (Amazon ECS)
  - Amazon OpenSearch Service
  - エラスティックロードバランシング
  - AWS Lambda
  - Amazon API Gateway
  - Amazon Relational Database Service (Amazon RDS)
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon Simple Storage Service (Amazon S3)

ターゲット アーキテクチャ

ツール

- 「AWS Database Migration Service (AWS DMS)」は、リレーショナルデータベース、データウェアハウス、NoSQL データベース、他の種類のデータストアを移行しやすくするクラウドサービスです。

- 「AWS アプリケーション移行サービス」は、ソースサーバーを AWS 上でネイティブに稼働するように自動的に変換します。また、組み込みの最適化オプションとカスタム最適化オプションにより、アプリケーションのモダナイゼーションを簡素化します。

エピック

発見と評価

タスク	説明	必要なスキル
発見要件に関するワークショップを開催する。	ビジネスチームや技術チームとワークショップを開催して、現在の状況を把握し、要件を収集し、移行戦略を検証します。移行の複雑さと範囲によっては、組織内で複数のワークショップが必要になる場合があります。 期間: 2 週間	スポンサー (オプション)、デリバリーマネージャー、ソリューションアーキテクト、OpenText リード、プロダクトオーナー
ソリューションと移行の要件を分析します。	計画中のソリューションの設計と移行プロセスに影響を与えるビジネス、機能、技術要件を分析し、文書化する。 期間: 1 週間	ソリューションアーキテクト、OpenText リーダー、プロダクトオーナー
既存の OpenText アーキテクチャを文書化してください。	コアコンポーネントや関連するすべてのアプリケーションやサービスを含め、既存の OpenText アーキテクチャを文書化してください。 期間: 1 週間	ソリューションアーキテクト、OpenText リーダー、プロダクトオーナー
計画中の AWS アーキテクチャを定義します。	特定されたコンポーネント、要件、および OpenText 互換性マトリックスを使用して、	ソリューションアーキテクト、OpenText リーダー、プロ

タスク	説明	必要なスキル
	計画する AWS アーキテクチャを定義します。OpenText 互換性マトリックスは、お使いの OpenText TeamSite バージョンのリリースノートに記載されています。 期間: 1 週間	ダクトオーナー、IT セキュリティ
計画している AWS アーキテクチャの規模を評価してください。	サイズ要件は、ワークロードやその他の機能以外の要件に応じて、アーキテクチャコンポーネントによって異なります。 期間: 2 日間	ソリューションアーキテクト、OpenText リード
TCO を計算します。	提案するソリューションの総保有コスト (TCO) を計算します。 期間: 2 日間	ソリューションアーキテクト、価格スペシャリスト
各コンポーネントの移行戦略を定義します。	AWS クラウドに移行する必要がある各コアまたは追加コンポーネントに 7 つの一般的な移行戦略 (7 R) のどれを使用するかを定義し、文書化します。 期間: 1 週間	ソリューションアーキテクト、OpenText リーダー、プロダクトオーナー
コンポーネントの移行プロセスを定義します。	ワークロードのコンポーネントごとに詳細な移行プロセスを定義します。 期間: 1 週間	ソリューションアーキテクト、OpenText リーダー、プロダクトオーナー、IT セキュリティ

タスク	説明	必要なスキル
グローバルな移行プロセスと依存関係を定義します。	コンポーネント、依存関係、事業継続性に関する移行の詳細を記載したグローバルな移行プロセスとカレンダーを作成します。 期間: 3 日間	ソリューションアーキテクト、OpenText リーダー、プロダクトオーナー、IT セキュリティ

セキュリティとコンプライアンス活動

タスク	説明	必要なスキル
セキュリティポリシーの作成	AWS アカウントで顧客管理のセキュリティポリシーを設定します。これには、未使用のアカウントを自動的にオフにするだけでなく、パスワードの複雑さやローテーションを含める必要があります。 カスタマー管理ポリシーの詳細については、AWS Identity and Access Management (IAM) ドキュメントの「カスタマー管理ポリシー」を参照してください。	ソリューションアーキテクチャ
IAM ユーザーを作成する	AWS マネジメントコンソール、AWS コマンドラインインターフェイス (AWS CLI)、AWS SDK へのアクセスを必要とする IAM ユーザーを作成します。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
	IAM ユーザーの作成方法について詳しくは、IAM ドキュメントの「AWS アカウント内での IAM ユーザーの作成」を参照してください。	
IAM グループを作成する	必要な IAM ユーザーグループ (管理者グループや開発者グループなど) を作成し、それらのグループに IAM ユーザーを追加します。 IAM グループの詳細については、IAM ドキュメントの「IAM ユーザーグループ」を参照してください。	ソリューションアーキテクチャ
セキュリティポリシーをアタッチします。	セキュリティポリシーを IAM グループまたはロールにアタッチします。 詳細については、IAM ドキュメント内にある「IAM ユーザーグループへのポリシーのアタッチ」を参照してください。	ソリューションアーキテクチャ
詳細請求をオンにします。	詳細については、AWS Billing and Cost Management ドキュメントの「使用状況とコストの監視」を参照してください。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
アカウントの連絡先情報を確認してください。	アカウントの連絡先が最新のものであり、組織内の 1 人以上の個人にマッピングされていることを確認します。 詳細については、AWS Billing and Cost Management ドキュメントの「AWS アカウントの管理」を参照してください。	ソリューションアーキテクト、プロダクトオーナー
セキュリティの連絡先情報を追加します。	セキュリティ連絡先情報を使用して連絡先情報を設定します。 詳細については、AWS Billing and Cost Management ドキュメントの「AWS アカウントの管理」を参照してください。	IT セキュリティ、ソリューションアーキテクト
EC2 インスタンスに IAM ロールを設定します。	EC2 インスタンスの IAM ロールを設定します。 詳細については、Amazon EC2 のドキュメントの「Amazon EC2 の IAM ロール」を参照してください。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
AWS サポートへのアクセスを設定します。	サポートセンターの AWS サポートへのアクセスとサポートケースの作成を必要とする IAM ユーザーに IAM ポリシーをアタッチします。 詳細については、AWS サポートドキュメントの「AWS サポートのアクセス許可」を参照してください。	ソリューションアーキテクチャ
CloudTrail を有効にします。	すべての AWS リージョンで AWS CloudTrail を自動的に有効にします。 詳細については、AWS CloudTrail ドキュメントの「`create-trail` の使用」を参照してください。	ソリューションアーキテクチャ
CloudTrail ログファイルの検証を有効にします。	CloudTrail ログファイルの検証を有効にします。 詳細については、AWS CloudTrail ドキュメントの「CloudTrail ログファイルの整合性の検証」を参照してください。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
CloudTrail ログを含むすべての S3 バケットへのアクセスを制限します。	CloudTrail ログファイルを含む S3 バケットへのアクセスを制限するバケットポリシーを適用します。 詳細については、AWS CloudTrail ドキュメントの「CloudTrail 用の Amazon S3 バケットポリシー」を参照してください。	ソリューションアーキテクチャ
CloudTrail を CloudWatch ログと統合する	CloudTrail によって生成された証跡を Amazon CloudWatch Logs と統合します。 詳細については、AWS CloudTrail ドキュメントの「CloudWatch Logs へのイベントの送信」を参照してください。	ソリューションアーキテクチャ
必要なすべてのリージョンで AWS Config を有効にします。	必要なすべてのリージョンで AWS Config を自動的に有効にします。 AWS CLI を使用して AWS Config をセットアップできます。詳細については、AWS Config ドキュメントの「AWS CLI による AWS Config の設定」を参照してください。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
S3 バケットへのアクセスのログ記録を有効にします。	CloudTrail を使用して S3 バケットのアクセスログ記録を自動化します。 詳細については、Amazon S3 ドキュメントにある「S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化」を参照してください。	ソリューションアーキテクチャ
CloudTrail 用の AWS KMS キーポリシーを設定します。	CloudTrail 用の AWS Key Management Service (AWS KMS) キーポリシーの設定を自動化します。 詳細については、AWS CloudTrail ドキュメントの「CloudTrail 用の AWS KMS キーポリシーの設定」を参照してください。	ソリューションアーキテクチャ
CloudTrail ログを暗号化します。	AWS KMS に保持されている顧客管理キーを使用して、CloudTrail ログのサーバー側暗号化を設定します。 詳細については、AWS CloudTrail ドキュメントの「AWS KMS で管理されたキー (SSE-KMS) による CloudTrail ログファイルの暗号化」を参照してください。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
KMS キーを自動的にローテーションします。	AWS KMS キーのローテーションを設定します。 詳細については、AWS KMS ドキュメントの「自動キーローテーションを有効または無効にする方法」を参照してください。	ソリューションアーキテクチャ
CloudWatch アラームを設定する	特定のイベントによって開始される Amazon CloudWatch アラームを設定します。たとえば、API への不正なリクエストやルートアカウントの使用などです。 詳細については、AWS セキュリティブログの「AWS アカウントのルートアクセスキーが使用された場合に通知を受け取る方法」を参照してください。	ソリューションアーキテクチャ
セキュリティグループを設定する	ポート 22 と 3389 では無制限のインバウンドトラフィックが許可されないようにセキュリティグループを設定します。	ソリューションアーキテクチャ

タスク	説明	必要なスキル
VPC フローログ記録を有効にします。	仮想プライベートクラウド (VPC) のネットワークインターフェイスとの間で行き来する IP トラフィックをキャプチャし、CloudWatch がキャプチャするように設定します。 詳細については、Amazon VPC ドキュメントの「フローログの作成」を参照してください。	ソリューションアーキテクチャ
デフォルトのセキュリティグループを変更して、すべてのトラフィックを制限します。	トラフィックがデフォルトで拒否され、アクセスがセキュリティグループを通じて明示的に許可されるように、各 VPC のデフォルトセキュリティグループを変更します。 詳細については、Amazon VPC ドキュメントの「VPC のセキュリティグループ」を参照してください。	ソリューションアーキテクチャ
VPC 間のルーティングテーブルを設定します。	VPC ピアリングのルーティングテーブルを、必要最小限のアクセス数で設定します。 詳細については、Amazon VPC ドキュメントの「VPC ピアリング接続のルートテーブルの更新」を参照してください。	ソリューションアーキテクチャ

新しい AWS インフラストラクチャのセットアップアクティビティ

タスク	説明	必要なスキル
AWS インフラストラクチャをプロビジョニングします。	AWS アカウントとリソースを作成します。 期間: 2 週間	DevOps エンジニア、ソリューションアーキテクト
DevOps ツールとプロセスをセットアップします。	継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインや自動テストフレームワークなどの DevOps ツールと手順を設定します。	DevOps エンジニア、ソリューションアーキテクト
コアコンポーネントの移行を自動化します。	既存のテンプレートやスクリプトを使用して、TeamSite、LiveSite、OpenDeploy、MediaBin などの OpenText 製品のインストールと設定を自動化できます。 期間: 1 週間	DevOps エンジニア、ソリューションアーキテクト、OpenText リーダー
追加コンポーネントの移行を自動化します。	OpenText コアコンポーネント (追加データベース、通信、監視、キャッシュコンポーネントなど) と統合されたその他のアプリケーションの移行を分析し、自動化します。 期間: 2 週間	DevOps エンジニア、ソリューションアーキテクト、OpenText リーダー
コアコンポーネントを適応させる。	OpenText コアコンポーネントのカスタマイズ (インテグレーションなど) に必要な変更を加える。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者、サードパーティ統合開発者、フロントエンド開発者

タスク	説明	必要なスキル
追加サービスの実装と設定。	AWS Lambda 関数や Amazon API Gateway など、あらゆる AWS の新しいサービスをプロビジョニング、設定、実装します。	DevOps エンジニア、ソリューションアーキテクト、サードパーティ統合開発者、フロントエンド開発者
他のコンポーネントを移行またはリファクタリングする。	必要なリファクタリングを含め、その他のコンポーネントを移行する。これには、カスタムメイドのレポートポータルや既存の API 統合レイヤーなどの外部アプリケーションが含まれます。	DevOps エンジニア、ソリューションアーキテクト、サードパーティ統合開発者、フロントエンド開発者
開発環境で移行を行います。	システムプロビジョニング、データ移行、アプリケーション移行、インストール、設定など、開発環境の移行アクティビティを自動化します。	DevOps エンジニア
本番環境で移行を行います。	システムプロビジョニング、データ移行、アプリケーション移行、インストール、設定など、運用環境の移行アクティビティを自動化します。	DevOps エンジニア

ネットワーク活動

タスク	説明	必要なスキル
各 VPC に対して CIDR ブロックを定義します。	デフォルト以外の VPC ごとに、Classless Inter-Domain Routing (CIDR) ブロック (IP 範囲とマスク) を定義します。	DevOps エンジニア、ソリューションアーキテクト

タスク	説明	必要なスキル
	期間: 1 週間未満	
サブネットとアベイラビリティーゾーンを定義します。	デフォルト以外の各 VPC で使用されるサブネットとアベイラビリティーゾーンを定義します。 期間: 1 週間未満	DevOps エンジニア、ソリューションアーキテクト
セキュリティグループを定義します。	AWS リソースのセキュリティを制御するためのセキュリティグループとセキュリティグループルールを定義します。 期間: 1 週間未満	DevOps エンジニア、ソリューションアーキテクト
ネットワーク ACL を定義します。	サブネット境界のセキュリティを制御するネットワークアクセスコントロールリスト (ACL) を定義します。 期間: 1 週間未満	DevOps エンジニア、ソリューションアーキテクト

データベースの移行

タスク	説明	必要なスキル
ソースデータベースを準備します。	AWS DMS を使用して、AWS クラウドへの継続的なレプリケーションに備えて各ソースデータベースを準備します。	DevOps エンジニア、ソリューションアーキテクト
OpenText コアコンポーネント用のデータベースを作成します。	Opentext TeamSite、LiveSite、および MediaBin の各コンポーネントに必要	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者

タスク	説明	必要なスキル
	なデータベースを作成します。OpenText のインストールドキュメントに従って、ユーザーとアクセス権が正しく設定されていることを確認してください。	
ソースデータベースサーバーからデータをコピーする。	OpenText コアコンポーネント用のデータをソースデータベースサーバーからターゲットデータベースサーバーにコピーするプロセスを自動化します。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者
データベースサーバーからのデータを同期します。	ソースデータベースからターゲットデータベースへの定期的なデータ同期処理を自動化します。	OpenText 開発者

コンテンツ移行アクティビティ

タスク	説明	必要なスキル
OpenText チームサイトのコンテンツストアをコピーしてください。	コンテンツストアをソース OpenText TeamSite サーバーからターゲット OpenText TeamSite サーバーにコピーするプロセスを自動化します。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者
ユーザーとグループをマップします。	OpenText TeamSite の内部ユーザー ID をターゲットシステム ID に内部でマッピングします。	OpenText リード

タスク	説明	必要なスキル
OpenText TeamSite コンテンツストアを同期します。	ソースとターゲットのコンテンツストアを定期的に同期するプロセスを自動化します。これは移行と QA プロセスの一環として実装されます。	OpenText 開発者
ウェブサーバーからデータをコピーする。	ソースウェブサーバーからターゲットウェブサーバーにデータをコピーするプロセスを自動化します。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者
ウェブサーバーのデータを同期します。	ソースとターゲットのウェブサーバーデータを定期的に同期するプロセスを自動化します。	OpenText 開発者
ウェブサーバーのファイルシステムからデータをコピーします。	コンテンツやその他のウェブアセットをソースウェブサーバーのファイルシステムからターゲットウェブサーバーにコピーするプロセスを自動化します。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者
ウェブ サーバーのファイルシステムを同期します。	ソースウェブサーバーのファイルシステムからターゲットウェブサーバーへ、コンテンツやその他のウェブ資産を定期的に同期するプロセスを自動化します。	OpenText 開発者

タスク	説明	必要なスキル
フィードとインデックスを生成します。	OpenText TeamSite やウェブサーバーのコンテンツをデータソースとして使用するフィードやその他のインデックス (ウェブ検索など) を生成するプロセスの実行プロセスを自動化します。	ソリューションアーキテクト、OpenText リーダー、OpenText 開発者
フィードとインデックスの生成を同期します。	データ同期後にフィードとインデックスを定期的に再生成するプロセスを自動化します。	OpenText 開発者

テストと QA アクティビティ

タスク	説明	必要なスキル
移行 QA を実行する。	ターゲット AWS 環境、アプリケーション、サービスをテストして、自動移行プロセスが正しく構築および設定されていることを確認します。	DevOps エンジニア、OpenText リード、QA テスター
パフォーマンステストを実施する。	特定のワークロードにおける応答性と安定性の観点からパフォーマンスをテストします。スケーラビリティや信頼性など、宛先システムのその他の品質属性を調査、測定、検証します。 このテストが役に立つためには、運用環境と同じ規模のテスト環境が必要です。	DevOps エンジニア、OpenText リーダー

タスク	説明	必要なスキル
	期間: 1 週間から 2 週間	
セキュリティテスト。	脆弱性スキャンとペネトレーションテストにより、データを保護し、必要に応じて機能を維持するアプリケーションのセキュリティメカニズムの潜在的な欠陥を明らかにします。 このテストが役に立つためには、ネットワークとセキュリティの点で実稼働環境と同等のテスト環境が必要です。 期間: 1 週間から 2 週間	DevOps エンジニア、OpenText リーダー

オペレーショナル統合アクティビティ

タスク	説明	必要なスキル
運用準備状況の確認	現在 IT 運用をどのように行っているか、また AWS クラウドでどのように運用するかを理解してください。このビジネス成果は、クラウド運用モデルを定義することで達成できます。 期間: 1 週間	DevOps エンジニア、OpenText リード、サービスデリバリーマネージャー
運用自動化への投資。	自動化に投資して、AWS の運用モデルを実現しましょう。	DevOps エンジニア、OpenText リード、サービスデリバリーマネージャー

タスク	説明	必要なスキル
運用を統合します。	現在の IT ツールを引き続き使用し、AWS クラウドに統合して拡張してください。	DevOps エンジニア、OpenText リード、サービスデリバリーマネージャー

カットオーバーアクティビティ

タスク	説明	必要なスキル
DNS を切り替えます。	ドメインネームシステム (DNS) を既存のホストから AWS クラウドベースのホストに手動で切り替えます。 所要時間: 1 時間	DevOps エンジニア、OpenText リーダー
ディザスタリカバリをテストします。	ディザスタリカバリ、バックアップ、リストアをテストし、自動テストを実行します。 所要時間: 1 日	DevOps エンジニア、OpenText リード、QA テスター
モニタリングと分析を検証します。	モニタリングと分析が機能していることを検証します。 所要時間: 2 時間	DevOps エンジニア、OpenText リーダー
古い環境をオフにし、サーバーのシャットダウンをリクエストします。	期間: 3 日間	DevOps エンジニア、OpenText リーダー

関連リソース

- カスタマー管理ポリシー
- 「AWS アカウントでの IAM ユーザーの作成」

- IAM ユーザーグループ
- 「IAM ユーザーグループへのポリシーのアタッチ」
- 「使用状況とコストのモニタリング」
- 「AWS アカウントの管理」
- Amazon EC2 の IAM ロール
- 「AWS サポートのアクセス許可」
- 「create-trail の使用」
- 「CloudTrail のログファイルの整合性検証を有効にする」
- 「CloudTrail の Amazon S3 バケットポリシー」
- 「CloudWatch Logs へのイベントの送信」
- AWS CLI を使用して AWS Config を設定する
- 「S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化」
- 「CloudTrail 用の AWS KMS キーポリシーを設定する」
- 「AWS KMS で管理されたキー (SSE-KMS) による CloudTrail ログファイルの暗号化」
- 「自動キーローテーションを有効または無効にする方法」
- 「AWS アカウントのルートアクセスキーが使用されたときに通知を受け取る方法」
- 「フローログの作成」
- VPC のセキュリティグループ
- 「VPC ピアリング接続のルートテーブルを更新する」

Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行

作成者:Sai Krishna Namburu (AWS) and Sindhusha Paturu (AWS)

概要

このパターンは、Amazon Aurora PostgreSQL 互換エディションと PostgreSQL 用 Amazon Relational Database Service (Amazon RDS) で Oracle キャラクターラージオブジェクト (CLOB) の値を個々の行に分割する方法を示しています。PostgreSQL は CLOB データ型をサポートしていません。

インターバルパーティションのあるテーブルはソース Oracle データベースで識別され、テーブル名、パーティションのタイプ、パーティションの間隔、およびその他のメタデータがキャプチャされ、ターゲットデータベースにロードされます。サイズが 1 GB 未満の CLOB データは、AWS Database Migration Service (AWS DMS) を使用してテキストとしてターゲットテーブルにロードできます。または、データを CSV 形式でエクスポートして Amazon Simple Storage Service (Amazon S3) バケットにロードし、ターゲット PostgreSQL データベースに移行できます。

移行後は、このパターンで提供されるカスタム PostgreSQL コードを使用して、CLOB データを改行文字識別子 (`CHR(10)`) に基づいて個々の行に分割し、ターゲットテーブルにデータを入力できます。

前提条件と制限

前提条件

- 区間パーティションと CLOB データ型のレコードを含む Oracle データベーステーブル。
- Aurora PostgreSQL 互換または Amazon RDS for PostgreSQL データベースで、ソーステーブルと同様のテーブル構造 (同じ列とデータ型) を備えています。

機能制限

- CLOB 値は 1 GB を超えることはできません。
- ターゲットテーブルの各行には、改行文字 ID が必要です。

製品バージョン

- Oracle 12c
- Aurora ポストグレス11.6

アーキテクチャ

次の図は、CLOB データを含むソース Oracle テーブルと、Aurora PostgreSQL 互換バージョン 11.6 の同等の PostgreSQL テーブルを示しています。

ツール

AWS サービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。
- 「Amazon Relational Database Service (Amazon RDS)」を使用して、AWS クラウドでの PostgreSQL リレーショナルデータベースをセットアップ、運用、スケーリングできます。
- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

以下のクライアントツールを使用して、Aurora PostgreSQL 互換データベースと Amazon RDS for PostgreSQL データベースへの接続、アクセス、管理を行うことができます。(これらのツールはこのパターンでは使用されません)。

- 「pgAdmin」は PostgreSQL 用のオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。
- 「DBeaver」は、開発者やデータベース管理者向けのオープンソースのデータベースツールです。このツールを使用して、データの操作、監視、分析、管理、移行を行うことができます。

ベストプラクティス

データベースを Oracle から PostgreSQL に移行するためのベストプラクティスについては、AWS ブログ記事「Oracle データベースを Amazon RDS PostgreSQL または Amazon Aurora PostgreSQL に移行するためのベストプラクティス:移行プロセスとインフラストラクチャに関する考慮事項」を参照してください。

大きなバイナリオブジェクトを移行するための AWS DMS タスクを設定するベストプラクティスについては、AWS DMS ドキュメントの「ラージバイナリオブジェクト (LOB) の移行」を参照してください。

エピック

CLOB データを識別してください。

タスク	説明	必要なスキル
CLOB データを分析します。	ソース Oracle データベースで CLOB データを分析して列ヘッダーが含まれているかどうかを確認し、データをターゲットテーブルに読み込む方法を決定できるようにします。 入力データを分析するには、次のクエリを使用します。 `SELECT * FROM clobdata_or;`	開発者
CLOB データをターゲットデータベースにロードします。	CLOB データを含むテーブルを Aurora または Amazon RDS ターゲットデータベースの中間 (ステージング) テーブルに移行します。AWS DMS を使用するか、CSV ファイルとして Amazon S3 バケットにアップロードできます。 このタスクで AWS DMS を使用する方法については、AWS DMS ドキュメントの「Oracle データベースのソースとしての使用」と「PostgreSQL データベースのターゲットと	移行エンジニア、DBA

タスク	説明	必要なスキル
	しての使用」を参照してください。 このタスクに Amazon S3 を使用する方法については、AWS DMS ドキュメントの「ターゲットとしての Amazon S3 の使用」を参照してください。	

ターゲット PostgreSQL テーブルを検証します。	ターゲットデータベースで以下のクエリを使用して、ヘッダーを含むターゲットデータをソースデータに対して検証します。
	
	`SELECT * FROM clobdata_pg;`
	`SELECT * FROM clobdatatarget;`
	
	結果を (最初のステップの) ソースデータベースからのクエリ結果と比較します。
		開発者

CLOB データを別々の行に分割します。	「追加情報」セクションに記載されているカスタム PostgreSQL コードを実行して、CLOB データを分割し、ターゲット PostgreSQL テーブルの別々の行に挿入します。	開発者

データを検証します。

タスク	説明	必要なスキル

ターゲットテーブルのデータを検証します。	次のクエリを使用して、ターゲットテーブルに挿入されたデータを検証します。
	`SELECT * FROM clobdata_pg;`
	`SELECT * FROM clobdatatarget;`
		開発者

関連リソース

- 「CLOB データ型」(Oracle ドキュメント)
- 「データタイプ」(PostgreSQL ドキュメンテーション)

追加情報

CLOB データを分割するための PostgreSQL 関数

```
do
$$
declare
totalstr varchar;
str1 varchar;
str2 varchar;
pos1 integer := 1;
pos2 integer ;
len integer;


begin
        select rawdata||chr(10) into totalstr from clobdata_pg;
        len :=  length(totalstr) ;
        raise notice 'Total length : %',len;
        raise notice 'totalstr : %',totalstr;
        raise notice 'Before while loop';
```

```
        while pos1 < len  loop


                  select position (chr(10) in totalstr) into pos2;
                 raise notice '1st position of new line : %',pos2;


                str1 := substring (totalstr,pos1,pos2-1);
                raise notice 'str1 : %',str1;


                   insert into clobdatatarget(data) values (str1);
                   totalstr := substring(totalstr,pos2+1,len);
                   raise notice 'new totalstr :%',totalstr;
                len :=  length(totalstr) ;


        end loop;
end
$$
LANGUAGE 'plpgsql' ;
```

入力例と出力例

データを移行する前に、次の例を使用して PostgreSQL コードを試すことができます。

3 行の入力を含む Oracle データベースを作成します。

```
CREATE TABLE clobdata_or (
id INTEGER GENERATED ALWAYS AS IDENTITY,
rawdata clob  );


insert into clobdata_or(rawdata) values (to_clob('test line 1') || chr(10) ||
 to_clob('test line 2') || chr(10) || to_clob('test line 3') || chr(10));
COMMIT;


SELECT * FROM clobdata_or;
```

これにより、以下の出力が表示されます。

id	ローデータ
1	テストライン 1 テストライン 2 テストライン 3

ソースデータを PostgreSQL ステージングテーブル (clobdata_pg) にロードして処理します。

```
SELECT * FROM clobdata_pg;

CREATE TEMP TABLE clobdatatarget (id1 SERIAL,data VARCHAR );

<Run the code in the additional information section.>

SELECT * FROM clobdatatarget;
```

これにより、以下の出力が表示されます。

id1	データ
1	テストライン 1 行目
2	テストライン 2 行目
3	テストライン 3 行目

データベースリンクを経由した直接 Oracle Data Pump Import を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する

作成者: Rizwan Wangde (AWS)

概要

多数のパターンでは、大規模な Oracle ワークロードを移行するのに推奨されるネイティブの Oracle ユーティリティである Oracle Data Pump を使用して、オンプレミスの Oracle データベースを Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行します。これらのパターンには、通常、アプリケーションスキーマまたはテーブルをダンプファイルにエクスポートし、ダンプファイルを Amazon RDS for Oracle のデータベースディレクトリに転送し、ダンプファイルからアプリケーションスキーマとデータをインポートすることが含まれます。

この方法を使用すると、データのサイズや Amazon RDS インスタンスへのダンプファイルの転送にかかる時間によっては、移行に時間がかかる場合があります。さらに、ダンプファイルは Amazon RDS インスタンスの Amazon Elastic Block Store (Amazon EBS) ボリュームにあります。このボリュームは、データベースとダンプファイルを保存するのに十分な大きさでなければなりません。インポート後にダンプファイルが削除されると、空のスペースは取得できなくなるため、未使用のスペースについては引き続きお支払いいただきます。

このパターンは、データベースリンク上で Oracle Data Pump API (`DBMS_DATAPUMP`) を使用して Amazon RDS インスタンスに直接インポートを実行することで、こうした問題を軽減します。このパターンは、移行元のデータベースと移行先のデータベース間のエクスポートとインポートの同時パイプラインを開始します。このパターンでは、ダンプファイルが作成または保存されないため、ダンプファイル用の EBS ボリュームのサイズを設定する必要はありません。この方法では、未使用のディスク容量にかかる毎月のコストを節約できます。

前提条件と制限

前提条件

- アクティブな Amazon Web Services (AWS) アカウント。
- Amazon RDS インスタンスのネットワークインフラストラクチャを提供するために、少なくとも 2 つのアベイラビリティゾーンにまたがるプライベートサブネットで構成された仮想プライベートクラウド (VPC)。
- オンプレミスデータセンターの Oracle データベース、または Amazon Elastic Compute Cloud (Amazon EC2) でセルフマネージドされた Oracle データベース。

- 単一のアベイラビリティーゾーン内の既存の Amazon RDS for Oracle インスタンス。1 つのアベイラビリティゾーンを使用すると、移行中の書き込みパフォーマンスが向上します。マルチ AZ 配置は、カットオーバーの 24 ~ 48 時間前に有効化できます。

  このソリューションは、Amazon RDS Custom for Oracle をターゲットとして使用することもできます。
- AWS Direct Connect（大規模なデータベースに推奨）。
- Amazon RDS インスタンスからオンプレミスの Oracle データベースへのインバウンド接続を許可するように設定されたオンプレミスのネットワーク接続とファイアウォールルール。

制限

- Amazon RDS for Oracle のデータベースサイズ制限は、2022 年 12 月時点で 64 テビバイト (TiB) です。
- Amazon RDS for Oracle DB インスタンスの 1 つのファイルの最大サイズは 16 TiB です。これは、複数のテーブルスペースにテーブルを分散する必要がある可能性があるため、知っておくことが重要です。

製品バージョン

- ソースデータベース: Oracle データベースバージョン 10g リリース 1 以降。
- ターゲットデータベース: Amazon RDS でサポートされているバージョンとエディションの最新リストについては、AWS ドキュメントの「Amazon RDS for Oracle」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスでもクラウドでも、セルフマネージド Oracle データベース

ターゲットテクノロジースタック

- Amazon RDS for Oracle または Amazon RDS Custom for Oracle

ターゲットアーキテクチャ

以下の図は、シングル AZ 環境でオンプレミスの Oracle データベースから Amazon RDS for Oracle に移行するためのアーキテクチャを示しています。矢印の方向は、アーキテクチャ内のデータフローを示しています。この図には、どのコンポーネントが接続を開始しているかは示されていません。

1. Amazon RDS for Oracle インスタンスは、オンプレミスのソース Oracle データベースに接続し、データベースリンクを介して全負荷移行を実行します。
2. AWS Database Migration Service（ AWS DMS) はオンプレミスのソース Oracle データベースに接続し、変更データキャプチャ (CDC) を使用して継続的なレプリケーションを実行します。
3. CDC の変更は Amazon RDS for Oracle データベースに適用されます。

ツール

AWS サービス

- AWS Database Migration Service（ AWS DMS ）は、データストアを に移行する AWS クラウドか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役立ちます。このパターンでは CDC と [データ変更のみの複製] 設定を使用します。
- AWS Direct Connect は、標準イーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、ネットワークパスでインターネットサービスプロバイダーをバイパス AWS のサービス しながら、仮想インターフェイスをパブリックに直接作成できます。
- Amazon Relational Database Service は、AWS クラウドで Oracle リレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます。

その他のツール

- Oracle Data Pump を使用すると、データやメタデータをあるデータベースから別のデータベースに高速で移動できます。
- Oracle インスタントクライアントや SQL Developer などのクライアントツールを使用して、データベースに接続して SQL クエリを実行します。

ベストプラクティス

はオンプレミスネットワークと 間の専用プライベートネットワーク接続 AWS Direct Connect を使用しますが AWS、転送中のデータのセキュリティとデータ暗号化を強化するには、次のオプションを検討してください。

- を使用する仮想プライベートネットワーク (VPN) AWS Site-to-Site VPN、またはオンプレミスネットワークから AWS ネットワークへの IPsec VPN 接続
- オンプレミスの Oracle データベースに構成された Oracle Database Native Network Encryption
- TLS を使用した暗号化

エピック

オンプレミスのソース Oracle データベースを準備する

タスク	説明	必要なスキル
移行先のデータベースから移行元のデータベースへのネットワーク接続を設定します。	ターゲット Amazon RDS インスタンスからオンプレミスのソース Oracle データベースへの受信接続を許可するように、オンプレミスネットワークとファイアウォールを設定します。	ネットワーク管理者、セキュリティエンジニア

適切な権限を持つデータベースユーザーを作成します。	Oracle Data Pump を使用してソースとターゲット間でデータを移行する権限を持つオンプレミスソース Oracle データベースにデータベースユーザーを作成します。
	`GRANT CONNECT to <migration_user>;`
	`GRANT DATAPUMP_EXP_FULL_DATABASE to <migration_user>;`
		DBA

タスク	説明	必要なスキル
	`GRANT SELECT ANY TABLE to <migration_user>;`	

CDC AWS DMS 移行用にオンプレミスのソースデータベースを準備します。	（オプション）Oracle Data Pump Full Load の完了後に、CDC AWS DMS 移行用にオンプレミスソースの Oracle データベースを準備します。
	1. Oracle Data Pump の移行中に FLASHBACK を管理するために必要な追加の権限を設定します。
	`GRANT FLASHBACK ANY TABLE to <migration_user>;`
	`GRANT FLASHBACK ARCHIVE ADMINISTER to <migration_user>;`
	2. のセルフマネージド Oracle ソースに必要なユーザーアカウント権限を設定するには AWS DMS、AWS DMS ドキュメントを参照してください。
	3. を使用して CDC 用の Oracle セルフマネージド型ソースデータベースを準備するには AWS DMS、AWS DMS ドキュメントを参照してください。
		DBA

タスク	説明	必要なスキル
SQL Developer をインストールして設定します。	SQL Developer をインストールしてソースデータベースとターゲットデータベースに接続し、SQL クエリを実行するように設定します。	DBA、移行エンジニア
テーブルスペースを作成するスクリプトを生成します。	次の SQL クエリの例を使用して、ソースデータベースにスクリプトを生成します。 `SELECT 'CREATE TABLESPACE ' tablespace_name ' DATAFILE SIZE 1G AUTOEXTEND ON MAXSIZE UNLIMITED;' from dba_tablespaces where tablespace_name not in ('SYSTEM', 'SYSAUX','TEMP','UNDOTBS1') order by 1;` スクリプトはターゲットデータベースに適用されます。	DBA

タスク	説明	必要なスキル
ユーザー、プロファイル、ロール、権限を作成するためのスクリプトを生成します。	データベースユーザー、プロファイル、ロール、権限を作成するスクリプトを生成するには、OracleSupport 文書「How to Extract DDL for User including Privileges and Roles Using dbms_metadata.get_ddl (Doc ID 2739952.1)」(Oracle アカウントが必要) にあるスクリプトを使用します。 スクリプトはターゲットデータベースに適用されます。	DBA

ターゲットの Amazon RDS for Oracle インスタンスを準備します。

タスク	説明	必要なスキル
ソースデータベースへのデータベースリンクを作成し、接続を確認します。	オンプレミスのソースデータベースへのデータベースリンクを作成するには、次のコマンド例を使用できます。 `CREATE DATABASE LINK link2src CONNECT TO <migration_user_account> IDENTIFIED BY <password> USING '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>) (PORT=<listener port>))(C`	DBA

タスク	説明	必要なスキル
	ONNECT_DATA=(SID= remote SID>)))'; 接続を確認するには、次の SQL コマンドを実行します。 select * from dual@link 2src; 応答が X であれば、接続は成功です。	
スクリプトを実行してターゲットインスタンスを準備します。	以前に生成されたスクリプトを実行して、ターゲットの Amazon RDS for Oracle インスタンスを準備します。 1. テーブルスペース 2. プロファイル 3. ロール これにより、Oracle Data Pump の移行時にスキーマとそのオブジェクトを確実に作成できるようになります。	DBA、移行エンジニア

データベースリンクを介して Oracle Data Pump Import を使用して、全負荷移行を実行します。

タスク	説明	必要なスキル
必要なスキーマを移行します。	必要なスキーマをソースのオンプレミスデータベースからターゲット Amazon RDS インスタンスに移行するには、	DBA

タスク	説明	必要なスキル
	「追加情報」セクションのコードを使用してください。 • 1 つのスキーマを移行するには、「追加情報」セクションのコード 1 を実行します。 • 複数のスキーマを移行するには、「追加情報」セクションのコード 2 を実行します。 移行のパフォーマンスを調整するには、次のコマンドを実行して並列プロセスの数を調整できます。 `DBMS_DATAPUMP.SET_ PARALLEL (handle => v_hdnl, degree => 4);`	

タスク	説明	必要なスキル
スキーマ統計を収集してパフォーマンスを向上させます。	「スキーマ統計の収集」コマンドは、データベースオブジェクトに関して収集された Oracle クエリオプティマイザー統計を返します。この情報を使用することで、オプティマイザはこれらのオブジェクトに対するクエリに最適な実行プランを選択できます。 `EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS(ownname => '<schema_name>');`	DBA

Oracle Data Pump と を使用してフルロード移行と CDC レプリケーションを実行する AWS DMS

タスク	説明	必要なスキル
ソースオンプレミスの Oracle データベースで SCN を取得します。	オンプレミスの Oracle データベースのシステム変更番号 (SCN) を取得します。SCN は全ロードインポートに使用し、CDC レプリケーションの開始点としても使用します。 ソースデータベースで現在の SCN を生成するには、次の SQL ステートメントを実行します。 `SELECT current_scn FROM V$DATABASE;`	DBA

タスク	説明	必要なスキル
スキーマのフルロード移行を実行します。	必要なスキーマ (FULL LOAD) をソースのオンプレミスデータベースからターゲット Amazon RDS インスタンスに移行するには、以下を実行します。 • 1 つのスキーマを移行するには、「追加情報」セクションのコード 3 を実行します。 • 複数のスキーマを移行するには、「追加情報」セクションのコード 4 を実行します。 コードで、 をソースデータベースからキャプチャした SCN <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE> に置き換えます。 `DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value => <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>);` 移行のパフォーマンスを調整するには、並列プロセスの数を調整できます。	DBA

タスク	説明	必要なスキル
	`DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);`	
移行したスキーマのトリガーを無効にします。	CDC AWS DMS 専用タスクを開始する前に、移行されたスキーマTRIGGERSで を無効にします。	DBA
スキーマ統計を収集してパフォーマンスを向上させます。	Gather Schema Statistics コマンドは、データベースオブジェクト用に収集された Oracle クエリオプティマイザ統計を返します。 `EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS(ownname => '<schema_name>');` この情報を使用して、オプティマイザーはこれらのオブジェクトに対するあらゆるクエリに最適な実行プランを選択できます。	DBA

タスク	説明	必要なスキル
を使用して AWS DMS 、ソースからターゲットへの継続的なレプリケーションを実行します。	を使用して AWS DMS 、ソース Oracle データベースからターゲット Amazon RDS for Oracle インスタンスへの継続的なレプリケーションを実行します。 詳細については、「 を使用した継続的なレプリケーションのタスクの作成 AWS DMS」およびブログ記事「 でのネイティブ CDC サポートの操作方法 AWS DMS」を参照してください。	DBA、移行エンジニア

Amazon RDS for Oracle へのカットオーバー

タスク	説明	必要なスキル
カットオーバーの 48 時間前にインスタンスでマルチ AZ を有効にします。	本番インスタンスの場合は、Amazon RDS インスタンスでマルチ AZ 配置を有効にして、高可用性 (HA) とディザスタリカバリ (DR) のメリットを活用することをお勧めします。	DBA、移行エンジニア
CDC AWS DMS のみのタスクを停止します (CDC がオンの場合) 。	1. AWS DMS タスクの Amazon CloudWatch メトリクスのソースレイテンシーとターゲットレイテンシーが 0 秒であることを確認します。	DBA

タスク	説明	必要なスキル
	2. CDC AWS DMS のみのタスクを停止します。	
トリガーを有効にします。	CDC タスクの作成前にTRIGGERS無効にした を有効にします。	DBA

関連リソース

AWS

- を使用した CDC 用の Oracle セルフマネージド型ソースデータベースの準備 AWS DMS
- を使用した継続的なレプリケーション用のタスクの作成 AWS DMS
- 高可用性を重視したマルチ AZ 配置
- でのネイティブ CDC サポートの操作方法 AWS DMS (ブログ記事)

Oracle のドキュメント

- DBMS_DATAPUMP

追加情報

コード 1: フルロード移行のみ、単一アプリケーションスキーマ

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''<schema_name>'')');  --
 To migrate one selected schema
    DBMS_DATAPUMP.METADATA_FILTER (hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')'); --
 To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
```

```
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

コード 2: フルロード移行のみ、複数のアプリケーションスキーマ

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'SCHEMA_LIST',
 '''<SCHEMA_1>'',''<SCHEMA_2>'', ''<SCHEMA_3>'''); -- To migrate multiple schemas
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

コード 3: CDC 専用タスク、単一アプリケーションスキーマ前のフルロード移行

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''<schema_name>'')');  --
 To migrate one selected schema
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value =>
 <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>); -- SCN required for AWS DMS CDC only task.
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
```

```
/
```

コード 4: CDC 専用タスク前のフルロード移行、複数のアプリケーションスキーマ

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN (operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE (handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'SCHEMA_LIST',
 '''<SCHEMA_1>'',''<SCHEMA_2>'', ''<SCHEMA_3>'''); -- To migrate multiple schemas
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value =>
 <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>); -- SCN required for AWS DMS CDC only task.
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

混合移行アプローチの方がうまくいくシナリオ

ソースデータベースに数百万行のテーブルがあり、LOBSEGMEGT 列のサイズが非常に大きいというまれなシナリオでは、このパターンによって移行が遅くなります。Oracle では、ネットワークリンクを介して LOBSEGMENT を 1 つずつ移行します。ソーステーブルから (LOB カラムのデータとともに) 1 つの行を抽出し、その行をターゲットテーブルに挿入し、すべての行が移行されるまでこのプロセスを繰り返します。データベースリンクを介した Oracle Data Pump は、LOBSEGMENT の一括ロードまたはダイレクトパスロードのメカニズムをサポートしていません。

この状況では、次のことをお勧めします。

- 次のメタデータフィルターを追加して、Oracle Data Pump の移行中に識別されたテーブルをスキップします。

```
dbms_datapump.metadata_filter(handle =>h1, name=>'NAME_EXPR', value => 'NOT IN
 (''TABLE_1'',''TABLE_2'')');
```

- AWS DMS タスク (フルロード移行、必要に応じて CDC レプリケーション) を使用して、識別されたテーブルを移行します。 AWS DMS はソース Oracle データベースから複数の行を抽出し、ターゲット Amazon RDS インスタンスにバッチに挿入してパフォーマンスを向上させます。

Oracle E-Business Suite を Amazon RDS Custom に移行

作成者:Simon Cunningham (AWS), Jaydeep Nandy (AWS), Nitin Saxena (AWS), and Vishnu Vinnakota (AWS)

概要

Oracle E-Business Suite は、財務、人事、サプライチェーン、製造などのエンタープライズ全体のプロセスを自動化するためのエンタープライズリソースプランニング (ERP) ソリューションです。クライアント、アプリケーション、データベースという 3 層のアーキテクチャで構成されています。以前は、Oracle E-Business Suite データベースを自己管理の「Amazon Elastic Compute Cloud (Amazon EC2) インスタンス」で実行する必要がありましたが、今では、「Amazon Relational Database Service (Amazon RDS) カスタム」のメリットを享受できるようになりました。

「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。データベース管理のタスクとオペレーションを自動化し、データベース管理者としてデータベース環境とオペレーティングシステムへのアクセスおよびカスタマイズを可能にします。Oracle データベースを Amazon RDS Custom に移行すると、バックアップタスクや高可用性の確保などの面倒な作業はAmazon Web Services (AWS) が行い、ユーザーは Oracle E-Business Suite のアプリケーションと機能のメンテナンスに集中できます。移行で考慮すべき主な要素については、AWS 規範ガイダンスの「Oracle データベースの移行戦略」を参照してください。

このパターンは、Oracle Recovery Manager (RMAN) バックアップと EC2 インスタンスと Amazon RDS Custom 間の「Amazon Elastic File System (Amazon EFS)」共有ファイルシステムを使用して Amazon EC2 上のスタンドアロン Oracle データベースを Amazon RDS Custom に移行するステップに焦点を当てています。このパターンでは RMAN フルバックアップ ([レベル 0] バックアップと呼ばれることもあります) を使用します。 簡単にするために、アプリケーションをシャットダウンし、データベースをマウントして開かないコールドバックアップを使用しています。(Oracle Data Guard または RMAN の複製をバックアップに使用することもできます。ただし、このパターンではこれらのオプションは対象外です。)

高可用性とディザスタリカバリを目的として AWS で Oracle E-Business Suite を設計する方法については、「アクティブスタンバイデータベースを使用して Amazon RDS Custom で Oracle E-Business Suite の HA/DR アーキテクチャを設定する」というパターンを参照してください。

> **Note**
>
> このパターンは、Oracle サポートノートへのリンクを提供します。これらのドキュメントにアクセスするには、「Oracle Support」アカウントが必要です。

前提条件と制限

前提条件

- Oracle Linux 7 または Red Hat Enterprise Linux (RHEL) バージョン 7.x を搭載した Amazon EC2 で実行されている Oracle バージョン 12.1.0.2 または 19c (最低 19.3) のソースデータベース。このパターンでは、ソースデータベース名が `VIS`、Oracle 19c の追加のコンテナデータベース名は `VISCDB` であることを前提としていますが、他の名前も使用できます。

  > **Note**
  >
  > このパターンは、オンプレミスネットワークと Amazon Virtual Private Cloud (Amazon VPC) の間に適切なネットワーク接続がある限り、オンプレミスの Oracle ソースデータベースでも使用できます。

- Oracle E-Business Suite バージョン 12.2.x アプリケーション (ビジョンインスタンス)。この手順はバージョン 12.2.11 でテストされています。
- 単一の Oracle E-Business Suite アプリケーション層。ただし、このパターンを複数のアプリケーション層で機能するように調整できます。
- Oracle 12.1.0.2 では、Amazon RDS Custom で 16 GB 以上のスワップスペースが設定されています。そうしないと、12c Examples CD に警告が表示されます。(このドキュメントで後述するように、Oracle 19c ではサンプル CD は必要ありません)。

移行を開始する前に、以下の手順を完了します。

1. Amazon RDS Custom で、データベース名 `VIS` (またはソースデータベース名) を使用して Oracle DB 用 Amazon RDS カスタムインスタンスを作成します。手順については、AWS ドキュメントの「Amazon RDS Custom の使用」と、ブログ記事「Oracle 用 Amazon RDS カスタム — データベース環境における新しい制御機能」を参照してください。これにより、データベース名はソースデータベースと同じ名前に設定されます。(空白のままにすると、EC2 インスタンスとデータベース名は `ORCL` に設定されます)。 少なくとも、ソースに適用されたパッチを使用して「カスタ

ムエンジンバージョン (CEV)」を作成してください。詳細については、Amazon RDS ドキュメントの「CEV 作成の準備」を参照してください。

Oracle 19c に関する注意事項:現在、Oracle 19c では Amazon RDS コンテナデータベースの名前をカスタマイズできます。デフォルト: RDSCDB。RDS カスタム Oracle インスタンスは、ソース EC2 インスタンスと同じシステム ID (SID) で作成してください。たとえば、このパターンでは、Oracle 19c SID はソースインスタンスの VISCDB であると想定されます。そのため、Amazon RDS Custom 上のターゲット Oracle 19c SID も VISCDB でなければならない。

2. Amazon EC2 ソースデータベースと一致する十分なストレージ、vCPU、メモリを備えた Amazon RDS Custom DB インスタンスを設定します。 そのためには、vCPU とメモリに基づいて「Amazon EC2 インスタンスタイプ」を一致させることができます。
3. Amazon EFS ファイルシステムを作成し、Amazon EC2 と Amazon RDS Custom インスタンスにマウントする。 手順については、「Oracle 用 Amazon RDS Custom を Amazon EFS と統合する」というブログ投稿を参照してください。このパターンは、Amazon EFS ボリュームをソースの Amazon EC2 とターゲットの両方の Amazon RDS Custom DB インスタンスを /RMAN にマウントしており、ソースとターゲットの間でネットワーク接続が可能であることを前提としています。「Amazon FSx」または任意の共有ドライブを使用して同じ方法を使用することもできます。

引き受け

このパターンは、アプリケーションとデータベースが論理ホスト名を使用していることを前提としているため、移行手順の数が減ります。これらのステップは物理ホスト名を使用するように調整できますが、論理ホスト名を使用すると移行プロセスの複雑さが軽減されます。論理ホスト名を使用する利点については、以下のサポートノートを参照してください。

- 12c については、Oracle Support ノート 2246690.1
- 19c については、Oracle Support ノート 2617788.1

このパターンは Oracle 12c から 19c へのアップグレードシナリオには対応しておらず、Amazon EC2 で実行されている同じバージョンの Oracle データベースを Amazon RDS Custom フォーOracle に移行することに重点を置いています。

Amazon RDS Custom for Oracle は、「Oracle ホームのカスタマイズをサポート」しています。(Oracleホームには Oracle バイナリが格納されます)。/rdsdbbin/oracle のデフォルトパスは、/d01/oracle/VIS/19c など、指定したパスに変更できます。わかりやすくするために、このパターンの説明ではデフォルトパス /rdsdbbin/oracle を想定しています。

制約事項

このパターンは以下の機能や構成をサポートしていません。

- データベース ARCHIVE_LAG_TARGET パラメータを 60 ~ 7,200 の範囲外の値に設定します。
- DB インスタンスログモードを無効にする (NOARCHIVELOG)
- EC2 インスタンスの EBS-optimized 属性をオフにする
- EC2 インスタンスにアタッチされた元の Amazon Elastic Block Store (Amazon EBS) ボリュームを変更する
- 新しい EBS ボリュームを追加するか、ボリュームタイプを gp2 から gp3 に変更します。
- TNS ファイルのサポート
- control_file の場所と名前の変更 (/rdsdbdata/db/VISCDB_A/controlfile/control-01.ctl でなければならず、そこで、VISCDB は CDB 名である)

これらおよびその他のサポートされていない設定に関する追加情報については、Amazon RDS ドキュメントの「サポートされていない設定の修正」を参照してください。

製品バージョン

Amazon RDS カスタムがサポートする Oracle データベースのバージョンとインスタンスクラスについては、「Oracle 用 Amazon RDS Custom の可用性と要件」を参照してください。

アーキテクチャ

次のアーキテクチャ図は、AWS の単一の「アベイラビリティーゾーン」で実行されている Oracle E-Business Suite システムを表しています。「アプリケーション層にはApplication Load Balancer」を介してアクセスされ、アプリケーションとデータベースはどちらもプライベートサブネットにあります。Amazon RDS Custom および Amazon EC2 データベース層は Amazon EFS 共有ファイルシステムを使用して RMAN バックアップファイルを保存し、アクセスします。

ツール

AWS サービス

- 「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。データベース管理のタスクとオペ

レーションを自動化し、データベース管理者としてデータベース環境とオペレーティングシステムへのアクセスおよびカスタマイズを可能にします。

- 「Amazon Elastic File System (Amazon EFS)」は、シンプルでサーバーレスの、伸縮自在なファイルシステムです。管理やプロビジョニングは必要ありません。このパターンでは、Amazon EFS 共有ファイルシステムを使用して RMAN バックアップファイルを保存し、アクセスします。
- 「AWS Secrets Manager」は、データベース認証情報、API キー、およびその他のシークレット情報を簡単にローテーション、管理、および取得できるようにする AWS マネージドサービスです。Amazon RDS Custom は、データベースの作成時にkey pair とデータベースユーザー認証情報をSecrets Manager に保存します。このパターンでは、Secrets Manager からデータベースユーザーパスワードを取得して、`RDSADMIN` および `ADMIN` ユーザーを作成し、システムパスワードとシステムパスワードを変更します。

その他のツール

- RMAN は Oracle データベースのバックアップとリカバリのサポートを提供するツールです。このパターンでは、RMAN を使用して Amazon EC2 上のソース Oracle データベースのコールドバックアップを実行し、Amazon RDS Custom で復元します。

ベストプラクティス

- 論理ホスト名を使用してください。これにより、クローニング後に実行する必要のあるスクリプトの数が大幅に減ります。詳細については、「Oracle サポートノート 2246690.1」を参照してください。
- Amazon RDS Custom はデフォルトで Oracle「自動メモリ管理」(AMM) を使用します。 hugemem カーネルを使用したい場合は、代わりに自動共有メモリ管理 (ASMM) を使用するように Amazon RDS Custom を設定できます。
- `memory_max_target` パラメータはデフォルトでは有効のままにしておきます。フレームワークはこのパラメータをバックグラウンドで使用してリードレプリカを作成します。
- Oracle フラッシュバックデータベースを有効にします。この機能は、(スイッチオーバーではなく) フェイルオーバーのテストシナリオでスタンバイ状態に戻す場合に役立ちます。
- データベース初期化パラメータについては、Oracle ソースデータベースの SPFILE を使用する代わりに、Oracle E-Business Suite 用の Amazon RDS Custom DB インスタンスによって提供される標準 PFILE をカスタマイズします。これは、Amazon RDS Custom でリードレプリカを作成する際に空白やコメントが原因で問題が発生するためです。データベース初期化パラメータの詳細については、「Oracle Support ノート 396009.1」を参照してください。

次の「エピック」セクションでは、Oracle 12.1.0.2 と 19c の手順を個別に説明していますが、それぞれ詳細が異なります。

エピック

ソースアプリケーションをシャットダウンします

タスク	説明	必要なスキル

アプリケーションをシャットダウンします。	ソースアプリケーションをシャットダウンするには、以下のコマンドを使用します。
	`$ su - applmgr`
	`$ cd $INST_TOP/admin/scripts`
	`$ ./adstpall.sh`
		DBA

.zip ファイルを作成します。	ソースアプリケーション層に `appsutil.zip` ファイルを作成します。このファイルを使用して、後で Amazon RDS Custom データベースノードを設定します。 `$ perl $AD_TOP/bin/admkappsutil.pl`	DBA
.zip ファイルを Amazon EFS にコピーします。	`appsutil.zip` を `$INST_TOP/admin/out` から共有 Amazon EFS ボリューム (`/RMAN/appsutil`) にコピーします。セキュアコピー (SCP) または別の転送メカニズムを使用してファイルを手動で転送できます。	DBA

ソースデータベースを事前にクローニングします。

タスク	説明	必要なスキル

Amazon EC2 のデータベース層を事前にクローニングします。	Oracle ユーザーとしてログインし、以下を実行します。
	`$ cd $ORACLE_HOME/appsutil/scripts/$CONTEXT_NAME`
	`$ perl adpreclone.pl dbTier`
	生成されたログファイルを確認して、操作が正常に完了したことを確認します。
		DBA

appsutil.zip を共有の Amazon EFS ファイルシステムへコピーします。	tar バックアップを作成し、`$ORACLE_HOME/appsutil` を共有 Amazon EFS ファイルシステム (例:`/RMAN/appsutil`) にコピーします。
	`$ cd $ORACLE_HOME`
	`$ tar cvf sourceappsutil.tar appsutil`
	`$ cp sourceappsutil.tar /RMAN/appsutil`
		DBA

ソース Amazon EC2 データベースのコールド RMAN フルバックアップを実行します

タスク	説明	必要なスキル
バックアップスクリプトを作成します。	ソースデータベースの RMAN 完全バックアップを共有の	DBA

タスク	説明	必要なスキル

	Amazon EFS ファイルシステムに実行します。
	
	簡単にするために、このパターンではコールド RMAN バックアップを実行します。ただし、これらの手順を変更して Oracle Data Guard でホット RMAN バックアップを実行することで、ダウンタイムを短縮できます。
	
	1. ソース Amazon EC2 データベースをマウントモードで起動します。
	
	`$ sqlplus / as sysdba`
	`$ SQL> shutdown immediate`
	`$ SQL> startup mount`
	
	2. RMAN バックアップスクリプトを作成して (使用している Oracle のバージョンに応じて以下の例を使用するか、既存の RMAN スクリプトのいずれかを実行します)、(この例では /RMAN) マウントした Amazon EFS ファイルシステムにデータベースをバックアップします。
	
	Oracle 12.1.0.2 の場合
	
	`$ vi FullRMANColdBackup.sh`
	`#!/bin/bash`	

タスク	説明	必要なスキル

	. /home/oracle/.bash
	_profile
	
	export ORACLE_SID=VIS
	export ORACLE_HOME=/
	d01/oracle/VIS/12.1.0
	export DATE=$(date +
	%y-%m-%d_%H%M%S)
	
	rman target / log=/RMAN
	/VISDB_${DATE}.log <<
	EOF
	run
	{
	allocate channel ch1
	device type disk format
	'/RMAN/visdb_full_
	bkp_%u';
	allocate channel ch2
	device type disk format
	'/RMAN/visdb_full_
	bkp_%u';
	crosscheck backup;
	delete noprompt
	obsolete;
	BACKUP AS COMPRESSED
	BACKUPSET DATABASE PLUS
	ARCHIVELOG;
	backup archivelog all;
	release channel ch1;
	release channel ch2;
	}
	EOF
	
	Oracle 19c の場合:
	
	$ vi FullRMANColdBackup
	.sh
	#!/bin/bash
	. /home/oracle/.bash
	_profile	

タスク	説明	必要なスキル

	export ORACLE_SI
	D=VISCDB
	export ORACLE_HOME=/
	d01/oracle/VIS/19c
	export DATE=$(date +
	%y-%m-%d_%H%M%S)
	
	rman target / log=/RMAN
	/VISDB_${DATE}.log <<
	EOF
	run
	{
	allocate channel ch1
	device type disk format
	'/RMAN/visdb_full_
	bkp_%u';
	allocate channel ch2
	device type disk format
	'/RMAN/visdb_full_
	bkp_%u';
	crosscheck backup;
	delete noprompt
	obsolete;
	BACKUP AS COMPRESSED
	BACKUPSET DATABASE PLUS
	ARCHIVELOG;
	backup archivelog all;
	backup current
	controlfile format '/
	RMAN/cntrl.bak';
	release channel ch1;
	release channel ch2;
	}
	EOF	

タスク	説明	必要なスキル

バックアップスクリプトを実行します。	権限を変更し、Oracle ユーザーとしてログインし、スクリプトを実行します。
	
	`$  chmod 755 FullRMANColdBackup.sh`
	`$ ./FullRMANColdBackup.sh`
		DBA

タスク	説明	必要なスキル

エラーを確認し、バックアップファイル名をメモします。	RMAN ログファイルにエラーがないか確認します。すべて問題なければ、制御ファイルのバックアップをリストアップしてください。出力ファイルの名前に注意してください。
	
	Oracle 12.1.0.2 の場合
	
	`RMAN> connect target /`
	
	`RMAN> list backup of controlfile;`
	
	`BS Key Type LV Size Device Type Elapsed Time Completion Time`
	`------- ---- -- ---------- ----------- ------------ ---------- --- ------------ ---------- ---------- ---------- ---------- --- ------------ ---------- ---------- ---------- ---------- --- ------------ ---------- ---------- ----------

タスク	説明	必要なスキル

	kp_100rlsbt は、後で Amazon RDS Custom でデータベースを復元するときに使用します。
	
	Oracle 19c の場合:
	
	`RMAN> connect target /`
	
	`RMAN> list backup of controlfile;`
	
	`BS Key Type LV Size Device Type Elapsed Time Completion Time`
	`------- ---- -- ---------- --------- ---------- ---------- ---

ターゲット Amazon RDS Custom データベースを設定します。

タスク	説明	必要なスキル

ホストファイルを変更し、ホスト名を設定します。	**Note**
	このセクションのコマンドは、ルートユーザーとして実行する必要があります。
	
	1. Amazon RDS Custom DB インスタンスの `/etc/hosts` ファイルを編集します。これを行う簡単な方法は、ソース Amazon EC2 データベースホストファイルからデータベースとアプリケーションホストのエントリをコピーすることです。
	
	`<IP-address> OEBS-app01.localdomain OEBS-app01 OEBS-app01log.localdomain OEBS-app01log`
	`<IP-address> OEBS-db01.localdomain OEBS-db01 OEBS-db01log.localdomain OEBS-db01log`
	
	`<IP-address>` はデータベースノードの IP アドレスで、Amazon RDS Custom IP アドレスに置き換える必要があります。論理ホスト名には `*log` が付加されます。
		DBA

タスク	説明	必要なスキル

	2. 以下の hostnamectl コマンドを実行してデータベースのホスト名を変更します。
	`$ sudo hostnamectl set-hostname --static persistent-hostname`
	以下に例を示します。
	`$ sudo hostnamectl set-hostname --static OEBS-db01log`
	追加情報については、「Knowledge Centerの静的ホスト名の割り当て」に関する記事を参照してください。
	3. Amazon RDS Custom DB インスタンスを再起動します。 データベースは後のステップで削除するので、シャットダウンの心配はありません。
	`$ reboot`
	4. Amazon RDS Custom DB インスタンスが復旧したら、ログインしてホスト名が変更されていることを確認します。
	`$ hostname`	

タスク	説明	必要なスキル
	`oebs-db01`	

Oracle E-Business Suite ソフトウェアをインストールします。	Oracle E-Business Suite 推奨の RPM を Amazon RDS Custom DB インスタンス上のOracleのホームロケーションにインストールします。詳細については、「Oracle Support ノート #1330701.1」を参照してください。以下に、その一部を示します。RPM リストはリリースごとに変わるため、必要な RPM がすべてインストールされていることを確認してください。
	
	ルートユーザーとして以下を実行します。
	
	`$ sudo yum -y update`
	`$ sudo yum install -y elfutils-libelf-devel*`
	`$ sudo yum install -y libXp-1.0.2-2.1*.i686`
	`$ sudo yum install -y libXp-1.0.2-2.1*`
	`$ sudo yum install -y compat-libstdc++-*`
	
	次のステップに進む前に、必要なパッチがすべてインストールされていることを確認してください。
		DBA

タスク	説明	必要なスキル

VNC サーバーをインストールします。	**Note**
	サンプル CD が不要になったため、Oracle 19c ではこのステップを省略できます。「Oracle サポートノート 2782085.1」を参照してください。
	
	Oracle 12.1.0.2 の場合
	
	VNC サーバーとそれに依存するデスクトップパッケージをインストールします。これは、次のステップで 12c Examples CD をインストールするための要件です。
	
	1. ルートユーザーとして以下を実行します。
	
	`$ sudo yum install -y tigervnc-server`
	`$ sudo yum install -y *kde*`
	`$ sudo yum install -y *xorg*`
	
	2. rdsdb ユーザーの VNC サーバーを起動し、VNC のパスワードを設定します。
	
	`$ su - rdsdb`
	`$ vncserver :1`
		DBA

タスク	説明	必要なスキル

	``
	$ vncpassword
	``	

タスク	説明	必要なスキル
12c サンプル CD をインストールします。	**Note** サンプル CD が不要になったため、Oracle 19c ではこのステップを省略できます。「Oracle サポートノート 2782085.1」を参照してください。 Oracle 12.1.0.2 の場合 1. インストールファイルを「https://edelivery.oracle.com/」からダウンロードします。 Oracle E-Business Suite 12.2.11 — Oracle Database 12c Release 1 (12.1.0.2) については、Linux x86-64 V100102-01.zip を探してください。 2. サンプル CD を格納するディレクトリを作成します。 `$ mkdir /RMAN/12c examples` 3. 任意の転送メカニズム (SCP など) を使用して Examples CD .zip ファイルをこのディレクトリにコピーします。	DBA

タスク	説明	必要なスキル

	`V100102-01.zip`
	
	4. 所有権を rdsdb に変更します。
	
	`$ chown -R rdsdb:rdsdb /RMAN/12cexamples`
	
	5. rdsdb ユーザーとして、ファイルを解凍します。
	
	`$ unzip V10010201.zip`
	
	6. VNC クライアントと Amazon RDS Customにアクセスできるクライアントから接続します。VNC へのアクセスを許可するために必要なネットワーク接続とファイアウォールポートが開いていることを確認してください。たとえば、display :1 で実行中の VNC サーバーでは、Amazon RDS Custom EC2 ホストに関連付けられたセキュリティグループでポート 5901 を開く必要があります。
	
	7. Examples CD をコピーしたディレクトリに移動します。
	
	`$ cd /RMAN/12cexamples/examples`	

タスク	説明	必要なスキル

	8. インストーラーを実行します。 oraInst.loc ファイルの場所を必ず確認してください。
	`./runInstaller -invPtrLoc /rdsdbbin/oracle.12.1.custom.r1.EE.1/oraInst.loc`
	9. Examples CD のインストール時には、以下のパラメーターを使用してください。
	`Skip Software Updates`
	`Select Oracle Home 12.1.0.2 (Oracle Base = /rdsdbbin)`
	`(Software Location = /rdsdbbin/oracle/12.1.custom.r1.EE.1)`
	10. インストールプログラムには 5 つのステップとプロンプトが含まれています。 インストールが完了するまで手順に従ってください。	

スターターデータベースを削除し、データベースファイルを保存するディレクトリを作成します。

タスク	説明	必要なスキル
自動化モードを一時停止します。	自動化が RMAN アクティビティに干渉しないように、	DBA

タスク	説明	必要なスキル
	次のステップに進む前に Amazon RDS Custom DB インスタンスの「自動化モード」を一時停止する必要があります。 次の AWS コマンドラインインターフェイス(AWS CLI) コマンドを使用して、自動化を一時停止します。(最初に「AWS CLI を設定」したことを確認してください)。 `aws rds modify-db-instance \ --db-instance-id entifier VIS \ --automation-mode all-paused \ --resume-full-au tomation-mode-minute 360 \ --region eu-west-1` 一時停止時間を指定するときは、RMAN の復元に十分な時間を確保してください。これはソースデータベースのサイズによって異なるため、360 の値を適宜変更してください。	

タスク	説明	必要なスキル

スターターデータベースをドロップします。	既存の Amazon RDS Custom データベースを削除します。
	
	Oracle ホームユーザーとして、以下のコマンドを実行します。(カスタマイズしていない限り、デフォルトユーザーは `rdsdb` です)。
	
	`$ sqlplus / as sysdba`
	`SQL> shutdown immediate ;`
	`SQL> startup nomount restrict;`
	`SQL> alter database mount;`
	`SQL> drop database;`
	`SQL> exit`
		DBA

タスク	説明	必要なスキル

データベースファイルを保存するディレクトリを作成します。	Oracle 12.1.0.2 の場合
	
	データベース、制御ファイル、データファイル、オンラインログ用のディレクトリを作成します。前のコマンドの `control_files` パラメーターの親ディレクトリ (この場合は `VIS_A`) を使用します。Oracle ホームユーザ (デフォルトでは `rdsdb`) として以下のコマンドを実行します。
	
	`$ mkdir -p /rdsdbdata/db/VIS_A/controlfile`
	`$ mkdir -p /rdsdbdata/db/VIS_A/datafile`
	`$ mkdir -p /rdsdbdata/db/VIS_A/onlinelog`
	
	Oracle 19c の場合:
	
	データベース、制御ファイル、データファイル、オンラインログ用のディレクトリを作成します。前のコマンドの `control_files` パラメーターの親ディレクトリ (この場合は `VISCDB_A`) を使用します。Oracle ホームユーザ (デフォルトでは `rdsdb`) として以下のコマンドを実行します。
		DBA

タスク	説明	必要なスキル

	`$ mkdir -p /rdsdbdata/db/cdb/VISCDB_A/controlfile`
	`$ mkdir -p /rdsdbdata/db/cdb/VISCDB_A/datafile`
	`$ mkdir -p /rdsdbdata/db/cdb/VISCDB_A/onlinelog`
	`$ mkdir -p /rdsdbdata/db/cdb/VISCDB_A/onlinelog/arch`
	`$ mkdir /rdsdbdata/db/pdb/VISCDB_A`	

タスク	説明	必要なスキル
Oracle E-Business Suite のパラメータファイルを作成して変更します。	このステップでは、サーバーパラメータファイル (SPFILE) をソースデータベースからコピーしません。代わりに、Amazon RDS Custom DB インスタンスで作成された標準パラメータファイル (PFILE) を使用して、Oracle E-Business Suite に必要なパラメータを追加します。 データベースをドロップすると、Amazon RDS 自動化によって `init.ora` ファイルのバックアップが作成され、Amazon RDS Custom データベースに関連付けられます。このファイルは `oracle_pfile` という名前で、`/rdsdbdata/config` にあります。 Oracle 12.1.0.2 の場合 1. `/rdsdbdata/config/oracle_pfile` を `$ORACLE_HOME` にコピーします。 `$ cp /rdsdbdata/config/oracle_pfile $ORACLE_HOME/dbs/initVIS.ora` 2. Amazon RDS Custom DB インスタンスの `initVIS.ora` ファイルを編集します。	DBA

タスク	説明	必要なスキル

	ソースのすべてのパラメータを検証し、必要に応じてパラメータを追加します。詳細については、「Oracle Support ノート 396009.1」を参照してください。
	
	Important
	追加するパラメータにコメントがないことを確認します。コメントがあると、リードレプリカの作成やポイントインタイムリカバリ (PITR) の発行など、自動化で問題が発生します。
	
	3. 要件に応じて、次のようなパラメータを `initVIS.ora` ファイルに追加します。
	
	`*.workarea_size_policy='AUTO'`
	`*.plsql_code_type='INTERPRETED'`
	`*.cursor_sharing='EXACT'`
	`*._b_tree_bitmap_plans=FALSE`
	`*.session_cached_cursors=500`
	`*.optimizer_adaptive_features=false`	

タスク	説明	必要なスキル
	*.optimizer_secur e_view_merging=false *.SQL92_SECURITY=TRUE *.temp_undo_enabled= true _system_trig_enabled = TRUE nls_language = american nls_territory = america nls_numeric_charact ers = "." nls_comp = binary nls_sort = binary nls_date_format = DD- MON-RR nls_length_semantics = BYTE aq_tm_processes = 1 _sort_eliminatio n_cost_ratio =5 _like_with_bind _as_equality = TRUE _fast_full_scan_enable d = FALSE _b_tree_bitmap_plans = FALSE optimizer_secure_view _merging = FALSE _optimizer_autostats_ job = FALSE parallel_max_servers = 8 parallel_min_servers = 0 parallel_degree_policy = MANUAL sec_case_sensitive_l ogon = FALSE compatible = 12.1.0 o7_dictionary_access ibility = FALSE	

タスク	説明	必要なスキル

	`utl_file_dir =/tmp`
	
	4. 次のように修正します。値はソースシステムによって異なるため、現在の設定に基づいて変更してください。
	
	`*.open_cursors=500`
	`*.undo_tablespace`
	`='APPS_UNDOTS1`
	
	5. SPFILE リファレンスを削除してください。
	
	`*.spfile='/rdsdbbi`
	`n/oracle/dbs/spfil`
	`eVIS.ora'`
	
	注意:
	
	• `control_files` と `db_unique_name` については、Amazon RDS Custom プロファイルによって提供されているとの値は変更しないでください。Amazon RDS はこれらの値を想定しています。これらから逸脱すると、future リードレプリカを作成しようとしたときに問題が発生します。
	• Amazon RDS Custom はデフォルトで「A自動メモリ管理 (AMM)」を使用します。hugemem を使用した	

タスク	説明	必要なスキル

	い場合は、自動共有メモリ管理 (ASMM) を使用するように Amazon RDS Custom を設定できます。
	• memory_max_target パラメータはデフォルトでは有効のままにしておきます。Amazon RDS フレームワークはこれをバックグラウンドで使用してリードレプリカを作成します。
	
	6. 以下の startup nomount コマンドを実行して、initVIS.ora ファイルに問題がないことを確認します。
	`SQL> startup nomount pfile=/rdsdbbin/oracle/dbs/initVIS.ora;`
	`SQL> create spfile='/rdsdbdata/admin/VIS/pfile/spfileVIS.ora' from pfile;`
	`SQL> exit`
	
	7. SPFILE のシンボリックリンクを作成します。
	`$ ln -s /rdsdbdata/admin/VIS/pfile/spfileVIS.ora $ORACLE_HOME/dbs/`
	
	Oracle 19c の場合:	

タスク	説明	必要なスキル
	1. /rdsdbdata/config/oracle_pfile を $ORACLE_HOME にコピーします。 `$ cp /rdsdbdata/config/oracle_pfile $ORACLE_HOME/dbs/initVISCDB.ora` 2. Amazon RDS Custom DB インスタンスの initVISCDB.ora ファイルを編集します。ソースのすべてのパラメータを検証し、必要に応じてパラメータを追加します。詳細については、「Oracle Support ノート 396009.1」を参照してください。 ⚠ Important 追加するパラメータにコメントがないことを確認します。 コメントがあると、リードレプリカの作成やポイントインタイムリカバリ (PITR) の発行など、自動化の問題の原因になります。 3. 要件に応じて、次のようなパラメータを initVISCD	

タスク	説明	必要なスキル
	B.ora ファイルに追加します。 *.instance_name=VISCDB *.sec_case_sensitive_logon= FALSE *.result_cache_max_size = 600M *.optimizer_adaptive_plans =TRUE *.optimizer_adaptive_statistics = FALSE *.pga_aggregate_limit = 0 *.temp_undo_enabled = FALSE *._pdb_name_case_sensitive = TRUE *.event='10946 trace name context forever, level 8454144' *.workarea_size_policy='AUTO' *.plsql_code_type='INTERPRETED' *.cursor_sharing='EXACT' *._b_tree_bitmap_plans=FALSE *.session_cached_cursors=500 *.optimizer_secure_view_merging=false *.SQL92_SECURITY=TRUE _system_trig_enabled = TRUE nls_language = american nls_territory = america	

タスク	説明	必要なスキル
	nls_numeric_charact ers = "." nls_comp = binary nls_sort = binary nls_date_format = DD- MON-RR nls_length_semantics = BYTE aq_tm_processes = 1 _sort_eliminatio n_cost_ratio =5 _like_with_bind _as_equality = TRUE _fast_full_scan_enable d = FALSE _b_tree_bitmap_plans = FALSE optimizer_secure_view _merging = FALSE _optimizer_autostats_ job = FALSE parallel_max_servers = 8 parallel_min_servers = 0 parallel_degree_policy = MANUAL 4. 次のように修正します。値はソースシステムによって異なるため、現在の設定に基づいて変更してください。 *.open_cursors=500 *.undo_tablespace ='UNDOTBS1' 5. SPFILE リファレンスを削除します。	

タスク	説明	必要なスキル
	`*.spfile='/rdsdbbin/oracle/dbs/spfileVISCDB.ora'` 注意: • `control_files` と `db_unique_name` については、Amazon RDS Custom プロファイルによって提供されているとの値は変更しないでください。Amazon RDS はこれらの値を想定しています。これらから逸脱すると、future リードレプリカを作成しようとしたときに問題が発生します。 • Amazon RDS Custom はデフォルトで「A自動メモリ管理 (AMM)」を使用します。hugemem を使用したい場合は、自動共有メモリ管理 (ASMM) を使用するように Amazon RDS Custom を設定できます。 • `memory_max_target` パラメータはデフォルトでは有効のままにしておきます。Amazon RDS フレームワークはこれをバックグラウンドで使用してリードレプリカを作成します。	

タスク	説明	必要なスキル

	6. 以下の `startup nomount` コマンドを実行して、`initVISCDB.ora` ファイルに問題がないことを確認します。
	`SQL> startup nomount pfile=/rdsdbbin/oracle/dbs/initVISCDB.ora;`
	`SQL> create spfile='/rdsdbdata/admin/VISCDB/pfile/spfileVISCDB.ora' from pfile;`
	`SQL> exit`
	7. SPFILE のシンボリックリンクを作成します。
	`$ ln -s /rdsdbdata/admin/VISCDB/pfile/spfileVISCDB.ora $ORACLE_HOME/dbs/`	

タスク	説明	必要なスキル

バックアップから Amazon RDS Custom データベースを復元します。	Oracle 12.1.0.2 の場合
	
	1. 先にソースでキャプチャしたバックアップファイルを使用して、コントロールファイルを復元します。
	
	`RMAN> connect target /`
	`RMAN> RESTORE CONTROLFILE FROM '/RMAN/visdb_full_bkp_100rl sbt';`
	
	`Starting restore at 10-APR-22`
	`using target database control file instead of recovery catalog`
	`allocated channel: ORA_DISK_1`
	`channel ORA_DISK_1: SID=201 device type=DISK`
	
	`channel ORA_DISK_1: restoring control file`
	`channel ORA_DISK_1: restore complete, elapsed time: 00:00:01`
	`output file name=/rdsdbdata/db/VIS_A/controlfile/control-01.ctl`
	`Finished restore at 10-APR-22`
	
	2. バックアップピースをカタログ化し、RMAN `restore` を発行できるようにします。
		DBA

タスク	説明	必要なスキル

	RMAN> alter database
	 mount;
	RMAN> catalog start
	 with '/RMAN/visdb';
	
	3. データベースを復元するスクリプトを作成します。
	
	$ vi restore.sh
	rman target / log=/home
	/rdsdb/rman.log << EOF
	run
	{
	set newname for database
	 to '/rdsdbdata/db/VIS
	_A/datafile/%b';
	restore database;
	switch datafile all;
	switch tempfile all;
	}
	EOF
	
	4. ソースをターゲット Amazon RDS Custom データベースに復元します。スクリプトの実行を許可するようにスクリプトの権限を変更し、restore.sh スクリプトを実行してデータベースを復元する必要があります。
	
	$ chmod 755 restore.sh
	$ nohup ./restore.sh &
	
	Oracle 19c の場合:	

タスク	説明	必要なスキル

	1. 先にソースでキャプチャしたバックアップファイルを使用して、コントロールファイルを復元します。
	
	`RMAN> connect target /`
	`RMAN> RESTORE CONTROLFILE FROM '/RMAN/cntrl.bak';`
	`Starting restore at 07-JUN-23`
	`using target database control file instead of recovery catalog`
	`allocated channel: ORA_DISK_1`
	`channel ORA_DISK_1: SID=201 device type=DISK`
	`channel ORA_DISK_1: restoring control file`
	`channel ORA_DISK_1: restore complete, elapsed time: 00:00:01`
	`output file name=/rdsdbdata/db/cdb/VISCDB_A/controlfile/control-01.ctl`
	`Finished restore at 07-JUN-23`
	
	2. バックアップピースをカタログ化し、RMAN `restore` を発行できるようにします。
	
	`RMAN> alter database mount;`
	`RMAN> catalog start with '/RMAN/visdb';`	

タスク	説明	必要なスキル

	start with コマンドで問題が発生した場合は、バックアップピースを個別に追加できます。例:
	
	`RMAN> catalog backuppiece '/RMAN/visdb_full_bkp_1d1e507m';`
	
	バックアップピースごとにコマンドを繰り返します。
	
	3. データベースを復元するスクリプトを作成します。要件に応じて、プラガブルデータベース名を修正します。使用可能なvCPUs の数に基づいて並行チャネルを割り当てて、リストアプロセスを高速化します。
	
	`$ vi restore.sh`
	`rman target / log=/home/rdsdb/rmancdb.log <<`
	`EOF`
	`run {`
	`allocate channel c1`
	`type disk;`
	`allocate channel c2`
	`type disk;`
	`.... .... ....`
	`allocate channel c<N>`
	`type disk;`
	`set newname for database`
	`to '/rdsdbdata/db/cdb/VISCDB_A/datafile/`
	`%b';`	

タスク	説明	必要なスキル

	set newname for database root to '/rdsdbdata/db/cdb/VISCDB_A/datafile/%f_%b';
	set newname for database "PDB$SEED" to '/rdsdbdata/db/cdb/pdbseed/%f_%b';
	set newname for pluggable database VIS to '/rdsdbdata/db/pdb/VISCDB_A/%f_%b';
	restore database;
	switch datafile all;
	switch tempfile all;
	release channel c1;
	release channel c2;
	release channel c3;
	....
	release channel c<N>;
	}
	EOF
	
	4. ソースをターゲット Amazon RDS Custom データベースに復元します。スクリプトの実行を許可するようにスクリプトの権限を変更し、restore.sh スクリプトを実行してデータベースを復元する必要があります。
	
	$ chmod 755 restore.sh
	$ nohup ./restore.sh &	

タスク	説明	必要なスキル

ログファイルに問題がないか確認してください。	Oracle 12.1.0.2 の場合
	
	1. rman.log ファイルを確認して、問題がないことを確認します。
	
	`$ cat /home/rdsdb/rman.log`
	
	2. コントロールファイルに登録されているログファイルのパスを確認します。
	
	`SQL> select member from v$logfile;`
	`MEMBER`
	`----------`
	`/d01/oracle/VIS/data/log1.dbf`
	`/d01/oracle/VIS/data/log2.dbf`
	`/d01/oracle/VIS/data/log3.dbf`
	
	3. ログファイルの名前を、ターゲットのファイルパスと一致するように変更します。前のステップで出力されたものと一致するためにパスを置き換えます。
	
	`SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/data/log1.`
		DBA

タスク	説明	必要なスキル
	dbf' TO '/rdsdbdata/db/VIS_A/onlinelog/log1.dbf'; SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/data/log2.dbf' TO '/rdsdbdata/db/VIS_A/onlinelog/log2.dbf'; SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/data/log3.dbf' TO '/rdsdbdata/db/VIS_A/onlinelog/log3.dbf'; Oracle 19c の場合: 1. rmancdb.log ファイルを確認して、問題がないことを確認します。 $ cat /home/rdsdb/rmancdb.log 2. コントロールファイルに登録されているログファイルのパスを確認します。 SQL> select member from v$logfile; MEMBER ---------- ---------- ---------- ---------- --------	

タスク	説明	必要なスキル
	/d01/oracle/VIS/oradata/VISCDB/redo03.log /d01/oracle/VIS/oradata/VISCDB/redo02.log /d01/oracle/VIS/oradata/VISCDB/redo01.log 3. ログファイルの名前を、ターゲットのファイルパスと一致するように変更します。前のステップで出力されたものと一致するためにパスを置き換えます。 SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/oradata/VISCDB/redo01.log' TO '/rdsdbdata/db/cdb/VISCDB_A/onlinelog/log1.dbf'; SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/oradata/VISCDB/redo02.log' TO '/rdsdbdata/db/cdb/VISCDB_A/onlinelog/log2.dbf'; SQL> ALTER DATABASE RENAME FILE '/d01/oracle/VIS/oradata/VISCDB/redo03.log' TO '/rdsdbdata/db/cdb/VISCDB_A/onlinelog/log3.dbf';	

タスク	説明	必要なスキル

	4. 制御ファイルに登録されているパス、ログファイルのステータス、およびグループ番号を確認します。
	
	`SQL> column REDOLOG_FILE_NAME format a50`
	`SQL> SELECT a.GROUP#, a.status, b.MEMBER AS REDOLOG_FILE_NAME, (a.BYTES/1024/1024) AS SIZE_MB`
	`FROM v$log a`
	`JOIN v$logfile b ON a.Group#=b.Group#`
	`ORDER BY a.GROUP#;`
	
	`GROUP# STATUS REDOLOG_FILE_NAME SIZE_MB`
	`1 CURRENT /rdsdbdata/db/cdb/VISCDB_A/onlinelog/log1.dbf 512`
	`2 INACTIVE /rdsdbdata/db/cdb/VISCDB_A/onlinelog/log2.dbf 512`
	`3 INACTIVE /rdsdbdata/db/cdb/VISCDB_A/onlinelog/log3.dbf 512`	

タスク	説明	必要なスキル
Amazon RDS Custom データベースを開いて OMF ログファイルを作成できることを確認します。	Oracle 用 Amazon RDS Custom は「Oracle マネージドファイル」(OMF) を使用して操作を簡素化します。リードレプリカはスタンドアロンインスタンスに昇格できますが、最初に OMF を使用してログファイルを作成する必要があります。これは、インスタンスの昇格時に正しいパスが使用されるようにするためです。リードレプリカを昇格させる方法の詳細については、「Amazon RDS ドキュメント」を参照してください。OMF ファイルを使用しないと、リードレプリカをプロモートしようとしたときに問題が発生する可能性があります。 1. `resetlogs` でデータベースを開きます。 `SQL> alter database open resetlogs;` **Note** ORA-00392: スレッド 1 のログ xx がクリアされており、オペレーションが許可されていない場合は、ORA-00392 のトラブ	DBA

タスク	説明	必要なスキル

	ルシューティングセクションのステップに従います。
	
	2. データベースが開いていることを確認します。
	
	`SQL> select open_mode from v$database;`
	`OPEN_MODE`
	`--------------------`
	`READ WRITE`
	
	3. OMF ログファイルを作成します。前のログファイルクエリの出力を使用して、要件に応じてグループ番号、グループ数、サイズを変更します。次の例はグループ 4 から始まり、わかりやすくするために 3 つのグループを追加しています。
	
	`SQL> alter database add logfile group 4 size 512M;`
	`Database altered.`
	`SQL> alter database add logfile group 5 size 512M;`
	`Database altered.`
	`SQL> alter database add logfile group 6 size 512M;`
	`Database altered.`	

タスク	説明	必要なスキル

	4. 以前の OMF 以外のファイルを削除します。要件と前のステップのクエリの出力に基づいてカスタマイズできる例を次に示します。
	
	`SQL> alter database drop logfile group 1;`
	`System altered.`
	`SQL> alter database drop logfile group 2;`
	`System altered.`
	`SQL> alter database drop logfile group 3;`
	`System altered.`
	
	Note
	ログファイルを削除しようとしたときに ORA-01624 エラーが発生した場合は、「トラブルシューティング」セクションを参照してください。
	
	5. 作成された OMF ファイルが表示されることを確認します。(ディレクトリパスは Oracle 12.1.0.2 と 19c では異なりますが、概念は同じです)。
	
	`SQL> select member from v$logfile;`	

タスク	説明	必要なスキル
	MEMBER ---------- ---------- ---------- /rsdbdata/db/cdb/ VISCDB_A/onlinelog/ o1_mf_4_ksrbslny_.log /rsdbdata/db/cdb/VIS CDB_A/onlinelog/o1 _mf_5_ksrchw0k_.log /rsdbdata/db/cdb/ VISCDB_A/onlinelog/ o1_mf_6_ksrcnl9v_.log 6. データベースを再起動し、SPFILE がインスタンスで使用されていることを確認します。 SQL> shutdown immediate SQL> startup SQL> show parameter spfile Oracle 12.1.0.2 の場合、このクエリは次の値を返します。 spfile /rsdbbin /oracle/dbs/spfile VIS.ora Oracle 19c の場合、クエリは次の値を返します。 spfile /rsdbbin /oracle/dbs/spfile VISCDB.ora	

タスク	説明	必要なスキル

	7. [Oracle 19c の場合のみ]、コンテナデータベースのステータスを確認し、必要に応じて開きます。
	
	`SQL> show pdbs`
	`CON_ID CON_NAME`
	`OPEN MODE`
	`RESTRICTED`
	`---------- ---------`
	`---------- ---------`
	`-`
	`2 PDB$SEED`
	`READ ONLY NO`
	`3 VIS`
	`MOUNTED NO`
	
	`SQL> alter session set`
	`container=VIS;`
	`Session altered.`
	
	`SQL> alter database`
	`open;`
	`Database altered.`
	
	`SQL> alter database save`
	`state;`
	`Database altered.`
	
	`SQL> show pdbs`
	`CON_ID CON_NAME`
	`OPEN MODE`
	`RESTRICTED`
	`---------- ---------`
	`----------`
	`---------- -----`
	`3 VIS`
	`READ`
	`WRITE NO`	

タスク	説明	必要なスキル

	`SQL> exit`
	
	8. PFILE を使用していないため、$ORACLE_HOME/dbs から init.ora ファイルを削除してください。
	
	`$ cd $ORACLE_HOME/dbs`
	
	Oracle 12.1.0.2 の場合は、次のコマンドを使用します。
	
	`$ pwd`
	`/rdsdbbin/oracle/dbs`
	`$ rm initVIS.ora`
	
	Oracle 19c の場合は、次のコマンドを使用してください。
	
	`$ pwd`
	`/rdsdbbin/oracle/dbs`
	`$ rm initVISCDB.ora`	

Secrets Manager からのパスワードの取得、ユーザーの作成、パスワードの変更

タスク	説明	必要なスキル
Secrets Manager からパスワードを取得します。	これらのステップは、コンソールで、または AWS CLI を使用して実行できます。次の手順は、コンソールの手順を提供します。 1. AWS マネジメントコンソールにサインインし、Amazon	DBA

タスク	説明	必要なスキル
	RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。 2. ナビゲーションペインで、[データベース] を選択し、Amazon RDS データベースを選択します。 3. [設定] を選択し、インスタンスのリソース ID をメモします (この形式は: db-WZ4WLCK6AOQ6TJGZKMGRCDCI3Y になります)。 4. AWS Secrets Manager のコンソールを「https://console.aws.amazon.com/secretsmanager/」で開きます。 5. do-not-delete-custom-<resource_id> と同じ名前のシークレットを選択します。ここで、resource-id はステップ 3 で書き留めたインスタンスの ID を指します。 6. [シークレットの値を取得する] を選択します。	

タスク	説明	必要なスキル

RDSADMIN ユーザーを作成します。	RDSADMIN は、Amazon RDS Custom DB インスタンスのモニタリングおよびオーケストレーターデータベースユーザーです。スターターデータベースは削除され、ターゲットデータベースは RMAN を使用してソースから復元されたため、Amazon RDS Custom モニタリングが期待どおりに動作するように、復元操作後にこのユーザーを再作成する必要があります。また、RDSADMIN ユーザー用に別のプロファイルとテーブルスペースを作成する必要があります。Oracle 12.1.0.2 と 19c では、手順が少し異なります。
	
	Oracle 12.1.0.2 の場合
	
	1. SQL プロンプトで次のコマンドを入力します。
	
	`SQL> set echo on`
	`feedback on serverout`
	`on`
	`SQL> @?/rdbms/admin/utl`
	`pwdmg.sql`
	
	`SQL> ALTER PROFILE`
	`DEFAULT`
	`LIMIT`
	`FAILED_LOGIN_ATTEMPTS`
	`UNLIMITED`
		DBA

タスク	説明	必要なスキル
	PASSWORD_LIFE_TIME UNLIMITED PASSWORD_VERIFY_F UNCTION NULL; 2. プロフィール RDSADMIN を作成します。 SQL> create profile RDSADMIN LIMIT COMPOSITE_LIMIT UNLIMITED SESSIONS_PER_USER UNLIMITED CPU_PER_SESSION UNLIMITED CPU_PER_CALL UNLIMITED LOGICAL_READS_PER _SESSION UNLIMITED LOGICAL_READS_PER_CALL UNLIMITED IDLE_TIME UNLIMITED CONNECT_TIME UNLIMITED PRIVATE_SGA UNLIMITED FAILED_LOGIN_ATTEMPTS 10 PASSWORD_LIFE_TIME UNLIMITED PASSWORD_REUSE_TIME UNLIMITED PASSWORD_REUSE_MAX UNLIMITED PASSWORD_VERIFY_F UNCTION NULL PASSWORD_LOCK_TIME 86400/86400 PASSWORD_GRACE_TIME 604800/86400;	

タスク	説明	必要なスキル

	3. SYS、SYSTEM、および DBSNMP ユーザープロファイルを RDSADMIN に設定します。
	
	`SQL> set echo on`
	`feedback on serverout`
	`on`
	`SQL> alter user SYS`
	`profile RDSADMIN;`
	`SQL> alter user SYSTEM`
	`profile RDSADMIN;`
	`SQL> alter user DBSNMP`
	`profile RDSADMIN;`
	
	4. RDSADMIN テーブルスペースを作成します。
	
	`SQL> create bigfile`
	`tablespace rdsadmin`
	`datafile size 7M`
	`autoextend on next 1m`
	`Logging online`
	`permanent blocksize`
	`8192 extent managemen`
	`t local autoallocate`
	`default nocompress`
	`segment space managemen`
	`t auto;`
	
	5. RDSADMIN ユーザーを作成します。RDSADMIN パスワードを、以前に Secrets Manager から取得したパスワードに置き換えます。	

タスク	説明	必要なスキル

	`SQL> create user rdsadmin identified by xxxxxxxxxx Default tablespace rdsadmin Temporary tablespace temp profile rdsadmin ;`
	
	6. RDSADMIN に権限を付与します。
	
	`SQL> grant select on sys.v_$instance to rdsadmin; SQL> grant select on sys.v_$archived_log to rdsadmin; SQL> grant select on sys.v_$database to rdsadmin; SQL> grant select on sys.v_$database_incarnation to rdsadmin; SQL> grant select on dba_users to rdsadmin; SQL> grant alter system to rdsadmin; SQL> grant alter database to rdsadmin; SQL> grant connect to rdsadmin with admin option; SQL> grant resource to rdsadmin with admin option; SQL> alter user rdsadmin account unlock identified by xxxxxxxxxxx;`	

タスク	説明	必要なスキル
	SQL> @?/rdbms/admin/use rlock.sql SQL> @?/rdbms/admin/utl rp.sql	
	Oracle 19c の場合: 1. SQL プロンプトで次のコマンドを入力します。 SQL> set echo on feedback on serverout on SQL> @?/rdbms/admin/utl pwdmg.sql SQL> alter profile default LIMIT FAILED_LOGIN_ATTEMPTS UNLIMITED PASSWORD_LIFE_TIME UNLIMITED PASSWORD_VERIFY_F UNCTION NULL; 2. プロフィール RDSADMIN を作成します。 **Note** RDSADMIN Oracle 19c C##では のプレフィックスがあります。これは、データベースパラメータcommon_user_prefix が に設定されているためで	

タスク	説明	必要なスキル
	すC##。 RDSADMIN には Oracle 12.1.0.2 のプレフィックスがありません。 SQL> create profile C##RDSADMIN LIMIT COMPOSITE_LIMIT UNLIMITED SESSIONS_PER_USER UNLIMITED CPU_PER_SESSION UNLIMITED CPU_PER_CALL UNLIMITED LOGICAL_READS_PER _SESSION UNLIMITED LOGICAL_READS_PER_CALL UNLIMITED IDLE_TIME UNLIMITED CONNECT_TIME UNLIMITED PRIVATE_SGA UNLIMITED FAILED_LOGIN_ATTEMPTS 10 PASSWORD_LIFE_TIME UNLIMITED PASSWORD_REUSE_TIME UNLIMITED PASSWORD_REUSE_MAX UNLIMITED PASSWORD_VERIFY_F UNCTION NULL PASSWORD_LOCK_TIME 86400/86400 PASSWORD_GRACE_TIME 604800/86400;	

タスク	説明	必要なスキル

	3. SYS、SYSTEM、および DBSNMP ユーザープロファイルを RDSADMIN に設定します。
	
	`SQL> alter user SYS profile C##RDSADMIN;`
	`SQL> alter user SYSTEM profile C##RDSADMIN;`
	`SQL> alter user DBSNMP profile C##RDSADMIN;`
	
	4. RDSADMIN テーブルスペースを作成します。
	
	`SQL> create bigfile tablespace rdsadmin datafile size 7M autoextend on next 1m Logging online permanent blocksize 8192 extent management local autoallocate default nocompress segment space management auto;`
	
	5. RDSADMIN ユーザーを作成します。RDSADMIN パスワードを、以前に Secrets Manager から取得したパスワードに置き換えます。
	
	`SQL> create user C##rdsadmin identified by xxxxxxxxxx profile C##rdsadmin container=all;`	

タスク	説明	必要なスキル
	6. RDSADMIN に権限を付与します。 SQL> grant select on sys.v_$instance to c##rdsadmin; SQL> grant select on sys.v_$archived_log to c##rdsadmin; SQL> grant select on sys.v_$database to c##rdsadmin; SQL> grant select on sys.v_$database_incarnation to c##rdsadmin; SQL> grant select on dba_users to c##rdsadmin; SQL> grant alter system to C##rdsadmin; SQL> grant alter database to C##rdsadmin; SQL> grant connect to C##rdsadmin with admin option; SQL> grant resource to C##rdsadmin with admin option; SQL> alter user C##rdsadmin account unlock identified by xxxxxxxxxxx; SQL> @?/rdbms/admin/userlock.sql SQL> @?/rdbms/admin/utlrp.sql	

タスク	説明	必要なスキル

マスターユーザーを作成します。	スターターデータベースは削除され、ターゲットデータベースは RMAN を使用してソースから復元されたため、マスターユーザーを再作成する必要があります。この例では、ユーザーは admin という名前になります。
	
	Oracle 12.1.0.2 の場合
	
	`SQL> create user admin identified by <password>; SQL> grant dba to admin`
	
	Oracle 19c の場合:
	
	`SQL> alter session set container=VIS;`
	
	`Session altered.`
	
	`SQL> create user admin identified by <password>;`
	
	`User created.`
	
	`SQL> grant dba to admin;`
	
	`Grant succeeded.`
		DBA

タスク	説明	必要なスキル

スーパーユーザーのパスワードを変更します。	1. Secrets Manager から取得したパスワードを使用して、システムパスワードを変更します。
	
	Oracle 12.1.0.2 の場合
	
	`SQL> alter user sys identified by xxxxxxxxxx;`
	`SQL> alter user system identified by xxxxxxxxxx;`
	
	Oracle 19c の場合:
	
	`SQL> alter user sys identified by xxxxxxxxxx container =all;`
	`SQL> alter user system identified by xxxxxxxxxx container =all;`
	
	1. EBS_SYSTEM パスワードを変更します。
	
	Oracle 12.1.0.2 の場合
	
	`SQL> alter user ebs_system identified by xxxxxxxxxx;`
	
	Oracle 19c の場合:
		DBA

タスク	説明	必要なスキル

	このバージョンでは、コンテナデータベースにも接続して、そこで EBS_SYSTEM パスワードを更新する必要があります。
	
	`SQL> alter session set container=vis;`
	`SQL> alter user ebs_system identified by xxxxxxxxxx;`
	
	`SQL> exit;`
	
	これらのパスワードを変更しない場合、Amazon RDS Custom は [データベースモニタリングユーザーまたはユーザー認証情報が変更されました] というエラーメッセージを表示します。	

Oracle E-Business Suite 用のディレクトリの作成、ETCC のインストール、オートコンフィグの実行

タスク	説明	必要なスキル
Oracle E-Business Suite に必要なディレクトリを作成します。	1. Amazon RDS Custom Oracle データベースで、Oracle ホームユーザーとして次のスクリプトを実行して、`$ORACLE_HOME/nls/data/9idata` の `9idata` ディレクトリを作成します。	

タスク	説明	必要なスキル

	このディレクトリは Oracle E-Business Suite に必要です。
	
	`perl $ORACLE_HOME/nls/data/old/cr9idata.pl`
	
	コンテキスト対応環境は後のステップで作成するので、ORA_NLS10 メッセージは無視してください。
	
	2. 共有の Amazon EFS ファイルシステムから先に作成した appsutil.tar ファイルをコピーし、Amazon RDS Custom Oracle ホームディレクトリに解凍します。 これで appsutil ディレクトリが $ORACLE_HOME ディレクトリ内に作成されます。
	
	`$ cd /RMAN/appsutil`
	`$ cp sourceappsutil.tar $ORACLE_HOME`
	`$ cd $ORACLE_HOME`
	`$ tar xvf sourceappsutil.tar appsutil`
	
	3. Amazon EFS 共有ファイルシステムに以前に保存した appsutil.zip ファイルをコピーします。 これはアプリケーション層で作成したファイルです。	

タスク	説明	必要なスキル

	Amazon RDS Custom DB インスタンスの rdsdb ユーザーとして:
	`$ cp /RMAN/appsutil/appsutil.zip $ORACLE_HOME`
	`$ cd $ORACLE_HOME`
	4. appsutil.zip ファイルを解凍して、Oracle ホームディレクトリに appsutil ディレクトリとサブディレクトリを作成します。
	`$ unzip -o appsutil.zip`
	-o のオプションを指定すると、一部のファイルが上書きされます。	

タスク	説明	必要なスキル

tsanames.ora ファイルと sqlnet.ora ファイルを設定します。	Autoconfig ツールでデータベースに接続できるように tnsnames.ora ファイルを設定する必要があります。次の例では、tnsnames.ora ファイルはソフトリンクされているが、デフォルトでは空であることがわかります。
	
	`$ cd $ORACLE_HOME/network/admin`
	`$ ls -ltr`
	`-rw-r--r-- 1 rdsdb database 373 Oct 31 2013 shrept.lst`
	`lrwxrwxrwx 1 rdsdb database 30 Feb 9 17:17 listener.ora -> /rdsdbdata/config/listener.ora`
	`lrwxrwxrwx 1 rdsdb database 28 Feb 9 17:17 sqlnet.ora -> /rdsdbdata/config/sqlnet.ora`
	`lrwxrwxrwx 1 rdsdb database 30 Feb 9 17:17 tnsnames.ora -> /rdsdbdata/config/tnsnames.ora`
	
	1. tnsnames.ora エントリを作成します。Amazon RDS の自動化によるファイルの解析方法により、エントリに空白、コメント、余分な行が含まれていないことを確認する必要があります。そうしな
		DBA

タスク	説明	必要なスキル

	いと、「create-db-instance-read-replica」などの一部の API を使用する際に問題が発生する可能性があります。次の例のコードを使用します。
	
	2. 必要に応じてポート、ホスト、SID を交換してください。
	
	`$ vi tnsnames.ora`
	`VIS=(DESCRIPTION=`
	`(ADDRESS_LIST=(ADD`
	`RESS=(PROTOCOL=TCP`
	`)(PORT=1521)(HOST=`
	`xx.xx.xx.xx)))(CON`
	`NECT_DATA=(SID=VIS)`
	`(SERVER=DEDICATED)))`
	
	Note
	ファイルに余分な行があってはなりません。行を削除しないと、future リードレプリカを作成する際に問題が発生する場合があります。リードレプリカの作成は、[アクティビティスロー例外: HostManagerException:どのホストでも RestrictReplication を正常に呼び出せません] というエラーメッ	

タスク	説明	必要なスキル

	セージで失敗することがあります。
	
	3. データベースにアクセスできることを確認します。
	
	`$ tnsping vis`
	`OK (0 msec)`
	
	4. [Oracle 19c の場合のみ]、`sqlnet.ora` ファイルを更新してください。これを行わないと、ORA-01017 というエラーが発生します。ユーザー名またはパスワードが無効で、データベースに接続しようとするとログオンが拒否されます。`$ORACLE_HOME/network/admin` の `sqlnet.ora` を以下と一致するように編集してください。
	
	`NAMES.DIRECTORY_PATH=(TNSNAMES, ONAMES, HOSTNAME)`
	`SQLNET.EXPIRE_TIME= 10`
	`SQLNET.INBOUND_CONNECT_TIMEOUT =60`
	`SQLNET.ALLOWED_LOGON_VERSION_SERVER=10`
	`HTTPS_SSL_VERSION=undetermined`
	
	5. 接続をテストします。	

タスク	説明	必要なスキル
	`$ sqlplus apps/****@vis`	

タスク	説明	必要なスキル

データベースを設定します。	データベースへの接続をテストしたので、appsutil ユーティリティを使用してデータベースを構成し、コンテキスト対応環境を作成できます。
	
	Oracle 12.1.0.2 の場合
	
	1. 以下の コマンドを実行します。
	
	`$ cd $ORACLE_HOME/appsutil/bin`
	`$ perl adbldxml.pl appsuser=apps`
	`Enter Hostname of Database server: oebs-db01`
	`Enter Port of Database server: 1521`
	`Enter SID of Database server: VIS`
	`Enter Database Service Name: VIS`
	`Enter the value for Display Variable: :1`
	`The context file has been created at: /rdsdbbin/oracle/appsutil/VIS_oebs-db01.xml`
	
	2. ルートユーザーから `oraInst.loc` を作成:
	
	`$ vi /etc/oraInst.loc`
	`inventory_loc=/rdsdbbin/oracle.12.1.c`
		DBA

タスク	説明	必要なスキル
	ustom.r1.EE.1/oraI nventory inst_group=database	
	3. 前の手順で作成したコンテキストファイルを使用して、コンテキストファイルをクローンして論理ホスト名を設定します。rdsdb ユーザーとして以下を実行します。 $ cd $ORACLE_HOME/appsu til/clone/bin $ perl adclonectx.pl \ contextfile=[ORA CLE_HOME]/appsutil/ [current context file] \ template=[ORACLE _HOME]/appsutil/te mplate/adxdbctx.tmp ここで、oebs-db01log は論理ホスト名を指します。以下に例を示します。 $ perl adclonectx.pl \ contextfile=/rdsdbbin/ oracle.12.1.custom.r1 .EE.1/appsutil/VIS _oebs-db01.xml \ template=/rdsdbbin/ oracle/appsutil/ template/adxdbctx.tmp Target System Hostname (virtual or normal) [oebs-db01] : oebs- db01log	

タスク	説明	必要なスキル
	Target System Base Directory : /rdsdbbin/ oracle Target Instance is RAC (y/n) [n] : n Target System Database SID : VIS Oracle OS User [rdsdb] : Oracle OS Group [rdsdb] : database Role separation is supported y/n [n] ? : n Target System utl_file_ dir Directory List : / tmp Number of DATA_TOP's on the Target System [1] : Target System DATA_TOP Directory 1 [/rdsdbbi n/oracle/data] : / rdsdbdata/db/VIS_A/ datafile/ Target System RDBMS ORACLE_HOME Directory [/rdsdbbin/oracle/ 12.1.0] : /rdsdbbin/ oracle Do you want to preserve the Display [:1] (y/n) : y Do you want the target system to have the same port values as	

タスク	説明	必要なスキル
	the source system (y/n) [y] ? : y The new database context file has been created : /rdsdbbin/oracle.12.1.custom.r1.EE.1/appsutil/clone/bin/VIS_oebs-db01log.xml contextfile=/rdsdbbin/oracle.12.1.custom.r1.EE.1/appsutil/clone/bin/VIS_oebs-db01log.xml	
	Oracle 19c の場合: 1. 以下の コマンドを実行します。 $ cd $ORACLE_HOME/appsutil/bin $ perl adbldxml.pl appsuser=apps Enter Hostname of Database server: oebs-db01 Enter Port of Database server: 1521 Enter SID of Database server: VIS Enter the database listener name:L_VISCDB_001 Enter the value for Display Variable: :1 The context file has been created at: /rdsdbbin/oracle/appsutil/VIS_oebs-db01.xml	

タスク	説明	必要なスキル

	2. ルートユーザーから oraInst.loc を作成:
	`$ vi /etc/oraInst.loc`
	`inventory_loc=/rdsdbbin/oracle/oraInventory`
	`inst_group=database`
	3. 前の手順で作成したコンテキストファイルを使用して、コンテキストファイルをクローンして論理ホスト名を設定します。rdsdb ユーザーとして以下を実行します。
	`$ cd $ORACLE_HOME/appsutil/clone/bin`
	`$ perl adclonectx.pl \`
	`contextfile=[ORACLE_HOME]/appsutil/[current context file] \`
	`template=[ORACLE_HOME]/appsutil/template/adxdbctx.tmp`
	ここで、oebs-db01log は論理ホスト名を指します。以下に例を示します。
	`$ perl adclonectx.pl \`
	`contextfile=/rdsdbbin/oracle/appsutil/VIS_oebs-db01.xml \`
	`template=/rdsdbbin/oracle/appsutil/template/adxdbctx.tmp`	

タスク	説明	必要なスキル
	Target System Hostname (virtual or normal) [oebs-db01] : oebs-db01log Target System Base Directory : /rdsdbbin/oracle Target Instance is RAC (y/n) [n] : n Target System CDB Name : VISCDB Target System PDB Name : VIS Oracle OS User [oracle] : rdsdb Oracle OS Group [dba] : database Role separation is supported y/n [n] ? : n Number of DATA_TOP's on the Target System [2] : Target System DATA_TOP Directory 1 [/d01/oracle/VISCDB] : /rdsdbdata/db/pdb/VISCDB_A Target System DATA_TOP Directory 2 [/d01/oracle/data] : /rdsdbdata/db/pdb/VISCDB_A/datafile Specify value for OSBACKUPDBA group [database] : Specify value for OSDGDBA group [database] : Specify value for OSKMDBA group [database] :	

タスク	説明	必要なスキル
	Specify value for OSRACDBA group [database] : Target System RDBMS ORACLE_HOME Directory [/d01/oracle/19.0. 0] : /rdsdbbin/oracle Do you want to preserve the Display [:1] (y/n) : y Do you want the target system to have the same port values as the source system (y/n) [y] ? : y Validating if the source port numbers are available on the target system.. Complete port informati on available at / rdsdbbin/oracle/a ppsutil/clone/bin/ out/VIS_oebs-db01log/ portpool.lst New context path and file name [VIS_oebs -db01log.xml] : / rdsdbbin/oracle/a ppsutil/VIS_oebs-d b01log.xml Do you want to overwrite it (y/n) [n] ? : y Replacing /rdsdbbin /oracle/appsutil/V IS_oebs-db01log.xml file. The new database context file has been created : contextfile=/rdsdbbin/ oracle/appsutil/VIS_o ebs-db01log.xml	

タスク	説明	必要なスキル
	`Check Clone Context logfile /rdsdbbin/oracle/appsutil/clone/bin/CloneContext_0609141428.log for details.`	

タスク	説明	必要なスキル

ETCC をインストールしてオートコンフィグを実行する。	1. Oracle E-Business Suite テクノロジー・コードレベル・チェッカー (ETCC) をインストールします。
	
	「My Oracle Support」からパッチ 17537119 をダウンロードし、README.txt の指示に従ってください。$ORACLE_HOME ディレクトリに etcc という名前のディレクトリを作成し、パッチを解凍して checkMTpatch.sh というスクリプトを作成し、そのスクリプトを実行してパッチバージョンを確認します。
	
	2. Autoconfig ユーティリティを実行し、新しい論理ホスト名コンテキストファイルを渡します。
	
	Oracle 12.1.0.2 の場合
	
	`cd $ORACLE_HOME/appsutil/bin`
	`$ ./adconfig.sh contextfile=/rdsdbbin/oracle.12.1.customstom.r1.EE.1/appsutil/clone/bin/VIS_oebs-db01log.xml`
	
	Oracle 19c の場合:
		DBA

タスク	説明	必要なスキル

	Autoconfig はリスナー名が CDBNAME と一致することを想定しています。そのため、バックアップされた元のリスナー設定ファイルは一時的に L_<CDBNAME>_001 が使用されます。
	
	`$ lsnrctl stop L_VISCDB_001`
	`$ cp -rp /rdsdbdata/config/listener.ora /rdsdbdata/config/listener.ora_orig`
	`$ vi /rdsdbdata/config/listener.ora`
	`:%s/L_VISCDB_001/VISCDB/g`
	`$ lsnrctl start VISCDB`
	`$ cd /rdsdbbin/oracle/appsutil`
	`$ . ./txkSetCfgCDB.env dboraclehome=/rdsdbbin/oracle.19.custom.r1.EE-CDB.1`
	
	`Oracle Home being passed: /rdsdbbin/oracle`
	
	`$ echo $ORACLE_HOME`
	`/rdsdbbin/oracle.19.custom.r1.EE-CDB.1`
	`$ export ORACLE_SID=VISCDB`
	`$ cd $ORACLE_HOME/appsutil/bin`
	`$ perl $ORACLE_HOME/appsutil/bin/t`	

タスク	説明	必要なスキル
	xkPostPDBCreationTasks.pl -dboraclehome=$ORACLE_HOME -outdir=$ORACLE_HOME/appsutil/log -cdbsid=VISCDB -pdbsid=VIS -appsuser=apps -dbport=1521 -servicetype=onpremise Enter the APPS Password: <apps password> Enter the CDB SYSTEM Password:<password from secrets manager> **Note** データベースディレクトリが変更されている場合は、Oracle サポートノート 2525754.1 の指示に従ってください。	

Amazon RDS Custom と Oracle E-Business Suite の TNS エントリを設定します。

タスク	説明	必要なスキル
Amazon RDS Custom と Oracle E-Business Suite の TNS エントリを設定します。	自動設定により TNS ファイルがデフォルトの場所に生成されます。Oracle 12.1.0.2 (CDB 以外) と Oracle19c PDB のデフォルトの場所は	DBA

タスク	説明	必要なスキル
	$ORACLE_HOME/network/admin/$<CONTEXT_NAME> です。Oracle 19c 用 CDB は、前のステップでオートコンフィグを実行したときに生成された環境ファイルの $TNS_ADMIN で定義されているデフォルト $ORACLE_HOME/network/admin/ を使用します。 Oracle 12.1.0.2 と 19c CDB では、Autoconfig によって生成される tnsnames.ora および listener.ora ファイルは、ホワイトスペースやコメントがないなどの Amazon RDS の要件を満たしていないため、これらは使用しません。代わりに、Amazon RDS Custom データベースに付属する汎用ファイルを使用して、システムが期待している内容への準拠を確保し、エラーの許容範囲を減らします。 たとえば、Amazon RDS Custom は次のような命名形式を想定しています。 `L_<INSTANCE_NAME>_001` Oracle 12.1.0.2 の場合、次のようになります。	

タスク	説明	必要なスキル

	`L_VIS_001`
	
	Oracle 19c の場合、次のようになります。
	
	`L_VISCDB_001`
	
	使用する `listener.ora` ファイルの例を以下に示します。これは Amazon RDS Custom データベースを作成したときに生成されました。この時点では、このファイルには変更を加えていないため、デフォルトのままにします。
	
	Oracle 12.1.0.2 の場合
	
	`$ cd $ORACLE_HOME/network/admin`
	`$ cat listener.ora`
	`ADR_BASE_L_VIS_001=/rdsdbdata/log/`
	`SID_LIST_L_VIS_001=(SID_LIST =`
	`(SID_DESC = (SID_NAME = VIS)(GLOBAL_DBNAME = VIS) (ORACLE_HOME = /rdsdbbin/oracle)))`
	`L_VIS_001=(DESCRIPTION_LIST =`
	`(DESCRIPTION =`
	`(ADDRESS = (PROTOCOL = TCP)(PORT = 1521)(HOST = xx.xx.xx.xx))) (DESCRIPTION =`
	`(ADDRESS = (PROTOCOL =`	

タスク	説明	必要なスキル

	TCP)(PORT = 1521)(HOST = 127.0.0.1))))
	SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_VIS_001=OFF
	
	[Oracle 19c の場合]:元の listener.ora ファイルをリスナー名 L_<INSTANCE_NAME>_001 で復元します。
	
	$ cd $ORACLE_HOME/network/admin
	
	$ cp -rp /rdsdbdata/config/listener.ora /rdsdbdata/config/listener.ora_autoconfig
	$ cp -rp /rdsdbdata/config/listener.ora_orig /rdsdbdata/config/listener.ora
	
	$ cat listener.ora
	SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_VISCDB_001=OFF
	ADR_BASE_L_VISCDB_001=/rdsdbdata/log/
	USE_SID_AS_SERVICE_L_VISCDB_001=ON
	L_VISCDB_001=(DESCRIPTION_LIST = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = 1521)(HOST = xx.xx.xx.xx))) (DESCRIPTION = (ADDRESS = (PROTOCOL =	

タスク	説明	必要なスキル
	TCP)(PORT = 1521)(HOST = 127.0.0.1)))) SID_LIST_L_VISCDB_001= (SID_LIST = (SID_DESC = (SID_NAME = VISCDB)(GLOBAL_DBNAME = VISCDB) (ORACLE_HOME = /rdsdbbin/oracle)))	

	Amazon RDS の標準オペレーション用にリスナー L_<INSTANCE_NAME>_001 を起動します。
	$ lsnrctl stop
	$ lsnrctl start L_VISCDB_001	

	Oracle 12.1.0.2 の場合 Oracle E-Business Suite 環境ファイルを編集して、Amazon RDS Custom ジェネリック TNS ファイルを使用するように $TNS_ADMIN パスを変更します。この環境ファイルは、以前にオートコンフィグを実行したときに作成されたものです。TNS_ADMIN ポストフィックスを削除して <CONTEXT_NAME> 変数を編集します。 Note 19c のデフォルトのホームは であ	

タスク	説明	必要なスキル

	り、Amazon RDS Custom のデフォルトと同じであるため$ORACLE_HOME/network/admin、環境ファイルは Oracle 12.1.0.2 でのみ編集する必要があります。
	
	たとえば、Oracle 12.1.0.2 では、以下のファイルを編集します。
	
	`$ vi $ORACLE_HOME/VIS_oebs-db01log.env`
	
	パスを次のように変更します。
	
	`TNS_ADMIN=”/rdsdbbin/oracle/network/admin/VIS_oebs-db01log”`
	`export TNS_ADMIN`
	
	変更後:
	
	`TNS_ADMIN=”/rdsdbbin/oracle/network/admin”`
	`export TNS_ADMIN`
	
	Note
	Autoconfig を実行するたびに、このステップ	

タスク	説明	必要なスキル
	を繰り返して、正しい TNS イフェイルが使用されていることを確認する必要があります。(12.1.0.2 のみ)。 Oracle 19c の場合: 1. データベース層のコンテキスト変数 s_cdb_tnsadmin の値を、<ORACLE_HOME>/network/admin/<CONTEXT_NAME> ではなく <ORACLE_HOME>/network/admin に変更します。 Note s_db_tnsadmin コンテキスト変数を更新しないでください。<ORACLE_HOME>/network/admin/<CONTEXT_NAME> のままにしておきます。 $. $ORACLE_HOME/VIS_oebs-db01log.env $ vi $CONTEXT_FILE	

タスク	説明	必要なスキル

	2. s_cdb_tnsadmin の値に に加えた変更を保存します。
	
	s_db_tnsadmin と s_cdb_tnsadmin の値は次のようになるはずです。PDB 名は VIS、データベースノードの論理名は oebs-db01log です。
	
	`$ grep -i tns_admin $CONTEXT_FILE <TNS_ADMIN oa_var="s_db_tnsadmin">/rdsdbbin/oracle/network/admin/VIS_oebs-db01log</TNS_ADMIN> <CDB_TNS_ADMIN oa_var="s_cdb_tnsadmin">/rdsdbbin/oracle/network/admin</CDB_TNS_ADMIN>`
	
	3. データベース層で Autoconfig を実行します。
	
	`$ . $ORACLE_HOME/VISCDB_oebs-db01log.env $ export ORACLE_PDB_SID=VIS $ sqlplus "/ as sysdba" @$ORACLE_HOME/appsutil/admin/adgrants.sql APPS $ sqlplus "/ as sysdba" @$ORACLE_HOME/rdbms/admin/utlrp.sql`	

タスク	説明	必要なスキル
	``` $. $ORACLE_HOME/VIS_o ebs-db01log.env $ echo $ORACLE_SID VIS $ cd $ORACLE_HOME/appsu til/scripts/$CONTE XT_NAME $ ./adautocfg.sh ```	

タスク	説明	必要なスキル

rdsdb ユーザーの環境を設定します。	Oracle 19c の場合は、このステップをスキップします。
	
	Oracle 12.1.0.2 の場合
	
	Autoconfig と TNS のエントリが完了したので、環境ファイルを rdsdb ユーザーのプロファイルに設定してロードする必要があります。
	
	Oracle E-Business Suite データベース .env ファイルを呼び出すように .bash_profile を更新します。環境がロードされていることを確認するには、プロファイルを更新する必要があります。この環境ファイルは、以前に Autoconfig を実行したときに作成されたものです。
	
	Autoconfig を実行すると、次のサンプル環境ファイルが作成されます。
	
	`. /rdsdbbin/oracle/VIS_oebs-db01log.env`
	
	rdsdb ユーザーとして:
	
	`cd $HOME`
	`vi .bash_profile`
	`export LD_LIBRARY_PATH=${ORACLE_HOME}/lib:${ORACLE_HOME}/ctx/lib`
		DBA

タスク	説明	必要なスキル
	`export SHLIB_PATH= ${ORACLE_HOME}/lib export PATH=$PATH: ${ORACLE_HOME}/bin alias sql='rlwrap -c sqlplus / as sysdba' . ${ORACLE_HOME}/VIS _oebs-db01log.env` **Note** Oracle 19c の場合、で CDB 環境をロードする必要はありません.bash_profile 。これは、デフォルト ORACLE_HOME が rdsdb (Oracle ホーム) ユーザーのデフォルトのホームであるデフォルトパス $ORACLE_HOME/network/admin に設定されているためです。	

タスク	説明	必要なスキル
Amazon RDS Custom 用にアプリケーションとデータベースを設定します。	Oracle 12.1.0.2 と 19c の両方で最初の 2 つのステップを実行します。以降のステップはバージョンごとに異なります。 1. アプリケーション層で `/etc/hosts` を編集し、データベースの IP アドレスを Amazon RDS Custom IP アドレスに変更します。 `xx.xx.xx.xx OEBS-db01.localdomain OEBS-db01 OEBS-db01log.localdomain OEBS-db01log` 論理ホスト名を使用しているため、データベースノードをほぼシームレスに置き換えることができます。 2. Amazon RDS Custom DB インスタンスでは、ソース EC2 インスタンスに割り当てられているセキュリティグループを Amazon RDS Custom DB インスタンスを反映するように追加または修正し、アプリケーションがノードにアクセスできるようにします。 Oracle 12.1.0.2 の場合	DBA

タスク	説明	必要なスキル

	3. 自動設定を実行します。アプリ所有者 (例:applmgr) として、以下を実行します。
	`$ cd $INST_TOP/admin/scripts`
	`$ ./adautocfg.sh`
	`AutoConfig completed successfully.`
	4. fnd_nodes エントリを確認します。
	`SQL> select node_name from apps.fnd_nodes`
	`NODE_NAME`
	`----------`
	`AUTHENTICATION`
	`OEBS-APP01LOG`
	`OEBS-DB01LOG`
	5. ログインできることを確認し、アプリケーションを起動します。
	`$ ./adstrtal.sh`
	Oracle 19c の場合:
	1. PDB が開いているかどうかを確認し、必要に応じて開きます。	

タスク	説明	必要なスキル
	SQL> show pdbs CON_ID CON_NAME OPEN MODE RESTRICTED ---------- --------- ---------- ---------- ---------- 2 PDB$SEED READ ONLY NO 3 VIS MOUNTED SQL> alter session set container=vis; SQL> alter database open; SQL> alter database save state; 2. apps として接続をテストします。 SQL> sqlplus apps/**** @vis 3. データベース層で Autoconfig を実行します。 $. $ORACLE_HOME/VIS_o ebs-db01log.env $ echo $ORACLE_SID VIS $ cd $ORACLE_HOME/appsu til/scripts/$CONTE XT_NAME $ ./adautocfg.sh	

タスク	説明	必要なスキル

	4. アプリケーション層で Autoconfig をアプリケーション所有者として実行します (例: applmgr)。
	`$ cd $INST_TOP/admin/scripts`
	`$ ./adautocfg.sh`
	`AutoConfig completed successfully.`
	5. fnd_nodes エントリを確認します。
	`SQL> select node_name from apps.fnd_nodes`
	`NODE_NAME`
	`----------`
	`AUTHENTICATION`
	`OEBS-APP01LOG`
	`OEBS-DB01LOG`
	6. アプリケーションを起動します。
	`$ ./adstrtal.sh`	

移行後の手順を実行する

タスク	説明	必要なスキル

自動化を再開して動作することを確認する。	次の AWS CLI コマンドを使用して、自動化を再開します。
	`aws rds modify-db-instance \ --db-instance-identifier vis \ --automation-mode full \`
	これで、データベースは Amazon RDS カスタムによって管理されました。たとえば、リスナーまたはデータベースがダウンした場合、Amazon RDS Custom エージェントはそれらを再起動します。これをテストするには、以下のようなコマンドを実行してください。
	ストップリスナーの例:
	`-bash-4.2$ lsnrctl stop vis`
	データベースをシャットダウンする例:
	`SQL> shutdown immediate ;`
		DBA

タスク	説明	必要なスキル
スキーマ、接続、メンテナンスタスクを検証します。	移行を完了するには、少なくとも以下のタスクを実行する必要があります。 • FS_CLONE を実行してパッチファイルシステムを同期します。 • スキーマの統計情報を収集します。 • 外部インターフェイスとシステムが新しい Amazon RDS Custom データベースに接続できることを確認します。 • バックアップとメンテナンスのスケジュールを設定します。 • ファイルシステムを切り替えるカットオーバーを実行して、AD Online Patching (ADOP) が期待どおりに機能していることを確認します。	DBA

トラブルシューティング

問題	ソリューション
ログファイルを削除しようとすると ORA-01624 エラーが表示されます。	ログファイルを削除しようとしたときに ORA-01624 エラーが表示される場合は、次の手順に従ってください。 次のコマンドを実行して、削除するログファイルのステータスが INACTIVE になるまで待ち

問題	ソリューション

	ます。V$log のステータスコードの詳細については、「Oracle のドキュメント」を参照してください。以下にコマンドの例とその出力を示します。
	
	`SQL> select group#, status from v$log;`
	
	`GROUP# STATUS`
	`---------- ----------------`
	`1 ACTIVE`
	`2 CURRENT`
	`3 UNUSED`
	`4 UNUSED`
	`5 UNUSED`
	`6 UNUSED`
	`6 rows selected.`
	
	この例では、ログファイル 1 は ACTIVE なので、先に追加した最初の新しいログファイルのステータスが CURRENT になるように、ログファイルを強制的に 3 回切り替える必要があります。
	
	`SQL> alter system switch logfile;`
	`System altered.`
	`SQL> alter system switch logfile;`
	`System altered.`
	`SQL> alter system switch logfile;`
	`System altered.`
	
	次の例のように、削除するログファイルがすべて INACTIVE になるまで待ってから、DROP LOGFILE コマンドを実行します。
	
	`SQL> select group#, status from v$log;`
	`GROUP# STATUS`
	`---------- ----------------`
	`1 INACTIVE`

問題	ソリューション
	2 INACTIVE 3 INACTIVE 4 CURRENT 5 UNUSED 6 UNUSED 6 rows selected.
resetlogs でデータベースを開くと、ORA-00392 エラーが表示されます。	「ORA-00392: スレッド 1 のログ xx はクリアされています。操作は許可されていません」というエラーが表示された場合は、以下のコマンドを実行して (xx をログファイル番号に置き換えてください)、open resetlogs コマンドを再実行してください。 SQL> alter database clear logfile group xx; SQL> alter database open resetlogs;

問題	ソリューション
Sysadmin またはアプリケーションユーザーを使用してアプリケーションに接続できない。	問題を確認するには、次の SQL クエリを実行します。 `SQL> select dbms_java.get_jdk_ version() from dual; select dbms_java.get_jdk_version() from dual ERROR at line 1: ORA-29548: Java system class reported: release of Java system classes in the database (19.0.0.0.220719 1.8) does not match that of the oracle executabl e (19.0.0.0.0 1.8)` 根本原因:ソースデータベースに複数のパッチが適用されたが、Amazon RDS Custom DB_HOME が新規インストールであるか、CEV の作成時に必要な RSU パッチ (OJVM など) を使用しなかったために CEV にすべてのパッチが含まれていませんでした。 これを検証するには、ソースパッチの詳細が `$ORACLE_HOME/sqlpath` 、`$ORACLE_HOME/.patch_storage` 、および `opatch – lsinventory` に記載されているかどうかを確認してください。 参照:datapatch-verbose が「パッチ xxxxxx: アーカイブされたパッチディレクトリが空です」というエラーで失敗する (ドキュメント ID 2235541.1) 修正:見つからないパッチ関連ファイルをソース (`$ORACLE_HOME/sqlpatch/`) から Amazon RDS Custom (`$ORACLE_HOME/ sqlpatch/`) にコピーし、`./datapatch -verbose` を再実行します。

問題	ソリューション

	以下に例を示します。
	`-bash-4.2$ cp -rp 18793246 20204035 20887355 22098146 22731026 $ORACLE_HOME/sqlpatch/`
	または、CDB および PDB で以下のコマンドを実行することにより、回避方法を使用できます。
	`@?/javavm/install/update_javavm_db.sql`
	次に PDB 上で以下のコマンドを実行します。
	`sql> alter session set container=vis; @?/javavm/install/update_javavm_db.sql`
	次に、テストをもう一度実行します。
	`SQL> select dbms_java.get_jdk_version() from dual;`

関連リソース

- 「Amazon RDS Custom の操作」(Amazon RDS ドキュメント)
- 「Oracle 向け Amazon RDS Custom — データベース環境における新しい制御機能」(AWS ニュースブログ)
- 「Amazon RDS Custom フォーOracleと Amazon EFS を統合」(AWS データベースブログ)
- 「AWS でのOracle E-Business Suite の移行」(AWS ホワイトペーパー)
- 「AWS でのOracle E-Business Suite のアーキテクチャ」(AWS ホワイトペーパー)
- 「アクティブスタンバイデータベースを使用して Amazon RDS Custom で Oracle E-Business Suite の HA/DR アーキテクチャをセットアップする」(AWS 規範ガイダンス)

追加情報

メンテナンスオペレーション

Oracle E-Business Suite データベースホームに新しいパッチを適用

bin ボリューム (/rdsdbbin) はアウトオブプレースアップグレードであるため、bin ボリュームの内容は「CEV のアップグレード」中に削除されます。そのため、CEV を使用してアップグレードを実行する前に、appsutil ディレクトリのコピーを作成する必要があります。

ソースの Amazon RDS Custom インスタンスで、CEV をアップグレードする前に、$ORACLE_HOME/appsutil のバックアップを取ってください。

> **Note**
>
> この例では、NFS ボリュームを使用します。ただし、代わりに Amazon Simple Storage Service (Amazon S3) へのコピーを使用できます。

1. appsutil をソースの Amazon RDS Custom インスタンスに保存するディレクトリを作成します。

```
$ mkdir /RMAN/appsutil.preupgrade
```

2. Tar を使用して、Amazon EFS ボリュームにコピーします。

```
$ tar cvf /RMAN/appsutil.preupgrade appsutil
```

3. tar ファイルが存在することを確認します。

```
$ bash-4.2$ ls -l /RMAN/appsutil.preupgrade
-rw-rw-r-- 1 rdsdb rdsdb 622981120 Feb  8 20:16 appsutil.tar
```

4. Amazon RDS ドキュメントの「RDS カスタム DB インスタンスのアップグレード」の手順に従って、最新の CEV (前提条件の CEV は既に作成されている) にアップグレードします。

OPATCH を使用して直接パッチを適用することもできます。Amazon RDS ドキュメントの「Oracle アップグレード用 RDS カスタムの要件と考慮事項」セクションを参照してください。

> **Note**
>
> ホストマシンの IP アドレスは、CEV パッチ適用プロセス中に変更されません。このプロセスではアウトオブプレースアップグレードが実行され、起動時に新しいビンボリュームが同じインスタンスにアタッチされます。

Oracle PeopleSoft を Amazon RDS Custom に移行

作成者:Gaurav Gupta (AWS)

概要

「Oracle PeopleSoft」は、企業全体のプロセスを対象とするエンタープライズリソースプランニング (ERP) ソリューションです。PeopleSoft のアーキテクチャは、クライアント、アプリケーション、データベースの 3 層構造です。PeopleSoft は「Amazon Relational Database Service (Amazon RDS)」で実行できます。PeopleSoft を「Amazon RDS Custom」でも実行できるようになり、基盤となるオペレーティングシステムにアクセスできるようになりました。

「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。Oracle データベースを Amazon RDS Custom に移行すると、Amazon Web Services (AWS) がバックアップタスクと高可用性を管理できるため、ユーザーは PeopleSoft のアプリケーションと機能のメンテナンスに集中できます。移行で考慮すべき主な要素については、AWS 規範ガイダンスの「Oracle データベースの移行戦略」を参照してください。

このパターンは、Amazon Elastic Compute Cloud (Amazon EC2) の PeopleSoft データベースを Oracle Recovery Manager (RMAN) のバックアップを使用して、Amazon RDS Custom に移行する手順に焦点を当てています。EC2 インスタンスと Amazon RDS Custom 間では「Amazon Elastic File System (Amazon EFS)」共有ファイルシステムを使用しますが、Amazon FSx や任意の共有ドライブを使用することもできます。このパターンでは RMAN フルバックアップ (レベル 0 バックアップと呼ばれることもあります) を使用します。

前提条件と制限

前提条件

- Oracle Linux 7、Oracle Linux 8、Red Hat Enterprise Linux (RHEL) 7、または RHEL 8 を搭載した Amazon EC2 上で実行されている Oracle バージョン 19C のソースデータベース。このパターンの例では、ソースデータベース名は `FSDM092` ですが、必須ではありません。

> **Note**
>
> このパターンは、オンプレミスの Oracle ソースデータベースでも使用できます。オンプレミスネットワークと仮想プライベートクラウド (VPC) 間には、適切なネットワーク接続が必要です。

- PeopleSoft 9.2 デモインスタンス。
- 単一の PeopleSoft アプリケーション層。ただし、このパターンを複数のアプリケーション層で機能するように調整できます。
- Amazon RDS Custom には 8 GB 以上のスワップスペースが設定されています。

機能制限

このパターンは以下の設定をサポートしていません。

- データベース ARCHIVE_LAG_TARGET パラメータを 60 ~ 7,200 の範囲外の値に設定します。
- DB インスタンスログモードを無効にする (NOARCHIVELOG)
- EC2 インスタンスの Amazon Elastic Block Store (Amazon EBS) 最適化属性をオフにする
- EC2 インスタンスにアタッチされた元の EBS ボリュームを変更する
- 新しい EBS ボリュームを追加するか、ボリュームタイプを gp2 から gp3 に変更します。
- パラメータの拡張 LOG_ARCHIVE_FORMAT フォーマットの変更 (*.arc が必須)
- 制御ファイルの場所と名前を多重化または変更する (必ず /rdsdbdata/db/*DBNAME*/controlfile/control-01.ctl に変更する必要がある)

これらおよびその他のサポートされていない設定に関する追加情報については、「Amazon RDS ドキュメント」を参照してください。

製品バージョン

Amazon RDS Custom がサポートする Oracle Database のバージョンとインスタンスクラスについては、「Amazon RDS Custom for Oracle の要件と制限」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Application Load Balancer
- Amazon EFS
- Amazon RDS Custom for Oracle
- AWS Secrets Manager
- Amazon Simple Storage Service (Amazon S3)

ターゲットアーキテクチャ

以下のアーキテクチャ図は、AWS の単一の「アベイラビリティーゾーン」で実行されている PeopleSoft システムを表しています。アプリケーション層には、「Application Load Balancer」を介してアクセスされます。アプリケーションとデータベースはどちらもプライベートサブネットにあり、Amazon RDS Custom と Amazon EC2 データベースインスタンスは Amazon EFS 共有ファイルシステムを使用して RMAN バックアップファイルを保存し、アクセスします。Amazon S3 は、カスタム RDS Oracle エンジンの作成と REDO ログメタデータの保存に使用されます。

ツール

ツール

AWS サービス

- 「Amazon RDS Custom for Oracle」は、基盤となるオペレーティングシステムとデータベース環境へのアクセスを必要とするレガシーアプリケーション、カスタムアプリケーション、パッケージアプリケーション向けのマネージドデータベースサービスです。バックアップや高可用性などのデータベース管理タスクを自動化します。
- 「Amazon Elastic File System (Amazon EFS)」は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。このパターンでは、Amazon EFS 共有ファイルシステムを使用して RMAN バックアップファイルを保存し、アクセスします。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。このパターンでは、Secrets Manager からデータベースユーザーのパスワードを取得して、`RDSADMIN` および `ADMIN` ユーザーを作成し、`sys` および `system` のパスワードを変更します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「Elastic Load Balancing (ELB)」は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。例えば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および 1 つまたは複数のアベイラビリティーゾーンの IP アドレスにトラフィックを分散できます。このパターンでは、Application Load Balancer を使用します。

その他のツール

- Oracle Recovery Manager (RMAN) は、Oracle データベースのバックアップとリカバリのサポートを提供します。このパターンでは、RMAN を使用して Amazon EC2 上のソース Oracle データベースのホットバックアップを実行し、Amazon RDS Custom で復元します。

ベストプラクティス

- データベース初期化パラメータについては、Oracle ソースデータベースの spfile を使用する代わりに、PeopleSoft 用の Amazon RDS Custom DB インスタンスによって提供される標準プロファイルをカスタマイズします。これは、Amazon RDS Custom でリードレプリカを作成する際に空白やコメントが原因で問題が発生するためです。データベース初期化パラメータの詳細については、Oracle Support ノート 1100831.1 (「Oracle Support」アカウントが必要) を参照してください。
- Amazon RDS Custom はデフォルトで Oracle の自動メモリ管理を使用します。Hugemem カーネルを使用したい場合は、代わりに自動共有メモリ管理を使用するように Amazon RDS Custom を設定できます。
- `memory_max_target` パラメータはデフォルトでは有効のままにしておきます。フレームワークはこれをバックグラウンドで使用してリードレプリカを作成します。
- Oracle フラッシュバックデータベースを有効にします。この機能は、フェイルオーバー (スイッチオーバーではない) のテストシナリオでスタンバイを復元する場合に便利です。

エピック

DB インスタンスとファイルシステムをセットアップする

タスク	説明	必要なスキル
DB インスタンスを作成します。	Amazon RDS コンソールで、FSDMO92 (またはソースデータベース名) という名前の DB 名で Amazon RDS Custom for Oracle DB インスタンスを作成します。 手順については、AWS ドキュメントの「Amazon RDS Custom の使用」と、ブログ	DBA

タスク	説明	必要なスキル
	記事「Oracle 用 Amazon RDS カスタム — データベース環境における新しい制御機能」を参照してください。これにより、データベース名はソースデータベースと同じ名前に設定されます。(空白のままにすると、EC2 インスタンスとデータベース名は ORCL に設定されます)。	

ソース Amazon EC2 データベースの RMAN 完全バックアップを実行します。

タスク	説明	必要なスキル
バックアップスクリプトを作成します。	RMAN バックアップスクリプトを作成して、マウントした Amazon EFS ファイルシステムにデータベースをバックアップします (次の例では /efs)。サンプルコードを使用するか、既存の RMAN スクリプトのいずれかを実行できます。 `#!/bin/bash` `Dt=`date +'%Y%m%d-%H` `%M'`` `BACKUP_LOG="rman-${O` `RACLE_SID}-$Dt"` `export TAGDATE=`date +` `%Y%m%d%H%M`;` `LOGPATH=/u01/scripts/l` `ogs`	DBA

タスク	説明	必要なスキル
	rman target / >>
$LOGPATH/rman-${OR
ACLE_SID}-$Dt << EOF
SQL "ALTER SYSTEM SWITCH
LOGFILE";
SQL "ALTER SESSION SET
NLS_DATE_FORMAT="D
D.MM.YYYY HH24:MI:S
S"";
RUN
{
ALLOCATE CHANNEL ch11
TYPE DISK MAXPIECESIZE
5G;
ALLOCATE CHANNEL ch12
TYPE DISK MAXPIECESIZE
5G;
BACKUP AS COMPRESSED
BACKUPSET FULL DATABASE
FORMAT '/efs/rma
n_backup/FSCM/%d_%T_
%s_%p_FULL' ;
SQL "ALTER SYSTEM
ARCHIVE LOG CURRENT";
BACKUP FORMAT '/efs/
rman_backup/FSCM/%d_
%T_%s_%p_ARCHIVE'
ARCHIVELOG ALL DELETE
ALL INPUT ;
BACKUP CURRENT
CONTROLFILE FORMAT '/
efs/rman_backup/FSCM/
%d_%T_%s_%p_CONTROL';
}
EXIT;
EOF	

タスク	説明	必要なスキル

バックアップスクリプトを実行します。	RMAN バックアップスクリプトを実行するには、Oracle Home User としてログインし、スクリプトを実行します。
	`$  chmod a+x rman_backup.sh`
	`$ ./rman_backup.sh &`
		DBA

タスク	説明	必要なスキル

エラーを確認し、バックアップファイル名をメモします。	RMAN ログファイルにエラーがないか確認します。すべて問題なければ、以下のコマンドを実行して制御ファイルのバックアップを一覧表示します。
	`RMAN> list backup of controlfile;`
	`using target database control file instead of recovery catalog`
	出力ファイルの名前に注意してください。
	`List of Backup Sets`
	`===================`
	`BS Key Type LV Size Device Type Elapsed Time Completion Time`
	`------- ---- -- ---------- ----------- ------------ ---------- ---------- ----------

タスク	説明	必要なスキル

	<pre>Control File Included:
	Ckp SCN: 165591599
	85898 Ckp time: 13-
	JUL-22</pre>
	Amazon RDS Custom でデータベースを復元するときは、バックアップコントロール `/efs/rman_backup/FSCM/FSDMO92_20220713_12_1_CONTROL` ファイルを使用します。	

ソースアプリケーション層をシャットダウンします

タスク	説明	必要なスキル

アプリケーションをシャットダウンします。	ソースアプリケーション層をシャットダウンするには、ユーティリティ psadmin または psadmin コマンドラインユーティリティを使用してください。
	1. ウェブサーバーをシャットダウンするには、以下のコマンドを実行します。
	<pre>psadmin -w shutdown -
	d "webserver domain
	name"</pre>
	2. アプリケーションサーバーをシャットダウンするには、次のコマンドを実行します。
		DBA、PeopleSoft 管理者

タスク	説明	必要なスキル

	`psadmin -c shutdown -d "application server domain name"`
	3. プロセススケジューラをシャットダウンするには、以下のコマンドを実行します。
	`psadmin -p stop -d "process scheduler domain name"`	

ターゲット Amazon RDS Custom データベースを設定します。

タスク	説明	必要なスキル
nfs-utils rpm パッケージをインストールします。	`nfs-utils rpm` パッケージをインストールするには、次のコマンドを実行します。 `$ yum install -y nfs-utils`	DBA
EFS ストレージをマウントします。	Amazon EFS コンソールページから Amazon EFS マウントコマンドを取得します。ネットワークファイルシステム (NFS) クライアントを使用して、EFS ファイルシステムを Amazon RDS インスタンスにマウントします。 `sudo mount -t nfs4 -o nfsvers=4.1,rsize=`	DBA

タスク	説明	必要なスキル
	`1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-xxxxxxxxxx.efs.eu-west-1.amazonaws.com:/ /efs` `sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-xxxxxxxxxx.efs.eu-west-1.amazonaws.com:/ /efs`	

スターターデータベースを削除し、データベースファイルを保存するディレクトリを作成します。

タスク	説明	必要なスキル
自動化モードを一時停止します。	自動化が RMAN の復元アクティビティを妨げないように、次のステップに進む前に Amazon RDS Custom DB インスタンスの「自動化モード」を一時停止する必要があります。 AWS コンソールまたは AWS コマンドラインインターフェイス(AWS CLI) コマンドを使用して、オートメーションを一時停止できます (最初に「AWS CLI を設定」したことを確認してください)。 `aws rds modify-db-instance \`	DBA

タスク	説明	必要なスキル
	``` --db-instance-id entifier peoplesoft- fscm-92 \ --automation-mode all- paused \ --resume-full-au tomation-mode-minute 360 \ --region eu-west-1 ``` 一時停止時間を指定するときは、RMAN の復元に十分な時間を確保してください。これはソースデータベースのサイズによって異なるため、360 の値を適宜変更してください。 また、自動化の一時停止の合計時間が、データベースのバックアップまたはメンテナンスの時間帯と重ならないようにしてください。	

タスク	説明	必要なスキル

PeopleSoft のパラメータファイルを作成して変更する	PeopleSoft 用のファイルを作成および変更するには、Amazon RDS Custom DB インスタンスで作成された標準ファイルを使用します。PeopleSoft に必要なパラメータを追加します。
	
	1. 次の `rds user rdsdb` コマンドを実行して切り替えます。
	
	`$ sudo su - rdsdb`
	
	2. スターターデータベースで SQL*Plus にログインし、以下のコマンドを実行して pfile を作成します。
	
	`SQL> create pfile from spfile;`
	
	これにより、`$ORACLE_HOME/dbs` に pfile が作成されます。
	3. このファイルをバックアップします。
	4. このファイルを編集して PeopleSoft パラメーターを追加または更新します。
	
	`*._gby_hash_aggregation_enabled=false`
	
	`*._unnest_subquery=false`
		DBA

タスク	説明	必要なスキル
	``` *.nls_language=' AMERICAN' *.nls_length_sem antics='CHAR' *.nls_territ ory='AMERICA' *.open_cursors=1000 *.db_files=1200 *.undo_tablespace=' UNDOTBS1' ``` PeopleSoft関連のパラメータは、「Oracle Support」ノート1100831.1に記載されています。 5. spfile 参照をファイルから削除します。 ``` *.spfile='/rdsdbbi n/oracle/dbs/spfil eFSDMO92.ora' ```	

タスク	説明	必要なスキル

スターターデータベースをドロップします。	既存の Amazon RDS Custom データベースを削除するには、次のコードを使用します。
	
	`$ sqlplus / as sysdba`
	`SQL> shutdown immediate ;`
	`SQL> startup mount exclusive restrict;`
	`SQL> drop database;`
	`SQL> exit`	

タスク	説明	必要なスキル
バックアップから Amazon RDS Custom データベースを復元します。	次のスクリプトを使用してデータベースを復元します。このスクリプトは、最初に制御ファイルを復元し、次に EFS マウントに保存されているバックアップピースからデータベース全体を復元します。 `#!/bin/bash` `Dt=`date +'%Y%m%d-%H%M'` `BACKUP_LOG="rman-${ORACLE_SID}-$Dt"` `export TAGDATE=`date +%Y%m%d%H%M`;` `LOGPATH=/rdsdbdata/scripts/logs` `rman target / >> $LOGPATH/rman-${ORACLE_SID}-$Dt << EOF` `restore controlfile from "/efs/rman_backup/FSCM/FSDMO92_20220713_12_1_CONTROL";` `alter database mount;` `run` `{` `set newname for database to '/rdsdbdata/db/FSDMO92_A/datafile/%f_%b';` `SET NEWNAME FOR TEMPFILE 1 TO '/rdsdbdata/db/FSDMO92_A/datafile/%f_%b';` `RESTORE DATABASE;` `SWITCH DATAFILE ALL;` `SWITCH TEMPFILE ALL;`	DBA

タスク	説明	必要なスキル
	RECOVER DATABASE; } EOF sqlplus / as sysdba >> $LOGPATH/rman-${OR ACLE_SID}-$Dt<<-EOF ALTER DATABASE RENAME FILE '/u01/psoft/db/ oradata/FSDMO92/redo0 1.log' TO '/rdsdbda ta/db/FSDMO92_A/on linelog/redo01.log'; ALTER DATABASE RENAME FILE '/u01/psoft/db/ oradata/FSDMO92/redo0 2.log' TO '/rdsdbda ta/db/FSDMO92_A/on linelog/redo02.log'; ALTER DATABASE RENAME FILE '/u01/psoft/db/ oradata/FSDMO92/redo0 3.log' TO '/rdsdbda ta/db/FSDMO92_A/on linelog/redo03.log'; alter database clear unarchived logfile group 1; alter database clear unarchived logfile group 2; alter database clear unarchived logfile group 3; alter database open resetlogs; EXIT EOF	

Secrets Manager からのパスワードの取得、ユーザーの作成、パスワードの変更

タスク	説明	必要なスキル
Secrets Manager からパスワードを取得します。	このステップは、AWS コンソールまたは AWS CLI を使用して実行できます。以下の手順は、コンソールの手順を示しています。 1. AWS マネジメントコンソール にサインインして、Amazon RDS コンソールを開きます。 2. ナビゲーションペインで、[データベース] を選択し、Amazon RDS データベースを選択します。 3. [設定] タブを選択し、インスタンスのリソース ID を書き留めます。これは db-<ID> という形式になります(たとえば、db-73GJNHLGDNZNDOXNWXSECUW6LE)。 4. Secrets Manager コンソールを開きます。 5. do-not-delete-custom-<resource_id> と同じ名前のシークレットを選択します。ここで、resource-id はステップ 3 で書き留めたリソース ID を指します。 6. [シークレットの値を取得する] を選択します。	DBA

タスク	説明	必要なスキル
	このパスワードは sys、system、rdsadmin、および admin の各ユーザーでも同じになります。	

タスク	説明	必要なスキル

RDSADMIN ユーザーを作成します。	RDSADMIN は、Amazon RDS Custom DB インスタンスのモニタリングとオーケストレーションを行うデータベースユーザーです。スターターデータベースは削除され、ターゲットデータベースは RMAN を使用してソースから復元されたため、Amazon RDS Custom モニタリングが期待どおりに動作するように、復元操作後にこのユーザーを再作成する必要があります。また、RDSADMIN ユーザー用に別のプロファイルとテーブルスペースを作成する必要があります。
	
	1. SQL プロンプトで次のコマンドを入力します。
	
	`SQL> set echo on feedback on serverout on`
	`SQL> @?/rdbms/admin/utlpwdmg.sql`
	`SQL> ALTER PROFILE DEFAULT LIMIT FAILED_LOGIN_ ATTEMPTS UNLIMITED PASSWORD_LIFE_TIME UNLIMITED PASSWORD_VERIFY_F UNCTION NULL;`
		DBA

タスク	説明	必要なスキル

	2. RDSADMIN プロフィールを作成します。
	
	`SQL> set echo on`
	` feedback on serverout`
	` on`
	`SQL> alter session set`
	` "_oracle_script"=t`
	`rue;`
	`SQL> CREATE PROFILE`
	` RDSADMIN`
	`LIMIT`
	`COMPOSITE_LIMIT`
	` UNLIMITED`
	`SESSIONS_PER_USER`
	` UNLIMITED`
	`CPU_PER_SESSION`
	` UNLIMITED`
	`CPU_PER_CALL`
	` UNLIMITED`
	`LOGICAL_READS_PER`
	`_SESSION UNLIMITED`
	`LOGICAL_READS_PER`
	`_CALL UNLIMITED`
	`IDLE_TIME UNLIMITED`
	`CONNECT_TIME`
	` UNLIMITED`
	`PRIVATE_SGA`
	` UNLIMITED`
	`FAILED_LOGIN_ATTE`
	`MPTS 10`
	`PASSWORD_LIFE_TIME`
	` UNLIMITED`
	`PASSWORD_REUSE_TIME`
	` UNLIMITED`
	`PASSWORD_REUSE_MAX`
	` UNLIMITED`
	`PASSWORD_VERIFY_F`
	`UNCTION NULL`
	`PASSWORD_LOCK_TIME`
	` 86400/86400`	

タスク	説明	必要なスキル
	PASSWORD_GRACE_TIME 604800/86400; 3. RDSADMIN テーブルスペースを作成します。 SQL> CREATE BIGFILE TABLESPACE rdsadmin '/rdsdbdata/db/FSD MO92_A/datafile/rd sadmin.dbf' DATAFILE SIZE 7M AUTOEXTEND ON NEXT 1m LOGGING ONLINE PERMANENT BLOCKSIZE 8192 EXTENT MANAGEMEN T LOCAL AUTOALLOCATE DEFAULT NOCOMPRES S SEGMENT SPACE MANAGEMENT AUTO; 4. RDSADMIN ユーザーを作成します。RDSADMIN パスワードを、以前に Secrets Manager から取得したパスワードに置き換えます。 SQL> CREATE USER rdsadmin IDENTIFIED BY xxxxxxxxxx DEFAULT TABLESPACE rdsadmin TEMPORARY TABLESPACE TEMP profile rdsadmin ; 5. RDSADMIN に権限を付与します。	

タスク	説明	必要なスキル
	``` SQL> GRANT "CONNECT" TO RDSADMIN WITH ADMIN OPTION; SQL> GRANT "RESOURCE " TO RDSADMIN WITH ADMIN OPTION; SQL> GRANT "DBA" TO RDSADMIN; SQL> GRANT "SELECT_C ATALOG_ROLE" TO RDSADMIN WITH ADMIN OPTION; SQL> GRANT ALTER SYSTEM TO RDSADMIN; SQL> GRANT UNLIMITED TABLESPACE TO RDSADMIN; SQL> GRANT SELECT ANY TABLE TO RDSADMIN; SQL> GRANT ALTER DATABASE TO RDSADMIN; SQL> GRANT ADMINISTER DATABASE TRIGGER TO RDSADMIN; SQL> GRANT ANY OBJECT PRIVILEGE TO RDSADMIN WITH ADMIN OPTION; SQL> GRANT INHERIT ANY PRIVILEGES TO RDSADMIN; SQL> ALTER USER RDSADMIN DEFAULT ROLE ALL; ``` 6. Set the SYS, SYSTEM, and DBSNMP user profiles to RDSADMIN.	

タスク	説明	必要なスキル

	`SQL> set echo on`
	`feedback on serverout`
	`on`
	`SQL> alter user SYS`
	`profile RDSADMIN;`
	`SQL> alter user SYSTEM`
	`profile RDSADMIN;`
	`SQL> alter user DBSNMP`
	`profile RDSADMIN;`	

マスターユーザーを作成します。	スターターデータベースは削除され、ターゲットデータベースは RMAN を使用してソースから復元されたため、マスターユーザーを再作成する必要があります。この例では、ユーザーは admin という名前になります。
	`SQL> create user`
	`admin identified by`
	`<password>;`
	`SQL> grant dba to admin`
		DBA

タスク	説明	必要なスキル

システムパスワードを変更します。	Secrets Manager から取得したパスワードを使用して、システムパスワードを変更します。
	
	`SQL> alter user sys identified by xxxxxxxxxx;`
	`SQL> alter user system identified by xxxxxxxxxx;`
	
	これらのパスワードを変更しない場合、Amazon RDS Custom は「データベースモニタリングユーザーまたはユーザー認証情報が変更されました」というエラーメッセージを表示します。
		DBA

Amazon RDS Custom と PeopleSoft の TNS エントリを設定します。

タスク	説明	必要なスキル
tnsnames ファイルを設定します。	アプリケーション層からデータベースに接続するには、アプリケーション層からデータベースに接続できるように `tnsnames.ora` ファイルを設定します。次の例では、`tnsnames.ora` ファイルへのソフトリンクはありますが、デフォルトではファイルは空であることがわかります。	DBA

タスク	説明	必要なスキル

	``
	$ cd /rdsdbbin/oracle/n
	etwork/admin
	$ ls -ltr
	-rw-r--r-- 1 rdsdb
	 database 1536 Feb 14
	 2018 shrept.lst
	lrwxrwxrwx 1 rdsdb
	 database 30 Apr 5
	 13:19 listener.ora -
	> /rdsdbdata/config/
	listener.ora
	lrwxrwxrwx 1 rdsdb
	 database 28 Apr 5
	 13:19 sqlnet.ora -
	> /rdsdbdata/config/
	sqlnet.ora
	lrwxrwxrwx 1 rdsdb
	 database 30 Apr 5
	 13:19 tnsnames.ora -
	> /rdsdbdata/config/
	tnsnames.ora
	``
	
	1. tsnames.ora エントリを作成します。Amazon RDS の自動化によるファイルの解析方法により、エントリに空白、コメント、余分な行が含まれていないことを確認する必要があります。そうしないと、「create-db-instance-read-replica」など、一部の API を使用する際に問題が発生する可能性があります。
	2. PeopleSoft データベースの要件に従ってポート、ホスト、SID を交換してくださ	

タスク	説明	必要なスキル

	い。次の例のコードを使用します。
	
	`$ vi tnsnames.ora`
	
	`FSDMO92=(DESCRIP TION = (ADDRESS_ LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = x.x.x.x)(PORT = 1521))) (CONNECT_ DATA = (SERVER = DEDICATED) (SID = FSDMO92)))`
	
	3. PeopleSoft データベースにアクセスできることを確認するには、次のコマンドを実行します。
	
	`$ tnsping FSDMO92`
	
	`TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 14-JUL-2022 10:16:45`
	
	`Copyright (c) 1997, 2021, Oracle. All rights reserved.`
	
	`Used parameter files: /rdsdbbin/oracle/network/admin/sqlnet.ora`
	
	`Used TNSNAMES adapter to resolve the alias`	

タスク	説明	必要なスキル
	``` Attempting to contact (DESCRIPT ION = (ADDRESS_ LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = x.x.x.x)(PORT = 1521))) (CONNECT_ DATA = (SERVER = DEDICATED) (SID = FSDMO92))) OK (0 msec) ```	

spfile ソフトリンクの作成

タスク	説明	必要なスキル
spfile ソフトリンクを作成します。	1. `/rdsdbdata/admin/FSDMO92/pfile` の場所に spfile を作成するには、次のコマンドを実行します。 ``` SQL> create spfile='/ rdsdbdata/admin/FS DMO92/pfile/spfile FSDMO92.ora' from pfile; ``` 2. `$ORACLE_HOME/dbs` に移動し、spfile のソフトリンクを作成します。 ``` ln -s '/rdsdbdata/ admin/FSDMO92/pfile/ spfileFSDMO92.ora' spfileFSDMO92.ora ```	DBA

タスク	説明	必要なスキル
	3. このファイルが作成されたら、spfile を使用してデータベースを停止して起動できます。	

移行後の手順を実行する

タスク	説明	必要なスキル
スキーマ、接続、メンテナンスタスクを検証する。	移行を完了するには、次のタスクを実行します。 • スキーマの統計情報を収集します。 • PeopleSoft アプリケーション層が新しい Amazon RDS Custom データベースに接続できることを確認します。 • バックアップとメンテナンスのスケジュールを設定します。	DBA

関連リソース

- 「Amazon RDS Custom の操作」
- 「Oracle 向け Amazon RDS Custom — データベース環境における新しいコントロール機能」(ブログ記事)
- 「Amazon RDS Custom for Oracle と Amazon EFS の統合」(ブログ記事)
- 「Amazon RDS を Oracle PeopleSoft データベースとして設定」(AWS ホワイトペーパー)

Oracle ROWID 機能を AWS の PostgreSQL に移行

作成者: Rakesh Raghav (AWS) と Ramesh Pathuri (AWS)

概要

このパターンでは、Oracle データベースの ROWID 疑似列機能を、Amazon Aurora PostgreSQL の Amazon Relational Database Service (Amazon RDS)、Amazon Aurora PostgreSQL　互換バージョン、またはAmazon Elastic Compute Cloud (Amazon EC2)に移行するオプションについて説明しています。

Oracle データベースでは、ROWID 疑似列がテーブル内の行の物理アドレスです。この疑似列は、プライマリキーがテーブルに存在しない場合でも行を一意に識別するために使用されます。PostgreSQL にも ctid と呼ばれる同様の疑似列がありますが、ROWID としては使用できません。「PostgreSQLのドキュメント」 で説明されているように、更新される場合、またはctid がすべての VACUUM のプロセスの後に変更される可能性があります。

PostgreSQLで ROWID 疑似列機能を作成する方法が3つあります：

- ROWID の代わりに、テーブルの行を識別するプライマリキー列を使用します。
- テーブルには論理的なプライマリキー/ユニークキー (複合キーの場合もあります) を使用します。
- 自動生成された値を含む列を追加し、ROWID を模倣するプライマリキー/ユニークキーにします。

このパターンでは、3 つの実装について順を追って説明し、各オプションの利点と欠点を説明します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 手続き型言語/PostgreSQL (PL/pgSQL) コーディングの専門知識
- ソース Oracle データベース
- Amazon RDS for PostgreSQL または Aurora PostgreSQL 互換クラスター、または PostgreSQL データベースをホストする EC2 インスタンス

機能制限

- このパターンは、ROWID の機能の回避策を提供します。PostgreSQL は Oracle データベースの ROWID と同等の機能を提供していません。

製品バージョン

- PostgreSQL 11.9 以降

アーキテクチャ

ソーステクノロジースタック

- Oracle Database

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換、Amazon RDS for PostgreSQL、または PostgreSQL データベースを備えた EC2 インスタンス

実装オプション

PostgreSQLで ROWID のサポート不足を回避するには、テーブルにプライマリキーまたはユニークインデックス、論理的プライマ的キー、またはID属性があるかどうかに応じて、3つの選択肢があります。どれを選択するかは、プロジェクトのスケジュール、現在の移行フェーズ、アプリケーションとデータベースコードへの依存関係によって異なります。

オプション	説明	利点	欠点
プライマリキーまたは一意のインデックス	Oracle テーブルにプライマリキーがある場合、このキーの属性を使用して、行を一意に識別することができます。	• 独自データベースの特徴には依存していません。。 • プライマリキーフィールドにはインデックスが付けられるため、パフォーマンス	• プライマリキーフィールドに切り替えるには、ROWID に依存するアプリケーションコードやデータベースコードを変更する必要があります。

		への影響は最小限です。	
論理プライマリキー/ユニーククキー	Oracle テーブルに論理プライマリキーがある場合、このキーの属性を使用して行を一意に識別できます。論理プライマリキーは、行を一意に識別できる 1 つまたは複数の属性で構成されますが、制約によってデータベースに適用されません。	• 独自データベースの特徴には依存していません。	• プライマリキーフィールドに切り替えるには、ROWID に依存するアプリケーションコードやデータベースコードを変更する必要があります。 • 論理プライマリキーの属性にインデックスが作成されない場合、パフォーマンスに重大な影響があります。ただし、パフォーマンスの問題を防ぐために唯一のインデックスを追加できます。

アイデンティティ属性	Oracle テーブルにプライマリキーがない場合、`GENERATED ALWAYS AS IDENTITY` として追加のフィールドを作成できます。この属性は、テーブルにデータが挿入されるたびに常に唯一の値を生成します。ですから、データ操作言語 (DML) 操作の行を一意に識別するために使用できます。	• 独自データベースの特徴には依存していません。 • PostgreSQL データベースは、属性を入力し、その一意性を維持します。	• アイデンティティ属性に切り替えるには、」`ROWID` に依存するアプリケーションおよびデータベースコードを変更する必要があります。 • 追加フィールドがインデックスされない場合、パフォーマンスに重大な影響を与えます。ただし、パフォーマンスの問題を避けるためにインデックスを追加できます。

ツール

- PostgreSQLのAmazon Relational Database Service (Amazon RDS) を使用して、 AWSクラウドで PostgreSQLリレーショナルデータベース (DB) のセットアップ、運用、スケーリングができます。
- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。このパターンでは、AWS CLI を使用して pgAdmin を介して SQL コマンドを実行できます。
- 「pgAdmin」は、PostgreSQLのオープンソース管理ツールです。データベースオブジェクトの作成、管理、使用を支援するグラフィカルインターフェイスを提供します。
- AWS Schema Conversion Tool (AWS SCT) は、ソースデータベーススキーマとカスタムコードの大部分を、ターゲットデータベースと互換性のある形式に自動的に変換することにより、異種データベース移行をサポートします。

エピック

ソーステーブルを識別する

タスク	説明	必要なスキル
ROWID 属性を使用する Oracle テーブルを特定します。	AWS Schema Conversion Tool (AWS SCT) を使用して、ROWID の機能を持つ Oracle テーブルを識別します。詳細については、「AWS SCTドキュメント」を参照してください。 -または- Oracle では、DBA_TAB_COLUMNS ビューを使用して、ROWID 属性を持つテーブルを識別します。これらのフィールドは、10 バイトの英数字を格納するために使用される場合があります。使用方法を決定し、必要に応じて VARCHAR フィールドに変換します。	DBA または開発者
これらのテーブルを参照するコードを識別します。	AWS SCT を使用して、移行評価レポートを生成し、ROWID より影響された手順を識別します。詳細については、「AWS SCTドキュメント」を参照してください。 -または- ソース Oracle データベースでは、dba_source のテーブルのテキストフィールドを使用して、ROWIDの機能を使用	DBA または開発者

タスク	説明	必要なスキル
	するオブジェクトを識別します。	

プライマリキーの使用を決定

タスク	説明	必要なスキル
プライマリキーがないテーブルを識別します。	ソース Oracle データベースで、DBA_CONSTRAINTS を使用してプライマリキーがないテーブルを識別します。この情報により、各テーブルの戦略を決定することを支援します。以下に例を示します。 `select dt.* from dba_tables dt where not exists (select 1 from all_constraints ct where ct.owner = Dt.owner and ct.table_name = Dt.table_name and ct.constraint_type = 'P' ) and dt.owner = '{schema}'`	DBA または開発者

ソリューションを識別して適用

タスク	説明	必要なスキル
プライマリキーが定義されているテーブル、または論理的なプライマリキーを持つテーブルに変更を適用します。	「追加情報」セクションに示されているアプリケーションコードおよびデータベースコードを変更して、固有のプライマリキーまたは論理プライマリキーを使用してテーブルの行を識別するようにします。	DBA または開発者
プライマリキーが定義されていない、または論理プライマリキーがないテーブルにフィールドを追加します。	タイプ `GENERATED ALWAYS AS IDENTITY` の属性を追加します。「追加情報」セクションに表示されているアプリケーションとデータベースのコードを変更します。	DBA または開発者
必要に応じてインデックスを追加します。	SQL のパフォーマンスを向上させるには、追加フィールドまたは論理プライマリキーにインデックスを追加します。	DBA または開発者

関連リソース

- 「PostgreSQL CTID」 (PostgreSQL ドキュメント)
- 「生成された列」 (PostgreSQL ドキュメント)
- 「ROWID 疑似カラム」 (Oracle ドキュメンテーション)

追加情報

以下のセクションでは、Oracle と PostgreSQL のコード例を示して、これらの 3 つのアプローチを説明します。

「シナリオ 1: プライマリユニークキーを使用」

以下の例では、testrowid_s1 テーブルを、emp_id をプライマリキーとして作成します。

Orackel コード:

```
create table testrowid_s1 (emp_id integer, name varchar2(10), CONSTRAINT testrowid_pk
 PRIMARY KEY (emp_id));
INSERT INTO testrowid_s1(emp_id,name) values (1,'empname1');
INSERT INTO testrowid_s1(emp_id,name) values (2,'empname2');
INSERT INTO testrowid_s1(emp_id,name) values (3,'empname3');
INSERT INTO testrowid_s1(emp_id,name) values (4,'empname4');
commit;

SELECT rowid,emp_id,name FROM testrowid_s1;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3pAAAAAAAMOAAA          1 empname1
AAAF3pAAAAAAAMOAAB          2 empname2
AAAF3pAAAAAAAMOAAC          3 empname3
AAAF3pAAAAAAAMOAAD          4 empname4

UPDATE testrowid_s1 SET name = 'Ramesh' WHERE rowid = 'AAAF3pAAAAAAAMOAAB' ;
commit;

SELECT rowid,emp_id,name FROM testrowid_s1;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3pAAAAAAAMOAAA          1 empname1
AAAF3pAAAAAAAMOAAB          2 Ramesh
AAAF3pAAAAAAAMOAAC          3 empname3
AAAF3pAAAAAAAMOAAD          4 empname4
```

PostgreSQL コード:

```
CREATE TABLE public.testrowid_s1
(
    emp_id integer,
    name character varying,
    primary key (emp_id)
);

insert into public.testrowid_s1 (emp_id,name) values
(1,'empname1'),(2,'empname2'),(3,'empname3'),(4,'empname4');
```

```
select emp_id,name from testrowid_s1;
 emp_id |   name
--------+----------
      1 | empname1
      2 | empname2
      3 | empname3
      4 | empname4

update testrowid_s1 set name = 'Ramesh' where emp_id = 2 ;

select emp_id,name from testrowid_s1;
 emp_id |   name
--------+----------
      1 | empname1
      3 | empname3
      4 | empname4
      2 | Ramesh
```

シナリオ 2: 論理プライマリキーを使用

以下の例では、 testrowid_s2 テーブルを emp_id を配して、論理プライマリキーとして作成します。

Orackel コード:

```
create table testrowid_s2 (emp_id integer, name varchar2(10) );
INSERT INTO testrowid_s2(emp_id,name) values (1,'empname1');
INSERT INTO testrowid_s2(emp_id,name) values (2,'empname2');
INSERT INTO testrowid_s2(emp_id,name) values (3,'empname3');
INSERT INTO testrowid_s2(emp_id,name) values (4,'empname4');
commit;

SELECT rowid,emp_id,name FROM testrowid_s2;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3rAAAAAAAMeAAA          1 empname1
AAAF3rAAAAAAAMeAAB          2 empname2
AAAF3rAAAAAAAMeAAC          3 empname3
AAAF3rAAAAAAAMeAAD          4 empname4

UPDATE testrowid_s2 SET name = 'Ramesh' WHERE rowid = 'AAAF3rAAAAAAAMeAAB' ;
commit;
```

```
SELECT rowid,emp_id,name FROM testrowid_s2;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3rAAAAAAAMeAAA          1 empname1
AAAF3rAAAAAAAMeAAB          2 Ramesh
AAAF3rAAAAAAAMeAAC          3 empname3
AAAF3rAAAAAAAMeAAD          4 empname4
```

PostgreSQL コード:

```
CREATE TABLE public.testrowid_s2
(
    emp_id integer,
    name character varying
);

insert into public.testrowid_s2 (emp_id,name) values
(1,'empname1'),(2,'empname2'),(3,'empname3'),(4,'empname4');

select emp_id,name from testrowid_s2;
 emp_id |   name
--------+----------
      1 | empname1
      2 | empname2
      3 | empname3
      4 | empname4

update testrowid_s2 set name = 'Ramesh' where emp_id = 2 ;

select emp_id,name from testrowid_s2;
 emp_id |   name
--------+----------
      1 | empname1
      3 | empname3
      4 | empname4
      2 | Ramesh
```

シナリオ 3: アイデンティティ属性を使用

以下の例では、プライマリキーなしで、またアイデンティティ属性を使用して、テーブル `testrowid_s3` を作成します。

Oracle コード:

```
create table testrowid_s3 (name varchar2(10));
INSERT INTO testrowid_s3(name) values ('empname1');
INSERT INTO testrowid_s3(name) values ('empname2');
INSERT INTO testrowid_s3(name) values ('empname3');
INSERT INTO testrowid_s3(name) values ('empname4');
commit;

SELECT rowid,name FROM testrowid_s3;
ROWID              NAME
------------------ ----------
AAAF3sAAAAAAAMmAAA empname1
AAAF3sAAAAAAAMmAAB empname2
AAAF3sAAAAAAAMmAAC empname3
AAAF3sAAAAAAAMmAAD empname4

UPDATE testrowid_s3 SET name = 'Ramesh' WHERE rowid = 'AAAF3sAAAAAAAMmAAB' ;
commit;

SELECT rowid,name FROM testrowid_s3;
ROWID              NAME
------------------ ----------
AAAF3sAAAAAAAMmAAA empname1
AAAF3sAAAAAAAMmAAB Ramesh
AAAF3sAAAAAAAMmAAC empname3
AAAF3sAAAAAAAMmAAD empname4
```

PostgreSQL コード:

```
CREATE TABLE public.testrowid_s3
(
    rowid_seq bigint generated always as identity,
    name character varying
);

insert into public.testrowid_s3 (name) values
('empname1'),('empname2'),('empname3'),('empname4');

select rowid_seq,name from testrowid_s3;
 rowid_seq |   name
-----------+----------
         1 | empname1
         2 | empname2
         3 | empname3
```

```
         4 | empname4

update testrowid_s3 set name = 'Ramesh' where rowid_seq = 2 ;

select rowid_seq,name from testrowid_s3;
 rowid_seq |   name
-----------+----------
         1 | empname1
         3 | empname3
         4 | empname4
         2 | Ramesh
```

Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する

作成者: Sai Parthasaradhi (AWS) と Veeranjaneyulu Grandhi (AWS)

概要

このパターンは、事前定義されたメタデータテーブルを使用して Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible エディションデータベースに移行する方法を示しています。

Oracle Database のエラーコードには、対応する PostgreSQL エラーコードが必ずあるとは限りません。このエラーコードの違いにより、ターゲット PostgreSQL アーキテクチャ内の手順または関数の処理ロジックを構成することが困難になる可能性があります。

PL/pgSQL プログラムにとって意味のあるソースデータベースとターゲットデータベースのエラーコードをメタデータテーブルに格納することで、プロセスを簡略化できます。次に、残りのプロセスロジックを続行する前に、有効な Oracle Database エラーコードにフラグを立て、それらを PostgreSQL の同等のエラーコードにマップするようにテーブルを構成します。Oracle Database のエラーコードがメタデータテーブルにない場合、例外が発生してプロセスは終了します。その後、エラーの詳細を手動で確認し、プログラムで必要な場合は新しいエラーコードをテーブルに追加できます。

この構成を使用すると、Amazon Aurora PostgreSQL-Compatible データベースは、ソース Oracle Database と同じ方法でエラーを処理できます。

> **Note**
>
> Oracle Database エラーコードを正しく処理するように PostgreSQL データベースを設定するには、通常、データベースとアプリケーションコードを変更する必要があります。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- インスタンスサービスとリスナーサービスが稼働しているソース Oracle Database
- 稼働中の Amazon Aurora PostgreSQL-Compatible クラスター

- Oracle Database に関する知識
- PostgreSQL データベースに関する知識

アーキテクチャ

次の図は、データエラーコードの検証と処理のための Amazon Aurora PostgreSQL-Compatible データベースワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. テーブルには、Oracle Database のエラーコードと分類、およびそれらに対応する PostgreSQL エラーコードと分類が格納されています。このテーブルには、事前に定義された特定のエラーコードが有効かどうかを分類する valid_error 列が含まれています。
2. PL/pgSQL 関数 (func_processdata) が例外を投げると、2 つ目の PL/pgSQL 関数 (error_validate) が呼び出されます。
3. error_validate 関数は Oracle Database のエラーコードを入力引数として受け入れます。次に、この関数は入力されたエラーコードをテーブルと照合して、エラーがテーブルに含まれているかどうかを確認します。
4. Oracle Database のエラーコードがテーブルに含まれている場合、error_validate 関数は TRUE 値を返し、プロセスロジックは続行されます。エラーコードがテーブルに含まれていない場合、関数は FALSE 値を返し、プロセスロジックは例外が発生して終了します。
5. 関数が FALSE 値を返すと、アプリケーションの機能責任者がエラーの詳細を手動で確認して、その有効性を判断します。
6. その後、新しいエラーコードは手動でテーブルに追加されるか、追加されないかのどちらかになります。エラーコードが有効でテーブルに追加された場合、error_validate 関数は次に例外が発生したときに TRUE 値を返します。エラーコードが有効ではなく、例外発生時に処理が失敗しなければならない場合、エラーコードはテーブルに追加されません。

テクノロジースタック

- Amazon Aurora PostgreSQL
- pgAdmin
- 「Oracle SQL Developer」

ツール

- Amazon Aurora PostgreSQL-Compatible バージョンは、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型のACID 準拠リレーショナルデータベースエンジンです。
- pgAdmin はオープンソースの PostgreSQL 向け管理開発ツールです。データベースオブジェクトの作成、保守、使用を簡素化するグラフィカルインターフェイスを提供します。
- Oracle SQL Developer は、従来のデプロイとクラウドデプロイの両方で Oracle Database の開発と管理を簡素化する無料の統合開発環境です。

エピック

Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する

タスク	説明	必要なスキル
Amazon Aurora PostgreSQL-Compatible データベースにテーブルを作成します。	次の PostgreSQL のテーブル作成 コマンドを実行します。 `( source_error_code numeric NOT NULL, target_error_code character varying NOT NULL, valid_error character varying(1) NOT NULL );`	PostgreSQL Developer、Oracle、RDS/Aurora for PostgreSQL
PostgreSQL エラーコードとそれに対応する Oracle Database のエラーコードをテーブルに追加する。	PostgreSQL の INSERT コマンドを実行して、必要なエラーコード値を error_codes テーブルに追加します。	PostgreSQL Developer、Oracle、RDS/Aurora for PostgreSQL

タスク	説明	必要なスキル
	PostgreSQL のエラーコードでは、文字可変データ型 (SQLSTATE 値) を使用する必要があります。Oracle のエラーコードでは数値データ型 (SQLCODE 値) を使用する必要があります。 Insert ステートメントの例: `insert into error_codes values (-1817,'2 2007','Y'); insert into error_codes values (-1816,'2 2007','Y'); insert into error_codes values (-3114,'0 8006','N');` **Note** Oracle 固有の Java データベース接続 (JDBC) の例外をキャッチする場合は、それらの例外を汎用的なクロスデータベース例外に置き換えるか、PostgreSQL 固有の例外に切り替える必要があります。	

タスク	説明	必要なスキル
PL/pgSQL 関数を作成してエラーコードを検証します。	PostgreSQL の関数の作成コマンドを実行して PL/pgSQL 関数を作成します。関数が以下を実行することを確認してください。 • プログラムが投げた Oracle エラーコードを許可する。 • error_codes テーブルにエラーコードがあるかどうかを確認する。 • エラーコードがメタデータテーブルに存在するかどうかに基づいて TRUE または FALSE の値を返す。	PostgreSQL Developer、Oracle、RDS/Aurora for PostgreSQL
PL/pgSQL 関数によって記録された新しいエラーコードを手動で確認する。	新しいエラーコードを手動で確認してください。 新しいエラーコードがユースケースに当てはまる場合は、PostgreSQL INSERT コマンドを実行して error_codes テーブルに追加します。 -または- 新しいエラーコードがユースケースに合わない場合は、そのエラーコードをテーブルに追加しないでください。エラーが発生すると、プロセスロジックは引き続き失敗し、例外が発生して終了します。	PostgreSQL Developer、Oracle、RDS/Aurora for PostgreSQL

関連リソース

付録 A. PostgreSQL エラーコード (PostgreSQL ドキュメント)

データベースエラーメッセージ (Oracle Databaseドキュメント)

Redis ワークロードを AWS 上の Redis Enterprise Cloud に移行

作成者:Antony Prasad Thevaraj (AWS) and Srinivas Pendyala (Redis)

概要

このパターンでは、Redis ワークロードをAmazon Web Services (AWS) 上の Redis Enterprise Cloud に移行するための大まかなプロセスについて説明します。移行手順について説明し、使用可能なツールの選択に関する情報を提供し、各ツールを使用する場合の利点、欠点、手順について説明します。Redis からワークロードを移行する際にさらにサポートが必要な場合は、Redis プロフェッショナルサービスをご利用いただくこともできます。

Redis OSS または Redis Enterprise Software をオンプレミスで実行している場合、データセンターで Redis データベースを管理することによる多大な管理オーバーヘッドと運用上の複雑さをよくご存知でしょう。ワークロードをクラウドに移行することで、この運用上の負担を大幅に軽減し、Redis が提供する完全にホストされたサービスとしてのデータベース (DBaaS) である「Redis Enterprise Cloud」を活用できます。この移行により、99.999% の可用性、アーキテクチャのシンプルさ、拡張性などの最新の Redis Enterprise Cloud on AWS 機能を利用しながら、ビジネスの俊敏性を高め、アプリケーションの信頼性を向上させ、全体的なコストを削減できます。

Redis Enterprise Cloud は、金融サービス、小売、医療、ゲームの分野だけでなく、不正検出、リアルタイムインベントリ、請求処理、セッション管理のソリューションを必要とするユースケースにも応用できる可能性があります。Redis Enterprise Cloud を使用して AWS リソースに接続できます。たとえば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されているアプリケーションサーバーや、AWS Lambda サービスとしてデプロイされているマイクロサービスに接続できます。

前提条件と制限

引き受け

- 現在、クラウドに移行したいオンプレミスのデータベースシステムを運用しています。
- 以下を含むワークロードの移行要件を確認しました。
  - データ整合性の要件
  - インフラストラクチャーとシステム環境の要件
  - データマッピングと変換の要件
  - 機能テストの要件
  - パフォーマンステストの要件
  - 検証要件

  - 定義済みのカットオーバー戦略
- 移行に必要なスケジュールとコストの見積もりを評価しました。
- 要件は、作業の範囲と、移行の一環として特定したシステムやデータベースを考慮に入れています。
- 実行責任者、説明責任者、協議先、報告先 (RACI) マトリックスで、ステークホルダーとその役割と責任を特定しました。
- すべての利害関係者から必要な合意と承認を受けています。

コスト

既存のソースデータベースの技術仕様 (メモリサイズ、スループット、合計データサイズなど) に応じて、Redis ソリューションアーキテクトは Redis Enterprise Cloud 上のターゲットシステムのサイジングを行うことができます。 一般的な料金情報については、Redis ウェブサイトの「Redis 料金」を参照してください。

人材とスキル

移行プロセスには以下の役割と責任があります。

ロール	説明	必要なスキル
移行ソリューションアーキテクト	移行戦略の定義、計画、実装に関する専門知識を持つテクニカルアーキテクト	ソースシステムとターゲットシステムに関する技術レベルおよびアプリケーションレベルの理解、クラウドへのワークロード移行の経験
データアーキテクト	さまざまなデータベースのデータソリューションの定義、実装、提供において幅広い経験を持つテクニカルアーキテクト	構造化データと非構造化データのデータモデリング、企業向けデータベースの実装に関する深い理解と経験
Redis ソリューションアーキテクト	適切なユースケースに最適なサイズの Redis クラスターの設計をアーキテクトできるテクニカルアーキテクト	さまざまなユースケースに対応する Redis ソリューションの設計とデプロイに関する専門知識

クラウドソリューションアーキテクト	クラウドソリューション、特に AWS のソリューションをより深く理解しているテクニカルアーキテクト	クラウド向けソリューションの設計に関する専門知識、ワークロードの移行、アプリケーションのモダナイゼーションの経験
エンタープライズアーキテクト	組織の技術情勢を完全に理解し、future ロードマップについて共通のビジョンを持ち、組織内のすべてのチームで標準化されたアーキテクチャのベストプラクティスを実践および確立するテクニカルアーキテクト	TOGAF などのソフトウェアアーキテクチャ認定資格、基本的なソフトウェアエンジニアリングスキル、ソリューションアーキテクチャ、エンタープライズアーキテクチャの専門知識
IT エンジニアまたは DevOps エンジニア	インフラストラクチャーの問題点の監視、メンテナンスタスクの実行、必要に応じた更新など、インフラストラクチャーの作成と保守を担当するエンジニア。	オペレーティングシステム、ネットワーク、クラウドコンピューティングなど、さまざまなテクノロジーに関する深い知識、Python、Bash、Rubyなどのプログラミング言語や、Docker、Kubernetes、Ansibleなどのツールに精通していること

アーキテクチャ

移行オプション

次の図は、オンプレミス (Redis ベースまたはその他) のデータソースを AWS に移行するためのオプションを示しています。Redis データベース (RDB) ファイルを Amazon Simple Storage Service (Amazon S3) にエクスポートしたり、Redis レプリケーション機能を使用したり、AWS DMS を使用したりするなど、選択できるいくつかの移行ツールを示しています。

1. オンプレミスデータソース: MySQL、PostgreSQL、Oracle、SQL Server、MariaDB など、Redis に基づいていないデータベース。

2. オンプレミスデータソース:Redis OSS や Redis エンタープライズソフトウェアなどの Redis プロトコルベースのデータベース。
3. Redis ベースのデータベースからデータを移行する最も簡単な方法は、RDB ファイルをエクスポートし、ターゲットの Redis Enterprise Cloud on AWS にインポートすることです。
4. また、Redis のレプリケーション機能 (`ReplicaOf`) を使用してソースからターゲットにデータを移行することもできます。
5. データ移行要件にデータ変換が含まれる場合は、Redis 入出力ツール (RIOT) を使用してデータを移行できます。
6. または、AWS Database Migration Service (AWS DMS) を使用して、SQL ベースのデータベースからのデータを移行できます。
7. データをターゲット Redis Enterprise Cloud on AWS に正常に移行するには、AWS DMS の仮想プライベートクラウド (VPC) ピアリングを使用する必要があります。

ターゲットアーキテクチャ

次の図は、Redis Enterprise Cloud on AWS の一般的なデプロイアーキテクチャと、このアーキテクチャを主要な AWS サービスでどのように使用できるかを示しています。

1. Redis Enterprise Cloud on AWS によってサポートされているビジネスアプリケーションに接続できます。
2. ビジネスアプリケーションは、自分の AWS アカウント、そのアカウント内の VPC で実行できます。
3. Redis Enterprise Cloud のデータベースエンドポイントを使用してアプリケーションに接続できます。例としては、EC2 インスタンスで実行されるアプリケーションサーバー、AWS Lambda サービスとしてデプロイされたマイクロサービス、Amazon Elastic Container Service (Amazon ECS) アプリケーション、Amazon Elastic Kubernetes Service（Amazon EKS）アプリケーションなどが含まれます。
4. VPC で実行されるビジネスアプリケーションには、Redis Enterprise Cloud VPC への VPC ピア接続が必要です。これにより、ビジネスアプリケーションはプライベートエンドポイントを介して安全に接続できます。
5. Redis Enterprise Cloud on AWS は、AWSにDBaaAWS としてデプロイされたインメモリNoSQLデータベースプラットフォームであり、Redisによって完全に管理されています。

6. Redis エンタープライズクラウドは、Redis が作成した標準 AWS アカウントの VPC 内にデプロイされます。
7. セキュリティ上の理由から、Redis Enterprise Cloud はプライベートエンドポイントとパブリックエンドポイントの両方からアクセスできるプライベートサブネットにデプロイされます。クライアントアプリケーションをプライベートエンドポイントの Redis に接続することをお勧めします。パブリックエンドポイントを使用する予定の場合は、「TLS を有効」にしてクライアントアプリケーションと Redis Enterprise Cloud 間のデータを暗号化することを強くお勧めします。

Redis の移行方法論は、AWS 規範ガイダンスのウェブサイトの「組織を動員して大規模な移行を加速する」で説明されている AWS 移行方法論と一致しています。

自動化とスケール

移行のための環境設定タスクは、AWS Landing Zone とコードとしての infrastructure as code (IaC) テンプレートを使用して自動化し、自動化と拡張を行うことができます。これらについては、このパターンの「エピック」セクションで説明しています。

ツール

データ移行要件に基づいて、さまざまな技術オプションから選択して、データを Redis Enterprise Cloud on AWS に移行できます。以下の表は、これらのツールの説明と比較です。

ツール	説明	利点	欠点
「RDB のエクスポート」と「インポート」	ソース (Redis OSS や Redis エンタープライズソフトウェアなど) データベースから RDB ファイルの形式でデータをエクスポートします。データベースが Redis OSS クラスターを通じて提供されている場合は、各マスターシャードを RDB にエクスポートします。	• シンプル • RDB 形式のデータをソースとしてエクスポートできるあらゆる Redis ベースのソリューション (Redis OSS や Redis エンタープライズソフトウェアを含む) で動作します。 • 簡単なプロセスでデータ整合性を実現します。	• データ変換要件に対応しておらず、論理的なデータベース統合もサポートしていない。 • データセットが大きいと時間がかかる。 • デルタ移行がサポートされていないと、ダウンタイムが長くなる可能性があります。

	その後、すべての RDB ファイルを 1 回の手順でインポートします。ソースデータベースが OSS クラスターに基づいているが、ターゲットデータベースが OSS Cluster API を使用していない場合は、標準の Redis クライアントライブラリを使用するようにアプリケーションのソースコードを変更する必要があります。
	データ変換要件または論理データベースのマージには、より複雑なプロセスが必要です。これについては、この表の後半の論理データベースのマージで説明しています。

「Redis レプリケーション機能」(アクティブ/パッシブ)	Redis OSS、エンタープライズソフトウェア、またはエンタープライズクラウドデータベースから Redis Enterprise Cloud データベースにデータを継続的に複製できます。初回同期後、Redis レプリケーション機能 (`ReplicaOf`) はデルタ移行を実行します。つまり、アプリケーションのダウンタイムはほとんど発生しません。 Redis のレプリケーション機能はアクティブ/パッシブな方法で使用することを目的としています。ターゲットはパッシブであると想定され、完全に再同期 (ソースデータベースからフラッシュおよび同期) されます。そのため、ソースとターゲットを切り替えるのはやや複雑です。 OSS クラスターのすべてのマスターシャードをソースとして指定することで、Redis OSS クラスターから標準のクラスター化さ	• 連続レプリケーション (初期データロードとそれに続くデルタ) をサポートします。 • ダウンタイムはほとんどありません (レプリケーションの遅延により異なります)。 • データの整合性を確保します。	• アクティブなサイトは 1 つだけなので、サイト間の切り替えはより複雑です。 • OSS クラスターから移行する場合、最大 32 のマスターシャードをサポートします。

	れた Redis Enterprise Cloud データベースに複製できます。ただし、Redis レプリケーション機能では最大 32 のソースデータベースを使用できます。		
AWS DMS	AWS DMS を使用して、サポートされているソースデータベースからターゲットの Redis データストアへ、最小限のダウンタイムでデータを移行することができます。詳細については、AWS DMS ドキュメントの「AWS DMS のターゲットとしての Redis の使用」を参照してください。	• NoSQL と SQL の両方のデータソースの移行をサポートします。 • AWS の他のサービスともうまく連携しています。 • ライブマイグレーションと変更データキャプチャ (CDC) のユースケースをサポートします。	• Redis のキー値には % などの特殊文字を含めることはできません。 • 行やフィールド名に特殊文字を含むデータの移行はサポートされていません。 • フルラージバイナリオブジェクト (LOB) モードはサポートしていません。

論理データベースマージ	特殊なデータベースマージ要件では、カスタムデータ移行ソリューションが必要になる場合があります。たとえば、Redis OSS には 4 つの論理データベース (SELECT 0..3) があっても、データを複数の Redis Enterprise Cloud データベースに移動する代わりに 1 つのデータベースエンドポイントを使用したい場合があります。Redis Enterprise は選択可能な論理データベースをサポートしていないため、ソースデータベースの物理データモデルを変換する必要があります。たとえば、各データベースインデックスをプレフィックス (0 から usr、1 から cmp など) にマップし、移行スクリプトまたは抽出、変換、ロード (ETL) ツールを使用して RDB ファイルを出力し、ターゲットデータベースにインポートできます。	• カスタムスクリプトを使用して、ターゲットシステムへの移行中のデータのシェーピングをきめ細かく制御できます。	• 移行を完了しないことにした場合、特に新しいデータをソースシステムにロールバックする必要がある場合、ロールバックは非常に困難になる可能性があります。 • 1 回限りの移行のための 1 回限りのソリューションを構築することが目標の場合、構築コストが高くなる可能性があります。 • 移行要件が頻繁に変わると、コード、インフラストラクチャ、開発時間、その他の分野のメンテナンスコストが高くなる可能性があります。

さらに、AWS の次のツールとサービスを使用できます。

評価とディスカバリーのツール:

- AWS Application Discovery Service
- 「Migration Evaluator」

アプリケーションおよびサーバー移行ツール:

- AWS Application Migration Service

「データベース移行ツール」:

- 「AWS Schema Conversion Tool (AWS SCT)」
- 「AWS Database Migration Service (AWS DMS)」

「データ移行ツール」:

- AWS Storage Gateway
- AWS DataSync
- AWS Direct Connect
- AWS Snowball
- Amazon Data Firehose

移行管理:

- AWS Migration Hub

AWS パートナーソリューション:

- 「AWS 移行コンピテンシーパートナー」

エピック

検出と評価のタスクを完了してください。

タスク	説明	必要なスキル
ワークロードを特定します。	移行したいワークロードの候補を特定します。移行するワークロードを選択する前に、以下を検討してください。 • このワークロードを移行する、または移行しないことのビジネス価値は何か? • このワークロードがターゲットシステムに正常に移行されない場合の緊急時対応計画はありますか? ビジネスへの影響が最大で、リスクが最小限になるワークロードを選択するのが理想的です。プロセス全体を反復的に行い、少しずつ移行してください。	データアーキテクト、ビジネスチャンピオン、移行プロジェクトスポンサー
データソースと要件を特定し、データモデルを設計します。	Redis はワークショップを実施して、発見までの時間を短縮し、プロジェクトの移行計画を定義しています。このワークショップの一環として、Redis チームはデータソースとソースデータモデルの要件を特定し、それらを Redis Enterprise Cloud でどの	Redis ソリューションアーキテクト

タスク	説明	必要なスキル
	ように改造できるかを分析します。 Redis 移行チーム (プロフェッショナルサービス) は、組織と共に詳細なデータモデル設計演習を行います。この演習の一環として、Redis チームは次のことを行います。 • ターゲット Redis データ構造を識別します。 • データマッピング戦略を定義します。 • 移行アプローチと推奨事項を文書化しています。 • データモデルを利害関係者と検討して最終決定する。	

タスク	説明	必要なスキル
ソースデータベースの特性を特定します。	ソース環境とターゲット環境で使用されている Redis 製品を特定してください。以下に例を示します。 • ソースデータベースは OSS クラスターデータベース、スタンドアロン Redis データベース、または Redis Enterprise データベースですか? • ターゲットデータベースは Redis Enterprise 標準データベースですか、それとも OSS クラスター互換データベースですか? • アプリケーションのソースコードにはどのような影響がありますか?	データアーキテクト
現在のシステム SLA とその他のサイジング指標を収集します。	スループット (1 秒あたりのオペレーション)、レイテンシー、データベースごとの全体的なメモリサイズ、および高可用性 (HA) 要件で表される現在のサービスレベルアグリーメント (SLA) を決定します。	データアーキテクト

タスク	説明	必要なスキル
ターゲットシステムの特性を特定します。	以下の質問に対する答えを決めてください。 • どのくらいの量のデータを移行する必要があるか。 • 一定量のデータを移行するには、どれくらいの時間がかかりますか? • 移行のダウンタイム要件は? サービスやアプリケーションが特定の期間利用できなくなっても問題ありませんか? もしそうなら、どのくらいの期間ですか ? • 移行されたデータにはどの程度一貫性があるべきか? ターゲットデータベースが少し矛盾する (古い) 状態になることはありますか? • データをターゲットデータベースにロードする前に変換する必要がありますか? (たとえば、移行する前に、選択可能な DB インデックスをプレフィックスに変換できます) • ソースデータベースには、ターゲットデータベースのホスト (たとえば、ピア VPC から、または暗号化を使用するパブリックエンドポイントから) からアクセスできますか?	データアーキテクト、Redis ソリューションアーキテクト (オプション)

タスク	説明	必要なスキル
	• Redis のテクニカルアーキテクトと一緒に、データサイジングと Redis クラスターサイジングの演習を行います。 • ネットワーク要件、インフラストラクチャ要件、ソフトウェアバージョン、ソフトウェアライセンスを特定し、移行前にコンポーネントを調達します。 • このデータの転送に関連するセキュリティ上の懸念はありますか?	
依存関係を確認します。	移行する現在のシステムのアップストリームとダウンストリームの依存関係を特定します。移行作業が、依存する他のシステム移行と連動していることを確認してください。たとえば、他のビジネスアプリケーションをオンプレミスから AWS クラウドに移行することを計画している場合は、それらのアプリケーションを特定し、プロジェクトの目標、タイムライン、利害関係者に基づいて調整します。	データアーキテクト、エンタープライズアーキテクト

タスク	説明	必要なスキル
移行ツールを特定します。	データ移行要件 (ソースデータやダウンタイム要件など) に応じて、前述の「ツール」セクションで説明したどのツールでも使用できます。さらに、次のことが行えます。 • CRDB デプロイメントによる双方向 (アクティブ-アクティブ) レプリケーション。 • カスタムのエクスポート/インポートスクリプト (DUMP/RESTORE コマンドの使用などによる)。 • 「RIOT」、「EcStats2」、ETL ツールなどのその他のエクスポート/インポートツールやヘルパーツール。 • Terraform や AWS CloudFormation テンプレートなどの IaC ツール。	移行ソリューションアーキテクト、Redis ソリューションアーキテクト
コンティンジェンシープランを作成します。	移行中に問題が発生した場合に備えて、ロールバックするための緊急時対応計画を立ててください。	プロジェクト管理、アーキテクトを含む技術チーム

セキュリティとコンプライアンスのタスクを完了してください。

タスク	説明	必要なスキル
Redis 管理コンソールをセキュリティで保護します。	管理コンソールを保護するには、「Redis」ドキュメントの指示に従ってください。	IT インフラストラクチャ管理者
Redis データベースを保護します。	Redis ドキュメントの以下のページを参照してください。 • 「ロールベースアクセスコントロールを定義します」。 • 「ネットワークセキュリティの定義」。 • 「TLS の有効化」。	
セキュアな Redis クラウド API。	「API を有効化する」と、Redis Cloud アカウントのすべての所有者の「API キーを管理」できます。API のセキュリティ機能の概要については、Redis ウェブサイトの「API 認証ドキュメント」を参照してください。	IT インフラストラクチャ管理者

新環境のセットアップ

タスク	説明	必要なスキル
AWS に新しい環境を設定します。	このタスクには以下が含まれます。 • 「AWS Landing Zone」のセットアップアクティビ	IT エンジニアまたは DevOps エンジニア

タスク	説明	必要なスキル
	ティ。ランディングゾーンは以下をサポートします。 • マルチアカウントデプロイ • 最低限のセキュリティベースライン • セキュリティベースラインと ISV の前提条件 (ネットワーク、セキュリティ設定など) を使用して新しいアカウントをプロビジョニングする自動方法 • 通知、集中ロギング、監視 • ISV ソフトウェア設定アクティビティ。これには、製品やワークロードの設定、変更など、移行に含める必要のある設定が含まれます。 • AWS CloudFormation または Terraform テンプレートの設定やカスタマイズなどの IaC アクティビティ。	

タスク	説明	必要なスキル
移行アーキテクチャをデプロイします。	1. AWS に Redis エンタープライズクラウドをセットアップします。 2. RIOT や AWS DMS などの移行ツールをインストールします。使用可能なツールのリストについては、「ツール」セクションを参照してください。 3. アプリケーション、マイグレーション、データベースレイヤー間の接続を確立します。 4. 各レイヤーを通過できるサンプルワークロードを作成し、少量のサンプルデータを移行します。 これで、実際のデータ移行パイプラインを実行してテストできます。	IT エンジニアまたは DevOps エンジニア

ネットワークを設定する

タスク	説明	必要なスキル
接続を確立します。	オンプレミスインフラストラクチャと AWS クラウドリソース間の接続を確立します。この機能を実現するには、セキュリティグループ、AWS Direct Connect、およびその他のリソースを使用してくだ	IT エンジニアまたは DevOps エンジニア

タスク	説明	必要なスキル
	さい。詳細については、AWS ウェブサイトの「データセンターを AWS に接続」を参照してください。	
VPC ピアリングの設定	ビジネスアプリケーションを実行する VPC (または移行ツールや AWS DMS レプリケーションサーバーを実行する EC2 インスタンス) と Redis Enterprise Cloud を実行する VPC との間で VPC ピアリングを確立します。手順については、Amazon VPC ドキュメントの「Amazon VPC の使用開始」を、Redis ドキュメントの「VPC ピアリングの有効化」を参照してください。	IT エンジニアまたは DevOps エンジニア

データを移行する

タスク	説明	必要なスキル
データ移行ツールを選択してください。	「ツール」セクションのテーブルで、これらのツールの説明、長所、短所を確認してください。 • RDS エクスポートとインポート • Redis レプリケーション機能 (`ReplicaOf`) • AWS DMS	移行ソリューションアーキテクト

タスク	説明	必要なスキル
	• 論理データベースマージ 次の行には、各ツールに関連するデータ移行タスクが説明されています。	

タスク	説明	必要なスキル
オプション 1: RDB のエクスポートとインポートを使用する。	1. ソースを切断する:(ビジネスアプリケーションを切断するなどして) ソースデータベースのトラフィックを停止します。 2. エクスポート:ソースデータベースのデータを RDB ファイルとしてエクスポートします。 3. ステージ: AWS 上の Redis Enterprise Cloud インスタンスにアクセスできる場所にデータをアップロードします (たとえば、S3 バケットや FTP サーバーにアップロードできます)。 4. インポート:RDB ファイルを (1 回のインポートステップですべて一覧表示して) Redis Enterprise Cloud のターゲットデータベースにインポートします。 5. 切り取り:ターゲットデータベースに移動します (たとえば、アプリケーションを接続してターゲットデータベースに接続します)。 詳細については、「Redis ドキュメント」を参照してください。	移行ソリューションアーキテクト、Redis ソリューションアーキテクト

タスク	説明	必要なスキル
オプション 2: Redis のレプリケーション機能 (アクティブ/パッシブ) を使用します。	1. データベースへのConnect: `ReplicaOf` ソースデータベースとターゲットデータベース間のリンクを確立します。 2. 初期同期の実行:ソースデータベースとターゲットデータベース間の初期同期が完了するまで待ちます。 3. ソースを切断する:(アプリケーションを切断するなどして) ソースデータベースのトラフィックを停止します。 4. デルタレプリケーションの実行:デルタがターゲットデータベースにレプリケートされるまで待ちます。 5. カットオーバー:(アプリケーションを接続するなどして) ターゲットデータベースに移動します。 6. 削除:ソースデータベースとターゲットデータベース間の `ReplicaOf` リンクを削除します。 詳細については、「Redis ドキュメント」を参照してください。	移行ソリューションアーキテクト、Redis ソリューションアーキテクト

タスク	説明	必要なスキル
オプション 3: AWS DMS を使用する。	1. AWS DMS レプリケーションインスタンスのセットアップ:このインスタンスはすべての移行プロセスを実行します。手順については、AWS DMS「ドキュメントの AWS DMS レプリケーションインスタンスの操作」を参照してください。 2. ソースデータベースの定義:ソースエンドポイントを定義します。ソースエンドポイントと AWS DMS レプリケーションサーバー間の接続をテストします。手順については、AWS DMS ドキュメントの「ソースエンドポイントとターゲットエンドポイントの作成」を参照してください。 3. ターゲットデータベースのセットアップ:AWS で Redis Enterprise Cloud をセットアップし、移行先のデータベースを設定します。 4. ターゲットデータベースの定義:ターゲットエンドポイントを定義します。AWS DMS が実行されている VPC と AWS で Redis Enterprise Cloud をホストする VPC との間で「VPC	移行ソリューションアーキテクト、Redis ソリューションアーキテクト

タスク	説明	必要なスキル
	ピアリング」が確立されていることを確認します。AWS DMS レプリケーションサーバーとターゲットデータベース間の接続をテストします。 5. AWS DMS タスクを作成する:データの移行に使用するテーブルとレプリケーションプロセスを定義するタスクまたはタスクセットを作成します。手順については、AWS DMS ドキュメントの「AWS DMS タスクの操作」を参照してください。 6. 移行: AWS DMS タスクを実行してデータを移行します。 7. カットオーバー:(アプリケーションを接続するなどして) ターゲットデータベースに移動します。	
オプション 4: 論理的なデータベースマージを使用する。	このオプションでは、ソースデータベースの物理データモデルを変換し、RDB ファイルを生成できる移行スクリプトまたは ETL ツールを使用します。Redis プロフェッショナルサービスは、必要に応じてこのステップを支援します。	移行ソリューションアーキテクト、Redis ソリューションアーキテクト

アプリケーションを移行する

タスク	説明	必要なスキル
プロジェクト管理のスケジュールと目標を調整する。	アプリケーションレイヤーの移行プロジェクトの目標、マイルストーン、タイムラインを Redis データ移行プロジェクトの目標と一致させてください。	プロジェクト管理
テストアクティビティを調整します。	アプリケーションレイヤーを AWS クラウドに移行して最新化したら、アプリケーションレイヤーを新しく移行した Redis Enterprise Cloud on AWS にポイントしてテストします。	テスト

テスト

タスク	説明	必要なスキル
テストプランを実装します。	実装フェーズで開発されたデータ移行ルーチンとスクリプトを、テスト要件に従ってサイトで実行します。	テスト
テストデータの品質	データを移行した後、データ品質をテストします。	テスト
機能をテストします。	データクエリとアプリケーション層をテストして、アプリケーションがソースシステムと同じレベルで動作していることを確認します。	テスト

カットオーバー

タスク	説明	必要なスキル
カットオーバーの決定を下してください。	アプリケーションレベルとデータベースレベルのテストがすべて完了したら、エグゼクティブリーダーシップチームと利害関係者は、テストチームが確認した最終結果に基づいて、AWS の新しい環境に移行するかどうかの最終決定を下します。	プロジェクト管理、ビジネスチャンピオン
AWS クラウドに切り替えます。	すべてが整っていることを確認したら、アプリケーション層を新しく移行したデータを指し、クライアントは AWS 上の新しい Redis Enterprise Cloud システムに基づいて実行されている新しいアプリケーション層を指します。	IT または DevOps エンジニア、データアーキテクト、移行ソリューションアーキテクト、Redis ソリューションアーキテクト

関連リソース

Redis リソース

- 「Redis エンタープライズクラウド-ドキュメント」
- 「RIOT」ツール (GitHub リポジトリ)
- 「テラフォームプロバイダー」(ダウンロード)

「AWS リソース」

- 「デモマイグレーション」
- 「AWS パートナーソリューション」
- ドキュメント

- ブログ記事
- 「ホワイトペーパー」
- チュートリアルと動画
- 「AWS クラウド移行」
- AWS 規範ガイダンス

追加情報

Redis ワークロードを AWS クラウドに移行するための標準セキュリティ要件については、AWS ウェブサイトの「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」と、Redis ウェブサイトの「Redis トラストセンター」を参照してください。

AWS SCT と AWS DMS を使用して Amazon EC2 上の SAP ASE を Amazon Aurora PostgreSQL 互換の Amazon Aurora PostgreSQL 互換に移行します

作成者: Amit Kumar (AWS) と Ankit Gupta (AWS)

概要

このパターンでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされている SAP Adaptive Server Enterprise (SAP ASE) データベースを、AWS Schema Conversion Tool (AWS SCT) と AWS Database Migration Service (AWS DMS) を使用して Amazon Aurora PostgreSQL 互換エディションに移行する方法を説明します。このパターンは、保存されたオブジェクトのデータ定義言語 (DDL) 変換とデータ移行の両方に焦点を当てています。

Aurora PostgreSQL 互換では、オンライントランザクション処理 (OLTP) ワークロードがサポートされます。このマネージドサービスは、必要に応じて自動的にスケーリングする構成を提供します。アプリケーションのニーズに基づいて、データベースを自動的に起動、シャットダウン、スケールアップ、またはスケールダウンできます。データベースインスタンスを管理しなくても、クラウドでデータベースを実行できます。Aurora PostgreSQL 互換では、使用頻度が低く、断続的、または予測不可能なワークロードのための、コスト効率の高いオプションが提供されています。

移行プロセスは主に 2 つのフェーズで構成されています。

- AWS SCT を使用したデータベーススキーマの変換
- AWS DMS を使用してデータを移行する

両方のフェーズの詳細な手順は、「エピック」セクションに記載されています。SAP ASE データベースで AWS DMS を使用する場合に固有の問題のトラブルシューティングについては、AWS DMS ドキュメントの「SAP ASE に関する問題のトラブルシューティング」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- サーバー、データベース、リスナーサービスが稼働している EC2 インスタンス上のソース SAP ASE データベース
- ターゲット Aurora PostgreSQL-Compatible データベース

制約事項

- 接続のポート番号は、必ず5432にします。
- 「huge_pages」機能はデフォルトでオンになっていますが、変更できます。
- ポイントインタイムリカバリ (PITR) の精度は 5 分です。
- クロスリージョンレプリケーションは現在使用できません。
- Aurora データベースの最大ストレージサイズは 128 TiB です。
- 最大 15 つのリードレプリカを作成できます。
- テーブルサイズの制限は Aurora クラスターボリュームのサイズによってのみ制約されるため、Aurora PostgreSQL 互換 DB クラスターの最大テーブルサイズは 32 TiB です。大きいテーブルの分割など、テーブル設計のベストプラクティスにしたがうことをお勧めします。

製品バージョン

- ソースデータベース：AWS DMS は現在 SAP ASE 15、15.5、15.7、16.x をサポートしています。SAP ASE バージョンサポートの最新情報については、「AWS DMS ユーザーガイド」を参照してください。
- ターゲットデータベース：PostgreSQL 9.4 移行 (バージョン 9.x), 10.x, 11.x, 12.x, 13.x, and 14.x. サポートされている最新の PostgreSQL バージョンについては、「AWS DMS ユーザーガイド」を参照してください。
- Amazon Aurora 1.x 以降。最新情報については、Aurora ドキュメントの「Aurora PostgreSQL 互換リリースとエンジンバージョン」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Amazon EC2 で稼働している SAP ASE データベース

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換 データベース

移行アーキテクチャ

ツール

- 「Amazon Aurora PostgreSQL 互換エディション」 は、PostgreSQL デプロイのセットアップ、運用、スケーリングを支援するフルマネージド型で ACID 準拠のリレーショナルデータベースエンジンです。
- 「AWS Schema Conversion Tool (AWS SCT)」は、ソースデータベーススキーマとほとんどのカスタムコードをターゲットデータベースと互換性のある形式に自動的に変換することで、異種データベース移行をサポートします。
- 「AWS DMS」では、複数のソースとターゲットのデータベースがサポートされています。詳細については、AWS DMS ドキュメントの「データ移行のソース」と「データ移行のターゲット」を参照してください。最も包括的なバージョンと機能サポートのため、最新バージョンの AWS DMS を使用することをお勧めします。

エピック

環境をセットアップする

タスク	説明	必要なスキル
ソース EC2 インスタンスでネットワークアクセスを設定します。	ソース SAP ASE データベースをホストする EC2 インスタンスにセキュリティグループを設定します。 手順については、Amazon EC2 ドキュメントの Linux インスタンス用の Linux インスタンス用の「Amazon EC2 セキュリティグループ」を参照してください。	システム管理者
Aurora PostgreSQL 互換 DB クラスターを作成します。	ターゲットデータベース用の Aurora PostgreSQL 互換クラスターをインストール、設定、起動します。 詳細については、Auroraドキュメントの Amazon Aurora	DBA

タスク	説明	必要なスキル
	DB クラスターに接続するを参照してください。	
ターゲット DB クラスターの認可を設定します。	ターゲットデータベースのセキュリティグループとファイアウォールを設定します。 手順については、Aurora ドキュメントの「Amazon Aurora DB クラスターの作成」を参照してください。	DBA、システム管理者

AWS SCT によるデータベーススキーマの変換

タスク	説明	必要なスキル
AWS SCT を起動します。	「AWS SCT ドキュメント」の指示に従って AWS SCT を起動します。 AWS SCT には、SAP ASE ソースデータベースのデータベーススキーマをターゲット Aurora PostgreSQL-Compatible インスタンスと互換性のある形式に自動変換するための、プロジェクトベースのユーザーインターフェイスが用意されています。	DBA
AWS SCT エンドポイントを作成します。	ソースSAP ASEデータベースとターゲット PostgreSQL データベースの エンドポイントを作成します。	DBA

タスク	説明	必要なスキル
	手順については、AWS SCTドキュメントを参照してください。	
評価レポートを生成します。	データベース移行評価レポートを作成して移行を評価し、互換性のないオブジェクトや機能を検出します。 手順については、AWS SCTドキュメントを参照してください。	DBA
スキーマを変換します。	「AWS SCT ドキュメント」の指示に従ってデータベーススキーマを変換します。	DBA
データベースオブジェクトを検証します。	AWS SCT がデータベースオブジェクトを変換できない場合、その名前とその他の詳細を識別します。これらのオブジェクトは手動で変換する必要があります。 これらの不一致を特定するには、AWS ブログ記事「SAP ASE から Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL に移行した後のデータベースオブジェクトの検証」の手順に従ってください。	DBA

AWS DMS の移行を分析する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。	SAP ASE データベースのバージョンで AWS DMS との互換性を確認してください。 詳細については、AWS DMS ドキュメントの「AWS DMS のソース」と「AWS DMS のターゲット」を参照してください。	DBA
ストレージタイプと容量の要件を特定します。	ソースデータベースのサイズに基づいて、ターゲットデータベースの適切なストレージ容量を選択します。	DBA、システム管理者
レプリケーションインスタンスのインスタンスタイプ、容量、その他の機能を選択します。	要件を満たすインスタンスタイプ、容量、ストレージ機能、ネットワーク機能を選択します。 ガイダンスについては、AWS DMS ドキュメントの「移行に適した AWS DMS レプリケーションインスタンスの選択」を参照してください。	DBA、システム管理者
ネットワークアクセスのセキュリティ要件を特定する。	ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。 AWS DMS ドキュメントの「レプリケーションインスタンス用のネットワークのセッ	DBA、システム管理者

タスク	説明	必要なスキル
	トアップ」のガイダンスに従ってください。	

データを移行する

タスク	説明	必要なスキル
AWS DMS に移行タスクを作成してデータを移行します。	データ移行するには、タスクを作成し、AWS DMSドキュメントの手順に従います。 最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。	DBA
データを検証します。	データがソースデータベースからターゲットデータベースに正確に移行されたことを確認するため、AWS DMS ドキュメントに記載されている「データ検証ガイドライン」に従ってください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略を特定する。	アプリケーションをクラウドに移行するための「7 つの戦略 (7R)」から 1 つを選択してください。	DBA、アプリ所有者、システム管理者
アプリケーション移行戦略に従ってください。	ターゲットデータベースの DNS 接続詳細の更新や動的ク	DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
	エリの更新など、アプリケーションチームが指定したデータベースタスクを完了します。	

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。	ソースデータベースからターゲットデータベースへの接続を切り替えます。 詳細については、「リレーショナルデータベースの移行戦略」の「カットオーバー」セクションを参照してください。	DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。	すべての移行タスク、レプリケーションインスタンス、エンドポイント、およびその他の AWS SCT および AWS DMS リソースを終了します。 詳細については、AWS DMS のドキュメントを参照してください。	DBA、システム管理者
プロジェクト文書を確認して検証する。	プロジェクト文書のすべてのステップを検証して、すべて	DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
	のタスクが正常に完了したことを確認します。	
プロジェクトを閉じます。	移行プロジェクトを閉じて、フィードバックを送ってください。	DBA、アプリ所有者、システム管理者

関連リソース

リファレンス

- Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする（AWS 規範ガイダンス）
- pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する（AWS 規範ガイダンス）
- Amazon Aurora の価格設定
- Amazon Aurora PostgreSQL 互換エディションのベストプラクティス (Amazon Aurora ドキュメント)
- AWS SCT のドキュメント
- AWS DMS のドキュメント
- SAP ASE データベースを &DMS; のソースとして使用する

チュートリアルと動画

- AWS Database Migration Service の使用開始
- AWS Database Migration Service（動画）

ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行

作成者：Chandra Sekhar Yaratha (AWS) と Igor Kovalchuk (AWS)

概要

このパターンでは、AWS Certificate Manager (ACM) を使用して、オンプレミスサーバーでホストされているウェブサイトまたは Microsoft インターネットインフォメーションサービス (IIS) 上の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスから既存のセキュアソケットレイヤー (SSL) 証明書を移行するためのガイダンスを提供します。その後、SSL 証明書は AWS の Elastic Load Balancing で使用できます。

SSL はデータを保護し、本人確認を行い、検索エンジンのランキングを向上させ、ペイメントカード業界データセキュリティ基準 (PCI DSS) の要件を満たすのに役立ち、顧客の信頼を高めます。これらのワークロードを管理する開発者や IT チームは、IIS サーバーや Windows サーバーなどの ウェブアプリケーションとインフラストラクチャがベースラインポリシーに準拠し続けることを望んでいます。

このパターンでは、既存の SSL 証明書を Microsoft IIS から手動でエクスポートし、個人情報交換 (PFX) 形式から ACM がサポートするプライベート拡張メール (PEM) 形式に変換し、AWS アカウントの ACM にインポートします。また、アプリケーションのApplication Load Balancer を作成し、インポートした証明書を使用するようにApplication Load Balancer を設定する方法についても説明します。その後、HTTPS 接続はApplication Load Balancer で終了されるため、ウェブサーバー上で追加の構成オーバーヘッドが発生することはありません。詳細については、 のCreate an HTTPS Listener for Your Application Load Balancerを参照してください。

Windows サーバーは.pfx または.p12 ファイルを使用して、パブリックキーファイル (SSL 証明書) と固有のプライベートキーファイルを格納します。認証局 (CA) が公開キーファイルを提供します。サーバーを使用して、証明書署名要求 (CSR) が作成された関連する秘密キーファイルを生成します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- ターゲットが使用する各アベイラビリティーゾーンで少なくとも 1 つのプライベートサブネットと 1 つのパブリックサブネットを持つ AWS の仮想プライベートクラウド (VPC)
- Windows サーバー2012 以降で実行されている IIS バージョン 8.0 以降
- IIS で実行されているウェブアプリケーション

- IIS サーバーへの管理者アクセス。

アーキテクチャ

ソーステクノロジースタック

- SSL を使用した IISウェブサーバーの実装により、データが暗号化された接続 (HTTPS) で安全に送信されるようにします。

ソースアーキテクチャ

ターゲットテクノロジースタック

- AWS アカウントの ACM 証明書
- インポートされた証明書を使用するように設定されたApplication Load Balancer
- プライベートサブネット内の Windows サーバーインスタンス

ターゲット アーキテクチャ

ツール

- AWS Certificate Manager (ACM) は、 ウェブサイトやアプリケーションを保護するパブリックおよびプライベート SSL/TLS X.509 証明書およびキーの作成、保存、更新に伴う複雑さに対処します。
- Elastic Load Balancing (ELB) は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。たとえば、1 つ以上のアベイラビリティーゾーンの EC2 インスタンス、コンテナ、IP アドレスにトラフィックを分散できます。

ベストプラクティス

- HTTP から HTTPS へのトラフィックリダイレクトを強制します。
- Application Load Balancer のセキュリティグループを適切に設定して、特定のポートへのインバウンドトラフィックのみを許可します。
- 複数のアベイラビリティーゾーンで EC2 インスタンスを起動し、高可用性を確保します。

- IP アドレスの代わりに、アプリケーションロードバランサーの DNS 名を指すようにアプリケーションのドメインを設定します。
- Application Load Balancer にアプリケーションレイヤーのヘルスチェック が設定されていることを確認します。
- ヘルスチェックのしきい値を設定します。
- Amazon CloudWatch を使用してApplication Load Balancer をモニターします。

エピック

.pfx ファイルをエクスポート

タスク	説明	必要なスキル
Windows サーバーから.pfx ファイルをエクスポートします。	Windows Server のオンプレミスの IIS マネージャーから SSL 証明書を.pfx ファイルとしてエクスポートするには： 1. スタート、管理者、インターネットインフォメーションサービス (IIS) マネージャーを選択します。 2. サーバー名を選択し、セキュリティでサーバー証明書をダブルクリックします。 3. エクスポートする証明書を選択し、エクスポートを選択します。 4. 証明書のエクスポートボックスで、.pfx ファイルの場所、パス、名前を選択します。 5. .pfx ファイルのパスワードを指定して確認します。	システム管理者

タスク	説明	必要なスキル
	> **Note** このパスワードは、.pfx ファイルをインストールするときに必要です。 6. [OK] を選択します。 これで、.pfx ファイルが指定した場所とパスに保存されるはずです。	

PFX でエンコードされた証明書を PEM 形式に変換します。

タスク	説明	必要なスキル
OpenSSL ツールキットをダウンロードし、インストールします。	1. シャイニングライトプロダクションズのウェブサイトから Win32/Win64 OpenSSL をダウンロードしてインストールします。 2. OpenSSL バイナリの場所をシステム PATH 変数に追加して、バイナリをコマンドラインで使用できるようにします。	システム管理者
PFX でエンコードされた証明書を PEM 形式に変換します。	次の手順では、PFX でエンコードされた署名付き証明書ファイルを PEM 形式の 3 つのファイルに変換します。	システム管理者

タスク	説明	必要なスキル
	• cert-file.pem リソースの SSL/TLS 証明書が含まれます。 • privatekey.pem パスワード保護されていない証明書のプライベートキーが含まれます。 • ca-chain.pem CA のルート証明書が含まれます。 PFX でエンコードされた証明書を変換するには： 1. Windows PowerShell を開きます。 2. 次のコマンドを使用して、プライベートキーと証明書をPFXファイルから抽出します。プロンプトが表示されたら、証明書のパスワードを入力します。 `openssl pkcs12 -in <filename>.pfx -nocerts -out withpw-privatekey.pem` このコマンドは、privatekey.pem という名前の PEM エンコードされたプライベートキーファイルを生成します。プロンプトが表示されたら、プライベートキーファイル	

タスク	説明	必要なスキル

	を保護するパスフレーズを入力します。
	3. 次のコマンドを実行して、パスフレーズを設定します。プロンプトが表示されたら、ステップ 2 で作成したパスフレーズを入力します。
	`openssl rsa -in withpw-privatekey.pem -out privatekey.pem`
	コマンドが成功する場合、「RSA キーの書き込み中」というメッセージが表示されます。
	4. 次のコマンドを使用して、PFX ファイルから PEM ファイルに証明書を転送します。
	`openssl pkcs12 -in <file_name>.pfx -clcerts -nokeys -out cert-file.pem`
	このコマンドは `cert-file.pem` という名前の PEMエンコードされた証明書を作成します。コマンドが成功する場合、「MAC 検証済み OK」というメッセージが表示されます。	

タスク	説明	必要なスキル
	5. PFX ファイルから CA チェーンファイルを作成します。次のコマンドを実行して、ca-chain.pem という名前のファイルを作成します。 `openssl pkcs12 -in <file_name>.pfx -cacerts -nokeys -chain -out ca-chain.pem` コマンドが成功する場合、「MAC 検証済み OK」というメッセージが表示されます。	

ACM に証明書をインポートします。

タスク	説明	必要なスキル
証明書をインポートするように準備します。	ACM コンソール で証明書のインポートを選択します。	クラウド管理者
証明書本文を提供します。	証明書本文の場合、インポートする PEM エンコードされた証明書を貼り付けます。 このエピックの他のタスクで説明されているコマンドと手順の詳細については、ACM ドキュメントの証明書のインポート を参照してください。	クラウド管理者
証明書のプライベートキーを指定します。	証明書のプライベートキーの場合、PEM エンコードされ、	クラウド管理者

タスク	説明	必要なスキル
	証明書のパブリックキーに一致する暗号化されていないプライベートキーを貼り付けます。	
証明書チェーン。	証明書チェーンでは、`CertificateChain.pem` ファイルに保存されている PEM エンコードされた証明書チェーンを貼り付けます。	クラウド管理者
証明書をインポートする	レビューとインポートを選択します。証明書に関する情報が正しいことを確認し、インポートを選択します。	クラウド管理者

「Application Load Balancer の作成」

タスク	説明	必要なスキル
ロードバランサーとリスナーを作成し、設定します。	Elastic Load Balancing ドキュメント の指示に従って、ターゲットグループを設定し、ターゲットを登録し、Application Load Balancer とリスナーを作成します。ポート 443 に 2 つ目のリスナー (HTTPS) を追加します。	クラウド管理者

トラブルシューティング

問題	ソリューション
Windows PowerShell は、システムパスに OpenSSL コマンドを追加した後でも、そのコマンドを認識しません。	`$env:path` をチェックして、OpenSSL バイナリの場所が含まれていることを保証します。 できない場合、PowerShell で次のコマンドを実行します。 `$env:path = $env:path + ";C:\OpenSSL-Win64\bin"`

関連リソース

ACM への証明書のインポート

- ACM コンソール
- インポートのための証明書とキー形式
- 証明書のインポート
- AWS 証明書マネージャーユーザーガイド

Application Load Balancer の作成

- Application Load Balancer の作成
- Application Load Balancer のユーザーガイド

メッセージキューを Microsoft Azure Service Bus から Amazon SQS に移行

作成者：Nisha Gambhir (AWS)

概要

このパターンでは、.NET Framework、.NET Core ウェブまたはコンソールアプリケーションを Microsoft Azure Service Bus キューメッセージングプラットフォームを使用していたものから Amazon Simple Queue Service (Amazon SQS) に移行する方法を説明します。

アプリケーションはメッセージングサービスで、他のアプリケーションとの間でデータを送受信します。これらのサービスは、分離された拡張性の高いマイクロサービス、分散システム、サーバーレスアプリケーションをクラウド上に構築するのに役立ちます。

Azure Service Bus キューは、キューイングとパブリッシュ/サブスクライブメッセージングをサポートする幅広い Azure メッセージングインフラストラクチャの一部となっています。

Amazon SQS は、フルマネージド型のメッセージキューイングサービスであり、マイクロサービス、分散システム、およびサーバーレスアプリケーションのデカップリングとスケーリングを容易にします。Amazon SQS は、メッセージ指向ミドルウェアの管理と運用に伴う複雑さとオーバーヘッドを排除し、開発者が差別化作業に集中できるようにします。Amazon SQS を使用すると、メッセージを失い、または他のサービスを利用する必要がなく、ソフトウェアコンポーネント間でメッセージを送受信、保存できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AAzure Service Bus キューを使用する .NET Framework、.NET Core ウェブまたはコンソールアプリケーション (サンプルコードが添付されています)

製品バージョン

- .NET Framework 3.5 以降または .NET Core 1.0.1、2.0.0 以降

アーキテクチャ

ソーステクノロジースタック

- Azure Service Busキューを使用してメッセージを送信する .NET (コアまたはフレームワーク) ウェブアプリケーションまたはコンソールアプリケーション

ターゲットテクノロジースタック

- Amazon SQS

ツール

ツール

- Microsoft Visual Studio

コード

Amazon SQS のために、AWS Identity and Access management (IAM) ポリシーを作成するには:

1. AWS マネジメントコンソールにサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

2. 左側のナビゲーションペインで、[Policies] (ポリシー)、[Create policy] (ポリシーの作成) の順にクリックします。

3. [JSON] タブを選択し、以下のコードを貼り付けます。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "sqs:DeleteMessage",
            "sqs:GetQueueUrl",
            "sqs:ChangeMessageVisibility",
            "sqs:SendMessageBatch",
            "sqs:ReceiveMessage",
            "sqs:SendMessage",
            "sqs:GetQueueAttributes",
            "sqs:ListQueueTags",
```

```
            "sqs:ListDeadLetterSourceQueues",
            "sqs:DeleteMessageBatch",
            "sqs:PurgeQueue",
            "sqs:DeleteQueue",
            "sqs:CreateQueue",
            "sqs:ChangeMessageVisibilityBatch",
            "sqs:SetQueueAttributes"
         ],
         "Resource": "arn:aws:sqs:*:<AccountId>:*"
      },
      {
         "Sid": "VisualEditor1",
         "Effect": "Allow",
         "Action": "sqs:ListQueues",
         "Resource": "*"
      }
   ]
}
```

4. [ポリシーの確認] を選択し、[名前] に入力して [ポリシーの作成] を選択します。

5. 新しく作成したポリシーを既存の IAM ロールにアタッチするか、新しいロールを作成します。

エピック

AWS で Amazon SQS をセットアップ

タスク	説明	必要なスキル
Amazon SQS IAMポリシーを作成します。	Amazon SQS へのアクセスを実現する IAM ポリシーを作成します。サンプルポリシーについては、「コード」セッションを参照してください。	システムエンジニア
AWS プロファイルを作成します。	AWS Tools for PowerShell コマンド「Set-AWScredential」を実行して、新しいプロファイルを作成します。アクセスキーとシークレットキーは、指定したプロファイル名の下	システムエンジニア

タスク	説明	必要なスキル
	のデフォルトの認証情報ファイルに保存されます。前に作成した Amazon SQS ポリシーをこのアカウントにリンクします。アクセスキー ID とシークレットアクセスキーを保管します。これらは次のステップで必要になります。	
SQS キューを作成します。	標準キューまたは先入れ先出し (FIFO) キューを作成できます。手順については、「参考文献」セクションのリンクを参照してください。	システムエンジニア

.NET アプリケーションコードを見直してください。

タスク	説明	必要なスキル
AWS Toolkit for Visual Studio をインストールします。	このツールキットは Microsoft Visual Studio の拡張機能で、.NET アプリケーションを AWS で簡単に構築してデプロイできます。インストールと使用方法については、「参考文献」セクションのリンクを参照してください。	アプリケーション開発
AWSSDK.SQS NuGet パッケージをインストールします。	AWSSDK.SQS をインストールするには、Visual Studio で [Manage NuGet Package] を選択するか、 [Install-Package AWSSDK.SQS] コマンドを実行します。	アプリケーション開発

タスク	説明	必要なスキル
.NET アプリケーションに AWSCredentials オブジェクトを作成します。	添付ファイルのサンプルアプリケーションは、AWSCredentials から継承する BasicAWScredentials オブジェクトを作成する方法を示しています。先ほど使用したアクセスキー ID とシークレットアクセスキーを使用することも、実行時にユーザープロファイルの一部として.aws フォルダーからオブジェクトにこれらを選択させることもできます。	アプリケーション開発
SQS クライアントオブジェクトを作成します。	.NET フレームワーク用の SQS クライアントオブジェクト (AmazonSQSClient) を作成します。これは Amazon.SQS ネームスペースの一部です。このオブジェクトは Microsoft.Azure.ServiceBus ネームスペースの一部である iQueueClient の代わりに必要です。	アプリケーション開発
SendMessageAsync メソッドを呼び出して SQS キューにメッセージを送信します。	メッセージをキューに送信するコードを変更して AmazonSQSClient.SendMessageAsync メソッドを使用します。詳細については、添付のコードサンプルを参照してください。	アプリケーション開発

タスク	説明	必要なスキル
SQS キューからメッセージを受信するには、ReceiveMessageAsync メソッドを呼び出します。	メッセージを受信するコードを変更して AmazonSQSClient.ReceiveMessageAsync メソッドを使用します。詳細については、添付のコードサンプルを参照してください。	アプリケーション開発
DeleteMessageAsync メソッドを呼び出して SQS キューからメッセージを削除します。	メッセージを削除するには、コードを QueueClient.CompleteAsync メソッドから AmazonSQSClient.DeleteMessageAsync メソッドに変更します。詳細については、添付のコードサンプルを参照してください。	アプリケーション開発

関連リソース

- .NET 用 AWS SDK 開発者ガイド
- 「Amazon SQS を使用したメッセージング」
- 「AWS SDK for .NET による Amazon SQS キューの作成と使用」
- Amazon SQS メッセージの送信
- 「Amazon SQS キューからメッセージを受信する」
- 「Amazon SQS キューからメッセージを削除する」
- AWS Toolkit for Visual Studio

追加情報

このパターンには 2 つのサンプルアプリケーションを含んでいます (添付ファイルのセクションを参照)。

- AzureSBTestApp には Azure Service Bus キューを使用するコードを含んでいます。

- AmazonSqsTestAppは Amazon SQS を使用します。これは.NET Core 2.2 を使用するコンソールアプリケーションで、メッセージの送受信の例を含んでいます。

注記:

- queueClient は iQueueClient のオブジェクトで、Microsoft.Azure.ServiceBusネームスペース (Microsoft.Azure.ServiceBus NuGet パッケージに含まれている) の一部です。
- amazonSQSClient は Amazon.SQS ネームスペース (AWSSDK.SQS NuGet パッケージに含まれている) の一部である AmazonSQSClient のオブジェクトです。
- コードが実行されている位置 (EC2 で実行されているかどうかなど) によっては、ロールに SQS キューに書き込む権限が必要です。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

でリレーショナルデータベースを MongoDB Atlas に移行する AWS

作成者: Battulga Purevragchaa (AWS)、Babu Srinivasan (MongoDB)、Igor Alekseev (AWS)

概要

このパターンでは、 で SQL Server、MySQL、PostgreSQL などのリレーショナルデータベースから MongoDB Atlas に移行する手順について説明します AWS クラウド。MongoDB Relational Migrator を使用して、リレーショナルデータベースから MongoDB Atlas へのデータ移行を高速化します。

このパターンは、 AWS 規範的ガイダンスウェブサイトの「 での MongoDB Atlas AWS への移行」ガイドに付属しています。そのガイドで説明されている移行シナリオの 1 つの実装手順について説明します。その他の移行シナリオについては、 AWS 「 規範ガイダンス」ウェブサイトの次のパターンを参照してください。

- でセルフホスト型 MongoDB 環境を MongoDB Atlas に移行する AWS
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

このパターンは、AWS System Integrator (SI) パートナーおよび AWS ユーザーを対象としています。

前提条件と制限

前提条件

- MongoDB Atlas に移行するソースリレーショナルデータベース (Oracle Database、SQL Server、PostgreSQL、MySQL、SAP/Sybase ASE など) 。
- リレーショナルデータベース、MongoDB Atlas、および に精通していること AWS のサービス。このパターンでは、移行ステップの一部を大まかに説明します。今後のバージョンで詳細が追加されます。

製品バージョン

- MongoDB バージョン 5.0 以降

アーキテクチャ

次の図は、リレーショナルデータベース管理システム (RDBMS) データベースから MongoDB Atlas への移行を示しています AWS。

さまざまな使用シナリオをサポートする MongoDB Atlas リファレンスアーキテクチャについては、AWS 「 規範ガイダンス」ウェブサイトの「 での MongoDB Atlas への移行 AWS」を参照してください。

ツール

- MongoDB Atlas は、MongoDB データベースをクラウドにデプロイして管理するためのフルマネージド型 Database as a Service (DBaaS) です。
- MongoDB Relational Migrator は、従来のリレーショナルデータベースから MongoDB へのデータのスムーズな移行を提供します。これにより、変換プロセスを自動化し、リレーショナルデータベースの構造化データモデルを MongoDB が提供する柔軟なドキュメント形式に変換できます。Relational Migrator は、データの整合性と関係性を維持し、移行を簡素化します。組織は、MongoDB が提供するスケーラビリティ、パフォーマンス、汎用性のメリットを活用しながら、既存のデータの知識を維持できます。

ベストプラクティス

で MongoDB を使用するためのベストプラクティスについては AWS、 AWS パートナーネットワークブログの投稿を参照してください。

エピック

発見と評価

タスク	説明	必要なスキル
リレーショナルデータベースのパラメータとサイズを決定します。	Relational Migrator のレコメンデーションと からの情報を使用して、合計インデックススペースdb.stats() のワーキングセットサイズを推定します。データスペースの一部が	アプリ所有者、DBA

タスク	説明	必要なスキル
	頻繁にアクセスされると仮定します。このタスクには約 1 週間かかりそうです。このストーリーと、このエピックの他のストーリーの詳細と例については、関連リソースセクションを参照してください。	
ネットワーク帯域幅要件を見積もります。	ネットワーク帯域幅要件を見積もるには、平均ドキュメントサイズに 1 秒あたりに提供されるドキュメント数を掛けます。基準として、クラスター上のノードが負担する最大トラフィックを考慮に入れます。クラスターからクライアントアプリケーションへのダウンストリームのデータ転送速度を計算するには、一定期間に返されたドキュメントの合計を使用します。アプリケーションがセカンダリノードから読み取る場合は、このドキュメントの合計数を、読み取り操作を実行できるノード数で割ります。データベースの平均ドキュメントサイズを確認するには、 `db.stats().avgObjSize` コマンドを使用します。このタスクには通常 1 日かかります。	DBA
Atlas 層を選択します。	MongoDB ドキュメントの指示に従って、正しい Atlas クラスター層を選択します。	DBA

タスク	説明	必要なスキル
カットオーバーを計画します。	アプリケーションのカットオーバーを計画します。	DBA、アプリ所有者

で新しい MongoDB Atlas 環境を設定する AWS

タスク	説明	必要なスキル
で新しい MongoDB Atlas クラスターを作成します AWS。	MongoDB Atlas で、クラスターの構築を選択します。新しいクラスターの作成ダイアログボックスで、クラウドプロバイダー AWS として を選択します。	DBA
AWS リージョン および グローバルクラスター設定を選択します。	Atlas クラスター AWS リージョン で使用できる のリストから選択します。必要に応じてグローバルクラスタを設定します。	DBA
クラスター階層を選択します。	お好みのクラスター階層を選択します。階層の選択によって、メモリ、ストレージ、IOPS の仕様などの要素が決まります。	DBA
追加のクラスター設定を構成します。	MongoDB のバージョン、バックアップ、暗号化オプションなどのクラスター設定を追加して行います。これらのオプションの詳細については、「関連リソース」セクションを参照してください。	DBA

セキュリティとコンプライアンスを設定

タスク	説明	必要なスキル
アクセスリストを設定します。	Atlas クラスターに接続するには、プロジェクトのアクセスリストにエントリを追加する必要があります。Atlas は TLS/SSL を使用して、データベースの Virtual Private Cloud (VPC) への接続を暗号化します。プロジェクトのアクセスリストを設定し、このエピックのストーリーの詳細については、関連リソースセクションを参照してください。	DBA
ユーザーの認証と認可を行います。	MongoDB Atlas クラスターにアクセスするデータベースユーザーを作成して認証する必要があります。プロジェクト内のクラスターにアクセスするには、ユーザーはそのプロジェクトに所属している必要があり、複数のプロジェクトに属していてもいいです。	DBA
カスタムロールを作成します。	(オプション) Atlas は、組み込みの Atlas データベースユーザーのアクセス許可が、必要なアクセス許可のセットをカバーしていない場合に、カスタムロールの作成をサポートします。	DBA
VPC ピアリングの設定	(オプション) Atlas は、 上の他の VPC との VPC ピア	AWS 管理者

タスク	説明	必要なスキル
	接続をサポートしています AWS。 VPCs	
AWS PrivateLink エンドポイントを設定します。	(オプション) AWS を使用して でプライベートエンドポイントを設定できます AWS PrivateLink。詳細については、Amazon VPC のドキュメント参照してください。	AWS 管理者
2 要素認証を有効にします。	(オプション) Atlasは、ユーザーがAtlasアカウントへのアクセスを制御できるようにする 2 要素認証 (2FA) をサポートしています。	AWS 管理者
LDAP によるユーザー認証と認可を設定します。	(オプション) Atlasは、Lightweight Directory Access Protocol (LDAP) によるユーザー認証および認可の実行をサポートします。	DBA
統合 AWS アクセスを設定します。	(オプション) Atlas Data Lake やカスタマーキー管理を使用した保管時の暗号化など、一部の Atlas 機能は、認証に AWS Identity and Access Management (IAM) ロールを使用します。	AWS 管理者

タスク	説明	必要なスキル
を使用して保管時の暗号化を設定します AWS KMS。	（オプション) Atlas は AWS Key Management Service 、(AWS KMS) を使用してストレージエンジンとクラウドプロバイダーのバックアップを暗号化することをサポートしています。	AWS 管理者
クライアント側のフィールドレベルの暗号化を設定します。	(オプション) Atlas は、フィールドの自動暗号化を含む、クライアント側のフィールドレベルの暗号化をサポートします。	AWS 管理者

データを移行する

タスク	説明	必要なスキル
MongoDB Relational Migrator をアクセスリストに追加します。	Relational Migrator をソースデータベースのアクセスリストに追加します。これにより、ソース環境がターゲットの Atlas クラスターに接続する準備に役立ちます。	DBA
リレーショナルデータベースオブジェクトを評価します。	MongoDB Relational Migrator を起動し、リレーショナルデータベースに接続します。評価を開始します。	DBA
移行パターンを受け入れるか、ビジネスニーズに基づいて変更することを選択します。	初期評価とパフォーマンスパラメータに基づいてリレーショナルマイグレーターが推奨するデータベースパターンを受け入れるか、ビジネス要	DBA

タスク	説明	必要なスキル
	件に基づいて変更することを選択します。	
MongoDB Atlas でターゲットのレプリカセットを起動します。	MongoDB Atlas でターゲットのレプリカセットを起動します。Relational Migrator で、移行する準備ができているを選択します。	DBA

操作統合の設定

タスク	説明	必要なスキル
MongoDB Atlas クラスターに接続します。	MongoDB Atlas クラスター接続が期待どおりに動作することを確認します。	アプリ所有者
クラスターデータと対話します。	クラスターデータを確認します。	DBA
クラスターをモニタリングします。	クラスターが正しく設定されていることを確認します。	DBA
クラスターデータをバックアップし、復元します。	クラスターデータのバックアップを定期的にスケジュールします。	DBA

関連リソース

以下のリンクはすべて、特に明記されていない限り、MongoDB ドキュメントのウェブページに移動します。

移行ガイド

- での MongoDB Atlas への移行 AWS (AWS 規範ガイダンス)

発見と評価

- 「メモリ」
- 「Atlas サンプルデータセットによるサイジングの例」
- 「モバイルアプリケーションのサイジング例」
- 「ネットワークトラフィック」
- 「クラスターの自動スケーリング」
- 「Atlas サイジングテンプレート」

セキュリティとコンプライアンスの設定

- 「IP アクセスリストエントリの設定」
- データベースユーザーを設定する
- Atlas UI へのアクセスを設定する
- カスタムデータベースロールの設定
- データベースユーザーを設定する
- 「ネットワークピアリング接続の設定」
- Atlas のプライベートエンドポイントについて
- 多要素認証オプションを管理する
- 「LDAP によるユーザー認証と認可の設定」
- 「Atlas データレイク」
- 「顧客キー管理による静的暗号化」
- ロールを引き受ける方法 (IAM ドキュメント)
- 「クライアント側のフィールドレベルの暗号化の設定」
- 自動暗号化
- MongoDB Atlas セキュリティコントロール
- 「MongoDB トラストセンター」
- クラスターのセキュリティ機能を設定する

で新しい MongoDB Atlas 環境を設定する AWS

- 「クラウドプロバイダーとリージョン」

- グローバルクラスターの管理
- クラスター階層の選択
- 追加設定を構成する
- 「Atlas の使用開始」
- Atlas UI へのアクセスを設定する
- クラスターの管理

データを移行する

- データの移行またはインポート

クラスターのモニタリング

- クラスターのモニタリング

オペレーションの統合

- 「クラスターへの接続」
- データを操作する
- クラスターのモニタリング
- データのバックアップ、復元、アーカイブ

ブログ記事

- MongoDB ドキュメントモデルを使用した RDBMS スキーマのモダナイズ

でセルフホスト型 MongoDB 環境を MongoDB Atlas に移行する AWS

作成者: Battulga Purevragchaa (AWS)、Babu Srinivasan (MongoDB)、Igor Alekseev (AWS)

概要

このパターンでは、セルフマネージド MongoDB 環境 (MongoDB コミュニティサーバー、エンタープライズサーバー、エンタープライズアドバンスト、mLab、または任意のマネージド MongoDB クラスターを含む) から の MongoDB Atlas に移行する手順について説明します AWS クラウド。「Atlas ライブマイグレーションサービス」を使用して、MongoDB から MongoDB Atlas へのデータ移行を加速します。

このパターンは、 AWS 規範的ガイダンスウェブサイトの「 での MongoDB Atlas への移行 AWS」ガイドに付属しています。そのガイドで説明されている移行シナリオの 1 つの実装手順について説明します。その他の移行シナリオについては、 AWS 「 規範ガイダンス」ウェブサイトの次のパターンを参照してください。

- でリレーショナルデータベースを MongoDB Atlas に移行する AWS
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

このパターンは、AWS Systems Integrator (SI) パートナーと AWS ユーザーを対象としています。

前提条件と制限

前提条件

- MongoDB アトラスに移行するソース MongoDB Enterprise Advanced、Community Server、またはその他のセルフマネージド MongoDB 環境。
- MongoDB、MongoDB Atlas、および に精通していること AWS のサービス。このパターンでは、移行ステップの一部を大まかに説明します。今後のバージョンで詳細が追加されます。

製品バージョン

- MongoDB バージョン 6.0.13 以降

アーキテクチャ

次の図は、MongoDB Enterprise Advanced データベースと MongoDB Community データベースから MongoDB MongoDB Atlas on にデータを移行するために使用される Atlas Live Migration Service を示しています AWS。このサービスは、ダウンタイムを最小限に抑え、継続的なデータ同期で、大規模で複雑なデータベースを MongoDB Atlas に移行する必要がある場合に使用します。このパターンでは、Atlas Live Migration Service を使用します。

次の図は、保護されたAWS PrivateLink接続 AWS を介して MongoDB Enterprise Advanced データベースと MongoDB MongoDB Community データベースから 上の MongoDB Atlas にデータを移行するためにも使用できる MongoDB ミラーサービス (mongomirror) を示しています。オンプレミスの MongoDB と MongoDB Atlas 間の継続的なデータレプリケーションmongomirrorに使用します。このツールはディザスタリカバリや段階的な移行に最適ですが、このパターンの範囲外です。

さまざまな使用シナリオをサポートするその他の MongoDB Atlas リファレンスアーキテクチャについては、 AWS 「 規範ガイダンス」ウェブサイトの「 での MongoDB Atlas への移行 AWS」を参照してください。

ツール

- MongoDB Atlas は、MongoDB データベースをクラウドにデプロイして管理するためのフルマネージド型 Database as a Service (DbaaS) です。
- Atlas Live Migration Service は、データベースを Atlas に移行するのに役立つ無料の MongoDB ユーティリティです。このサービスは、カットオーバーまで、ソースデータベースを移行先データベースと同期させます。カットオーバーの準備ができたら、アプリケーションインスタンスを停止し、デスティネーションの Atlas クラスターを指定して再起動します。このサービスにアクセスするには、MongoDB Atlas クラスターからデータベースオプションを選択します。
- mongomirror は、既存の MongoDB レプリカセットから MongoDB Atlas レプリカセットにデータを手動で移行するためのツールです。 mongomirror では、既存のレプリカセットまたはアプリケーションをシャットダウンしたり、ユーザーまたはロールデータをインポートしたり、設定データベースをコピーしたりする必要はありません。MongoDB ドキュメントmongomirrorからダウンロードできます。

ベストプラクティス

で MongoDB を使用するためのベストプラクティスについては AWS、 AWS パートナーネットワークブログの投稿を参照してください。

エピック

発見と評価

タスク	説明	必要なスキル
クラスターサイズを決定します。	合計インデックススペースの からの情報を使用して、ワーキングセットのサイズdb.stats() を推定します。データスペースの一部が頻繁にアクセスされると仮定します。または、独自の仮定に基づいてメモリ要件を見積もることもできます。このタスクには約 1 週間かかりそうです。この記事の詳細と例、およびこのエピックの他のストーリーについては、関連リソースセクションを参照してください。	DBA、アプリ所有者
ネットワーク帯域幅要件を見積もります。	ネットワーク帯域幅要件を見積もるには、平均ドキュメントサイズに 1 秒あたりに提供されるドキュメント数を掛けます。基準として、クラスター上のノードが負担する最大トラフィックを考慮に入れます。クラスターからクライアントアプリケーションへのダウンストリームのデータ転送速度を計算するには、一定	DBA

タスク	説明	必要なスキル
	期間に返されたドキュメントの合計を使用します。アプリケーションがセカンダリノードから読み取る場合は、このドキュメントの合計数を、読み取り操作を実行できるノード数で割ります。データベースの平均ドキュメントサイズを確認するには、 `db.stats().avgObjSize` コマンドを使用します。このタスクには通常 1 日かかります。	
Atlas 層を選択します。	MongoDB ドキュメントの指示に従って、正しい Atlas クラスター層を選択します。	DBA
カットオーバーを計画します。	アプリケーションのカットオーバーを計画します。	DBA、アプリ所有者

AWS に新しい MongoDB Atlas 環境を設定します

タスク	説明	必要なスキル
で新しい MongoDB Atlas クラスターを作成します AWS。	Atlas にログインし、プロジェクトの概要ページを開きます。クラスターを作成するには、作成ボタンを選択します。詳細については、MongoDB のドキュメントをご参照ください。	DBA
AWS リージョン および グローバルクラスター設定を選択します。	Atlas クラスター AWS リージョン で使用できる のリストから選択します。必要に応	DBA

タスク	説明	必要なスキル
	じてグローバルクラスタを設定します。詳細については、MongoDB のドキュメントをご参照ください。	
クラスター階層を選択します。	お好みのクラスター階層を選択します。階層の選択によって、メモリ、ストレージ、IOPS の仕様などの要素が決まります。	DBA
追加のクラスター設定を構成します。	MongoDB のバージョン、バックアップ、暗号化オプションなどのクラスター設定を追加して行います。これらのオプションの詳細については、「関連リソース」セクションを参照してください。	DBA

セキュリティとコンプライアンスを設定

タスク	説明	必要なスキル
ユーザーの認証と認可を行います。	MongoDB Atlas クラスターにアクセスするデータベースユーザーを作成して認証する必要があります。プロジェクト内のクラスターにアクセスするには、ユーザーはそのプロジェクトに属している必要があり、複数のプロジェクトに属している必要があります。Atlas は AWS Identity and Access Management (IAM) に基づく認証もサポートして	DBA

タスク	説明	必要なスキル
	います。詳細については、MongoDB のドキュメントをご参照ください。	
カスタムロールを作成します。	(オプション) Atlas は、組み込みの Atlas データベースユーザーのアクセス許可が、必要なアクセス許可のセットをカバーしていない場合に、カスタムロールの作成をサポートします。	DBA
VPC ピアリングの設定	(オプション) Atlas は、 上の他の VPCs との仮想プライベートクラウド (VPC) ピア接続をサポートしています AWS。	AWS 管理者
AWS PrivateLink エンドポイントを設定します。	(オプション) AWS を使用して でプライベートエンドポイントを設定できます AWS PrivateLink。詳細については、Amazon VPC のドキュメント参照してください。	AWS 管理者
2 要素認証を有効にします。	(オプション) Atlasは、ユーザーがAtlasアカウントへのアクセスを制御できるようにする 2 要素認証 (2FA) をサポートしています。	AWS 管理者
LDAP によるユーザー認証と認可を設定します。	(オプション) Atlasは、Lightweight Directory Access Protocol (LDAP) によるユーザー認証および認可の実行をサポートします。	AWS 管理者

タスク	説明	必要なスキル
統合 AWS アクセスを設定します。	（オプション) Atlas Data Lake やカスタマーキー管理を使用した保管時の暗号化など、一部の Atlas 機能は、認証に IAM ロールを使用します。	AWS 管理者
を使用して保管時の暗号化を設定します AWS KMS。	（オプション) Atlas は AWS Key Management Service 、(AWS KMS) を使用してストレージエンジンとクラウドプロバイダーのバックアップを暗号化することをサポートしています。	AWS 管理者
クライアント側のフィールドレベルの暗号化を設定します。	(オプション) Atlas は、フィールドの自動暗号化を含む、クライアント側のフィールドレベルの暗号化をサポートします。	AWS 管理者

データを移行する

タスク	説明	必要なスキル
MongoDB Atlas でターゲットレプリカセットを選択します。	送信先の Atlas クラスターに移動し、省略記号 (...) ボタンを選択します。クラスターリストで、このボタンがクラスター名の下に表示されます。クラスターの詳細では、ボタンが接続ボタンと設定ボタンの横の右側に表示されます。詳細については、 MongoDB	DBA

タスク	説明	必要なスキル
	のドキュメントをご参照ください。	
Atlas Live Migration Service をアクセスリストに追加します。	Atlas Live Migration Service を AWS ソースクラスターのアクセスリストに追加します。これにより、ソース環境がターゲットの Atlas クラスターに接続する準備に役立ちます。	DBA
Atlas Live Migration Service を使用して移行を実行します。	移行の開始 を選択します。カットオーバーの準備ボタンが緑色に変わったら、カットオーバーを実行します。Atlas クラスタのパフォーマンスメトリクスを確認してください。新しいデータベースを指すように、すべてのアプリケーションレイヤーのデータベース接続を更新することを検討してください。	DBA

操作統合の設定

タスク	説明	必要なスキル
MongoDB Atlas クラスターに接続します。	MongoDB Atlas クラスター接続が期待どおりに動作することを確認します。	アプリ所有者
クラスターデータと対話します。	クラスターデータをテストします。	DBA
クラスターをモニタリングします。	クラスターが正しく設定されていることを確認します。	DBA

タスク	説明	必要なスキル
クラスターデータをバックアップし、復元します。	クラスターデータのバックアップを定期的にスケジュールします。	DBA

トラブルシューティング

問題	ソリューション

エラー: 指定されたソースに到達できませんでした	• ソースクラスターの IP アクセスリストに正しいサブネット範囲を追加したことを確認してください。4 つの必要なサブネット範囲は、ライブ移行モーダルウィンドウで確認できます。
	• 指定したホスト名がパブリック IP アドレスに解決されていることを確認します。コマンドプロンプトで、次のいずれかのコマンドを使用します。
	`nslookup <hostname>`
	`ping <hostname>`
	• プルライブ移行と互換性のない VPC ピアリング接続を使用しないようにしてください。VPC ピアリング接続が唯一のオプションである場合は、`mongomirror` 代わりにを使用します。

エラー: ホスト名を解決できませんでした	指定されたホスト名の IP アドレスが見つかりませんでした。指定されたホスト名が正しく、パブリックにアクセス可能であることを確認します。
その他のエラー	その他のエラーが発生した場合は、MongoDB ドキュメントの「ライブ移行 (プル) のトラブルシューティング」を参照してください。

関連リソース

特に明記されていない限り、以下のリンクはすべて MongoDB ドキュメントのウェブページに移動します。

移行ガイド

- での MongoDB Atlas への移行 AWS (AWS 規範ガイダンス)

レガシー移行

- MongoDB の古いバージョンの移行

発見と評価

- 「メモリ」
- 「Atlas サンプルデータセットによるサイジングの例」
- 「モバイルアプリケーションのサイジング例」
- 「ネットワークトラフィック」
- 「クラスターの自動スケーリング」
- 「Atlas サイジングテンプレート」

セキュリティとコンプライアンスの設定

- 「IP アクセスリストエントリの設定」
- データベースユーザーを設定する
- Atlas UI へのアクセスを設定する
- カスタムデータベースロールの設定
- データベースユーザーを設定する
- 「ネットワークピアリング接続の設定」
- Atlas のプライベートエンドポイントについて
- 多要素認証オプションを管理する
- 「LDAP によるユーザー認証と認可の設定」
- 「Atlas データレイク」
- 「顧客キー管理による静的暗号化」

- ロールを引き受ける方法 (IAM ドキュメント)
- 「クライアント側のフィールドレベルの暗号化の設定」
- 自動暗号化
- MongoDB Atlas セキュリティコントロール
- 「MongoDB トラストセンター」
- クラスターのセキュリティ機能を設定する

で新しい MongoDB Atlas 環境を設定する AWS

- 「クラウドプロバイダーとリージョン」
- グローバルクラスターを管理する
- クラスター階層の選択
- 追加設定を構成する
- 「Atlas の使用開始」
- Atlas UI へのアクセスを設定する
- クラスターの管理

データを移行する

- データの移行またはインポート

クラスターのモニタリング

- クラスターのモニタリング

オペレーションの統合

- 「クラスターへの接続」
- データを操作する
- クラスターのモニタリング
- データのバックアップ、復元、アーカイブ

トレーニング

- MongoDB Atlas を使用したライブ移行

追加情報

詳細については、MongoDB ドキュメントの以下のトピックを参照してください。

- サーバーレスインスタンスにデータを移動するには、Compass を使用してデータをエクスポートおよびインポートするか、セルフマネージドツールでデータを移行します。詳細については、「サーバーレスインスタンスの制限」を参照してください。
- Atlas の新しいクラスターにデータをロードするには、「Atlas にデータをロードする」を参照してください。
- テスト目的でクラスターのコピーを作成するには、「自己管理型デプロイのバックアップ方法」を参照してください。
- 移行するアプリケーションにほぼ継続的な稼働時間が必要な場合は、MongoDB サポートに連絡して、稼働時間の要件とクラスター設定を共有してください。
- 詳細については、「データの移行またはインポート」を参照してください。

Oracle Data Pump と AWS DMS を使用して Oracle JD Edwards EnterpriseOne データベースを AWS に移行します

作成者: Thanigaivel Thirumalai (AWS)

概要

JD Edwards EnterpriseOne データベースを 「Amazon Relational Database Service (Amazon RDS」) に移行して実行できます。データベースを Amazon RDS に移行すると、AWS がバックアップタスクと高可用性セットアップを処理するので、お客様は EnterpriseOne アプリケーションとその機能のメンテナンスに集中できます。移行プロセス中に考慮すべき主な要因の包括的なリストについては、AWS 規範ガイダンスの「Oracle データベースの移行戦略」を参照してください。

EnterpriseOne データベースを移行するには、次のようないくつかの方法があります。

- スキーマとテーブルの作成には Oracle ユニバーサルBatch エンジン (UBE) R98403 を使用し、移行には AWS Database Migration Service (AWS DMS) を使用
- スキーマとテーブルの作成には DB ネイティブツールを使用し、移行には AWS DMS を使用する
- 既存データの移行 (全ロード) には DB ネイティブツールを使用し、変更データキャプチャ (CDC) タスクには AWS DMS を使用する

このパターンは 3 番目のオプションを対象としています。「AWS DMS」 とその CDC 特徴量を備えた Oracle Data Pump を使用して、オンプレミスの EnterpriseOne データベースを Oracle 向けの Amazon RDS for Oracle に移行する方法について説明します。

「Oracle JD Edwards EnterpriseOne」は、製品や物理的資産の製造、構築、流通、サービス、管理を行う組織向けのエンタープライズリソースプランニング (ERP) ソリューションです。JD Edwards EnterpriseOne は、さまざまなハードウェア、オペレーティングシステム、データベースプラットフォームをサポートしています。

JD Edwards EnterpriseOne などの重要な ERP アプリケーションを移行する際には、ダウンタイムを最小限に抑えることが重要です。AWS DMS では、ソースデータベースからターゲットデータベースへのフルロードと連続レプリケーションの両方をサポートして、ダウンタイムを最小限に抑えます。AWS DMS では、移行のリアルタイムモニタリングと口グ記録も提供されるため、ダウンタイムの原因となる可能性のある問題を特定して解決するのに役立ちます。

AWS DMS で変更を複製する場合、データベースログから変更を読み取る開始点として、時刻またはシステム変更番号 (SCN) を指定する必要があります。AWS DMS がこれらの変更に確実にアクセ

スできるようにするには、指定された期間 (15 日を推奨) サーバー上でこれらの口グにアクセスできるようにしておくことが重要です。

前提条件と制限

前提条件

- お客様の AWS クラウド環境にターゲットデータベースとしてプロビジョニングされた Amazon RDS for Oracle データベース。手順については、「Amazon ECR ドキュメント」を参照してください。
- オンプレミスで、または AWS の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されている EnterpriseOne データベース。

> **Note**
>
> このパターンは、オンプレミスから AWS に移行するように設計されていますが、EC2 インスタンスで EnterpriseOne データベースを使用してテストされています。オンプレミス環境から移行する予定の場合は、適切なネットワーク接続を設定する必要があります。

- スキーマの詳細です。EnterpriseOne に移行する予定の Oracle データベーススキーマ (DV920 など) を特定します。移行プロセスを開始する前に、スキーマに関する以下の詳細情報を収集します。
  - スキーマサイズ
  - オブジェクトタイプごとのオブジェクトの数
  - 無効なオブジェクトの数

機能制限

- ターゲット Amazon RDS for Oracle データベースで必要なスキーマを作成する必要があります。AWS DMS はそれらのスキーマを自動的に作成しません。(「エピック」 セクションでは、Data Pump を使用してスキーマをエクスポートおよびインポートする方法について説明しています)。ターゲットの Oracle データベースに対して、スキーマ名がすでに存在している必要があります。ソーススキーマからのテーブルがユーザーまたはスキーマにインポートされ、 AWS DMS が管理者、またはシステムアカウントを使用して、ターゲットインスタンスに接続します。複数のスキーマを移行するには、複数のレプリケーションタスクを作成します。また、データをターゲットインスタンス上の異なるスキーマに移行することもできます。これを行うには、AWS DMS テーブルマッピングのスキーマ変換ルールを使用します。

- このパターンはデモデータセットでテストされています。データセットとカスタマイズの互換性を検証することをお勧めします。
- このパターンでは、Microsoft Windows 上で実行されている EnterpriseOne データベースを使用します。ただし、AWS DMS でサポートされている他のオペレーティングシステムでも同じプロセスを使用できます。

アーキテクチャ

次の図は、Oracle データベースで EnterpriseOne をソースデータベースとして、Amazon RDS for Oracle データベースをターゲットデータベースとして実行しているシステムを示しています。データはソース Oracle データベースからエクスポートされ、Oracle Data Pump を使用してターゲット Amazon RDS for Oracle データベースにインポートされ、AWS DMS を使用して CDC 更新用に複製されます。

1. Oracle Data Pump はソースデータベースからデータを抽出し、そのデータは Amazon RDS for Oracle データベースターゲットに送信されます。
2. CDC データは、ソースデータベースから AWS DMS のソースエンドポイントに送信されます。
3. ソースエンドポイントから AWS DMS レプリケーションインスタンスにデータが送信され、そこでレプリケーションタスクが実行されます。
4. レプリケーションタスクが完了すると、データは AWS DMS のターゲットエンドポイントに送信されます。
5. ターゲットエンドポイントから、データは Amazon RDS for Oracle データベースインスタンスに送信されます。

ツール

AWS サービス

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「OracleのAmazon Relational Database Service (Amazon RDS)」 によって、AWS クラウドで Oracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。

その他のサービス

- 「Oracle Data Pump」 を使用すると、データやメタデータをあるデータベースから別のデータベースに高速に移動できます。

ベストプラクティス

LOB への移行

ソースデータベースに、ターゲットデータベースに移行する必要のあるラージバイナリオブジェクト (LOB) が含まれている場合、AWS DMS には次のオプションがあります。

- Full LOB mode - サイズにかかわらず、 AWS DMS はすべての LOB をソースからターゲットに移行します。移行は他のモードよりも遅くなりますが、データが切り捨てられないという利点があります。パフォーマンスを向上させるには、新しいレプリケーションインスタンスに別のタスクを作成して、LOB が数メガバイトを超えるテーブルを移行できます。
- 制限付き LOB モード — LOB 列データの最大サイズを指定します。これにより、AWS DMS はリソースを事前に割り当て、LOB を一括適用できます。LOB 列のサイズがタスクで指定されたサイズを超える場合、AWS DMS はデータを切り捨てて AWS DMS ログファイルに警告を送信します。LOB データサイズが制限された LOB サイズ内にある場合、制限付き LOB モードを使用することでパフォーマンスを向上させることができます。
- インライン LOB モード — 小さい LOB と大きな LOB の両方を複製することで、データを切り捨てたり、タスクのパフォーマンスを低下させたりすることなく LOB を移行できます。まず、`InlineLobMaxSize` パラメータの値を指定します。この値は、フル LOB モードがに設定されている場合に `true` のみ使用できます。AWS DMS タスクでは、小さな LOB をインラインで転送するため、効率が向上します。AWS DMS では、ソーステーブルからルックアップを実行して、大きな LOB を移行します。ただし、インライン LOB モードは全ロードフェーズでのみ機能します。

シーケンス値の生成

AWS DMS CDC プロセス中、インクリメンタルシーケンス番号はソースデータベースから複製されません。シーケンス値の不一致を避けるには、すべてのシーケンスのソースから最新のシーケンス値を生成し、それをターゲット Amazon RDS for Oracle データベースに適用する必要があります。

AWS Secrets Manager

認証情報を管理しやすくするために、ブログ記事「AWS Secrets Manager を使用して AWS DMS エンドポイントの認証情報を管理する」の手順に従うことをお勧めします。

パフォーマンス

- レプリケーションインスタンス – 最適なインスタンスサイズを選択するためのガイダンスについては、AWS DMS ドキュメントの「レプリケーションインスタンスに最適なサイズの選択」を参照してください。
- 接続オプション – レイテンシーの問題を避けるため、適切な接続オプションを選択することをお勧めします。AWS Direct Connect は、企業のデータセンターと AWS 間の専用接続であるため、AWS リソースへの最短経路を提供します。転送中に、ネットワークトラフィックは AWS グローバルネットワーク上に残り、インターネットを経由することはありません。これにより、VPN やパブリックインターネットを使用する場合と比較して、ボトルネックにぶつかったり、レイテンシーが予期せず増加したりする可能性が低くなります。
- ネットワーク帯域幅 – パフォーマンスを最適化するには、ネットワークのスループットが速いことを確認してください。オンプレミスのソースデータベースと AWS DMS の間で VPN トンネルを使用している場合は、帯域幅がワークロードに十分であることを確認してください。
- タスク並列処理 – 全ロード中に複数のテーブルをparallel ロードすることで、データ複製を高速化できます。このパターンでは RDBMS エンドポイントを使用するため、このオプションは全ロードプロセスにのみ適用されます。タスクの並列処理は、parallel `MaxFullLoadSubTasks` に実行される全負荷サブタスクの数を決定するパラメーターによって制御されます。デフォルトでは、このパラメーターは 8 に設定されています。つまり、フルモードでは 8 つのテーブル (テーブルマッピングで選択した場合) がまとめてロードされます。このパラメーターは、タスクの JSON スクリプトの全ロードタスク設定セクションで調整できます。
- テーブル並列処理 – AWS DMS では、複数のparallel スレッドを使用して 1 つの大きなテーブルをロードすることもできます。これは、何十億ものレコードがあり、複数のパーティションやサブパーティションがある Oracle ソーステーブルに特に便利です。ソーステーブルがパーティション化されていない場合は、列の境界を使用してparallel ロードできます。
- 負荷の分割 – 負荷を複数のタスクまたは AWS DMS インスタンスに分割する場合、変更をキャプチャするときはトランザクションの境界を覚えておいてください。

エピック

Oracle Data Pump を使用してエンタープライズワンのスキーマをエクスポートします

タスク	説明	必要なスキル
SCN を生成します。	ソースデータベースがアクティブで EnterpriseOne アプリケーションで使用されている場合は、Oracle Data Pump	DBA

タスク	説明	必要なスキル

	を使用してデータエクスポートを開始します。まず、Oracle Data Pump によるエクスポート時のデータ整合性を保つため、また AWS DMS の CDC の開始点として、ソースデータベースからシステム変更番号 (SCN) を生成する必要があります。
	
	ソース・データベースから現在のSCNを生成するには、次のSQL文を使用します。
	
	`SQL> select current_scn from v$database;`
	
	`CURRENT_SCN`
	`-----------`
	`30009727`
	
	生成された SCN を保存します。SCN は、データをエクスポートし、AWS DMS レプリケーションタスクを作成する場合に使用します。	

タスク	説明	必要なスキル

パラメータファイルを作成します。	スキーマをエクスポートするためのパラメータファイルを作成するには、次のコードを使用できます。
	
	`directory=DMS_DATA_PUMP_DIR logfile=export_dms.log dumpfile=export_dms_data.dmp schemas=<schema name> flashback_scn=<SCN from previous command>`
	
	Note
	また、要件に基づいて、次のコマンドDATA_PUMP_DIR を使用して独自の を定義することもできます。
	
	`SQL> CREATE OR REPLACE DIRECTORY DMS_DATA_PUMP_DIR AS '<Directory for dump>'; Directory created.`
	
	`SQL> GRANT READ, WRITE ON DIRECTORY DMS_DATA_PUMP_DIR TO SYSTEM; Grant succeeded.`
		DBA

タスク	説明	必要なスキル
スキーマをエクスポートします。	エクスポートを実行するには、expdp 以下のユーティリティを使用します。 C:\Users\Administrator>expdp system/********@<DB Name> PARFILE='<Path to PAR file create above>' Export: Release 19.0.0.0.0 - Production on *** *** ** **:**:** **** Version 19.3.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Connected to: Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 - Production Starting "SYSTEM"."SYS_EXPORT_SCHEMA_02": system/**********@<DB Name>PARFILE='E:\exp_dms_datapump.par' Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS Processing object type SCHEMA_EXPORT/TABL	DBA

タスク	説明	必要なスキル
	E/STATISTICS/TABLE _STATISTICS Processing object type SCHEMA_EXPORT/STAT ISTICS/MARKER Processing object type SCHEMA_EXPORT/USER Processing object type SCHEMA_EXPORT/ROLE _GRANT Processing object type SCHEMA_EXPORT/DEFA ULT_ROLE Processing object type SCHEMA_EXPORT/TABL ESPACE_QUOTA Processing object type SCHEMA_EXPORT/PRE_ SCHEMA/PROCACT_SCHEMA Processing object type SCHEMA_EXPORT/TABLE/ TABLE Processing object type SCHEMA_EXPORT/TABL E/GRANT/OWNER_GRANT/ OBJECT_GRANT Processing object type SCHEMA_EXPORT/TABLE/ INDEX/INDEX Processing object type SCHEMA_EXPORT/TABLE/ CONSTRAINT/CONSTRAINT . . exported "<Schema Name>"."<Table Name>" 228.9 MB 496397 rows Master table "SYSTEM". "SYS_EXPORT_SCHEMA _02" successfully loaded/unloaded	

タスク	説明	必要なスキル
	``` ********** ********** ********** ********** **** Dump file set for SYSTEM.SYS_EXPORT_ SCHEMA_02 is: E:\DMSDUMP\EXPORT_ DMS_DATA.DMP Job "SYSTEM"."SYS_EXPO RT_SCHEMA_02" successfully completed at *** *** ** **:**:** **** elapsed 0 00:01:57 ```	

Oracle Data Pump を使用してエンタープライズワンのスキーマをインポートします

タスク	説明	必要なスキル
ダンプファイルを、ターゲットインスタンスに転送します。	DBMS_FILE_TRANSFER ユーティリティを使用してファイルを転送するには、ソースデータベースから Amazon RDS for Oracle インスタンスへのデータベースリンクを作成する必要があります。リンクが確立されたら、ユーティリティを使用して Data Pump ファイルを Amazon RDS インスタンスに直接転送できます。 または、Data Pump ファイルを「Amazon Simple Storage Service (Amazon S3)」に転送し、Amazon RDS for Oracle	DBA

タスク	説明	必要なスキル

	インスタンスにインポートできます。このオプションの詳細については、「追加情報」セクションを参照してください。
	
	ターゲットの DB インスタンスの Amazon RDS マスターユーザーに接続する ORARDSDB というデータベースリンクを作成するには、ソースデータベースで次のコマンドを実行します。
	
	`sqlplus / as sysdba`
	
	`SQL*Plus: Release 19.0.0.0.0 on *** *** ** **:**:** **** Version 19.3.0.0.0`
	
	`Copyright (c) 1982, 2019, Oracle. All rights reserved.`
	
	`Connected to: Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 Version 19.3.0.0.0`
	
	`SQL> create database link orardsdb connect to admin identifie d by "******" using '(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = orcl.**** **.us-east-1.rds.a`	

タスク	説明	必要なスキル
	mazonaws.com)(PORT = 1521))(CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = orcl)))'; Database link created. SQL>	
データベースリンクをテストします。	データベースリンクをテストして、sqlplus を使用して Amazon RDS for Oracle のターゲットデータベースに接続できることを確認します。 SQL> select name from v$database@orardsdb; NAME --------- ORCL	DBA

タスク	説明	必要なスキル

ダンプファイルをターゲットデータベースに転送します。	ダンプファイルを Amazon RDS for Oracle データベースにコピーするには、DATA_PUMP_DIR デフォルトのディレクトリを使用するか、次のコードを使用して独自のディレクトリを作成します。このコードはターゲット Amazon RDS インスタンスで実行する必要があります。
	`exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'DMS_TARGET_PUMP_DIR');`
	`PL/SQL procedure successfully completed.`
	次のスクリプトでは、という名前のダンプファイルを、orardsdb という名前のデータベースリンクを使用して、EXPORT_DMS_DATA.DMP ソースのインスタンスからターゲット Amazon RDS for Oracle Database にコピーします。このスクリプトはソースデータベースインスタンスで実行する必要があります。
	`BEGIN DBMS_FILE_TRANSFER.PUT_FILE(`
		DBA

タスク	説明	必要なスキル
	source_directory_ob ject => 'DMS_DATA _PUMP_DIR', source_file_name => 'EXPORT_DMS_DATA.D MP', destination_directory_ object => 'DMS_TARG ET_PUMP_DIR', destination_file_name => 'EXPORT_DMS_DATA.D MP', destination_database => 'orardsdb'); END; PL/SQL procedure successfully completed .	
ターゲットデータベース内のダンプファイルを一覧表示します。	PL/SQL プロシージャが完了したら、次のコードを使用して Amazon RDS for Oracle データベースにデータダンプファイルを一覧表示できます。 select * from table (rdsadmin.rds_file _util.listdir(p_di rectory => 'DMS_TARG ET_PUMP_DIR'));	DBA

タスク	説明	必要なスキル

ターゲットインスタンスに JDE 固有のユーザーを作成します。	ターゲットインスタンスで以下のコマンドを使用して JD Edwards のプロファイルとロールを作成します。
	
	`SQL> CREATE PROFILE "JDEPROFILE" LIMIT IDLE_TIME 15;`
	`Profile created.`
	
	`SQL> CREATE ROLE "JDE_ROLE";`
	`Role created.`
	
	`SQL> CREATE ROLE "JDEADMIN";`
	`CREATE ROLE "JDEUSER";`
	`Role created.`
	`Role created.`
	
	ロールに必要なアクセス許可を保証します。
	
	`SQL> GRANT CREATE ANY SEQUENCE TO JDE_ROLE;`
	`GRANT DROP ANY SEQUENCE TO JDE_ROLE;`
	`GRANT CREATE ANY TRIGGER TO JDE_ROLE;`
	`GRANT DROP ANY TRIGGER TO JDE_ROLE;`
		DBA、JDE、CNC

タスク	説明	必要なスキル

ターゲットインスタンスにテーブルスペースを作成します。	この移行に含まれるスキーマに対して以下のコマンドを使用して、ターゲットインスタンスに必要なテーブルスペースを作成します。
	`SQL> CREATE TABLESPACE <Tablespace Name for Tables>;`
	`Tablespace created.`
	
	`SQL> CREATE TABLESPACE <Tablespace Name for Indexes>;`
	`Tablespace created.`
		DBA、JDE、CNC

タスク	説明	必要なスキル
ターゲットデータベースでインポートを開始します。	インポートプロセスを開始する前に、データダンプファイルを使用して、ターゲット Amazon RDS for Oracle データベースにロール、スキーマ、テーブルスペースを設定します。 インポートを実行するには、Amazon RDS プライマリユーザーアカウントでターゲットデータベースにアクセスし、Amazon RDS for Oracle Database を含む `tnsnames.ora` ファイル内の接続文字列名を使用します `tns-entry`。必要に応じて、データダンプファイルを別のテーブルスペースまたは別のスキーマ名でインポートする再マップオプションを含めることができます。 インポートを開始するには、次のコードを使用します。 `impdp admin@orardsdb directory=DMS_TARGET_PUMP_DIR logfile=import.log dumpfile=EXPORT_DMS_DATA.DMP` インポートを正常に完了させるには、インポートログファイルにエラーがないかどうかを確認し、オブジェクト数、	DBA

タスク	説明	必要なスキル
	行数、無効なオブジェクトなどの詳細を確認します。無効なオブジェクトがある場合は、それらを再コンパイルします。さらに、ソースとターゲットのデータベースオブジェクトを比較して、一致することを確認します。	

ソースとターゲットのエンドポイントで AWS DMS レプリケーションインスタンスをプロビジョニングします

タスク	説明	必要なスキル
テンプレートをダウンロードする。	AWS CloudFormation「DMS_Instance.yaml」テンプレートをダウンロードして、AWS DMS レプリケーションインスタンスとそのソースエンドポイントとターゲットエンドポイントをプロビジョニングします。	クラウド管理者、DBA
スタックの作成を開始します。	1. AWS マネジメントコンソールにサインインし、「https://console.aws.amazon.com/cloudformation」の AWS CloudFormation コンソールを開きます。 2. [スタックの作成] を選択してください。 3. [テンプレートの指定] で、[テンプレートファイル	クラウド管理者、DBA

タスク	説明	必要なスキル
	のアップロード] を選択します。 4. [ファイルを選択]を選択します。 5. `DMS_instance.yaml` ファイルを選択します。 6. [Next (次へ)] を選択します。	

タスク	説明	必要なスキル
パラメータを指定します。	1. スタック名に対して、スタック名を入力します。 2. AWS DMS インスタンスパラメータには、以下のパラメータを入力します。 • DMSInstanceType — ビジネスニーズに基づいて、AWS DMS レプリケーションインスタンスに必要なインスタンスを選択します。 • DMSStorageSize — 移行のサイズに基づいて、AWS DMS インスタンスのストレージサイズを入力します。 3. ソース Oracle データベース設定には、次のパラメータを入力します。 • SourceOracleEndpointID — ソース Oracle データベースサーバー名 • SourceOracleDatabaseName — 該当するソースデータベースのサービス名またはセッション ID (SID) • SourceOracleUserName — ソースデータベースのユーザー名 (デフォルトは `system`) • SourceOracleDBPassword — ソースデータ	クラウド管理者、DBA

タスク	説明	必要なスキル
	ベースのユーザー名のパスワード • SourceOracleDBPort — ソースデータベースのポート 4. Oracle データベース設定のターゲット RDS に対して、次のパラメータを入力します。 • TargetRDSoracleEndpointID — ターゲット RDS データベースエンドポイント • TargetRDSoracleDatabaseName — ターゲット RDS データベース名 • TargetRSoracleUsername — ターゲット RDS ユーザー名 • ターゲット RDS OracleDB パスワード — ターゲット RDS パスワード • ターゲット Oracle DB ポート — ターゲット RDS データベースポート 5. VPC、サブネット、セキュリティグループの設定では、次のパラメータを入力します。 • VPCID — レプリケーションインスタンスの VPC	

タスク	説明	必要なスキル
	• vpcSecurityGroupID — レプリケーションインスタンスの VPC セキュリティグループ • DMSSubnet1 — アベイラビリティーゾーン 1 のサブネット • DMSSubnet2 — アベイラビリティーゾーン 2 のサブネット 6. [Next (次へ)] を選択します。	
スタックを作成します。	1. スタックオプションの設定のページで、タグに対して、いずれのオプションの値を入力します。 2. [Next (次へ)] を選択します。 3. レビューページで詳細を確認し、送信を選択します。 プロビジョニングは約 5 ~ 10 分で完了します。AWS CloudFormation スタックページに CREATE_COMPLETE と表示されたら完了です。	クラウド管理者、DBA

タスク	説明	必要なスキル
エンドポイントをセットアップします。	1. 「https://console.aws.amazon.com/dms/v2/」の AWS DMS コンソールを開きます。 2. リソース管理では、レプリケーションインスタンスを選択し、レプリケーションインスタンスを確認します。 3. リソース管理では、エンドポイントを選択し、エンドポイントを確認します。	クラウド管理者、DBA
接続をテストします。	ソースエンドポイントとターゲットエンドポイントのステータスが Active になったら、接続をテストします。各エンドポイント (ソースとターゲット) でテストを実行を選択し、ステータスが成功と表示されることを確認します。	クラウド管理者、DBA

ライブレプリケーション用の AWS DMS レプリケーションタスクを作成する

タスク	説明	必要なスキル
レプリケーションタスクを作成します。	次のステップを使用して AWS DMS レプリケーションタスクを作成します。 1. 「https://console.aws.amazon.com/dms/v2/」の AWS DMS コンソールを開きます。	クラウド管理者、DBA

タスク	説明	必要なスキル
	2. ナビゲーションペインのデータの移行で、データベース移行タスクを選択します。 3. タスク設定ボックスのタスク識別子に、タスク ID を入力します。 4. レプリケーションインスタンスに対して、作成した DMS レプリケーションインスタンスを選択します。 5. ソースデータベースエンドポイントで、ソースエンドポイントを選択します。 6. ターゲットデータベースエンドポイントでは、ターゲット Amazon RDS for Oracle データベースを選択します。 7. 移行タイプでは、データ変更のみを複製を選択します。補足ロギングを有効にする必要があるというメッセージを受け取った場合は、「トラブルシューティング」セクションの指示に従ってください。 8. タスク設定ボックスで、ログシーケンス番号を指定を選択します。 9. システム変更番号には、ソース Oracle データベースから生成した Oracle デー	

タスク	説明	必要なスキル
	タベース SCN を入力します。 10.検証を有効にするを選択します。 11.CloudWatch ログを有効にするを選択します。 この特徴量を有効にすることで、データと「Amazon CloudWatch Logs」を検証して AWS DMS レプリケーションインスタンスログを確認できます。 12.選択ルールで、以下を入力します。 • スキーマに対して、スキーマの入力を選択します。 • スキーマ名には、JDE スキーマ名 (例:DV920) を入力します。 • テーブル名に対して、 % を入力します。 • アクションには含めるを選択します。 13.[Create task] (タスクの作成) を選択します。 タスクを作成すると、AWS DMS は CDC スタートモードで指定した SCN から Amazon RDS for Oracle データベースインスタンスへの継続的な変	

タスク	説明	必要なスキル
	更を移行します。CloudWatch ログを確認して移行を確認することもできます。	
レプリケーションタスクを繰り返します。	前のステップを繰り返して、移行に含まれる他の JD Edwards スキーマのレプリケーションタスクを作成します。	クラウド管理者、DBA、JDE CNC 管理者

ターゲットの Amazon RDS for Oracle Database でデータベーススキーマを検証

タスク	説明	必要なスキル
データ転送を検証します。	AWS DMS タスクが開始されたら、タスクページのテーブル統計タブをチェックして、データに加えられた変更を確認できます。 進行中のレプリケーションのステータスは、コンソールのデータベース移行タスクページでモニタリングできます。 詳細については、「AWS DMS データの検証」を参照してください。	クラウド管理者、DBA

カットオーバー

タスク	説明	必要なスキル
レプリケーションを開始します。	レプリケーション手順を中止し、ソースアプリケーションサービスを停止します。	クラウド管理者、DBA
JD Edwards アプリケーションを起動します。	ターゲットの JD Edwards プレゼンテーションおよびロジック層アプリケーションを AWS で起動し、Amazon RDS for Oracle データベースに転送します。 アプリケーションにアクセスすると、Amazon RDS for Oracle データベースとのすべての接続が確立されていることに気付くはずです。	DBA、JDE CNC 管理者
ソースデータベースをオフにします。	接続がないことを確認したら、ソースデータベースをオフにできます。	DBA

トラブルシューティング

問題	ソリューション

「継続的なレプリケーションのためにソースデータベースへの追加ロギング」を有効にするよう求める警告メッセージが表示されます。	以下のコマンドを入力して補足ロギングを有効にします。
	
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;`

問題	ソリューション

	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;`

AWS DMS に、サプリメンタルロギングがオフになっています。	AWS DMS では、補足ログ記録はデフォルトでオフになっています。ソース Oracle エンドポイントで有効にするには: 1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/dms/v2/://www.com」で AWS DMS コンソールを開きます。 2. [Endpoints] (エンドポイント) を選択します。 3. サプリメンタル ログを追加する Oracle ソース エンドポイントを選択します。 4. Modify を選択します。 5. [Advanced] (詳細) を選択し、[Extra connection attributes] (追加の接続属性) テキストボックスに以下のコードを追加します。 `addSupplementalLogging=Y` 6. Modify を選択します。

CDB レベルでは補足ロギングは有効になっていません。	1. このコマンドを入力します。
	`SQL> alter session set container = CDB$ROOT;`
	`Session altered.`
	2. ステップを繰り返して、サプリメンタルロギング作成を有効にします。

問題	ソリューション
「テストエンドポイントが失敗しました:アプリケーションステータス:1020912、アプリケーションメッセージ:LogMiner は Oracle PDB 環境ではサポートされていません。エンドポイントの初期化に失敗しました。」というエラーメッセージが表示されます。	このエラーメッセージが表示される場合は、LogMiner の代わりにバイナリリーダーを使用できます。 エンドポイント設定で、ソースデータベースの追加の接続属性に次の行を追加します。 `useLogMinerReader=N;useBfile=Y;`

関連リソース

- AWS Database Migration Service の使用開始
- 「AWS Database Migration Service ベストプラクティス」
- 「AWS クラウドへの Oracle データベースの移行」
- 「AWS CloudFormation の AWS Database Migration Service リソースタイプリファレンス」
- 「AWS DMS エンドポイントの認証情報を AWS Secrets Manager で管理」
- 「AWS データベース移行サービスの移行タスクのトラブルシューティング」
- 「AWS Database Migration Service ベストプラクティス」

追加情報

Amazon S3 を使用してファイルを転送する

Amazon S3 にファイルを転送するには、AWS CLI 、 m たは (CLI)、 Amazon S3 コンソールを使用できます。ファイルを Amazon S3 に転送したら、Amazon RDS for Oracle インスタンスを使用して Amazon S3 からデータポンプファイルをインポートできます。

代替方法として Amazon S3 統合を使用してダンプファイルを転送することを選択した場合は、次の手順を実行します。

1. S3 バケットを作成する。
2. Oracle データパンプを使用して、ソースデータベースからデータをエクスポートします。
3. S3 バケットにデータパンプファイルをアップロードします。

4. S3 バケットから、ターゲットの Amazon RDS for Oracle Database にデータポンプファイルをダウンロードします。
5. Data Pump ファイルを使用してインポートを実行します。

> **Note**
>
> S3 インスタンスと RDS インスタンス間で大きなデータファイルを転送するには、Amazon S3 Transfer Acceleration 機能を使用することをお勧めします。

AWS DMS を使用して Oracle PeopleSoft データベースを AWS に移行する

作成者: sampath kathirvel (AWS)

概要

「Oracle PeopleSoft」は、企業全体のプロセスを対象とするエンタープライズリソースプランニング (ERP) ソリューションです。PeopleSoft のアーキテクチャは、クライアント、アプリケーション、データベースの 3 層構造です。PeopleSoft は 「Amazon Relational Database Service (Amazon RDS)」で実行できます。

Oracle データベースを Amazon RDS に移行すると、Amazon Web Services (AWS) がバックアップタスクと高可用性セットアップを処理するので、お客様は PeopleSoft アプリケーションとその機能のメンテナンスに集中できます。移行プロセス中に考慮すべき主な要因の包括的なリストについては、AWS 規範ガイダンス の「Oracle データベースの移行戦略」を参照してください。

このパターンは、AWS Database Migration Service (AWS DMS) とその変更データキャプチャ (CDC) 特徴量を備えた Oracle Data Pump を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行するためのソリューションを提供します。

Oracle PeopleSoft などの重要な ERP アプリケーションを移行する際には、ダウンタイムを最小限に抑えることが重要です。AWS DMS では、ソースデータベースからターゲットデータベースへのフルロードと連続レプリケーションの両方をサポートすることで、ダウンタイムを最小限に抑えます。AWS DMS では、移行のリアルタイムモニタリングと口グ記録も提供されるため、ダウンタイムの原因となる問題を特定して解決することを支援します。

AWS DMS で変更を複製する場合、AWS DMS がデータベースログから変更を読み取るための開始点として、時刻またはシステム変更番号 (SCN) を指定する必要があります。AWS DMS がこれらの変更にアクセスすることを保証するには、指定された期間でサーバーでこれらのログにアクセスできるようにしておくことが重要です。

前提条件と制限

前提条件

- ターゲットデータベースとして AWS クラウド環境内の Amazon RDS for Oracle データベースをプロビジョニングしました。
- オンプレミスまたは AWS クラウドの Amazon Elastic Compute Cloud (Amazon EC2) 上で稼働している Oracle PeopleSoft データベース。

> **Note**
>
> このパターンは、オンプレミスから AWS に移行するように設計されていますが、Amazon EC2 インスタンスで Oracle Database を使用してテストされています。オンプレミスから移行するには、適切なネットワーク接続を設定する必要があります。

- スキーマの詳細です。Oracle PeopleSoft アプリケーションを Amazon RDS for Oracle に移行する場合、移行する Oracle データベーススキーマ (SYSADM など) を特定する必要があります。移行プロセスを開始する前に、スキーマに関する以下の詳細情報を収集します。
  - サイズ
  - オブジェクトタイプごとのオブジェクトの数
  - 無効なオブジェクトの数。

  この情報は移行プロセスに役立ちます。

制約事項

- このシナリオは PeopleSoft DEMO データベースでのみテストされています。大規模なデータセットではテストされていません。

アーキテクチャ

次の図表では、ソースデータベースとして Oracle データベースを実行するインスタンス、及び ターゲットデータベースとして Amazon RDS for Oracle を実行するインスタンスを示しています。データは、Oracle Data Pump を使用してソース Oracle データベースからターゲット Amazon RDS for Oracle データベースにエクスポートおよびインポートされ、AWS DMS を使用して CDC が変更された場合にレプリケートされます。

1. 最初のステップでは、Oracle Data Pump を使用してソースデータベースからデータを抽出し、そのデータを Amazon RDS for Oracle データベースターゲットに送信します。
2. データは、ソースデータベースから AWS DMS のソースエンドポイントに送信されます。
3. ソースエンドポイントから AWS DMS レプリケーションインスタンスにデータが送信され、そこでレプリケーションタスクが実行されます。

4. レプリケーションタスクが完了する後、データが AWS DMS のターゲットエンドポイントに送信されます。
5. ターゲットエンドポイントから、データは Amazon RDS for Oracle データベースインスタンスに送信されます。

ツール

AWS サービス

- 「AWS Database Migration Service (AWS DMS)」を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- 「OracleのAmazon Relational Database Service (Amazon RDS)」によって、AWS クラウドで Oracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。

その他のサービス

- 「Oracle Data Pump」を使用すると、データやメタデータを一つのデータベースから別のデータベースに高速に移動することを支援します。

ベストプラクティス

LOB への移行

ソースデータベースに、ターゲットデータベースに移行する必要のあるラージバイナリオブジェクト (LOB) が含まれている場合、AWS DMS には次のオプションがあります。

- Full LOB mode - サイズにかかわらず、 AWS DMS はすべての LOB をソースからターゲットに移行します。移行が遅くなりますが、データが切り捨てられないという利点があります。パフォーマンスを向上させるには、新しいレプリケーションインスタンスに別のタスクを作成して、LOB が数メガバイトを超えるテーブルを移行できます。
- 制限付き LOB モード — LOB 列データの最大サイズを指定します。これにより、AWS DMS はリソースを事前に割り当て、LOB を一括適用できます。LOB 列のサイズがタスクで指定されたサイズを超える場合、AWS DMS はデータを切り捨てて AWS DMS ログファイルに警告を送信します。LOB データサイズが制限された LOB サイズ内にある場合、制限付き LOB モードを使用することでパフォーマンスを向上させることができます。
- インライン LOB モード — 小さい LOB と大きな LOB の両方を複製することで、データを切り捨てたり、タスクのパフォーマンスを低下させたりすることなく LOB を移行できます。ま

ず、InlineLobMaxSize パラメーターの値を指定します。このパラメーターは、フル LOB モードが true に設定されている場合にのみ使用できます。AWS DMS タスクでは、小さな LOB をインラインで転送するため、効率が向上します。AWS DMS では、ソーステーブルからルックアップを実行して、大きな LOB を移行します。ただし、インライン LOB モードは全ロードフェーズでのみ機能します。

シーケンス値の生成

AWS DMS による変更データキャプチャプロセスでは、増分シーケンス番号はソースデータベースから複製されないことに注意します。シーケンス値の不一致を避けるには、すべてのシーケンスのソースから最新のシーケンス値を生成し、それをターゲット Amazon RDS for Oracle データベースに適用する必要があります。

認証情報管理

AWS リソースを保護するために、AWS Identity and Access Management (IAM) の「ベストプラクティス」に従うことをお勧めします。

エピック

ソースとターゲットのエンドポイントで AWS DMS レプリケーションインスタンスをプロビジョニングします

タスク	説明	必要なスキル
テンプレートをダウンロードする。	「DMS_Instance.yaml」AWS CloudFormation テンプレートをダウンロードして、AWS DMS レプリケーションインスタンスとそのソースエンドポイントとターゲットエンドポイントをプロビジョニングします。	クラウド管理者、DBA
スタックの作成を開始します。	1. AWS マネジメントコンソールで、CloudFormation を選択します。 2. [スタックの作成] を選択してください。	クラウド管理者、DBA

タスク	説明	必要なスキル
	3. [テンプレートの指定] で、[テンプレートファイルのアップロード] を選択します。 4. [ファイルを選択]を選択します。 5. `DMS_instance.yaml` ファイルを選択します。 6. [Next (次へ)] を選択します。	

タスク	説明	必要なスキル
パラメータを指定します。	1. スタック名に対して、スタック名を入力します。 2. AWS DMS インスタンスパラメータで、次のパラメータを入力します。 • DMSInstanceType — ビジネスニーズに基づいて、AWS DMS レプリケーションインスタンスに必要なインスタンスを選択します。 • DMSStorageSize — 移行のサイズに基づいて、AWS DMS インスタンスのストレージサイズを入力します。 3. ソース Oracle データベース設定には、次のパラメータを入力します。 • SourceOracleEndpointID — ソース Oracle データベースサーバー名 • SourceOracleDatabaseName — 該当するソースデータベースのサービス名またはセッション ID (SID) • SourceOracleUserName — ソースデータベースのユーザー名 (デフォルトはシステム) • SourceOracleDBPassword — ソースデータ	クラウド管理者、DBA

タスク	説明	必要なスキル
	ベースのユーザー名のパスワード • SourceOracleDBPort — ソースデータベースのポート 4. Oracle データベース構成のターゲット RDS で、次のパラメータを入力します。 • TargetRDSoracleEndpointID — ターゲット RDS データベースエンドポイント • TargetRDSoracleDatabaseName — ターゲット RDS データベース名 • TargetRSoracleUsername — ターゲット RDS ユーザー名 • ターゲット RDS OracleDB パスワード — ターゲット RDS パスワード • ターゲット Oracle DB ポート — ターゲット RDS データベースポート 5. VPC、サブネット、セキュリティグループの設定で、次のパラメータを入力します。 • VPCID — レプリケーションインスタンスの VPC	

タスク	説明	必要なスキル
	• vpcSecurityGroupID — レプリケーションインスタンスの VPC セキュリティグループ • DMSSubnet1 — アベイラビリティーゾーン 1 のサブネット • DMSSubnet2 — アベイラビリティーゾーン 2 のサブネット 6. [Next (次へ)] を選択します。	
スタックを作成します。	1. スタックオプションの設定のページで、タグに対して、いずれのオプションの値を入力します。 2. [Next (次へ)] を選択します。 3. レビューページで詳細を確認し、送信を選択します。 プロビジョニングは約 5 ~ 10 分で完了します。AWS CloudFormation スタックページに CREATE_COMPLETE と表示されたら完了です。	クラウド管理者、DBA

タスク	説明	必要なスキル
エンドポイントをセットアップします。	1. AWS マネジメントコンソールから、データベース移行サービスを選択します。 2. リソース管理でレプリケーションインスタンスを選択します。 3. リソース管理でエンドポイントを選択します。	クラウド管理者、DBA
接続をテストします。	ソースエンドポイントとターゲットエンドポイントのステータスにアクティブになったら、接続をテストします。各エンドポイント (ソースとターゲット) でテストを実行を選択し、ステータスが成功と表示されることを確認します。	クラウド管理者、DBA

Oracle データポンプを使用して、オンプレミスの Oracle データベースから PeopleSoft スキーマをエクスポートします。

タスク	説明	必要なスキル
SCN を生成します。	ソースデータベースがアクティブでアプリケーションで使用中になったら、Oracle Data Pump でデータエクスポートを開始します。まず、Oracle Data Pump によるエクスポート時のデータ整合性を保つため、また AWS DMS の変更データの開始点として、ソースデータベースか	DBA

タスク	説明	必要なスキル

	らシステム変更番号 (SCN) を生成する必要があります。
	
	ソースデータベースから現在の SCN を生成するには、次の SQL ステートメントを入力します。
	
	`SQL> select name from v$database;`
	`SQL> select name from v$database;`
	`NAME`
	`---------`
	`PSFTDMO`
	`SQL> SELECT current_scn FROM v$database;`
	`CURRENT_SCN`
	`-----------`
	`23792008`
	
	生成された SCN を保存して、データをエクスポートしたり、AWS DMS レプリケーションタスクを作成したりするときに使用します。	

タスク	説明	必要なスキル

パラメータファイルを作成します。	スキーマをエクスポートするためのパラメータファイルを作成するには、次のコードを使用できます。
	`$ cat exp_datapmp.par`
	`userid=system/*******`
	`directory=DATA_P`
	`UMP_DIR`
	`logfile=export_dms_`
	`sample_user.log`
	`dumpfile=export_dms_`
	`sample_data_%U.dmp`
	`schemas=SYSADM`
	`flashback_scn=237920`
	`08`
	Note
	また、要件に基づいて、次のコマンドDATA_PUMP_DIR を使用して独自の を定義することもできます。
	`SQL> CREATE OR REPLACE`
	` DIRECTORY DATA_PUMP`
	`_DIR AS '/opt/oracle/`
	`product/19c/dbhome_1/`
	`dmsdump/';`
	`Directory created.`
	`SQL> GRANT READ, WRITE`
	` ON DIRECTORY DATA_PUMP`
	`_DIR TO system;`
	`Grant succeeded.`
		DBA

タスク	説明	必要なスキル
	SQL> SQL> SELECT owner, directory_name, directory_path FROM dba_directories WHERE directory_name='DA TA_PUMP_DIR'; OWNER DIRECTORY_NAME DIRECTORY_PATH ---------- ---------- ---------- ---------- ---------- ---------- SYS DATA_PUMP_DIR /opt/ oracle/product/19c/dbh ome_1/dmsdump/	

タスク	説明	必要なスキル

スキーマをエクスポートします。	エクスポートを実行するには、expdp ユーティリティを使用します。
	
	`$ expdp parfile=exp_datapmp.par`
	`.....`
	`Transferring the dump file with DBMS_FILE_TRANSFER to Target:`
	`. . exported "SYSADM"."PS_XML_TEMPLT_LNG" 6.320 KB 0 rows`
	`. . exported "SYSADM"."PS_XML_TEMPLT_LNK" 6.328 KB 0 rows`
	`. . exported "SYSADM"."PS_XML_XLATDEF_LNG" 6.320 KB 0 rows`
	`. . exported "SYSADM"."PS_XML_XLATITM_LNG" 7.171 KB 0 rows`
	`. . exported "SYSADM"."PS_XPQRYRUNCNTL" 7.601 KB 0 rows`
	`. . exported "SYSADM"."PS_XPQRYRUNPARM" 7.210 KB 0 rows`
	`. . exported "SYSADM"."PS_YE_AMOUNTS" 9.351 KB 0 rows`
	`. . exported "SYSADM"."PS_YE_DATA" 16.58 KB 0 rows`
	`. . exported "SYSADM"."PS_YE_EE" 6.75 KB 0 rows`
	`. . exported "SYSADM"."PS_YE_W2CP_AMOUNTS" 9.414 KB 0 rows`
		DBA

タスク	説明	必要なスキル
	. . exported "SYSADM". "PS_YE_W2CP_DATA" 20.94 KB 0 rows . . exported "SYSADM". "PS_YE_W2C_AMOUNTS" 10.27 KB 0 rows . . exported "SYSADM". "PS_YE_W2C_DATA" 20.95 KB 0 rows . . exported "SYSADM". "PS_ZBD_JOBCODE_TBL" 14.60 KB 0 rows . . exported "SYSADM". "PTGRANTTBL" 5.468 KB 0 rows Master table "SYSTEM". "SYS_EXPORT_SCHEMA _01" successfully loaded/unloaded ** Dump file set for SYSTEM.SYS_EXPORT_ SCHEMA_01 is: /opt/oracle/pr oduct/19c/dbhome_1 /dmsdump/export_dm s_sample_data_01.dmp Job "SYSTEM"."SYS_EXPO RT_SCHEMA_01" successfully completed at Mon Dec 19 20:13:57 2022 elapsed 0 00:38:22	

Oracle Data Pump を使用して PeopleSoft スキーマを Amazon RDS for Oracle データベースにインポートします

タスク	説明	必要なスキル

ダンプファイルを、ターゲットインスタンスに転送します。	DBMS_FILE_TRANSFER を使用してファイルを転送するには、ソースデータベースから Amazon RDS for Oracle インスタンスへのデータベースリンクを作成する必要があります。リンクが確立されたら、ユーティリティを使用して Data Pump ファイルを RDS インスタンスに直接転送できます。
	
	または、Data Pump ファイルを「Amazon Simple Storage Service (Amazon S3)」に転送し、Amazon RDS for Oracle インスタンスにインポートできます。このオプションの詳細については、「追加情報」セクションを参照してください。
	
	ターゲットの DB インスタンスの Amazon RDS マスターユーザーに接続する ORARDSDB というデータベースリンクを作成するには、ソースデータベースで次のコマンドを実行します。
	
	`$sqlplus / as sysdba`
	`$ SQL> create database link orardsdb connect`
		DBA

タスク	説明	必要なスキル
	to admin identified by "*****" using '(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = testpsft.*******.us-west-2.rds.amazonaws.com)(PORT = 1521))(CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = orcl)))'; Database link created.	
データベースリンクをテストします。	データベースリンクをテストして、sqlplus を使用して Amazon RDS for Oracle のターゲットデータベースに接続できることを確認します。 SQL> SQL> select name from v$database@orardsdb; NAME --------- ORCL SQL>	DBA

タスク	説明	必要なスキル

ダンプファイルをターゲットデータベースに転送します。	ダンプファイルを Amazon RDS for Oracle データベースにコピーするには、DATA_PUMP_DIR デフォルトのディレクトリを使用するか、次のコードを使用して独自のディレクトリを作成できます。
	`exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'TARGET_PUMP_DIR') ;`
	次のスクリプトでは、ソースのインスタンスから export_dms_sample_data_01.dmp という名前のダンプファイルを、orardsdb という名前のデータベースリンクを使用して、ターゲット Amazon RDS for Oracle Database にコピーします。
	`$ sqlplus / as sysdba SQL> BEGIN DBMS_FILE_TRANSFER .PUT_FILE( source_directory _object => 'DATA_PUMP_DIR', source_file_name => 'export_dms_sample _data_01.dmp',`
		DBA

タスク	説明	必要なスキル
	destination_directory _object => 'TARGET_P UMP_DIR'', destination_file_name => 'export_dms_sample _data_01.dmp', destination_database => 'orardsdb'); END; / PL/SQL procedure successfully completed .	
ターゲットデータベース内のダンプファイルを一覧表示します。	PL/SQL プロシージャが完了したら、次のコードを使用して Amazon RDS for Oracle データベースにデータダンプファイルを一覧表示できます。 SQL> select * from table (rdsadmin.rds_file _util.listdir(p_di rectory => 'TARGET_P UMP_DIR'));	DBA

タスク	説明	必要なスキル
ターゲットデータベースでインポートを開始します。	インポートプロセスを開始する前に、データダンプファイルを使用して、ターゲット Amazon RDS for Oracle データベースにロール、スキーマ、テーブルスペースを設定します。 インポートを実行するには、Amazon RDS マスターユーザーアカウントでターゲットデータベースにアクセスし、Amazon RDS for Oracle Database `tns-entry` を含む `tnsnames.ora` ファイル内の接続文字列名を使用します。必要に応じて、データダンプファイルを別のテーブルスペースまたは別のスキーマ名でインポートする再マップオプションを含めることができます。 インポートを開始するには、次のコードを使用します。 `impdp admin@orardsdb directory=TARGET_PUMP_DIR logfile=import.log dumpfile=export_dms_sample_data_01.dmp` インポートを正常に完了させるには、インポートログファイルにエラーがないかどうか	DBA

タスク	説明	必要なスキル
	を確認し、オブジェクト数、行数、無効なオブジェクトなどの詳細を確認します。無効なオブジェクトがある場合は、それらを再コンパイルします。さらに、ソースとターゲットのデータベースオブジェクトを比較して、一致することを確認します。	

CDC を使用して AWS DMS レプリケーションタスクを作成し、ライブレプリケーションを実行する

タスク	説明	必要なスキル
レプリケーションタスクを作成します。	次のステップを使用して AWS DMS レプリケーションタスクを作成します。 1. AWS DMS コンソールの変換と移行で、データベース移行タスクを選択します。 2. タスク設定では、タスク識別子にタスク拡張子を入力します。 3. レプリケーションインスタンスに対して、作成した DMS レプリケーションインスタンスを選択します。 4. ソースデータベースエンドポイントで、ソースエンドポイントを選択します。 5. ターゲットデータベースエンドポイントでは、ターゲット Amazon RDS for	クラウド管理者、DBA

タスク	説明	必要なスキル
	Oracle データベースを選択します。 6. 移行タイプでは、データ変更のみを複製を選択します。補足ロギングを有効にする必要があるというメッセージを受け取った場合は、追加情報セクションの指示に従ってください。 7. タスク設定で、ログシーケンス番号を指定を選択します。 8. システム変更番号には、ソース Oracle データベースから生成した Oracle データベース SCN を入力します。 9. 検証を有効にするを選択します。 10. CloudWatch ログを有効にするを選択します。 この特徴量を有効にすることで、データと「Amazon CloudWatch Logs」を検証して AWS DMS レプリケーションインスタンスログを確認できます。 11. 選択ルールで、以下を入力します。 • スキーマに対して、スキーマの入力を選択します。	

タスク	説明	必要なスキル
	• スキーマ名に SYSADM を入力します。 • テーブル名に対して、 % を入力します。 • アクションには含めるを選択します。 12変換ルールで、以下を入力します。 • ターゲットに対して、テーブルを選択します。 • スキーマ名に対して、スキーマの入力を選択します。 • スキーマ名に SYSADM を入力します。 • アクションには、名前を変更を選択します。 13[Create task] (タスクの作成) を選択します。 タスクを作成する後、CDC スタートモードで提供した SCN から Amazon RDS for Oracle データベースインスタンスに CDC を移行します。CloudWatch ログを確認して確認することもできます。	

ターゲットの Amazon RDS for Oracle Database でデータベーススキーマを検証

タスク	説明	必要なスキル
データ転送を検証します。	AWS DMS タスクが開始されたら、タスクページのテーブル統計タブをチェックして、データに加えられた変更を確認できます。 進行中のレプリケーションのステータスは、コンソールのデータベース移行タスクページでモニタリングできます。 詳細については、「AWS DMS データの検証」を参照してください。	クラウド管理者、DBA

カットオーバー

タスク	説明	必要なスキル
レプリケーションを開始します。	レプリケーション手順を中止し、ソースアプリケーションサービスを停止します。	クラウド管理者、DBA
PeopleSoft ミドルティアを起動します。	ターゲットの PeopleSoft ミドルティアアプリケーションを AWS で起動し、最近移行した Amazon RDS for Oracle データベースに転送します。 アプリケーションにアクセスすると、すべてのアプリケーション接続が Amazon RDS for Oracle データベースと確立	データベース管理者、PeopleSoft 管理者

タスク	説明	必要なスキル
	されていることに気付くはずです。	
ソースデータベースをオフにします。	ソースデータベースへの接続がなくなったことを確認したら、オフにできます。	DBA

関連リソース

- AWS Database Migration Service の使用開始
- 「AWS Database Migration Service ベストプラクティス」
- 「AWS クラウドへの Oracle データベースの移行」

追加情報

Amazon S3 を使用してファイルを転送する

Amazon S3 にファイルを転送するには、AWS CLI 、 m たは (CLI)、 Amazon S3 コンソールを使用できます。ファイルを Amazon S3 に転送したら、Amazon RDS for Oracle インスタンスを使用して Amazon S3 からデータポンプファイルをインポートできます。

代替方法として Amazon S3 統合を使用してダンプファイルを転送することを選択した場合は、次の手順を実行します。

1. S3 バケットを作成する。
2. Oracle データパンプを使用して、ソースデータベースからデータを エクスポートします。
3. S3 バケットにデータパンプファイルをアップロードします。
4. S3 バケットから、ターゲットの Amazon RDS for Oracle Database にデータポンプファイルをダウンロードします。
5. Data Pump ファイルを使用してインポートを実行します。

> **Note**
>
> S3 インスタンスと RDS インスタンス間で大きなデータファイルを転送するには、Amazon S3 Transfer Acceleration 機能を使用することをお勧めします。

補足ロギングを有効にする

継続的なレプリケーションのソースデータベースで「追加ロギング」 を有効にする警告メッセージが表示された場合、下記のステップを使用します。

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
```

オンプレミス MySQL データベースを Amazon RDS for MySQL に移行する

作成者: Lorenzo Mota (AWS)

概要

このパターンでは、オンプレミス MySQL データベースを Amazon Relational Database Service (Amazon RDS) for MySQL に移行するガイダンスを提供します。このパターンでは、完全なデータベース移行のための AWS Database Migration Service （ AWS DMS) または mysqldump などのネイティブ MySQL ツールの使用について説明します。このパターンは主に DBA とソリューションアーキテクトを対象としています。小規模または大規模プロジェクトで、テスト手順 (少なくとも 1 回のテストサイクルを推奨) として、または最終的な移行手順として使用できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの MySQL ソースデータベース

制限

- データベースサイズ制限: 64 TB

製品バージョン

- MySQL バージョン 5.5、5.6、5.7、8.0 サポートされているバージョンの最新リストについては、AWS ドキュメントの「Amazon RDS での MySQL」を参照してください。を使用している場合は AWS DMS、MySQL 互換データベースを現在サポートされている MySQL バージョンのターゲットとして使用する AWS DMS」も参照してください AWS DMS。 MySQL

アーキテクチャ

ソーステクノロジースタック

- オンプレミス MySQL データベース

ターゲットテクノロジースタック

- MySQL を実行中の Amazon RDS DB インスタンス

ターゲットアーキテクチャ

次の図は、移行後の Amazon RDS for MySQL のターゲット実装を示しています。

AWS データ移行アーキテクチャ

の使用 AWS DMS :

次の図は、 AWS DMS を使用してカットオーバーまで完全および増分の変更を送信する場合のデータ移行アーキテクチャを示しています。オンプレミスから へのネットワーク接続は要件 AWS に応じて異なり、このパターンの対象外です。

ネイティブ MySQL ツールを使用する:

次の図は、ネイティブ MySQL ツールを使用する場合のデータ移行アーキテクチャを示しています。エクスポートダンプファイルは Amazon Simple Storage Service (Amazon S3) にコピーされ、カットオーバー AWS 前に で Amazon RDS for MySQL データベースにインポートされます。オンプレミスから へのネットワーク接続は要件 AWS に応じて異なり、このパターンの対象外です。

注意:

- ダウンタイムの要件とデータベースのサイズに応じて、 AWS DMS または 変更データキャプチャ (CDC) ツールを使用すると、カットオーバー時間を最小限に抑えることができます。 AWS DMS は、新しいターゲットへのカットオーバー時間を最小 (通常は分) に短縮できます。データベースのサイズとネットワークレイテンシーが短い場合は、mysqldump を使用したオフライン戦略で十分です。(おおよその時間を把握するためにテストすることをお勧めします。)
- 通常、 などの CDC 戦略では、オフラインオプションよりも多くのモニタリングと複雑さ AWS DMS が必要です。

ツール

- AWS サービス： AWS Database Migration Service （ AWS DMS ） は、データストアを AWS クラウド d に移行するか、クラウドとオンプレミスのセットアップの組み合わせ間で移行するのに役

立ちます。でサポートされている MySQL ソースデータベースとターゲットデータベースの詳細については AWS DMS、MySQL 互換データベースの への移行 AWS」を参照してください。ソースデータベースが でサポートされていない場合は AWS DMS、別の方法を選択してデータを移行する必要があります。

- ネイティブ MySQL ツール: mysqldump
- サードパーティツール: Percona XtraBackup

エピック

移行を計画する

タスク	説明	必要なスキル
データベースのバージョンを検証します。	ソースとターゲットデータベースのバージョンを検証します。	DBA
ハードウェア要件を特定します。	ターゲットサーバーのハードウェア要件を特定します。	DBA、システム管理者
ストレージ要件を特定します。	ターゲットデータベースのストレージ要件 (ストレージタイプ、容量など) を特定します。	DBA、システム管理者
インスタンスタイプを選択します。	容量、ストレージ機能、ネットワーク機能に基づいてターゲットインスタンスのタイプを選択します。	DBA、システム管理者
ネットワークアクセス要件を特定します。	ソースとターゲットデータベースのネットワークアクセスのセキュリティ要件を特定します。	DBA、システム管理者
サポートされていないオブジェクトを特定します。	サポートされていないオブジェクト (ある場合) を特定し、移行作業を決定します。	DBA

タスク	説明	必要なスキル
依存関係を特定します。	リモートデータベースへの依存関係を特定します。	DBA
アプリケーション移行戦略を決定します。	クライアントアプリケーションの移行戦略を決定します。	DBA、アプリ所有者、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。	ルートテーブル、インターネットゲートウェイ、NAT ゲートウェイ、サブネットを設定します。詳細については、Amazon RDS ドキュメントの VPC および Amazon RDS を参照してください。	システム管理者
セキュリティグループを作成します。	要件に応じて、ポートと CIDR 範囲、または特定の IP を設定します。MySQL のデフォルトポートは 3306 です。詳細については、Amazon RDS ドキュメントのセキュリティグループでアクセスをコントロールするを参照してください。	システム管理者
Amazon RDS for MySQL DB インスタンスを設定して起動します。	手順については、Amazon RDS ドキュメントの Amazon RDS DB インスタンスを作成するを参照してください。サポートされているバージョンを確認します。	システム管理者

データの移行 – オプション 1 (ネイティブツールを使用)

タスク	説明	必要なスキル
ネイティブ MySQL ツールまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。	手順については、mysqldump や Percona XtraBackup (物理的な移行用) などの MySQL ツールのドキュメントを参照してください。 オプションの詳細については、ブログ投稿 MySQL 用の Amazon RDS for MySQL または Amazon Aurora MySQL への移行オプションを参照してください。	DBA

データの移行 – オプション 2 (使用 AWS DMS)

タスク	説明	必要なスキル
でデータを移行します AWS DMS。	手順については、 AWS DMS ドキュメントを参照してください。	DBA

カットオーバー前に予備タスクを実行する

タスク	説明	必要なスキル
オブジェクト数の不一致を修正します。	ソースデータベースと新しいターゲットデータベースからオブジェクト数を収集します。ターゲットデータベースの不一致を修正します。	DBA

タスク	説明	必要なスキル
依存関係を確認します。	他のデータベース間の依存関係 (リンク) が有効で、予想どおりに機能しているかどうかを確認します。	DBA
テストを実行します。	テストサイクルの場合は、クエリテストを実行し、メトリクスを収集し、問題を修正します。	DBA

カットオーバー

タスク	説明	必要なスキル
ターゲットデータベースに切り替えます。	クライアントアプリケーションを新しいインフラストラクチャに切り替えます。	DBA、アプリ所有者、システム管理者
テストサポートを提供します。	機能アプリケーションテストをサポートします。	DBA

プロジェクトを閉じる

タスク	説明	必要なスキル
リソースをシャットダウンします。	移行用に作成した一時 AWS リソースをシャットダウンします。	DBA、システム管理者
プロジェクト文書を検証します。	プロジェクト文書を確認して検証する。	DBA、アプリ所有者、システム管理者
メトリクスを収集します。	移行の所要時間、手動作業と自動作業の割合、コスト削減	DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
	などのメトリクスを収集します。	
プロジェクトを終了します。	プロジェクトを終了し、フィードバックを提供します。	DBA、アプリ所有者、システム管理者
ソースデータベースを運用停止します。	移行タスクとカットオーバータスクがすべて完了したら、オンプレミスデータベースの使用を停止します。	DBA、システム管理者

関連リソース

リファレンス

- リレーショナルデータベースの移行戦略
- AWS DMS website
- AWS DMS ドキュメント
- 「Amazon RDS ドキュメント」
- Amazon RDS の価格設定
- Amazon VPC と Amazon RDS
- 「Amazon RDS マルチ AZ 配置」
- Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミス MySQL データベースを Aurora MySQL に移行する
- Amazon RDS DB インスタンスストレージ

チュートリアル

- の開始方法 AWS DMS
- 「Amazon RDS の開始方法」

オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する

作成者: Henrique Lobao (AWS)、Jonathan Pereira Cruz (AWS)、Vishal Singh (AWS)

概要

このパターンは、オンプレミス Microsoft SQL Server データベースから Amazon Relational Database Service (Amazon RDS) for SQL Server に移行するガイダンスを提供します。移行の 2 つのオプションについて説明します。AWS データ移行サービス (AWS DMS) を使用か、またはデータベースコピーウィザードなどのネイティブ Microsoft SQL Server ツールを使用します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターにあるソース Microsoft SQL Server データベース

制限

- データベースサイズの上限: 16 TB

製品バージョン

- SQL サーバー 2014 ~ 2019、エンタープライズ、スタンダード、ワークグループ、開発者エディション。サポートされているバージョンと機能の最新リストについては、AWS ドキュメントの Amazon RDS 上の Microsoft SQL Server を参照してください。AWS DMS を使用中の場合、AWS DMS でサポートされている SQL Server バージョンについては、AWS DMS のターゲットとして Microsoft SQL Server データベースを 使用するも参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス Microsoft SQL Server データベース

ターゲットテクノロジースタック

- Amazon RDS for SQL Server DB インスタンス

ソースアーキテクチャとターゲットアーキテクチャ

AWS DMS を使用する:

ネイティブ SQL Server ツールの使用:

ツール

- AWS DMS は、数タイプのソースデータベースとターゲットデータベースをサポートしています。詳細については、AWS DMS の段階的手順を参照してください。AWS DMS がソースデータベースをサポートしていない場合は、データを移行する別の方法を選択します。
- Microsoft SQL Serverのネイティブツールには、バックアップと復元、データベースコピーウィザード、データベースのコピーとアタッチが含まれます。

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を識別します。		DBA、システム管理者
ストレージ要件 (ストレージタイプと容量) を識別します。		DBA、システム管理者
容量、ストレージ機能、ネットワーク機能に基づき、適切		DBA、システム管理者

タスク	説明	必要なスキル
なインスタンスタイプを選択します。		
ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、システム管理者
アプリケーション移行戦略を特定します。		DBA、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。		システム管理者
セキュリティグループを作成します。		システム管理者
Amazon RDS DB インスタンスを設定して起動します。		DBA、システム管理者

データ移行 — オプション 1

タスク	説明	必要なスキル
ネイティブ SAP ASE ツールまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。		DBA

データ移行 — オプション 2

タスク	説明	必要なスキル
AWS DMS でデータを移行します。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、アプリ所有者、システム管理者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、システム管理者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者、システム管理者
移行の所要時間、手動タスクと自動タスクの割合、コスト		DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
削減などのメトリクスを収集します。		
プロジェクトを終了し、フィードバックを提供します。		DBA、アプリ所有者、システム管理者

関連リソース

リファレンス

- Amazon Web Services に Microsoft SQL Server をデプロイする
- AWS DMS ウェブサイト
- Amazon RDS の価格設定
- AWS 上のMicrosoft 製品
- AWS 上の Microsoft ライセンシング
- AWS 上の Microsoft SQL Server
- Microsoft SQL Server DB インスタンスでの Windows 認証の使用
- 「Amazon RDS マルチ AZ 配置」

チュートリアルと動画

- AWS DMS の使用開始
- Amazon RDS の使用開始
- DMS (ビデオ)
- Amazon RDS (ビデオ)

Rclone を使用して Microsoft Azure Blob から Amazon S3 にデータを移行する

作成者: Suhas Basavaraj (AWS)、Aidan Keane (AWS)、Corey Lane (AWS)

概要

このパターンでは、Rclone を使用して Microsoft Azure Blob オブジェクトストレージから Amazon Simple Storage Service (Amazon S3) バケットにデータを移行する方法を説明します。このパターンを使用すると、1 回限りの移行や継続的なデータの同期を行うことができます。Rclone は Go で記述されたコマンドラインプログラムです。クラウドプロバイダーのさまざまなストレージテクノロジー間でデータを移動するために使用されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Azure Blob コンテナサービスに保存されているデータ

アーキテクチャ

ソーステクノロジースタック

- Azure Blob ストレージコンテナ

ターゲットテクノロジースタック

- Amazon S3 バケット
- Amazon Elastic Compute Cloud (Amazon EC2) インスタンス

アーキテクチャ

ツール

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- Rclone は、rsync にインスパイアされたオープンソースのコマンドラインプログラムです。多くのクラウドストレージプラットフォームで、ファイルを管理するために使用できます。

ベストプラクティス

Azure から Amazon S3 にデータを移行する場合、不必要なコストや転送速度の低下を避けるため、以下の事項に注意してください。

- AWS インフラストラクチャは、AWS リージョン `us-east-1` (北バージニア)や Azure リージョン `East US` など、Azure ストレージアカウントや Blob コンテナと同じ地域に作成します。
- NAT ゲートウェイは、入口帯域幅と出口帯域幅の両方でデータ転送コストが発生するため、可能な限り使用しないでください。
- Amazon S3 用の VPC ゲートウェイエンドポイント を使用すると、パフォーマンスが向上します。
- Intel x86 インスタンスよりも低コストで高いパフォーマンスを実現するためには、AWS Graviton2 (ARM) プロセッサベースの EC2 インスタンスの使用を検討してください。Rcloneは 何度もクロスコンパイルされており、プリコンパイルされた ARM バイナリを提供します。

エピック

AWS と Azure のクラウドリソースを準備する

タスク	説明	必要なスキル
送信先の S3 バケットを準備します。	適切な AWS リージョンに新しい S3 バケットを作成するか、移行するデータの宛先として既存のバケットを選択します。	AWS 管理者
Amazon EC2 用の IAM インスタンスロールを作成します。	Amazon EC2 用の新しい AWS Identity and Access Management (IAM) ロールを作成します。このロールは、送信先の S3 バケットへの書き込みアクセス権を EC2 インスタンスに与えます。	AWS 管理者
IAM インスタンスロールをポリシーにアタッチする	IAM コンソールまたは AWS コマンドラインインターフェイス (AWS CLI) を使用して、	AWS 管理者

タスク	説明	必要なスキル
	送信先の S3 バケットへの書き込みアクセスを許可するEC2インスタンスロールのインラインポリシーを作成します。ポリシーの例については、「追加情報」セクションを参照してください。	
EC2 インスタンスを起動します。	新しく作成した IAM サービスロールを使用するように設定された Amazon Linux EC2 インスタンスを起動します。このインスタンスは、インターネット経由で Azure パブリック API エンドポイントにアクセスする必要があります。 **Note** AWS Graviton ベースの EC2 インスタンスを使用してコストを削減することを検討してください。Rclone は ARM がコンパイルされたバイナリを提供します。	AWS 管理者

タスク	説明	必要なスキル
Azure AD サービスプリンシパルを作成します。	Azure CLI を使用して、ソース Azure Blob ストレージコンテナへの読み取り専用アクセス権がある Azure Active Directory (Azure AD) サービスプリンシパルを作成します。手順については、「追加情報」セクションを参照してください。これらの認証情報を EC2 インスタンス上の場所 `~/azure-principal.json` に格納します。	クラウド管理者

Rclone のインストールと設定

タスク	説明	必要なスキル
Rclone をダウンロードしてインストールします。	Rclone コマンドラインプログラムをダウンロードしてインストールします。インストール手順については、Rclone インストールのドキュメントを参照してください。	AWS 全般、クラウド管理者
Rclone を設定します。	次の `rclone.conf` サンプルファイルをコピーします。`AZStorageAccount` を Azure Storage アカウント名に置き換え、`us-east-1` をS3バケットのある AWS リージョンに置き換えます。このファイルを EC2 インスタンス上の場所 `~/.config/`	AWS 全般、クラウド管理者

タスク	説明	必要なスキル
	rclone/rclone.conf に保存します。 [AZStorageAccount] type = azureblob account = AZStorage Account service_principal_f ile = azure-pri ncipal.json [s3] type = s3 provider = AWS env_auth = true region = us-east-1	

タスク	説明	必要なスキル

Rclone の設定を確認してください。	Rclone が設定され、権限が正しく機能していることを確認するには、Rclone がプロファイルを解析し、Azure Blob コンテナと S3 バケット内のオブジェクトにアクセスできることを確認します。検証コマンドの例については、以下を参照してください。
	• 設定済みのリモートが設定ファイルに表示されます。これにより、設定ファイルが正しく解析されていることを確認できます。 出力を確認し、`rclone.conf` ファイルと一致することを確認します。
	`rclone listremotes`
	`AZStorageAccount:`
	`s3:`
	• 設定したアカウントの Azure Blob コンテナを一覧表示します。`AZStorageAccount` を `rclone.conf` ファイルで使用したストレージアカウント名に置き換えます。
	`rclone lsd AZStorageAccount:`
	`2020-04-29 08:29:26 docs`
		AWS 全般、クラウド管理者

タスク	説明	必要なスキル

	• Azure Blob コンテナ内のファイルを一覧表示します。このコマンドのドキュメントを、Azure ストレージアカウントの実際の Blob コンテナ名で置き換えます。
	
	`rclone ls AZStorageAccount:docs`
	`824884 administrator-en.a4.pdf`
	
	• AWS アカウント内のバケットを一覧表示します。
	
	`[root@ip-10-0-20-157 ~]# rclone lsd s3:`
	`2022-03-07 01:44:40 amzn-s3-demo-bucket1`
	`2022-03-07 01:45:16 amzn-s3-demo-bucket2`
	`2022-03-07 02:12:07 amzn-s3-demo-bucket3`
	
	• S3 バケット内のファイルを一覧表示します。
	
	`[root@ip-10-0-20-157 ~]# rclone ls s3:amzn-s3-demo-bucket1`
	`template0.yaml`
	`template1.yaml`	

Rclone を使用してデータを移行します。

タスク	説明	必要なスキル

コンテナからデータを移行します。	Rclone コピーまたは同期コマンドを実行します。
	
	例: コピー
	
	このコマンドは、コピー元の Azure Blob コンテナから、送信先 S3 バケットにデータをコピーします。
	
	`rclone copy AZStorageAccount:blob-container s3:amzn-s3-demo-bucket1`
	
	例: 同期
	
	このコマンドは、ソースの Azure Blob コンテナと移行先の S3 バケット間でデータを同期します。
	
	`rclone sync AZStorageAccount:blob-container s3:amzn-s3-demo-bucket1`
	
	Important
	同期コマンドを使用すると、ソースコンテナに存在しないデータは送信先 S3 バケットから削除されます。
		AWS 全般、クラウド管理者

タスク	説明	必要なスキル
コンテナを同期します	最初のコピーが完了したら、Rclone 同期コマンドを実行して継続的な移行を実行し、移行先の S3 バケットにない新規ファイルのみをコピーします。	AWS 全般、クラウド管理者
データが正常に移行されたことを確認します。	データが移行先の S3 バケットに正常にコピーされたことを確認するには、Rclone lsd コマンドと ls コマンドを実行します。	AWS 全般、クラウド管理者

関連リソース

- 「Amazon S3 ユーザーガイド」 (AWS ドキュメント)
- 「Amazon EC2 の IAM ロール」 (Amazon EC2 ドキュメント)
- 「Microsoft Azure Blob コンテナの作成」 (Microsoft Azure ドキュメント)
- 「Rclone コマンド」 (Rclone ドキュメント)

追加情報

EC2 インスタンスのロールポリシーの例

このポリシーは、アカウント内の特定のバケットへの読み取り/書き込みアクセスを EC2 インスタンスに提供します。バケットでサーバー側の暗号化にカスタマーマネージドキーを使用する場合、ポリシーには AWS Key Management Service (AWS KMS) への追加アクセスが必要となる場合があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
```

```
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
              "arn:aws:s3:::amzn-s3-demo-bucket/*",
              "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

読み取り専用の Azure AD サービスプリンシパルの作成

Azure サービスプリンシパルは、顧客のアプリケーション、サービス、自動化ツールが特定の Azure リソースにアクセスするためのセキュリティ ID です。これは、特定のロールを持ち、リソースへのアクセスが厳密に制御されているユーザー ID (ログイン名とパスワードまたは証明書) と考えることができます。最小特権の権限に従い、Azure 内のデータを誤って削除されないように保護する読み取り専用のサービスプリンシパルを作成するには、次の手順に従います。

1. Microsoft Azure クラウドアカウントポータルにログインし、PowerShell でクラウドシェルを起動するか、ワークステーションで Azure コマンドラインインターフェイス (CLI) を使用します。
2. サービスプリンシパルを作成し、Azure Blob ストレージアカウントへの「読み取り専用」アクセス権限を持つように設定します。このコマンドの JSON 出力を `azure-principal.json` というローカルファイルに保存します。ファイルは EC2 インスタンスにアップロードされます。　括弧 ({ と }) で示されているプレースホルダー変数を Azure サブスクリプション ID、リソースグループ名、ストレージアカウント名で置き換えます。

```
az ad sp create-for-rbac `
--name AWS-Rclone-Reader `
--role "Storage Blob Data Reader" `
--scopes /subscriptions/{Subscription ID}/resourceGroups/{Resource Group Name}/
providers/Microsoft.Storage/storageAccounts/{Storage Account Name}
```

カウチベースサーバーから AWS 上のカウチベースカペラへの移行

Battulga Purevragchaa (AWS)、Mark Gamble、Saurabh Shanbhag (AWS) によって作成されました

概要

Couchbase Capella は、ミッションクリティカルなアプリケーション (ユーザープロファイル、オンラインカタログ、在庫管理など) 向けのフルマネージド型の NoSQL データベース・アズ・ア・サービス (DBaaS) です。Couchbase Capella は、Couchbase が管理するAmazon Web Services (AWS) アカウントで DBaaS ワークロードを管理します。Capellaを使用すると、複数のクラスター、複数のAWS リージョン、マルチクラウド、ハイブリッドクラウドのレプリケーションを単一のインターフェイスで簡単に実行および管理できます。

Couchbase Capella を使用すると、Couchbase Server アプリケーションを瞬時にスケーリングできるため、マルチノードクラスターを数分で作成できます。Couchbase Capella は、「SQL++」、「全文検索」、「イベントサービス」、「分析サービス」 など、Couchbase サーバーのすべての機能をサポートしています。また、インストール、アップグレード、バックアップ、一般的なデータベースメンテナンスを管理する必要もありません。

このパターンでは、自己管理型の 「Couchbase Server」 環境を AWS クラウドに移行するための手順とベストプラクティスを説明しています。このパターンは、オンプレミスでもクラウドでも稼働している Couchbase Server クラスターから Couchbase Capella にデータとインデックスを移行するための反復可能なプロセスを提供します。これらの手順を使用すると、移行中の問題を回避し、移行プロセス全体をスピードアップできます。

このパターンには次の 2 つの移行オプションがあります。

- オプション 1 は、移行するインデックスが 50 個未満の場合に適しています。
- オプション 2 は、移行するインデックスが 50 個を超える場合に適しています。

自己管理型の Couchbase Server に「サンプルデータを設定して」、移行ガイドに従うこともできます。

移行オプション 2 を選択した場合や、デフォルト値以外のスコープやコレクションを使用する場合は、追加情報セクションにあるサンプル設定ファイルを使用する必要があります。

前提条件と制限

前提条件

- 既存の Couchbase Capella 有料アカウント。また、「AWS で Couchbase Capella アカウント」を作成し、Couchbase Capella の無料トライアルを使用してから、有料アカウントにアップグレードしてクラスターを移行用に設定することもできます。試用版から始めるには、「Couchbase Capella スタートガイド」の指示に従ってください。
- オンプレミスまたはクラウドサービスプロバイダーにデプロイされた既存の自己管理型の Couchbase Server 環境。
- 移行オプション 2 の場合は、Couchbase シェルと設定ファイル。設定ファイルを作成するには、追加情報セクションにあるサンプルファイルを使用できます。
- Couchbase サーバーと Couchbase Capella の管理に精通していること。
- コマンドラインインターフェイス (CLI) で TCP ポートを開いてコマンドを実行する方法

移行プロセスには、次の表に示す役割と専門知識も必要です。

ロール	専門知識	責任
Couchbase 管理者	• カウチベースサーバーとカウチベース力ペラに精通していること • コマンドラインに関する基本的な知識は役に立ちますが、必須ではありません。	• Couchbase サーバーと Capella 固有のタスク
システム管理者、IT 管理者	• 自己管理型の Couchbase Server システム環境と管理に精通していること	• 自己管理型の Couchbase Server クラスターノードでポートを開き、IP アドレスを決定する

機能制限

- このパターンは、データ、インデックス、および「Couchbase 全文検索インデックスを Couchbase」サーバーから AWS 上の Couchbase Capella に移行するために使用されます。このパターンは「Couchbase イベントサービス」の移行や「Couchbase Analytics」の移行には適用されません。

- カウチベースカペラは複数の AWS リージョンで利用できます。Capella がサポートするリージョンの最新情報については、Couchbase ドキュメントの「Amazon Web Services」を参照してください。

製品バージョン

- 「Couchbase サーバー (コミュニティまたはエンタープライズ) エディションバージョン 5.x 以降」

アーキテクチャ

ソーステクノロジースタック

- カウチベースサーバー

ターゲットテクノロジースタック

- カウチベース・カペラ

ターゲットアーキテクチャ

1. カウチベースカペラにアクセスするには、カペラコントロールプレーンを使用します。 Capella コントロールプレーンを使用して、次のことを行うことができます。
   - アカウントを管理および監視します。
   - クラスターとデータ、インデックス、ユーザーとグループ、アクセス許可、監視、イベントを管理します。
2. クラスターが作成されました。
3. Capella データプレーンは Couchbase が管理する AWS アカウントにあります。新しいクラスターを作成すると、Couchbase Capella は選択した AWS リージョンの複数のアベイラビリティーゾーンにクラスターをデプロイします。
4. Couchbase アプリケーションは、AWS アカウントの VPC で開発およびデプロイできます。通常、この VPC は 「VPC ピアリング」を通じて Capella データプレーンにアクセスします。

ツール

- 「Couchbase クロスデータセンターレプリケーション (XDCR)」は、さまざまなクラウドプロバイダーやさまざまなデータセンターにあるクラスター間でデータを複製するのに役立ちます。自己管理型の Couchbase サーバー· クラスターから Couchbase Capella にデータを移行するために使用されます。

> **Note**
>
> XDCR を Couchbase Server Community Edition で使用して Couchbase Capella に移行することはできません。代わりに 「cbexport」を使用できます。詳細については、コミュニティエディションからのデータの移行エピックを参照してください。

- 「Couchbase Shell」は Couchbase サーバーと Couchbase Capella 用のコマンドラインシェルで、ローカルおよびリモートの Couchbase クラスターにアクセスできます。このパターンでは、Couchbase シェルを使用してインデックスを移行します。
- 「cbexport」は Couchbase クラスターからデータをエクスポートするための Couchbase ユーティリティです。「カウチベースサーバーのCLI ツール」に含まれています。

エピック

移行の準備をする

タスク	説明	必要なスキル
セルフマネージドの Couchbase Server クラスターのサイズを評価します。	Couchbase サーバー用の「Couchbase Web コンソール」にログインし、自己管理型クラスターのノードとバケットを評価します。 1. クラスタノードのリストを表示するには、ナビゲーションバーのサーバータブを選択します。	Couchbase 管理者

タスク	説明	必要なスキル
	2. ノード数を記録し、リスト上の各ノードを選択してプロパティを表示します。 3. 個々のノードのメモリとストレージを記録します。 4. ナビゲーションバーの Buckets タブを選択し、リスト内の各バケットを選択してプロパティを表示します。各バケットの RAM 割り当てとコンフリクト解決の設定を記録します。 Couchbase Capella でターゲットクラスターのサイズ設定と設定を行う際の一般的なガイドとして、自己管理型の Couchbase Server クラスター設定を使用します。 Couchbase Capella のサイジングに関するより詳細な情報が必要な場合は、「Couchbase」にお問い合わせください。	

タスク	説明	必要なスキル
自己管理型の Couchbase サーバークラスターに Couchbase サービスのディストリビューションを記録します。	1. Couchbase Web コンソールでサーバータブを選択し、クラスターノードのリストを表示します。 2. 各ノードを選択してプロパティを表示し、各ノード (データサービス、クエリサービス、インデックスサービス、検索サービス、分析サービス、イベントサービス) の Couchbase Service ディストリビューションを記録します。	Couchbase 管理者
セルフマネージドの Couchbase Server クラスターノードの IP アドレスを記録します。	(コミュニティエディションを使用する場合、このステップは無視します。) クラスター内の各ノードの IP アドレスを記録します。これらは後で Couchbase Capella クラスターの許可リストに追加されます。	Couchbase 管理者、システム管理者

Couchbase Capella にリソースをデプロイして設定する

タスク	説明	必要なスキル
テンプレートの選択。	1. Couchbase Capella コントロールプレーンにログインし、メインナビゲーションでダッシュボードタブまたはクラスタータブを選択し、クラスターの作成を選択します。	Couchbase 管理者

タスク	説明	必要なスキル
	2. 自己管理型の Couchbase Server クラスターの評価で記録した情報を使用して、設定の要件を満たすクラスターテンプレートを選択します。適切なテンプレートが見つからない場合は、クラスター・サイジング・エディターでカスタム・テンプレートを選択します。	
ノードを選択して設定します。	ノード数、サービス分布、コンピュートまたは RAM、ストレージなど、自己管理型の Couchbase Server クラスター環境に合わせてノードを選択し、設定します。 Couchbase Capella は 「多次元スケーリング」のベストプラクティスを採用しています。サービスとノードは、デプロイメントのベストプラクティスに従ってのみ選択できます。つまり、自己管理型の Couchbase Server クラスターの設定と完全に一致させることはできないということかもしれません。	Couchbase 管理者

タスク	説明	必要なスキル
クラスターをデプロイします。	サポートゾーンとサポートパッケージを選択し、クラスターをデプロイします。詳細な手順と手順については、Couchbase「ドキュメントの「クラスターの作成」」を参照してください。 **⚠ Important** Couchbase Capella 無料トライアルを使用している場合は、移行を開始する前に有料アカウントに変換する必要があります。アカウントを変換するには、Couchbase Capella コントロールプレーンの請求セクションを開き、「アクティベーション ID を追加」を選択します。アクティベーション ID は、Couchbase Sales との購入契約を完了した後、または「AWS Marketplace」を通じて購入を行った後に、請求連絡先のメールアドレスに送信されます。	Couchbase 管理者

タスク	説明	必要なスキル
データベース認証情報ユーザーを作成します。	データベース認証情報はクラスター固有のもので、ユーザー名、パスワード、およびバケット権限セットで構成されます。このユーザーは、バケットの作成とバケットデータへのアクセスに必要です。 Couchbase Capella コントロールプレーンで、Couchbase Capella ドキュメントの「データベース認証情報の設定」の指示に従って、新しいクラスターのデータベース認証情報を作成します。 **Note** リモートまたは Couchbase Capella UI を介して特定のクラスターのバケットデータにアクセスする場合、組織ユーザーには組織ロールの認証情報を割り当てる必要があります。これは、アプリやインテグレーションで一般的に使用されるデータベース認証情報とは異なります。組織ユーザーを作成すると、Couchbase Capella クラスターにターゲットバケット	Couchbase 管理者

タスク	説明	必要なスキル
	を作成して管理できます。	
マイグレーションオプション 2 を使用している場合は、Couchbase シェルをインストールします。	Couchbase シェルは、自己管理型の Couchbase サーバーと Couchbase Capella クラスターの両方にネットワークアクセスできる任意のシステムにインストールできます。詳細については、Couchbase シェルのドキュメンテーションの「Couchbase シェルのバージョン 1.0.0-beta.5 のインストール」を参照してください。 コマンドラインターミナルで「セルフマネージドクラスターへの接続をテストして」、Couchbase シェルがインストールされていることを確認します。	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
IP アドレスを許可します。	1. Couchbase Capella コントロールプレーンでクラスター」を選択し、次にターゲットクラスターを選択します。 2. クラスターの Connect タブを選択し、Manage Allowed IP の下に表示されているクラスターの接続エンドポイントを記録します。 3. Couchbase シェルをインストールしたシステムの IP アドレスと自己管理型の Couchbase Server クラスターインスタンスの IP アドレスを許可された IP アドレスとして追加するには、次の操作を行います。 a. ワイド・エリア・ネットワークで許可された IP を管理を選択します。 b. 許可された IP アドレスを追加を選択し、Couchbase シェルをインストールしたシステムの IP アドレスを入力して、「IP アドレスを追加を選択します。 c. 前のステップを繰り返して自己管理の Couchbase Server クラスターインスタンスの IP アドレスを追加します。	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
	許可される IP アドレスの詳細については、Couchbase ドキュメントの「許可される IP アドレスの設定」を参照してください。	

タスク	説明	必要なスキル
証明書の設定	1. クラスターのルート証明書をダウンロードするには、ルート証明書でダウンロードを選択します。 2. .pem ファイル拡張子を使用して、Couchbase シェルを実行するシステム上のフォルダーにルート証明書を保存します。 3. 次に、自己管理型の Couchbase Server Web コンソールにログインし、左側のナビゲーションバーでセキュリティを選択し、「証明書」タブを選択します。 4. 自己管理型の Couchbase Server クラスターのルート証明書をコピーし、Couchbase Capella クラスターのルート証明書ファイルを保存したのと同じフォルダに.pem ファイルとして保存します。ルート証明書について詳しくは、「Couchbase Server ドキュメンテーションのルート証明書」を参照してください。	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
Couchbase シェルの設定ファイルを作成します。	Couchbase シェルインストールのホームディレクトリ (例:/<HOME_DIRECTORY>/.cbsh/config) に設定ドットファイルを作成します。詳細については、Couchbase ドキュメントの「Config dotfiles」を参照してください。 ソースクラスターとターゲットクラスターの接続プロパティーを設定ファイルに追加します。追加情報セクションにあるサンプルファイルを使用でき、クラスターの設定を編集できます。 .cbsh 更新した設定を含む設定ファイルをフォルダー (例:/<HOME_DIRECTORY>/.cbsh/config) に保存します。	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
ターゲットバケットを作成する。	Couchbase ドキュメントの「バケットの作成」の指示に従って、ソースバケットごとに Couchbase Capella クラスターに 1 つの「ターゲットバケット」を作成します。 ターゲット・バケット設定は、自己管理型の Couchbase Server クラスターのバケットのバケット名、メモリー設定、およびコンフリクト解決設定と一致する必要があります。	Couchbase 管理者

タスク	説明	必要なスキル

スコープとコレクションを作成する。	すべてのバケットには、`_default._default` キースペースを含むデフォルトのスコープとコレクションが含まれています。 スコープとコレクションに他のキースペースを使用している場合は、ターゲット Capella クラスターに同一のキースペースを作成する必要があります。
	
	1. Couchbase シェルをインストールしたシステム上でコマンドラインターミナルを開きます。
	2. Couchbase シェルを起動するには、以下のコマンドを実行します。
	
	`./cbsh`
	
	3. 移行するバケットごとに、以下のコマンドを実行して Capella クラスターにスコープとコレクションを作成します。<BUCKET_NAME> 移行するバケットの名前に置き換えることを確認します。
	
	`scopes --clusters "On-Prem-Cluster" --bucket <BUCKET_NAME> | select scope | where scope != "_default" | each`
		Couchbase 管理者

タスク	説明	必要なスキル
	{ |it| scopes create $it.scope --clusters "Capella-Cluster" } collections --clusters "On-Prem-Cluster" --bucket <BUCKET_NAME> | select scope collection | where $it.scope != "_default" | where $it.collection != "_default" | each { |it| collections create $it.collection --clusters "Capella-Cluster" --bucket <BUCKET_NAME> --scope $it.scope }	

エンタープライズエディションからのデータの移行

タスク	説明	必要なスキル
自己管理型の Couchbase サーバー・クラスター・ノードで TCP ポートを開きます。	自己管理型の Couchbase Server クラスターのノードで XDCR 通信用の適切なポートが開いていることを確認します。詳細については、「Couchbase サーバーポートドキュメント」を参照してください。	Couchbase 管理者、システム管理者
Couchbase Server Enterprise Edition を使用している場合は、Couchbase XDCR をセットアップします。	1. Couchbase Capella コントロールプレーンのメインナビゲーションでクラスタ」を選択し、マイグレーションするターゲットクラスタを選択します。	Couchbase 管理者

タスク	説明	必要なスキル
	2. ルート証明書でコピーを選択します。 3. 自己管理型の Couchbase Server Web コンソールにログインし、メインナビゲーションで XDCR を選択します。次に、リモートの追加を選択します。 4. 以下の設定を入力します。 • クラスター名 — Capella クラスター接続の名前 • IP/ホスト名 — Couchbase Capella クラスターの接続エンドポイント • リモートクラスターのユーザー名 — Couchbase Capella クラスターのデータベースユーザー • パスワード — Couchbase Capella クラスタのデータベースユーザーパスワード • セキュア・コネクションを有効化 — 選択済み • 完全 (パスワードとデータを TLS で暗号化) — 選択済み 5. 前にコピーした Capella クラスタのルート証明書を貼り付け、[保存] を選択します。	

タスク	説明	必要なスキル
カウチベース XDCR を起動します。	1. 自己管理型の Couchbase Server Web コンソールで、メインナビゲーションで XDCR を選択し、「レプリケーションを追加」を選択します。 2. 以下の設定を入力します。 • バケットから複製 — 移行するソースバケットを選択します。 • リモートバケット — ターゲットバケット名を入力します。 • リモートクラスタ — 以前に作成したターゲットクラスタを選択します。 3. レプリケーションを保存を選択します。レプリケーション処理は数秒以内に開始されます。	Couchbase 管理者

オプション 1 を使用してインデックスを移行します。

タスク	説明	必要なスキル
セルフマネージド・クラスター・インデックスを Couchbase Capella に移行します。	⚠ Important 移行するインデックスが 50 未満の場合、このプロセスをお勧めします。移行するインデックスが 50 個以上ある場合は、移行オプ	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
	ション 2 を使用することをお勧めします。 1. Couchbase Web コンソールで、「インデックス」を選択します。 2. インデックスのリストで、移行する最初のインデックスを選択します。その後、インデックス定義が表示されます。 3. CREATE ステートメントを使用してインデックス定義をコピーしますが、コピーはしないでください `WITH { "defer_build":true }` 。 たとえば、次のインデックス定義の例ではコピーのみを行います `CREATE INDEX` `` `cityindex` ON `travel-sample`(`city`) `` 。 `CREATE INDEX `cityindex` ON `travel-sample`(`city`) WITH { "defer_build":true }` 4. Couchbase Capella コントロールプレーンでクラスターを選択し、次にター	

タスク	説明	必要なスキル
	ゲットクラスターを選択します。 5. ツール」ドロップダウン・リストでクエリー・ワークベンチを選択します。CREATE 前にコピーしたステートメントをクエリエディターに貼り付け、Execute を選択します。これによりインデックスが作成され、構築されます。 6. インデックスが作成されたことを確認するには、ツールドロップダウンリストからインデックスを選択します。リストには、インデックスが作成および作成されたことが表示されます。 7. 移行するインデックスごとに、このプロセスを繰り返します。	

オプション 2 を使用してインデックスを移行します。

タスク	説明	必要なスキル
インデックス定義を移行します。	⚠ Important 移行するインデックスが 50 個を超える場合は、このプロセスをお勧めします。移行するインデックスが 50 個	Couchbase 管理者、システム管理者

タスク	説明	必要なスキル

	未満の場合は、移行オプション 1 を使用することをお勧めします。
	
	1. Couchbase シェルをインストールしたシステム上でコマンドラインターミナルを開きます。
	2. Couchbase シェルを起動するには、以下のコマンドを実行します。
	`./cbsh`
	3. セルフマネージドの Couchbase Server クラスターに接続するには、以下のコマンドを実行します。
	`cb-env cluster On-Prem-Cluster`
	4. 自己管理型の Couchbase Server クラスターから Couchbase Capella クラスターにインデックス定義を移行するには、移行する各バケットに対して以下のコマンドを実行します。必ず、<BUCKET_NAME> 移行するインデックスに対応するバケット名に置き換えてください。この移行オプションでは、ターゲットバケット名がソースバケット	

タスク	説明	必要なスキル
	名と同一である必要があります。 `query indexes --definitions | where bucket =~ <BUCKET_NAME>| get definition | each { |it| query $it --clusters Capella-Cluster }`	

タスク	説明	必要なスキル

インデックス定義を構築します。	1. コンテキストを Couchbase Capella クラスターに切り替えるには、以下のコマンドを実行します。
	
	`cb-env cluster Capella-Cluster`
	
	2. Couchbase Capella クラスターに移行されたインデックス定義を作成するには、以下のコマンドを実行し、<BUCKET_NAME> 構築するインデックスに対応するバケット名に置き換えます。
	
	`query 'SELECT RAW CONCAT("BUILD INDEX ON ", k , "(['", CONCAT2 ("','", inames), "']);") FROM system:indexes AS s LET bid = CONCAT("` ",s.bucket_id, "`"), sid = CONCAT("`", s.scope_id, "`"), kid = CONCAT("` ", s.keyspace_id, "`"), k = NVL2(bid, CONCAT2(".", bid, sid, kid), kid) WHERE s.namespa ce_id = "default" AND s.bucket_id = "" GROUP BY k LETTING inames = ARRAY_AGG (s.name) FILTER (WHERE s.state =`
		Couchbase 管理者、システム管理者

タスク	説明	必要なスキル
	'deferred') HAVING ARRAY_LENGTH(inames) > 0;' | each { | it| query $it } 3. 各バケットについて、これを繰り返します。	

フルテキスト検索インデックスの移行

タスク	説明	必要なスキル
自己管理型クラスターの全文検索インデックスを Couchbase Capella に移行します。	1. Couchbase ウェブコンソールで検索を選択します。 2. 全文検索 (FTS) インデックスのリストで、移行する最初の FTS インデックスを選択し、インデックス定義 JSON を表示を選択し、クリップボードにコピーを選択します。インデックス名とそれが属するバケットをメモしておきます。 3. Couchbase Capella コントロールプレーンでクラスター」を選択し、次にターゲットクラスターを選択します。 4. 「ツール」ドロップダウンリストで「全文検索」を選択します。 5. インデックスのインポートを選択し、FTS インデックス定義を貼り付けます。	Couchbase 管理者

タスク	説明	必要なスキル
	6. インデックス名を入力し、セルフマネージドクラスタに記載されている正しいバケットを選択してから、Create を選択します。 7. 移行する必要がある FTS インデックスごとにこのプロセスを繰り返します。	

Couchbase コミュニティエディションからのデータの移行

タスク	説明	必要なスキル

セルフマネージドの Couchbase Server コミュニティエディションからデータをエクスポートします。	暗号化された XDCR は Couchbase コミュニティエディションでは使用できません。Couchbase コミュニティエディションからデータをエクスポートし、そのデータを Couchbase Capella に手動でインポートできます。
	
	ソースバケットからデータをエクスポートするには、`cbexport` コマンドラインで使用します。
	
	次のコマンドが例として提供されます。
	
	`cbexport json \`
	`--cluster localhost \`
	`--bucket <SOURCE BUCKET NAME> \`
	`--format lines \`
		Couchbase 管理者

タスク	説明	必要なスキル
	``` --username <USERNAME> \ --password <PASSWORD> \ --include-key cbkey \ --scope-field cbscope \ --collection-field cbcoll \ --output cbexporte d_data.json ``` cbkey、cbscope、cbcoll、cd_data.json は任意のラベルであることに注意してください。これらはプロセスの後半で参照されるため、別の名前を付ける場合は、書き留めておいてください。	

タスク	説明	必要なスキル
データをカウチベースカペラにインポートします。	1. Couchbase Capella コントロールプレーンでクラスターを選択し、次にターゲットクラスターを選択します。 2. ツールドロップダウンリストでインポートを選択します。これにより、次の 6 つのステップを含むウィザードが開きます。 a. バケット — ターゲットバケットを選択します。 b. ファイル — JSON を選択し、ライン を選択して、ウェブブラウザを使用する を選択します。データが大量にある場合、手動オプションがあります。`cbexport` で作成されたファイルを選択します。 c. コレクション — カスタムコレクションマッピングを選択します。 Community Edition データベースがスコープやコレクションを使用しない場合、または _default のみを使用している場合は、代わりに単一コレクションを選択オプションを選択できます。	Couchbase 管理者

タスク	説明	必要なスキル
	コレクションマッピング式には、%cbscope%.%cbcoll% と入力します。この式が正しく機能することを確認するには、次のようなサンプルデータを貼り付けます。 `{ "cbscope" :"inventory", "cbcoll":"landmark", "cbkey":" landmark_3991" }` d. キー — 顧客生成を選択します。(インポートするデータのキーを保存したくない場合は、代わりに自動生成 UUID を選択してステップ 5 に進んでください)。キー名ジェネレータエクスプレッションには、%cbkey% と入力します。この式が正しく動作することを確認するには、サンプルデータを貼り付けます。 e. コンフィギュレーション — フィールドを無視を選択し、「cbscope」、「cbcoll」、「cbkey」と入力します。これらのフィールドには、インポート後にターゲットバ	

タスク	説明	必要なスキル
	ケットに格納する必要のない一時的な情報が含まれます。その他の設定はデフォルト値のままにしておきます。 f. インポート — 確認し、準備ができたらインポートを選択します。アップロードとデータのインポートを待ちます。 大きなファイルの場合、Couchbase Capella は cURL を使ったコマンドラインインポートをサポートしています。インポートオプションの詳細については、Couchbase Capella ドキュメントの「データのインポート」を参照してください。	

移行のテストと検証

タスク	説明	必要なスキル
データ移行の確認。	1. Couchbase Capella コントロールプレーンでクラスターを選択し、次にクラスとリストでターゲットクラスターを選択します。 2. ターゲットクラスターのバケットタブを選択します。ターゲットバケットのアイテム(ドキュメント) の数が	Couchbase 管理者

タスク	説明	必要なスキル
	ソースバケットのアイテム数と一致することを確認します。 3. ターゲットクラスターのツールドロップダウンリストで「ドキュメント」を選択します。すべてのドキュメントが移行されたことを確認します。 4. (オプション) すべてのデータが移行されたら、削除してレプリケーションを停止できます。詳細については、Couchbase ドキュメントの「レプリケーションの削除」を参照してください。	
インデックスの移行を検証します。	Couchbase Capella コントロールプレーンのターゲットクラスターのツールドロップダウンリストで、インデックスを選択します。インデックスが移行され、構築されていることを確認します。	Couchbase 管理者

タスク	説明	必要なスキル
クエリ結果を検証します。	1. ターゲットクラスターのツールロップダウンリストのCouchbase Capella コントロールプレーンで、クエリワークベンチを選択します。 2. サンプル N1QL クエリ、またはアプリケーションで使用されているクエリを実行します。自己管理型の Couchbase Server クラスターでクエリを実行したときと同じ結果が得られることを確認します。	Couchbase 管理者
全文検索結果を検証します (FTS インデックスを移行した場合に適用)。	1. ターゲットクラスターのツールロップダウンリストのCouchbase Capella コントロールプレーンで、全文検索を選択します。 2. FTS インデックスの名前を選択して選択します。 3. [検索] を選択してください。 4. サンプル検索クエリを入力し、Search を選択します。 5. 結果が自己管理クラスターで検索を実行したときと同じであることを確認します。	Couchbase 管理者

関連リソース

移行の準備をする

- 「カウチベース・カペラの無料トライアルを始めましょう。」
- 「カウチベースカペラのクラウドプロバイダー要件」
- 「カウチベース・カペラのサイジング・ガイドライン」

データとインデックスを移行してください。

- 「カウチベース XDCR」
- 「カウチベース、シェル、ドキュメンテーション」

カウチベース・カペラ SLA とサポート

- 「カウチベースカペラサービスレベルアグリーメント (SLA」)
- 「カウチベース・カペラ・サービス・サポート・ポリシー」

追加情報

次のコード例は、「Couchbase Shellの設定ファイル」です。

```
Version = 1

[[clusters]]
identifier = "On-Prem-Cluster"
hostnames = ["<SELF_MANAGED_COUCHBASE_CLUSTER>"]
default-bucket = "travel-sample"
username = "<SELF_MANAGED_ADMIN>"
password = "<SELF_MANAGED_ADMIN_PWD>"
tls-cert-path = "/<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"

[[clusters]]
identifier = "Capella-Cluster"
hostnames = ["<COUCHBASE_CAPELLA_ENDPOINT>"]
default-bucket = "travel-sample"
username = "<CAPELLA_DATABASE_USER>"
```

```
password = "<CAPELLA_DATABASE_USER_PWD>"
tls-cert-path = "/<ABSOLUTE_PATH_TO_COUCHBASE_CAPELLA_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"
```

設定ファイルを保存する前に、以下の表を参照して、独自のソースとターゲットのクラスター情報を追加したことを確認します。

<SELF_MANAGED_COUCHBASE_CLUSTER>	セルフマネージドの Couchbase サーバークラスターの IP アドレスを使用します。
<SELF_MANAGED_ADMIN>	セルフマネージドの Couchbase サーバークラスターの管理者ユーザーを使用します。
<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>	自己管理型の Couchbase Server クラスターに保存されているルート証明書ファイルへの絶対パスを使用してください。
<COUCHBASE_CAPELLA_ENDPOINT>	Couchbase Capella クラスターの接続エンドポイントを使用します。
<CAPELLA_DATABASE_USER>	Couchbase Capella クラスターにはデータベースユーザーを使用します。
<CAPELLA_DATABASE_USER_PWD>	Couchbase Capella クラスターのユーザーパスワードを使用します。
<ABSOLUTE_PATH_TO_COUCHBASE_CAPELLA_ROOT_CERT>	Couchbase Server クラスターに対して、保存されたルート証明書ファイルの絶対パスを使用します。

IBM WebSphere アプリケーションサーバーから Amazon EC2 の Apache Tomcat に移行

作成者: Neal Ardeljan (AWS) と Afroz Khan (AWS)

概要

このパターンでは、IBM WebSphere アプリケーションサーバー (WAS) を実行しているオンプレミスの Red Hat Enterprise Linux (RHEL) 6.9 以降のシステムから、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで Apache Tomcat を実行している RHEL 8 に移行する手順を段階的に説明します。

このパターンは、以下のソースバージョンとターゲットバージョンに適用されます：

- WebSphere アプリケーションサーバー 7.x から Apache Tomcat 8 (Java 7 以降搭載)
- WebSphere アプリケーションサーバー 8.x から Apache Tomcat 8 (Java 7 以降搭載)
- WebSphere アプリケーションサーバー 8.5.5.x から Apache Tomcat 9 (Java 8 以降搭載)
- WebSphere アプリケーションサーバー 8.5.5.x から Apache Tomcat 10 (Java 8 以降搭載)

前提条件と制限

前提条件

- アクティブなAWS アカウント
- ソース Java コード、以下を前提とします：
  - Java 7 以降の Java 開発キット (JDK) バージョンを使用
  - Spring または Apache Struts フレームワークを使用
  - エンタープライズ Java Beans (EJB) フレームワークや、Tomcat ですぐに利用できないその他の WebSphere サーバー機能は使用しないでください。
  - 主にサーブレットまたは Java サーバーページ (JSP) を使用
  - Java データベース接続性 (JDBC) コネクタを使用してデータベースに接続します
- ソース IBM WebSphere アプリケーションサーバーのバージョン 7.x 以降
- Apache Tomcat バージョン 8.5 以降がターゲット

アーキテクチャ

ソーステクノロジースタック

- Apache Struts モデルビューコントローラー (MVC) フレームワークを使用して構築されたウェブアプリケーション
- IBM WebSphere アプリケーションサーバーバージョン 7.x または 8.x 上で動作するウェブアプリケーション
- Lightweight Directory Access Protocol (LDAP) コネクタを使用して、 LDAP ディレクトリ (iPlanet/eTrust) に接続するウェブアプリケーション
- IBM Tivoli アクセスマネージャー(TAM) 接続を使用して TAM ユーザーパスワードを更新するアプリケーション (現在の実装では、アプリケーションは PD.jar を使用しています)

オンプレミスのデータベース

- Oracle Database 21c (21.0.0.0)
- Oracle Database 19c (19.0.0.0)
- Oracle Database 12c Release 2 (12.2.0.1)
- Oracle Database 12c Release 1 (12.1.0.2)

ターゲットテクノロジースタック

- EC2 インスタンスの RHEL 上で実行されている Apache Tomcat バージョン 8 (またはそれ以降)
- Oracle 向け PostgreSQLのAmazon Relational Database Service (Amazon RDS)[1]

Amazon RDS でサポートされている Oracle のバージョンの詳細については、「Amazon RDS for Oracle」 のウェブサイトを参照してください。

ターゲットアーキテクチャ

ツール

- アプリケーション層: Java アプリケーションを WAR ファイルに再構築します。
- データベース層: Oracle のネイティブバックアップおよび復元。
- Jakarta EE 向けの Apache Tomcat 移行ツール。このツールは、Apache Tomcat 9 上で動作する Java EE 8 用に作成されたウェブアプリケーションを、Jakarta EE 9 を実装している Apache Tomcat 10 上で動作するように自動的に変換します。

エピック

移行を計画する

タスク	説明	必要なスキル
アプリケーション検出、現在の状態フットプリント、パフォーマンスベースラインを完了します。		BA、移行リーダー
ソースとターゲットデータベースのバージョンを検証します。		DBA
ターゲットサーバーの EC2 インスタンスのハードウェア要件を識別します。		DBA、SysAdmin
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、SysAdmin
容量、ストレージ特徴量、ネットワーク特徴量に基づいて適切な EC2 インスタンスタイプを選択します。		DBA、SysAdmin
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。		DBA、SysAdmin
アプリケーション移行戦略とツールを識別します。		DBA、移行リーダー
アプリケーションの移行設計と移行ガイドを完成させます。		ビルドリード、移行リード

タスク	説明	必要なスキル
アプリケーション移行ランブックを完成させます。		ビルドリード、カットオーバーリード、テストリード、移行リード

インフラストラクチャを構成する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成		SysAdmin
セキュリティグループを作成します。		SysAdmin
Amazon RDS for Oracle を設定し、起動します。		DBA、SysAdmin

データを移行する

タスク	説明	必要なスキル
データベースバックアップファイルを取得するためのエンドポイントを作成するか、エンドポイントのアクセスを取得します。		DBA
ネイティブデータベースエンジンまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。	詳細については、「追加情報」セクションの「データベースオブジェクトとデータの移行」を参照してください。	DBA

アプリケーションを移行する

タスク	説明	必要なスキル
移行のための変更要求 (CR) を提出します。		カットオーバーリード
移行のCR の承認を取得します。		カットオーバーリード
アプリケーション移行ランブックのアプリケーション移行戦略に従います。	詳細については、「追加情報」セクションの「アプリケーション層の設定」を参照してください。	DBA、移行エンジニア、アプリ所有者
アプリケーションをアップグレードします (必要な場合)。		DBA、移行エンジニア、アプリ所有者
機能テスト、非機能テスト、データ検証テスト、SLAテスト、パフォーマンステストを完了します。		テストリード、アプリオーナー、アプリユーザー

カットオーバー

タスク	説明	必要なスキル
アプリ所有者またはビジネスオーナーから承認を取得します。		カットオーバーリード
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、移行エンジニア、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、移行エンジニア、SysAdmin
プロジェクト文書を確認して検証する。		移行リード
移行の所要時間、手動タスクと自動タスクの割合、コスト削減などのメトリクスを収集します。		移行リード
プロジェクトを終了し、フィードバックを提供します。		移行リーダー、アプリ所有者

関連リソース

リファレンス

- 「Apache Tomcat 10.0 ドキュメント」
- 「Apache Tomcat 9.0 ドキュメント」
- 「Apache Tomcat 8.0 ドキュメント」
- 「Apache Tomcat 8.0 インストールガイド」
- 「Apache Tomcat JNDI ドキュメント」
- 「Amazon RDS for Oracle ウェブサイト」
- 「Amazon RDS の価格設定」
- 「OracleとAmazon Web Services」
- Amazon RDS 上の Oracle
- 「Amazon RDS マルチ AZ 配置」

チュートリアルと動画

- 「Amazon RDS の開始方法」

追加情報

「データベースオブジェクトとデータの移行」

たとえば、Oracle のネイティブバックアップ/リストアユーティリティを使用する場合:

1. Amazon Simple Storage Service (Amazon S3) のバックアップを作成します (オプション)。
2. Oracle DB データをネットワーク共有フォルダにバックアップします。
3. 移行ステージングサーバーにログインして、ネットワーク共有フォルダーをマップします。
4. ネットワーク共有フォルダーからデータを S3 バケットにコピーします。
5. Amazon RDS マルチ AZ 配置をリクエストします。
6. オンプレミスデータベースのバックアップを Amazon RDS for Oracle に復元します。

アプリケーション層をセットアップします

1. Apache Tomcat のウェブサイトから Tomcat 8 (または 9/10) をインストールします。
2. アプリケーションと共有ライブラリーを WAR ファイルにパッケージ化します。
3. WAR ファイルを Tomcat にデプロイします。
4. WebSphere から見つからない共有ライブラリーを `Linux cat` するために、開始ログを監視します。
5. いずれかのWebSphere 固有のデプロイ記述子拡張を `Linux cat` するために、開始レコードを監視します。
6. 見つからない依存 Java ライブラリーを WebSphere サーバーから収集します。
7. WebSphere 固有のデプロイ記述子要素を Tomcat と互換性のある同等の要素で修正します。
8. 依存する Java ライブラリーと更新済みのデプロイ記述子を使用して、WAR ファイルをリビルドします。
9. LDAP 設定、データベース設定、テスト接続を更新します (Apache Tomcat ドキュメントの「レルム設定のハウツー」と「JNDI データソースのハウツー」を参照)。
10. インストールされたアプリケーションを、復元された Amazon RDS for Oracle データベース に対しテストします。
11. EC2 インスタンスからLinux用の新しい Amazon マシンイメージ (AMI) を作成します。
12. 完成したアーキテクチャを、Application Load Balancer と Auto Scaling グループで起動します。

13Application Load Balancer を指し示すように、URL を更新します (WebSEAL ジャンクションを使用して) 。

14構成管理データベース (CMDB)を更新します。

IBM WebSphere アプリケーションサーバーから Amazon EC2 上の Apache Tomcat への自動スケーリングによる移行

作成者: Kevin Yung (AWS) と Afroz Khan (AWS)

概要

このパターンは、Amazon EC2 Auto Scaling が有効になっている Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで Java アプリケーションを IBM WebSphere アプリケーションサーバーから Apache Tomcat に移行するためのガイダンスを提供します。

このパターンを使用すると、以下を実現できます。

- IBM のライセンスコストの削減
- 高可用性を重視したマルチ AZ 配置
- Amazon EC2 Auto Scaling によるアプリケーションの耐障害性の向上

前提条件と制限

前提条件

- Java アプリケーション (バージョン 7.x または8.x) は LAMP スタックで開発する必要があります。
- ターゲット状態は、Linux ホストで Java アプリケーションをホストすることです。このパターンは Red Hat Enterprise Linux (RHEL) 7 環境で正常に実装されました。他の Linux ディストリビューションでもこのパターンに従うことができますが、Apache Tomcat ディストリビューションの構成を参照する必要があります。
- Java アプリケーションの依存関係を理解しておく必要があります。
- 変更を加えるには、Java アプリケーションのソースコードへのアクセスが必要です。

制限事項とリプラットフォームに関する変更

- エンタープライズアーカイブ (EAR) コンポーネントについて理解し、すべてのライブラリがウェブコンポーネント WAR ファイルにパッケージ化されていることを確認してください。Apache Maven WAR プラグインを構成し、WAR ファイルのアーティファクトを生成する必要があります。

- Apache Tomcat 8 を使用する場合、servlet-api.jar とアプリケーションパッケージの組み込み jar ファイルとの間に競合が発生することが知られています。この問題を解決するには、アプリケーションパッケージから servlet-api.jar を削除します。
- Apache Tomcat 構成のクラスパスにある WEB-INF/リソースを構成する必要があります。デフォルトでは、JAR ライブラリはディレクトリにロードされません。または、すべてのリソースを src/main/resources の下にデプロイすることもできます。
- Java アプリケーション内にハードコーディングされたコンテキストルートがあるかどうかを確認し、Apache Tomcat の新しいコンテキストルートを更新します。
- JVM ランタイムオプションを構成するには、Apache Tomcat の bin フォルダに JAVA_OPTS、JAVA_HOME などの構成ファイル setenv.sh を作成します。
- 認証はコンテナレベルで構成され、Apache Tomcat 構成のレルムとしてセットアップされます。認証は次の 3 つのレルムのいずれかで確立されます。
  - JDBC データベースレルムは、JDBC ドライバーがアクセスするリレーショナルデータベース内のユーザーを検索します。
  - DataSource データベースレルムは、JNDI がアクセスするデータベース内のユーザーを検索します。
  - JNDI ディレクトリレルムは、JNDI プロバイダーがアクセスする Lightweight Directory Access Protocol (LDAP) ディレクトリ内のユーザーを検索します。検索には以下が必要です。
    - LDAP 接続の詳細: ユーザー検索ベース、検索フィルター、ロールベース、ロールフィルター
    - 主要な JNDI ディレクトリレルム: LDAP に接続し、ユーザーを認証し、ユーザーがメンバーとなっているすべてのグループを取得します。
- 権限: web.xml 内の権限制約をチェックするロールベースの権限を持つコンテナの場合、ウェブリソースを定義し、制約で定義されたロールと比較する必要があります。LDAP にグループロールマッピングがない場合は、web.xml で属性 <security-role-ref> を設定してグループロールマッピングを行う必要があります。構成ドキュメントの例については、Oracle ドキュメントを参照してください。
- データベース接続: Amazon Relational Database Service (Amazon RDS) のエンドポイント URL と接続の詳細を使用して Apache Tomcat でリソース定義を作成します。JNDI 検索を使用して DataSource を参照するようにアプリケーションコードを更新します。WebSphere で定義されている既存の DB 接続は、WebSphere の JNDI 名を使用するため機能しません。JNDI 名と DataSource タイプの定義を含む <resource-ref> エントリを web.xml に追加できます。サンプル構成ドキュメントを見るには、Apache Tomcat ドキュメントを参照してください。
- ロギング: デフォルトでは、Apache Tomcat はコンソールまたはログファイルにログを記録します。logging.properties を更新することでレルムレベルのトレースを有効にできます (「Tomcat で

ロギングする」を参照)。Apache Log4j を使用してログをファイルに追加する場合は、tomcat-juli をダウンロードしてクラスパスに追加する必要があります。

- セッション管理: アプリケーションの負荷分散とセッション管理用に IBM WebSEAL をそのまま使用する場合は、変更は必要ありません。AWS の Application Load Balancer または Network Load Balancer を使用して IBM WebSEAL コンポーネントを置き換える場合は、Memcached クラスターで Amazon ElastiCache インスタンスを使用してセッション管理を設定し、オープンソースのセッション管理を使用できるように Apache Tomcat をセットアップする必要があります。
- IBM WebSEAL フォワードプロキシを使用している場合は、AWS に新しい Network Load Balancer をセットアップする必要があります。WebSEAL ジャンクション構成には、Network Load Balancer が提供する IP を使用してください。
- SSL 構成: エンドツーエンドの通信にはセキュアソケットレイヤー (SSL) を使用することをお勧めします。Apache Tomcat で SSL サーバー構成をセットアップするには、Apache Tomcat ドキュメントに記載されている指示に従ってください。

アーキテクチャ

ソーステクノロジースタック

- IBM WebSphere アプリケーション

ターゲットテクノロジースタック

- アーキテクチャは Elastic Load Balancing (バージョン 2) を使用しています。アイデンティティ管理と負荷分散に IBM WebSEAL を使用している場合は、AWS 上のNetwork Load Balancer を選択して IBM WebSEAL リバースプロキシと統合できます。
- Java アプリケーションは、Amazon EC2 Auto Scaling グループの EC2 インスタンスで実行されている Apache Tomcat アプリケーションサーバーにデプロイされます。CPU 使用率などの Amazon CloudWatch メトリクスに基づいてスケーリングポリシーを設定できます。
- 負荷分散のための IBM WebSEAL の使用を廃止する場合は、Amazon ElastiCache for Memcached をセッション管理に使用できます。
- バックエンドデータベースでは High Availability (Multi-AZ) for Amazon RDS をデプロイし、データベースエンジンタイプを選択できます。

ターゲットアーキテクチャ

ツール

- CloudFormation
- コマンドラインインターフェイス (CLI)
- Apache Tomcat (バージョン 7.x または 8.x)
- RHEL 7 or Centos 7
- Amazon RDS マルチ AZ デプロイ
- Amazon ElastiCache for Memcached (オプション)

エピック

VPC をセットアップする

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成		
サブネットを作成します。		
必要に応じてルートテーブルを作成する。		
ネットワークアクセスコントロールリスト (ACL)		
AWS Direct Connect または会社の VPN 接続をセットアップする。		

アプリケーションをリプラットフォームする

タスク	説明	必要なスキル
アプリケーションビルド Maven 構成をリファクタリングして WAR アーティファクトを生成する。		
Apache Tomcat のアプリケーション依存関係データソースをリファクタリングする。		
Apache Tomcat の JNDI 名を使用するようにアプリケーションのソースコードをリファクタリングする。		
WAR アーティファクトを Apache Tomcat にデプロイする。		
アプリケーションの検証とテストを完了する。		

ネットワークを構成する

タスク	説明	必要なスキル
依存関係サービスへの接続を許可するように会社ファイアウォールを構成する。		
エンドユーザーが AWS の Elastic Load Balancing にアクセスできるように企業ファイアウォールを構成する。		

アプリケーションインフラストラクチャを作成する

タスク	説明	必要なスキル
EC2 インスタンスにアプリケーションを作成してデプロイする。		
セッション管理用の Amazon ElastiCache for Memcached クラスターを作成する。		
バックエンドデータベース用の Amazon RDS マルチ AZ インスタンスを作成する。		
AWS Certificate Manager (ACM) を使用して SSL 証明書を作成し、インポートします。		
SSL 証明書をロードバランサーにインストールする。		
SSL 証明書を Apache Tomcat サーバーにインストールする。		
アプリケーションの検証とテストを完了する。		

カットオーバー

タスク	説明	必要なスキル
既存のインフラストラクチャをシャットダウンする。		

タスク	説明	必要なスキル
データベースを本番環境から Amazon RDS に復元する。。		
DNS を変更してアプリケーションを切り離す。		

関連リソース

リファレンス

- Apache Tomcat 7.0 ドキュメント
- Apache Tomcat 7.0 インストールガイド
- Apache Tomcat JNDI ドキュメント
- 「Amazon RDS マルチ AZ 配置」
- Amazon ElastiCache for Memcached

チュートリアルと動画

- 「Amazon RDS の開始方法」

.NET アプリケーションを Microsoft Azure App Serviceから AWS Elastic Beanstalk に移行

作成者：Raghavender Madamshitti (AWS)

概要

このパターンでは、Microsoft Azure App Service でホストされた .NET ウェブアプリケーションを AWS Elastic Beanstalk に移行する方法を説明します。アプリケーションを Elastic Beanstalk に移行するには、次の 2 つの方法があります。

- AWS Toolkit for Visual Studio を使用-Microsoft Visual Studio IDE 用のこのプラグインは、カスタム .NET アプリケーションを AWS にデプロイする最も簡単でわかりやすい方法を提供しています。この方法で、.NET コードを直接 AWS にデプロイし、SQL Server データベース用の Amazon Relational Database Service (Amazon RDS) などのサポートリソースを Visual Studio から直接作成できます。
- Elastic Beanstalk へのアップロードとデプロイ-各 Azure App Service には、メモリダンプやデプロイログのキャプチャ、設定パラメータの表示、デプロイパッケージへのアクセスに便利な Kudu というバックグラウンドサービスが含まれています。Kudu コンソールで Azure App Serviceのコンテンツにアクセスし、デプロイパッケージを抽出し、Elastic Beanstalk コンソールのアップロードとデプロイオプションを使用してパッケージを Elastic Beanstalk にアップロードできます。

このパターンは、2 番目の方法 (Kudu 経由で Elastic Beanstalk にアプリケーションをアップロードする) を説明しています。このパターンは、AWS Elastic Beanstalk、Amazon Virtual Private Cloud (Amazon VPC)、Amazon CloudWatch、Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling、Amazon Simple Storage Service (Amazon S3) と Amazon Route 53 を含む AWS サービスも使用します。

.NET ウェブアプリケーションは Amazon EC2 Auto Scaling Group で実行される AWS Elastic Beanstalk にデプロイされます。CPU 使用率などの Amazon CloudWatch メトリックスに基づき、スケーリングポリシーを設定できます。データベースの場合、アプリケーションとビジネス要件により、Multi-AZ 環境で Amazon RDS または Amazon DynamoDB を使用することができます。

前提条件と制限

前提条件

- アクティブな AWS アカウント

- Azure App Service で実行される .NET ウェブアプリケーション
- Azure App Service Kudu コンソールを使用する権限

製品バージョン

- .NET Core (x64) 1.0.1、2.0.0 以降、または .NET Framework 4.x、3.5 (「Windows Server の .NET プラットフォーム履歴」を参照)
- Windows Server 2012 以降で実行されるInternet Information Services (IIS) バージョン 8.0 以降
- .NET 2.0 または 4.0 ランタイム。

アーキテクチャ

ソーステクノロジースタック

- .NET Framework 3.5 以降、または.NET Core 1.0.1、2.0.0 以降を使用して開発され、Azure App Service (web app または API app) でホストされているアプリケーション

ターゲットテクノロジースタック

- Amazon EC2 Auto Scaling グループで実行される AWS Elastic Beanstalk

移行アーキテクチャ

デプロイのワークフロー

ツール

ツール

- .NET Core or .NET Framework
- C#
- IIS
- Kudu コンソール

AWS サービスと機能

- 「AWS Elastic Beanstalk」— Elastic Beanstalk は、.NET ウェブアプリケーションをデプロイ、スケーリングするための使いやすいサービスです。Elastic Beanstalk は、キャパシティのプロビジョニング、ロードバランシング、自動スケーリングを自動的に管理します。
- 「Amazon EC2 Auto Scaling グループ」 — Elastic Beanstalk には、環境内の Amazon EC2 インスタンスを管理する Auto Scaling グループが含まれています。単一インスタンス環境では、Auto Scaling グループは常に1 つのインスタンスが実行されているよう確認します。負荷分散される環境では、実行する範囲のグループを設定すると、Amazon EC2 Auto Scaling は、負荷に基づき、必要に応じてインスタンスを追加または削除します。
- 「Elastic Load Balancing」 — AWS Elastic Beanstalk でロードバランスを有効にすると、環境内の EC2 インスタンス間でトラフィックを分散するロードバランサーが作成されます。
- 「Amazon CloudWatch」 — Elastic Beanstalk は Amazon CloudWatch を自動的に使用して、アプリケーションや環境のリソースに関する情報を提供します。Amazon CloudWatch は、標準メトリックス、カスタムメトリックスおよびアラームをサポートしています。
- 「Amazon Route 53」— Amazon Route 53 は、可用性と拡張性の高いドメインネームシステム (DNS) のウェブサービスです。Route 53 エイリアスレコードで、カスタムドメイン名を AWS Elastic Beanstalk 環境にマッピングできます。

エピック

VPC をセットアップする

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を設定します。	AWS アカウントで、必要な情報により VPC を作成します。	システム管理者
サブネットを作成します。	VPC 内に 2 つのサブネットを作成します。	システム管理者
ルートテーブルを作成します。	ご要件に基づいてルートテーブルを作成します。	システム管理者

Elastic Beanstalk のセットアップ

タスク	説明	必要なスキル
Azure App Service Kudu コンソールにアクセスします。	Azure ポータルから Kudu にアクセスするには、App Service ダッシュボードに移動し、 Advanced Tools 、Go を選択します。または、以下のように Azure App Service URL を変更することもできます。`https://<appservicename>.scm.azurewebsites.net`	アプリ開発者、システム管理者
Kudu からデプロイパッケージをダウンロードします。	DebugConsole オプションを選択して Windows PowerShell に移動します。これにより、Kudo コンソールが開きます。`wwwroot` フォルダに移動してダウンロードします。これにより、Azure App Service のデプロイパッケージが zip ファイルとしてダウンロードされます。例としては、添付ファイルを参照してください。	アプリ開発者、システム管理者
Elastic Beanstalk 用のパッケージを作成します。	Azure App Service からダウンロードしたデプロイパッケージを解凍します。`aws-windows-deployment-manifest.json` という名前の JSON ファイルを作成します (このファイルは .NET Core アプリケーションにのみ必要です)。`aws-windo`	アプリ開発者、システム管理者

タスク	説明	必要なスキル
	`ws-deployment-manifest.json` と Azure App Service デプロイパッケージファイルを含む zip ファイルを作成します。例としては、添付ファイルを参照してください。	
新しい Elastic Beanstalk アプリケーションを作成します。	Elastic Beanstalk コンソールで開きます。既存のアプリケーションを選択するか、新しいアプリケーションを作成します。	アプリ開発者、システム管理者
環境を作成します。	Elastic Beanstalk コンソールの Actions メニューで、環境を作成するを選択します。ウェブサーバー環境と.NET/IIS プラットフォームを選択します。アプリケーションコードについては、[アップロード] を選択します。Elastic Beanstalk 用に準備した zip ファイルをアップロードし、環境を作成するを選択します。	アプリ開発者、システム管理者
Amazon CloudWatch を設定します。	デフォルトでは、基本的な CloudWatch モニタリングが有効です。設定を変更したい場合は、Elastic Beanstalk ウィザードで公開アプリケーションを選択し、次に Monitoring を選択します。	システム管理者

タスク	説明	必要なスキル
デプロイパッケージが Amazon S3 にあることを確認します。	アプリケーション環境が作成されると、S3 バケットにデプロイパッケージが表示されます。	アプリ開発者、システム管理者
アプリケーションをテストします。	環境が作成されたら、Elastic Beanstalk コンソールの中の URL で、アプリケーションをテストします。	システム管理者

関連リソース

- 「AWS Elastic Beanstack コンセプト」(Elastic Beanstalk のドキュメント)
- 「Elastic Beanstalk での .NET の開始方法」(Elastic Beanstalk のドキュメント)
- 「Kudu コンソール」(GitHub)
- 「[Kudu] を使って Azure Web Appsを管理する」(GS Lab の記事)
- 「カスタム ASP.NET Core Elastic Beanstalk デプロイ」(AWS Toolkit for Visual Studio ユーザーガイド)
- 「Elastic Load Balancing のドキュメント」
- 「AWS Elastic Beanstalk がサポートするプラットフォーム」(Elastic Beanstalk のドキュメント)
- 「AWS に Web Application をデプロイ」(C# Corner の記事)
- 「Auto Scaling Groupのサイズのスケーリング」(Amazon EC2 のドキュメント)
- 「Amazon RDS での高可用性 (Multi-AZ)」(Amazon RDS のドキュメント)

追加情報

Notes (メモ)

- オンプレミスまたは Azure SQL Server データベースを Amazon RDS に移行する場合は、データベース接続の詳細も更新する必要があります。
- テスト目的で、サンプルデモアプリケーションが添付されています。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon ECS で Oracle WebLogic から Apache Tomcat (ToMee) に移行する

作成者: Anya Epishcheva (AWS) と Harshad Gohil (AWS)

概要

このパターンでは、Oracle WebLogic を実行しているオンプレミスの Oracle Solaris SPARC システムを、Amazon Elastic Container Service (Amazon ECS) で Apache ToMee (コンテナサポートが追加された Apache Tomcat) を実行する Docker コンテナベースのインストールに移行する手順について説明します。

Oracle WebLogic から Tomcat に移行するアプリケーションに関連するデータベースを移行する方法については、このカタログのデータベース移行パターンを参照してください。

ベストプラクティス

Java および Java Enterprise Edition (Java EE) ウェブアプリケーションを移行する手順は、アプリケーションが使用するコンテナ固有のリソースの数によって異なります。Spring ベースのアプリケーションは、デプロイメントコンテナへの依存関係が少ないため、通常は移行が容易です。これとは対照的に、Enterprise JavaBeans (EJB) や、スレッドプール、Java 認証認可サービス (JAAS)、Container-Managed Persistence (CMP) などのマネージドコンテナリソースを使用する Java EE アプリケーションはより多くの労力を要します。

Oracle Application Server 向けに開発されたアプリケーションでは、Oracle Identity Management スイートがよく使用されます。オープンソースのアプリケーションサーバーに移行するお客様は、多くの場合、SAML ベースのフェデレーションを使用して ID 管理とアクセス管理を再実装することを選択します。また、Oracle Identity Management スイートからの移行が選択肢にない場合に Oracle HTTP Server Webgate を使用する企業もあります。

Java および Java EE ウェブアプリケーションは、AWS Fargate や Amazon ECS などの Docker ベースの AWS サービスにデプロイするのに最適です。お客様はしばしば、ターゲットアプリケーションサーバーの最新バージョン (TomEEなど) と Java 開発キット (JDK) がプリインストールされた Docker イメージを選択します。ベースの Docker イメージの上にアプリケーションをインストールし、Amazon Elastic Container Registry (Amazon ECR) レジストリに公開し、それを使用して AWS Fargate または Amazon ECS にアプリケーションをスケーラブルにデプロイします。

アプリケーションのデプロイには伸縮性があること、つまり、トラフィックやワークロードに応じてアプリケーションインスタンスの数をスケールインまたはスケールアウトできるのが理想的です。これは、需要に応じて容量を調整するためには、アプリケーションインスタンスをオンラインにするか終了する必要があることを意味します。

Java アプリケーションを AWS に移行するときは、ステートレスにすることを検討してください。これは、コンテナ化を使用して水平スケーリングを可能にする AWS Well-Architected フレームワーク の主要なアーキテクチャ原則です。たとえば、ほとんどの Java ベースのウェブアプリケーションはユーザーセッション情報をローカルに保存します。Amazon Elastic Compute Cloud (Amazon EC2) の自動スケーリングやその他の理由によるアプリケーションインスタンスの終了に備えて、ユーザーセッション情報をグローバルに保存する必要があるからです。これにより、ウェブアプリケーションユーザーは、ウェブアプリケーションに再接続したり再ログインしたりすることなく、シームレスかつ透過的に作業を続けることができます。このアプローチには、Amazon ElastiCache for Redis やグローバルデータベースへのセッション状態の保存など、いくつかのアーキテクチャオプションがあります。TomEE などのアプリケーションサーバーにはプラグインがあり、Redis、データベース、その他のグローバルデータストアを介してセッションの保存と管理を可能にします。

Amazon CloudWatch や AWS X-Ray と簡単に統合できる、共通の一元化されたログおよびデバッグツールを使用してください。移行は、アプリケーションのライフサイクル機能を向上させる機会となります。たとえば、継続的な統合や継続的なデリバリー (CI/CD) パイプラインを使用して変更を簡単に行えるように、ビルドプロセスを自動化したい場合です。そのためには、ダウンタイムなしでデプロイできるようにアプリケーションを変更する必要があるかもしれません。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- ソース Java コードと JDK
- Oracle WebLogic で構築されたソースアプリケーション
- アイデンティティ管理とアクセス管理のための定義済みソリューション (SAML または Oracle Webgate)
- アプリケーションセッション管理のための定義済みソリューション (Amazon ElastiCache や同等の方法を使用するか、または必要に応じてアプリケーションをステートレスにする)
- チームが Apache TomEE に移植できるように J2EE 固有のライブラリをリファクタリングする必要があるかどうかの理解 (Apache ウェブサイトの「Java EE 7 実装状況」を参照)
- セキュリティ要件に基づいて強化された TomEE イメージ
- ターゲット TomEE がプリインストールされたコンテナイメージ
- アプリケーションの修正 (ログ、デバッグビルド、認証など) が合意され、必要に応じて実施されている

製品バージョン

- Oracle WebLogic OC4J、9i、10g
- Tomcat 7 (Java 1.6 以降を搭載)

アーキテクチャ

ソーステクノロジースタック

- Oracle WebLogic を使用して構築されたウェブアプリケーション
- Oracle Webgate または SAML 認証を使用するウェブアプリケーション
- Oracle Database バージョン 10g 以降に接続されているウェブアプリケーション

ターゲットテクノロジースタック

- Amazon ECS 上で実行されている TomEE (コンテナサポートが追加された Apache Tomcat) (「Deploying Java Web Applications」および 「Java Microservices on Amazon ECS」も参照してください)
- Oracle 用の Amazon Relational Database Service (Amazon RDS)。Amazon RDS でサポートされている Oracle バージョンについては、Amazon RDS for Oracle を参照してください

ターゲットアーキテクチャ

ツール

TomEE で動作させるには、Java アプリケーションを .war ファイルに再構築する必要があります。TomEE 上でアプリケーションを操作するためにアプリケーションの変更が必要な場合があります。必要な構成オプションと環境プロパティが正しく定義されていることを確認してください。

また、Java Naming and Directory Interface (JNDI) ルックアップと JavaServer ページ (JSP) 名前空間が正しく定義されている必要があります。組み込みの T ライブラリとの名前の衝突を避けるため、アプリケーションが使用するファイル名を確認することを検討してください。たとえば、persistence.xml は Apache OpenJPA フレームワーク (TomEE では OpenEJB にバンドルされている) で構成目的で使用されるファイル名です。PUI の persistence.xml ファイルには Spring フレームワークの Bean 宣言が含まれています。

TomEE バージョン 7.0.3 以降 (Tomcat 8.5.7 以降) は、特殊文字を含む未加工の (エンコードされていない) URL に対して HTTP 400 レスポンス (不正リクエスト) を返します。サーバーからの応答は、エンドユーザーには空白ページとして表示されます。旧バージョンの TomEE と Tomcat では、URL にエンコードされていない一部の特殊文字を使用できました。ただし、CVE-2016-6816 ウェブサイトに記載されているように、安全ではないと見なされています。URL エンコーディングの問題を解決するには、JavaScript を介してブラウザに直接渡される URL を、未加工の文字列として使用するのではなく encodeURI() メソッドでエンコードする必要があります。

.war ファイルを TomEE にデプロイした後、Linux cat の起動ログをモニタリングして、見つからない共有ライブラリや Oracle 固有の拡張機能がないかを確認し、Tomcat ライブラリから不足しているコンポーネントを追加してください。

一般的な手順

- TomEE 上でアプリケーションを構成します。
- アプリケーションサーバー固有の構成ファイルとリソースを特定し、ソースからターゲット形式まで再構成します。
- JNDI リソースを特定して再構成します。
- EJB 名前空間とルックアップを、ターゲットアプリケーションサーバーが必要とする形式に調整します (該当する場合)。
- JAAS アプリケーションコンテナ固有のセキュリティロールとプリンシパルマッピング (該当する場合) を再構成します。
- アプリケーションと共有ライブラリを.war ファイルにパッケージ化します。
- 指定の Docker コンテナを使用して、.war ファイルを TomEE にデプロイします。
- 開始ログをモニタリングして、見つからない共有ライブラリとデプロイメント記述子の拡張子がないか確認します。見つかった場合は、最初のタスクに戻ってください。
- インストールしたアプリケーションを、復元された Amazon RDS データベースと照合してテストします。
- 「Docker コンテナのデプロイ」の手順に従って、ロードバランサーと Amazon ECS クラスターを含むアーキテクチャ全体を起動します。
- ロードバランサーを指すように URL を更新します。
- 構成管理データベース (CMDB) を更新します。

エピック

移行を計画する

タスク	説明	必要なスキル
アプリケーションの検出 (現在の状態フットプリントとパフォーマンスベースライン) を行います。		BA、移行リーダー
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		DBA
ソースとターゲットのアプリケーション設計 (ID とセッション管理) を検証します。		DBA、移行エンジニア、アプリ所有者
ターゲットサーバーインスタンスのハードウェア要件とストレージ要件を特定します。		DBA、SysAdmin
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、SysAdmin
ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、SysAdmin
アプリケーション移行戦略とツールを特定します。		DBA、移行リーダー
アプリケーションの移行設計と移行ガイドを完成させます。		ビルドリード、移行リード

タスク	説明	必要なスキル
アプリケーション移行ランブックを完成させます。		ビルドリード、カットオーバーリード、テストリード、移行リード

インフラストラクチャを構成する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成		SysAdmin
セキュリティグループを作成します。		SysAdmin
Amazon RDS DB インスタンスを構成および起動します。		DBA、SysAdmin
Amazon ECS デプロイを構成します。		SysAdmin
アプリケーションを Docker イメージとしてパッケージ化します。		SysAdmin
イメージを Amazon ECR レジストリにプッシュする (または、このステップをスキップして Amazon ECS クラスターにプッシュする)。		SysAdmin
アプリケーションと Amazon ECS サービスのオプションのタスク定義を構成します。		SysAdmin
クラスターを構成し、セキュリティ構成を確認		SysAdmin

タスク	説明	必要なスキル
し、AWS Identity and Access Management (IAM) ロールを構成します。		
セットアップを起動し、アプリケーション移行ランブックに従ってテストを実行します。		SysAdmin

データを移行する

タスク	説明	必要なスキル
セキュリティ保証チームの許可を得て、本番データを AWS に移行します。		DBA、移行エンジニア、アプリ所有者
エンドポイントを作成してアクセスし、データベースのバックアップファイルを取得します。		DBA
ネイティブ データベースエンジンまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。		DBA
アプリケーション移行ランブックから必要なテストを実行して、データ移行が成功したことを確認します。		DBA、移行エンジニア、アプリ所有者

アプリケーションを移行する

タスク	説明	必要なスキル
移行用の変更リクエスト (CR) を作成します。		カットオーバーリード
移行のためのCR 承認を得ます。		カットオーバーリード
アプリケーション移行ランブックに記載されているアプリケーション移行戦略に従います。		DBA、移行エンジニア、アプリ所有者
アプリケーションをアップグレードします (必要な場合)。		DBA、移行エンジニア、アプリ所有者
機能テスト、非機能テスト、データ検証、SLA、パフォーマンステストを完了します。		テストリード、アプリオーナー、アプリユーザー

カットオーバー

タスク	説明	必要なスキル
アプリ所有者またはビジネスオーナーから承認を得ます。		カットオーバーリード
テーブルトピックの演習を実施して、カットオーバーランブックのすべてのステップを順を追って説明します。		DBA、移行エンジニア、アプリ所有者
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、移行エンジニア、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		DBA、移行エンジニア、SysAdmin
プロジェクト文書を確認して検証する。		移行リード
移行の所要時間、手動とツールの比率、コスト削減などのメトリクスを収集します。		移行リード
プロジェクトを終了し、フィードバックを提供します。		移行リーダー、アプリ所有者

関連リソース

リファレンス

- Apache Tomcat 7.0 ドキュメント
- Apache Tomcat 7.0 インストールガイド
- Apache Tomcat JNDI ドキュメント
- Apache TomE ドキュメント
- 「Amazon RDS for Oracle」
- Amazon RDS の料金
- Oracleと
- Amazon RDS に関するOracleのドキュメンテト
- 「Amazon RDS マルチ AZ 配置」
- Amazon ECS の開始
- Amazon RDS の開始

チュートリアルと動画

- Best Practices for Running Oracle Databases on Amazon RDS (re:Invent 2018 presentation)

AWS DMS を使用して Oracle データベースを Amazon EC2 から Amazon RDS for Oracle に移行する

作成者: Chethan Gangadharaiah (AWS) と Brian motzer (AWS)

概要

このパターンは、AWS Database Migration Service (AWS DMS) を使用して、Amazon Elastic Compute Cloud (Amazon EC2) 上の Oracle データベースを Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行する手順を示しています。このパターンでは Oracle SQL Developer や SQL *Plus を使用して Oracle DB インスタンスに接続し、一部のタスクを自動化する AWS CloudFormation テンプレートも含まれています。

Amazon RDS for Oracle に移行すると、Amazon RDS がデータベースのプロビジョニング、バックアップとリカバリ、セキュリティパッチ、バージョンアップグレード、ストレージ管理などのデータベース管理タスクを引き受けながら、ビジネスとアプリケーションに集中できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon EC2 の Oracle Database 用の Amazon マシンイメージ (AMI)

製品バージョン

- AWS DMS は、Enterprise、Standard、Standard One、および Standard Two エディションの Amazon RDS インスタンスデータベースの Oracle バージョン 11g (バージョン 11.2.0.3.v1 以降)、12c、および 18c をサポートします。サポートされているバージョンに関する最新情報については、AWS ドキュメントの「Using an Oracle Database as a Target for AWS DMS」を参照してください。(添付されている AWS CloudFormation テンプレートは Oracle バージョン 12c をソースデータベースとして使用しています。)
- Oracle SQL Developer 4.0.3

アーキテクチャ

ソースアーキテクチャ

- Oracle Database on Amazon EC2

ターゲットアーキテクチャ

- 「Amazon RDS for Oracle」

移行アーキテクチャ

ツール

- AWS DMS – AWS Database Migration Service (AWS DMS) は、データベースを AWS に迅速かつ安全に移行するのに役立ちます。同種移行と異種移行の両方をサポートします。サポートされている Oracle データベースのバージョンとエディションについては、AWS ドキュメントの「Using an Oracle Database as a Source for AWS DMS」および「Using an Oracle Database as a Target for AWS DMS」を参照してください。
- Oracle SQL Developer または SQL *Plus – これらのツールを使用すると、Amazon RDS for Oracle DB インスタンスに接続できます。

エピック

ターゲットデータベースのセットアップ

タスク	説明	必要なスキル
Amazon RDS for Oracle DB インスタンスを作成します。	AWS マネジメントコンソールにサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。Oracle データベースの適切なエンジン、テンプレート、データベース認証情報設定、インスタンスタイプ、ストレージ、マルチ AZ 設定、仮想プライベートクラウド (VPC) と構成、ログイン認証情報、その他の設定を選択して Oracle DB インスタンスを作成します。手順に	開発者

タスク	説明	必要なスキル
	ついては、「関連リソース」セクションのリンクを参照してください。または、添付ファイルの AWS CloudFormation テンプレート (Create_RDS.yaml) を使用して Amazon RDS for Oracle DB インスタンスを作成します。	
Amazon RDS Connect し、Oracle ユーザーに権限を付与します。	セキュリティグループを変更して、ローカルマシンと AWS DMS レプリケーションインスタンスから接続するための適切なポートを開きます。接続を構成するときは、VPC の外部からデータベースに接続できるように、「パブリックにアクセス可能」オプションが選択されていることを確認してください。ログイン認証情報を使用して Oracle SQL Developer または SQL *Plus で Amazon RDS Connect し、AWS DMS ユーザーを作成し、データベースを変更するために必要な権限を AWS DMS ユーザーに付与します。	開発者

ソース EC2 インスタンスのセキュリティグループを設定します。

タスク	説明	必要なスキル
Oracle データベースが稼働しているかどうかを確認してください。	Secure Shell (SSH) を使用して EC2 インスタンスに接続し、SQL *Plus を使用して Oracle データベースに接続してみます。	開発者
セキュリティグループを変更します。	EC2 インスタンスのセキュリティグループを変更して適切なポートを開き、ローカルマシンと AWS DMS レプリケーションインスタンスから接続できるようにします。	開発者

AWS DMS のセットアップ

タスク	説明	必要なスキル
AWS DMS レプリケーションインスタンスを作成します。	AWS DMS では、Amazon RDS for Oracle DB インスタンスと同じ VPC にレプリケーションインスタンスを作成します。レプリケーションインスタンスの名前と説明を指定し、インスタンスクラスとレプリケーションエンジンのバージョンを選択し (デフォルトを使用)、Amazon RDS DB インスタンスを作成した VPC を選択し、必要に応じてマルチ AZ 設定を設定し、ストレージを割り当て、アベイラビリティゾーンを指定	DBA

タスク	説明	必要なスキル
	し、追加設定を行います。または、添付ファイルの AWS CloudFormation テンプレート (DMS.yaml) を使用してこのステップを実装することもできます。	
ソースおよびターゲットデータベースエンドポイントに接続します。	エンドポイント ID、エンジン、サーバー、ポート、ログイン認証情報、その他の接続属性を指定して、ソースデータベースエンドポイントとターゲットデータベースエンドポイントを作成します。ソースサーバーには、Oracle データベースをホストしている EC2 インスタンスのパブリック DNS を使用します。ターゲットサーバーには、Amazon RDS for Oracle のエンドポイントを使用してください。テストを実行して、ソース接続とターゲット接続が機能していることを確認します。または、添付ファイルの AWS CloudFormation テンプレート (DMS.yaml) を使用してこのステップを実装することもできます。	DBA

タスク	説明	必要なスキル
AWS DMS タスクを作成します。	AWS DMS タスクを作成して、ソースエンドポイントからターゲットエンドポイントにデータを移行するか、ソースエンドポイントとターゲットエンドポイント間のレプリケーションをセットアップするか、あるいはその両方を行います。AWS DMS タスクを作成するときは、レプリケーションインスタンス、ソースエンドポイント、ターゲットエンドポイント、移行タイプ (データのみ、レプリケーションのみ、または両方)、テーブルマッピング、およびフィルタを指定します。AWS DMS タスクを実行し、タスクをモニタリングし、テーブル統計を確認し、Amazon CloudWatch でログをチェックします。または、添付ファイルの AWS CloudFormation テンプレート (DMS.yaml) を使用してこのステップを実装することもできます。	DBA

関連リソース

- Amazon RDS DB インスタンスの作成
- Oracle データベースエンジンを実行している DB インスタンスへの接続
- AWS DMS のドキュメント
- AWS DMS Step-by-Step Walkthroughs

- 「AWS クラウドへの Oracle データベースの移行」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Logstash を使用して、オンプレミス Oracle データベースを Amazon OpenSearch Service へ移行する

作成者: Aditya Goteti (AWS)

概要

このパターンは、Logstash を使用して、オンプレミス Oracle データベースから Amazon OpenSearch Service へデータを移動する方法について説明します。アーキテクチャ上の考慮事項、必要なスキルセットと推奨事項が含まれます。データは単一テーブル、または全文検索を実行する必要がある複数テーブルからのものです。

OpenSearch サービスは、仮想プライベートクラウド (VPC) 内で設定、または IP ベースの制限で公開できます。このパターンは、OpenSearch サービスが VPC 内で設定されるシナリオを説明します。Logstash は Oracle データベースからデータを収集して JSON 形式に構文解析し、そのデータを OpenSearch サービスへのフィードに使用されます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Java 8 (Logstash 6.4.3 で必要)
- AWS 仮想プライベートネットワーク (AWS VPN) を使用して確立される、オンプレミスデータベースサーバーと VPC 内の Amazon Elastic Compute Cloud (Amazon EC2) インスタンス間の接続性
- データベースから OpenSearch Service にプッシュされる必要なデータを取得するクエリ
- Java Database Connectivity (JDBC) ドライバー

制限

- Logstash はデータベースから物理削除されるレコードは識別できない

製品バージョン

- Oracle Database 12c
- OpenSearch Service 6.3
- Logstash 6.4.3

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース
- オンプレミス AWS VPN

ターゲットテクノロジースタック

- VPC
- EC2 インスタンス
- OpenSearch Service
- Logstash
- NAT ゲートウェイ (EC2 インスタンスのオペレーティングシステムアップデート、Java 8、Logstash、プラグインのインストール用)

データ移行アーキテクチャ

ツール

- Logstash 6.4.3
- JDBC 入力プラグイン (ダウンロードおよび詳細情報)
- ログスタッシュ出力プラグイン (logstash-output-amazon_es)
- Oracle JDBC ドライバー

エピック

移行を計画する

タスク	説明	必要なスキル
ソースデータベースのサイズを特定します。	ソースデータのサイズは、インデックスに設定するシャードの数の決定に使用するパラメータの 1 つです。	DBA、データベース開発者

タスク	説明	必要なスキル
各列のデータ型と対応するデータを分析します。	OpenSearch Service は、これまで見えていなかったフィールドがドキュメントに見つかると、データ型を動的にマッピングします。明示的に宣言する必要のある特定のデータ型または形式 (日付フィールドなど) がある場合は、インデックス作成時にフィールドを特定し、そのフィールドのマッピングを定義します。	アプリ所有者、開発者、データベース開発者
プライマリーキーまたは一意キーがある列があるかどうかを確認します。	更新または挿入時に Amazon OpenSearch Service のレコードが重複しないようにするには、`amazon_es` プラグインの出力セクションで `document_id` 設定をする必要があります (たとえば、`customer_id` がプライマリキーの `document_id => "%{customer_id}"`)。	アプリ所有者、開発者

タスク	説明	必要なスキル
追加された新しいレコードの数と頻度を分析し、レコードがどのような頻度で削除されるかを確認します。	このタスクは、ソースデータの増加率を把握するために必要です。データの読み取り量が多く、挿入がまれな場合は、1 つのインデックスにできます。新しいレコードが頻繁に挿入され、削除されない場合、シャードサイズは推奨最大サイズの 50 GB を簡単に超える可能性があります。この場合は、Logstash と エイリアスを使用してアクセスできるコードでインデックスパターンを設定することで、インデックスを動的に作成できます。	アプリ所有者、開発者
必要なレプリカの数を決定します。		アプリ所有者、開発者
インデックスに設定するシャード数を決定します。		アプリ所有者、開発者
専用マスターノード、データノード、EC2 インスタンスのインスタンスタイプを特定します。	詳細については、関連リソースセクションを参照してください。	アプリ所有者、開発者
必要な専用マスターノードとデータノード数を決定します。	詳細については、関連リソースセクションを参照してください。	

データを移行する

タスク	説明	必要なスキル
EC2 インスタンスを起動します。	AWS VPN が接続されている VPC 内で EC2 インスタンスを起動します。	Amazon VPC コンストラクト、AWS VPN
EC2 インスタンスに Logstash をインストールします。		開発者
Logstash プラグインをインストールします。	必要な Logstash プラグイン `jdbc-input` と `logstash-output-amazon_es` をインストールします。	開発者
Logstash を設定します。	Logstashキーストアを作成して、AWS Secrets Manager キー、データベース認証情報などの機密情報を保存し、そのレファレンスを Logstash 設定ファイルに置きます。	開発者
デッドレターキューと永続キューを設定します。	デフォルトで、データにマッピングエラーまたはその他の問題が含まれるために処理できないイベントが Logstash で遭遇すると、Logstash パイプラインは失敗したイベントをハングまたはドロップします。このような状況でデータ損失を防ぐため、失敗したイベントをドロップせずにデッドレターキューに書き込むように Logstash を設定できます。異常終了時のデータ損失に対して保護するため	開発者

タスク	説明	必要なスキル
	に、Logstash にはメッセージキューをディスクに保存する永続キュー機能があります。永続キューは Logstash のデータに永続性を提供します。	
Amazon OpenSearch Service ドメインを作成します。	AWS Identity and Access Management (IAM) 認証情報でリクエストに署名する必要がないアクセスポリシーで Amazon OpenSearch Service ドメインを作成します。Amazon OpenSearch Service ドメインは、同じ VPC 内に作成する必要があります。また、分析に基づいてインスタンスタイプを選択し、専用ノードとマスターノードの数を設定する必要があります。	開発者
必要な Amazon OpenSearch Service ログを設定します。	詳細については、OpenSearch サービスドキュメントを参照してください。	
インデックスを作成します。		開発者

タスク	説明	必要なスキル
Logstash を起動します。	Logstash をバックグラウンドサービスとして実行します。Logstash は設定された SQL クエリを実行し、データを取得して JSON 形式に変換し、OpenSearch サービスに送ります。初回ロードの場合は、Logstash 設定ファイルでスケジューラーを設定しないでください。	開発者
ドキュメントを確認します。	インデックスの文書数と、すべての文書がソースデータベースに存在するかどうかを確認します。初回ロード時にインデックスに追加され、Logstash の停止に使用されます。 Logstashの設定を変更して、クライアントの要件に応じて一定の間隔で実行されるスケジューラーを追加し、Logstash を再起動します。Logstash は前回の実行後に更新または追加されたレコードのみを選択し、最終実行タイムスタンプは Logstash 設定ファイルの `last_run_metadata_path => “/usr/share/logstash/.logstash_jdbc_last_run”` プロパティで設定したファイルに保存されます。	開発者

関連リソース

- 推奨される CloudWatch アラーム
- Amazon OpenSearch Service の専用マスターノード
- Amazon OpenSearch Service ドメインのサイジング
- Logstash ドキュメント
- JDBC 入力プラグイン
- Logstash 出力プラグイン
- Amazon OpenSearch Service

オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する

作成者: Baji Shaik (AWS)、Pavan Pusuluri (AWS)

概要

このパターンでは、オンプレミスの Oracle データベースを Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する際の手順を説明しています。　移行プロセスの一環として、移行計画を作成し、ソースデータベースに基づいてターゲットデータベースのインフラストラクチャに関する重要な要素を検討します。　ビジネス要件とユースケースに基づいて、次の 2 つの移行オプションから 1 つ選択できます。

- AWS Database Migration Service (AWS DMS) – AWS DMS を使用すると、データベースを迅速かつセキュアに AWS クラウドに移行することができます。移行中でもソースデータベースが完全に維持され、このデータベースを利用するアプリケーションのダウンタイムは最小限に抑えられます。変更データキャプチャ (CDC) と呼ばれるプロセスを通じて最初の全ロード移行を完了した後に、AWS DMS を使用して進行中の変更をキャプチャするタスクを作成することで、移行時間を短縮できます。
- ネイティブ Oracle ツール — Oracle GoldenGate for CDC　を使用した Data Pump エクスポートおよび Data Pump インポートなどのネイティブ Oracle ツールを使用してデータベースを移行できます。オリジナルのエクスポートユーティリティや、オリジナルのインポートユーティリティなどの Oracle ネイティブツールを使用して、全ロード時間を短縮することもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスの Oracle データベース
- Amazon RDS Oracle データベース (DB) インスタンス

制限

- データベースサイズの上限: 64 TB

製品バージョン

- Oracle バージョン 11g (バージョン 11.2.0.3.v1 以降) および 12.2 および 18c。サポートされているバージョンとエディションの最新のリストについては、AWS ドキュメントの「Amazon RDS for Oracle」を参照してください。サポートされているバージョンの最新リストについては、AWS DMS ドキュメントの「AWS DMS のソースとして Oracle データベースを使用する」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース

ターゲットテクノロジースタック

- 「Amazon RDS for Oracle」

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、AWS DMS を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する方法を示しています。

この図表は、次のワークフローを示しています：

1. 既存のデータベースユーザーを作成または使用し、そのユーザーに必要な AWS DMS アクセス権限を付与し、ARCHIVELOG モードをオンにして、補足のログ記録を設定します。
2. オンプレミスと AWS ネットワーク間のインターネットゲートウェイを設定します。
3. AWS DMS でソースとターゲットのエンドポイント を設定します。
4. AWS DMS レプリケーションタスクを設定して、ソースデータベースからターゲットデータベースにデータを移行します。
5. 移行後のアクティビティをターゲットデータベースで実行します。

次の図は、ネイティブ Oracle ツールを使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する方法を示しています。

この図表は、次のワークフローを示しています：

1. Oracle Export (exp) および Import (imp) ユーティリティを使用して Oracle データベースユーザーを作成または使用し、Oracle データベースをバックアップするために必要なアクセス権限を付与します。
2. オンプレミスと AWS ネットワーク間のインターネットゲートウェイを設定します。
3. バックアップデータベースを使用するように 踏み台ホストの Oracle クライアントを設定します。
4. Amazon Simple Storage Service (Amazon S3) バケットにバックアップデータベースをアップロードします。
5. データベースバックアップを Amazon S3 から Amazon RDS for Oracle データベースに復元します。
6. Oracle GoldenGate for CDC を設定します。
7. 移行後のアクティビティをターゲットデータベースで実行します。

ツール

- 「AWS Database Migration Service (AWS DMS)」 を使用して、データストアを AWS クラウドへ、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間に移行します。
- Oracle ネイティブツールは、同種移行の実行に役立ちます。Oracle Data Pump を使用して、ソースデータベースとターゲットデータベース間でデータを移行できます。このパターンでは、Oracle Data Pump を使用して、ソースデータベースからターゲットデータベースへのフルロードを実行します。
- Oracle GoldenGate を使用すると、2 つ以上のデータベース間で論理レプリケーションを実行できます。　このパターンでは、GoldenGate を使用して、初回ロード後に Oracle Data Pump を使用して変更の差分を複製します。

エピック

移行を計画する

タスク	説明	必要なスキル
プロジェクト文書を作成し、データベースの詳細を記録します。	1. 移行目標、移行要件、プロジェクトの主な利害関係者、プロジェクトのマイル	DBA

タスク	説明	必要なスキル
	ストーン、プロジェクトの期限、主要メトリクス、移行リスク、リスク軽減計画を文書化します。 2. RAM、IOPS、CPU など、ソースデータベースに関する重要な情報を文書化します。後にこの情報を使用して、適切なターゲット DB インスタンスを決定します。 3. ソースデータベースとターゲットデータベースのバージョンを検証します。	

タスク	説明	必要なスキル
ストレージ要件を特定します。	以下の項目を含むストレージ要件を特定して文書化します。 1. ソース DB インスタンスに割り当てられたストレージを計算します。 2. ソース DB インスタンスの過去からの増加メトリクスを収集します。 3. ターゲット DB インスタンスの将来の増加を予測します。 **Note** 汎用 (gp2) SSD ボリュームの場合、ストレージ 1 GB あたり 3 IOPS を取得します。ソースデータベースの読み取りおよび書き込み IOPS の合計数を計算することで、ストレージを割り当てます。	DBA、SysAdmin

タスク	説明	必要なスキル
コンピューティング要件に基づいて適切なインスタンスタイプを選択してください。	1. ターゲット DB インスタンスのコンピューティング要件を決定します。 2. パフォーマンスの問題を特定します。 3. 以下の要因を考慮して、適切なインスタンスタイプを決定してください。 • ソース DB インスタンスの CPU 使用率 • ソース DB インスタンスの IOPS (読み取り/書き込み操作) • ソース DB iインスタンスのメモリフットプリント	SysAdmin
ネットワークアクセスのセキュリティ要件を特定します。	1. ソースおよびターゲットのデータベースのネットワークおよびホストアクセスのセキュリティ要件を特定して文書化します。 2. アプリケーションがデータベースと通信できるように、適切なセキュリティグループを設定します。	DBA、SysAdmin
アプリケーション移行戦略を特定します。	1. 移行カットオーバー戦略を決定し、文書化します。 2. アプリケーションの目標復旧時間 (RTO) と目標復旧時点 (RPO) を決定して文書化し、それに応じてカットオーバーを計画します。	DBA、SysAdmin、アプリ所有者

タスク	説明	必要なスキル
移行リスクを特定します。	データベースを評価し、移行に特有のリスクと軽減策を文書化します。以下に例を示します。 • ログを記録しないテーブルを特定し、リカバリ時にデータが損失するリスクを強調します。 • ソースデータベースのユーザーと権限を抽出し、Amazon RDS の権限との競合を明確にします。 • アラートログに Oracle 固有のエラーや警告がないか確認します。 • ターゲット DB インスタンスでサポートされている機能とサポートされていない機能を特定します。 • ターゲット DB バージョンエンジンの廃止された機能を確認してください。	DBA

インフラストラクチャを設定する

タスク	説明	必要なスキル
VPC を作成します。	ターゲットデータベースインスタンス用の新しい Amazon Virtual Private Cloud (Amazon VPC) を作成 します。	SysAdmin

タスク	説明	必要なスキル
セキュリティグループを作成します。	新しい VPC にセキュリティグループを作成して、DB インスタンスへのインバウンド接続を許可します。	SysAdmin
Amazon RDS for Oracle DB インスタンスを作成します。	新しい VPC とセキュリティグループを使用してターゲットデータベースインスタンスを作成してから、ターゲット DB インスタンスを作成します。	SysAdmin

オプション 1 - ネイティブの Oracle またはサードパーティーのツールを使用してデータを移行する

タスク	説明	必要なスキル
ソースデータベースを準備します。	1. Data Pump ディレクトリを作成するか、既存のディレクトリを使用します。 2. 移行ユーザーを作成し、Data Pump 抽出を実行する権限を付与します。 3. ソースデータベースから SQL スクリプトとしてロール、ユーザー、テーブルスペースを抽出します。 4. 抽出した Data Pump ダンプをターゲット DB インスタンス `data pump` ディレクトリに転送します	DBA、SysAdmin
ターゲットデータベースを準備します。	1. すべてのデータベースオプション (テキストや Java など) がターゲット Amazon RDS for Oracle DB インス	DBA、SysAdmin

タスク	説明	必要なスキル
	タンスにインストールされているか、有効になっていることを確認します。 2. Data Pump ディレクトリを作成するか、既存のディレクトリを使用します。 3. 移行ユーザーを作成し、Data Pump インポートを実行する権限を付与します。 4. ターゲット DB インスタンスに必要なテーブルスペース、ユーザー、ロールを作成します。 5. 転送された Data Pump エクスポートダンプをターゲットデータベースにインポートします。 6. インポート時またはオブジェクト作成時に除外されたインデックスをすべて作成します。 7. インポート時に除外された制約をすべて作成します。 8. 無効なオブジェクトを検証または再コンパイルします。 9. 無効なインデックスを再構築します。 10.ソースデータベースとターゲットデータベースのデータベースオブジェクト数を検証します。	

タスク	説明	必要なスキル
	11.オブジェクト数の間に不一致が見つかった場合は解決します。	

オプション 2 - AWS DMS を使用してデータを移行する

タスク	説明	必要なスキル
データを準備する	1. ソースデータベースのデータをクリーンアップします。 2. レプリケーションインスタンスを作成します。。 3. ソースエンドポイントとターゲットエンドポイントを作成します。 4. 移行できるテーブルとオブジェクトの数を特定します。	DBA
データを移行します。	1. 外部キーの制約とトリガーをターゲットデータベースにドロップします。 2. ターゲットデータベースにセカンダリインデックスをドロップします。 3. ソースデータベースからターゲットデータベースへの AWS DMS フルロードタスクの設定を行います。 4. 外部キーを有効にします。	DBA

タスク	説明	必要なスキル
	5. AWS DMS CDC を有効にして、進行中の変更を複製できるようにします。 6. トリガーを有効にします。 7. シーケンスを更新します。 8. ソースデータとターゲットデータを検証します。	

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。	1. Oracle へのアプリケーションサービスとクライアント接続をすべて停止します。 2. AWS DMS タスクを実行します。 3. ロールバックタスクを設定します (例えば、Amazon RDS データベースからオンプレミスの Oracle データベースに CDC を逆転させる)。 4. データを検証します。 5. 新しい Amazon RDS for Oracle DB インスタンスに Amazon Route 53 を設定して、新しいターゲットデータベースでアプリケーションサービスを開始します。 6. 新しい Amazon RDS for Oracle DB インスタンスに	DBA、SysAdmin、アプリ所有者

タスク	説明	必要なスキル
	Amazon CloudWatch モニタリングを追加します。	
ロールバックプランを実装します。	1. Amazon RDS for Oracle DB インスタンスを指しているすべてのアプリケーションサービスを停止します。 2. AWS DMS タスクを使用して、変更をソースのオンプレミス Oracle データベースにロールバックします。 3. オンプレミスの Oracle データベースから Amazon RDS for Oracle データベースで実行されている AWS DMS タスクを停止します。 4. アプリケーションをソース Oracle データベースで設定します。 5. ロールバックデプロイが完了していることを確認します。	DBA、アプリ所有者

移行プロジェクトを閉じる

タスク	説明	必要なスキル
リソースをクリーンアップします。	AWS DMS レプリケーションインスタンスや S3 バケットなどの一時的な AWS リソースをシャットダウンまたは削除します。	DBA、SysAdmin

タスク	説明	必要なスキル
プロジェクト文書を確認します。	移行計画文書と目標を確認し、必要な移行手順がすべて完了したことを確認します。	DBA、SysAdmin、アプリ所有者
メトリクスを収集します。	移行が完了するまでにかかった時間、手動タスクとツールベースのタスクとの割合、コストの削減、その他の関連メトリクスなど、移行に関する主要なメトリクスを記録します。	DBA、SysAdmin、アプリ所有者
プロジェクトを終了します。	移行プロジェクトを終了し、その労力に関するフィードバックを集めます。	DBA、SysAdmin、アプリ所有者

関連リソース

リファレンス

- Oracle データベースの AWS クラウドへの移行 (AWS 規範ガイダンス)
- 「AWS Database Migration Service」 (AWS DMS ドキュメント)
- 「Amazon RDS 料金表」 (Amazon RDS ドキュメント)

チュートリアルと動画

- AWS Database Migration Service の使用開始 (AWS DMSドキュメント)
- 「Amazon RDS リソース」 (Amazon RDS ドキュメント)
- 「AWS Database Migration Service (DMS)」 (YouTube)

Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する

作成者: Mohan Annam (AWS)、Brian motzer (AWS)

概要

このパターンでは、Oracle Data Pump を使用して、Oracle データベースをオンプレミスのデータセンターから Amazon Relational Database Service (Amazon RDS) for Oracle DB インスタンスに移行する方法について説明します。

このパターンでは、ソースデータベースからデータダンプ ファイルを作成し、そのファイルを Amazon Simple Storage Service (Amazon S3) バケットに保存してから、Amazon RDS for Oracle DB インスタンスにデータを復元します。このパターンは、移行に AWS Database Migration Service (AWS DMS) を使用すると制限が発生する場合に便利です。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Identity and Access Management (IAM) でのロールの作成と Amazon S3 マルチパートアップロードに必要な権限
- ソースデータベースからデータをエクスポートするのに必要な権限
- AWS コマンドラインインターフェイス (AWS CLI) をインストール済みおよび設定済み

製品バージョン

- Oracle Data Pump は、Oracle Database 10g リリース 1 (10.1) 以降のバージョンでのみ使用できます。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの Oracle データベース

ターゲットテクノロジースタック

- 「Amazon RDS for Oracle」
- SQL クライアント (Oracle SQL Developer)
- S3 バケット

ソースアーキテクチャとターゲットアーキテクチャ

ツール

AWS サービス

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。このパターンでは、IAM を使用して Amazon S3 から Amazon RDS for Oracle にデータを移行するために必要なロールとポリシーを作成します。
- 「OracleのAmazon Relational Database Service (Amazon RDS)」 によって、AWS クラウドで Oracleリレーショナルデータベースをセットアップ、運用、スケーリングができます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- Oracle Data Pump を使用すると、あるデータベースから別のデータベースにデータやメタデータを高速に移動できます。このパターンでは、Oracle Data Pump を使用してデータダンプ (.dmp) ファイルを Oracle サーバーにエクスポートし、データダンプ ファイルを Amazon RDS for Oracle にインポートします。詳細については、Amazon RDS ドキュメントの「Amazon RDS の Oracle にデータをインポートする」を参照してください。
- Oracle SQL Developer は、従来のデプロイとクラウドデプロイの両方で Oracle Database の開発と管理を簡素化する統合開発環境です。オンプレミスの Oracle データベースと Amazon RDS for Oracle の両方と相互作用して、データのエクスポートとインポートに必要な SQL コマンドを実行します。

エピック

S3 バケットを作成する

タスク	説明	必要なスキル
バケットを作成します。	S3 バケットを作成するには、AWS のドキュメントの手順に従います。	AWS システム管理者

IAM ロールを作成してポリシーを割り当てる

タスク	説明	必要なスキル
IAM 許可を設定します。	権限を設定するには、「AWS ドキュメント」の指示に従ってください。	AWS システム管理者

ターゲットの Amazon RDS for Oracle DB インスタンスを作成し、Amazon S3 統合ロールを関連付ける

タスク	説明	必要なスキル
ターゲットの Amazon RDS for Oracle DB インスタンスを停止します。	Amazon RDS for Oracle インスタンスを作成するには、「AWS ドキュメント」の指示に従ってください。	AWS システム管理者
ロールを DB インスタンスに関連付けます。	ロールをインスタンスに関連付けるには、「AWS ドキュメント」の指示に従ってください。	DBA

ターゲットデータベースにデータベースユーザーを作成します。

タスク	説明	必要なスキル

ユーザーを作成します。	Oracle SQL Developer または SQL*Plus からターゲットの Amazon RDS for Oracle データベースに接続し、次の SQL コマンドを実行して、スキーマをインポートするユーザーを作成します。
	`create user SAMPLE_SCHEMA identified by <PASSWORD>;`
	`grant create session, resource to <USER NAME>;`
	`alter user <USER NAME> quota 100M on users;`
		DBA

ソース Oracle データベースからエクスポートファイルを作成する

タスク	説明	必要なスキル

データダンプファイルを作成する。	DATA_PUMP_DIR ディレクトリに sample.dmp というダンプファイルを作成して SAMPLE_SCHEMA ユーザーをエクスポートするには、次のスクリプトを使用します。
	`DECLARE`
	`  hdnl NUMBER;`
	`BEGIN`
	`  hdnl := dbms_datapump.open(operation => 'EXPORT',`
		DBA

タスク	説明	必要なスキル

	job_mode => 'SCHEMA',
	
	job_name => NULL);
	
	dbms_datapump.add_file(handle => hdnl,
	
	filename => 'sample.dmp',
	
	directory => 'DATA_PUMP_DIR',
	
	filetype => dbms_datapump.ku$_file_type_dump_file);
	
	dbms_datapump.add_file(handle => hdnl,
	
	filename => 'export.log',
	
	directory => 'DATA_PUMP_DIR',
	
	filetype => dbms_datapump.ku$_file_type_log_file);
	
	dbms_datapump.metadata_filter(hdnl, 'SCHEMA_EXPR', 'IN (''SAMPLE_SCHEMA'')');
	
	dbms_datapump.start_job(hdnl);
	END;	

タスク	説明	必要なスキル
	`/` ローカル DATA_PUMP_DIR ディレクトリにある export.log ファイルを参照して、エクスポートの詳細を確認します。	

ダンプファイルを S3 バケットにアップロードします。

タスク	説明	必要なスキル
データダンプファイルをソースから S3 バケットにアップロードします。	AWS CLI を使用して、次のコマンドを実行します。 `aws s3 cp sample.dmp s3://<bucket_created_epic_1>/`	DBA

S3 バケットから RDS インスタンスにエクスポートファイルをダウンロードします。

タスク	説明	必要なスキル
データダンプファイルを Amazon RDS にダウンロードします。	ダンプファイル sample.dmp を S3 バケットから Amazon RDS for Oracle データベースにコピーするには、次の SQL コマンドを実行します。この例では、sample.dmp ファイルが S3 バケット my-s3-integration1 から Oracle ディレクトリ DATA_PUMP_DIR にダウンロードされます。RDS イン	AWS システム管理者

タスク	説明	必要なスキル

	スタンスには、データベースとエクスポートファイルの両方を収容するのに十分なディスクスペースが割り当てられていることを確認してください。
	
	`-- If you want to download all the files in the S3 bucket remove the p_s3_prefix line.`
	
	`SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3( p_bucket_name => 'my-s3-integration', p_s3_prefix => 'sample.dmp', p_directory_name => 'DATA_PUMP_DIR') AS TASK_ID FROM DUAL;`
	
	前のコマンドはタスク ID を出力します。タスク ID のデータを確認してダウンロードのステータスを確認するには、以下のコマンドを実行します。
	
	`SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-<task_id>.log'));`	

タスク	説明	必要なスキル
	DATA_PUMP_DIR ディレクトリ内のファイルを確認するには、次のコマンドを実行します。 `SELECT filename, type,filesize/1024 /1024 size_megs ,to_char(mtime,'DD -MON-YY HH24:MI:SS') timestamp FROM TABLE(rdsadmin.rds _file_util.listdir (p_directory => upper('DATA_PUMP_D IR'))) order by 4;`	

ターゲットデータベース内のダンプファイルをインポートする

タスク	説明	必要なスキル
スキーマとデータを Amazon RDS に復元します。	ダンプファイルを sample_schema データベーススキーマにインポートするには、SQL Developer または SQL*Plus から次の SQL コマンドを実行します。 `DECLARE hdnl NUMBER; BEGIN hdnl := DBMS_DATA PUMP.OPEN( operation => 'IMPORT', job_mode => 'SCHEMA', job_name= >null);`	DBA

タスク	説明	必要なスキル

	<pre>DBMS_DATAPUMP.ADD_
	FILE(handle => hdnl,
	 filename => 'sample.d
	mp', directory =>
	 'DATA_PUMP_DIR',
	 filetype => dbms_data
	pump.ku$_file_type
	_dump_file);
	
	DBMS_DATAPUMP.ADD_FILE
	(handle => hdnl,
	 filename => 'import.l
	og', directory =>
	 'DATA_PUMP_DIR',
	 filetype => dbms_data
	pump.ku$_file_type
	_log_file);
	
	DBMS_DATAPUMP.
	METADATA_FILTER(hd
	nl,'SCHEMA_EXPR','
	IN (''SAMPLE_SCHEMA''
	)');
	
	DBMS_DATAPUMP.START_J
	OB(hdnl);
	
	END;
	/</pre> インポートのログファイルを表示するには、以下のコマンドを実行します。 <pre>SELECT text FROM
	 table(rdsadmin.rds
	_file_util.read_te
	xt_file('DATA_PUMP
	_DIR','import.log'));</pre>	

DATA_PUMP_DIR ディレクトリからダンプファイルを削除します。

タスク	説明	必要なスキル

エクスポートファイルを一覧表示してクリーンアップします。	DATA_PUMP_DIR ディレクトリのエクスポートファイルを一覧表示して削除し、次のコマンドを実行します。
	`-- List the files`
	`SELECT filename,`
	`type,filesize/1024`
	`/1024 size_megs`
	`,to_char(mtime,'DD`
	`-MON-YY HH24:MI:S`
	`S') timestamp FROM`
	` TABLE(rdsadmin.rds`
	`_file_util.listdir`
	`(p_directory =>`
	` upper('DATA_PUMP_D`
	`IR'))) order by 4;`
	`-- Remove the files`
	`EXEC UTL_FILE.`
	`FREMOVE('DATA_PUMP`
	`_DIR','sample.dmp');`
	`EXEC UTL_FILE.FREMOVE('`
	`DATA_PUMP_DIR','im`
	`port.log');`
		AWS システム管理者

関連リソース

- 「Amazon S3 統合」
- 「DB インスタンスを作成する」
- 「Amazon RDS の Oracle にデータをインポートする」
- 「Amazon S3 ドキュメント」
- 「IAM ドキュメント」

- 「Amazon RDS ドキュメント」
- 「Oracle Data Pump のドキュメント」
- 「Oracle SQL Developer」

pglogic を使用して Amazon EC2 上の PostgreSQL から Amazon RDS for PostgreSQL に移行する

作成者: Rajesh Madiwale (AWS)

概要

このパターンは、PostgreSQLの pglogical 拡張機能を使用して、PostgreSQL データベース (バージョン 9.5 以降) を Amazon Elastic Compute Cloud (Amazon EC2) から PostgreSQL 用 Amazon Relational Database Service (Amazon RDS) に移行する手順を概説しています。pglogical 拡張は、Amazon RDS の PostgreSQL バージョン 10 でサポートされています。

前提条件と制限

前提条件

- 適切なタイプの Amazon RDS インスタンスを選択します。詳細については、「Amazon RDS のインスタンスタイプ」を参照してください。
- PostgreSQL のソースバージョンとターゲットバージョンが同じであることを確認します。
- Amazon EC2 でpglogical extension with PostgreSQLをインストールして統合します。

製品バージョン

- Amazon RDS でサポートされている機能を備えた PostgreSQL バージョン 10 以降 (AWS ドキュメントの「Amazon RDS 上の PostgreSQL」を参照)。このパターンは、Amazon RDS 上の PostgreSQL 9.5 から PostgreSQL バージョン 10 への移行でテストされましたが、Amazon RDS 上のそれ以降のバージョンの PostgreSQL にも適用されます。

アーキテクチャ

データ移行アーキテクチャ

ツール

- pglogical 拡張機能
- PostgreSQL ネイティブユーティリティ: pg_dump と pg_restore

エピック

pglogic 拡張機能を使用してデータを移行する。

タスク	説明	必要なスキル
Amazon RDS PostgreSQL DB インスタンスを作成する。	Amazon RDS で PostgreSQL DB インスタンスを設定します。手順については、Amazon RDS for PostgreSQL ドキュメントを参照してください。	DBA
ソース PostgreSQL データベースからスキーマダンプを取得し、ターゲット PostgreSQL データベースで復元する。	1. pg_dump ユーティリティを -s オプションとともに使用して、ソースデータベースからスキーマファイルを生成します。 2. psql ユーティリティと -f オプションを使用して、スキーマをターゲットデータベースに読み込みます。	DBA
論理デコードを有効にする。	Amazon RDS DB パラメータグループで、rds.logical_replication 静的パラメータを 1 に設定します。手順については、Amazon RDS ドキュメントを参照してください。	DBA
pglogical 拡張機能をソースデータベースとターゲットデータベースで作成する。	1. ソース PostgreSQL データベースに、pglogical 拡張機能を作成します。 `psql -h <amazon-ec2-endpoint> -d target-dbname -U target-dbuser -c "create`	DBA

タスク	説明	必要なスキル
	extension pglogical ;" 2. ターゲット PostgreSQL データベースに、pglogical 拡張機能を作成します。 `psql -h <amazon-rds-endpoint> -d source-dbname -U source-dbuser -c "create extension pglogical ;"`	

ソース PostgreSQL データベースにパブリッシャーを作成する。	パブリッシャーを作成するには、以下を実行します。
	`psql -d dbname -p 5432 <<EOF`
	`SELECT pglogical.create_node( node_name := 'provider1', dsn := 'host=<ec2-endpoint> port=5432 dbname=source-dbname user=source-dbuser' );`
	`EOF`
		DBA

タスク	説明	必要なスキル

リプリケーションセットを作成し、テーブルとシーケンスを追加する。	ソース PostgreSQL データベースにリプリケーションセットを作成し、そのリプリケーションセットにテーブルとシーケンスを追加するには、以下を実行します。
	`psql -d dbname -p 5432 <<EOF`
	`SELECT pglogical.replication_set_add_all_tables('default', '{public}'::text[],synchronize_data := true);`
	`EOF`
		DBA

サブスクライバーを作成します。	ターゲット PostgreSQL データベースにサブスクライバーを作成するには、以下を実行します。
	`psql -h <rds-endpoint> -d target-dbname -U target-dbuser <<EOF`
	`SELECT pglogical.create_node(`
	`node_name := 'subscriber1',`
	`dsn := 'host=<rds-endpoint> port=5432 dbname=target-dbname password=postgres user=target-dbuser');`
	`EOF`
		DBA

タスク	説明	必要なスキル
サブスクリプションを作成する。	ターゲット PostgreSQL データベースにサブスクリプションを作成するには、以下を実行します。 `psql -h <rds-endpoint> -d target -U postgres <<EOF SELECT pglogical.create_subscription( subscription_name := 'subscription1', replication_sets := array['default'], provider_dsn := 'host=<ec2-endpoint> port=5432 dbname=<source-dbname> password=<password> user=source-dbuser' );`	DBA

データを検証する

タスク	説明	必要なスキル
ソースデータベースとターゲットデータベースを確認します。	ソースデータベースとターゲットデータベースをチェックして、データが正常に複製されていることを確認します。ソーステーブルとターゲットテーブルの `select count(1)` を使用して基本的な検証を実行できます。	DBA

関連リソース

- 「Amazon RDS」
- Amazon RDS 上の PostgreSQL の論理レプリケーション (Amazon RDS ドキュメント)
- pglogical (GitHub リポジトリ)
- 「pglogicalの制限 」(GitHub リポジトリの README ファイル)
- ロジカルレプリケーションを使用して PostgreSQL をオンプレミスまたは Amazon EC2 から Amazon RDS に移行する (AWS データベースブログ)

オンプレミス PostgreSQL データベースを Aurora PostgreSQL に移行する

作成者: Baji Shaik (AWS) および Jitender Kumar (AWS)

概要

Amazon Aurora PostgreSQL 互換エディションは、ハイエンドの商用データベースのパフォーマンスと可用性と、オープンソースデータベースのシンプルさとコスト効率を組み合わせています。Aurora は、同じ AWS リージョン内の 3 つのアベイラビリティーゾーンにストレージをスケーリングすることでこれらの利点を実現しており、最大 15 のリードレプリカインスタンスをサポートして読み取りワークロードをスケールアウトし、1 つのリージョン内で高可用性を実現します。Aurora グローバルデータベースを使用すると、PostgreSQL データベースを最大 5 つのリージョンに複製して、リージョンに障害が発生した場合のリモート読み取りアクセスとディザスタリカバリができます。このパターンは、オンプレミス PostgreSQL ソースデータベースを Aurora PostgreSQL 互換データベースに移行する手順を説明します。このパターンには、AWS データ移行サービス (AWS DMS) を使用、またはネイティブ PostgreSQL ツール (「pg_dump」、「pg_restore」、「psql」など) 、またはサードパーティツールを使用する 2 つの移行オプションが含まれます。

このパターンで説明する手順は、 Amazon Relational Database Service (Amazon RDS) と Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをターゲットとする PostgreSQL データベースにも適用されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの SAP ASE ソースデータベース
- Aurora PostgreSQL 互換 DB インスタンスまたはAmazon RDS for PostgreSQL DB インスタンス

制限

- データベースのサイズ制限は、Amazon RDS for PostgreSQL では 64 TB、Aurora PostgreSQL 互換では 128 TB です。
- AWS DMS 移行オプションを使用している場合は、PostgreSQL データベースをソースとして使用する場合の AWS DMS の制限を確認してください。

製品バージョン

- Amazon RDS での PostgreSQL のメジャーバージョンとマイナーバージョンのサポートについては、Amazon RDS ドキュメントのAmazon RDS for PostgreSQL の更新を参照してください。
- Aurora での PostgreSQL サポートについては、Aurora ドキュメントのAmazon Aurora PostgreSQL の更新を参照してください。
- AWS DMS 移行オプションを使用している場合は、AWS DMS ドキュメントのサポートされている PostgreSQL バージョンを参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミスの PostgreSQL データベース

ターゲットテクノロジースタック

- Aurora PostgreSQL 互換 DB インスタンス

ソースアーキテクチャ

ターゲットアーキテクチャ

データ移行アーキテクチャ

AWS DMS の使用

ネイティブ PostgreSQL ツールの使用

ツール

- AWS Database Migration Service (AWS DMS)は、データストアを AWS クラウドに移行、またはクラウドとオンプレミス設定の組み合わせ間で移行する際に役立ちます。このサービスは、さまざまなソースデータベースとターゲットデータベースをサポートしています。AWS DMS での使用がサポートされている PostgreSQL のソースデータベースとターゲットデータベースのバージョン

と エディションを検証する方法については、AWS DMS ソースとして PostgreSQL データベースを使用するを参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。

- ネイティブ PostgreSQL ツールには、pg_dump、pg_restore、psql が含まれます。

エピック

移行を分析する

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンを検証します。	AWS DMS を使用している場合は、サポートされているバージョンの PostgreSQL を使用していることを確認してください。	DBA
ストレージタイプと容量の要件を特定します。	1. ソースデータベースインスタンスに割り当てられたストレージを計算します。 2. ソースデータベースインスタンスの過去の増加メトリクスを収集します。 3. ターゲットデータベースインスタンスの将来の成長予想を予測します。 4. ソースデータベースの読み取り/書き込み IOPS の合計数を計算してストレージを割り当てます。汎用 SSD (gp2) ボリュームは 1 GB の各ストレージに 3 IOPS を提供します。	DBA、システム管理者
適切なインスタンスタイプ、容量、ストレージ機能、ネットワーク機能を選択します。	ターゲットデータベースインスタンスのコンピュート要件を決定します。追加の注	DBA、システム管理者

タスク	説明	必要なスキル
	意が必要と思われる既知のパフォーマンス問題を確認します。以下の要素を考慮して適切なインスタンスタイプを決定してください。 • ソースデータベースインスタンスの CPU 使用率 • ソースデータベースインスタンスの IOPS (読み取り/書き込み操作) • ソースデータベースインスタンスのメモリフットプリント 詳細については、Aurora ドキュメントの Aurora DB インスタンスクラスを参照してください。	
ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。	アプリケーションがデータベースと通信できるようにする適切なセキュリティグループを決定します。	DBA、システム管理者
アプリケーション移行戦略を特定します。	• アプリケーションの複雑さに基づき、移行カットオーバー戦略を決定します。 • プリケーションの目標復旧時間 (RTO) と目標復旧時点 (RPO) を決定し、それに応じてカットオーバーを計画します。	DBA、アプリ所有者、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
VPC を作成します。	ターゲットデータベースインスタンス用の新しい仮想プライベートクラウド (VPC) を作成します。	システム管理者
セキュリティグループを作成します。	(前のエピックで決めたように) VPC 内にセキュリティグループを作成して、データベースインスタンスへのインバウンド接続を許可します。	システム管理者
Aurora DB クラスターを構成して起動します。	新しい VPC とセキュリティグループを使用してターゲットデータベースインスタンスを作成し、インスタンスを起動します。	システム管理者

データを移行する — オプション 1 (AWS DMS を使用)

タスク	説明	必要なスキル
移行前の手順を完了します。	1. ソースデータベースのデータをクリーンアップします。 2. レプリケーションインスタンスを作成します。 3. ソースおよびターゲット DB エンドポイントの作成 4. 移行できるテーブルとオブジェクトの数を特定します。	DBA

タスク	説明	必要なスキル
移行前の手順を完了します。	1. 外部キーの制約とトリガーをターゲットデータベースにドロップします。 2. ターゲットデータベースのセカンダリインデックスをドロップします。 3. 全ロードタスクを使用して、ソースからターゲットデータベースにデータを移行します。 4. 外部キーを有効化します。 5. フラッシュカット移行を使用中で、アプリケーションのダウンタイムを最小限にする必要がある場合は、変更データキャプチャ(CDC)を有効化して、進行中の変更を複製します。 6. トリガーを有効化します。 7. シーケンスを更新します。 8. ソースとターゲットデータベースを検証します。	DBA
データを検証します。	データがソースからターゲットに正確に移行されたことを確認するためには、AWS DMS ドキュメントのデータ検証手順に従います。	DBA

データを移行する – オプション 2 (pg_dump と pg_restore を使用)

タスク	説明	必要なスキル
ソースデータベースを準備します。	1. pg_dump backup を格納するディレクトリがない場合は作成します。 2. データベースオブジェクトで pg_dump を実行する権限がある移行ユーザーを作成します。 3. EC2 インスタンスに接続し、pg_dump backup を実行します。 詳細については、pg_dumpドキュメントと AWS DMS ドキュメントのウォークスルーを参照してください。	DBA
ターゲットデータベースを準備します。	1. データベースオブジェクトで pg_restore を使用する権限がある移行ユーザーを作成します。 2. pg_restore を使用してデータベースダンプをインポートします。 詳細については、pg_restoreドキュメントと AWS DMS ドキュメントのウォークスルーを参照してください。	DBA
データを検証します。	1. ソースデータベースとターゲットデータベースのデー	DBA

タスク	説明	必要なスキル
	タベースオブジェクト数を比較します。 2. オブジェクト数に不一致が見つかった場合は解決します。	

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。	最初のエピックで作成したアプリケーション移行戦略を実装します。	DBA、アプリ所有者、システム管理者

ターゲットデータベースにカットオーバーする

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。	1. オンプレミス PostgreSQL データベースを指すすべてのアプリケーションサービスとクライアント接続を停止します。 2. AWS DMS タスクを実行します。 3. 必要に応じて、ロールバックタスク (Aurora PostgreSQL 互換からオンプレミス PostgreSQL データベースへのリバース CDC) を設定します。 4. データを検証します。	DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
	5. 新しい Aurora PostgreSQL 互換 DB インスタンスに Amazon Route 53 を設定して、新しいターゲットでアプリケーションサービスを開始します。 6. 新しい Aurora PostgreSQL 互換の DB インスタンスに、Amazon CloudWatch と Performance Insights の監視を追加します。	
移行をロールバックする必要がある場合。	1. Aurora PostgreSQL 互換データベースを指す、すべてのアプリケーションサービスを停止します。 2. 前のストーリーで作成した AWS DMS タスクを使用して、変更をソースのオンプレミス PostgreSQL データベースにロールバックします。 3. オンプレミス PostgreSQL データベースから Aurora PostgreSQL 互換データベースで実行中の AWS DMS タスクを停止します。 4. ソースのオンプレミス PostgreSQL データベースを指すようにアプリケーションを設定します。 5. すべてのロールバックデプロイが完了していることを確認します。	DBA、アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
リソースをシャットダウンします。	一時的な AWS リソースをシャットダウンします。	DBA、システム管理者
ドキュメントを検証します。	プロジェクト文書を確認して検証する。	DBA、アプリ所有者、システム管理者
メトリクスを収集します。	移行の所要時間、手動とツールによるコスト削減の割合などのメトリクスを収集します。	DBA、アプリ所有者、システム管理者
プロジェクトを閉じます。	プロジェクトを閉じて、フィードバックします。	DBA、アプリ所有者、システム管理者

関連リソース

リファレンス

- AWS Database Migration Service
- VPC とAmazon Aurora
- Amazon Aurora の価格設定
- AWS DMS ソースとしての PostgreSQL データベースの使用
- AWS DMS レプリケーションインスタンスの作成方法
- AWS DMS を使用したソースエンドポイントとターゲットエンドポイントの作成方法

追加リソース

- AWS DMS の使用開始
- データ移行の段階的説明
- Amazon Aurora のリソース

オンプレミス Microsoft SQL Server データベースを、Linux を実行中の Amazon EC2 上の Microsoft SQL Server に移行する

作成者: Tirumala Dasari (AWS)

概要

このパターンでは、バックアップユーティリティと復元ユーティリティを使用して、Microsoft Windows で実行中のオンプレミス Microsoft SQL Server データベースから Amazon Elastic Compute Cloud (Amazon EC2) Linux インスタンス上の Microsoft SQL Server に移行する方法を説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Microsoft SQL Server を搭載した Amazon EC2 Linux AMI (Amazon マシンイメージ)
- オンプレミス Windows と Linux EC2 インスタンス上の Microsoft SQL Server 間の AWS Direct Connect

アーキテクチャ

ソーステクノロジースタック

- Microsoft SQL Server データベースのオンプレミス版

ターゲットテクノロジースタック

- Microsoft SQL Server データベースを搭載した Linux EC2 インスタンス

データベース移行アーキテクチャ

ツール

- WinSCP — このツールを使用すると、Windows ユーザーは Linux ユーザーとファイルを簡単に共有できます。

- Sqlcmd — このコマンドラインユーティリティを使用すると、T-SQL ステートメントまたはバッチを SQL Server のローカルインスタンスとリモートインスタンスに送信できます。このユーティリティは、バッチ処理またはユニットテストなどの反復的なデータベースタスクに非常に便利です。

エピック

SQL Server を搭載した EC2 Linux インスタンスを準備する

タスク	説明	必要なスキル
Linux オペレーティングシステムを提供し、Microsoft SQL Server を含む AMI を選択します。		SysAdmin
AMI を設定して EC2 インスタンスを作成します。		SysAdmin
セキュリティグループのインバウンドルールとアウトバウンドルールを作成します。		SysAdmin
Microsoft SQL Server データベースの Linux EC2 インスタンスを設定します。		DBA
ソースデータベースと同様にユーザーを作成し、権限を付与します。		アプリ所有者、DBA
Linux EC2 インスタンスに SQL Server ツールと sqlcmd ユーティリティをインストールします。		DBA

データベースをバックアップし、バックアップファイルを Linux EC2 インスタンスに移動する

タスク	説明	必要なスキル
オンプレミス SQL Server データベースをバックアップします。		DBA
Microsoft SQL Server に WinSCP をインストールします。		DBA
Microsoft SQL Server を実行中の Linux EC2 インスタンスにバックアップファイルを移動します。		DBA

SQL Server を実行中の Linux EC2 インスタンスにデータベースを復元する

タスク	説明	必要なスキル
sqlcmd ユーティリティを使用して、データベースバックアップファイルからデータベースを復元します。		DBA
データベースオブジェクトとデータを検証します。		開発者、テストエンジニア

Windows SQL サーバーから Linux EC2 インスタンスの Windows SQL サーバーへのカットオーバー

タスク	説明	必要なスキル
データベースオブジェクトとデータを検証します。		開発者、テストエンジニア

タスク	説明	必要なスキル
オンプレミス Microsoft SQL Server データベースから、Microsoft SQL Server を実行中の Linux EC2 インスタンスへカットオーバーします。		DBA

関連リソース

- Amazon Linux および Ubuntu AMIs で SQL Server 2017 を設定する方法
- Linux インスタンスへの SQL ツールのインストール
- オンプレミス Microsoft SQL Server データベースから Linux EC2 インスタンス上の Microsoft SQL Server へのバックアップと復元

リンクされたサーバーを使用して、オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する

作成者: Kevin Yung (AWS)、Vishal Singh (AWS)、Viqash Adwani (AWS)

概要

リンクサーバーを使用すると、Microsoft SQL Server はデータベースサーバーの他のインスタンスで SQL ステートメントを実行できます。このパターンでは、オンプレミスの Microsoft SQL Server データベースを Microsoft SQL Server 用の Amazon Relational Database Service (Amazon RDS) に移行して、コスト削減と可用性の向上を実現する方法を説明します。現在、Amazon RDS for Microsoft SQL Server は、Amazon Virtual Private Cloud (Amazon VPC) ネットワーク外の接続をサポートしていません。

このパターンを使用すると、以下の目的を達成できます。

- リンクされたサーバーの機能を損なうことなく、Microsoft SQL Server を Amazon RDS for Microsoft SQL Server に移行すること。
- リンクされたMicrosoft SQL Serverをさまざまな段階で優先順位付けして移行すること。

前提条件と制限

前提条件

- 「Amazon RDS での Microsoft SQL Server」 が、必要な特徴量をサポートしているかどうかを確認してください。
- 「Amazon RDS for Microsoft SQL Server をデフォルトの照合順序で使用するか」、データベースレベルで照合順序を設定して使用できることを確認してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス データベース (Microsoft SQL サーバー)

ターゲットテクノロジースタック

- Amazon RDS for SQL Server

ソースステートアーキテクチャ

ターゲットステートアーキテクチャ

ターゲットの状態では、リンクサーバーを使用して Amazon RDS for Microsoft SQL Server を Amazon RDS に移行します。このアーキテクチャでは、Network Load Balancer を使用して、Amazon RDS for Microsoft SQL Server からのトラフィックを Microsoft SQL Server を実行しているオンプレミスサーバーにプロキシします。次の図は、Network Load Balancer のリバースプロキシ機能を示しています。

ツール

- CloudFormation
- Network Load Balancer
- Amazon RDS for SQL Server が、複数のアベイラビリティーゾーンにある (マルチ AZ)
- 「AWS Database Migration Service (AWS DMS)」

エピック

ランディングゾーン VPC を作成

タスク	説明	必要なスキル
CIDR 割り当てを作成します。		AWS SysAdmin

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成		AWS SysAdmin
VPC サブネットを作成します。		AWS SysAdmin
サブネットアクセスコントロールリスト (ACL) の作成		AWS SysAdmin
サブネットルートテーブルを作成します。		AWS SysAdmin
AWS Direct Connect または仮想プライベートネットワーク (VPN) を使用して接続を作成します。		AWS SysAdmin

データベースを Amazon RDS に移行します。

タスク	説明	必要なスキル
Amazon RDS for Microsoft SQL Server DB インスタンスを作成します。		AWS SysAdmin
AWS DMS レプリケーションインスタンスを作成します。		AWS SysAdmin
ソースデータベースとターゲットデータベースのエンドポイントを作成します。		AWS SysAdmin
移行タスクを作成し、全ロード後に連続レプリケーションを ON に設定します。		AWS SysAdmin

タスク	説明	必要なスキル
Amazon RDS for Microsoft SQL Server がオンプレミスの SQL Server データベースにアクセスできるように、ファイアウォールの変更をリクエストしてください。		AWS SysAdmin
Network Load Balancer を作成します。		AWS SysAdmin
データセンターのデータベースサーバーをターゲットとするターゲットグループを作成します。	データセンター (DC) のフェイルオーバーイベントを組み込むには、ターゲット設定でホスト名を使用することをおすすめします。	AWS SysAdmin
リンクサーバー設定用の SQL ステートメントを実行します。	Microsoft SQL 管理ツールを使用して、リンクサーバーを追加するための SQL ステートメントを Amazon RDS for Microsoft SQL Server DB インスタンスに対して実行します。SQL ステートメントで、Network Load Balancer のホスト名を使用するように @datasrc を設定します。Amazon RDS for Microsoft SQL Server DB インスタンスに対して、Microsoft SQL 管理ツールを使用することで、リンクされたサーバーログイン認証情報を追加します。	AWS SysAdmin
SQL Server の機能をテストして検証します。		AWS SysAdmin

タスク	説明	必要なスキル
カットオーバーを作成します。		AWS SysAdmin

関連リソース

- 「Amazon RDS の Microsoft SQL Server 用の一般的な管理タスク」
- 「Microsoft SQL Server の照合順序と文字セット」
- 「Network Load Balancer のドキュメント」
- 「Amazon RDS for Microsoft SQL Server でリンクされたサーバーを実装する（ブログ記事）」

ネイティブバックアップと復元メソッドを使用して、オンプレミスの Microsoft SQL サーバーデータベースを Amazon RDS for SQL Server に移行

ティルマラ・ダサリ (AWS)、デヴィッド・ケイロス (AWS)、ヴィシャル・シン (AWS) が制作

概要

このパターンは、オンプレミスの Microsoft SQL Server データベースを SQL Server DB インスタンスの Amazon Relational Database Service (Amazon RDS) に移行する方法 (同種移行) について説明しています。移行プロセスは SQL Server のネイティブバックアップと復元方法に基づいています。SQL Server Management Studio (SSMS) を使用してデータベースバックアップファイルを作成し、Amazon Simple Storage Service (Amazon S3) バケットを使用してバックアップファイルを保存してから、Amazon RDS for SQL Server にバックアップファイルを復元します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- S3 バケットと Amazon RDS for SQL Server DB インスタンスにアクセスするための AWS Identity and Access Management (IAM) ロールポリシー

制約事項

- このパターンで説明されているプロセスでは、データベースのみを移行します。SQL ログインまたはデータベースユーザー (SQL Server エージェントジョブを含む) は、追加の手順が必要なため、移行されません。

製品バージョン

- SQL Server 2017: サポートされているバージョンと特徴量の最新リストについては、AWS ドキュメントの「Amazon RDS 上の Microsoft SQL Server」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- オンプレミス Microsoft SQL Server データベース

ターゲットテクノロジースタック

- Amazon RDS for SQL Server DB インスタンス

データ移行アーキテクチャ

ツール

- Microsoft SQL Server Management Studio (SSMS) は、SQL Server インフラストラクチャを管理するための統合環境です。SQL Server とやり取りする豊富なスクリプトエディターを備えたユーザーインターフェイスとツールグループを備えています。

エピック

Amazon RDS for SQL Server DB インスタンスを作成

タスク	説明	必要なスキル
Amazon RDS for SQL Server でデータベースエンジンとして SQL Server を選択します。		DBA
SQL Server Express Editionを選択します。		DBA
データベースの詳細を指定します。	DB を作成する詳細情報について、「Amazon RDS のドキュメント」を参照してください。	DBA、アプリ所有者

オンプレミスの SQL Server データベースからバックアップファイルを作成

タスク	説明	必要なスキル
SSMS を使用してオンプレミスの SQL Server データベースConnect。		DBA
データベースのバックアップを作成します。	手順については、「SSMS ドキュメント」を参照してください。	DBA、アプリ所有者

Amazon S3 にバックアップファイルをアップロードします。

タスク	説明	必要なスキル
Amazon S3 にバケットを作成します。	詳細については、Amazon S3 のドキュメント を参照してください。	DBA
バックアップファイルを S3 バケットにアップロードします。	詳細については、Amazon S3 のドキュメント を参照してください。	SysOps 管理者

Amazon RDS for SQL Server でデータベースを復元する

タスク	説明	必要なスキル
オプショングループを Amazon RDS に追加します。	1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。 2. ナビゲーションペインで、オプショングループ、グループの作成の順に選択します。	SysOps 管理者

タスク	説明	必要なスキル
	3. オプショングループの情報を入力し、Create を選択します。 4. SQLSERVER_BACKUP_RESTORE オプショングループに オプションを追加するを選択します。 詳細については、「Amazon RDS ドキュメント」を参照してください。	
データベースを復元します。	1. Amazon RDS for SQL Server に SSMS でConnectします。 2. データベースを復元するには、`msdb.dbo.rds_restore_database` ストアドプロシージャを呼び出します。	DBA

ターゲットデータベースの検証

タスク	説明	必要なスキル
オブジェクトとデータを検証する。	ソースデータベースと Amazon RDS for SQL Server の間のオブジェクトとデータを検証します。 **Note** このタスクはデータベースのみを移行しま	アプリ所有者、DBA

タスク	説明	必要なスキル
	す。ログインとジョブは移行されません。	

カットオーバー

タスク	説明	必要なスキル
アプリケーションのトラフィックをリダイレクトします。	検証後、アプリケーションのトラフィックを Amazon RDS for SQL Server DB インスタンスにリダイレクトします。	アプリ所有者、DBA

関連リソース

- 「Amazon RDS ドキュメント」
- 「Amazon RDS for SQL Server のドキュメント」
- 「Microsoft SQL Server データベースエンジンのオプション」

AWS DMS と AWS SCT を使用して Microsoft SQL Server データベースを Aurora MySQL に移行

作成者: Mark Szalkiewicz (AWS) と Pavan Pusuluri (AWS)

概要

このパターンは、オンプレミスまたは Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにある Microsoft SQL Server データベースを Amazon Aurora MySQL Server インスタンスに移行する方法を示しています。このパターンでは AWS Database Migration Service (AWS DMS) と AWS Schema Conversion Tool (AWS SCT) を使用して、データ移行とスキーマ変換を行います。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターまたは EC2 インスタンスにあるMicrosoft SQL Server データベース
- AWS SCT コネクタ用の Java データベース接続 (JDBC) ドライバー。ローカルマシンまたは AWS SCT がインストールされている EC2 インスタンスにインストールされます。

制限

- データベースサイズの上限: 64 TB

製品バージョン

- Enterprise、Standard、Workgroup と Developer エディションの Microsoft SQL Server 2008、2008R2、2012、2014、2016 と 2017。Web と Express エディションは AWS DMS にサポートされていません。サポートされているバージョンの最新リストについては、「Microsoft SQL Server データベースの &DMS; のソースとしての使用」を参照してください。最も包括的なバージョンと機能サポートのため、AWS DMS の最新バージョンを使用することをお勧めします。AWS SCT でサポートされている Microsoft SQL Server のバージョンについては、「AWS SCT のドキュメント」を参照してください。
- MySQL バージョン 5.5、5.6、5.7. サポートされているバージョンの最新リストについては、「MySQL 互換データベースの AWS DMS のターゲットとしての使用」を参照してください。

アーキテクチャ

ソーステクノロジースタック

次のいずれかです:

- オンプレミスの Microsoft SQL Server データベース
- EC2 インスタンスにおける Microsoft SQL Server データベース。

ターゲットテクノロジースタック

- Aurora MySQL

データ移行アーキテクチャ

- AWS Cloud で実行されている Microsoft SQL Server データベースから

- オンプレミスデータセンターで実行している Microsoft SQL Server データベースから

ツール

- AWS DMS - AWS データ移行サービス (AWS DMS) は、Oracle、SQL Server、MySQL、PostgreSQL など、よく使用されている商用データベースやオープンソースデータベースとの間におけるデータを移行するのに役立ちます。AWS DMS を使用して、オンプレミスのインスタンス間 (AWS クラウドセットアップを使用)、またはクラウドセットアップとオンプレミスセットアップの組み合わせの間で、AWS クラウドにデータを移行できます。
- AWS SCT —「AWS Schema Conversion Tool」(AWS SCT) は、ソースデータベーススキーマと大部分のカスタムコード (ビュー、ストアドプロシージャ、関数など) をターゲットデータベースと互換性のある形式に自動的に変換し、異種データベースを簡単に移行できるようにします。

エピック

移行の準備をする

タスク	説明	必要なスキル
ソースとターゲットデータベースのバージョンとエンジンを検証します。		DBA
ソースデータベースとターゲットデータベース用のアウトバンドセキュリティグループを作成します。		SysAdmin
必要に応じて AWS SCT の EC2 インスタンスを作成して設定します。		DBA
AWS SCT および関連ドライバーの最新バージョンをダウンロードします。		DBA
ソースデータベースに前提条件となるユーザーと権限を追加して検証します。		DBA
ワークロード用の AWS SCT プロジェクトを作成し、ソースデータベースに接続します。		DBA
評価レポートを生成し、実現可能性を評価します。		DBA

ターゲットデータベースの準備

タスク	説明	必要なスキル
Amazon Aurora をデータベースエンジンとして使用して、ターゲット Amazon RDS DB インスタンスを作成します。		DBA
ソースからユーザ、ロール、権限のリストを抽出します。		DBA
既存のデータベースユーザーを新しいデータベースユーザーにマッピングします。		アプリ所有者
ターゲットデータベースにユーザーを作成します。		DBA
前のステップのロールをターゲットデータベースに適用します。		DBA
ソースデータベースにおけるデータベースオプション、パラメーター、ネットワークファイル、データベースリンクを確認し、ターゲットデータベースへの適用性を評価します。		DBA
関連する設定をターゲットに適用します。		DBA

オブジェクトを転送

タスク	説明	必要なスキル
ターゲットデータベースへの AWS SCT 接続を設定します。		DBA
AWS SCT を使用してスキーマを変換します。	AWS SCT は、ソースデータベーススキーマとほとんどのカスタムコードをターゲットデータベースと互換性のある形式に自動的に変換します。任意のコードは、自動的に変換できないツールは明確にマークされるため、手動で変換できます。	DBA
生成された SQL レポートを確認し、エラーと警告をすべて保存します。		DBA
自動スキーマ変更をターゲットに適用するか、.sql ファイルとして保存します。		DBA
AWS SCT がターゲット上にオブジェクトを作成したことを確認します。		DBA
自動的に変換できなかった項目は手動で書き換え、却下、または再設計します。		DBA
生成されたロールとユーザー権限を適用し、例外がないか確認します。		DBA

データを移行する

タスク	説明	必要なスキル
移行方法を決定します。		DBA
AWS DMS コンソールからレプリケーションインスタンスを作成します。	AWS DMS の使用方法の詳細については、「関連リソース」セクションのリンクを参照してください。	DBA
ソースおよびターゲットエンドポイントを作成します。		DBA
レプリケーションタスクを作成します。		DBA
レプリケーションタスクを開始し、ログをモニタリングします。		DBA

アプリケーションを移行する

タスク	説明	必要なスキル
AWS SCT を使用して、アプリケーションコード内の SQL 項目を分析し、変換します。	エンジン間でデータベーススキーマを変換するときは、古いデータベースエンジンの代わりに新しいデータベースエンジンとやり取りするように、アプリケーションの SQL コードを更新する必要があります。変換された SQL コードは表示、分析、編集、保存できます。AWS SCT の使用方法の詳細については、「関連	アプリ所有者

タスク	説明	必要なスキル
	リソース」セクションのリンクを参照してください。	
AWS に新しいアプリケーションサーバーを作成します。		アプリ所有者
アプリケーションコードを新しいサーバーに移行します。		アプリ所有者
ターゲットデータベースとドライバー用にアプリケーションサーバーを設定します。		アプリ所有者
アプリケーションのソースデータベースエンジンに固有のコードを修正します。		アプリ所有者
ターゲットエンジンに合わせてアプリケーションコードを最適化します。		アプリ所有者

カットオーバー

タスク	説明	必要なスキル
新規ユーザー、権限、コード変更のいずれかをすべてターゲットに適用します。		DBA
変更用にアプリケーションをロックします。		アプリ所有者
すべての変更がターゲットデータベースに反映されたことを検証します。		DBA

タスク	説明	必要なスキル
新しいターゲットデータベースを新しいアプリケーションにポイントします。		アプリ所有者
すべて再チェックします。		アプリ所有者
本番稼働。		アプリ所有者

プロジェクトを閉じる

タスク	説明	必要なスキル
の一時的な AWS リソース (AWS DMS レプリケーションインスタンスや AWS SCT に使用される EC2 インスタンス) をシャットダウンします。		DBA、アプリ所有者
内部チーム向けの AWS DMS プロセスに関するフィードバックを更新します。		DBA、アプリ所有者
AWS DMS プロセスを改訂し、必要に応じてテンプレートを改善します。		DBA、アプリ所有者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者
移行の所要時間、手動とツールによるコスト削減の割合などのメトリクスを収集します。		DBA、アプリ所有者
プロジェクトを閉じて、フィードバックします。		DBA、アプリ所有者

関連リソース

リファレンス

- 「AWS DMS ユーザーガイド」
- 「AWS SCT ユーザーガイド」
- 「Amazon Aurora の料金]

チュートリアルと動画

- AWS Database Migration Service の使用開始
- 「AWS Schema Conversion Tool の開始方法」
- 「Amazon RDS のリソース」
- 「AWS DMS のステップバイステップのチュートリアル」

ネイティブツールを使用して オンプレミスの MariaDB Amazon RDS for MariaDB に移行する

作成者: Shyam Sunder Rakhecha (AWS)

概要

このパターンは、ネイティブツールを使用して オンプレミスの MariaDB 用の Amazon Relational Database Service (Amazon RDS) に移行するためのガイダンスを提供します。MySQL ツールがインストールされている場合は、mysql と mysqldump を使用できます。MariaDB ツールがインストールされている場合は、mariadb と mariadb ダンプを使用できます。MySQL ツールと MariaDB ツールは同じオリジンですが、MariaDB バージョン 10.6 以降では若干の違いがあります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの MariaDB ソースデータベース

制限

- データベースサイズの上限: 64 TB

製品バージョン

- MariaDB バージョン 10.0-10.6 (サポートされているバージョンの最新リストについては、AWS ドキュメントの 「Amazon RDS 上の MariaDB」 を参照してください)

アーキテクチャ

ソーステクノロジースタック

- オンプレミスデータセンターの MariaDB データベース

ターゲットテクノロジースタック

- Amazon RDS for MariaDB の DB インスタンス

ターゲット アーキテクチャ

データ移行アーキテクチャ

ツール

- ネイティブ MySQL ツール:mysql と mysqldump
- ネイティブの MariaDB ツール:マリアデータベースとマリアダブダンプ

エピック

移行を計画する

タスク	説明	必要なスキル
ソースとターゲットのデータベースのバージョンとエンジンを検証します。		DBA
ターゲットサーバーインスタンスのハードウェア要件を特定します。		DBA、システム管理者
ストレージ要件 (ストレージタイプと容量) を特定します。		DBA、システム管理者
容量、ストレージ機能、ネットワーク機能に基づき、適切なインスタンスタイプを選択します。		DBA、システム管理者
ソースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定します。		DBA、システム管理者

タスク	説明	必要なスキル
アプリケーション移行戦略を特定します。		DBA、アプリ所有者、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC) を作成します。		システム管理者
セキュリティグループを作成します。		システム管理者
MariaDB を実行する Amazon RDS DB インスタンスを設定し、起動します。		システム管理者

データを移行する

タスク	説明	必要なスキル
ネイティブツールを使用してデータベースオブジェクトとデータを移行します。	ソースデータベースで、mysqldump または mariadb-dump を使用して、データベースオブジェクトとデータを含む出力ファイルを作成します。ターゲットデータベースでは、mysql または mariadb を使用してデータを復元します。	DBA
データを検証します。	ソースデータベースとターゲットデータベースをチェッ	DBA

タスク	説明	必要なスキル
	クして、データ移行が成功したことを確認します。	

アプリケーションを移行する

タスク	説明	必要なスキル
アプリケーション移行戦略に従います。		DBA、アプリ所有者、システム管理者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを新しいインフラストラクチャに切り替えます。		DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
一時的な AWS リソースをシャットダウンします。		システム管理者
プロジェクト文書を確認して検証する。		DBA、アプリ所有者、システム管理者
移行までの時間、ツールによるコスト削減などに関する指標を収集します。		DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
プロジェクトを終了し、フィードバックを提供します。		DBA、アプリ所有者、システム管理者

関連リソース

Amazon RDS リファレンス

- Amazon RDS for MariaDB
- 「Amazon Virtual Private Cloud VPC および Amazon RDSAmazon Aurora」
- 「Amazon RDS マルチ AZ 配置」
- 「Amazon RDS の価格設定」

MySQL と MariaDB のリファレンス

- 「mariadb ダンプ/mysqldump」
- 「mysql コマンドラインクライアント」

チュートリアルと動画

- 「Amazon RDS の開始方法」

オンプレミス MySQL データベースを Aurora MySQL に移行する

作成者: Igor Obradovic (AWS)

概要

このパターンでは、オンプレミスの MySQL ソースデータベースを Amazon Aurora MySQL 互換エディションに移行する方法について説明します。移行には、 AWS Database Migration Service (AWS DMS) を使用するか、mysqldbcopy やmysqldump などのネイティブ MySQL ツールを使用するという 2 つのオプションについて説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスデータセンターの MySQL ソースデータベース

制約事項

- データベースサイズの上限:128 TB

製品バージョン

- MySQL バージョン 8.0 (Aurora MySQL バージョン 3) は、標準サポートで利用できます。
- MySQL バージョン 5.7 (Aurora MySQL バージョン 2) は、延長サポートで追加料金で利用できます。

サポートされているバージョンの最新リストについては、 AWS ドキュメントの「Amazon Aurora バージョン」を参照してください。を使用している場合は AWS DMS、「 でサポートされている for MySQL バージョンのターゲットとして MySQL 互換データベースを使用する AWS DMS」も参照してください AWS DMS。 MySQL

アーキテクチャ

ソーステクノロジースタック

- オンプレミス MySQL データベース

ターゲットテクノロジースタック

- Amazon Aurora MySQL 互換エディション

ターゲット アーキテクチャ

Aurora データはクラスターボリュームに保存されます。クラスターボリュームは、ソリッドステートドライブ (SSDs。クラスターボリュームは、単一の AWS リージョンの 3 つのアベイラビリティーゾーン間のデータのコピーで構成されます。データはアベイラビリティーゾーン間で自動的にレプリケートされるため、耐久性が高く、データ損失の可能性が低くなります。

Aurora は、データベースボリュームを多くのディスクにまたがる 10 GB のセグメントに自動的に分割します。データベースボリュームの各 10 GB チャンクは、3 つのアベイラビリティーゾーンにまたがる 6 つの方法でレプリケートされます。次の図は、Aurora DB クラスター内のクラスターボリューム、ライター DB インスタンス、リーダー DB インスタンスの関係と、コンピューティング容量とストレージの分離を示しています。このアーキテクチャの詳細については、Aurora のドキュメントとよくある質問を参照してください。

データ移行アーキテクチャ

の使用 AWS DMS：

次の図は、 を使用して AWS クラウド、オンプレミス MySQL データベースを の Aurora MySQL 互換クラスターに移行する方法を示しています AWS DMS。

ネイティブ MySQL ツールを使用する:

次の図は、mysqldbcopy やmysqldump などのネイティブ MySQL ツールを使用した AWS クラウド、オンプレミス MySQL データベースの 内の Aurora MySQL 互換クラスターへの移行を示しています。

ツール

- AWS Database Migration Service （ AWS DMS ） は、複数のソースデータベースエンジンとターゲットデータベースエンジンをサポートしています。でサポートされている MySQL ソースデータ

ベースとターゲットデータベースの詳細については AWS DMS、MySQL 互換データベースの への移行 AWS」を参照してください。最も包括的なバージョンと機能のサポート AWS DMS には、の最新バージョンを使用することをお勧めします。

- mysqldbcopy MySQL は、単一のサーバー上またはサーバー間で MySQL データベースをコピーする MySQL ユーティリティです。
- mysqldump MySQL は、バックアップまたは移行の目的で MySQL データベースからダンプファイルを作成する MySQL ユーティリティです。

エピック

移行を計画する

タスク	説明	必要なスキル
バージョンとエンジンを検証します。	ソースデータベースとターゲットデータベースのデータベースバージョンとエンジンを検証します。	DBA
ハードウェア要件を特定します。	ターゲットサーバーインスタンスのハードウェア要件を特定します。	DBA、システム管理者
ストレージ要件を特定します。	ストレージ要件 (ストレージタイプと容量) を特定します。	DBA、システム管理者
インスタンスタイプを選択します。	コンピューティング、ストレージ、ネットワークの要件に基づいて、適切なインスタンスタイプを選択します。	DBA、システム管理者
ネットワークアクセスのセキュリティ要件を決定します。	ソースデータベースとターゲットデータベースのネットワークアクセスセキュリティ要件を特定する。	DBA、システム管理者
戦略を決定します。	アプリケーション移行戦略を特定します。	DBA、アプリ所有者、システム管理者

インフラストラクチャを設定する

タスク	説明	必要なスキル
仮想プライベートクラウド (VPC)の作成	手順については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「VPC の作成」を参照してください。	システム管理者
セキュリティグループを作成します。	手順については、Amazon VPC ドキュメントの「VPC のセキュリティグループを作成する」を参照してください。	システム管理者
で Aurora MySQL 互換 DB クラスターを設定して起動します AWS アカウント。	手順については、Aurora ドキュメントの「Amazon Aurora DB クラスターの作成」を参照してください。	システム管理者

データ移行 - オプション 1

タスク	説明	必要なスキル
ネイティブ MySQL ツールまたはサードパーティツールを使用して、データベースオブジェクトとデータを移行します。	手順については、mysqldbcopy や mysqldump などの MySQL ツールのドキュメントを参照してください。	DBA

データ移行 — オプション 2

タスク	説明	必要なスキル
でデータを移行します AWS DMS。	手順については、 AWS DMS ドキュメントの「ソースと	DBA

タスク	説明	必要なスキル
	して MySQL 互換データベースを使用する」および「ターゲットとして MySQL 互換データベースを使用する」を参照してください。	

アプリケーションを移行する

タスク	説明	必要なスキル
戦略に従います。	アプリケーション移行戦略に従います。	DBA、アプリ所有者、システム管理者

カットオーバー

タスク	説明	必要なスキル
アプリケーションクライアントを切り替えます。	アプリケーションクライアントを切り替えて、新しい Aurora クラスターエンドポイントに接続します。	DBA、アプリ所有者、システム管理者

プロジェクトを閉じる

タスク	説明	必要なスキル
リソースをシャットダウンします。	一時 AWS リソースをシャットダウンします。	DBA、システム管理者
ドキュメントを確認します。	プロジェクト文書を確認して検証する。	DBA、アプリ所有者、システム管理者
メトリクスを収集します。	移行にかかる時間、手動ステップとツールの使用率、コ	DBA、アプリ所有者、システム管理者

タスク	説明	必要なスキル
	スト削減などのメトリクスを収集します。	
移行プロジェクトを完了します。	プロジェクトを終了し、フィードバックを提供します。	アプリ所有者、DBA、システム管理者

関連リソース

リファレンス

- Amazon Aurora MySQL DB クラスターへのデータの移行
- AWS DMS website
- AWS DMS ドキュメント
- 「Amazon Aurora の料金]
- Aurora MySQL DB クラスターの作成と接続
- Amazon VPC と Amazon RDS
- 「Amazon Aurora ドキュメント」

チュートリアルと動画

- の開始方法 AWS DMS
- Amazon Aurora の使用開始

Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミスの MySQL データベースを Aurora MySQL に移行する

作成者: Rohan Jamadagni (AWS), sajith menon (AWS), and Udayasimha Theepireddy (AWS)

概要

このパターンでは、Percona XtraBackup を使用して、大規模なオンプレミスの MySQL データベースを Amazon Aurora MySQL に効率的に移行する方法を説明します。Percona XtraBackup は、MySQL ベースのサーバー用のオープンソースのノンブロッキングバックアップユーティリティです。このパターンは、Amazon Elastic File System (Amazon EFS) を使用して Amazon Simple Storage Service (Amazon S3) にバックアップをアップロードする時間を短縮し、Amazon Aurora MySQL にバックアップを復元する方法を示しています。このパターンには、Percona のインクリメンタルバックアップを作成して、ターゲット Aurora MySQL データベースに適用されるバイナリログの数を最小限に抑える方法の詳細も記載されています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS Identity and Access Management (IAM) ロールとポリシーの作成権限
- オンプレミスの MySQL データベースと AWS 上の仮想プライベートクラウド (VPC) 間のネットワーク接続

機能制限

- ソースサーバーは、ネットワークファイルシステム (NFS) クライアント (nfs-utils/nfs-common) をインストールできる Linux ベースのシステムである必要があります。
- バックアップファイルのアップロードに使用する S3 バケットは、サーバー側の暗号化 (SSE-S3/SSE-KMS) のみをサポートしています。
- Amazon S3 では、バックアップファイルのサイズが 5 TB に制限されます。バックアップファイルが 5 TB を超える場合は、それを複数の小さいファイルに分割できます。
- S3 バケットにアップロードされるソースファイルの数は、百万ファイルを超えることはできません。
- このパターンでは、Percona XtraBackup 完全バックアップと増分バックアップのみがサポートされます。`--tables`、`--tables-exclude`、`--tables-file`、`--databases`、`--databases-`

exclude、または --databases-file を使用する部分バックアップはサポートされていません。

- Aurora は、ユーザー、関数、ストアドプロシージャ、またはタイムゾーン情報をソース MySQL データベースから復元しません。

製品バージョン

- ソースデータベースは MySQL バージョン 5.5、5.6、または 5.7 である必要があります。
- MySQL 5.7 では、Percona XtraBackup 2.4 を使用する必要があります。
- MySQL 5.6 と 5.6 では、Percona XtraBackup 2.3 または 2.4 を使用する必要があります。

アーキテクチャ

ソーステクノロジースタック

- Linux ベースのオペレーティングシステム。
- MySQL サーバー
- Percona XtraBackup

ターゲットテクノロジースタック

- Amazon Aurora
- Amazon S3
- Amazon EFS

ターゲット アーキテクチャ

ツール

AWS サービス

- 「Amazon Aurora」はフルマネージドリレーショナルデータベースエンジンで、MySQL のデプロイを簡単に、コスト効率よく設定、操作、スケーリングすることができます。Aurora MySQL は MySQL のドロップイン代替品です。

- 「Amazon Elastic File System (Amazon EFS)」は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- 「Percona XtraBackup」は、データベースを中断したりブロックしたりすることなく、MySQL データベースのストリーミング、圧縮、および増分バックアップを実行するオープンソースのユーティリティです。

エピック

「Amazon EFS ファイルシステムの作成」

タスク	説明	必要なスキル
Amazon EFS マウントターゲットに関連付けるセキュリティグループを作成します。	AWS Transit Gateway 経由でオンプレミスデータベースに VPN アタッチメントを設定したセキュリティグループを VPC に作成します。このストーリーやその他のストーリーで説明されているコマンドと手順の詳細については、このパターンの最後にある「関連リソース」セクションのリンクを参照してください。	AWS DevOps/データベース管理者
セキュリティグループルールを編集します。	NFS タイプ、ポート 2049、オンプレミスデータベースサーバーの IP 範囲をソースとして使用して、インバウンドルールを追加します。デフォルトでは、アウトバウンドルールはすべてのトラフィッ	AWS DevOps/データベース管理者

タスク	説明	必要なスキル
	クの流出を許可します。そうでない場合は、NFS ポートへの接続を開くアウトバウンドルールを追加してください。さらに 2 つのインバウンドルールを追加します。ポート 2049 (ソース:同じセキュリティグループのセキュリティグループ ID) とポート 22 (ソース:EC2 インスタンスに接続する IP 範囲) です。	
ファイルシステムを作成します。	マウントターゲットでは、前の記事で作成した VPC とセキュリティグループを使用します。オンプレミスデータベースの I/O 要件に基づいてスループットモードとパフォーマンスを選択します。オプションで、保管時の暗号化を有効にします。	AWS DevOps/データベース管理者

ファイルシステムをマウントします。

タスク	説明	必要なスキル
EC2 インスタンスに関連付ける IAM インスタンスプロファイルロールを作成します。	Amazon S3 にオブジェクトをアップロードしてアクセスするためのアクセス許可を持つ IAM ロールを作成します。バックアップをポリシーリソースとして保存する S3 バケットを選択します。	AWS DevOps

タスク	説明	必要なスキル
EC2 インスタンスを作成します。	Linux ベースの EC2 インスタンスを起動し、前のステップで作成した IAM インスタンスプロファイルロールと、前に作成したセキュリティグループをアタッチします。	AWS DevOps
NFS クライアントをインストールします。	オンプレミスのデータベースサーバーと EC2 インスタンスに NFS クライアントをインストールします。インストール手順については、「追加情報」を参照してください。	DevOps
Amazon EFS ファイルシステムをマウントします。	オンプレミスと EC2 インスタンスに、Amazon EFS ファイルシステムをマウントします。各サーバーで、バックアップを保存するディレクトリを作成し、マウントターゲットエンドポイントを使用してファイルシステムをマウントします。例については、「追加情報」を参照してください。	DevOps

MySQL ソースデータベースのバックアップの作成

タスク	説明	必要なスキル
Percona XtraBackup のインストール	Percona XtraBackup 2.3 または 2.4 (MySQL データベースのバージョンによって異なります) をオンプレミスのデータベースサーバーにインストー	データベース管理者

タスク	説明	必要なスキル
	ルします。インストールリンクについては、「関連リソース」セクションを参照してください。	
ソースデータベースのスキーマとテーブルの数を数えます。	ソース MySQL データベース内のスキーマとオブジェクトの数を収集して書き留めます。これらの数を使用して、移行後に Aurora MySQL データベースを検証します。	データベース管理者
(オプション) ソースデータベースの最新のバイナリログシーケンスを書き留めておきます。	ダウンタイムを最小限に抑えるために、ソースデータベースと Aurora MySQL 間でバイナリログのレプリケーションを確立する場合は、このステップを実行します。log-bin を有効にし、server_id が一意である必要があります。バックアップを開始する直前に、ソースデータベースの現在のバイナリログシーケンスを書き留めておきます。完全バックアップのみを使用する場合は、完全バックアップの直前にこの手順を実行してください。完全バックアップの後に増分バックアップを行う予定の場合は、Aurora MySQL DB インスタンスで復元する最後の増分バックアップの直前にこのステップを実行してください。	データベース管理者

タスク	説明	必要なスキル
ソース MySQL データベースのフルバックアップを開始します。	Percona XtraBackup を使用して MySQL ソースデータベースの完全バックアップを作成します。フルバックアップと増分バックアップのコマンドの例については、「追加情報」セクションを参照してください。	データベース管理者
(オプション) Percona XtraBackup を使用して増分バックアップを作成します。	インクリメンタルバックアップを使用すると、ソースデータベースを Aurora MySQL と同期するために適用する必要があるバイナリログの量を減らすことができます。サイズが大きくてトランザクション量の多いデータベースでは、バックアップ中に大量のバイナリログが生成されることがあります。増分バックアップを作成して共有の Amazon EFS ファイルシステムに保存することで、データベースのバックアップとアップロードにかかる時間を大幅に短縮できます。詳細については、「追加情報」を参照してください。Aurora への移行プロセスを開始する準備が整うまで、引き続き増分バックアップを行います。	データベース管理者

タスク	説明	必要なスキル
バックアップの作成	このステップでは、バックアップ中に処理されていたトランザクションのトランザクションログがバックアップに適用されます。引き続き、各増分バックアップにトランザクションログ (--apply-log-only) を適用して、最後のバックアップを除いてバックアップをマージします。例については、「追加情報」セクションを参照してください。このステップの後、マージされた完全バックアップは「~/<efs_mount_name>/fullbackup」に保存されます。	データベース管理者
最後にマージされたバックアップを圧縮して分割します。	最終的な統合バックアップを準備したら、tar、zip、split コマンドを使用して、バックアップの小さな ZIP ファイルを作成します。例については、「追加情報」セクションを参照してください。	データベース管理者

Aurora MySQL DB クラスターにバックアップを復元します。

タスク	説明	必要なスキル
Amazon S3 にバックアップファイルをアップロードします。	バックアップファイルが保存されている Amazon EFS ファイルシステムは、オンプレミスデータベースと EC2 インスタンスの両方にマウン	AWS DevOps

タスク	説明	必要なスキル
	トされるため、バックアップファイルは EC2 インスタンスですぐに使用できます。Secure Shell (SSH) を使用して EC2 インスタンスに接続し、圧縮されたバックアップファイルを新規または既存の S3 バケットにアップロードします。例: aws s3 sync ~/<efs_mount_name>/fullbackup s3://<bucket_name>/fullbackup 詳細については、「関連リソース」セクションのリンクを参照してください。	
Aurora が Amazon S3 にアクセスするためのサービスロールを作成します。	「rds.amazonaws.com」という信頼のある IAM ロールと、バックアップファイルが保存されている S3 バケットに Aurora がアクセスできるようにするポリシーを作成します。必要な権限は、ListBucket、GetObject、GetObjectVersion です。	AWS DevOps

タスク	説明	必要なスキル
Aurora のネットワーク設定を作成します。	少なくとも 2 つのアベイラビリティーゾーンと、ソースデータベースへのアウトバウンド接続を可能にするサブネットルートテーブル設定を含むクラスター DB サブネットグループを作成します。オンプレミスデータベースへのアウトバウンド接続を許可し、管理者が Aurora DB クラスターに接続できるようにするセキュリティグループを作成します。詳細については、「関連リソース」セクションのリンクを参照してください。	AWS DevOps/データベース管理者
Aurora MySQL DB クラスターにバックアップを復元します。	Amazon S3 にデータをアップロードしたバックアップから復元します。ソースデータベースの MySQL バージョンを指定し、バックアップファイルをアップロードした S3 バケット名とフォルダパスのプレフィックスを指定し (たとえば、「追加情報」セクションの例では「フルバックアップ」)、Aurora に Amazon S3 へのアクセスを許可するために作成した IAM ロールを指定します。	AWS DevOps/データベース管理者

タスク	説明	必要なスキル
Aurora MySQL データベースを検証します。	復元された Aurora DB クラスター内のスキーマとオブジェクトの数を、ソースデータベースから取得した数と照合して検証します。	データベース管理者
バイナリログのレプリケーションを設定します。	Aurora DB クラスターに復元された最後のバックアップを作成する前に、前にメモしたバイナリログシーケンスを使用してください。ソースデータベースにレプリケーションユーザーを作成し、「追加情報」セクションの指示に従って適切な権限を付与し、Aurora でのレプリケーションを有効にし、レプリケーションが同期されていることを確認します。	AWS DevOps/データベース管理者

関連リソース

Amazon EFS ファイルシステムの作成

- 「セキュリティグループの作成」(Amazon VPC ドキュメント)
- 「トランジットゲートウェイ VPN アタッチメント」(Amazon VPC ドキュメント)
- 「AWS Transit Gateway を使用した VPN スループットのスケーリング」(ネットワーキングとコンテンツ配信に関するブログ)
- 「Amazon EFS ファイルシステムの作成」(Amazon EFS ドキュメント)
- 「マウントターゲットの作成」(Amazon EFS ドキュメント)
- 「保管中のデータの暗号化」(Amazon EFS のドキュメント)

ファイルシステムのマウント

- Amazon EC2 の IAM ロール (Amazon EC2 のドキュメント)
- 「Amazon EC2 Linux インスタンスの起動」(Amazon EC2 ドキュメント)
- 「NFS クライアントのインストール」 (Amazon EFS ドキュメント)
- 「オンプレミスクライアントに Amazon EFS ファイルシステムをマウントする」(Amazon EFS ドキュメント)
- 「EFS ファイルシステムのマウント」(Amazon EFS ドキュメント)

MySQL ソースデータベースのバックアップの作成

- 「Percona XtraBackup 2.3 のインストール」(Percona XtraBackup ドキュメント)
- 「Percona XtraBackup 2.4 のインストール」(Percona XtraBackup ドキュメント)
- 「レプリケーションマスター構成の設定」(MySQL ドキュメント)
- 「外部 MySQL データベースから Aurora MySQL DB クラスターへのデータ移行」(Aurora ドキュメント)
- 「インクリメンタルバックアップ」(Percona XtraBackup ドキュメンテーション)

Amazon Aurora MySQL へのバックアップの復元

- 「バケットの作成」(Amazon S3 ドキュメント)
- 「SSH を使用した Linux インスタンスへの接続」(Amazon EFS ドキュメント)
- 「AWS CLI の設定」(AWS CLI ドキュメント)
- 「同期コマンド」(AWS CLI コマンドリファレンス)
- 「Amazon S3 リソースにアクセスするための IAM ポリシーの作成」(Aurora ドキュメント)
- 「DB クラスターの前提条件」 (Aurora ドキュメント)
- 「DB サブネットグループの使用」(Aurora ドキュメント)
- 「プライベート DB インスタンスの VPC セキュリティグループの作成」(Aurora ドキュメント)
- 「S3 バケットからの Aurora MySQL DB クラスターの復元」(Aurora ドキュメント)
- 「MySQL または別の Aurora DB クラスターとのレプリケーションの設定」(Aurora ドキュメント)
- 「mysql.rds_set_external_master プロシージャ」(Amazon RDS 上の MySQL SQL リファレンス)
- 「mysql.rds_start_replication プロシージャ」(Amazon RDS 上の MySQL SQL リファレンス)

その他のリファレンス

- 「外部 MySQL データベースから Aurora MySQL DB クラスターへのデータ移行」(Aurora ドキュメント)
- 「MySQL サーバーのダウンロード」(Oracle ウェブサイト)

チュートリアルと動画

- 「Amazon S3 を使用して MySQL データを Aurora MySQL DB クラスターに移行」(AWS ナレッジセンター)
- 「Amazon EFS のセットアップとマウント」(ビデオ)

追加情報

NFS クライアントのインストール

- Red Hat または類似の Linux オペレーティングシステムを使用している場合は、以下のコマンドを使用してください。

```
$ sudo yum -y install nfs-utils
```

- Ubuntu または類似の Linux オペレーティングシステムを使用している場合は、以下のコマンドを使用してください。

```
$ sudo apt-get -y install nfs-common
```

詳細については、Amazon EFS ドキュメント」の「チュートリアル」を参照してください。

Amazon EFS ファイルシステムのマウント

次のコマンドを使用してください。

```
mkdir ~/<efs_mount_name>
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-IP:/ ~/<efs_mount_name>
```

詳細については、Amazon EFS ドキュメントの「チュートリアル」と「EFS ファイルシステムのマウント」を参照してください。

MySQL ソースデータベースのバックアップの作成

フルバックアップ

次のようなコマンドを使用して、バックアップを取って圧縮し、それぞれ 1 GB の小さなチャンクに分割します。

```
xtrabackup --backup --user=dbuser --password=<password> --binlog-info=AUTO --stream=tar
 --target-dir=~/<efs_mount_name>/fullbackup | gzip - | split -d --bytes=1024MB - ~/
<efs_mount_name>/fullbackup/backup.tar.gz &
```

完全バックアップの後にインクリメンタルバックアップを行う予定がある場合は、バックアップを圧縮したり分割したりしないでください。代わりに、以下のようなコマンドを使用します。

```
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/
<efs_mount_name>/fullbackup/
```

増分バックアップ

`--incremental-basedir` パラメータにはフルバックアップパスを使用してください。例:

```
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/
<efs_mount_name>/incremental/backupdate --incremental-basedir=~/<efs_mount_name>/
fullbackup
```

ここで、[basedir] はフルバックアップと xtrabackup_checkpoints ファイルへのパスです。

バックアップの詳細については、Aurora ドキュメントの「外部 MySQL データベースから Amazon Aurora MySQL DB クラスターへのデータ移行」を参照してください。

バックアップの準備

フルバックアップを準備するには:

```
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup
```

増分バックアップを準備するには:

```
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup --
incremental-dir=~/<efs_mount_name>/incremental/06062020
```

最終バックアップを準備するには:

```
xtrabackup --prepare --target-dir=~/<efs_mount_name>/fullbackup --incremental-dir=~/
<efs_mount_name>/incremental/06072020
```

詳細については、Percona XtraBackup ドキュメントの「インクリメンタルバックアップ」を参照してください。

マージしたバックアップを圧縮して分割する

統合したバックアップを「 ~/<efs_mount_name>/fullbackup」に圧縮するには:

```
tar -zcvf <backupfilename.tar.gz> ~/<efs_mount_name>/fullbackup
```

バックアップを分割するには:

```
split -d -b1024M --verbose  <backupfilename.tar.gz> <backupfilename.tar.gz>
```

バイナリログのレプリケーションをセットアップする

ソースデータベースにレプリケーションユーザーを作成し、適切な権限を付与するには:

```
CREATE USER 'repl_user'@'' IDENTIFIED BY ''; GRANT REPLICATION CLIENT, REPLICATION
 SLAVE ON *.* TO 'repl_user'@'';
```

Aurora DB クラスターに接続して Aurora でのレプリケーションを有効にするには、DB クラスターパラメータグループのバイナリログを有効にします。設定 `binlog_format = mixed` (混合モードを推奨)。この変更では、更新を適用するためにインスタンスを再起動する必要があります。

```
CALL mysql.rds_set_external_master ('sourcedbinstanceIP', sourcedbport, 'repl_user',
 '', 'binlog_file_name', binlog_file_position, 0); CALL mysql.rds_start_replication;
```

レプリケーションが同期していることを確認するには:

```
SHOW Slave Status \G;
```

「マスターに対する遅れ秒数」フィールドには、オンプレミスデータベースから Aurora がどれだけ遅れているかが示されます。

AWS App2Container を使用したオンプレミスの Java プリケーションの AWS への移行

作成者:Dhananjay Karanjkar (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

AWS App2Container (A2C) は、コードを変更することなく、仮想マシンで実行されている既存のアプリケーションをコンテナに変換するのに役立つコマンドラインツールです。A2Cは、サーバーで実行されているアプリケーションを発見し、依存関係を特定し、Amazon Elastic Container Service (Amazon ECS) および Amazon Elastic Kubernetes Service (Amazon EKS) にシームレスにデプロイするための関連アーティファクトを生成する。

このパターンは、アプリケーションサーバーにデプロイされたオンプレミスの Java アプリケーションを、ワーカーマシンから App2Container を使用して AWS Fargate または Amazon EKS にリモート移行する手順を示しています。

ワーカーマシンは、以下のユースケースにご使用いただけます。

- Java アプリケーションが実行されているアプリケーションサーバーでは Docker のインストールは許可されていないか、使用できません。
- 異なる物理サーバーまたは仮想サーバーにデプロイされた複数のアプリケーションの移行を管理する必要があります。

このパターンでは、AWS CodeCommit、 AWS CodePipeline、 を使用します AWS CodeBuild。

前提条件と制限

前提条件

- Linux サーバー上で Java アプリケーションを実行しているアプリケーションサーバー
- Linux オペレーティングシステムを搭載したワーカーマシン
- 20 GB 以上の空きディスク容量があるワーカーマシン

機能制限

- すべてのアプリケーションに対応しているわけではありません。詳細については、「Linux でサポートされるアプリケーション」を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Linux サーバー上で実行されている Java アプリケーション

ターゲットテクノロジースタック

- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- Amazon Elastic Container Registry
- AWS Fargate

ターゲット アーキテクチャ

ツール

ツール

- 「AWS App2Container」— AWS App2Container (A2C) は、オンプレミスのデータセンターや仮想マシン上で動作するアプリケーションを、Amazon ECS や Amazon EKS で管理されるコンテナ内で動作するようにリフトとシフトするためのコマンドラインツールです。
- 「AWS CodeBuild」— AWS CodeBuild はクラウドの完全マネージド型のビルドサービスです。CodeBuild はソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイできるアーティファクトを作成します。
- 「AWS CodeCommit」— AWS CodeCommit は、Amazon Web Services がホスティングするバージョン管理サービスで、アセット (ドキュメント、ソースコード、バイナリファイルなど) をクラウド上に非公開で保存と管理するために使用できます。
- 「AWS CodePipeline」— AWS CodePipeline は、ソフトウェアのリリースに必要な手順のモデル化、可視化、および自動化に使用できる継続的な配信サービスです。

- 「Amazon ECS」— Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理に使用される、高度にスケーラブルで高速のコンテナ管理サービスです。
- 「Amazon ECR」— Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージドコンテナイメージレジストリサービスです。
- 「Amazon EKS」— Amazon Elastic Kubernetes Service (Amazon EKS) は、独自の Kubernetes コントロールプレーンやノードをインストール、運用、保守することなく、AWS 上で Kubernetes を実行するために使用できるマネージドサービスです。
- 「AWS Fargate」— AWS Fargate は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのサーバーやクラスタを管理することなくコンテナを実行するために、Amazon ECS で使用できる技術です。Fargateを使用すると、コンテナを実行するために仮想マシンのクラスターをプロビジョニング、設定、スケールする必要がありません。

エピック

認証情報の設定

タスク	説明	必要なスキル
アプリケーションサーバーにアクセスするためのシークレットを作成します。	ワーカーマシンからリモートでアプリケーションサーバーにアクセスするには、AWS Secrets Manager でシークレットを作成します。シークレットには、SSH プライベートキー、または証明書と SSH プライベートキーを使用できます。詳細については、「AWS App2Container シークレットの管理」を参照してください。	DevOps、開発者

ワーカーマシンのセットアップ

タスク	説明	必要なスキル
tar ファイルをインストールします。	`sudo yum install -y tar` を実行します。	DevOps、開発者
AWS CLI をインストールします。	Amazon コマンドラインインターフェイス (CLI) をインストールするには、`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"` を実行します。 `awscliv2.zip` を解凍します。 `sudo ./aws/install` を実行します。	DevOps、開発者
App2 コンテナをインストールします。	以下の コマンドを実行します。 `curl -o AWSApp2Container-installer-linux.tar.gz https://app2container-release-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/AWSApp2Container-installer-linux.tar.gz`	DevOps、開発者

タスク	説明	必要なスキル

	`sudo tar xvf AWSApp2Container-installer-linux.tar.gz`
	
	`sudo ./install.sh`	

プロファイルを設定します。	AWS のデフォルトプロファイルを設定するには、`sudo aws configure` を実行します。 名前付きの AWS デフォルトプロファイルを設定するには、`sudo aws configure --profile <profile name>` を実行します。	DevOps、開発者

Docker をインストールします。	以下のコマンドを実行します。
	
	`sudo yum install -y docker`
	
	`sudo systemctl enable docker & sudo systemctl restart docker`	

タスク	説明	必要なスキル
App2 コンテナを初期化します。	App2Container を初期化するには、次の情報が必要です。 • `workspace` :アプリケーションのコンテナ化アーティファクトを保存します。少なくとも 20 GB の空きディスク容量があるディレクトリパスを指定することをお勧めします。 • `awsProfile` :サーバに設定されている AWS プロファイル。これは、Amazon S3 にアーティファクトをアップロードし、`containerize` コマンドを実行し、Amazon ECS または Amazon EKS にデプロイするための AWS アーティファクトを生成するために必要です。 • `s3Bucket`: AWS Artifact を抽出して保存する方法。 • `metricsReportPermission` :報告されたメトリックスを収集して保存する。 • `dockerContentTrust` : Docker イメージに署名します。 `sudo app2container init` を実行します。	DevOps、開発者

ワーカーマシンの設定

タスク	説明	必要なスキル
アプリケーションサーバーに App2Container コマンドをリモート接続して実行するようにワーカーマシンを設定します。	ワーカーマシンを設定するには、以下の情報が必要です。 • `Server FQDN` :アプリケーションサーバーの完全修飾ドメイン名。 • `Server IP address` :アプリケーションサーバーの IP アドレス。FQDN または IP アドレスのどちらかで十分です。 • `SecretARN` : Secrets Manager に保存されている、アプリケーションサーバーへの接続に使用されるシークレットの Amazon リソースネーム (ARN)。 • `AuthMethod` : `key` または `cert` 認証方法。 `sudo app2container remote configure` を実行します。	DevOps、開発者

ワーカーマシン上のアプリケーションの検出、分析、抽出

タスク	説明	必要なスキル
オンプレミスの Java アプリケーションをご覧ください。	アプリケーションサーバーで実行されているすべてのアプリケーションをリモートで検	開発者、DevOps

タスク	説明	必要なスキル
	出するには、次のコマンドを実行します。 `sudo app2container remote inventory --target <FQDN/IP of App server>` このコマンドは、`inventory.json` にデプロイされたアプリケーションのリストを生成します。	
検出されたアプリケーションを分析します。	インベントリ段階で取得した application-id を使用して各アプリケーションをリモート分析するには、以下のコマンドを実行します。 `sudo app2container remote analyze --application-id <java-app-id> --target <FQDN/IP of App Server>` これにより、ワークスペースの場所に `analysis.json` ファイルが生成されます。このファイルが生成されたら、必要に応じてコンテナ化パラメータを変更できます。	開発者、DevOps

タスク	説明	必要なスキル
分析したアプリケーションを抽出します。	分析したアプリケーションのアプリケーションアーカイブを生成するには、次のコマンドをリモートで実行します。これにより、ワークスペースの場所に tar バンドルを生成します。 `sudo app2container remote extract --application-id <application id> --target <FQDN/IP of App Server>` 抽出されたアーティファクトはローカルワーカーマシン上で生成できます。	開発者、DevOps

抽出したアーティファクトをワーカーマシンにコンテナ化します。

タスク	説明	必要なスキル
抽出したアーティファクトをコンテナ化する。	次のコマンドを実行して、前のステップで抽出したアーティファクトをコンテナ化します。 `sudo app2container containerize --input-archive <tar bundle location on worker machine>`	開発者、DevOps

タスク	説明	必要なスキル
ターゲットを確定する。	ターゲットを確定するには、containerize コマンドの実行時に作成される deployment.json を開きます。AWS Fargate をターゲットとして指定するには、createEcsArtifacts を true に設定します。Amazon EKS をターゲットとして設定するには、createEksArtifacts を true に設定します。	開発者、DevOps

AWS Artifact の生成とプロビジョニング

タスク	説明	必要なスキル
ワーカーマシンに AWS デプロイアーティファクトを生成します。	デプロイメントアーティファクトを生成するには、以下のコマンドを実行します。 `sudo app2container generate app-deployment --application-id <application id>` これにより、ワークスペースに ecs-master.yml AWS CloudFormation テンプレートが生成されます。	DevOps
アーティファクトをプロビジョニングします。	生成されたアーティファクトをさらにプロビジョニングするには、次のコマンドを実行して AWS CloudFormation テ	DevOps

タスク	説明	必要なスキル
	ンプレートをデプロイします。 `aws cloudformation deploy --template-file <path to ecs-master.yml> --capabilities CAPABILITY_NAMED_IAM --stack-name <application id>-ECS`	
パイプラインを生成する。	前のストーリーで作成した `pipeline.json` を、必要に応じて変更します。次に、`generate pipeline` コマンドを実行してパイプラインデプロイアーティファクトを生成します。	DevOps

関連リソース

- 「App2Container とは?」
- 「AWS App2Container に関するブログ投稿」
- 「AWS CLI 設定の基本」
- 「Amazon ECS 用 Docker の基本」
- 「Docker コマンド」

AWS の大規模移行における共有ファイルシステムの移行

アミット・ルドララジュ（AWS）、サム・アパ（AWS）、ビーメスワラオ・バラ（AWS）、ウォーリー・ルー（AWS）、サンジーブ・プラカサム（AWS）によって作成された

概要

300 台以上のサーバを移行することは「大規模な移行」と見なされます。大規模な移行の目的は、ワークロードを既存のオンプレミスデータセンターから AWS クラウドに移行することであり、これらのプロジェクトは通常、アプリケーションとデータベースのワークロードに焦点を当てています。ただし、共有ファイルシステムには細心の注意と個別の移行計画が必要です。このパターンは、共有ファイルシステムの移行プロセスを説明し、大規模な移行プロジェクトの一環としてそれらを正常に移行するためのベストプラクティスを提供します。

「共有ファイルシステム」(SFS) は、「ネットワーク」または「クラスター」ファイルシステムとも呼ばれ、複数のサーバーにマウントされるファイル共有です。共有ファイルシステムには、ネットワークファイルシステム (NFS)、共通インターネットファイルシステム (CIFS)、サーバーメッセージブロック (SMB) などのプロトコルでアクセスします。

これらのシステムは、移行するホスト専用でもブロックデバイスとしても表示されないため、AWS Application Migration Service などの標準の移行ツールでは移行されません。ほとんどのホストの依存関係は透過的に移行されますが、依存ファイルシステムの調整と管理は個別に処理する必要があります。

共有ファイルシステムの移行は、検出、計画、準備、切り取り、検証というフェーズで行います。このパターンと添付のワークブックでは、共有ファイルシステムを、Amazon Elastic File System (Amazon EFS)、Amazon EFS)、Amazon FSx for NetApp ONTAP、Amazon FSx for Windows File Server などの AWS ストレージサービスに移行します。ファイルシステムを転送するには、AWS DataSync または NetApp SnapMirror などのサードパーティツールを使用できます。

> **Note**
>
> このパターンは、AWS クラウドへの大規模な移行に関する AWS 規範ガイダンスシリーズの一部です。このパターンには、SFS をサーバーのウェーブプランに組み込むためのベストプラクティスと手順が含まれています。大規模な移行プロジェクトの外部で 1 つ以上の共有ファイルシステムを移行する場合は、「Amazon EFS」、「Amazon FSx for Windows File Server」、「Amazon FSx for NetApp ONTAP」の AWS ドキュメントのデータ転送手順を参照してください。

前提条件と制限

前提条件

前提条件は、ソースとターゲットの共有ファイルシステム、およびユースケースによって異なる場合があります。次は、最も一般的な問題を示しています。

- アクティブなAWS アカウント
- 大規模な移行プロジェクトのアプリケーションポートフォリオの発見が完了し、ウェーブプランの作成が開始されました。詳細については、「AWS の大規模な移行のためのポートフォリオプレイブック」を参照してください。
- オンプレミスデータセンターと AWS 環境間の送受信トラフィックを許可する仮想プライベートクラウド (VPC) とセキュリティグループ。詳細については、ネットワークから Amazon VPC への接続オプションおよびAWS DataSyncネットワーク要件を参照してください。
- AWS CloudFormation スタックを作成する権限、または Amazon EFS または Amazon FSx リソースを作成するための権限。詳細については、「CloudFormation ドキュメント」、「Amazon EFS ドキュメント」、または「Amazon FSx ドキュメント」を参照してください。
- AWS DataSync を使用して移行を実行する場合は、次の権限が必要です。
  - AWS DataSync がAWS CloudWatch Logs ロググループにログを送信する権限。詳細については、「DataSync がログを CloudWatch ロググループにアップロードすることを許可する」を参照してください。
  - CloudWatch Logs ロググループにアクセスするアクセス許可。詳細については、「CloudWatch イベントリソースへのアクセス許可の管理の概要」を参照してください。
  - DataSync でエージェントとタスクを作成するための権限。詳細については、[AWS DataSyncに必要なIAMアクセス許可] を参照してください。

制約事項

- このパターンは、大規模な移行プロジェクトの一環として SFS を移行するように設計されています。アプリケーション移行のウェーブプランに SFS を組み込む際のベストプラクティスと手順が記載されています。大規模な移行プロジェクトの外部で 1 つ以上の共有ファイルシステムを移行する場合は、「Amazon EFS」、「Amazon FSx for Windows File Server」、「Amazon FSx for NetApp ONTAP」の AWS ドキュメントのデータ転送手順を参照してください。
- このパターンは、一般的に使用されているアーキテクチャ、サービス、移行パターンに基づいています。ただし、大規模な移行プロジェクトや戦略は組織によって異なる場合があります。要件に基

づいて、このソリューションまたは提供されているワークブックをカスタマイズする必要がある場合があります。

アーキテクチャ

ソーステクノロジースタック

次の 1 つ以上。

- Linux (NFS) ファイルサーバー
- Windows (SMB) ファイルサーバー
- NetApp ストレージアレイ
- Dell EMC Isilon ストレージアレイ

ターゲットテクノロジースタック

次の 1 つ以上。

- Amazon Elastic File System
- Amazon FSx for NetApp ONTAP
- Amazon FSx for Windows File Server

ターゲット アーキテクチャ

図表に示す内容は以下のステップです。

1. AWS Direct Connect や AWS Site-to-Site VPN などの AWS サービスを使用して、オンプレミスデータセンターと AWS クラウド間の接続を確立します。
2. DataSync エージェントは、オンプレミスデータセンターでインストールします。
3. ウェーブプランによると、DataSync を使用してソース共有ファイルシステムからターゲットの AWS ファイル共有にデータを複製します。

移行フェーズ

次の図は、大規模な移行プロジェクトで SFS を移行するためのフェーズと大まかな手順を示しています。

このパターンの「エピック」セクションには、移行を完了して添付のワークブックを使用する方法の詳細な説明が記載されています。次に、この段階的アプローチの手順を示します。

[Phase] (フェーズ)	ステップ
検出	1. 検出ツールを使用して、サーバー、マウントポイント、IP アドレスなど、共有ファイルシステムに関するデータを収集します。 2. 構成管理データベース (CMDB) または移行ツールを使用して、移行ウェーブ、環境、アプリケーション所有者、IT サービス管理 (ITSM) サービス名、組織単位、アプリケーション ID などのサーバーに関する詳細を収集します。
計画	3. 収集した SFS とサーバーに関する情報を使用して SFS ウェーブプランを作成します。 4. ビルドワークシートの情報を使用して、各 SFS について、ターゲット AWS サービスと移行ツールを選択します。
準備	5. Amazon EFS、Amazon FSx for NetApp ONTAP、Amazon FSx for Windows ファイルサーバー、Amazon FSx for Windows File Server でターゲットインフラストラクチャを設定します。 6. DataSync などのデータ転送サービスを設定し、初期データ同期を開始します。初回の同期が完了したら、定期的な同期をスケジュールに従って実行するように設定できます。 7. SFS ウェーブプランを IP アドレスやパスなどのターゲットファイル共有に関する情報で更新します。

カットオーバー	8. ソース SFS にアクティブにアクセスしているアプリケーションを停止します。 9. データ転送サービスで、最終的なデータ同期を行います。 10. 同期が完了したら、CloudWatch Logs のログデータを確認して、同期が完全に正常に行われたことを確認します。
検証	11. サーバーで、マウントポイントを新しい SFS パスに変更します。 12. アプリケーションを再起動して検証します。

ツール

AWS サービス

- Amazon CloudWatch Logs は、すべてのシステム、アプリケーション、 からのログを一元化するのに役立ちます。一元化により、ログを監視して安全にアーカイブできます。
- 「AWS DataSync」は、ファイルまたはオブジェクトデータを AWS ストレージサービス間で移動したり、AWS ストレージサービス間で移動したりするのに役立つオンラインデータ転送および検出サービスです。
- Amazon Elastic File System (Amazon EFS) は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。
- 「Amazon FSx」は、業界標準の接続プロトコルをサポートし、AWS リージョン全体で高い可用性とレプリケーションを提供するファイルシステムを提供します。

その他のツール

- 「SnapMirror」は、指定されたソースボリュームまたは qtree からターゲットボリュームまたは「qtree」にそれぞれデータを複製するネットアップのデータ複製ツールです。このツールを使用して、ネットアップのソースファイルシステムを Amazon FSx for ONTAP に移行できます。
- 「Robocopy」は、「堅牢なファイルコピー」の略で、Windows 用のコマンドラインディレクトリおよびコマンドです。このツールでは、Amazon FSx for Windows File Server に移行できます。

ベストプラクティス

ウェーブプランニングアプローチ

大規模な移行プロジェクトのウェーブを計画するときは、レイテンシーとアプリケーションパフォーマンスを考慮してください。SFS と依存アプリケーションがクラウドとオンプレミスのデータセンターなど、異なる場所で動作している場合、レイテンシーが増加し、アプリケーションのパフォーマンスに影響する可能性があります。ウェーブプランを作成する際に使用できるオプションは以下のとおりです。

1. SFS とすべての依存サーバーを同じウェーブ内で移行 — この方法では、パフォーマンスの問題が回避され、マウントポイントを複数回再構成するなどのやり直しが最小限に抑えられます。アプリケーションと SFS 間のレイテンシーを非常に低く抑える必要がある場合に推奨されます。ただし、ウェーブプランニングは複雑で、その目的は通常、依存関係のグループから変数を削除することであり、追加することではありません。また、この方法は多数のサーバーが同じ SFS にアクセスする場合にはお勧めできません。ウェーブが大きくなりすぎるためです。
2. 最後に依存するサーバーを移行した後で SFS を移行 — たとえば、複数のサーバーが SFS にアクセスしていて、それらのサーバーが第 4 波、第 6 波、第 7 波に移行する予定の場合は、SFS を第 7 段階に移行するようにスケジュールします。

   多くの場合、この方法は大規模な移行には最も論理的であり、遅延の影響を受けやすいアプリケーションには推奨されます。これにより、データ転送に関連するコストが削減されます。また、高層アプリケーションは通常、開発アプリケーションと QA アプリケーションの後に最後に移行するようにスケジュールされているため、SFS と上位層 (本番環境など) アプリケーション間の遅延時間も最小限に抑えられます。

   ただし、このアプローチにはやはり検出、計画、俊敏性が必要です。早い段階で SFS を移行する必要があったかもしれません。最初の依存ウェーブから SFS を含むウェーブまでの間、アプリケーションが追加のレイテンシーに耐えられることを確認します。アプリ所有者とディスカバリーセッションを実施し、レイテンシーの影響を最も受けやすいアプリケーションと同じウェーブにアプリケーションを移行します。依存アプリケーションの移行後にパフォーマンスの問題が発見された場合は、SFS をできるだけ早く移行できるよう迅速に方向転換する準備をしてください。
3. 大規模な移行プロジェクトの終了時に SFS を移行 — SFS 内のデータへのアクセス頻度が低い場合や、アプリケーションのパフォーマンスにとって重要ではない場合など、遅延が要因ではない場合は、この方法が推奨されます。このアプローチにより、移行が効率化され、カットオーバー作業が簡単になります。

アプリケーションのレイテンシー感度に基づいて、これらの方法を組み合わせることができます。たとえば、アプローチ 1 または 2 を使用してレイテンシーの影響を受けやすい SFS を移行し、方法 3 を使用して残りの SFS を移行できます。

AWS ファイルシステムサービスの選択

AWS はファイルストレージ用のクラウドサービスをいくつか提供しています。パフォーマンス、スケール、アクセシビリティ、統合、コンプライアンス、コスト最適化について、それぞれに異なる利点と制限があります。論理的なデフォルトオプションがいくつかあります。たとえば、現在のオンプレミスファイルシステムが Windows Server を運用している場合、Amazon FSx for Windows File Server がデフォルトの選択肢です。または、オンプレミスのファイルシステムがNetApp ONTAPを実行している場合は、Amazon FSx for NetApp ONTAPがデフォルトの選択肢です。ただし、アプリケーションの要件に基づいてターゲットサービスを選択したり、クラウド運用上のその他のメリットを実現したりすることもできます。詳細については、「デプロイメントに適した AWS ファイルストレージサービスの選択 (AWS Summit プレゼンテーション)」を参照してください。

移行ツールの選択

Amazon EFS と Amazon FSx は、AWS DataSync を使用して共有ファイルシステムを AWS クラウドに移行することをサポートしています。サポートされているストレージシステムとサービス、利点、ユースケースの詳細については、「AWS DataSync とは」を参照してください。DataSync を使用してファイルを転送するプロセスの概要については、「AWS DataSync 転送の仕組み」を参照してください。

また、次のようなサードパーティ製ツールもいくつかあります。

- Amazon FSx for NetApp ONTAPを選択した場合は、NetApp SnapMirror を使用してオンプレミスのデータセンターからクラウドにファイルを移行できます。SnapMirror はブロックレベルのレプリケーションを使用します。これは DataSync よりも高速で、データ転送プロセスの所要時間を短縮できます。詳細については、「NetApp SnapMirror を使用した FSx for ONTAP への移行」を参照してください。
- Amazon FSx for Windows File Server を選択すると、Robocopy を使用してファイルをクラウドに移行できます。詳細は、「RRobocopy を使用して既存のファイルを FSx for Windows File Server へ移行する」を参照してください。

エピック

検出

タスク	説明	必要なスキル
SFS ディスカバリーワークブックを準備する。	1. このパターンの「添付ファイル」セクションにあるワークブックをダウンロードします。これには「SFS-Discovery-Workbook.xlsx」と「SFS-Wave-Plan-Workbook.xlsx」の 2 つのファイルが含まれています。 2. Microsoft Excel で「SFS ディスカバリーワークブック」ファイルを開きます。 3. [Dashboard (ダッシュボード)] で、以下の操作を実行できます。 • 「A」列にある環境名を更新します。 • 「B」 列では、環境の順序を更新して、優先度が低い (1) から高い優先度の順になります。 • 「D ~ E」列で、ウェーブスケジュールを更新します。 • 「C」と「K」列で、AWS アカウント名を更新します。 • 「L」列の VPC ID を更新します。	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	• 「M—O」列で、サブネット ID を更新します。 4. ワークブックテンプレートの残りの部分を確認し、組織やユースケースに必要なその他の値を更新します。 5. ワークブックを保存します。	

タスク	説明	必要なスキル
ソース SFS に関する情報を収集します。	1. お好みの検出ツールを使用して、該当するすべてのストレージデバイス、Linux サーバー、Windows サーバーのすべての SFS マウントを特定します。以下の情報を収集するには以下の情報が必要です。 • クライアントデバイス • クライアント IP アドレス • SFS の詳細 • マウントポイント **Note** 移行ランブックにマウントポイントの詳細を追加して、移行後に SFS を再マウントできます。 2. 「SFS ディスカバリーワークブック」ファイルを開きます。 3. 「Wave-Sheet」ワークシートで、次の操作を行います。 • 数式の「サーバーの場所」 (D) 列で、オンプレミスソースの CIDR 範囲の形式が自分の範囲に合っていることを確認します。たとえば、FQDN	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	が 10.0.0.0/8 の場合は、10.*.*.* と入力します。 • 数式の「SFS ロケーション」 (E) 列で、ターゲット VPC の CIDR 範囲の形式が自分の範囲に合っていることを確認します。たとえば、FQDN が 176.16.0.0/16 の場合は、176.16.*.* と入力します。 4. 「SFS データ」ワークシートで、次の操作を行います。 • 「サーバー名」(A)列に、SFS がマウントされているサーバーの名前を入力します。 • 「SFS パス」 (B) 列に SFS の名前を入力します。 • 「IP アドレス」 (C) 列に、サーバーの IP アドレスを入力します。 • マウントポイントや SFS サイズなど、検出中に収集したその他の関連情報を追加します。このデータを後でウェーブプランニングの計算に変更できます。	

タスク	説明	必要なスキル
	5. ワークブックを保存します。	

タスク	説明	必要なスキル
サーバーに関する情報を収集します。	1. CMDB または移行ツールに記録されたデータを使用して、SFS マウントのあるサーバーに関する次の情報をすべて特定します。 • [Server name] (サーバー名) • IP アドレス • ウェーブ • 組織単位 (OU) • DEV、QAやPRODなどのサーバ環境 • アプリケーション名 • アプリ所有者と連絡先情報 2. 「SFS ディスカバリーワークブック」ファイルを開きます。 3. 「サーバーデータ」ワークシートの「A ~ H」列に、ソースサーバーについて収集した情報を入力します。次の点に注意してください: • 「Wave #」(C) 列に、ウェーブ名 (Wave1など)、範囲外 ()、OOSまたはRetireを入力します。 • 「アプリ所有者の連絡先」 (H)] 列の場合は、メールアドレスが正しいことを確認します。この	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	メールアドレスは、「アプリ所有者 」(G) 列に入力した名前に基づいて自動的に生成されます。必要に応じて、正しいメールアドレスを反映するように値を手動で更新してください。 • 数式を含む「I ~ J」列は変更しないでください。 4. ワークブックを保存します。	

計画

タスク	説明	必要なスキル
SFS ウェーブプランを構築してください。	1. 「SFS ディスカバリーワークブック」ファイルを開きます。 2. 検出フェーズで収集された情報がすべて正確かつ最新であることを確認します。 3. 「Wave-Sheet」ワークシートで、「SFS wave」(K) 列を値1に基づいてフィルタリングします。これは最初のウェーブに含まれるすべての SFS のリストです。 **Note** この列0の の値は、SFS が移行の	ビルドリード、カットオーバーリード、マイグレーションエンジニア、マイグレーションリード

タスク	説明	必要なスキル
	範囲外であることを示します。これは、SFS が既に AWS でホストされているか、共有にアクセスするサーバーが移行の対象外であるためと考えられます。 4. このウェーブでこれらの SFS を移行することを確認してください。SFS をウェーブに割り当てる方法の詳細については、「ベストプラクティス」セクションの「ウェーブプランニングアプローチ」を参照してください。 5. フィルターされた値を含むセルを選択してコピーします。列タイトルを含むヘッダー行はコピーしないでください。 6. 以前にダウンロードした「SFS-Wave-Plan-ワークブック」ファイルを開きます。 7. 「ディスカバリーからのエクスポート」ワークシートで、セル「A2」を選択します。 8. コピーしたデータを貼り付けます。	

タスク	説明	必要なスキル
	9. 「SFS-ディスカバリー・ワークブック」と「SFS-Wave-Plan-ワークブック」ファイルを保存します。	

タスク	説明	必要なスキル
対象の AWS サービスと移行ツールを選択します。	1. 「SFS-Wave-Plan-ワークブック」ファイルの「ディスカバリーからエクスポート」ワークシートで、「古いパス」(C) 列の値を選択してコピーします。 2. 「Build-Wave」ワークシートで、セル「A2」を選択します。 3. コピーしたデータを貼り付けます。このワークシートの列 B ~ M は、このパスに関連する他のデータを反映するように自動的に更新されます。 4. 「A」列の重複する値をすべて削除します。 手順については、「重複する値の削除 (Microsoft Support Web サイト)」を参照してください。 5. 「ターゲットパターンまたはサービス」(F) 列で、推奨ターゲット AWS サービスを確認し、必要に応じて更新します。詳細については、このパターンの「ベストプラクティス」セクションの「AWS ファイルシステムサービスの選択」を参照してください。 6. 「移行方法 (G)」列で、推奨する移行ツールを確認	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	し、必要に応じて更新してください。詳細については、このパターンの「ベストプラクティス」セクションにある「移行ツールの選択」を参照してください。 7. 「SFS-Discovery-ワークブック」ファイルを保存します。このウェーブのウェーブプランの作成が完了しました。 8. これらの手順を繰り返して、ウェーブごとにウェーブプランを作成します。ウェーブプランは移行中に変更される可能性があるため、事前に計画するのは 5 ウェーブまでにすることをお勧めします。	

準備

タスク	説明	必要なスキル
ターゲットファイルシステムを設定します。	ウェーブプランに記録された詳細に従って、ターゲット AWS アカウント、VPC、サブネットにターゲットファイルシステムを設定します。手順については、次の AWS ドキュメントを参照してください。 • Amazon EFS	移行エンジニア、移行リーダー、AWS 管理者

タスク	説明	必要なスキル
	• Amazon FSx for NetApp ONTAP • Amazon FSx for Windows File Server	

タスク	説明	必要なスキル
移行ツールをセットアップし、データを転送します。	1. AWS DataSync を使用している場合は、DataSync タスクのロギングを設定します。手順については、「AWS DataSync タスクアクティビティのロギング」を参照してください。 2. 移行ツールを設定し、選択したツールの指示に従って初期データ転送を実行します。 • Amazon EFS については、以下を参照してください。 • AWS DataSync を使用してファイルを EFS を転送する • ONTAP 用 Amazon FSx については、以下を参照してください。 • NetApp SnapMirror を使用した FSx for ONTAP への移行 • AWS DataSyncを使用した FSx for ONTAP への移行 • Amazon FSx for Windows File Server については、以下を参照してください。 • AWS DataSync を使用して、既存のファイルを FSx for Windows	AWS 管理者、クラウド管理者、移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	File Server への移行する • Robocopy を使用して、既存のファイルを FSx for Windows File Server に移行する 3. ソース SFS への変更は、最初の転送中または転送後に発生する可能性があります。データの同期を維持するために、ソースとターゲットのファイルシステムの間で定期的なデータ転送を設定します。 • DataSync を使用している場合は、「AWS DataSync タスクのスケジュール設定」を参照してください。DataSync は、ソース SFS 内の変更されたファイルまたは新しいファイルのみを転送します。 • サードパーティ製ツールを使用している場合は、選択したツールのドキュメントを参照してください。	

タスク	説明	必要なスキル
ウェーブプランを更新してください。	1. 現在のウェーブの「SFS-Wave-Plan-ワークブック」ファイルを開きます。 2. 「Build-Wave」ワークシートの「新規パス IP アドレス (N)」列に、ターゲットファイルシステムの IP アドレスを入力します。IP アドレスを検索するには、次のいずれかを実行します。 • Windows File Server 用 FSx の場合、Amazon FSx コンソールで「ファイルシステム」を選択し、ファイルシステムを選択して、「ネットワークとセキュリティ」セクションを表示します。 • FSx for ONTAP については、「マウントボリューム」を参照してください。 • Amazon EFS については、「IP アドレスによるマウント」を参照してください。 3. 「新しいパス (O)」列に、新しいマウントパスを入力します。マウントパスは、ファイルシステムの DNS 名です。マウントパスを特定するには、次のいずれかの操作をします。	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	• FSx for Windows File Server の場合、Amazon FSx コンソールで「ファイルシステム」を選択し、ファイルシステムを選択してから「アタッチ」を選択します。 • FSx for ONTAP については、「ファイルシステムの詳細」ページを参照してください。手順については、「マウントボリューム」を参照してください。 • 詳細については、「Amazon EFS」を参照してください。 4. 「再マウントの概要」ワークシートで、「新しいパス (C)」列と「新しいパス IP アドレス (D)」列に更新された値が反映されていることを確認します。 5. 組織が Linux と Windows のファイルシステムをカットオーバー後に再マウントするためのランブックを用意していることを確認します。一般的な手順については、以下を参照してください。 • EFS ファイルシステムをマウントする	

タスク	説明	必要なスキル
	• FSx for Windows File Server ファイル共有へアクセスする • FSx for ONTAP ボリュームのマウント 6. 依存サーバーがこの段階に含まれていない場合は、「App-Team-Communication」ワークシートに記録してください。標準のWave通信には含まれていない可能性があるため、それぞれのアプリケーションまたはサーバーの所有者に知らせてください。 7. ウェーブプランを完了した後に SFS がウェーブから削除された場合は、「Descoped」ワークシートで追跡してください。	

カットオーバー

タスク	説明	必要なスキル
アプリケーションの停止	アプリケーションまたはクライアントがソース SFS で読み取り/書き込み操作をアクティブに行っている場合は、最終的なデータ同期を実行する前にそれらを停止してください。手順については、アプリケーションのマニュアル	アプリ所有者、アプリ開発者

タスク	説明	必要なスキル
	を参照するか、読み取り/書き込みアクティビティを停止するための内部プロセスを参照してください。たとえば、「Web サーバーの起動または停止 (IIS 8) (Microsoft ドキュメント)」または「systemctl によるシステムサービスの管理」(Red Hat ドキュメント) を参照してください。	
最後のデータ転送を行います。	1. 移行ツールで、最後のデータ転送タスクまたはジョブを手動で実行して、ターゲットファイルシステムをソース SFS と同期させます。手順については、「DataSync タスクの開始」を参照するか、選択したサードパーティの移行ツールのドキュメントを参照してください。 2. 転送が完了するまで待つ 詳細については、「Amazon CloudWatch による AWS DataSync アクティビティのモニタリング」および「コマンドラインからの DataSync タスクのモニタリング」を参照してください。	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
データ転送を検証します。	AWS DataSync を使用している場合は、以下を実行して最終的なデータ転送が正常に完了したことを確認します。 1. AWS DataSync コンソールで、タスクと実行 ID (`task-0000-exec-1111` など) をメモします。 2. DataSync タスクの「タスクログ」セクションに移動します。 3. 「CloudWatch ロググループ」リンクを選択します。 4. ログで、タスクと実行 ID を検索します。 5. 転送エラーがあれば書き留めておきます。詳細については、DataSync のドキュメントの「一般的なエラー」を参照してください。 6. 以下を確認してください。 • ソース SFS とターゲット SFS のファイルリストを比較して、すべてのデータが転送されたことを確認します。 • ソース SFS とターゲットの SFS のファイルアクセス権限を比較します。	移行エンジニア、移行リーダー

タスク	説明	必要なスキル
	サードパーティツールを使用している場合は、選択した移行ツールのドキュメントにあるデータ転送検証手順を参照してください。	

検証

タスク	説明	必要なスキル
ファイルシステムを再マウントし、アプリケーションの機能とパフォーマンスを検証します。	1. この段階で依存サーバーを移行した場合は、「SFS-Wave-Plan-Workbook」ファイルの「再マウントの概要」ワークシートの「新規サーバーIPアドレス (F)」列にサーバーの「新しい IP アドレス」を入力します。 2. すべてのサーバーで、ファイルシステムのマウントポイントを古いパスから新しいパスに更新します。「準備」フェーズで説明した再マウントには、組織のランブックを使用してください。 3. マウントを確認し、ファイルが存在することを確認して、ファイルシステムが正しくマウントされ、アクセス可能であることを確認します。通常、インフラストラクチャー・チームがこれらの作業を行います。	AWS システム管理者、アプリ所有者

タスク	説明	必要なスキル
	4. アプリケーションを再起動し、アプリケーション所有者または QA チームに依頼して、アプリケーションの必要に応じてアプリケーションの機能テストとパフォーマンステストを完了させます。	

トラブルシューティング

問題	ソリューション
Microsoft Excel のセルの値は更新されません。	フィルハンドルをドラッグして、サンプル行の数式をコピーします。詳細については、「Windows」または「Mac」の説明書 (Microsoft Support Web サイト) を参照してください。

関連リソース

AWS ドキュメント

- AWS DataSync ドキュメント
- Amazon EFS ドキュメント
- Amazon FSx ドキュメンテーション
- AWS クラウドへの大規模な移行
  - AWS の大規模移行ガイド
  - AWS の大規模な移行のためのポートフォリオプレイブック

トラブルシューティング

- AWS DataSync の問題のトラブルシューティング

- Amazon EFS のトラブルシューティング
- Amazon FSx for Windows File Server のトラブルシューティング
- Amazon FSx for NetApp ONTAP のトラブルシューティング

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する

作成者: Dhairya Jindani (AWS)、Baji Shaik (AWS)

概要

Oracle GoldenGate は、異種データベースおよび IT 環境向けのリアルタイムのデータキャプチャおよびレプリケーションサービスです。ただし、このサービスは現在、Oracle 用 Amazon Relational Database Service (Amazon RDS) をサポートしていません。サポートされているデータベースのリストについては、異種データベース用 Oracle GoldenGate (Oracle ドキュメント) を参照してください。このパターンでは、Oracle GoldenGate および Oracle GoldenGate フラットファイルアダプタを使用して、ソース Oracle データベースからフラットファイルを生成する方法について説明します。このデータベースは、オンプレミスでも Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでもかまいません。その後、それらのフラットファイルを、Amazon RDS for Oracle データベースインスタンスにインポートできます。

このパターンでは、Oracle GoldenGate を使用してソース Oracle データベースからトレイルファイルを抽出します。データポンプは、証跡ファイルを Amazon EC2 インスタンスである統合サーバーにコピーします。統合サーバーでは、Oracle GoldenGate はフラットファイルアダプタを使用して、トレイルファイルのトランザクションデータキャプチャに基づいて一連の連続フラットファイルを生成します。Oracle GoldenGate は、データを区切り文字で区切られた値または長さで区切られた値としてフォーマットします。次に Oracle SQL*Loader を使用して、フラットファイルをターゲット Amazon RDS for Oracle データベースインスタンスにインポートします。

対象者

このパターンは、Oracle GoldenGate の基本的な構成要素に関する経験と知識があるユーザーを対象としています。詳細については、「Overview of the Oracle GoldenGate Architecture」(Oracle ドキュメント) を参照してください。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- Oracle GoldenGate ライセンス。
- Oracle GoldenGate アダプタの個別のライセンス。

- オンプレミスまたは Amazon EC2 インスタンスで実行されているソース Oracle データベース。
- 統合サーバーとして使用される Amazon EC2 Linux インスタンス。詳細については、「チュートリアル: Amazon EC2 Linux インスタンスの開始方法」(Amazon EC2 ドキュメント) を参照してください。
- ターゲットの Amazon RDS for Oracle DB インスタンスを停止します。詳細については、「Oracle DB インスタンスを作成して接続する」(Amazon RDS ドキュメント) を参照してください。

製品バージョン

- Oracle データベースエンタープライズエディションバージョン 10g、11g、12c、またはそれ以降
- Oracle GoldenGate バージョン 12.2.0.1.1 以降

アーキテクチャ

ソーステクノロジースタック

Oracle データベース (オンプレミスまたは Amazon EC2 インスタンス)

ターゲットテクノロジースタック

「Amazon RDS for Oracle」

ソースアーキテクチャとターゲットアーキテクチャ

1. Oracle GoldenGate は、ソースデータベースのログからトレイルを抽出します。
2. データポンプは証跡を抽出し、統合サーバーに移行します。
3. Oracle GoldenGate フラットファイルアダプタは、証跡、ソース定義、および抽出パラメータを読み取ります。
4. 抽出を終了すると、制御ファイルとフラットデータファイルが生成されます。
5. フラットデータファイルを の Amazon RDS for Oracle データベースインスタンスに移行します AWS クラウド。

ツール

AWS のサービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Oracle 用 Amazon Relational Database Service (Amazon RDS) は、 で Oracle リレーショナルデータベースをセットアップ、運用、スケーリングするのに役立ちます AWS クラウド。

その他のサービス

- Oracle GoldenGate は、あるデータベースから別の異種データベースまたはフラットファイルなどの別のターゲットトポロジへのデータの複製、フィルタリング、変換を支援するサービスです。
- Oracle GoldenGate アプリケーションアダプタを使用すると、Oracle GoldenGate はソースデータベースのトレイルファイルにキャプチャされたトランザクションデータから一連のシーケンシャルフラットファイルと制御ファイルを作成できます。これらのアダプタは、データウェアハウスアプリケーションや、独自仕様またはレガシーアプリケーションの抽出、変換、ロード (ETL) 操作に広く使用されています。Oracle GoldenGate はこのキャプチャを実行し、異種データベース、プラットフォーム、オペレーティングシステムにほぼリアルタイムで適用します。アダプタは、CSV や Apache Parquet など、さまざまな形式の出力ファイルをサポートします。生成されたこれらのファイルを読み込んで、データをさまざまな異種データベースにロードできます。

エピック

ソースデータベースサーバーに Oracle GoldenGate をセットアップします。

タスク	説明	必要なスキル
Oracle GoldenGate をダウンロードします。	ソースデータベースサーバーで、Oracle GoldenGate バージョン 12.2.0.1.1 以降をダウンロードします。手順については、「Downloading Oracle GoldenGate」(Oracle ドキュメント) を参照してください。	DBA
Oracle GoldenGate をインストールします。	手順については、「Installing Oracle GoldenGate」(Oracle	DBA

タスク	説明	必要なスキル
	ドキュメント) を参照してください。	
Oracle GoldenGate のセットアップ	手順については、「Preparing the Database for Oracle GoldenGate」(Oracle ドキュメント) を参照してください。	DBA

インテグレーションサーバーに Oracle GoldenGate をセットアップします。

タスク	説明	必要なスキル
Oracle GoldenGate をダウンロードします。	インテグレーションサーバーで、Oracle GoldenGate バージョン 12.2.0.1.1 以降をダウンロードします。手順については、「Downloading Oracle GoldenGate」(Oracle ドキュメント) を参照してください。	DBA
Oracle GoldenGate をインストールします。	異種環境用のディレクトリの作成、マネージャプロセスの設定、defgen ファイルの作成を行います。手順については、「Installing Oracle GoldenGate」(Oracle ドキュメント) を参照してください。	DBA

Oracle GoldenGate データキャプチャ設定を変更します。

タスク	説明	必要なスキル
Oracle GoldenGate アダプタを準備します。	インテグレーションサーバーで、Oracle GoldenGate アダ	DBA

タスク	説明	必要なスキル
	プタソフトウェアを設定します。次のコマンドを実行します。 1. Oracle Software Delivery Cloud から ggs_Adapters_Linux_x64.zip をダウンロードします。 2. ggs_Adapters_Linux_x64.zip を解凍します。 3. 次のコマンドを実行してアダプターをインストールします。 `tar -xvf ggs_Adapters_Linux_x64.tar`	
データポンプを設定します。	ソースサーバーで、トレイルファイルをソースサーバーから統合サーバーに転送するようにデータポンプを設定します。データポンプパラメータファイルとトレイルファイルディレクトリを作成します。手順については、「Configuring the Flat File Adapter」(Oracle ドキュメント) を参照してください。	DBA

フラットファイルの生成と移行

タスク	説明	必要なスキル
フラットファイルを生成します。	抽出ファイルと制御ファイルを作成し、統合サーバーで抽出プロセスを開始します。これにより、データベースの変更が抽出され、ソースデータベースがフラットファイルに書き込まれます。手順については、「Using the Flat File Adapter」(Oracle ドキュメント) を参照してください。	DBA
フラットファイルを、ターゲットデータベースにロードします。	フラットファイルを、ターゲットの Amazon RDS for Oracle データベースインスタンスにロードします。詳細については、「Oracle SQL*Loader を使用したインポート」(Amazon RDS ドキュメント) を参照してください。	DBA

トラブルシューティング

問題	ソリューション
Oracle GoldenGate フラットファイルアダプタがエラーを生成します。	アダプタエラーの説明については、「エラーメッセージの検索」(Oracle ドキュメント) を参照してください。トラブルシューティングの手順については、「Troubleshooting the Flat File Adapter」(Oracle ドキュメント) を参照してください。

関連リソース

- Installing Oracle GoldenGate (Oracle ドキュメント)
- Configuring Oracle GoldenGate (Oracle ドキュメント)
- Understanding Oracle GoldenGate Adapters (Oracle ドキュメント)
- Configuring the Flat File Adapter (Oracle ドキュメント)

Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションへのデータベース移行をサポートするように Python と Perl アプリケーションを変更する

作成者: Dwarika Patra (AWS)、Deepesh Jayaprakash (AWS)

概要

このパターンでは、Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションにデータベースを移行する際に、場合によっては必要になるアプリケーションリポジトリに変更します。このパターンでは、これらのアプリケーションが Python または Perl ベースであることを前提としており、これらのスクリプト言語には個別の指示が記載されています。

SQL Server データベースから Aurora PostgreSQL 互換に移行するには、スキーマ変換、データベースオブジェクト変換、データ移行、およびデータ読み込みが必要です。 PostgreSQL と SQL Server には (データ型、接続オブジェクト、構文、ロジックに関する) 違いがあるため、PostgreSQL で正しく動作するようにコードベースに必要な変更を追加するのが最も難しい移行作業です。

Python ベースのアプリケーションでは、接続オブジェクトとクラスはシステム全体に分散されています。また、Python コードベースは複数のライブラリを使用してデータベースに接続している場合があります。データベース接続インターフェースが変更された場合、アプリケーションのインラインクエリを実行するオブジェクトも変更する必要があります。

Perl ベースのアプリケーションの場合、変更には接続オブジェクト、データベース接続ドライバー、静的および動的なインライン SQL ステートメント、およびアプリケーションが複雑な動的 DML クエリと結果をセットで処理する方法が含まれます。

アプリケーションを移行する際には、FTP サーバーを Amazon Simple Storage Service (Amazon S3) アクセスに置き換えるなど、AWS で可能な機能強化を検討することもできます。

アプリケーションの移行プロセスには、以下の問題があります。

- 接続オブジェクト。接続オブジェクトが複数のライブラリや、関数呼び出しを含むコードに分散している場合、PostgreSQL をサポートするようにオブジェクトを変更する共通の方法を見つける必要があるかもしれません。
- レコードの取得または更新中のエラーや例外の処理。変数、結果セット、またはデータフレームを返す条件付きの作成、読み取り、更新、削除 (CRUD) 操作をデータベースで実行すると、エラーや例外によってアプリケーションエラーが発生し、連鎖的な影響が発生する可能性があります。これらの処理は、適切な検証を行いながら、ポイントを保存して慎重に処理する必要があります。このようなセーブポイントの 1 つとして、`BEGIN...EXCEPTION...END` ブロック内で大規模なインライン SQL クエリやデータベースオブジェクトを呼び出すことがあります。

- トランザクションの制御と検証。手動および自動のコミットとロールバックが含まれます。Perl 用の PostgreSQL ドライバーでは、常に自動コミット属性を明示的に設定する必要があります。
- 動的 SQL クエリを処理します。これには、クエリが意図したとおりに動作することを確実にするため、クエリロジックと反復テストを詳しく理解する必要があります。
- パフォーマンス コードを変更しても、アプリケーションのパフォーマンスが低下しないようにする必要があります。

このパターンでは、変換プロセスを詳しく説明します。

前提条件と制限

前提条件

- Python と Perl の構文に関する実用的な知識。
- SQL Server および PostgreSQL に関する基本スキル
- 既存のアプリケーションアーキテクチャを理解しましょう。
- アプリケーションコード、SQL Server データベース、および PostgreSQL データベースにアクセスできます。
- アプリケーションの変更を開発、テスト、検証するための認証情報を使用して Windows または Linux (または他の Unix) 開発環境にアクセスできます。
- Python ベースのアプリケーションでは、データフレームを処理するための Pandas や、データベース接続用の psycopg2 や SQLAlchemy など、アプリケーションが必要とする標準の Python ライブラリが必要です。
- Perl ベースのアプリケーションでは、依存ライブラリまたはモジュールを含む Perl パッケージが必要です。包括的な Perl Archive Network (CPAN) モジュールは、ほとんどのアプリケーション要件をサポートします。
- 必要なすべての依存関係のあるカスタマイズされたライブラリまたはモジュール。
- SQL Server への読み込みアクセスと Aurora への読み込み/書き込みアクセス用のデータベース認証情報。
- PostgreSQL は、サービスやユーザーによるアプリケーションの変更を検証およびデバッグします。
- アプリケーションの移行中に Visual Studio Code、Sublime Text、pgAdmin などの開発ツールにアクセスできます。

機能制限

- Python や Perl のバージョン、モジュール、ライブラリ、パッケージの中には、クラウド環境と互換性のないものがあります。
- SQL Server に使用されているサードパーティのライブラリやフレームワークの中には、PostgreSQL への移行をサポートするように置き換えられないものがあります。
- パフォーマンスの変動により、アプリケーション、インライン Transact-SQL (T-SQL) クエリ、データベース関数、ストアドプロシージャの変更が必要な場合があります。
- PostgreSQL は、テーブル名、列名、および他のデータベースオブジェクトの小文字名をサポートします。
- UUID 列などの一部のデータタイプは小文字のみで保存されます。Python と Perl のアプリケーションは、このような大文字と小文字の違いを処理する必要があります。
- 文字エンコーディングの違いは、PostgreSQL データベース内の対応するテキスト列の正しいデータタイプで処理する必要があります。

製品バージョン

- Python 3.6 以降 (オペレーティングシステムをサポートするバージョンを使用してください)
- Perl 5.8.3 以降 (オペレーティングシステムをサポートするバージョンを使用してください)
- Aurora PostgreSQL 互換エディション 4.2 以降 (「詳細」を参照してください)

アーキテクチャ

ソーステクノロジースタック

- スクリプト作成 (アプリケーションプログラミング) 言語: Python 2.7 以降または Perl 5.8
- データベース: Microsoft SQL Server バージョン 13
- オペレーティングシステム: Red Hat Enterprise Linux (RHEL) 7

ターゲットテクノロジースタック

- スクリプト作成 (アプリケーションプログラミング) 言語:Python 3.6 以降または Perl 5.8 以降
- データベース: Aurora PostgreSQL 互換 4.2
- オペレーティングシステム: RHEL 7

移行アーキテクチャ

ツール

AWS のサービスとツール

- Aurora PostgreSQL 互換エディションは、フルマネージド型で PostgreSQL 互換の、ACID 準拠のリレーショナルデータベースエンジンです。ハイエンドの商用データベースのスピードと信頼性を、オープンソースデータベースのコスト効率でご利用いただけます。PostgreSQL を Aurora PostgreSQL に差し替えることで、新規および既存の PostgreSQL のデプロイを簡単に、コスト効率よく設定、操作、スケーリングできるようになります。
- AWS コマンドラインインターフェイス (AWS CLI) はオープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

その他のツール

- psycopg2 や SQLAlchemy などの Python および PostgresSQL データベース接続ライブラリ
- Perl とその DBI モジュール
- PostgreSQL インタラクティブターミナル (psql)

エピック

アプリケーションリポジトリを PostgreSQL に移行する — 大まかな手順

タスク	説明	必要なスキル
以下のコード変換手順に従って、アプリケーションを PostgreSQL に移行してください。	1. PostgreSQL 用のデータベース固有の ODBC ドライバーとライブラリを設定します。たとえば、Perl と pyodbc 用の CPAN モジュール、psycopg2、または Python 用の SQLAlchemy のいずれかを使用できます。 2. これらのライブラリを使用して Aurora PostgreSQL 互	アプリ開発者

タスク	説明	必要なスキル
	換に接続することにより、データベースオブジェクトを変換できます。 3. 既存のアプリケーションモジュールのコード変更を適用して、互換性のある T-SQL ステートメントを取得します。 4. データベース固有の関数呼び出しとストアドプロシージャをアプリケーションコードに書き換えます。 5. インライン SQL クエリに使用されるアプリケーションの変数とそのデータ型の変更を処理します。 6. 互換性のないデータベース固有の関数を処理します。 7. データベース移行のために、変換されたアプリケーションコードのエンドツーエンドのテストを完了します。 8. Microsoft SQL Server の結果を PostgreSQL に移行したアプリケーションと比較します。 9. Microsoft SQL Server と PostgreSQL の間でアプリケーションパフォーマンスのベンチマークを実行します。	

タスク	説明	必要なスキル
	10.アプリケーションが呼び出すストアドプロシージャまたはインライン T-SQL ステートメントを変更して、パフォーマンスを向上させます。 以下のエピックでは、Python および Perl アプリケーションの変換タスクの一部について詳しく説明します。	

タスク	説明	必要なスキル
移行の各ステップでチェックリストを使用します。	アプリケーション移行の各手順 (最終ステップを含む) のチェックリストに以下の項目を追加してください。 • PostgreSQL のドキュメントを参照して、すべての変更に PostgreSQL 標準との互換性があることを確認してください。 • 列の整数値と浮動小数点値を確認します。 • 挿入、更新、抽出された行の数と、列名、日付/タイムスタンプを確認します。差分ユーティリティを使用するか、スクリプトを作成してこれらのチェックを自動化します。 • 大規模なインライン SQL ステートメントのパフォーマンスチェックを行い、アプリケーション全体のパフォーマンスをチェックします。 • 複数の try/catch ブロックを使用して、データベース操作の正しいエラー処理や、プログラムの正常な終了を確認します。 • ログ記録の処理が適切に行われていることを確認してください。	アプリ開発者

アプリケーションの分析と更新 — Python コードベース

タスク	説明	必要なスキル
既存の Python コードベースを分析します。	アプリケーションの移行プロセスを円滑に進めるため、以下の項目を分析に含める必要があります。 • コード内の接続オブジェクトをすべて特定します。 • 互換性のないインライン SQL クエリ (T-SQL ステートメントやストアドプロシージャなど) をすべて特定し、必要な変更を分析します。 • コードのドキュメントを確認し、制御フローをトレースしてコードの機能を理解しましょう。これは、後でアプリケーションのパフォーマンステストや負荷を比較するときに役立ちます。 • アプリケーションの目的を理解しておくと、データベース変換後に効果的にテストできます。　データベース移行による変換の候補となるほとんどの Python アプリケーションは、他のソースからデータベーステーブルにデータを読み込むフィード、またはテーブルからデータを取得して、	アプリ開発者

タスク	説明	必要なスキル
	レポートの作成や検証を実行するための API 呼び出しに適したさまざまな出力形式 (CSV、JSON、フラットファイルなど) に変換するエクストラクターのいずれかです。	

タスク	説明	必要なスキル

PostgreSQL をサポートするようにデータベース接続を変換します。	ほとんどの Python アプリケーションは、次のように pyodbc x ライブラリを使用して SQL Server データベースに接続します。
	
	`import pyodbc`
	`....`
	`try:`
	`conn_string = "Driver=ODBC Driver 17 for SQL Server;UID={};PWD={};Server={};Database={}".format (conn_user, conn_password, conn_server, conn_database)`
	`conn = pyodbc.connect(conn_string)`
	`cur = conn.cursor()`
	`result = cur.execute(query_string)`
	`for row in result:`
	`print (row)`
	`except Exception as e:`
	`print(str(e))`
	
	PostgreSQL をサポートするようにデータベース接続を変換します。
	
	`import pyodbc`
	`import psycopg2`
	`....`
	`try:`
		アプリ開発者

タスク	説明	必要なスキル
	conn_string = 'postgresql+psycopg2://'+ conn_user+':'+conn_password+'@'+conn_server+'/'+conn_database conn = pyodbc.connect(conn_string, connect_args={'options':'-csearch_path=dbo'}) cur = conn.cursor() result = cur.execute(query_string) for row in result: print (row) except Exception as e: print(str(e))	

タスク	説明	必要なスキル
インライン SQL クエリを PostgreSQL に変更します。	インライン SQL クエリを PostgreSQL 互換の形式に変換します。たとえば、次の SQL Server クエリはテーブルから文字列を取得します。 `dtype = “type1” stm = ‘“SELECT TOP 1 searchcode FROM TypesTable (NOLOCK) WHERE code=”’ + “’” + str(dtype) + “’” # For Microsoft SQL Server Database Connection engine = create_engine(‘mssql+pyodbc:///?odbc_connect=%s’ % urllib.parse.quote_plus(conn_string), connect_args={‘connect_timeout’:login_timeout}) conn = engine_connect() rs = conn.execute(stm) for row in rs: print(row)`  変換後、PostgreSQL 互換のインライン SQL クエリは以下のようになります。  `dtype = “type1” stm = ‘“SELECT searchcode FROM TypesTable WHERE code=”’ + “’” + str(dtype) + “’ LIMIT 1”`	アプリ開発者

タスク	説明	必要なスキル
	`# For PostgreSQL Database Connection engine = create_engine(‘postgres+psycopg2://%s’ %conn_string, connect_args={‘connect_timeout’:login_timeout}) conn = engine.connect() rs = conn.execute(stm) for row in rs: print(row)`	

タスク	説明	必要なスキル

動的 SQL クエリの処理。	動的 SQL は、1 つのスクリプトに存在することも、複数の Python スクリプトに含めることもできます。　先ほどの例では、Python の文字列置換関数を使用して動的 SQL クエリを構築するために変数を挿入する方法を説明しました。別の方法として、該当する箇所に変数を使用してクエリ文字列を追加することもできます。
	
	次の例では、関数から返された値に基づいてクエリ文字列が作成されます。
	
	`query = '"SELECT id from equity e join issues i on e.permId=i.permId where e.id'"`
	`query += get_id_filter(ids) + " e.id is NOT NULL`
	
	このような動的クエリは、アプリケーションの移行中によく使用されます。以下のステップに従って、動的クエリの処理を行います。
	
	• 全体的な構文 (JOIN 句を含む SELECT ステートメントの構文など) をチェックします。
		アプリ開発者

タスク	説明	必要なスキル
	• クエリで使用している変数や列名 (i や id など) をすべて確認してください。 • クエリで使用されている関数、引数、戻り値 (get_id_filter およびその引数 ids など) を確認します。	

タスク	説明	必要なスキル

結果セット、変数、データフレームを処理します。	Microsoft SQL Server では、`fetchone()` または `fetchall()` などの Python メソッドを使用してデータベースから結果セットを取得します。`fetchmany(size)` を使用して、結果セットから返されるレコードの数を指定することもできます。これを実行するには、次の例に示すように、pyodbc 接続オブジェクトを使用します。
	
	pyodbc (Microsoft SQL Server)
	
	`import pyodbc`
	`server = 'tcp:myse`
	`rver.database.wind`
	`ows.net'`
	`database = 'exampledb'`
	`username = 'exampleu`
	`sername'`
	`password = 'examplep`
	`assword'`
	`conn = pyodbc.co`
	`nnect('DRIVER={ODB`
	`C Driver 17 for SQL`
	`Server};SERVER='+s`
	`erver+';DATABASE='`
	`+database+';UID='+`
	`username+';PWD='+`
	`password)`
	`cursor = conn.cursor()`
	`cursor.execute("SELECT`
	`* FROM ITEMS")`
	`row = cursor.fe`
	`tchone()`
	`while row:`
		アプリ開発者

タスク	説明	必要なスキル

	<pre> print(row[0])
	    row = cursor.fe
	tchone()</pre>
	Aurora では、PostgreSQL への接続や結果セットの取得などの同様のタスクを実行するには、psycopg2 または SQLAlchemy のどちらかを使用します。これらの Python ライブラリは、次の例に示すように、PostgreSQL データベースレコードを走査するための接続モジュールとカーソルオブジェクトを提供します。
	
	psycopg2 (Aurora PostgreSQL 互換)
	
	<pre>import psycopg2
	query = "SELECT * FROM
	 ITEMS;"
	//Initialize
	 variables
	host=dbname=user=
	password=port=sslm
	ode=connect_timeou
	t=""
	connstring = "host='{h
	ost}' dbname='{
	dbname}' user='{user}'
	 \
	password='{passw
	ord}'port='{port}'
	".format(host=host
	,dbname=dbname,\
	user=user,password=
	password,port=port)</pre>	

タスク	説明	必要なスキル
	conn = psycopg2. connect(connstring) cursor = conn.cursor() cursor.execute(query) column_names = [column[0] for column in cursor.description] print("Column Names: ", column_names) print("Column values: " for row in cursor: print("itemid :", row[0]) print("itemdescrpt ion :", row[1]) print("it emprice :", row[3]))	
	SQL Alchemy (Aurora PostgreSQL 互換) from sqlalchemy import create_engine from pandas import DataFrame conn_string = 'postgres ql://core:database @localhost:5432/ex ampledatabase' engine = create_en gine(conn_string) conn = engine.co nnect() dataid = 1001 result = conn.exec ute("SELECT * FROM ITEMS") df = DataFrame (result.fetchall())	

タスク	説明	必要なスキル

	`df.columns = result.keys()`
	`df = pd.DataFrame()`
	`engine.connect()`
	`df = pd.read_sql_query(sql_query, engine, coerce_float=False)`
	`print("df=", df)`	

移行中や移行後にアプリケーションをテストします。	移行後の Python アプリケーションのテストは継続的なプロセスです。移行には接続オブジェクトの変更 (psycopg2 または SQLAlchemy)、エラー処理、新機能 (データフレーム)、インライン SQL の変更、大容量レプリケーション機能 (COPY ではなく bcp)、および同様の変更が含まれるため、アプリケーションの移行中および移行後に慎重にテストする必要があります。以下の項目を確認してください。 • エラー条件と処理 • 移行後のレコードの不一致 • レコードの更新または削除 • アプリケーション実行の所要時間	アプリ開発者

アプリケーションの分析と更新 — Perl コードベース

タスク	説明	必要なスキル
既存の Perl コードベースを分析します。	アプリケーションの移行プロセスを円滑に進めるため、以下の項目を分析に含める必要があります。以下の項目を特定する必要があります。 • 任意の INI または設定ベースのコード • データベース固有の標準オープンデータベース接続 (ODBC) Perl ドライバー、または任意のカスタマイズされたドライバー • インラインクエリと T-SQL クエリに必要なコード変更 • さまざまな Perl モジュール間の相互作用 (たとえば、複数の機能コンポーネントによって呼び出されるまたは使用される単一の Perl ODBC 接続オブジェクト) • データセットと結果セットの処理 • 外部の依存 Perl ライブラリ • アプリケーションで使用されるすべての API • Perl バージョンとの互換性と Aurora PostgreSQL 互換性のあるドライバー互換性	アプリ開発者
Perl アプリケーションと DBI モジュールからの接続を	Perl ベースのアプリケーションは通常、Perl プログラミン	アプリ開発者

タスク	説明	必要なスキル
PostgreSQL をサポートするように変換します。	グ言語の標準データベースアクセスモジュールである Perl DBI モジュールを使用します。 同じ DBI モジュールを SQL Server 用と PostgreSQL 用の異なるドライバーで使用できます。 必要な Perl モジュール、インストール、およびその他の手順の詳細については、DBD::Pg のドキュメントを参照してください。次の例は、`examplete st-aurorapg-databa se.cluster-samplec lusture.us-east-.r ds.amazonaws.com` で Aurora PostgreSQL 互換に接続しています。 `#!/usr/bin/perl use DBI; use strict; my $driver = "Pg"; my $hostname = “examplet est-aurorapg-datab ase-sampleclusture .us-east.rds.amazo naws.com” my $dsn = "DBI:$driver: dbname = $hostname ;host = 127.0.0.1;port = 5432"; my $username = "postgres "; my $password = "pass123" ;`	

タスク	説明	必要なスキル
	`$dbh = DBI->connect("dbi:Pg:dbname=$hostname;host=$host;port=$port;options=$options", $username, $password, {AutoCommit => 0, RaiseError => 1, PrintError => 0} );`	

タスク	説明	必要なスキル

インライン SQL クエリを PostgreSQL に変更します。	アプリケーションには、SELECT、DELETE、UPDATE など、PostgreSQL でサポートされていないクエリ句を含むインライン SQL クエリが含まれている場合があります。たとえば、TOP や NOLOCK などのクエリキーワードは PostgreSQL ではサポートされていません。次の例では、TOP、NOLOCK、およびブール型変数の処理方法を示しています。
	
	In SQL Server:
	
	`$sqlStr = $sqlStr . "WHERE a.student_id in (SELECT TOP $numofRecords c_student_id \ FROM active_student_record b WITH (NOLOCK) \ INNER JOIN student_contributor c WITH (NOLOCK) on c.contributor_id = b.c_st)`
	
	PostgreSQL の場合は、次のように変換してください。
	
	`$sqlStr = $sqlStr . "WHERE a.student_id in (SELECT TOP $numofRecords c_student_id \ FROM active_student_record b INNER JOIN`
		アプリ開発者

タスク	説明	必要なスキル
	student_contributor c \ on c.contributor_id = b.c_student_contr_id WHERE b_current_1 is true \ LIMIT $numofRecords)"	

タスク	説明	必要なスキル
動的 SQL クエリと Perl 変数を処理します。	動的 SQL クエリは、アプリケーションのランタイム時に作成される SQL ステートメントです。これらのクエリは、アプリケーションのランタイムに特定の条件に応じて動的に構築されるため、特定の条件に依存するので、クエリの全文は実行時に把握できません。例としては、毎日上位 10 銘柄を分析する財務分析アプリケーションがあります。これらの株式は毎日変動します。SQL テーブルはトップパフォーマーに基づいて作成され、その値はランタイムになるまでわかりません。 この例のインライン SQL クエリは、変数に設定された結果を取得するためにラッパー関数に渡され、変数は条件を使用してテーブルが存在するかどうかを判断します。 • テーブルが存在する場合は作成せずに、何らかの処理を行ってください。 • テーブルが存在していない場合は、テーブルを作成し、処理を行います。 次に、変数処理の例を示します。次に、このユースケースの SQL Server クエリと	アプリ開発者

タスク	説明	必要なスキル

	PostgreSQL クエリを示します。
	
	`my $tableexists = db_read( arg 1, $sql_qry, undef, 'writer'); my $table_already_exists = $tableexists->[0]{table_exists}; if ($table_already_exists){ # do some thing } else { # do something else }`
	
	SQL Server:
	
	`my $sql_qry = “SELECT OBJECT_ID('$backendTable', 'U') table_exists", undef, 'writer') ";`
	
	PostgreSQL:
	
	`my $sql_qry = “SELECT TO_REGCLASS('$backendTable', 'U') table_exists", undef, 'writer')";`
	
	次の例では、インライン SQL の Perl 変数を使用して、SELECT と一緒に JOIN ステートメントを実行し、	

タスク	説明	必要なスキル

	テーブルのプライマリキーとキーカラムの位置を取得します。
	
	SQL Server:
	
	`my $sql_qry = "SELECT column_name', character_maximum_length \ FROM INFORMATION_SCHEMA.COLUMNS \ WHERE TABLE_SCHEMA='$example_schemaInfo' \ AND TABLE_NAME='$example_table' \ AND DATA_TYPE IN ('varchar','nvarchar');";`
	
	PostgreSQL:
	
	`my $sql_qry = "SELECT c1.column_name, c1.ordinal_position \ FROM information_schema.key_column_usage AS c LEFT \ JOIN information_schema.table_constraints AS t1 \ ON t1.constraint_name = c1.constraint_name \ WHERE t1.table_name = $example_schemaInfo'.'$example_table' \ AND t1.constraint_type = 'PRIMARY KEY' ;";`	

PostgreSQL をサポートするために、Perl ベースまたは Python ベースのアプリケーションに追加の変更を加えます。

タスク	説明	必要なスキル
他の SQL Server コンストラクトを PostgreSQL に変換します。	次の変更は、プログラミング言語にかかわらず、すべてのアプリケーションに適用されます。 • アプリケーションが使用するデータベースオブジェクトを、新しい適切なスキーマ名で修飾します。 • PostgreSQLの特徴量で大文字と小文字を区別してマッチングを行う LIKE 演算子を扱います。 • DATEDIFF、DATEADD、GETD 演算子など、サポートされていないデータベース固有の関数を処理します。同等の PostgreSQL 互換関数については、「追加情報」セクションの「ネイティブ SQL 関数または組み込み SQL 関数」を参照してください。 • 比較ステートメントのブール値を処理します。 • 関数からの戻り値を処理します。レコードセット、データフレーム、変数、ブール値などが該当します。 アプリケーションの要件と PostgreSQL をサ	アプリ開発者

タスク	説明	必要なスキル
	ポートするように処理してください。 • 新しいユーザー定義の PostgreSQL 関数で匿名ブロック (`BEGIN TRAN` など) を処理します。 • 行の一括挿入を変換します。アプリケーション内部から呼び出される SQL Server コピー (`bcp`) ユーティリティに相当する PostgreSQL ユーティリティは `COPY` です。 • 列の連結演算子を変換します。SQL Server は文字列の連結に `+` を使用しますが、PostgreSQLは `||` を使用します。	

パフォーマンスを向上する

タスク	説明	必要なスキル
AWS のサービスを活用してパフォーマンスを向上させましょう。	AWS クラウドに移行すると、アプリケーションとデータベースのデザインを改良して AWS のサービスを活用できます。たとえば、Aurora PostgreSQL 互換データベースサーバーに接続されている Python アプリケーションからのクエリが、元の Microsoft SQL Server クエリよりも時間がかかる場	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル
	合は、Aurora サーバーから Amazon Simple Storage Service (Amazon S3) バケットに履歴データのフィードを直接作成し、Amazon Athena ベースの SQL クエリを使用してユーザーダッシュボード用のレポートと分析データクエリを生成することを検討できます。	

関連リソース

- Perl
- Perl DBI モジュール
- Python
- psycopg2
- SQLAlchemy
- 一括コピー - PostgreSQL
- 一括コピー - Microsoft SQL サーバー
- PostgreSQL
- Amazon Aurora PostgreSQL の操作

追加情報

Microsoft SQL Server と Aurora PostgreSQL 互換はいずれも ANSI SQL に準拠しています。ただし、Python または Perl アプリケーションを SQL Server から PostgreSQL に移行する場合は、構文、列データ型、ネイティブデータベース固有の関数、一括挿入、および大文字と小文字の区別に互換性がないことに注意する必要があります。

次のセクションでは、考えられる不一致について詳しく説明します。

データ型の比較

データタイプを SQL Server から PostgreSQL に変更すると、アプリケーションが操作する結果データに大きな違いが生じる可能性があります。データ型の比較については、Sqlines ウェブサイトの表を参照してください。

ネイティブの SQL 関数またはビルトイン SQL 関数

一部の関数の動作は、SQL Server データベースと PostgreSQL データベースで異なります。次の表に例を示しています。

Microsoft SQL Server	説明	PostgreSQL
`CAST`	値を 1 つのデータ型から別のデータ型へ変換します。	PostgreSQL `type ::operator`
`GETDATE()`	現在のデータベースシステムの日付と時刻を `YYYY-MM-DD hh:mm:ss.mmm` 形式で返します。	`CLOCK_TIMESTAMP`
`DATEADD`	日付に時刻/日付の間隔を追加します。	`INTERVAL expression`
`CONVERT`	値を特定のデータ形式に変換します。	`TO_CHAR`
`DATEDIFF`	2 つの日付間の日数の差を返します。	`DATE_PART`
`TOP`	`SELECT` 結果セットの行数を制限します。	`LIMIT/FETCH`

匿名ブロック

構造化 SQL クエリは、宣言、実行可能ファイル、例外処理などのセクションに分かれています。次の表は、シンプルな匿名ブロックの Microsoft SQL Server バージョンと PostgreSQL バージョンを比較したものです。複雑な匿名ブロックの場合は、アプリケーション内でカスタムデータベース関数を呼び出すことをお勧めします。

Microsoft SQL Server	PostgreSQL

```
my $sql_qry1=
my $sql_qry2 =
my $sqlqry = "BEGIN TRAN
$sql_qry1 $sql_qry2
if @\@error !=0 ROLLBACK
TRAN
else COMIT TRAN";
```

```
my $sql_qry1=
my $sql_qry2 =
my $sql_qry = " DO \$\$
BEGIN
$header_sql $content_sql
END
\$\$";
```

他の違い

- 行の一括挿入: Microsoft SQL Server の bcp ユーティリティ に相当する PostgreSQL は COPY です。
- 大文字と小文字の区別: PostgreSQL では列名の大文字と小文字が区別されるため、SQL Server の列名を小文字または大文字に変換する必要があります。これは、データを抽出または比較したり、結果セットや変数に列名を入力するときの要因になります。次の例では、値を大文字または小文字で保存できる列を表示しています。

```
my $sql_qry = "SELECT $record_id FROM $exampleTable WHERE LOWER($record_name) =
 \'failed transaction\'";
```

- 連結: SQL Server は文字列連結の演算子として + を使用し、PostgreSQL は || を使用します。
- 検証: PostgreSQL のアプリケーションコードで使用する前に、インライン SQL クエリと関数をテストして検証する必要があります。
- ORM ライブラリの追加:`既存のデータベース接続ライブラリを SQLAlchemy や PynoMoDB などの Python ORM ライブラリに含めたり、置き換えたりすることもできます。これにより、オブジェクト指向のパラダイムを使用して、データベースからデータを簡単にクエリして操作できるようになります。

で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

作成者: Battulga Purevragchaa (AWS)、Babu Srinivasan (MongoDB)、Igor Alekseev (AWS)

概要

このパターンでは、IBM Db2 およびメインフレームデータベースや Sybase などの他のデータベースから の MongoDB Atlas にデータを移行する手順について説明します AWS クラウド。を使用してAWS Glue、MongoDB Atlas へのデータ移行を高速化します。

このパターンは、 AWS 規範ガイダンスウェブサイトの での MongoDB Atlas への移行 AWSガイドに付属しています。そのガイドで説明されている移行シナリオの 1 つの実装手順について説明します。その他の移行シナリオについては、 AWS 「 規範ガイダンス」ウェブサイトの次のパターンを参照してください。

- でセルフホスト MongoDB 環境を MongoDB Atlas に移行する AWS
- でリレーショナルデータベースを MongoDB Atlas に移行する AWS

このパターンは、AWS マネージドサービスパートナーと AWS ユーザーを対象としています。

前提条件と制限

前提条件

- MongoDB Atlas に移行する SAP、Sybase、IBM Db2 などのソースデータベース。
- SAP、Sybase、IBM Db2、MongoDB Atlas、 などのデータベースに精通していること AWS のサービス。

製品バージョン

- MongoDB バージョン 5.0 以降。

アーキテクチャ

次の図は AWS Glue Studio、、Amazon Kinesis Data Streams、および MongoDB Atlas を使用したバッチデータのロードとデータストリーミングを示しています。

このリファレンスアーキテクチャでは AWS Glue Studio 、 を使用して抽出、変換、ロード (ETL) パイプラインを作成し、データを MongoDB Atlas に移行します。は MongoDB Atlas と AWS Glue クローラー 統合され、データガバナンスを容易にします。Amazon Kinesis Data Streams を使用して、データをバッチで移植するか、MongoDB Atlas にストリーミングできます。

バッチデータのロード

バッチデータ移行の詳細については、 AWS ブログ記事「Compose your ETL jobs for MongoDB Atlas with AWS Glue」を参照してください。

データストリーミング

さまざまな使用シナリオをサポートする MongoDB Atlas リファレンスアーキテクチャについては、AWS 「 規範ガイダンス」ウェブサイトの「 での MongoDB Atlas への移行 AWS」を参照してください。

ツール

はフルマネージド ETL サービスAWS Glueです。これにより、データストアとデータストリーム間でのデータの分類、整理、強化、移動を確実に行うことができます。

Amazon Kinesis Data Streams」は、大量のデータレコードをリアルタイムで収集して処理するのに役立ちます。

MongoDB Atlas」は、MongoDB データベースをクラウドにデプロイして管理するためのフルマネージド型のサービスとしてのデータベース (DbaaS) です。

ベストプラクティス

ガイドラインについては、MongoDB GitHub リポジトリの「Best Practices Guide for MongoDB」を参照してください。 GitHub

エピック

発見と評価

タスク	説明	必要なスキル
クラスターサイズを決定します。	合計インデックススペースの からの情報を使用して、ワーキングセットのサイズdb.stats() を推定します。データスペースの一部が頻繁にアクセスされると仮定します。または、前提に基づいてメモリ要件を見積もることができます。このタスクには約 1 週間かかりそうです。この詳細と例、およびこのエピックの他のストーリーについては、関連リソースセクションのリンクを参照してください。	MongoDB DBA、アプリケーションアーキテクト
ネットワーク帯域幅要件を見積もります。	ネットワーク帯域幅要件を見積もるには、平均ドキュメントサイズに 1 秒あたりに提供されるドキュメント数を掛けます。基準として、クラスター上のノードが負担する最大トラフィックを考慮に入れます。クラスターからクライアントアプリケーションへのダウンストリームのデータ転送速度を計算するには、一定期間に返されたドキュメントの合計を使用します。アプリケーションがセカンダリノードから読み取る場合は、この	MongoDB DBA

タスク	説明	必要なスキル
	ドキュメントの合計数を、読み取り操作を実行できるノード数で割ります。データベースの平均ドキュメントサイズを確認するには、 `db.stats().avgObjSize` コマンドを使用します。このタスクには通常 1 日かかります。	
Atlas 層を選択します。	MongoDB ドキュメントの指示に従って、正しい Atlas クラスター層を選択します。	MongoDB DBA
カットオーバーを計画します。	アプリケーションのカットオーバーを計画します。	MongoDB DBA、アプリケーションアーキテクト

AWS に新しい MongoDB Atlas 環境を設定します

タスク	説明	必要なスキル
で新しい MongoDB Atlas クラスターを作成します AWS。	MongoDB Atlas で、クラスターの構築を選択し、クラウドプロバイダー AWS としてを選択します。	MongoDB DBA
AWS リージョン および グローバルクラスター設定を選択します。	Atlas クラスター AWS リージョン で使用できる のリストから選択します。必要に応じてグローバルクラスタを設定します。	MongoDB DBA
クラスター階層を選択します。	お好みのクラスター階層を選択します。階層の選択によって、メモリ、ストレー	MongoDB DBA

タスク	説明	必要なスキル
	ジ、IOPS の仕様などの要素が決まります。	
追加のクラスター設定を構成します。	MongoDB のバージョン、バックアップ、暗号化オプションなどのクラスター設定を追加して行います。これらのオプションの詳細については、「関連リソース」セクションを参照してください。	MongoDB DBA

セキュリティとコンプライアンスを設定

タスク	説明	必要なスキル
アクセスリストを設定します。	Atlas クラスターに接続するには、プロジェクトのアクセスリストにエントリを追加する必要があります。Atlasは、Transport Layer Security (TLS) / Secure Sockets Layer (SSL) を使用して、データベースの仮想プライベートクラウド (VPC) への接続を暗号化します。プロジェクトのアクセスリストを設定し、このエピックのストーリーの詳細については、関連リソースセクションのリンクを参照してください。	MongoDB DBA
ユーザーの認証と認可を行います。	MongoDB Atlas クラスターにアクセスするデータベースユーザーを作成して認証する必要があります。プロジェク	MongoDB DBA

タスク	説明	必要なスキル
	ト内のクラスターにアクセスするには、ユーザーはそのプロジェクトに属している必要があり、複数のプロジェクトに属している必要があります。 AWS Identity and Access Management (IAM) で認可を有効にすることもできます。詳細については、MongoDB ドキュメントの「IAM による認証のセットアップ」を参照してください。	
カスタムロールを作成します。	(オプション) Atlas は、組み込みの Atlas データベースユーザー権限が目的の権限セットをカバーしていない場合、カスタムロールの作成をサポートします。	MongoDB DBA
VPC ピアリングを設定します。	(オプション) Atlas は、他の AWS VPC との VPC ピア接続をサポートしています。VPCs	MongoDB DBA
AWS PrivateLink エンドポイントを設定します。	(オプション) AWS を使用して でプライベートエンドポイントを設定できますAWS PrivateLink。	MongoDB DBA
2 要素認証を有効にします。	(オプション) Atlasは、ユーザーがAtlasアカウントへのアクセスを制御できるようにする 2 要素認証 (2FA) をサポートしています。	MongoDB DBA

タスク	説明	必要なスキル
LDAP によるユーザー認証と認可を設定します。	(オプション) Atlasは、Lightweight Directory Access Protocol (LDAP) によるユーザー認証および認可の実行をサポートします。	MongoDB DBA
統合 AWS アクセスを設定します。	(オプション) Atlas Data Lake やカスタマーキー管理を使用した保管時の暗号化など、一部の Atlas 機能は、認証に IAM ロールを使用します。	MongoDB DBA
を使用して保管時の暗号化を設定します AWS KMS。	(オプション) Atlas は AWS Key Management Service 、(AWS KMS) を使用してストレージエンジンとクラウドプロバイダーのバックアップを暗号化することをサポートしています。	MongoDB DBA
CSFLE をセットアップします。	(オプション) Atlas は、フィールドの自動暗号化など、クライアント側のフィールドレベル暗号化 (CSFLE) をサポートしています。	MongoDB DBA

データを移行する

タスク	説明	必要なスキル
MongoDB Atlas でターゲットのレプリカセットを起動します。	MongoDB Atlas でターゲットのレプリカセットを起動します。Atlas Live Migration	MongoDB DBA

タスク	説明	必要なスキル
	Service で、移行する準備ができているを選択します。	
AWS Glue と MongoDB Atlas の接続を確立します。	を使用して MongoDB Atlas (ターゲットデータベース) AWS Glue クローラー AWS Glue に接続します。このステップは、移行用のターゲット環境を準備するのに役立ちます。詳細については、AWS Glue のドキュメントを参照してください。	MongoDB DBA
ソースデータベースまたはソースストリーム AWS Glue との の接続を確立します。	これにより、移行用のターゲット環境を準備することができます。	MongoDB DBA
データ変換を設定します。	レガシー構造化スキーマから MongoDB の柔軟なスキーマにデータを移行するように変換ロジックを設定します。	MongoDB DBA
データを移行します。	移行をスケジュールします AWS Glue Studio。	MongoDB DBA

操作統合の設定

タスク	説明	必要なスキル
クラスターに接続します。	MongoDB Atlas クラスターに接続します。	アプリ開発者
データを操作します。	クラスターデータと対話します。	アプリ開発者

タスク	説明	必要なスキル
クラスターをモニタリングします。	MongoDB Atlas クラスターをモニタリングします。	MongoDB DBA
データをバックアップおよび復元します。	クラスターデータをバックアップし、復元します。	MongoDB DBA

トラブルシューティング

問題	ソリューション
問題が発生した場合	MongoDB Atlas CloudFormation リソースリポジトリの「トラブルシューティング」を参照してください。

関連リソース

以下のリンクはすべて、特に明記されていない限り、MongoDB ドキュメントのウェブページに移動します。

移行ガイド

- での MongoDB Atlas への移行 AWS (AWS 規範ガイダンス)

発見と評価

- 「メモリ」
- 「Atlas サンプルデータセットによるサイジングの例」
- 「モバイルアプリケーションのサイジング例」
- 「ネットワークトラフィック」
- 「クラスターの自動スケーリング」
- 「Atlas サイジングテンプレート」

セキュリティとコンプライアンスの設定

- 「IP アクセスリストエントリの設定」
- データベースユーザーを設定する
- Atlas UI へのアクセスを設定する
- カスタムデータベースロールの設定
- データベースユーザーを設定する
- 「ネットワークピアリング接続の設定」
- Atlas のプライベートエンドポイントについて
- 多要素認証オプションを管理する
- 「LDAP によるユーザー認証と認可の設定」
- 「Atlas データレイク」
- 「顧客キー管理による静的暗号化」
- ロールを引き受ける方法 (IAM ドキュメント)
- 「クライアント側のフィールドレベルの暗号化の設定」
- 自動暗号化
- MongoDB Atlas セキュリティコントロール
- 「MongoDB トラストセンター」
- クラスターのセキュリティ機能を設定する

で新しい MongoDB Atlas 環境を設定する AWS

- 「クラウドプロバイダーとリージョン」
- グローバルクラスターの管理
- クラスター階層の選択
- 追加設定を構成する
- 「Atlas の使用開始」
- Atlas UI へのアクセスを設定する
- クラスターの管理

データを移行する

- データの移行またはインポート

クラスターのモニタリング

- クラスターのモニタリング

オペレーションの統合

- 「クラスターへの接続」
- データを操作する
- クラスターのモニタリング
- データのバックアップ、復元、アーカイブ

GitHub リポジトリ

- を使用して MongoDB Atlas にデータをストリーミングする AWS Glue

ワークロード別の移行パターン

トピック

- IBM
- Microsoft
- 該当なし
- オープンソース
- Oracle
- SAP

IBM

- AWS DMS を使用して Db2 データベースを Amazon EC2 から Aurora MySQL 互換のデータベースに移行する
- ログ配信を使用して Db2 for LUW を Amazon EC2 に移行することで、システム停止時間を短縮する
- 高可用性ディザスタリカバリ機能を備えた Db2 for LUW を Amazon EC2 に移行する
- AWS DMS と AWS SCT を使用して IBM Db2 on Amazon EC2 から Aurora PostgreSQL 互換に移行する
- IBM WebSphere アプリケーションサーバーから Amazon EC2 の Apache Tomcat に移行
- で IBM Db2、SAP、Sybase、およびその他のデータベースから MongoDB Atlas にデータをストリーミングする AWS

Microsoft

- Microsoft ワークロードの検出と AWS への移行を加速する
- Microsoft SQL Server から Amazon Aurora PostgreSQL 互換エディションへのデータベース移行をサポートするように Python と Perl アプリケーションを変更する
- Microsoft Excel と Python を使用して AWS DMS タスク用の AWS CloudFormation テンプレートを作成
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon S3 にエクスポートする
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- EC2 Windows インスタンスを AWS Managed Services アカウントに取り込み、移行
- メッセージキューを Microsoft Azure Service Bus から Amazon SQS に移行
- AWS DMS を使用して Microsoft SQL Server データベースを Amazon EC2 から Amazon DocumentDB に移行します
- AWS DMS と AWS SCT を使用して Microsoft SQL Server データベースを Aurora MySQL に移行
- .NET アプリケーションを Microsoft Azure App Serviceから AWS Elastic Beanstalk に移行
- オンプレミスの Microsoft SQL Server データベースを Amazon EC2 に移行する
- オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- リンクされたサーバーを使用して、オンプレミス Microsoft SQL Server データベースを Amazon RDS for SQL Server に移行する
- ネイティブバックアップと復元メソッドを使用して、オンプレミスの Microsoft SQL サーバーデータベースを Amazon RDS for SQL Server に移行
- AWS DMS を使用してオンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- AWS SCT データ抽出エージェントを使用して、オンプレミス Microsoft SQL Server データベースを Amazon Redshift に移行する
- オンプレミス Microsoft SQL Server データベースを、Linux を実行中の Amazon EC2 上の Microsoft SQL Server に移行する
- Rclone を使用して Microsoft Azure Blob から Amazon S3 にデータを移行する
- でリレーショナルデータベースを MongoDB Atlas に移行する AWS
- ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行
- AWS Cloud でオンプレミスワークロードをリホストする: 移行チェックリスト
- Microsoft SQL サーバーを AWS クラウドに移行した後に接続エラーを解決する

- Amazon FSX を使用して SQL Server Always On FCI 向けのマルチ AZ インフラストラクチャをセットアップする

該当なし

- へのリホスト移行中にファイアウォールリクエストの承認プロセスを作成する AWS

オープンソース

- Aurora PostgreSQL-Compatible でのアプリケーションユーザーとロールの作成
- AWS SCT と AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行する AWS CLIAWS CloudFormation
- ネイティブツールを使用して オンプレミスの MariaDB Amazon RDS for MariaDB に移行する
- オンプレミス MySQL データベースを Amazon EC2 に移行する
- オンプレミス MySQL データベースを Amazon RDS for MySQL に移行する
- オンプレミス MySQL データベースを Aurora MySQL に移行する
- オンプレミス PostgreSQL データベースを Aurora PostgreSQL に移行する
- Couchbase Server データベースを Amazon EC2 に移行する
- IBM WebSphere アプリケーションサーバーから Amazon EC2 上の Apache Tomcat への自動スケーリングによる移行
- Oracle GlassFish から AWS Elastic Beanstalk へ移行する
- pglogic を使用して Amazon EC2 上の PostgreSQL から Amazon RDS for PostgreSQL に移行する
- AWS デベロッパーツールを使用して ML ビルド、トレイン、およびデプロイのワークロードを Amazon SageMaker に移行する
- AWS App2Container を使用したオンプレミスの Java プリケーションの AWS への移行
- Percona XtraBackup、Amazon EFS、Amazon S3 を使用してオンプレミスの MySQL データベースを Aurora MySQL に移行する
- Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行
- Redis ワークロードを AWS 上の Redis Enterprise Cloud に移行
- RHEL ソースサーバーを再起動した後、SELinux を無効にせずに AWS レプリケーションエージェントを自動的に再起動する
- pg_transport を使用して 2 つの Amazon RDS DB インスタンス間でPostgreSQL データベースを転送する

Oracle

- Oracle の VARCHAR2 (1) データ型を Amazon Aurora PostgreSQL のブールデータ型に変換
- PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート
- Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行
- Aurora PostgreSQL 互換のファイルエンコーディングを使用して BLOB ファイルを TEXT にロード
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- Amazon RDS for Oracle データベースを別の に移行 AWS アカウント し、 AWS リージョン を使用して継続的なレプリケーション AWS DMS を行う
- Amazon RDS for Oracle DB インスタンスを別の VPC へ移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon EC2 に移行する
- Logstash を使用して、オンプレミス Oracle データベースを Amazon OpenSearch Service へ移行する
- AWS DMS と AWS SCT を使用してオンプレミスの Oracle データベースを Amazon RDS for MySQL に移行する
- オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- データベースリンクを経由した直接 Oracle Data Pump Import を使用して、オンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle Data Pump を使用してオンプレミスの Oracle データベースを Amazon RDS for Oracle に移行する
- Oracle バイスタンダーと AWS DMS を使用して、オンプレミスの Oracle データベースを Amazon RDS for PostgreSQL に移行する
- オンプレミスの Oracle データベースを Oracle Amazon EC2 に移行する
- AWS DMS と AWS SCT を使用して、Oracle データベースを Amazon EC2 から Amazon RDS for MariaDB に移行する
- AWS DMS を使用して Oracle データベースを Amazon EC2 から Amazon RDS for Oracle に移行する
- AWS DMS を使用して Oracle データベースを Amazon DynamoDB に移行する
- Oracle GoldenGate フラットファイルアダプタを使用して Oracle データベースを Amazon RDS for Oracle に移行する
- AWS DMS と AWS SCT を使用して Oracle データベースを Amazon Redshift に移行する

- AWS DMS と AWS SCT を使用して Oracle データベースを Aurora PostgreSQL に移行
- Oracle Data Pump と AWS DMS を使用して Oracle JD Edwards EnterpriseOne データベースを AWS に移行します
- AWS DMS を使用して Oracle パーティションテーブルを PostgreSQL に移行
- AWS DMS を使用して Oracle PeopleSoft データベースを AWS に移行する
- オンプレミスの Oracle データベースから Aurora PostgreSQL にデータを移行する
- Amazon RDS for Oracle から Amazon RDS for MySQL に移行する
- マテリアライズドビューと AWS DMS を使用して Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for PostgreSQL に移行
- Oracle GoldenGate を使用して Oracle Database から Amazon RDS for PostgreSQL に移行
- AWS DMS と AWS SCT を使用して Oracle on Amazon EC2 から Amazon RDS for MySQL に移行する
- AWS DMS を使用して Oracle から Amazon DocumentDB に移行する
- Amazon ECS で Oracle WebLogic から Apache Tomcat (ToMee) に移行する
- 関数ベースのインデックスを Oracle から PostgreSQL に移行する
- レガシーアプリケーションを Oracle Pro*C から ECPG に移行する
- Oracle CLOB 値を AWS 上の PostgreSQL の個々の行に移行
- Oracle Database のエラーコードを Amazon Aurora PostgreSQL-Compatible データベースに移行する
- Oracle E-Business Suite を Amazon RDS Custom に移行
- エクステンションを使用して Oracle のネイティブ関数を PostgreSQL に移行
- Oracle PeopleSoft を Amazon RDS Custom に移行
- Oracle ROWID 機能を AWS の PostgreSQL に移行
- Oracle SERIALLY_REUSABLE プラグマパッケージを PostgreSQL に移行
- 仮想生成列をOracleから PostgreSQL に移行
- Aurora PostgreSQL-Compatible で Oracle UTL_FILE 機能をセットアップする
- Oracle から Amazon Aurora PostgreSQL への移行後にデータベースオブジェクトを検証する

SAP

- オンプレミスの SAP ASE データベースを Amazon EC2 に移行
- AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する
- AWS SCT と AWS DMS を使用して Amazon EC2 上の SAP ASE を Amazon Aurora PostgreSQL 互換の Amazon Aurora PostgreSQL 互換に移行します
- アプリケーション移行サービスを使用することで、同種の SAP 移行のカットオーバー時間を短縮する

その他のパターン

- をインストールして IBM z/OS AWS のサービス からアクセスする AWS CLI
- CAST Highlight を使用して、AWS クラウドへの移行に向けたアプリケーションの準備状況を評価
- SQL Server データベースを AWS 上の MongoDB Atlas に移行する際のクエリパフォーマンスを評価する
- DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する
- AWS Lambda および Task Scheduler を使用して、Amazon EC2 で実行されている SQL Server Express Edition のデータベースタスクを自動化する
- AWS クラウドで高度なメインフレームファイルビューアを構築
- Hybrid Linked Mode を使用して VMware Cloud on AWS へのデータセンター拡張を構成する
- プライベートネットワーク経由でアプリケーション移行サービスのデータプレーンをコントロールプレーンに接続
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化
- JSON Oracleクエリを PostgreSQL データベース SQL に変換
- Teradata NORMALIZE 時間的特徴量を Amazon Redshift SQL に変換
- Teradata RESET WHEN 特徴量を Amazon Redshift SQL に変換
- を使用してアカウント間で Amazon DynamoDB テーブルをコピーする AWS Backup
- Amazon EC2 にプライベート静的 IP を使用して Cassandra クラスターをデプロイしてリバランスを回避する
- AWS CDK と TypeScript を使用してマルチスタックのアプリケーションをデプロイする
- Aurora PostgreSQL のカスタムエンドポイントを使用して Oracle RAC ワークロードをエミュレートします
- AWR レポートを使用して Oracle データベースの Amazon RDS エンジンサイズを推定
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する
- Aurora PostgreSQL の動的 SQL ステートメントの匿名ブロックを処理
- Aurora PostgreSQL 互換にオーバーロードされた Oracle関数を処理
- Amazon ECR リポジトリに移行するときに、重複するコンテナイメージを自動的に特定する
- VMware vRealize Network Insight を VMware Cloud on AWS と統合する
- Amazon RDS for Oracle DB インスタンスを AMS を使用する他のアカウントに移行する

- MirrorMaker を使用して、オンプレミスの Apache Kafka クラスターを Amazon MSK に移行する
- AWS Glue を使用して Apache Cassandra ワークロードを Amazon Keyspaces に移行する
- コンテナワークロードを Azure Red Hat OpenShift (ARO) から Red Hat OpenShift Service on AWS (ROSA) に移行する
- SharePlex と AWS DMS を使用して、Oracle 8i または 9i から Amazon RDS for Oracle に移行
- WANdisco LiveData Migrator を使用して Hadoop データを Amazon S3 に移行する
- 100 個以上の引数を持つ Oracle 関数とプロシージャを PostgreSQL に移行
- Oracle の OUT バインド変数を PostgreSQL データベースに移行
- 同じホスト名の SAP HSR を使用して SAP HANA を AWS に移行します
- 分散可用性グループを使用して SQL Server を AWS に移行する
- HCX OS アシストの移行を使用して、 VMware Cloud on AWSに VM を移行
- Micro Focus Enterprise ServerとLRS VPSX/MFIを使用して、AWS 上のメインフレームのオンライン印刷ワークロードを最新化
- Rocket Enterprise Server と LRS PageCenterX AWS を使用してメインフレーム出力管理をモダナイズする
- F5 から AWS のApplication Load Balancer に移行するときの HTTP ヘッダーを変更
- VMware Aria Operations for Logs を使用して VMware Cloud on から Splunk AWS にログを送信する
- Terraform を使用してデータベース移行用の CI/CD パイプラインを設定する
- AWS Elastic Disaster Recoveryで Oracle JD Edwards EnterpriseOne のディザスタリカバリをセットアップする
- AWS Private CA と AWS RAM を使用してプライベート証明書の管理を簡素化する
- 大規模な Db2 z/OS データを CSV ファイルで Amazon S3 に転送する

モダナイゼーション

トピック

- CAST イメージングでのソフトウェアアーキテクチャの分析と視覚化
- CAST Highlight を使用して、AWS クラウドへの移行に向けたアプリケーションの準備状況を評価
- DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする
- Amazon OpenSearch Service におけるマルチテナントサーバーレスアーキテクチャの構築
- AWS CDK と TypeScript を使用してマルチスタックのアプリケーションをデプロイする
- AWS SAM を使用してネストされたアプリケーションのデプロイを自動化
- AWS Lambda トークン自動販売機を使用して Amazon S3 の SaaS テナント分離を実装する
- AWS Step Functions を使用して、サーバーレス Saga パターンを実装する
- AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。
- ASP.NET ウェブフォームアプリケーションを AWS で最新化
- AWS Fargate を使用して、イベント駆動型でスケジュール済みの大規模ワークロードを実行する
- C# と AWS CDK を使用するサイロモデル用の SaaS アーキテクチャでのテナントオンボーディング
- CQRS とイベントソーシングを使用してモノリスをマイクロサービスに分解する
- その他のパターン

CAST イメージングでのソフトウェアアーキテクチャの分析と視覚化

アルピタ・シンハ (キャスト・ソフトウェア) とジェームズ・ハレル (キャスト・ソフトウェア) が制作

概要

このパターンは、CAST イメージングを使用して複雑なソフトウェアシステムを視覚的に操作し、ソフトウェア構造を正確に分析する方法を示しています。CAST イメージングをこのように使用することで、特にモダナイゼーションの目的で、アプリケーションアーキテクチャについて、より多くの情報に基づいた決定を下すことができます。

CAST イメージングでアプリケーションアーキテクチャを表示するには、まず CAST コンソール経由からアプリケーションのソースコードをオンボードする必要があります。その後、コンソールはアプリケーションのデータを CAST イメージングに公開します。CAST イメージングでは、アプリケーションアーキテクチャをレイヤーごとに視覚化してナビゲートできます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 「CAST イメージング用Amazon マシンイメージ (AMI)」
- 以下を含む Amazon Elastic Compute Cloud (Amazon EC2) インスタンス (メモリ最適化された r5.xlarge Amazon EC2 インスタンスが推奨されています)
  - 4 vCPU
  - 32 GB RAM
  - 最小 500 GB 以上の汎用ソリッドステートドライブ (SSD) ボリューム (gp3) ボリューム
- CAST コンソールと CAST イメージングのライセンスキー (必要なライセンスキーを入手するには、「aws.contact-me@castsoftware.com で CAST にお問い合わせください」)
- 分析したいアプリケーションの完全なソースコード (圧縮 (.zip) 形式)
- Microsoft Edge、 Mozilla Firefox、 または Google Chrome

アーキテクチャ

次のダイアグラムは、CAST コンソールを使用してアプリケーションのソースコードをオンボーディングし、CAST イメージングで表示するワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. CAST は、フロントエンド、ミドルウェア、バックエンドのコードをリバースエンジニアリングして、アプリケーションのソースコードメタデータを生成します。
2. CAST によって生成されたアプリケーションデータは CAST イメージングに自動的にインポートされ、そこで視覚化および分析できます。

このプロセスの仕組みのスナップショットは次のとおりです。

ツール

- 「CAST Imaging」はブラウザベースのアプリケーションで、ソフトウェアシステムを視覚的に表示してナビゲートできるため、アーキテクチャについて情報に基づいた決定を下すことができます。
- 「CAST コンソール」はブラウザベースのアプリケーションで、CAST AIP 分析の設定、実行、管理に役立ちます。

> **Note**
>
> CAST イメージングと CAST コンソールは、CAST イメージングの AMI に含まれています。

エピック

CAST イメージ環境のセットアップ

タスク	説明	必要なスキル
CAST コンソールの初期設定を実行します。	1. ウェブブラウザを開き、次の URL を入力して CAST コンソールに接続します: http://localhost:8081 2. プロンプトが表示されたら、CAST コンソールライ	ソフトウェアアーキテクト、開発者、テクニカルリーダー

タスク	説明	必要なスキル
	センスキーを入力します。[次へ] を選択します。 3. 設定を確認します。変更が不要な場合は、保存して完了を選択します。	
CAST イメージングの初期設定を実行します。	1. Web ブラウザを開き、次の URL を入力して CAST イメージングに接続します: http://localhost:8083 2. プロンプトが表示されたら、ユーザー名とパスワードの両方に管理者と入力してログインします。 3. プロンプトが表示されたら、CAST イメージングライセンスキーを入力します。更新を選択して、キーを保存します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー

タスク	説明	必要なスキル
CAST Extend ローカルサーバーを設定します。	(オプション) デフォルトでは、CAST Extend ローカルサーバーはオフラインモードで機能するように設定されています。これが問題なければ、追加の設定は不要です。ただし、CAST Extend ローカルサーバーをオンライン/プロキシモードで CAST Extend に直接接続して設定したい場合は、次の手順に従ってください。 **Note** CAST Extend 認証情報については、CAST Extend 登録ページを参照してください。 1. デスクトップの CAST Extend 管理センターのショートカットを使用して ウェブブラウザを読み込み、CAST Extend ローカルサーバーに接続します。 2. オンラインオプションを選択します。 3. CAST Extend の認証情報 (E メールとパスワード) を入力し、保存 を選択してプロセスを完了します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー

CAST イメージングへのアプリケーションのオンボーディング

タスク	説明	必要なスキル
アプリケーションのソースコードを準備します。	アプリケーションのソースコードを 1 つの圧縮.zip ファイルに保存します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
CAST コンソールにアプリケーションを追加します。	1. ウェブブラウザを開き、次の URL を入力して CAST コンソールに接続します: http://localhost:8081 2. プロンプトが表示されたら、ユーザー名とパスワードの両方に管理者と入力してログインします。 3. [アプリケーションの追加] を選択します。次に、アプリケーション名を入力して Add を選択します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
ソースコード配信ウィザードを開きます。	CAST コンソールで作成したアプリケーションを探します。次に、バージョンの追加を選択します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
アプリケーションのソースコードをアップロードします。	次のいずれかを行います： • アプリケーションのソースコードを含む.zip ファイルをソースコード配信ウィザードにドラッグアンドドロップします。 – または – • クラウドをアップロードアイコンを選択します。次に、アプリケーション	ソフトウェアアーキテクト、開発者、テクニカルリーダー

タスク	説明	必要なスキル
	のソースコードを含む.zip ファイルを開きます。	
分析プロセスを開始します。	1. 配信ウィザードで、バージョンの詳細を入力し、設定オプションを指定します。詳細については、CAST イメージングドキュメントの 「CAST イメージングの標準オンボーディング」 を参照してください。 2. CAST イメージングに公開オプションが選択されていることを確認します。次に、続行を選択します。 **Note** 続行を選択すると、ソースコードの分析プロセスが開始されます。CAST コンソールの進行状況ウィンドウには、分析プロセスの各ステップが表示され、分析が完了すると通知が表示されます。	ソフトウェアアーキテクト、開発者、テクニカルリーダー

CAST イメージングに公開された分析結果とデータを検証します。

タスク	説明	必要なスキル
ステータスとログを確認する。	すべての分析操作が完了したら、進捗ウィンドウに成功メッセージが表示されていることを検証します。 **Note** 各分析アクションの個々のログは、完了直後に確認できます。特定のアクションのログを確認するには、進捗状況ウィンドウでログを表示を選択します。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
アプリケーションの詳細を確認します。	「アプリケーションの詳細」パネルで、分析結果の詳細を確認します。発見されたテクノロジーとソースコードの構成を必ず確認してください。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
CAST イメージングを検証してアクセスします。	1. CAST コンソールのアプリケーション管理ペインで、アプリケーションのバージョンステータスが「処理されたイメージ」であることを確認します。 CAST イメージアイコンが表示されます。 2. CAST イメージングアイコンを選択すると、CAST イメージング内のアプリケー	ソフトウェアアーキテクト、開発者、テクニカルリーダー

タスク	説明	必要なスキル
	ションデータに直接移動できます。 **Note** イメージング処理ステータスは、ソースコードが分析され、CAST イメージングインスタンスにアップロードされたことを意味します。	

CAST イメージングでアプリケーションの分析を始めましょう

タスク	説明	必要なスキル
CAST イメージングにログインします。	キャストイメージングを開き、デフォルトの管理者認証情報 (管理者/管理者) を入力します。アプリケーションのデータが表示されます。	ソフトウェアアーキテクト、開発者、テクニカルリーダー
CAST イメージングでアプリケーションのデータを調べることができます。	CAST イメージング機能を使用してソフトウェアアーキテクチャの表示を開始します。 CAST イメージングの機能の使用方法に関する簡単なクイックチュートリアルを参照するには、ヘルプアイコンを選択して CAST イメージングヘルパーを表示してください。	ソフトウェアアーキテクト、開発者、テクニカルリーダー

タスク	説明	必要なスキル
	詳細については、「CASTイメージングのユーザーガイド」を参照してください。	

関連リソース

CAST コンソールドキュメント

- 「ログイン」
- 「CAST コンソールによるオプション設定」

CAST イメージングドキュメント

- 「CAST イメージングのアプリケーションオンボーディング - 前提条件」
- 「CAST イメージング用の新規アプリケーションの追加」
- 「CAST イメージングの標準オンボーディング — 結果のチェック」
- 「ログイン」
- 「設定オプション — 管理センター GUI」

AWS での CAST イメージングに関するその他のリソース

- 「CAST により加速された AWS へのアプリケーション最新化 — テクニカル」 (AWS PartnerCast オンラインセミナー、無料アカウントが必要)
- 「CAST と AWS Migration Hub Refactor Spacesを使用してレガシーアプリケーションを最新化」 (AWS ブログ投稿)
- 「CAST イメージングによる AWS アーキテクチャへのアプリケーションのモダナイゼーション」 (AWS ワークショップ)
- 「AWS Marketplace: CAST イメージング」
- AWS リソースのすべての CAST

CAST Highlight を使用して、AWS クラウドへの移行に向けたアプリケーションの準備状況を評価

作成:グレッグ・リベラ (キャスト・ソフトウェア)

概要

CAST Highlightは、迅速なアプリケーションポートフォリオ分析を実行するためのSoftware as a Service (SaaS) ソリューションです。このパターンでは、CAST Highlight を設定して使用して、組織の IT ポートフォリオ全体にわたるカスタムソフトウェアアプリケーションのクラウド対応状況を評価し、最新化またはAmazon Web Services (AWS) クラウドへの移行を計画する方法について説明します。

CAST Highlight は、アプリケーションのクラウド対応状況に関する洞察を生成し、移行前に取り除く必要のあるコードブロッカーを特定し、それらのブロッカーを取り除くための労力を推定し、移行後に個々のアプリケーションが使用できる AWS サービスを推奨します。

このパターンは、CAST Highlight をセットアップして使用する手順を示しています。CAST Highlight は、新規ユーザー設定、アプリケーション管理、キャンペーン管理、ソースコード分析、結果分析の 5 つのステップで構成されています。アプリケーションのスキャンと分析を正常に行うには、このパターンのエピックセクションのすべてのステップを完了する必要があります。

前提条件と制限

前提条件

- ポートフォリオマネージャー権限を持つアクティブな CAST Highlight アカウント。
- CAST Highlight ローカルエージェントをインストールするには、ローカルコンピュータに 300 MB 以上の空きディスクスペースと 4 GB のメモリが必要です。
- Microsoft Windows 8 以降
- アプリケーションのソースコードは、ローカルエージェントがインストールされているマシンからアクセスできるテキストファイルに保存する必要があります。ソースコードは社外に出ることはなく、すべてのコードはローカルでスキャンされます。

アーキテクチャ

次の図は、CAST Highlight を使用する場合のワークフローを示しています。

ワークフローの主なステップは、以下のとおりです。

1. CAST Highlight ポータルにログインし、ローカルエージェントをダウンロードして、ローカルコンピューターにインストールします。Amazon Simple Storage Service (Amazon S3) は、ローカルエージェントインストールパッケージを保存します。
2. ソースコードファイルをスキャンし、結果ファイルを生成します。
3.
> **Important**
>
> 結果ファイルを CAST ハイライトポータルにアップロードします。: ソースコードは結果ファイルに含まれません。

4. スキャンした各アプリケーションのアンケートの質問に回答します。
5. CAST Highlight ポータルで利用できるダッシュボードとレポートをご覧ください。Amazon Relational Database Service (Amazon RDS) は、コードスキャン、分析結果、CAST Highlight ソフトウェアデータを保存します。

テクノロジースタック

CAST Highlight は、アプリケーションクラウドの準備状況を分析するために以下のテクノロジーをサポートしています。

- Java
- COBOL
- C#
- C++
- Clojure
- PHP
- JavaScript
- TypeScript
- Python
- Microsoft Transact-SQL
- VB.Net

- Kotlin
- Scala
- Swift

自動化とスケール

- 「CLI アナライザー」 を使用して CAST ハイライト分析プロセスを自動化できます。

ツール

すべての前提条件が満たされていれば、このパターンにはツールは必要ありません。ただし、ソースコード管理 (SCM) ユーティリティ、コード抽出ツール、その他のツールなどのオプションツールを使用してソースコードファイルを管理することもできます。

エピック

新しいユーザーのセットアップ

タスク	説明	必要なスキル
CAST Highlight アカウントを有効にして、パスワードを選択します。	CAST Highlight を初めて使用するすべてのユーザーには、アカウント有効化メールが届きます。アクティベーションリンクをクリックして CAST Highlight アカウントを有効化し、パスワードを入力してアクティベーションプロセスを完了します。	該当なし
CAST ハイライトポータルにログインします。	CAST Highlight ホームページは、新しいパスワードを入力すると表示されます。ユーザー認証情報を使用して CAST Highlight ポータルにログインします。	該当なし

アプリケーション管理

タスク	説明	必要なスキル
アプリケーションレコードを作成します。	CAST Highlight ポータルで、ポートフォリオ管理セクションのアプリケーション管理タブに移動します。画面上部のアプリケーションタイルで、追加を選択します。	該当なし
アプリケーション名を選択します。	アプリケーションの名前を入力して、保存を選択します。この名前は CAST Highlight のアプリケーションレコードに使用されます。	該当なし
すべてのアプリケーションについて同じ手順を繰り返します。	スキャンするアプリケーションごとに上記の手順を繰り返します。	該当なし

キャンペーン管理

タスク	説明	必要なスキル
キャンペーンの作成	CAST Highlightは、「キャンペーン」という用語を使って、特定の時期に分析される一連のアプリケーションを記述します。CAST Highlight ポータルで、ポートフォリオ管理セクションの管理キャンペーンタブにナビゲーションします。[キャンペーンの作成] を選択して、キャンペーン作成画面を起動します。	該当なし

タスク	説明	必要なスキル
キャンペーンの名前を入力し、終了日を選択します。	キャンペーンの名前を入力し、終了日を選択します。 **⚠ Important** 寄稿者は、キャンペーンの終了日以降にアプリケーション分析結果を送信することはできません。	該当なし
ソースコードのスキャン、アンケートの回答、ドメインとアプリケーションの範囲を含めることを決定します。	ソースコード分析データを定性的な情報で強化するために使用される標準アンケートを 1 つ以上選択します。調査カテゴリは、「ビジネスインパクト」、「ソフトウェアメンテナンス作業」、「CloudReady」、「アプリケーションプロパティ」、「グリーンインパクト」です。キャンペーンに分析されたドメインとアプリケーションを選択します。 **⚠ Important** キャンペーンを開始する前に、必ずスキャンするすべてのアプリケーションをアプリケーションの管理セクションに追加してください。	該当なし

タスク	説明	必要なスキル
起動メッセージをカスタマイズします。	キャンペーン内のアプリケーションに関連するすべてのコントリビューターにメールで送信されるリリースメッセージをカスタマイズします。	該当なし
キャンペーンを起動します。	[完了] を選択してキャンペーンを開始します。	該当なし

ソースコード分析

タスク	説明	必要なスキル
CAST ハイライトローカルエージェントをダウンロードします。	CAST Highlight ポータルでアプリケーションスキャンを選択して、ローカルエージェントをローカルコンピューターにダウンロードします。	該当なし
ローカルエージェントをインストールします。	CASTHighlightSetup.exe インストールプログラムを起動し、表示されるセットアップ手順に従います。Local エージェントをインストールしたら、アプリケーションを分析する準備が整います。	該当なし
ローカルエージェントコードスキャンの範囲を定義します。	コード分析はファイルレベルで行われ、ファイル間の論理的なリンクや依存関係は考慮されません。すべてのファイルは同等でアプリケーションの一部とみなされます。	該当なし

タスク	説明	必要なスキル
	正確で一貫性のある結果を得るには、ローカルエージェントで利用できるファイルまたはフォルダの除外特徴量を使用してコードスキャンの範囲を準備してください。	
オープンソースまたは COTS パッケージを含めてください。	(オプション) オープンソースまたは商用の既製 (COTS) パッケージを含める場合は、スキャンする予定のフォルダーにそれらが含まれていることを確認します。通常、外部ライブラリは「third-party」などのサブフォルダーにグループ化され、メインコードは「src/main」ファイルフォルダーに配置されることがよくあります。	該当なし
テストクラスは除外します。	テストクラスは通常、コンパイルされたアプリケーションの一部ではないため、ソースコードの分析から除外されます。ただし、必要に応じてスキャンに含めることもできます。	該当なし
SCM、ビルド、デプロイフォルダーは除外します。	より一貫した結果を得るには、スキャンに SCM フォルダー、ビルドフォルダー、デプロイフォルダー (.git ファイルや.svn ファイルなど) を含めないでください。	該当なし

タスク	説明	必要なスキル
依存関係ファイルを含めてください。	スキャンするフォルダーに物理ファイルが含まれていないフレームワークや依存関係について詳しく知りたい場合は、依存関係ファイル (pom.xml、build.gradle、package.json、.vcsproj ファイルなど) を必ず含めてください。	該当なし
ローカルエージェントを呼び出します。	ローカル Windows マシンでローカルエージェントを実行します。	該当なし
ソースコードを保存するフォルダを選択します。	ソースコードを保存するフォルダを選択します。ローカルエージェントが検出するフォルダを複数追加できます。ローカルエージェントはネットワークパスによるソース検出をサポートしていますが、ソースフォルダがローカルマシンにあることを確認する必要があります。 **Important** ソースフォルダに 10,000 を超えるファイルがある場合は、複数のスキャンを実行することをお勧めします。	該当なし

タスク	説明	必要なスキル
ファイル検索を開始します。	ローカルエージェントダッシュボードでファイルの検出を選択します。ローカルエージェントは、フォルダとサブフォルダ内のファイルを検出し、それらのテクノロジーを検出します。キャンセルボタンを選択すると、いつでも検出をキャンセルできます。 ファイルの検出が完了すると、ローカルエージェントは見つかったフォルダとファイルを一覧表示します。Technologies 列には、関連するテクノロジーとファイル数が表示されます。Path 列には、フォルダーとファイルの場所が表示されます。	該当なし

タスク	説明	必要なスキル
ソースコードのスキャン設定を調整します。	(オプション) ローカルエージェントスキャンを絞り込むには、特定のフォルダーまたはファイルの 1 つ以上のテクノロジーを無効にすることができます。すべてのテクノロジが非アクティブ化されると、フォルダまたはファイルはスキャンの範囲から除外されます。 テクノロジーを無効にするには、無効にするテクノロジーの黄色いラベルを選択します。ファイルまたはフォルダーの上にカーソルを置いたときにフィルターアイコンを選択して、テクノロジーを特定のファイルまたはフォルダーに関連付けることもできます。これらの設定は保存され、フォルダやファイルの検索処理を迅速に行えます。	該当なし
ソースコードスキャンを開始します。	スキャンを設定したら、[ファイルをスキャン] を選択してスキャン処理を開始します。	該当なし

タスク	説明	必要なスキル
緑または灰色のラベルを確認します。	ソースコードのスキャンが完了すると、フォルダーレベルとファイルレベルでステータスラベルが表示されます。 緑色のラベルは、関連する技術でファイルが正しくスキャンされたことを意味します。 グレーのラベルは、ファイルがスキャンされておらず、除外されていることを意味します。除外された理由は、各ファイルのラベルにカーソルを合わせると表示されます。ファイルを除外する理由としては、バイナリファイル、読み取り不可能なファイル、見つからないファイル、外部ライブラリ、エンコードされたファイル、生成されたファイル、構文エラー、想定される言語ではないコンテンツ、十分な分析基準に準拠していないコード、サイズ制限 (10 MB) を超えるファイル、タイムアウトの問題、アナライザーが使用できないなどがあります。	該当なし
スキャン設定を変更し、コードを再スキャンします。	(オプション) スキャン設定を変更し、[ファイルをスキャン] を選択してファイルを再度スキャンできます。	該当なし

タスク	説明	必要なスキル
スキャン結果を確認します。	スキャン結果が要件を満たしている場合は、[結果の確認]を選択します。	該当なし
ローカルエージェントによって検出されたフレームワークとソフトウェアライブラリを表示します。	アプリケーションで使用または参照され、コードスキャン中にローカルエージェントによって検出されたフレームワークとソフトウェアライブラリを表示します。 これらのリストの要素は、個別の切り替えボタンを選択することで、そのまま使用することも無視することもできます。 「依存関係の確認」を選択して続行します。 **Important** フレームワークがオフになっている場合、フレームワークは CAST Highlight ポータルにリストされていないか、アプリケーションにアタッチされていません。	該当なし

タスク	説明	必要なスキル
コードスキャン結果を保存します。	ローカルエージェントは、コードスキャン結果の概要をテクノロジーごとにまとめて表示します。[Save] を選択し、結果を保存するフォルダを指定します。ローカルエージェントは、スキャンごとにすべての分析結果を含む.zip ファイルを 1 つ生成します。 ローカルエージェントは、異なるテクノロジーとルートソースフォルダの数に応じて、FolderName.Technology.Date.csv という命名構造を持つ 1 つまたは複数の.csv ファイルを自動的に生成します。	該当なし
コードスキャンファイルを CAST ハイライトポータルにアップロードします。	CAST Highlight ポータルのアプリケーションスキャンセクションで分析したアプリケーションを選択します。結果をアップロードを選択し、.csv ファイルを選択します。.csv ファイルを個別にアップロードすることもできます。各ファイルがアップロードされると、アップロードの記録が画面に表示されます。	該当なし

タスク	説明	必要なスキル
必要に応じて分析結果ファイルを削除します。	(オプション) 分析結果ファイルは、アップロード処理中にゴミ箱アイコンを選択していつでも削除できます。 ⚠ Important Portfolio Manager 権限を持つユーザー、または結果をアップロードした寄稿者のみが結果を削除できます。	該当なし
アプリケーションアンケートに回答します。	アンケートが必要なアプリケーションには、アンケートボタンが表示されます。Survey を選択し、調査の各セクションの質問に回答し、終了したら Submit を選択します。 アンケートの進行状況は画面の上部に表示されます。必須情報をすべて送信したら、結果を送信できます。ただし、すべての質問に回答することで、組織の CAST Highlight インスタンスのデータを充実させることができます。	該当なし

タスク	説明	必要なスキル
コードスキャン結果を送信します。	アプリケーションの.csv 結果ファイルをすべてアップロードし、アンケートの質問に回答したら、アプリケーションスキャンセクションで送信を選択します。このステップは、プロセスを完了し、結果が CAST Highlight ポータルに表示されるようにするために必要です。	該当なし

結果分析

タスク	説明	必要なスキル
CAST ハイライトポータルのホームページをご覧ください。	CAST Highlight ポータルのホームページには、ソフトウェアヘルス、CloudReady、ポートフォリオ全体のオープンソース安全性スコアなど、アプリケーションポートフォリオに関する大まかな情報を含むタイルが含まれています。ホームページには、オンボードアプリケーションの数も表示されます。CAST ハイライト指標の定義と測定方法の詳細については、「CAST ハイライト — 指標と方法論 (Microsoft PowerPoint プレゼンテーション)」を参照してください。	該当なし

タスク	説明	必要なスキル
CloudReady ダッシュボードをご覧ください。	CloudReady タイルを選択して「CloudReady」ダッシュボードを開きます。これは、アプリケーションのクラウド対応状況を評価するための主要なポートフォリオレベルのダッシュボードです。クラウド移行のためのポートフォリオロードマップの計画と開発に役立ちます。	該当なし

タスク	説明	必要なスキル
クラウド向けポートフォリオアドバイザーダッシュボードをご覧ください。	Portfolio Advisor for Cloud ダッシュボードは、アプリケーションを自動的に推奨移行カテゴリに分類します。セグメンテーションは各アプリケーションの技術的特性に基づいています。要因には、ソースコード分析 (クラウドへの対応状況、ソフトウェアの耐障害性など) や、調査から得られたビジネスへの影響などがあります。右上の Compute を選択して、最初のセグメンテーション推奨事項を生成します。 ダッシュボード上部のグラフのバブルは、ポートフォリオ内の各アプリケーションを推奨セグメンテーション別にまとめたものです。各アプリケーションは、各アプリケーションの関連指標を含むグラフの下のデータ表にも一覧表示されています。 推奨できるセグメントには以下が含まれます。 • リホスト — Infrastructure as a Service (IaaS) ソリューションを使用してアプリケーションをクラウドに移行するために、アプリケーションのインフラス	該当なし

タスク	説明	必要なスキル
	トラクチャ構成をリフトアンドシフトことを推奨します。 • リファクタリング — コンテナ・アズ・ア・サービス (CaaS) またはPlatform as a Service (PaaS) ソリューションを使用してアプリケーションを移行できるように、アーキテクチャや機能を変更せずにアプリケーション・コードを適度に変更することを推奨します。 • リアーキテクト — アプリケーションの状態を改善するためにアプリケーションコードを大幅に変更し、PaaSソリューションを使用して移行の準備をするか、Function as a Service (FaaS) ソリューションを使用してサーバーレスアプリケーションとしてデプロイすることを推奨しています。 • 再構築 — アプリケーションのコードを破棄して PaaS ソリューションを使用してクラウドで再開発するか、FaaS ソリューションを使用してサーバーレスアプリケーションとして再開発することを推奨しています。	

タスク	説明	必要なスキル
	• 廃止 — アプリケーションを完全に廃棄するか、場合によっては市販のSoftware as a Service (SaaS) の代替品に置き換えることを推奨しています。	
セグメンテーションの推奨事項を変更。	場合によっては、CAST Highlightが推奨するセグメントを変更することもできます。これを行うには、データテーブルでアプリケーションを参照し、アプリケーション名の横にあるドロップダウンリストから別のセグメントを選択します。右上の保存を選択し、この変更を保存します。 このデータは、右上のエクスポートを選択していつでもエクスポートできます。	該当なし

タスク	説明	必要なスキル
分析するアプリケーションを選択します。	Portfolio Advisor for Cloud ダッシュボードで、アプリケーションバブルを選択してそのアプリケーションを分析します。より詳細な分析を開始するには、バブルチャートの後の表でアプリケーションの名前を選択します。 コードインサイト (ソフトウェアヘルスパターン)、トレンド、ソフトウェア構成 (オープンソースリスク) など、さまざまなダッシュボードを使用して個々のアプリケーションを分析できます。	該当なし

タスク	説明	必要なスキル
個々のアプリケーションの CloudReady の結果を分析します。	CloudReady タブを選択すると、アプリケーションの全体的な CloudReady スコアが表示されます。このスコアは、CloudReady アンケートへの回答と CloudReady コードスキャンの組み合わせに基づく加重平均です。アンケートの質問への回答は、タイルの下の表に表示されます。 CloudReady コードスキャンを選択すると、コードスキャンの結果が表示されます。アプリケーションコードがスキャンされた CloudReady パターンのリストがあります。リストには以下の列が含まれます。 • クラウド要件は特定のコードパターンです。 • テクノロジーはパターンのプログラミング言語です。「インパクト」とは、パターンがアプリケーションに及ぼす影響です (C = コード、F = フレームワーク、A = アーキテクチャ)。 • 重大度は、移行前にこのパターンに対処することの重要性の程度として定義されます。	該当なし

タスク	説明	必要なスキル
	• 貢献度とは、このパターンが全体の CloudReady スコアにどの程度寄与するかということです。パターンが緑色の場合はブースターとなり、CloudReady スコアが上昇します。パターンが赤の場合は障害となり、CloudReady スコアが下がります。パターンに色が付いていない場合は、検出されなかったブロッカーであり、CloudReady スコアが上がります。 • ロードブロッカーとは、ブロッカーパターンが個別に出現する回数です。ロードブロック番号を選択すると、パターンが検出されたソースコードファイルのリストが表示されます。 • エスト。工数とは、各行の障害を解決するのにかかる推定日数です。	

タスク	説明	必要なスキル
Microsoft Excel にデータをエクスポートします。	(オプション) データをエクスポートしてさらに分析するには、[Excel にエクスポート] を選択します。アプリケーション分析結果データを使用して、アプリケーションのクラウド対応状況をさらに分析し、移行前にどのコードを更新する必要があるかを判断できます。	該当なし
推奨事項の表示	CloudReady Code Scan の横にある推奨事項を選択すると、クラウドサービスの推奨事項画面が表示されます。これにより、アプリケーションが採用できる AWS サービスをその特性に基づいて特定できます。 このステップを繰り返して、分析したすべてのアプリケーションの推奨事項を表示します。	該当なし

関連リソース

キャンペーン管理

- 「CAST ハイライト財団認定トレーニングセクション 3: ポートフォリオ構成」 (ビデオ)

ソースコード分析

- 「CAST ハイライト財団認定トレーニングセクション 4: アプリケーション分析」 (ビデオ)

その他のリソース

- 「AWS Marketplace CAST ハイライト」
- 「AWS と CAST: アプリケーションのモダナイゼーションを加速」
- 「CAST Highlight — ドキュメント、製品チュートリアル、サードパーティツール」
- 「CAST ハイライト — クラウド対応製品デモ」 (ビデオ)
- 「CAST Highlight によるアプリケーションポートフォリオの近代化」 (AWS ワークショップ)

DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする

作成者: Tabby Ward (AWS)

概要

このパターンでは、サーバー群を管理しなくても、Amazon DynamoDB テーブルから古いデータを削除し、Amazon Web Services (AWS) の Amazon Simple Storage Service (Amazon S3) バケットにアーカイブする手順を示します。

このパターンでは、Amazon DynamoDB の Time-toLive (TTL) を使用して古い項目を自動的に削除し、Amazon DynamoDB Streams を使用して TTL の有効期限が切れた項目をキャプチャします。次に DynamoDB Streams を AWS Lambda に接続し、サーバーをプロビジョニングしたり管理したりせずにコードを実行します。

新しい項目が DynamoDB ストリームに追加されると、Lambda 関数が開始され、Amazon Data Firehose 配信ストリームにデータが書き込まれます。Firehose は、データをアーカイブとして Amazon S3 にロードするためのシンプルでフルマネージド型のソリューションを提供します。

DynamoDB は、ウェブページのクリックストリームデータや、センサーや接続されたデバイスからのモノのインターネット (IoT) データなどの時系列データを保存するためによく使用されます。アクセス頻度の低い項目を削除するのではなく、監査目的でアーカイブしたいと考えるお客様が多くいます。TTL は、タイムスタンプ属性に基づいてアイテムを自動的に削除することで、このアーカイブを簡素化します。

TTL によって削除された項目は DynamoDB Streams で識別できます。DynamoDB Streams は、DynamoDB Streams で識別できます。DynamoDB Streams は、項目レベルの変更に関するシーケンスを時間順にキャプチャし、そのシーケンスを最大 24 時間ログに保存します。このデータは Lambda 関数で使用して Amazon S3 バケットにアーカイブすることで、ストレージコストを削減できます。コストをさらに削減するために、Amazon S3 ライフサイクルルールを作成して、(作成されるとすぐに) データを最も低コストのストレージクラス (S3 Glacier インスタント取得、S3 Glacier フレキシブル取得、長期ストレージ用の Amazon S3 Glacier Deep Archive など) に自動的に移行できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント

- macOS、Linux、または Windows にインストールおよび設定されている AWS コマンドラインインターフェイス (AWS CLI) バージョン 1.7 以降。
- Python 3.7 以降。
- インストールおよび設定されている Booto3。Boto3 がまだインストールされていない場合は、`python -m pip install boto3` コマンドを実行してインストールします。

アーキテクチャ

テクノロジースタック

- Amazon DynamoDB
- Amazon DynamoDB Streams
- Amazon Data Firehose
- AWS Lambda
- Amazon S3

1. アイテムは TTL によって削除されます。
2. DynamoDB ストリームトリガーは Lambda ストリームプロセッサ関数を呼び出します。
3. Lambda 関数は、Firehose 配信ストリームにレコードをバッチ形式で配置します。
4. データレコードは S3 バケットにアーカイブされます。

ツール

- AWS CLI – AWS コマンドラインインターフェイス (AWS CLI) は、AWS のサービスを管理するための統合ツールです。
- Amazon DynamoDB – Amazon DynamoDB は、どのような規模でも一桁のミリ秒単位のパフォーマンスを実現するキーバリューおよびドキュメントデータベースです。
- Amazon DynamoDB Time to Live (TTL) – Amazon DynamoDB TTL は、項目ごとのタイムスタンプを定義して、項目がいつ不要になるかを判断するのに役立ちます。
- Amazon DynamoDB Streams – Amazon DynamoDB Streams は、DynamoDB テーブル内の項目レベルの変更の時系列シーケンスをキャプチャし、この情報をログに最大 24 時間保存します。

- Amazon Data Firehose – Amazon Data Firehose は、ストリーミングデータをデータレイク、データストア、分析サービスに確実にロードする最も簡単な方法です。
- AWS Lambda – AWS Lambda を使用すると、サーバーのプロビジョニングや管理を必要とせずにコードを実行できます。支払いは、使用したコンピューティング時間に対する料金のみになります。
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。

コード

このパターンのコードは、GitHub 内の「Archive items to S3 using DynamoDB TTL」リポジトリで利用できます。

エピック

DynamoDB テーブル、TTL、および DynamoDB ストリームをセットアップする

タスク	説明	必要なスキル

DynamoDB テーブルを作成します。	AWS CLI を使用して、`Reservation` というテーブルを DynamoDB に作成します。ランダムな読み取りキャパシティユニット (RCU) と書き込みキャパシティユニット (WCU) を選択し、テーブルに 2 つの属性 (`ReservationID` と `ReservationDate`) を与えます。
	`aws dynamodb create-table \`
	`--table-name Reservation \`
	`--attribute-definitions Attribute`
		クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	Name=ReservationID ,AttributeType=S AttributeName=Rese rvationDate,Attrib uteType=N \ --key-schema Attribute Name=ReservationID ,KeyType=HASH AttributeName=Rese rvationDate,KeyTyp e=RANGE \ --provisioned-th roughput ReadCapac ityUnits=100,Write CapacityUnits=100 ReservationDate は、TTL を有効にするために使用される工ポックタイムスタンプです。	
DynamoDB TTL を有効にする。	AWS CLI を使用して ReservationDate 属性の DynamoDB TTL を有効にします。 aws dynamodb update-ti me-to-live \ --table-name Reservati on\ --time-to-live-spe cification Enabled=t rue,AttributeName= ReservationDate	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル

DynamoDB ストリームをオンにする。	AWS CLI を使用して、NEW_AND_OLD_IMAGES ストリームタイプを使用して Reservation テーブルの DynamoDB ストリームを有効にします。
	
	`aws dynamodb update-table \`
	`--table-name Reservation \`
	`  --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES`
	
	このストリームには、新しい項目、更新された項目、削除された項目、および TTL によって削除された項目のレコードが含まれます。TTL によって削除されたアイテムのレコードには、手動で削除されたアイテムと区別するためのメタデータ属性が追加されています。TTL 削除の userIdentity フィールドは、DynamoDB サービスが削除アクションを実行したことを示します。
	
	このパターンでは、TTL によって削除されたアイテムのみがアーカイブされますが、eventName が REMOVE
		クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	で、userIdentity に dynamodb.amazonaws.com に等しい principalId が含まれるレコードのみをアーカイブすることもできます。	

S3 バケットの作成と設定

タスク	説明	必要なスキル
S3 バケットを作成する。	AWS CLI を使用して AWS リージョンに宛先 S3 バケットを作成し、 をリージョンus-east-1 に、amzn-s3-demo-destination-bucket をバケットの名前に置き換えます。 `aws s3api create-bucket \ --bucket amzn-s3-demo-destination-bucket \ --region us-east-1` 名前空間はすべての AWS アカウントによって共有されるため、S3 バケットの名前がグローバルに一意であることを確認してください。	クラウドアーキテクト、アプリ開発者
S3 バケットの 30 日間のライフサイクルポリシーを作成する。	1. AWS マネジメントコンソールにサインインし、Amazon SNS コンソールを開きます。	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	2. Firehose のデータを含む S3 バケットを選択します。 3. S3 バケットで、[管理] タブを選択し、[ライフサイクルルールの追加] を選択します。 4. [ライフサイクルルール] ダイアログボックスにルールの名前を入力し、バケットの 30 日間のライフサイクルルールを設定します。	

Firehose 配信ストリームを作成する

タスク	説明	必要なスキル
Firehose 配信ストリームを作成して設定します。	GitHub リポジトリから `CreateFireHoseToS3.py` コードサンプルをダウンロードして編集します。 このコードは Python で記述されており、Firehose 配信ストリームと AWS Identity and Access Management (IAM) ロールを作成する方法を示しています。IAM ロールには、Firehose が送信先 S3 バケットに書き込むために使用できるポリシーがあります。	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	スクリプトを実行するには、次のコマンドとコマンドライン引数を使用します。 引数 1= <Your_S3_bucket_ARN> 、先ほど作成したバケットの Amazon リソースネーム (ARN) です 引数 2= Firehose 名 (このパイロットは を使用していますfirehose_to_s3_stream) 引数 3= IAM ロール名 (このパイロットでは firehose_to_s3 を使用) `python CreateFireHoseToS3.py <Your_S3_Bucket_ARN> firehose_to_s3_stream firehose_to_s3` 指定した IAM ロールが存在しない場合、スクリプトは信頼できる関係ポリシーと、十分な Amazon S3 アクセス権限を付与するポリシーを使用してアサインロールを作成します。これらのポリシーの例については、「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル
Firehose 配信ストリームを確認します。	AWS CLI を使用して Firehose 配信ストリームを記述し、配信ストリームが正常に作成されたことを確認します。 `aws firehose describe-delivery-stream --delivery-stream-name firehose_to_s3_stream`	クラウドアーキテクト、アプリ開発者

Firehose 配信ストリームを処理する Lambda 関数を作成する

タスク	説明	必要なスキル
Lambda 関数の信頼ポリシーを作成する。	次の情報を使用して信頼ポリシーファイルを作成します。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }`	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	これにより、関数に AWS リソースへのアクセス権限が付与されます。	
Lambda 関数の実行ロールを作成する。	実行ロールを作成するには、次のコードを実行します。 `aws iam create-role --role-name lambda-ex --assume-role-policy-document file://TrustPolicy.json`	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル

ロールにアクセス権限を追加します。	ロールにアクセス権限を追加するには、attach-policy-to-role コマンドを使用します。
	
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/AmazonKinesisFirehoseFullAccess`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/IAMFullAccess`
		クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル

Lambda 関数を作成する。	次のコマンドを実行して、コードリポジトリから LambdaStreamProcessor.py ファイルを圧縮します。
	`zip function.zip LambdaStreamProcessor.py`
	Lambda 関数を作成するときは、Lambda 実行ロール ARN が必要になります。ARN を取得するには、次のコードを実行します。
	`aws iam get-role \ --role-name lambda-ex`
	Lambda 関数を作成するには、次のコードを実行します。
	`# Review the environment variables and replace them with your values.`
	`aws lambda create-function --function-name LambdaStreamProcessor \ --zip-file fileb://function.zip --handler LambdaStreamProcessor.handler --runtime python3.8 \ --role {Your Lamda Execution Role ARN}\`
		クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	`--environment Variables="{firehose_name=firehose_to_s3_stream,bucket_arn = <Your_S3_bucket_ARN>,iam_role_name = firehose_to_s3, batch_size=400}"`	
Lambda 関数のトリガーを設定する。	AWS CLI を使用して Lambda 関数を呼び出すトリガー (DynamoDB Streams) を設定します。バッチサイズを 400 にするのは、Lambda の同時実行の問題が発生しないようにするためです。 `aws lambda create-event-source-mapping --function-name LambdaStreamProcessor \ --batch-size 400 --starting-position LATEST \ --event-source-arn <Your Latest Stream ARN From DynamoDB Console>`	クラウドアーキテクト、アプリ開発者

関数をテストする

タスク	説明	必要なスキル
タイムスタンプが期限切れの項目を Reservation テーブルに追加する。	機能をテストするには、エポックタイムスタンプが期限切れの項目を Reservation テーブルに追加します。TTL	クラウドアーキテクト

タスク	説明	必要なスキル
	はタイムスタンプに基づいて自動的に項目を削除します。 Lambda 関数は DynamoDB ストリームのアクティビティ時に開始され、イベントをフィルタリングして REMOVE アクティビティや削除された項目を識別します。次に、レコードを Firehose 配信ストリームにバッチ形式で配置します。 Firehose 配信ストリームは、`firehosetos3example/year=current year/month=current month/day=current day/hour=current hour/` プレフィックス が付いた送信先 S3 バケットに項目を転送します。 **Important** データの取得を最適化するには、「追加情報」セクションで詳しく説明`ErrorOutputPrefix` されている `Prefix`と を使用して Amazon S3 を設定します。	

リソースをクリーンアップする

タスク	説明	必要なスキル
すべてのリソースを削除する。	使用していないサービスに対して課金されることがないように、リソースをすべて削除します。	クラウドアーキテクト、アプリ開発者

関連リソース

- Managing your storage lifecycle
- Amazon S3 ストレージクラス
- AWS SDK for Python (Boto3) documentation

追加情報

Firehose 配信ストリームの作成と設定 — ポリシーの例

Firehose の信頼関係ポリシーのサンプルドキュメント

```
firehose_assume_role = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Sid': '',
                'Effect': 'Allow',
                'Principal': {
                    'Service': 'firehose.amazonaws.com'
                },
                'Action': 'sts:AssumeRole'
            }
        ]
    }
```

S3 アクセス権限ポリシーの例

```
s3_access = {
```

```
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Action": [
                    "s3:AbortMultipartUpload",
                    "s3:GetBucketLocation",
                    "s3:GetObject",
                    "s3:ListBucket",
                    "s3:ListBucketMultipartUploads",
                    "s3:PutObject"
                ],
                "Resource": [
                    "{your s3_bucket ARN}/*",
                    "{Your s3 bucket ARN}"
                ]
            }
        ]
    }
```

機能のテスト – Amazon S3 の設定

データ取得を最適化するために、次の `Prefix` および`ErrorOutputPrefix` を備えた Amazon S3 の設定が選択されています。

Prefix

```
firehosetos3example/year=! {timestamp: yyyy}/month=! {timestamp:MM}/day=!
 {timestamp:dd}/hour=!{timestamp:HH}/
```

Firehose はまず、S3 バケット`firehosetos3example`のすぐ下に という名前のベースフォルダを作成します。次に、`!{timestamp:yyyy}`、`!{timestamp:MM}`、`!{timestamp:dd}` および `!{timestamp:HH}` という式を評価し、Java DateTimeFormatter フォーマットを使用して年、月、日、時間に変換します。

例えば、Unix エポックタイムでのおよその到着タイムスタンプが 1604683577 の場合、`year=2020`、`month=11`、`day=06` および `hour=05` と評価されます。したがって、データレコードが配信される Amazon S3 内の場所は `firehosetos3example/year=2020/month=11/day=06/hour=05/` と評価されます。

ErrorOutputPrefix

```
firehosetos3erroroutputbase/!{firehose:random-string}/!{firehose:error-output-type}/!
{timestamp:yyyy/MM/dd}/
```

ErrorOutputPrefix により、S3 バケットの直下に firehosetos3erroroutputbase という名前のベースフォルダが作成されます。式 !{firehose:random-string} は、ztWxkdg3Thg などの 11 文字のランダムな文字列として評価されます。失敗したレコードが配信される Amazon S3 オブジェクトの場所は、firehosetos3erroroutputbase/ztWxkdg3Thg/processing-failed/2020/11/06/ と評価される可能性があります。

Amazon OpenSearch Service におけるマルチテナントサーバーレスアーキテクチャの構築

作成者: Tabby Ward (AWS)、Nisha Gambhir (AWS)

概要

Amazon OpenSearch Service は、ポピュラーなオープンソース検索および分析エンジンである Elasticsearch を簡単にデプロイ、運用、スケールするためのマネージド型サービスです。OpenSearch Service は、ログやメトリクスなどのストリーミングデータのフリーテキスト検索と、ほぼリアルタイムの取り込みとダッシュボード化を提供します。

Software as a Service (SaaS) プロバイダーは、OpenSearch Service を頻繁に使用して、複雑性とダウンタイムを削減しながら、スケーラブルで安全な方法で顧客インサイトを取得するなど、幅広いユースケースに対処します。

マルチテナント環境で OpenSearch Service を使用すると、SaaS ソリューションのパーティション化、分離、デプロイ、管理に影響する一連の考慮事項が導入されます。SaaS プロバイダーは、絶えず変化するワークロードに合わせて Elasticsearch クラスターを効果的にスケールする方法を検討する必要があります。また、階層化やノイズの多い隣接条件が、パーティショニングモデルにどのような影響を与えるかを考慮する必要があります。

このパターンでは、Elasticsearch コンストラクトを使用して、テナントデータを表現および隔離するために使用されるモデルを検討します。さらに、このパターンは、マルチテナント環境で OpenSearch Service を使用したインデックス作成と検索を示す例として、シンプルなサーバーレスリファレンスアーキテクチャに焦点を当てています。これにより、すべてのテナント間で同じインデックスを共有しながら、テナントのデータ分離を維持するプールデータパーティショニングのモデルが実現されます。このパターンでは、Amazon API Gateway、 AWS Amazon Simple Storage Service (Amazon S3) AWS Lambda、OpenSearch Service の各サービスを使用します。 Amazon API Gateway

プールモデルとその他のデータパーティショニングモデルの詳細については、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント

- AWS Command Line Interface （ AWS CLI) バージョン 2.x、macOS、Linux、または Windows にインストールおよび設定済み
- Python バージョン 3.9
- pip3 — Python ソースコードは .zip ファイルとして提供され、Lambda 関数にデプロイされます。コードをローカルで使用またはカスタマイズする場合は、次の手順に従ってソースコードを開発して再コンパイルします。
  1. Python スクリプトと同じディレクトリでコマンド `pip3 freeze > requirements.txt` を実行して `requirements.txt` ファイルを生成します。
  2. 依存関係 `pip3 install -r requirements.txt` をインストールします。

機能制限

- このコードは Python で実行され、現在、他のプログラミング言語はサポートされていません。
- サンプルアプリケーションには、 AWS クロスリージョンまたはディザスタリカバリ (DR) サポートは含まれません。
- このパターンは、デモンストレーションのみを目的としています。実稼働環境では使用しないでください。

アーキテクチャ

以下の図に、このパターンのアーキテクチャについて概要を示します。このアーキテクチャには、以下の項目が含まれます。

- コンテンツのインデックス作成とクエリを行う Lambda
- 検索を実行する OpenSearch Service
- ユーザーとの API インタラクションを提供する API Gateway
- (インデックスが付いていない) 未処理のデータを保存するための Amazon S3
- ログをモニタリングする Amazon CloudWatch
- AWS Identity and Access Management テナントロールとポリシーを作成するための (IAM)

自動化とスケール

わかりやすく AWS CLI するために、このパターンでは を使用してインフラストラクチャをプロビジョニングし、サンプルコードをデプロイします。 AWS CloudFormation テンプレートまたは AWS Cloud Development Kit (AWS CDK) スクリプトを作成して、パターンを自動化できます。

ツール

AWS のサービス

- AWS CLI は、コマンドラインシェルでコマンドを使用して AWS のサービス および リソースを管理するための統合ツールです。
- Lambda はサーバーをプロビジョニングしたり管理しなくてもコードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- API Gateway は、REST、HTTP、WebSocket APIsで作成、公開、保守、モニタリング、保護 AWS のサービス するための です。
- Amazon S3 は、ウェブ上のどこからでも、いつでも任意の量の情報を保存および取得できるオブジェクトストレージサービスです。
- OpenSearch Service は、Elasticsearch を大規模にコスト効率よくデプロイ、保護、実行することを容易にするフルマネージドサービスです。

コード

添付ファイルには、このパターンのサンプルが記載されています。具体的には次のとおりです。

- `index_lambda_package.zip` – プールモデルを使用して OpenSearch Service でデータをインデックス化するための Lambda 関数。
- `search_lambda_package.zip` – OpenSearch Service でデータを検索するための Lambda 関数。
- `Tenant-1-data` — Tenant-1 の (インデックスが付いていない) 未処理のデータをサンプリングします。
- `Tenant-2-data` — Tenant-2 の (インデックスが付いていない) 未処理のデータをサンプリングします。

> **Important**
>
> このパターンのストーリーには、Unix、Linux、macOS 用にフォーマットされた AWS CLI コマンド例が含まれています。Windows の場合は、各行末のバックスラッシュ (\) Unix 連結文字をキャレット (^) に置き換えてください。

> **Note**
>
> AWS CLI コマンドで、角括弧 (<>) 内のすべての値を正しい値に置き換えます。

エピック

S3 バケットの作成と設定

タスク	説明	必要なスキル
S3 バケットを作成する。	に S3 バケットを作成します AWS リージョン。このバケットには、インデックスが付いていないサンプルアプリケーションのテナントデータが格納されます。 名前空間はすべての によって共有されるため、S3 バケットの名前がグローバルに一意であることを確認してください AWS アカウント。 S3 バケットを作成するには、次のように AWS CLI create-bucket コマンドを使用します。 `aws s3api create-bucket \`	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	`--bucket <tenantrawdata> \`
	`--region <your-AWS-Region>`
	
	`tenantrawdata` は S3 バケット名です。(「バケット命名ガイドライン」に従った任意の一意の名前を使用できます)。	

Elasticsearch クラスターの作成と設定

タスク	説明	必要なスキル

OpenSearch Service ドメインを作成する。	AWS CLI create-elasticsearch-domain コマンドを実行して、OpenSearch Service ドメインを作成します。
	
	`aws es create-elasticsearch-domain \`
	`--domain-name vpc-cli-example \`
	`--elasticsearch-version 7.10 \`
	`--elasticsearch-cluster-config InstanceType=t3.medium.elasticsearch,InstanceCount=1 \`
	`--ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=10 \`
	`--domain-endpoint-options "{\"EnforceHTTPS\": true}" \`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	<pre> --encryption-at-re
	st-options "{\"Enabl
	ed\": true}" \
	  --node-to-node-
	encryption-options
	 "{\"Enabled\": true}"
	 \
	  --advanced-securit
	y-options "{\"Enabl
	ed\": true, \"Interna
	lUserDatabaseEnabled
	\": true, \
	    \"MasterUserOption
	s\": {\"MasterUserName
	\": \"KibanaUser\", \
	    \"MasterUserPasswo
	rd\": \"NewKiba
	naPassword@123\"}}" \
	  --vpc-options
	 "{\"SubnetIds\":
	 [\"<subnet-id>\"],
	 \"SecurityGroupIds\":
	 [\"<sg-id>\"]}" \
	  --access-policies
	 "{\"Version\":
	 \"2012-10-17\",
	 \"Statement\":
	 [{ \"Effect\":
	 \"Allow\", \
	    \"Principal\":
	 {\"AWS\": \"*\" },
	 \"Action\":\"es:*\",
	 \
	    \"Resource\":
	 \"arn:aws:es:<regi
	on>:<account-id>:d
	omain\/vpc-cli-exa
	mple\/*\" }] }"</pre>
	テスト用ドメインであるため、インスタンス数は 1 に	

タスク	説明	必要なスキル
	設定されています。ドメインの作成後は、詳細を変更できないため、`advanced-security-options` パラメータを使用してきめ細かいアクセス制御を有効にする必要があります。 このコマンドは、Kibana コンソールにログインできるマスターユーザー名 (`KibanaUser`) とパスワードを作成します。 このドメインは仮想プライベートクラウド (VPC) の一部であるため、使用するアクセスポリシーを指定して、必ず Elasticsearch インスタンスにアクセスする必要があります。 詳細については、 AWS ドキュメントの「VPC 内での Amazon OpenSearch Service ドメインの起動」を参照してください。	

タスク	説明	必要なスキル

踏み台ホストをセットアップします。	Kibana コンソールにアクセスするための踏み台ホストとして、Amazon Elastic Compute Cloud (Amazon EC2) Windows インスタンスをセットアップします。Elasticsearch セキュリティグループは、Amazon EC2 セキュリティグループからのトラフィックを許可する必要があります。　手順については、ブログ記事「踏み台サーバーを使用して EC2 インスタンスへのネットワークアクセスを制御する」を参照してください。
	
	踏み台ホストがセットアップされ、インスタンスに関連付けられているセキュリティグループが使用可能になったら、 AWS CLI authorize-security-group-ingress コマンドを使用して Elasticsearch セキュリティグループにアクセス許可を追加して、Amazon EC2 (踏み台ホスト) セキュリティグループからのポート 443 を許可します。
	
	`aws ec2 authorize-security-group-ingress \`
	`  --group-id <SecurityGroupIdfElasticSearch> \`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	`--protocol tcp \`
	`--port 443 \`
	`--source-group <SecurityGroupIdfBashionHostEC2>`	

Lambda 関数の作成と設定

タスク	説明	必要なスキル

Lambda 実行ロールを作成します。	AWS CLI create-role コマンドを実行して、Lambda インデックス関数に AWS のサービス および リソースへのアクセスを許可します。
	`aws iam create-role \`
	`--role-name index-lambda-role \`
	`--assume-role-policy-document file://lambda_assume_role.json`
	ここで、`lambda_assume_role.json` は、次のように Lambda 関数にアクセス`AssumeRole` 許可を付与する JSON ドキュメントです。
	`{`
	`"Version": "2012-10-17",`
	`"Statement": [`
	`{`
	`"Effect": "Allow",`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	<pre>"Principa
	l": {
	
	"Service": "lambda.a
	mazonaws.com"
	},
	"Action":
	"sts:AssumeRole"
	}
	]
	}</pre>	

タスク	説明	必要なスキル

マネージドポリシーを Lambda ロールにアタッチします。	AWS CLI attach-role-policy コマンドを実行して、前のステップで作成したロールに管理ポリシーをアタッチします。これらの2つのポリシーは、Elastic Network Interface を作成し、CloudWatch Logs にログを書き込む権限をロールに与えます。
	
	`aws iam attach-role-policy \`
	`  --role-name index-lambda-role \`
	`  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	`aws iam attach-role-policy \`
	`  --role-name index-lambda-role \`
	`  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

Lambda インデックス関数に、S3 オブジェクトの読み取り権限を与えるポリシーを作成します。	AWS CLI create-policy コマンドを に実行して、Lambda インデックス関数に S3 バケット内のオブジェクトを読み取るs3:GetObject アクセス許可を付与します。
	`aws iam create-policy \ --policy-name s3-permission-policy \ --policy-document file://s3-policy.json`
	ファイルは、S3 オブジェクトへの読み取りアクセスを許可するアクセスs3:GetObject 許可を付与する、以下に示す JSON ドキュメントs3-policy.json です。S3 バケットの作成時に別の名前を使用した場合は、Resource セクションに正しいバケット名を入力します。
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::<tenantrawdata>/*" }`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	<pre>] }</pre>	

Amazon S3 のアクセス許可ポリシーを Lambda 実行ロールにアタッチします。	AWS CLI attach-role-policy コマンドを実行して、前のステップで作成した Amazon S3 アクセス許可ポリシーを Lambda 実行ロールにアタッチします。
	<pre>aws iam attach-role-policy \
	  --role-name index-lambda-role \
	  --policy-arn <PolicyARN></pre>
	ここで、PolicyARN は Amazon S3 のアクセス許可ポリシーの Amazon リソースネーム (ARN) です。前のコマンド出力からこの値を取得できます。
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

Lambda インデックス関数を作成します。	AWS CLI create-function コマンドを実行して、OpenSearch Service にアクセスする Lambda インデックス関数を作成します。
	
	`aws lambda create-function \`
	`  --function-name index-lambda-function \`
	`  --zip-file fileb://index_lambda_package.zip \`
	`  --handler lambda_index.lambda_handler \`
	`  --runtime python3.9 \`
	`  --role "arn:aws:iam::account-id:role/index-lambda-role" \`
	`  --timeout 30 \`
	`  --vpc-config "{\"SubnetIds\": [\"<subnet-id1\>", \"<subnet-id2>\"], \`
	`  \"SecurityGroupIds\": [\"<sg-1>\"]}"`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
Amazon S3 が Lambda インデックス関数を呼び出すことを許可します。	AWS CLI add-permission コマンドを実行して、Lambda インデックス関数を呼び出すアクセス許可を Amazon S3 に付与します。 `aws lambda add-permission \ --function-name index-lambda-function \ --statement-id s3-permissions \ --action lambda:InvokeFunction \ --principal s3.amazonaws.com \ --source-arn "arn:aws:s3:::<tenantrawdata>" \ --source-account "<account-id>"`	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
Amazon S3 イベントの Lambda トリガーを追加します。	Amazon S3 `ObjectCreated` イベントが検出されたときに Lambda インデックス関数に通知を送信するには、 AWS CLI put-bucket-notification-configuration コマンドを実行します。インデックス関数は、オブジェクトが S3 バケットにアップロードされるたびに実行されます。 `aws s3api put-bucket-notification-configuration \ --bucket <tenantrawdata> \ --notification-configuration file://s3-trigger.json` ファイル `s3-trigger.json` は現在のフォルダにある JSON ドキュメントで、Amazon S3 `ObjectCreated` イベントが発生したときにリソースポリシーを Lambda 関数に追加します。	クラウドアーキテクト、クラウド管理者

Lambda 検索関数の作成と設定

タスク	説明	必要なスキル
Lambda 実行ロールを作成します。	AWS CLI create-role コマンドを実行して、Lambda 検索関数に AWS のサービス および	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	リソースへのアクセスを許可します。
	
	`aws iam create-role \ --role-name search-lambda-role \ --assume-role-policy-document file://lambda_assume_role.json`
	
	ここで、`lambda_assume_role.json` は現在フォルダ内にある JSON ドキュメントで、以下のように、Lambda 関数に `AssumeRole` の権限を付与します。
	
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }`	

タスク	説明	必要なスキル

マネージドポリシーを Lambda ロールにアタッチします。	AWS CLI attach-role-policy コマンドを実行して、前のステップで作成したロールに管理ポリシーをアタッチします。これらの2つのポリシーは、Elastic Network Interface を作成し、CloudWatch Logs にログを書き込む権限をロールに与えます。
	`aws iam attach-role-policy \`
	`  --role-name search-lambda-role \`
	`  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	`aws iam attach-role-policy \`
	`  --role-name search-lambda-role \`
	`  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
Lambda 検索関数を作成します。	AWS CLI create-function コマンドを実行して、OpenSearch Service にアクセスする Lambda 検索関数を作成します。 `aws lambda create-function \ --function-name search-lambda-function \ --zip-file fileb://search_lambda_package.zip \ --handler lambda_search.lambda_handler \ --runtime python3.9 \ --role "arn:aws:iam::account-id:role/search-lambda-role" \ --timeout 30 \ --vpc-config "{\"SubnetIds\": [\"<subnet-id1\>", \"<subnet-id2>\"], \ \"SecurityGroupIds\": [\"<sg-1>\"]}"`	クラウドアーキテクト、クラウド管理者

IAM ロールを作成および設定

タスク	説明	必要なスキル
テナントの IAM ロールを作成します。	AWS CLI create-role コマンドを実行して、検索機能のテストに使用される 2 つのテナントロールを作成します。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	`aws iam create-role \ --role-name Tenant-1-role \ --assume-role-policy-document file://assume-role-policy.json`
	
	`aws iam create-role \ --role-name Tenant-2-role \ --assume-role-policy-document file://assume-role-policy.json`
	
	この `assume-role-policy.json` ファイルは、現在のフォルダ内にある JSON ドキュメントで、Lambda 実行ロールに `AssumeRole` のアクセス権限を付与します。
	
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "<Lambda execution role for index function>", "AWS": "<Lambda execution role for search function>" }, "Action": "sts:AssumeRole"`	

タスク	説明	必要なスキル

	`}`
	`]`
	`}`	

タスク	説明	必要なスキル

新規 IAM ポリシーを作成します。	AWS CLI create-policy コマンドを実行して、Elasticsearch オペレーションへのアクセスを許可するテナントポリシーを作成します。
	
	`aws iam create-policy \ --policy-name tenant-policy \ --policy-document file://policy.json`
	
	この policy.json ファイルは、現在のフォルダ内の JSON ドキュメントで、Elasticsearch にアクセス権限を付与します。
	
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "es:ESHttpDelete", "es:ESHttpGet", "es:ESHttpHead", "es:ESHttpPost", "es:ESHttpPut", "es:ESHttpPatch" ],`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	<pre>"Resource": ["<ARN of Elasticsearch domain created earlier>"] }] }</pre>	

テナント IAM ポリシーをテナントロールにアタッチします。	AWS CLI attach-role-policy コマンドを実行して、前のステップで作成した 2 つのテナントロールにテナント IAM ポリシーをアタッチします。
	
	<pre>aws iam attach-role-policy \ --policy-arn arn:aws:iam::account-id:policy/tenant-policy \ --role-name Tenant-1-role
	
	aws iam attach-role-policy \ --policy-arn arn:aws:iam::account-id:policy/tenant-policy \ --role-name Tenant-2-role</pre>
	ポリシー ARN は、前のステップの出力から取得されます。
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

Lambda にロールを引き受ける権限を付与する IAM ポリシーを作成します。	AWS CLI create-policy コマンドを実行して、Lambda がテナントロールを引き受けるためのポリシーを作成します。
	`aws iam create-policy \ --policy-name assume-tenant-role-policy \ --policy-document file://lambda_policy.json`
	この `lambda_policy.json` ファイルは、現在のフォルダ内にある JSON ドキュメントで、`AssumeRole` にアクセス権限を付与します。
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "<ARN of tenant role created earlier>" } ] }`
	`Resource` には、ワイルドカード文字を使用すると、テナントごとに新しいポリシー
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	を作成する必要がなくなります。	

Lambda インデックス ロールに Amazon S3 へのアクセスを許可する IAM ポリシーを作成します。	AWS CLI create-policy コマンドを実行して、S3 バケット内のオブジェクトにアクセスするアクセス許可を Lambda インデックスロールに付与します。
	`aws iam create-policy \ --policy-name s3-permission-policy \ --policy-document file://s3_lambda_policy.json`
	この `s3_lambda_policy.json` ファイルは、現在のフォルダ内にある JSON ポリシードキュメントです。
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::tenantrawdata/*" } ] }`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

ポリシーを Lambda 実行ロールにアタッチします。	AWS CLI attach-role-policy コマンドを実行して、前のステップで作成したポリシーを、前に作成した Lambda インデックスと検索実行ロールにアタッチします。
	
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/assume-tenant-role-policy \`
	`  --role-name index-lambda-role`
	
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/assume-tenant-role-policy \`
	`  --role-name search-lambda-role`
	
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/s3-permission-policy \`
	`  --role-name index-lambda-role`
	
	ポリシー ARN は、前のステップの出力から取得されます。
		クラウドアーキテクト、クラウド管理者

検索ドメインを作成して設定する

タスク	説明	必要なスキル

API ゲートウェイで REST API を作成します。	AWS CLI create-rest-api コマンドを実行して、REST API リソースを作成します。
	
	`aws apigateway create-rest-api \`
	`  --name Test-Api \`
	`  --endpoint-configuration "{ \"types\": [\"REGIONAL\"] }"`
	
	エンドポイント設定タイプでは、 EDGEの代わりに を指定REGIONALして、特定の ではなくエッジロケーションを使用できます AWS リージョン。
	
	コマンド出力の id フィールドの値に注目してください。これは、以降のコマンドで使用する API ID です。
		クラウドアーキテクト、クラウド管理者

API リソースの検索を作成します。	API リソースを検索し、Lambda 検索関数をリソース名 search で起動します。(Lambda インデックス関数の API は、オブジェクトが S3 バケットにアップロードされると自動的に実行されるため、作成する必要はありません)。 1. AWS CLI get-resources コマンドを実行して、ルー	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	トパスの親 ID を取得します。
	
	`aws apigateway get-resources \ --rest-api-id <API-ID>`
	
	ID フィールドの値をメモします。この親 ID は次のコマンドで使用します。
	
	`{ "items": [ { "id": "zpsri964ck", "path": "/" } ] }`
	
	2. AWS CLI create-resource コマンドを実行して、検索 API のリソースを作成します。parent-id では、前のコマンドの ID を指定します。
	
	`aws apigateway create-resource \ --rest-api-id <API-ID> \ --parent-id <Parent-ID> \ --path-part search`	

タスク	説明	必要なスキル
検索 API の GET メソッドを作成します。	the AWS CLI put-method コマンドを実行して、検索 API のGET メソッドを作成します。 `aws apigateway put-method \ --rest-api-id <API-ID> \ --resource-id <ID from the previous command output> \ --http-method GET \ --authorization-type "NONE" \ --no-api-key-required` `resource-id` では、`create-resource` コマンドの ID を指定します。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
検索 API の GET メソッドを作成します。	the AWS CLI put-method-response コマンドを実行して、検索 API のメソッドレスポンスを追加します。 `aws apigateway put-method-response \ --rest-api-id <API-ID> \ --resource-id <ID from the create-resource command output> \ --http-method GET \ --status-code 200 \ --response-models "{\"application/json\": \"Empty\"}"` の場合は`resource-id` 、前の`create-resource` コマンドの出力から ID を指定します。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
検索 API のプロキシ Lambda 統合を設定します。	the AWS CLI put-integration コマンドを実行して、Lambda 検索関数との統合を設定します。 `aws apigateway put-integration \ --rest-api-id <API-ID> \ --resource-id <ID from the create-resource command output> \ --http-method GET \ --type AWS_PROXY \ --integration-http-method GET \ --uri arn:aws:apigateway:region:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account-id>:function:<function-name>/invocations`  `resource-id` では、前の `create-resource` コマンドの ID を指定します。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
API ゲートウェイに Lambda 検索関数を呼び出す権限を付与します。	AWS CLI add-permission コマンドを実行して、検索関数を使用するアクセス許可を API Gateway に付与します。 `aws lambda add-permission \ --function-name <function-name> \ --statement-id apigateway-get \ --action lambda:InvokeFunction \ --principal apigateway.amazonaws.com \ --source-arn "arn:aws:execute-api:<region>:<account-id>:api-id/*/GET/search` search ではなく別の API リソース名を使用している場合は、source-arn パスを変更します。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

検索 API をデプロイします。	AWS CLI create-deployment コマンドを実行して、 という名前のステージリソースを作成しますdev。
	
	`aws apigateway create-deployment \`
	`  --rest-api-id <API-ID> \`
	`  --stage-name dev`
	
	API を更新する場合は、同じ AWS CLI コマンドを使用して同じステージに再デプロイできます。
		クラウドアーキテクト、クラウド管理者

Kibana ロールを作成して設定します。

タスク	説明	必要なスキル
Kibana コンソールにログインします。	1. OpenSearch Service コンソールのドメインダッシュボードで Kibana へのリンクを見つけます。この URL は <domain-endpoint>/_plugin/kibana/ の形式になります。 2. 最初のエピックで設定した踏み台ホストを使用して Kibana コンソールにアクセスします。 3. OpenSearch Service ドメインを作成したときに、前の	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	ステップのマスターユーザー名とパスワードを使用して Kibana コンソールにログインします。 4. テナントを選択するよう求められた場合、[Private] を選択します。	

タスク	説明	必要なスキル
Kibana ロールの作成と設定	データを隔離し、あるテナントが別のテナントのデータを取得できないようにするには、ドキュメントセキュリティを使用する必要があります。ドキュメントセキュリティを使用すると、テナントがテナント ID を含むドキュメントのみにアクセスできるようになります。 1. Kibana コンソールのナビゲーションペインで、[セキュリティ]、[ロール] を選択します。 2. 新規テナントロールを作成します。 3. クラスターのアクセス許可を に設定します。これにより`indices_all` 、OpenSearch Service インデックスに対する作成、読み取り、更新、削除 (CRUD) アクセス許可が付与されます。 4. インデックスへのアクセス権限を `tenant-data` インデックスに制限します。(インデックス名は Lambda 検索関数とインデックス関数の名前と一致する必要があります。) 5. インデックス権限を `indices_all` に設定し	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	て、インデックス関連のすべての操作をユーザーが実行できるようにします。(要件に応じて、よりきめ細かくアクセスできるように操作を制限できます)。 6. ドキュメントレベルのセキュリティを確保するには、次のポリシーを使用してテナント ID でドキュメントをフィルタリングし、共有インデックス内のテナントにデータを隔離します。 `{ "bool": { "must": { "match": { "TenantId": "Tenant-1" } } } }` インデックス名、プロパティ、値では、大文字と小文字が区別されます。	

タスク	説明	必要なスキル
ユーザーをロールにマッピングします。	1. ロールについて [マッピングされたユーザー] タブを選択し、[ユーザーのマッピング] を選択します。 2. [バックエンドロール] セクションで、前に作成した IAM テナントロールの ARN を指定してから、マップを選択します。これにより、IAM テナントロールが Kibana ロールにマッピングされ、テナント固有の検索ではそのテナントのデータのみが返されるようになります。たとえば、Tenant-1 の IAM ロール名が `Tenant-1-Role` の場合、Tenant-1 の Kibana ロールの [バックエンドロール] ボックスで `Tenant-1-Role` の ARN を指定します(「テナントロールの作成および設定」エピックから)。 3. テナント 2 では、ステップ 1 と 2 を繰り返します。 テナントのオンボーディング時に、テナントロールと Kibana ロールの作成を自動化することをお勧めします。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
テナント-データ インデックスを作成します。	ナビゲーションペインの [管理] で [開発ツール] を選択し、次のコマンドを実行します。このコマンドは、TenantId プロパティのマッピングを定義する tenant-data インデックスを作成します。 `PUT /tenant-data { "mappings": { "properties": { "TenantId": { "type": "keyword"} } } }`	クラウドアーキテクト、クラウド管理者

Amazon S3 と の VPC エンドポイントを作成する AWS STS

タスク	説明	必要なスキル
Amazon S3 の VPC エンドポイントを作成します。	AWS CLI create-vpc-endpoint コマンドを実行して、Amazon S3 の VPC エンドポイントを作成します。エンドポイントは、VPC の Lambda インデックス関数が Amazon S3 にアクセスできるようにします。 `aws ec2 create-vpc-endpoint \ --vpc-id <VPC-ID> \ --service-name com.amazonaws.us-east-1.s3 \`	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	`--route-table-ids <route-table-ID>` vpc-id には、Lambda インデックス関数に使用している VPC を指定します。service-name には、Amazon S3 エンドポイントの正しい URL を使用します。route-table-ids には、VPC エンドポイントに関連付けられているルートテーブルを指定します。	

タスク	説明	必要なスキル
の VPC エンドポイントを作成します AWS STS。	AWS CLI create-vpc-endpoint コマンドを実行して、 AWS Security Token Service () の VPC エンドポイントを作成しますAWS STS。エンドポイントは、VPC 内の Lambda インデックスと検索関数がアクセスできるようにします AWS STS。関数は、IAM ロールを引き受ける AWS STS ときにを使用します。 `aws ec2 create-vpc-endpoint \ --vpc-id <VPC-ID> \ --vpc-endpoint-type Interface \ --service-name com.amazonaws.us-east-1.sts \ --subnet-id <subnet-ID> \ --security-group-id <security-group-ID>` vpc-id には、Lambda インデックス関数と Lambda 検索関数に使用している VPC を指定します。subnet-id には、このエンドポイントを作成するサブネットを指定します。security-group-id には、このエンドポイントを関連付けるセキュリティグループを指定します。(Lambda が使用するセキュリ	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	ティグループと同じである可能性があります)。	

マルチテナンシーとデータ隔離のテスト

タスク	説明	必要なスキル
インデックス関数と検索関数の Python ファイルを更新します。	1. `index_lambda_package.zip` ファイルで、ファイルを編集 `lamba_index.py` して AWS アカウント ID、AWS リージョンおよび Elasticsearch エンドポイント情報を更新します。 2. `search_lambda_package.zip` ファイルで、`lambda_search.py` ファイルを編集して AWS アカウント ID、 AWS リージョンおよび Elasticsearch エンドポイント情報を更新します。 Elasticsearch エンドポイントは、OpenSearch Service コンソールの概要タブから取得できます。これは、`<AWS-Region>.es.amazonaws.com` という形式です。	クラウドアーキテクト、アプリ開発者
Lambda コードを作成します。	Python ファイルに加えた変更で Lambda コードを更新するには、 AWS CLI update-fu	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル

	nction-code コマンドを使用します。
	
	`aws lambda update-function-code \ --function-name index-lambda-function \ --zip-file fileb://index_lambda_package.zip`
	
	`aws lambda update-function-code \ --function-name search-lambda-function \ --zip-file fileb://search_lambda_package.zip`	

タスク	説明	必要なスキル

S3 バケットに未処理のデータをアップロードします。	Tenant-1 および Tenant-2 オブジェクトのデータをtenantrawdata バケットにアップロードするには、AWS CLI cp コマンドを使用します (この目的のために作成した S3 バケットの名前を指定します) 。
	`aws s3 cp tenant-1-data s3://tenantrawdata`
	`aws s3 cp tenant-2-data s3://tenantrawdata`
	S3 バケットは、データがアップロードされるたびに Lambda インデックス関数を実行するように設定されているため、ドキュメントには Elasticsearch でインデックスが付けられます。
		クラウドアーキテクト、クラウド管理者

Kibana コンソールからデータを検索します。	Kibana コンソールで、以下のクエリを実行します。 `GET tenant-data/_search` このクエリは、Elasticsearch でインデックスに登録されているすべてのドキュメントを表示します。この場合、Tenant-1 と Tenant-2 の Tenant-2つの異なるドキュメントが表示されます。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

API ゲートウェイ から検索 API をテストします。	1. API ゲートウェイコンソールで、検索 API を開き、検索リソース内の GET メソッドを選択して、[テスト] を選択します。
	2. テストウィンドウで、テナント ID に次のクエリ文字列 (大文字と小文字を区別) を入力し、[テスト] を選択します。
	`TenantId=Tenant-1`
	Lambda 関数は、ドキュメントレベルのセキュリティに基づいてテナントドキュメントをフィルタリングするクエリを OpenSearch Service に送信します。このメソッドは Tenant-1 に属するドキュメントを返します。
	3. クエリ文字列を次のように変更します。
	`TenantId=Tenant-2`
	このクエリは Tenant-1 に属するドキュメントを返します。
		クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	画面の図については、「追加情報」セクションを参照してください。	
リソースをクリーンアップします。	アカウントへの追加料金が発生しないように、作成したすべてのリソースをクリーンアップします。	AWS DevOps、クラウドアーキテクト、クラウド管理者

関連リソース

- AWS SDK for Python (Boto)
- AWS Lambda ドキュメント
- API Gateway ドキュメント
- 「Amazon S3 ドキュメント」
- 「Amazon OpenSearch Service」
  - 「Amazon OpenSearch Service のきめ細かなアクセスコントロール」
  - 「Amazon OpenSearch Service を用いて検索アプリケーションを作成する」
  - 「VPC 内で Amazon OpenSearch Service ドメインを起動する」

追加情報

データパーティショニングのモデル

マルチテナントシステムで使用される一般的なデータパーティショニングモデルには、サイロ、プール、ハイブリッドの 3 つがあります。選択するモデルは、環境のコンプライアンス、ノイジーネイバー、運用、隔離の要件によって異なります。

サイロモデル

サイロモデルでは、各テナントのデータをテナントデータが混在しない個別のストレージエリアに保存します。OpenSearch Service でサイロモデルを実装するには、テナントあたりのドメインとテナントあたりのインデックスの 2 つのアプローチを使用できます。

- テナントあたりのドメイン – テナントごとに個別の OpenSearch Service ドメイン (Elasticsearch クラスターと同義) を使用できます。各テナントを独自のドメインに配置することで、データをスタンドアロンコンストラクトに配置することに関連するすべてのメリットが得られます。しかし、このアプローチでは、管理とアジリティの面で課題が生じています。隔離型であるため、テナントの運営状況や活動を集計して評価することが難しくなっています。これは、各 OpenSearch Service ドメインに本番環境のワークロード用に少なくとも 3 つのマスターノードと 2 つのデータノードが必要です。

- テナントあたりのインデックス – OpenSearch Service クラスター内の個別のインデックスにテナントデータを配置できます。この方法では、インデックスの作成時にテナント識別子を使用し、インデックス名にテナント識別子を付加して名前を付けます。テナントごとにインデックスを付けるアプローチでは、テナントごとに完全に分離したクラスターを導入しなくても、サイロの目標を達成できます。　しかし、インデックスの数が増えると、このアプローチではより多くのシャードが必要になり、マスターノードがより多くの割り当てとリバランスを処理する必要があるため、メモリの負荷がかかる可能性があります。

サイロモデルでの隔離 — サイロモデルでは、IAM ポリシーを使用して、各テナントのデータを保持するドメインまたはインデックスを隔離します。これらのポリシーは、あるテナントが別のテナントのデータにアクセスすることを防止します。サイロ隔離モデルを実装するには、テナントリソースへのアクセスを制御するリソースベースのポリシーを作成できます。通常、これはドメインアクセスポリシーで、Elasticsearch インデックスや API など、プリンシパルがドメインのサブリソースに対して実行できるアクションを指定します。IAM アイデンティティベースのポリシーでは、OpenSearch Service 内のドメイン、インデックス、または APIs に対して許可または拒否されたアクションを指定できます。IAM ポリシーの `Action` 要素は、ポリシーによって許可または拒否される特定のアクションを記述し、`Principal`　要素は、影響を受けるアカウント、ユーザー、またはロールを指定します。

以下のサンプルポリシーは、`es:*` で指定された `tenant-1` ドメイン上のサブリソースへの完全なアクセス権を Tenant-1 にのみ付与します。`Resource` 要素の末尾に付いた `/*` は重要であり、このポリシーがドメイン自体ではなく、ドメインのサブリソースに適用されることを示します。このポリシーが有効な場合、テナントは新しいドメインの作成や、既存のドメイン設定を変更できません

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::<aws-account-id>:user/Tenant-1"
         },
         "Action": "es:*",
         "Resource": "arn:aws:es:<Region>:<account-id>:domain/tenant-1/*"
      }
   ]
}
```

インデックスサイロモデルごとにテナントを実装するには、このサンプルポリシーを変更して、インデックス名を指定して Tenant-1 を指定されたインデックスにさらに制限する必要があります。次のサンプルポリシーでは、Tenant-1 を `tenant-index-1` インデックスに制限しています。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Tenant-1"
         },
         "Action": "es:*",
         "Resource": "arn:aws:es:<Region>:<account-id>:domain/test-domain/tenant-
index-1/*"
      }
   ]
}
```

プールモデル

プールモデルでは、すべてのテナントデータが同じドメイン内のインデックスに保存されます。テナント識別子はデータ (ドキュメント) に含まれ、パーティションキーとして使用されるため、どのデータがどのテナントに属しているか判断できます。このモデルでは、管理オーバーヘッドが削減されます。プールインデックスの操作と管理は、複数のインデックスを管理するよりも容易で効率的です。しかし、テナントデータが同じインデックスに混在しているため、サイロモデルが提供する自然なテナント隔離が失われます。このアプローチでは、ノイジーネイバー効果によりパフォーマンスが低下する可能性もあります。

プールモデルでのテナント隔離 — 一般的に、テナント隔離をプールモデルに実装するのは困難です。サイロモデルで使用されている IAM メカニズムでは、ドキュメントに保存されているテナント ID に基づいて隔離を記述することはできません。

代わりに、Open Distro for Elasticsearch で提供されるきめ細かなアクセス制御 (FGAC) サポートを使用する方法があります。FGAC では、インデックス、ドキュメント、またはフィールドレベルでアクセス許可を制御できます。　FGAC は、リクエストごとに、ユーザーの認証情報を評価し、ユーザーを認証するか、アクセスを拒否します。FGAC がユーザーを認証すると、そのユーザーにマッピングされているすべてのロールを取得し、アクセス許可のセット一式を使用してリクエストの処理方法を決定します。

プールされたモデルで必要な隔離を実現するには、「ドキュメントレベルのセキュリティ」を使用できます。これにより、ロールをインデックス内のドキュメントのサブセットに制限できます。以下のサンプルロールは、クエリを Tenant-1 に制限します。このロールを Tenant-1 に適用することで、必要な隔離を実現できます。

```
{
   "bool": {
     "must": {
       "match": {
         "tenantId": "Tenant-1"
       }
     }
   }
 }
```

ハイブリッドモデル

ハイブリッドモデルでは、同じ環境でサイロモデルとプールモデルを組み合わせて、各テナントレイヤー (無料、標準、プレミアム階層など) に独自のエクスペリエンスを提供します。各層は、プールモデルで使用されているのと同じセキュリティプロファイルに従います。

ハイブリッドモデルでのテナント隔離 — ハイブリッドモデルでは、ドキュメントレベルで FGAC セキュリティモデルを使用することでテナントを隔離できるプールモデルと同じセキュリティプロファイルに従います。この戦略はクラスター管理を簡素化し、アジリティを提供しますが、アーキテクチャの他の側面が複雑になります。たとえば、どのモデルを各テナントに関連付けるか決めるには、

コードをさらに複雑化する必要があります。また、単一テナントのクエリによってドメイン全体が飽和状態になり、他のテナントのエクスペリエンスが低下しないようにする必要があります。

API ゲートウェイでのテスト

Tenant-1 クエリのテストウィンドウ

Tenant-2 クエリのテストウィンドウ

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CDK と TypeScript を使用してマルチスタックのアプリケーションをデプロイする

作成者：Dr. Rahul Sharad Gaikwad (AWS)

概要

このパターンは、AWS Cloud Development Kit (AWS CDK) と TypeScript を使用して、Amazon Web Services (AWS) にアプリケーションをデプロイするための段階的なアプローチを示しています。例として、このパターンはサーバーレスのリアルタイム分析アプリケーションをデプロイします。

このパターンはネストされたスタックアプリケーションをビルドしてデプロイします。親の AWS CloudFormation スタックは、子スタックまたはネストされたスタックを呼び出します。 各子スタックは、CloudFormation スタックで定義されている AWS リソースをビルドしてデプロイします。コマンドラインインターフェイス (CLI) コマンド `cdk` である AWS CDK ツールキットは、CloudFormation スタックの主要なインターフェイスです。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 既存の仮想プライベートクラウド (VPC) とサブネット
- インストールおよび設定済みの AWS CDK Toolkit。
- 管理者権限とアクセスキーセットを持つユーザー。
- Node.js
- AWS コマンドラインインターフェイス (AWS CLI)

制限

- AWS CDK は AWS CloudFormation を使用しているため、AWS CDK アプリケーションには CloudFormation サービスクォータが適用されます。詳細については、「AWS CloudFormation のクォータ」を参照してください。

製品バージョン

このパターンは、以下のツールとバージョンを使用して構築され、テストされています。

- AWS CDK ツールキット 1.83.0
- Node.js 14.13.0
- npm 7.0.14

このパターンは、AWS CDK または npm のどのバージョンでも機能するはずです。Node.js のバージョン 13.0.0 から 13.6.0 は AWS CDK と互換性がないことに注意してください。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Amplify Console
- Amazon API Gateway
- AWS CDK
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon Data Firehose
- Amazon Kinesis Data Streams
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)

ターゲットアーキテクチャ

次の図は、AWS CDK と TypeScript を使用したマルチスタックのアプリケーションのデプロイを示しています。

次の図は、サンプルサーバーレスリアルタイムアプリケーションのアーキテクチャを示します。

ツール

ツール

- AWS Amplify Console は、フルスタックのウェブおよびモバイルアプリケーションを AWS にデプロイするためのコントロールセンターです。Amplify コンソールホスティングは、継続的なデプロイでフルスタックのサーバーレス Web アプリをホストするための Git ベースのワークフローを提供します。Admin UI は、フロントエンドのウェブ開発者やモバイル開発者が AWS コンソールの外部でアプリケーションのバックエンドを作成および管理するためのビジュアルインターフェイスです。
- Amazon API Gateway は、あらゆる規模の REST、HTTP、WebSocket API を作成、公開、管理、モニタリング、保護するための AWS のサービスです。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK Toolkit は、AWS CDK アプリとの対話に役立つコマンドラインのクラウド開発キットです。cdk CLI コマンドは、AWS CDK アプリケーションを操作するための主要なツールです。アプリを実行し、定義したアプリケーションモデルを調べ、AWS CDK によって生成された AWS CloudFormation テンプレートを作成してデプロイします。
- Amazon CloudFront は、.html、.css、.js、画像ファイルなどの静的および動的 Web コンテンツの配布を高速化する Web サービスです。CloudFront は、エッジロケーションと呼ばれるデータセンターの世界規模のネットワークを通じてコンテンツを配信し、レイテンシーを短縮し、パフォーマンスを向上させます。
- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。ユーザーは、直接サインインしても、サードパーティを介してサインインしてもかまいません。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。
- Amazon Data Firehose は、Amazon S3、Amazon Redshift、Amazon OpenSearch Service、Splunk、カスタム HTTP エンドポイント、またはサポートされているサードパーティーサービスプロバイダーが所有する HTTP エンドポイントなどの送信先にリアルタイムのストリーミングデータを配信するためのフルマネージドサービスです。
- Amazon Kinesis Data Streams は、データレコードの大規模なストリームをリアルタイムで収集および処理するためのサービスです。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。

- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コード

このパターンのコードは添付されています。

エピック

AWS CDK ツールキットをインストール

タスク	説明	必要なスキル
AWS CDK ツールキットをインストールします。	AWS CDK Toolkit をグローバルにインストールするには、次のコマンドを実行します。 `npm install -g aws-cdk`	DevOps
バージョンを確認します。	AWS CDK Toolkit のバージョンを確認するには、次のコマンドを実行します。 `cdk --version`	DevOps

AWS 認証情報の設定

タスク	説明	必要なスキル

認証情報を設定します。	認証情報を設定するには、`aws configure` コマンドを実行し、プロンプトに従います。
	
	`$aws configure`
	`AWS Access Key ID`
	`[None]:`
		DevOps

タスク	説明	必要なスキル

	`AWS Secret Access Key [None]: your_secret_access_key`
	`Default region name [None]:`
	`Default output format [None]:`	

プロジェクトコードのダウンロード

タスク	説明	必要なスキル
添付のプロジェクトコードをダウンロードしてください。	ディレクトリとファイル構造の詳細については、「追加情報」セクションを参照してください。	DevOps

AWS CDK 環境を起動する

タスク	説明	必要なスキル
環境を起動します。	使用したいアカウントと AWS リージョンに AWS CloudFormation テンプレートをデプロイするには、次のコマンドを実行します。 `cdk bootstrap <account>/<Region>` 詳細については、AWS ドキュメントを参照してください。	DevOps

プロジェクトの構築とデプロイ

タスク	説明	必要なスキル
プロジェクトをビルドします。	プロジェクトコードをビルドするには、`npm run build` コマンドを実行します。	DevOps
プロジェクトをデプロイします。	プロジェクトコードをデプロイするには、`cdk deploy` コマンドを実行します。	

出力の確認

タスク	説明	必要なスキル
スタックの作成を確認します。	AWS マネジメントコンソールで、[CloudFormation] を選択します。プロジェクトのスタックで、親スタックと 2 つの子スタックが作成されていることを確認します。	DevOps

アプリケーションをテストする

タスク	説明	必要なスキル
データを Kinesis Data Streams に送信します。	Amazon Kinesis Data Generator (KDG) を使用して KinKinesis Data Streams にデータを送信するように AWS アカウントを設定します。詳細については、「Amazon Kinesis Data Generator」を参照してください。	DevOps

タスク	説明	必要なスキル
Amazon Cognito ユーザーを作成します。	Amazon Cognito ユーザーを作成するには、Kinesis Data Generator のヘルプページの「Amazon Cognito ユーザーの作成」セクションから cognito-setup.json CloudFormation テンプレートをダウンロードしてください。テンプレートを起動し、Amazon Cognito のユーザー名とパスワードを入力します。 [出力] タブには Kinesis Data Generator URL が一覧表示されます。	DevOps
Kinesis Data Generator にログイン	KDG にログインするには、指定した Amazon Cognito 認証情報と Kinesis データジェネレーター URL を使用します。	DevOps
アプリケーションをテストします。	KDG の [レコードテンプレート]、[テンプレート 1] に「追加情報」セクションのテストコードを貼り付け、[データを送信] を選択します。	DevOps
API Gateway をテストします。	データが取り込まれたら、GET メソッドを使用してデータを取得して API ゲートウェイをテストします。	DevOps

関連リソース

リファレンス

- AWS クラウド開発キット
- AWS CDK on GitHub
- ネストされたスタックの操作
- AWS サンプル例 - サーバーレスリアルタイム分析

追加情報

ディレクトリとファイルの詳細

このパターンでは、次の 3 つのスタックが設定されます。

- `parent-cdk-stack.ts` – このスタックは親スタックとして機能し、2 つの子アプリケーションをネストされたスタックとして呼び出します。
- `real-time-analytics-poc-stack.ts` – このネストされたスタックには、インフラストラクチャとアプリケーションコードが含まれています。
- `real-time-analytics-web-stack.ts` – このネストされたスタックには、静的な Web アプリケーションコードのみが含まれます。

重要なファイルとその機能

- `bin/real-time-analytics-poc.ts` – AWS CDK アプリケーションのエントリポイント。`lib/` で定義されたすべてのスタックをロードします。
- `lib/real-time-analytics-poc-stack.ts` – AWS CDK アプリケーションのスタックの定義 (`real-time-analytics-poc`)。
- `lib/real-time-analytics-web-stack.ts` – AWS CDK アプリケーションのスタックの定義 (`real-time-analytics-web-stack`)。
- `lib/parent-cdk-stack.ts` – AWS CDK アプリケーションのスタックの定義 (`parent-cdk`)。
- `package.json` – npm モジュールマニフェスト。アプリケーション名、バージョン、依存関係が含まれます。
- `package-lock.json` – npm が管理します。
- `cdk.json` – アプリケーションを実行するためのツールキット。
- `tsconfig.json` – プロジェクトの TypeScript を設定します。
- `.gitignore` – Git がソースコントロールから除外すべきファイルのリスト。
- `node_modules` – npm が管理します。プロジェクトの依存関係を含みます。

親スタックの次のコードセクションでは、子アプリケーションをネストされた AWS CDK スタックとして呼び出します。

```
import * as cdk from '@aws-cdk/core';
import { Construct, Stack, StackProps } from '@aws-cdk/core';
import { RealTimeAnalyticsPocStack } from './real-time-analytics-poc-stack';
import { RealTimeAnalyticsWebStack } from './real-time-analytics-web-stack';


export class CdkParentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);


    new RealTimeAnalyticsPocStack(this, 'RealTimeAnalyticsPocStack');
    new RealTimeAnalyticsWebStack(this, 'RealTimeAnalyticsWebStack');
  }
}
```

テスト用のコード

```
session={{date.now('YYYYMMDD')}}|sequence={{date.now('x')}}|
reception={{date.now('x')}}|instrument={{random.number(9)}}|
l={{random.number(20)}}|price_0={{random.number({"min":10000,
 "max":30000})}}|price_1={{random.number({"min":10000, "max":30000})}}|
price_2={{random.number({"min":10000, "max":30000})}}|
price_3={{random.number({"min":10000, "max":30000})}}|
price_4={{random.number({"min":10000, "max":30000})}}|
price_5={{random.number({"min":10000, "max":30000})}}|
price_6={{random.number({"min":10000, "max":30000})}}|
price_7={{random.number({"min":10000, "max":30000})}}|
price_8={{random.number({"min":10000, "max":30000})}}|
```

API ゲートウェイのテスト

API ゲートウェイコンソールで、GET メソッドを使用して API ゲートウェイをテストします。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS SAM を使用してネストされたアプリケーションのデプロイを自動化

作成者:Dr. Rahul Sharad Gaikwad (AWS)、Dmitry Gulin (AWS)、Ishwar Chauthaiwale (AWS) と Tabby Ward (AWS)

概要

Amazon Web Services (AWS) では、AWS サーバーレスアプリケーションモデル (AWS SAM) は関数、API、データベース、およびイベントソースマッピングを表現するための省略構文を提供するオープンソースフレームワークです。リソースごとに数行入力するだけで、必要なアプリケーションを定義し、YAML を使用してモデル化できます。デプロイ時に、SAM は SAM 構文を AWS CloudFormation 構文に変換して拡張します。これにより、サーバーレスアプリケーションをより迅速に構築できます。

AWS SAM は、AWS プラットフォームでのサーバーレスアプリケーションの開発、デプロイ、管理を簡素化します。標準化されたフレームワーク、迅速なデプロイ、ローカルテスト特徴量、リソース管理、開発ツールとのシームレスな統合、支援コミュニティを提供します。これらの特徴量により、サーバーレスアプリケーションを効率的かつ効果的に構築するための貴重なツールとなっています。

このパターンでは、AWS SAM テンプレートを使用して、ネストされたアプリケーションのデプロイを自動化します。ネストされたアプリケーションは、別のアプリケーション内のアプリケーションです。親アプリケーションは子アプリケーションを呼び出します。これらはサーバーレスアーキテクチャのコンポーネントとして疎結合されています。

ネストされたアプリケーションを使用すると、独自に作成および管理されているが、AWS SAM と Serverless Application Repository を使用して構成されているサービスまたはコンポーネントを再利用することで、高度に洗練されたサーバーレスアーキテクチャを迅速に構築できます。ネストされたアプリケーションは、より強力なアプリケーションを構築し、作業の重複を避け、チームや組織全体で一貫性とベストプラクティスを確保するのに役立ちます。ネストされたアプリケーションを示すために、このパターンでは「AWS サーバーレスショッピングカートアプリケーションの例をデプロイします」。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 既存の仮想プライベートクラウド (VPC) とサブネット

- Visual Studio Code などの統合開発環境 (詳細については、「AWS で構築するツール」を参照してください)
- pip install wheel を使用してインストールされた Python ホイールライブラリ (まだインストールされていない場合)

機能制限

- サーバーレスアプリケーションでネストできるアプリケーションの最大数は 200 です。
- ネストされたアプリケーションで使用できるパラメータの最大数は 60 です。

製品バージョン

- このソリューションは AWS SAM コマンドラインインターフェイス (AWS SAM CLI) バージョン 1.21.1 上に構築されていますが、このアーキテクチャは新しい AWS SAM CLI バージョンでも動作するはずです。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon API Gateway
- AWS SAM
- Amazon Cognito
- Amazon DynamoDB
- Lambda
- Amazon Simple Queue Service (Amazon SQS) キュー

ターゲットアーキテクチャ

次の図は、API を呼び出してショッピングサービスにユーザーリクエストを送信する方法を示しています。必要なすべての情報を含むユーザーのリクエストは、Amazon API Gatewayと Amazon Cognito オーソライザーに送信されます。Amazon Cognito オーソライザーは API の認証と認可のメカニズムを実行します。

DynamoDB で項目を追加、削除、または更新すると、DDynamoDB Streams にイベントが送信され、次に Lambda 関数が開始されます。同期ワークフローの一部として古い項目がすぐに削除され

ないように、メッセージは SQS キューに入れられ、メッセージを削除するワーカー関数が開始されます。

このソリューションセットアップでは、AWS SAM CLI が AWS CloudFormation スタックのインターフェイスとして機能します。AWS SAM テンプレートはネストされたアプリケーションを自動的にデプロイします。親 SAM テンプレートが子テンプレートを呼び出し、親 CloudFormation スタックが子スタックをデプロイします。各子スタックは、AWS SAM CloudFormation テンプレートで定義されている AWS リソースを構築します。

1. スタックを構築してデプロイします。
2. Auth CloudFormation スタックには Amazon Cognito が含まれています。
3. プロダクトCloudFormation スタックには Lambda 関数と Amazon API Gateway が含まれています。
4. Shopping CloudFormation スタックには、Lambda 関数、Amazon API Gateway、SQS キュー、Amazon DynamoDB データベースが含まれています。

ツール

ツール

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Serverless Application Model (AWS SAM)」は、 AWS クラウドのサーバーレスアプリケーションを構築するために支援するオープンソースフレームワークです。

- Amazon Simple Queue Service (Amazon SQS) は、分散したソフトウェアシステムとコンポーネントの統合と切り離しを支援し、セキュアで耐久性があり、利用可能なホスト型キューを提供します。

コード

このパターンのコードは、GitHub 内の「AWS SAM ネストスタックのサンプル」リポジトリで利用できます。

エピック

AWS SAM CLI のインストール

タスク	説明	必要なスキル
AWS SAM CLI をインストールします。	AWS SAM CLI をインストールするには、「AWS SAM ドキュメント」の手順を参照してください。	DevOps エンジニア

AWS 認証情報を設定します。	AWS SAM CLI がユーザーに代わって AWS サービスを呼び出せるように AWS 認証情報を設定するには、`aws configure` コマンドを実行してプロンプトに従います。
	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: your_secret_access_key`
	`Default region name [None]:`
	`Default output format [None]:`
		DevOps エンジニア

タスク	説明	必要なスキル
	認証情報をセットアップする詳細情報については、「権限とアクセス承認情報」を参照してください。	

AWS SAM プロジェクトを初期化する

タスク	説明	必要なスキル

AWS SAM コードリポジトリを複製します。	1. 次のコマンドを入力して、このパターンの「aws sam ネストスタックのサンプル」リポジトリを複製します。
	`git clone https://github.com/aws-samples/aws-sam-nested-stack-sample.git`
	2. 次のコマンドを入力して、クローンされたディレクトリに移動します。
	`cd aws-sam-nested-stack-sample`
		DevOps エンジニア

テンプレートをデプロイしてプロジェクトを初期化します。	プロジェクトを初期化するには、`SAM init` コマンドを実行します。テンプレートソースを選択するよう求められたら、`Custom Template Location` を選択します。	DevOps エンジニア

SAM テンプレートコードをコンパイルしてビルドする。

タスク	説明	必要なスキル
AWS SAM アプリケーションテンプレートを確認します。	ネストされたアプリケーションのテンプレートを確認します。この例では、以下のネストされたアプリケーションテンプレートを使用しています。 • `auth.yaml` — このテンプレートは、Amazon Cognito や AWS Systems Manager Parameter Store などの認証関連リソースを設定します。 • `product-mock.yaml` — このテンプレートは、Lambda 関数や Amazon API Gateway などの製品関連リソースをデプロイします。 • `shoppingcart-service.yaml` — このテンプレートは、AWS Identity and Access Management (IAM)、DynamoDB テーブル、Lambda 関数などのショッピングカート関連リソースを設定します。	DevOps エンジニア
親テンプレートを見直します。	ネストされたアプリケーションテンプレートを呼び出すテンプレートを確認します。この例では、親テンプレートは `template.yml` です。	DevOps エンジニア

タスク	説明	必要なスキル
	個別のアプリケーションはすべて単一の親テンプレート `template.yml` にネストされています。	
AWS SAM テンプレートコードをコンパイルしてビルドします。	AWS SAM CLI を使用して、次のコマンドを実行します。 `sam build`	DevOps エンジニア

AWS SAM テンプレートをデプロイする

タスク	説明	必要なスキル
アプリケーションをデプロイします。	ネストされたアプリケーション CloudFormation を作成して AWS 環境にコードをスタックしてデプロイする SAM テンプレートコードを起動するには、以下のコマンドを実行します。 `sam deploy --guided --stack-name shopping-cart-nested-stack --capabilities CAPABILITY_IAM CAPABILITY_AUTO_EXPAND` コマンドを実行すると、いくつかの質問が表示されます。すべての質問に y で答えてください。	DevOps エンジニア

デプロイメントを確認する

タスク	説明	必要なスキル
スタックを確認する	AWS SAM テンプレートで定義された AWS CloudFormation スタックと AWS リソースを確認するには、次の手順を実行します。 1. AWS マネジメントコンソールにログインして、CloudFormation コンソールにナビゲーションします。 2. 親スタックと子スタックが一覧表示されていることを確認します。 この例では、`sam-shopping-cart` はネストされた Auth、Product、Shopping スタックを呼び出す親スタックです。 プロダクトスタックは、プロダクト API ゲートウェイ URL リンクを出力として提供します。	DevOps エンジニア

関連リソース

リファレンス

- 「AWS サーバーレスアプリケーションモデル (AWS SAM)」
- 「GitHub の AWS SAM」
- 「サーバーレスショッピングカートマイクロサービス」 (AWS サンプルアプリケーション)

チュートリアルと動画

- 「サーバーレスアプリケーションの構築」
- 「AWS オンラインテックトーク:AWS SAM によるサーバーレスアプリケーションの構築とデプロイ」

追加情報

コードがすべて揃うと、この例は次のようなディレクトリ構造になります。

- 「sam_stacks」 — このフォルダーには `shared.py` レイヤーが含まれています。レイヤーは、ライブラリ、カスタムランタイム、その他の依存関係などを含むファイルアーカイブです。レイヤーを使用することで、関数のライブラリを使用することができます。デプロイパッケージに含める必要はありません。
- product-mock-service — このフォルダーには、製品に関連するすべての Lambda 関数とファイルが含まれています。
- ショッピングカートサービス — このフォルダには、ショッピング関連のすべての Lambda 関数とファイルが含まれています。

AWS Lambda トークン自動販売機を使用して Amazon S3 の SaaS テナント分離を実装する

タビー・ウォード (AWS)、スラバン・ペリヤタンビ (AWS)、トーマス・デイビス (AWS) によって作成された

概要

マルチテナントSaaSアプリケーションは、テナントの分離が維持されるようにシステムを実装する必要があります。テナントデータを同じ Amazon Web Services (AWS) リソースに保存する場合 (複数のテナントが同じ Amazon Simple Storage Service (Amazon S3) バケットにデータを保存する場合など)、テナント間のアクセスが発生しないようにする必要があります。トークン自動販売機 (TVM) は、テナントデータを分離する 1 つの方法です。これらのマシンは、トークンの生成方法の複雑さを抽象化しながら、トークンを取得するメカニズムを提供します。開発者は、TVM がどのようにトークンを生成するかについての詳細な知識がなくても使用できます。

このパターンは、AWS Lambda を使用して TVM を実装しています。TVM は、S3 バケット内の 1 つの SaaS テナントのデータへのアクセスを制限する一時的なセキュリティトークンサービス (STS) 認証情報で構成されるトークンを生成します。

TVM とこのパターンで提供されるコードは通常、JSON ウェブトークン (JWT) から派生したクレームに使用され、AWS リソースのリクエストをテナントスコープAWS Identity および Access Management (IAM) ポリシーに関連付けます。このパターンのコードを基礎として使用して、JWT トークンで提供されるクレームに基づいてスコープ付きの一時的な STS 認証情報を生成する SaaS アプリケーションを実装できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- AWS コマンドラインインターフェイス (AWS CLI) 「バージョン 1.19.0 以降」は、macOS、Linux、または Windows にインストールおよび設定されています。または、AWS CLI 「バージョン 2.1 以降」を使用することもできます。

機能制限

- このコードは Java で実行され、現在他のプログラミング言語はサポートされていません。

- サンプルアプリケーションには、AWS クロスリージョンサポートやディザスタリカバリ (DR) サポートは含まれていません。
- このパターンは、SaaS アプリケーション用の Lambda TVM がスコープ付きのテナントアクセスを提供する方法を示しています。実稼働環境での使用は想定されていません。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Lambda
- Amazon S3
- IAM
- AWS Security Token Service (AWS STS)

ターゲット アーキテクチャ

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Security Token Service (AWS STS)」を使用すると、ユーザー用の、権限が制限された一時的な認証情報をリクエストできます。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コード

このパターンのソースコードは添付ファイルとして提供され、以下のファイルが含まれています。

- `s3UploadSample.jar`は、JSON ドキュメントを S3 バケットにアップロードする Lambda 関数のソースコードを提供します。
- `tvm-layer.zip`には、Lambda 関数が S3 バケットにアクセスして JSON ドキュメントをアップロードするためのトークン (STS 一時認証情報) を提供する再利用可能な Java ライブラリが用意されています。
- `token-vending-machine-sample-app.zip`は、これらのアーティファクトの作成に使用されるソースコードとコンパイル手順を提供します。

これらのファイルを使用するには、次のセクションの指示に従います。

エピック

可変値の決定

タスク	説明	必要なスキル
可変値を決定します。	このパターンの実装には、一貫して使用しなければならない変数名がいくつか含まれています。各変数に使用すべき値を決定し、以降のステップで要求されたらその値を指定します。 <AWS Account ID> – このパターンを実装している AWS アカウントに関連付けられている 12 桁のアカウント ID。AWS アカウント ID を検索する方法については、IAM ドキュメントの「AWS アカウント ID とエイリアス」を参照してください。 <AWS Region>–このパターンを実装している AWS リージョン。AWS リージョンの詳	クラウド管理者

タスク	説明	必要なスキル
	細については、AWS ウェブサイトの「リージョンとアベイラビリティーゾーン」を参照してください。 <sample-tenant-name>—アプリケーションで使用するテナントの名前。わかりやすくするためにこの値には英数字のみを使用することをお勧めしますが、「S3 オブジェクトキーの有効な名前」を使用できます。 <sample-tvm-role-name>—TVM とサンプルアプリケーションを実行する Lambda 関数にアタッチされている IAM ロールの名前。ロール名は、スペースを含まない大文字と小文字の英数字からなる文字列です。カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (_)、等号 (=)、プラス記号 (+) を含めることもできます。ロール名はアカウント内で一意である必要があります。 <sample-app-role-name>—スコープ付きの一時的な STS 認証情報を生成するときに Lambda 関数が引き受ける IAM ロールの名前。ロール名は、スペースを含まない大文字と小文字の英数字からな	

タスク	説明	必要なスキル
	る文字列です。カンマ (,)、ピリオド (.)、アットマーク (@)、アンダースコア (_)、等号 (=)、プラス記号 (+) を含めることもできます。ロール名はアカウント内で一意である必要があります。 <sample-app-function-name>– Lambda 関数の名前。これは最大長が 64 文字の文字列です。 <sample-app-bucket-name>–特定のテナントを対象とする権限でアクセスする必要がある S3 バケットの名前。S3 バケット名* • 3～63 文字の長さにする。 • 小文字、数字、ピリオド (.)、ハイフン (-) のみで構成されるようにしてください。 • 名前の最初と最後は、文字または数字でなければなりません。 • IP アドレスの形式 (例 : 192.168.5.4) にすることはできません。 • パーティション内で一意にする必要があります。パーティションはリージョンのグループです。AWS には、現在 aws (標準リージョ	

タスク	説明	必要なスキル
	ン)、aws-cn (中国リージョン)、および aws-us-gov (AWS GovCloud [米国] リージョン) の 3 つのパーティションがあります。	

S3 バケットを作成する

タスク	説明	必要なスキル
サンプルアプリケーション用の S3 バケット環境を作成します。	次の AWS CLI コマンドを使用して、 S3 バケットを作成します。コードスニペットに「<sample-app-bucket-name>」の値を指定します。 `aws s3api create-bucket --bucket <sample-app-bucket-name>` Lambda サンプルアプリケーションは JSON ファイルをこのバケットにアップロードします。	クラウド管理者

IAM TVM ロールとポリシーを作成する

タスク	説明	必要なスキル
TVM ロールを作成します。	IAM ロールを作成するには、次のいずれかの AWS CLI コマンドを使用します。コマンドに「<sample-tvm-role-name>」の値を指定します。	クラウド管理者

タスク	説明	必要なスキル

	macOS または Linux シェルの場合：
	
	`aws iam create-role \ --role-name <sample-tvm-role-name> \ --assume-role-policy-document '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ]}'`
	
	Windows コマンドラインの場合：
	
	`aws iam create-role ^ --role-name <sample-tvm-role-name> ^ --assume-role-policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": {\"Service\": \"lambda.amazonaws.com\"}, \"Action\":`	

タスク	説明	必要なスキル

	<pre>\"sts:AssumeRole\"
	}]}"</pre>
	Lambda サンプルアプリケーションは、アプリケーションが呼び出されるとこのロールを引き受けます。スコープポリシーを使用してアプリケーションロールを引き受けることができるため、S3 バケットにアクセスするための幅広いアクセス権限がコードに付与されます。	

タスク	説明	必要なスキル

インライン TVM ロールポリシーを作成します。	次の AWS CLI コマンドのいずれかを使用して、IAM ポリシーを作成します。「<sample-tvm-role-name>」、「<AWS Account ID>」、「<sample-app-role-name>」の値をコマンドに指定します。
	
	macOS または Linux 上の Bash シェルの場合：
	
	`aws iam put-role-policy \`
	`--role-name <sample-tvm-role-name> \`
	`--policy-name assume-app-role \`
	`--policy-document '{`
	`"Version": "2012-10-17",`
	`"Statement": [`
	`{`
	`"Effect": "Allow",`
	`"Action": "sts:AssumeRole",`
	`"Resource": "arn:aws:iam::<AWS Account ID>:role/<sample-app-role-name>"`
	`}`
	`]}'`
	
	Windows コマンドラインの場合：
		クラウド管理者

タスク	説明	必要なスキル
	`aws iam put-role-policy ^ --role-name <sample-tvm-role-name> ^ --policy-name assume-app-role ^ --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Action\": \"sts:AssumeRole\", \"Resource\": \"arn:aws:iam::<AWS Account ID>:role/<sample-app-role-name>\"}]}"` このポリシーは、 マネージドのロール にアタッチされます。これにより、S3 バケットにアクセスするための幅広い権限を持つアプリケーションロールを引き受けることができるようコードに付与されます。	

タスク	説明	必要なスキル

マネージド Lambda ポリシーをアタッチします。	次の AWS CLI コマンドを使用して、AWSLambdaBasicExecutionRole IAM ポリシーを作成します。「<sample-tvm-role-name>」の値をコマンドに指定します。
	
	`aws iam attach-role-policy \ --role-name <sample-tvm-role-name> \ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	Windows コマンドラインの場合：
	
	`aws iam attach-role-policy ^ --role-name <sample-tvm-role-name> ^ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	この管理ポリシーは TVM ロールにアタッチされ、Lambda が Amazon CloudWatch にログを送信することを許可します。
		クラウド管理者

IAM アプリケーションロールとポリシーを作成する。

タスク	説明	必要なスキル
アプリケーションロールを作成します。	IAM ロールを作成するには、次のいずれかの AWS CLI コマンドを使用します。「<sample-app-role-name>」、「<AWS Account ID>」、「<sample-tvm-role-name>」の値をコマンドに指定します。 macOS または Linux 上の Bash シェルの場合： `aws iam create-role \ --role-name <sample-app-role-name> \ --assume-role-policy-document '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam::<AWS Account ID>:role/<sample-tvm-role-name>" }, "Action": "sts:AssumeRole" } ]}'`	クラウド管理者

タスク	説明	必要なスキル
	Windows コマンドラインの場合： `aws iam create-role ^ --role-name <sample-app-role-name> ^ --assume-role-policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\",\"Principal\": {\"AWS\": \"arn:aws:iam::<AWS Account ID>:role/<sample-tvm-role-name>\"},\"Action\": \"sts:AssumeRole\" }]}"` Lambda サンプルアプリケーションは、S3 バケットへのテナントベースのアクセスを取得するためのスコープポリシーを使用してこのロールを引き受けます。	

タスク	説明	必要なスキル
インラインアプリケーションロールポリシーを作成します。	次の AWS CLI コマンドのいずれかを使用して、IAM ポリシーを作成します。「<sample-app-role-name>」、「<sample-app-bucket-name>」の値をコマンドに指定します。 macOS または Linux 上の Bash シェルの場合： `aws iam put-role-policy \ --role-name <sample-app-role-name> \ --policy-name s3-bucket-access \ --policy-document '{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["s3:PutObject", "s3:GetObject", "s3:DeleteObject"], "Resource": "arn:aws:s3:::<sample-app-bucket-name>/*" }, { "Effect": "Allow",`	クラウド管理者

タスク	説明	必要なスキル

	<pre>"Action":
	["s3:ListBucket"],
	"Resource
	": "arn:aws:s3:::<sam
	ple-app-bucket-name>"
	}
	]}'</pre>
	Windows コマンドラインの場合：
	<pre>aws iam put-role-policy
	^
	--role-name <sample-a
	pp-role-name> ^
	--policy-name s3-bucket
	-access ^
	--policy-documen
	t "{\"Version\":
	\"2012-10-17\",
	\"Statement\":
	[{\"Effect\": \"Allow
	\", \"Action\":
	[\"s3:PutObject\",
	\"s3:GetObject\",
	\"s3:DeleteObject\
	"], \"Resource\":
	\"arn:aws:s3:::<sa
	mple-app-bucket-na
	me>/*\"}, {\"Effect\":
	\"Allow\", \"Action\
	": [\"s3:ListBucket
	\"], \"Resource\":
	\"arn:aws:s3:::<sa
	mple-app-bucket-name>
	\"}]}"</pre>
	このポリシーは、 マネージドのロール にアタッチされます。S3 バケット内のオブジェ	

タスク	説明	必要なスキル
	クトに幅広くアクセスできます。サンプルアプリケーションが役割を引き受けると、これらの権限は TVM の動的に生成されたポリシーによって特定のテナントに限定されます。	

TVM を使用して Lambda サンプルアプリケーションを作成する

タスク	説明	必要なスキル
コンパイルされたソースファイルをダウンロードします。	添付ファイルとして含まれている`s3UploadSample.jar` および`tvm-layer.zip` ファイルをダウンロードします。これらのアーティファクトの作成に使用されたソースコードとコンパイル手順は、`token-vending-machine-sample-app.zip` に記載されています。	クラウド管理者
Lambda コードを作成します。	次の AWS CLI コマンドを使用して、TVM を Lambda にアクセスできるようにする Lambda レイヤーを作成します。 **Note** ダウンロードした場所からこのコマンドを実行していない場	クラウド管理者、アプリ開発者

タスク	説明	必要なスキル

	合は tvm-layer .zip 、 --zip-file パラメータtvm-layer.zip で への正しいパスを指定します。
	
	`aws lambda publish-layer-version \ --layer-name sample-token-vending-machine \ --compatible-runtimes java11 \ --zip-file fileb://tvm-layer.zip`
	
	Windows コマンドラインの場合：
	
	`aws lambda publish-layer-version ^ --layer-name sample-token-vending-machine ^ --compatible-runtimes java11 ^ --zip-file fileb://tvm-layer.zip`
	
	このコマンドは、再利用可能な TVM ライブラリを含む Lambda レイヤーを作成します。	

タスク	説明	必要なスキル
Lambda 関数を作成します。	CLI コマンドを使用して、Lambda 関数を作成します。「<sample-app-function-name>」、「<AWS Account ID>」、「<AWS Region>」、「<sample-tvm-role-name>」、「<sample-app-bucket-name>」、「<sample-app-role-name>」の値をコマンドに指定します。 **Note** をダウンロードした場所からこのコマンドを実行していない場合は`s3UploadSample.jar`、`--zip-file` パラメータ`s3UploadSample.jar` で への正しいパスを指定します。 `aws lambda create-function \ --function-name <sample-app-function-name> \ --timeout 30 \ --memory-size 256 \ --runtime java11 \ --role arn:aws:iam::<AWS Account`	クラウド管理者、アプリ開発者

タスク	説明	必要なスキル
	ID>:role/<sample-tvm-role-name> \ --handler com.amazon.aws.s3UploadSample.App \ --zip-file fileb://s3UploadSample.jar \ --layers arn:aws:lambda:<AWS Region>:<AWS Account ID>:layer:sample-token-vending-machine:1 \ --environment "Variables={S3_BUCKET=<sample-app-bucket-name>, ROLE=arn:aws:iam::<AWS Account ID>:role/<sample-app-role-name>}" コマンドラインウィンドウ aws lambda create-function ^ --function-name <sample-app-function-name> ^ --timeout 30 ^ --memory-size 256 ^ --runtime java11 ^ --role arn:aws:iam::<AWS Account ID>:role/<sample-tvm-role-name> ^ --handler com.amazon.aws.s3UploadSample.App ^ --zip-file fileb://s3UploadSample.jar ^ --layers arn:aws:lambda:<AWS Region>:<	

タスク	説明	必要なスキル
	`AWS Account ID>:layer :sample-token-vend ing-machine:1 ^ --environment "Variable s={S3_BUCKET=<samp le-app-bucket-name >,ROLE=arn:aws:iam ::<AWS Account ID>:role/<sample-app- role-name>}"` このコマンドは、サンプルアプリケーションコードと TVM レイヤーをアタッチした Lambda 関数を作成します。また、S3_BUCKET とROLEの 2 つの環境変数も設定します。サンプルアプリケーションでは、これらの変数を使用して、引き受けるロールと JSON ドキュメントをアップロードする S3 バケットを決定します。	

サンプルアプリケーションをデプロイしてテストする

タスク	説明	必要なスキル
Lambda サンプルアプリケーションを呼び出します。	次のいずれかの AWS CLI コマンドを使用して、予想されるペイロードで Lambda サンプルアプリケーションを起動します。「<sample-app-functi on-name>」、「<sample-t enant-name>」の値をコマンドに指定します。	クラウド管理者、アプリ開発者

タスク	説明	必要なスキル

	macOS および Linux システムの場合：
	
	`aws lambda invoke \ --function <sample-app-function-name> \ --invocation-type RequestResponse \ --payload '{"tenant": "<sample-tenant-name>"}' \ --cli-binary-format raw-in-base64-out response.json`
	
	Windows コマンドラインの場合：
	
	`aws lambda invoke ^ --function <sample-app-function-name> ^ --invocation-type RequestResponse ^ --payload "{\"tenant\": \"<sample-tenant-name>\"}" ^ --cli-binary-format raw-in-base64-out response.json`
	
	このコマンドは Lambda 関数を呼び出し、結果を`response.json` ドキュメントに返します。多くの UNIX ベースのシステムでは、`response.json` を`/dev/stdout` に変更すると、別のファイルを作成せずに結	

タスク	説明	必要なスキル
	果をシェルに直接出力できます。 **Note** この Lambda 関数の後続の呼び出しで <sample-tenant-name> 値を変更すると、JSON ドキュメントの場所とトークンが提供するアクセス許可が変更されます。	
S3 バケットを表示して、作成されたオブジェクトを確認します。	先ほど作成したS3バケット（<sample-app-bucket-name>）を参照する。このバケットには、「<sample-tenant-name>」という値のS3オブジェクトのプレフィックスが含まれています。そのプレフィックスの下に、UUID が付いた名前の JSON ドキュメントがあります。サンプルアプリケーションを複数回呼び出すと、JSON ドキュメントがさらに追加されます。	クラウド管理者

タスク	説明	必要なスキル
サンプルアプリケーションの Cloudwatch ログを表示します。	「<sample-app-function-name>」という名前の Lambda 関数に関連付けられているクラウドウォッチログを表示します。手順については、AWS Lambda ドキュメントの「AWS Lambda 用の Amazon CloudWatch logsへのアクセス」を参照してください。TVM によって生成されたテナントスコープのポリシーは、これらのログで確認できます。このテナントを対象とするポリシーは、Amazon S3「PutObject」、「GetObject」、「DeleteObject」および「ListBucket」 API にサンプルアプリケーションのアクセス権限を付与しますが、「<sample-tenant-name>」に関連するオブジェクトプレフィックスのアクセス権限のみを許可します。それ以降のサンプルアプリケーションの呼び出しで「<sample-tenant-name>」を変更すると、TVM は呼び出しペイロードで指定されたテナントに対応するスコープ付きポリシーを更新します。この動的に生成されたポリシーは、SaaS アプリケーションで TVM を使用してテナントスコープのアクセス	クラウド管理者

タスク	説明	必要なスキル
	を維持する方法を示しています。 TVM 機能は Lambda レイヤーで提供されるため、コードを複製しなくても、アプリケーションで使用される他の Lambda 関数に接続できます。 動的に生成されるポリシーの図については、「追加情報」セクションを参照してください。	

関連リソース

- 動的に生成される IAM ポリシーによるテナントの分離 (ブログ記事)
- SaaS 環境における動的に生成された分離ポリシーの適用 (ブログ記事)
- AWS SaaS Boost (SaaS 製品を AWS に移行するのに役立つオープンソースのリファレンス環境)

追加情報

次の Amazon Cloudwatch ログには、このパターンで TVM コードによって生成された動的に生成されたポリシーが示されています。このスクリーンショットでは、「<sample-tenant-name>」はDOC-EXAMPLE-BUCKET、「<sample-app-bucket-name>」は`test-tenant-1`です。このスコープポリシーによって返される STS 認証情報は、オブジェクトkey prefix `test-tenant-1` に関連付けられているオブジェクトを除き、S3 バケット内のオブジェクトに対してアクションを実行できません。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Step Functions を使用して、サーバーレス Saga パターンを実装する

タビー・ウォード（AWS）、ローハン・メータ（AWS）、リンピー・テワニ（AWS）によって作成された

概要

マイクロサービスアーキテクチャの主な目標は、アプリケーションの俊敏性、柔軟性、および市場投入までの時間を短縮するために、分離された独立したコンポーネントを構築することです。デカップリングにより、各マイクロサービスコンポーネントには独自のデータ永続化レイヤーが設けられます。分散型アーキテクチャでは、ビジネストランザクションが複数のマイクロサービスにまたがる可能性があります。これらのマイクロサービスは単一のアトミック性、一貫性、分離、耐久性 (ACID) トランザクションを使用できないため、部分的なトランザクションになってしまう可能性があります。この場合、すでに処理されたトランザクションを元に戻すには、何らかの制御ロジックが必要です。ディストリビュートサガパターンは、通常、この目的のために使用されます。

サガパターンは、分散アプリケーションの一貫性を確立し、複数のマイクロサービス間のトランザクションを調整してデータ整合性を維持するのに役立つ障害管理パターンです。サガパターンを使用すると、トランザクションを実行するすべてのサービスがイベントをパブリッシュし、後続のサービスがチェーン内の次のトランザクションを実行するようトリガーします。これはチェーン内の最後のトランザクションが完了するまで続きます。ビジネストランザクションが失敗した場合、sagaは一連の補償トランザクションを組織し、それまでのトランザクションによって行われた変更を取り消します。

このパターンは、AWS Step Functions、AWS Lambda、Amazon DynamoDB などのサーバーレステクノロジーを使用して、サンプルアプリケーション (旅行の予約を処理する) の設定とデプロイを自動化する方法を示しています。サンプルアプリケーションでは、Saga 実行コーディネーターを実装するために Amazon API Gateway と Amazon Simple Notification Service (Amazon SNS) も使用します。パターンは、AWS Cloud Development Kit (AWS CDK)、AWS サーバーレスアプリケーションモデル (AWS SAM)、Terraform などのinfrastructure as code (IaC フレームワークを使用してデプロイできます。

Saga パターンとその他のデータ永続性パターンの詳細については、AWS 規範ガイダンス ウェブサイトの「マイクロサービスにおけるデータ永続性の有効化」ガイドを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- AWS CloudFormation スタックを作成するための権限。詳細については、「ACK ドキュメント」の「Install an ACK Controller」を参照してください。
- 任意の IaC フレームワーク (AWS CDK、AWS SAM、または Terraform) を AWS アカウントで設定することで、フレームワーク CLI を使用してアプリケーションをデプロイできます。
- NodeJSは、アプリケーションの構築とローカルでの実行に使用されます。
- 任意のコードエディター (Visual Studio Code、Sublime、Atom など)。

製品バージョン

- NodeJS バージョン 14
- AWS CDK バージョン 2.37.1
- AWS SAM バージョン 1.71.0
- Terraform バージョン 1.3.7

制約事項

イベントソーシングは、すべてのコンポーネントがゆるく結合されていて互いに直接認識できないマイクロサービスアーキテクチャに saga オーケストレーションパターンを実装する自然な方法です。トランザクションのステップ数が少ない (3 ~ 5) 場合は、サガパターンが最適かもしれません。ただし、マイクロサービスの数とステップの数が増えるにつれて、複雑さも増します。

この設計を使用すると、トランザクションパターンをシミュレートするためにすべてのサービスを実行する必要があるため、テストとデバッグが難しくなる可能性があります。

アーキテクチャ

ターゲットアーキテクチャ

提案されたアーキテクチャでは、AWS Step Functions を使用して、フライトの予約、レンタカーの予約、休暇の支払い処理といったサガパターンを構築しています。

以下のワークフロー図は、旅行予約システムの一般的なフローを示しています。ワークフローは、航空旅行の予約 (「ReserveFlight」)、車の予約 (「ReserveCarrental」)、支払いの処理 (「支払い処理」)、フライトの予約の確認 (「Confirm Flight」)、レンタカーの確認 (「Confirm Carrental」) で構成され、これらのステップが完了すると成功通知が送られます。ただし、これらのトランザクションのいずれかを実行中にエラーが発生すると、システムは逆方向にフェイルオーバーし始めます。たとえば、支払い処理 (「processPayment」) でエラーが発生すると返金 (「RefundPayment」) がトリガーされ、レンタカーとフライトのキャンセル

(「CancelRentalReservation」と「CancelFlightReservation」) がトリガーされ、トランザクション全体が失敗メッセージで終了します。

このパターンでは、図で強調表示されているタスクごとに個別の Lambda 関数がデプロイされるほか、フライト、レンタカー、支払い用の 3 つの DynamoDB テーブルもデプロイされます。各 Lambda 関数は、トランザクションが確認されたかロールバックされたかに応じて、それぞれの DynamoDB テーブルの行を作成、更新、または削除します。このパターンでは、Amazon SNS を使用してサブスクライバーにテキスト (SMS) メッセージを送信し、トランザクションが失敗または成功したことを通知します。

自動化とスケール

IaC フレームワークのいずれかを使用して、このアーキテクチャの構成を作成できます。お好みの Iac について、以下のリンクの 1 つを使用します。

- AWS CDK によるデプロイ
- AWS SAM によるデプロイ
- テラフォームによるデプロイ

ツール

AWS サービス

- AWS Step Functionsは、Lambda関数と他のサービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。Step Functions のグラフィカルコンソールでは、アプリケーションのワークフローを一連のイベント駆動型ステップとして確認できます。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。DynamoDB を使用して、任意の量のデータを保存および取得できるデータベーステーブルを作成し、任意のレベルのリクエストトラフィックを処理できます。
- AWS Lambda はサーバーをプロビジョニングしたり管理しなくてもコードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- Amazon API Gateway は、あらゆる規模の REST、HTTP、WebSocket API を作成、公開、管理、モニタリング、保護するための AWS のサービスです。

- Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバーへのメッセージ配信を提供するマネージドサービスです。
- 「AWS Cloud Development Kit (AWS CDK)」は、TypeScript、JavaScript、Python、Java、C#/.Net などの使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースを定義するためのソフトウェア開発フレームワークです。
- AWS サーバーレスアプリケーションモデル (AWS SAM) は、サーバーレスアプリケーションを構築するためのオープンソースフレームワークです。関数、API、データベース、イベントソースマッピングを表現するための省略構文を提供します。

コード

IaC テンプレート (AWS CDK、AWS SAM、または Terraform)、Lambda 関数、DynamoDB テーブルなど、サガパターンを示すサンプルアプリケーションのコードは、次のリンクにあります。最初のエピックの指示に従って、これらをインストールします。

- AWS CDK によるデプロイ
- AWS SAM によるデプロイ
- テラフォームによるデプロイ

エピック

パッケージのインストール、コンパイル、ビルド

タスク	説明	必要なスキル
npm パッケージをインストールします。	新しいディレクトリを作成し、ターミナルでそのディレクトリに移動し、このパターンの前の「コード」セクションから選択したGitHub リポジトリをクローンします。 `package.json` ファイルが含まれているルートフォルダーで、次のコマンドを実行し	開発者、クラウドアーキテクト

タスク	説明	必要なスキル
	て、すべての Node Package Manager (NPM) パッケージをダウンロードしてインストールします。 `npm install`	
Compile Script	ルートフォルダーで、次のコマンドを実行して、必要なすべての JavaScript ファイルを作成するように TypeScript トランスパイラーに指示します。 `npm run build`	開発者、クラウドアーキテクト
変更を監視して再コンパイルする。	ルートフォルダーで、別のターミナルウィンドウで次のコマンドを実行してコードの変更を監視し、変更が検出されたらコードをコンパイルします。 `npm run watch`	開発者、クラウドアーキテクト
ユニットテストを実行します (AWS CDK のみ)。	AWS CDK を使用している場合は、ルートフォルダで次のコマンドを実行して Jest ユニットテストを実行します。 `npm run test`	開発者、クラウドアーキテクト

ターゲット AWS アカウントにリソースをデプロイ

タスク	説明	必要なスキル

デモスタックを AWS にデプロイします。	**Important**
	アプリケーションは AWS リージョンに依存しません。プロファイルを使用する場合は、「AWS コマンドラインインターフェイス (AWS CLI) プロファイル」または「AWS CLI 環境変数」を使用してリージョンを明示的に宣言する必要があります。
	
	ルートフォルダで、次のコマンドを実行してデプロイアセンブリを作成し、デフォルトの AWS アカウントとリージョンにデプロイします。
	
	AWS CDK
	`cdk bootstrap`
	`cdk deploy`
	
	AWS SAM
	`sam build`
	`sam deploy --guided`
	
	Terraform
		開発者、クラウドアーキテクト

タスク	説明	必要なスキル

	`terraform init`
	`terraform apply`
	
	この処理は、完了まで数分、またはそれ以上かかる場合があります。このコマンドは、AWS CLI に設定されたデフォルトの認証情報を使用します。
	
	デプロイの完了後にコンソールに表示される API ゲートウェイ URL をメモしておきます。この情報は Saga 実行フローをテストする際に必要になります。	

デプロイされたスタックを現在の状態と比較します。	ルートフォルダーで以下のコマンドを実行して、ソースコードに変更を加えた後、デプロイされたスタックを現在の状態と比較します。
	
	AWS CDK
	
	`cdk diff`
	
	AWS SAM
	
	`sam deploy`
	
	Terraform
	
	`terraform plan`
		開発者、クラウドアーキテクト

実行フローのテスト

タスク	説明	必要なスキル
Saga 実行フローをテストします。	スタックをデプロイしたときに前のステップで書き留めた API ゲートウェイ URL に移動します。この URL により、ステートマシンが起動します。さまざまな URL パラメータを渡してステートマシンのフローを操作する方法の詳細については、「追加情報」セクションを参照してください。 結果を表示するには、AWS マネジメントコンソールにサインインし、Step Functions コンソールに移動します。ここでは、Saga ステートマシンのすべてのステップを確認できます。DynamoDB テーブルを表示して、挿入、更新、削除されたレコードを確認することもできます。画面を頻繁に更新すると、トランザクションのステータスが`pending`から`confirmed` に変わるのを確認できます。 予約が成功または失敗したときに SMS メッセージを受信するように、`stateMachine.ts` ファイル内のコードを携帯電話番号で更新することで SNS トピックを購読できます。詳細については、	開発者、クラウドアーキテクト

タスク	説明	必要なスキル
	「追加情報」セクションの「Amazon SNS」を参照してください。	

クリーンアップ

タスク	説明	必要なスキル

リソースをクリーンアップします。	このアプリケーションにデプロイされたリソースをクリーンアップするには、次のコマンドの 1 つを使用します。
	AWS CDK
	`cdk destroy`
	AWS SAM
	`sam delete`
	Terraform
	`terraform destroy`
		アプリ開発者、クラウドアーキテクト

関連リソース

テクニカルペーパー

- AWS でのマイクロサービスの実装
- サーバーレスアプリケーションレンズ
- マイクロサービスにおけるデータ永続化の有効化

AWS サービスのドキュメント

- AWS CDK の使用開始」
- AWS SAM の使用開始
- AWS Step Functions
- Amazon DynamoDB
- Lambda
- Amazon API Gateway
- Amazon SNS

チュートリアル

- サーバーレスコンピューティングのハンズオンワークショップ

追加情報

コード

テスト目的で、このパターンは API ゲートウェイ と、Step Functions ステートマシンをトリガーするテスト Lambda 関数をデプロイします。Step Functions を使用すると、「ReserveFlight」、「ReserveCarrental」、「ProcessPayment」、「Confirm Flight」、「Confirm CarRental」 の障害を模倣するrun_typeパラメータを渡すことで、旅行予約システムの機能を制御できます。

sagaLambda 関数 (sagaLambda.ts) は API ゲートウェイ URL のクエリパラメータから入力を受け取り、次の JSON オブジェクトを作成し、それをStep Functions に渡して実行させます。

```
let input = {
"trip_id": tripID, //  value taken from query parameter, default is AWS request ID
"depart_city": "Detroit",
"depart_time": "2021-07-07T06:00:00.000Z",
"arrive_city": "Frankfurt",
"arrive_time": "2021-07-09T08:00:00.000Z",
"rental": "BMW",
"rental_from": "2021-07-09T00:00:00.000Z",
"rental_to": "2021-07-17T00:00:00.000Z",
"run_type": runType // value taken from query parameter, default is "success"
};
```

次の URL パラメータを渡すことで、Step Functions ステートマシンのさまざまなフローを試すことができます。

- 実行成功 – https://{api gateway url}
- リザーブフライトの失敗 – https://{api gateway url}?runType=failFlightsReservation
- フライト確認の失敗 – https://{api gateway url}?runType=failFlightsConfirmation
- レンタカー予約の失敗 – https://{api gateway url}?runType=failCarRentalReservation
- レンタカー確認の失敗 – https：//{API ゲートウェイ url}? runType=レンタカー確認失敗
- 支払い処理失敗 – https://{api gateway url}?runType=failPayment
- トリップ ID を渡す – https://{api gateway url}?tripID={by default, trip ID will be the AWS request ID}

IaC テンプレート

リンクされたリポジトリには、サンプル旅行予約アプリケーション全体を作成するのに使用できる IaC テンプレートが含まれています。

- AWS CDK によるデプロイ
- AWS SAM によるデプロイ
- テラフォームによるデプロイ

DynamoDB テーブル

フライト、レンタカー、支払いテーブルのデータモデルは次のとおりです。

```
Flight Data Model:
 var params = {
      TableName: process.env.TABLE_NAME,
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: flightReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: flightReservationID},
        'depart_city' : {S: event.depart_city},
        'depart_time': {S: event.depart_time},
        'arrive_city': {S: event.arrive_city},
        'arrive_time': {S: event.arrive_time},
        'transaction_status': {S: 'pending'}
      }
    };

Car Rental Data Model:
var params = {
```

```
      TableName: process.env.TABLE_NAME,
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: carRentalReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: carRentalReservationID},
        'rental': {S: event.rental},
        'rental_from': {S: event.rental_from},
        'rental_to': {S: event.rental_to},
        'transaction_status': {S: 'pending'}
      }
    };

Payment Data Model:
var params = {
      TableName: process.env.TABLE_NAME,
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: paymentID},
        'trip_id' : {S: event.trip_id},
        'id': {S: paymentID},
        'amount': {S: "750.00"}, // hard coded for simplicity as implementing any
 monetary transaction functionality is beyond the scope of this pattern
        'currency': {S: "USD"},
        'transaction_status': {S: "confirmed"}
      }
    };
```

Lambda 関数

Step Functions でのステートマシンフローと実行をサポートするために、以下の関数が作成されます。

- フライトを予約：フライトを予約するために、`transaction_status`が`pending`であるレコードをDynamoDB Flightsテーブルに挿入します。
- フライトを確認：DynamoDB フライトテーブルのレコードを更新して`transaction_status`を`confirmed`に設定し、フライトを確認します。
- フライト予約をキャンセル：DynamoDB フライトテーブルからレコードを削除して、保留中のフライトをキャンセルします。
- レンタカーを予約する：レンタカーを予約するために、`transaction_status`が`pending`であるレコードをDynamoDB CarRentalsテーブルに挿入します。

- レンタカーの確認：DynamoDB CarRentals テーブルのレコードを更新してtransaction_statusをconfirmedに設定し、レンタカーを確認します。
- レンタカーの予約をキャンセル：DynamoDB CarRentals テーブルからレコードを削除して、保留中のレンタカーをキャンセルします。
- 支払い処理：支払いのレコードを DynamoDB 支払いテーブルに挿入します。
- 支払いをキャンセル：DynamoDB ペイメントテーブルから支払いのレコードを削除します。

Amazon SNS

サンプルアプリケーションでは、SMS メッセージを送信したり、予約の成功または失敗を顧客に通知したりするために、次のトピックとサブスクリプションを作成します。サンプルアプリケーションのテスト中にテキストメッセージを受信したい場合は、ステートマシン定義ファイル内の有効な電話番号で SMS サブスクリプションを更新してください。

AWS CDK スニペット (次のコードの 2 行目に電話番号を追加) :

```
const topic = new  sns.Topic(this, 'Topic');
topic.addSubscription(new subscriptions.SmsSubscription('+11111111111'));
const snsNotificationFailure = new tasks.SnsPublish(this ,'SendingSMSFailure', {
topic:topic,
integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
message: sfn.TaskInput.fromText('Your Travel Reservation Failed'),
});

const snsNotificationSuccess = new tasks.SnsPublish(this ,'SendingSMSSuccess', {
topic:topic,
integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
message: sfn.TaskInput.fromText('Your Travel Reservation is Successful'),
});
```

AWS SAM スニペット (+1111111111文字列を有効な電話番号に置き換える) :

```
  StateMachineTopic11111111111:
    Type: 'AWS::SNS::Subscription'
    Properties:
      Protocol: sms
      TopicArn:
        Ref: StateMachineTopic
      Endpoint: '+11111111111'
    Metadata:
```

```
      'aws:sam:path': SamServerlessSagaStack/StateMachine/Topic/+11111111111/Resource
```

Terraform スニペット (+111111111文字列を有効な電話番号に置き換える) :

```
resource "aws_sns_topic_subscription" "sms-target" {
  topic_arn = aws_sns_topic.topic.arn
  protocol  = "sms"
  endpoint  = "+11111111111"
}
```

予約成功

以下のフローは、「フライトの予約」、「レンタカーの予約」、「支払い処理」の後に「フライトの確認」と「レンタカーの確認」の順で予約が成功するフローです。予約が成功したことは、SNS トピックの購読者に送信される SMS メッセージを通じてお客様に通知されます。

予約失敗

このフローは、サガ· パターンの失敗の一例です。フライトやレンタカーの予約後に「ProcessPayment」が失敗すると、手順は逆の順序でキャンセルされます。 予約が解除され、SNS トピックのサブスクライバーに送信される SMS メッセージを通じてお客様に障害が通知されます。

AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。

作成者：Dr. Rahul Sharad Gaikwad (AWS)

概要

「Amazon ECS Anywhere」は、Amazon Elastic Container Service (Amazon ECS) の拡張機能です。ECS Anywhere を使用して、ネイティブの Amazon ECS タスクをオンプレミス環境または顧客管理型環境にデプロイできます。この機能は、コストを削減し、複雑なローカルコンテナのオーケストレーションと操作を軽減することに役立ちます。ECS Anywhere を使用して、オンプレミス環境とクラウド環境の両方でコンテナアプリケーションをデプロイして実行できます。これにより、チームが複数のドメインやスキルセットを習得しまたは複雑なソフトウェアを独自に管理しする必要がなくなります。

このパターンは、AWS Cloud Development Kit (「AWS CDK」) スタックを使用して ECS Anywhere を設定する手順を示しています。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI)は、「インストール」および「設定」されています。(AWS CLI ドキュメントの「AWS CLI のインストール、更新、アンインストール」を参照してください)。
- AWS CDK Toolkit がインストールされ、設定されています。(AWS CDK ドキュメントの「AWS CDK Toolkit」を参照し、手順に従ってバージョン 2 をグローバルにインストールします)。
- ノードパッケージマネージャー (npm) は、TypeScript で AWS CDK 用にインストールし設定されています。(npm ドキュメントの「Node.js と npm のダウンロードとインストール」を参照してください。)

機能制限

- 制限と考慮事項については、「Amazon ECS のドキュメント」の「外部インスタンス (Amazon ECS Anywhere)」を参照してください。

製品バージョン

- AWS CDK Toolkit バージョン 2
- npm バージョン 7.20.3 以降
- Node.js バージョン 16.6.1 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation
- AWS CDK
- Amazon ECS Anywhere
- AWS Identity and Access Management (IAM)

ターゲットアーキテクチャ

次の図は、このパターンで実装されたように、TypeScript でAWS CDK を使用する ECS Anywhere 設定の高レベルのシステムアーキテクチャを示しています。

1. AWS CDK スタックをデプロイする場合、AWS に CloudFormation スタックが作成されます。
2. CloudFormation スタックは Amazon ECS クラスターと関連する AWS リソースをプロビジョニングします。
3. Amazon ECS クラスターに外部インスタンスを登録するには、仮想マシン (VM) に AWS Systems Manager Agent (SSM Agent) をインストールし、その VM を AWS Systems Manager 管理型インスタンスとして登録する必要があります。
4. また、Amazon ECS コンテナエージェントと Docker を VM にインストールして、Amazon ECS クラスターの外部インスタンスとして登録する必要があります。
5. 外部インスタンスを Amazon ECS クラスターに登録して設定すると、外部インスタンスとして登録された VM 上で複数のコンテナを実行できます。

自動化とスケール

このパターンで提供される「GitHub リポジトリ」、AWS CDK をInfrastructure as Code (IaC) ツールとして使用して、このアーキテクチャの設定を作成します。AWS CDK は、リソースのオーケストレーションと ECS Anywhere のセットアップに役立ちます。

ツール

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

コード

このパターンのソースコードは、「Amazon ECS Anywhere CDK Samples」リポジトリにある GitHub で利用できます。リポジトリをクローンして使用するには、次のセクションの指示に従います。

エピック

AWS CDK の設定を確認

タスク	説明	必要なスキル
AWS CDK のバージョンを確認します。	以下のコマンドを使用して、AWS CDK Toolkit のバージョンを確認します。 `cdk --version` このパターンには AWS CDK バージョン 2 が必要です。旧バージョンの AWS CDK を使用している場合は、「AWS CDK のドキュメント」の指示に従って更新してください。	DevOps エンジニア
AWS 認証情報を設定します。	認証情報を設定するには、`aws configure` コマンドを実行し、プロンプトに従ってください。 `$aws configure`	DevOps エンジニア

タスク	説明	必要なスキル
	AWS Access Key ID [None]: <your-access-key-ID> AWS Secret Access Key [None]: <your-secret-access-key> Default region name [None]: <your-Region-name> Default output format [None]:	

AWS CDK 環境の起動

タスク	説明	必要なスキル
AWS SAM コードリポジトリを複製します。	以下のコマンドを使用して、このパターンの GitHub コードリポジトリを複製します。 `git clone https://github.com/aws-samples/amazon-ecs-anywhere-cdk-samples.git`	DevOps エンジニア
環境を起動します。	使用したいアカウントと AWS リージョンに AWS CloudFormation テンプレートをデプロイするには、以下のコマンドを実行します。 `cdk bootstrap <account-number>/<Region>` 詳細については、AWS CDK ドキュメントの「ブートスト	DevOps エンジニア

タスク	説明	必要なスキル
	ラップ」を参照してください。	

プロジェクトを構築してデプロイ

タスク	説明	必要なスキル

パッケージの依存関係をインストールし、TypeScript ファイルをコンパイルします。	パッケージの依存関係をインストールし、次のコマンドを実行して TypeScript ファイルをコンパイルします。
	`$cd amazon-ecs-anywhere-cdk-samples`
	`$npm install`
	`$npm fund`
	これらのコマンドは、すべてのパッケージをサンプルリポジトリからインストールします。
	Important
	パッケージの欠落に関するエラーが発生した場合は、次のいずれかのコマンドを使用します。
	`$npm ci`
	—または—
		DevOps エンジニア

タスク	説明	必要なスキル
	`$npm install -g @aws-cdk/<package_name>` 詳細については、npm のドキュメントの「npm ci」と「npm install」を参照してください。	
プロジェクトをビルドします。	プロジェクト コードをビルドするには、次のコマンドを実行します。 `npm run build` プロジェクトの構築とデプロイの詳細については、「AWS CDK のドキュメント」の「初めての AWS CDK アプリケーション」を参照してください。	DevOps エンジニア
プロジェクトをデプロイします。	プロジェクトコードをデプロイするには、コマンドを実行します。 `cdk deploy`	DevOps エンジニア
スタックの作成と出力を検証します。	「https://console.aws.amazon.com/cloudformation」で AWS CloudFormation コンソールを開き、 `EcsAnywhereStack` stack. Outputs タブを選択して、外部 VM で実行するコマンドが表示されます。	DevOps エンジニア

オンプレミスマシンの設定

タスク	説明	必要なスキル
Vagrant を使用して VM を設定します。	デモンストレーションの目的で、「HashiCorp Vagrant」を使用して VM を作成できます。Vagrant は、ポータブルな仮想ソフトウェア開発環境を構築し保守するためのオープンソースユーティリティです。Vagrantfile が置かれているルートディレクトリから `vagrant up` コマンドを実行して Vagrant VM を作成します。詳細については、「Vagrant のドキュメント」を参照してください。	DevOps エンジニア
VM を外部インスタンスとして登録します。	1. `vagrant ssh` コマンドを使用して Vagrant VM にログインします。詳細については、「Vagrant のドキュメント」を参照してください。 2. VM を AWS Systems Manager に登録しまたは外部インスタンスをアクティブ化するために使用できるアクティベーションコードと ID を作成します。このコマンドからの出力に、 `ActivationId` と `ActivationCode` 値が含まれます。 `aws ssm create-activation --iam-role EcsAnywhereInstanc`	DevOps エンジニア

タスク	説明	必要なスキル
	eRole | tee ssm-activ ation.json 3. アクティベーション ID とコード値を出力します。 export ACTIVATIO N_ID=<activation-ID> export ACTIVATIO N_CODE=<activation- code> 4. オンプレミスのサーバーまたは仮想マシン (VM) で、インストールスクリプトをダウンロードします。 curl -o "ecs-anywhere- install.sh" "https:// amazon-ecs-agent.s 3.amazonaws.com/ec s-anywhere-install -latest.sh" && sudo chmod +x ecs-anywhere- install.sh 5. オンプレミス サーバーまたは VM でインストール スクリプトを実行します。 sudo ./ecs-anywhere-ins tall.sh \ --cluster test-ecs- anywhere \ --activation-id $ACTIVATION_ID \ --activation-code $ACTIVATION_CODE \	

タスク	説明	必要なスキル
	`--region <Region>` VM の設定と登録の詳細については、「Amazon ECS のドキュメント」の「クラスターへの外部インスタンスの登録」を参照してください。	

ECS Anywhere と外部 VM のステータスを確認してください。	仮想ボックスが Amazon ECS コントロールプレーンに接続され、実行中になっているかを確認するには、以下のコマンドを使用します。
	
	`aws ssm describe-instance-information`
	`aws ecs list-container-instances --cluster $CLUSTER_NAME`
		DevOps エンジニア

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップして削除する。	このパターンを確認したら、追加料金が発生しないように、作成したリソースを削除する必要があります。クリーンアップするには、以下のコマンドを実行します。 `cdk destroy`	DevOps エンジニア

関連リソース

- 「Amazon ECS Anywhere のドキュメント」
- 「Amazon ECS Anywhere デモ」
- 「Amazon ECS Anywhere Workshop Samples」

ASP.NET ウェブフォームアプリケーションを AWS で最新化

作成者：Vijai Anand Ramalingam (AWS) と Sreelaxmi Pai (AWS)

概要

このパターンでは、レガシーのモノリス ASP.NET://www.com Web Forms アプリケーションを AWS の ASP.NET://www.com.com. Core に移植してモダナイズする手順について説明します。

ASP.NET ウェブフォームアプリケーションを ASP.NET Core に移行すると、Linux のパフォーマンスの利点、コスト削減、Linuxの強固なエコシステムを活用できます。ただし、これは手作業による多大な労力がかかる可能性があります。このパターンでは、レガシーアプリケーションは段階的なアプローチを使用して段階的に最新化され、その後 AWS クラウドにコンテナ化されます。

ショッピングカートのレガシー全体のアプリケーションを検討します。ASP.NET Web フォームアプリケーションとして作成され、コードビハインド (`aspx.cs`) ファイルを含む [.aspx] ページで構成されていると仮定してみましょう。最新化プロセスには、3 つのステップがあります。

1. 適切な分解パターンを使用して、モノリスをマイクロサービスに分割します。詳細については、AWS 規範ガイダンスのウェブサイトにある「モノリスをマイクロサービスに分解する」ガイドを参照してください。
2. レガシー ASP.NET ウェブフォーム (.NET フレームワーク) アプリケーションを.NET 5 以降の ASP.NET Core に移行します。このパターンでは、Porting Assistant for .NET でASP.NET Web フォームアプリケーションをスキャンし、ASP.NET Core との非互換性を確認します。これにより、手動での移行作業が軽減されます。
3. React を使用して Web Forms UI layer を再開発します。このパターンには UI の再開発が含まれていません。手順については、「React のドキュメント」の「React アプリを作成」を参照してください。
4. ウェブフォームのコードビハインドファイル (ビジネスインターフェース) を ASP.NET Core web API として再開発します。このパターンでは、NDepend レポートで必要なファイルと依存関係を識別しやすくなります。
5. Porting Assistant for .NET で、レガシーアプリケーション内のビジネスロジックやデータアクセスなどの共有プロジェクトや共通プロジェクトを .NET 5 以降にアップグレードします。
6. AWS サービスを追加してアプリケーションを補完します。たとえば、「Amazon CloudWatch Logs」を使用して、アプリケーションのログのモニタリング、保存、アクセスを行うことができます。また、「AWS Systems Manager」を使用して、アプリケーションの設定を保存できます。

7. 最新のASP.NET Core Applicationsをコンテナ化します。このパターンでは、Visual Studio の Linux を対象とする Docker ファイルを作成し、Docker Desktop でローカルでテストします。このステップは、レガシーアプリケーションがオンプレミスまたは Amazon Elastic Compute Cloud (Amazon EC2) Windows インスタンスですでに実行されていることを前提としています。詳細については、「Amazon EC2 Linux インスタンスで ASP.NET Core web API Docker コンテナを実行する」というパターン」を参照してください。
8. 最新の ASP.NET Core アプリケーションを、Amazon Elastic Container Service (Amazon ECS) にデプロイします。このパターンはデプロイステップには適用されません。手順については、「Amazon ECS Workshop」を参照してください。

> **Note**
>
> このパターンは、UI 開発、データベースモダナイゼーション、またはコンテナデプロイのステップには適用されません。

前提条件と制限

前提条件

- 「Visual Studio」または「Visual Studio Code」をダウンロードし、インストールします。
- AWS マネジメントコンソールと AWS コマンドラインインターフェイス (AWS CLI) バージョン 2 で AWS アカウントにアクセスします。(「AWS CLI の設定手順」を参照してください。)
- AWS Toolkit for Visual Studio (「セットアップ手順」をを参照)。
- 「ダウンロード済み」でインストール済みのDocker デスクトップ。
- 「ダウンロード済み」でインストール済みの .NET SDK、。
- 「ダウンロード済み」でインストール済みの NDepend ツール。Visual Studio 用の NDepend 拡張機能をインストールするには、`NDepend.VisualStudioExtension.Installer` を実行します (手順を参照)。ご要件に応じて、Visual Studio 2019 または 2022 を選択できます。
- 「 ダウンロード済みでインストール済み 」の Porting Assistant for .NET。

アーキテクチャ

ショッピングカートアプリケーションの最新化

次の図は、従来の ASP.NET ショッピングカートアプリケーションの最新化プロセスを示しています。

ターゲットアーキテクチャ

次の図は、AWS の最新のショッピングカートアプリケーションのアーキテクチャを示しています。ASP.NET Core Web API は、Amazon ECS クラスターにデプロイされます。記録と設定サービスは Amazon CloudWatch Logs と AWS Systems Manager により、提供されています。

ツール

AWS サービス

- 「Amazon ECS」— Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理に使用される、高度にスケーラブルで高速のコンテナ管理サービスです。AWS Fargate が管理するサーバーレスインフラ上でタスクやサービスを実行できます。または、インフラストラクチャをより詳細に制御するために、管理する EC2 インスタンスのクラスターでタスクとサービスを実行できます。
- 「Amazon CloudWatch Logs」— Amazon CloudWatch Logs により、使用中のすべてのシステム、アプリケーション、AWS のサービスからのログを、一元管理することができます。ログを表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。
- 「AWS Systems Manager」— AWS Systems Manager は、AWS でインフラストラクチャの表示と制御に使用できる AWS サービスです。Systems Manager コンソールを使用すると、複数の AWS サービスからの運用データを表示し、AWS リソース全体の運用タスクを自動化できます。Systems Manager は、マネージドインスタンスをスキャンし、検出されるポリシー違反を報告（または是正措置を講じる）して、セキュリティとコンプライアンスを維持できます。

ツール

- 「Visual Studio」または「Visual Studio Code」— .NET アプリケーション、ウェブ APIおよびその他のプログラムを構築するためのツール。
- 「AWS Toolkit for Visual Studio」— AWS サービスを使用する .NET アプリケーションの開発、デバッグ、デプロイに役立つ Visual Studio の拡張機能。

- 「Docker Desktop」— コンテナ化されたアプリケーションの構築とデプロイを簡単にするツール。
- 「NDepend」— .NET コードの依存関係、品質問題、コード変更をモニタリングするアナライザー。
- 「Porting Assistant for .NET」— .NET コードをスキャンして.NET Core との非互換性を特定し、移行作業を見積もる分析ツール。

エピック

レガシーアプリケーションを .NET 5 以降のバージョンに移行

タスク	説明	必要なスキル
.NET Framework のレガシーアプリケーションを.NET 5 にアップグレードします。	Porting Assistant for .NET で、従来の ASP.NET Web フォームアプリケーションを.NET 5 以降に変換できます。「Porting Assistant for .NET のドキュメント」の指示に従ってください。	アプリ開発者
NDepend レポートを生成します。	ASP.NET Web フォームアプリケーションをマイクロサービスに分解して最新化すると、レガシーアプリケーションのすべての [.cs] ファイルが必要なくなる場合があります。NDepend を使用すると、任意のコードビハインド (.cs) ファイルのレポートを生成して、すべての呼び出し元と呼び出し先を取得できます。このレポートは、マイクロサービス内の必要なファイルのみを特定して使用するのに役立ちます。	アプリ開発者

タスク	説明	必要なスキル
	NDepend をインストールしたら (「前提条件」セクションを参照)、Visual Studio でレガシーアプリケーションのソリューション (.sln ファイル) を開き、次の手順に従います。 1. Visual Studio でレガシーアプリケーションを構築します。 2. Visual Studio のメニューバーで nDepend と新しい NDepend プロジェクトを現在の VS ソリューションに添付を選択します。 3. .NET アセンブリの分析を選択します。 4. 分析が完了したら、Solution Explorer でプロジェクトに移動します。レポートを生成したいコードビハインドファイル (`listproducts.aspx.cs` など) を右クリックして、Show on Dependency Graph を選択します。 5. ナビゲーションバーで [呼び出し元と呼び出し先] を選択して、コードクエリの編集を選択します。 6. クエリとルール編集ペインで、ダウンロード矢印を	

タスク	説明	必要なスキル
	選択して、Excel にエクスポートを選択します。 このプロセスにより、すべての呼び出し元と呼び出し先を一覧表示するコードビハインドファイルのレポートが生成されます。ディペンデンシーグラフについて、詳細は、「NDepend のドキュメント」を参照してください。	

タスク	説明	必要なスキル
新しい .NET 5 ソリューションを作成します。	最新の ASP.NET Core web API 用の新しい .NET 5 (またはそれ以降) 構造を作成するには: 1. Visual Studio を開きます。 2. 新しい空のソリューションを作成します。 3. レガシーアプリケーションに基づき、.NET 5 (またはそれ以降) をターゲットとする新しいプロジェクトを作成します。ショッピングカートアプリケーションのレガシープロジェクトと新しいプロジェクトの例については、「追加情報」セクションを参照してください。 4. 前のステップで説明した NDepend レポートを使用して、必要なファイルをすべて識別してください。以前にアップグレードしたアプリケーションからこれらのファイルをコピーし、新しいソリューションに追加します。 5. ソリューションを構築し、すべての問題を見直します。 プロジェクトとソリューションの作成について、詳細は、	アプリ開発者

タスク	説明	必要なスキル
	「Visual Studio のドキュメント」を参照してください。 **Note** ソリューションを構築し、機能を検証するときに、NDepend が識別したファイルに加えて、ソリューションに追加する複数のファイルを特定できます。	

アプリケーションコードを更新します。

タスク	説明	必要なスキル
ASP.NET Core でウェブ API を実装します。	従来のモノリスショッピングカートアプリケーションで特定したマイクロサービスの 1 つが製品だと仮定しましょう。前のエピックで製品用の ASP.NET Core Web API プロジェクトを新規作成しました。このステップでは、製品に関連するすべての Web フォーム (.aspx ページ) を識別して最新化します。前述の「アーキテクチャ」セクションで説明したように、 [製品] が 4 つのウェブフォームで構成されていると仮定します。	アプリ開発者

タスク	説明	必要なスキル
	• 製品の一覧表示 • 製品の表示 • 製品の追加/編集 • 製品の削除 各 ウェブフォームを分析し、何らかのロジックを実行するためにデータベースに送信されるすべてのリクエストを識別し、応答を取得する必要があります。各リクエストをウェブ API エンドポイントとして実装できます。ウェブフォームを考慮すると、製品には以下のエンドポイントを設定できます。 • `/api/products` • `/api/products/{id}` • `/api/products/add` • `/api/products/update/{id}` • `/api/products/delete/{id}` 前述のように、ビジネスロジック、データアクセス、共有/共通プロジェクトなど、.NET 5 にアップグレードした他のすべてのプロジェクトを再利用できます。	

タスク	説明	必要なスキル
Amazon CloudWatch Logs を設定します。	「Amazon CloudWatch Logs」を使用して、アプリケーションのログのモニタリング、保存、アクセスを行うことができます。AWS SDK で Amazon CloudWatch Logs にデータを記録できます。また、「NLog」、「Log4Net」、「ASP.NET Core ロギングフレームワーク」など一般的な.NET ロギングフレームワークを使用して、.NET アプリケーションを CloudWatch Logs と統合することもできます。 このステップの詳細については、ブログ記事の「Amazon CloudWatch Logs と.NET ロギングフレームワーク」を参照してください。	アプリ開発者

タスク	説明	必要なスキル
AWS Systems Manager Parameter Store を設定します。	「AWS Systems Manager Parameter Store」で、接続文字列などのアプリケーション設定をアプリケーションコードとは別に保存できます。NuGet パッケージ「Amazon.Extensions.Configuration.SystemsManager」を使用すると、アプリケーションがこれらの設定を AWS Systems Manager Parameter Store から .NET Core 設定システムにロードする方法が簡単になります。 このステップの詳細については、ブログ記事の「AWS Systems Manager向けの.NET Core 設定」を参照してください。	アプリ開発者

認証と認可の追加

タスク	説明	必要なスキル
認証には共有 Cookie を使用してください。	従来のモノリスアプリケーションの最新化は反復的なプロセスであり、モノリスと最新化されたバージョンが共存する必要があります。共有 Cookie を使用すると、2 つのバージョン間でシームレスな認証を実現できます。最新の ASP.NET Core アプリケー	アプリ開発者

タスク	説明	必要なスキル
	ションが Cookie を検証している間、従来の ASP.NET アプリケーションは引き続きユーザーの認証情報を検証して Cookie を発行します。 手順とサンプルコードについては、「サンプル GitHub プロジェクト」を参照してください。	

コンテナをローカルで構築して実行

タスク	説明	必要なスキル
Visual Studio を使用して Docker イメージを作成します。	このステップでは、Visual Studio for .NET Core web API を使用して Docker ファイルを作成します。 1. Visual Studio を開きます。 2. Solution Explorer でお客様のプロジェクトのコンテキスト（右クリック）メニューから、Add/Docker Support を選択します。 3. ターゲットオペレーティングシステムとして Linux を選択します。 Visual Studio はお客様のプロジェクト用の Docker ファイルを作成します。サンプル Docker ファイルについては	アプリ開発者

タスク	説明	必要なスキル
	、Microsoft のウェブサイトで「Docker 用 Visual Studio Container Tools」を参照してください。	

タスク	説明	必要なスキル

Docker Desktop を使用してコンテナを構築して実行します。	これで Docker Desktop でコンテナを構築、作成、実行できるようになりました。
	
	1. [コマンドプロント] ウィンドウを開きます。Docker ファイルが存在するソリューションフォルダーに移動します。次のコマンドを実行して Docker イメージを作成します。
	
	`docker build -t aspnetcorewebapiimage -f Dockerfile .`
	
	2. 次のコマンドを実行して、すべての Docker イメージを表示します。
	
	`docker images`
	
	3. 次のコマンドを実行して、コンテナを作成し、実行します。
	
	`docker run -d -p 8080:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage`
	
	4. Docker Desktop を開き、Containers/Apps を選択します。aspnetcorewebapicontainer
		アプリ開発者

タスク	説明	必要なスキル
	実行中という新しいコンテナが表示されます。	

関連リソース

- 「Amazon EC2 Linux インスタンスで ASP.NET Core web API Docker コンテナを実行する」(AWS 規範ガイダンス)
- 「Amazon ECS Workshop」
- 「AWS CloudFormation を使用したCodeDeploy による ECS ブルー/グリーンデプロイの実行」(AWS CloudFormationドキュメント)
- 「NDepend で使用開始」(NDepend のドキュメント)
- 「Porting Assistant for .NET」

追加情報

次の表は、従来のショッピングカートアプリケーションのサンプルプロジェクトと、最新の ASP.NET Core アプリケーションの同等のプロジェクトの例を示します。

レガシーソリューション：

[Project name] (プロジェクト名)	プロジェクトテンプレート	Target framework
ビジネスインターフェイス	クラスライブラリ	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。
BusinessLogic	クラスライブラリ	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。
WebApplication	ASP.NET フレームワークアプリケーション	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。

UnitTests	NUnit Test Project	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。
共有 -> 共通	クラスライブラリ	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。
共有-> フレームワーク	クラスライブラリ	特定のランタイムライブラリまたは .NET Framework の最小バージョンが必要です。

新しいソリューション：

[Project name] (プロジェクト名)	プロジェクトテンプレート	Target framework
BusinessLogic	クラスライブラリ	.NET 5.0
<WebAPI>	ASP.NET Core Web API	.NET 5.0
<WebAPI>.UnitTests	NUnit 3 Test Project	.NET 5.0
共有 -> 共通	クラスライブラリ	.NET 5.0
共有-> フレームワーク	クラスライブラリ	.NET 5.0

AWS Fargate を使用して、イベント駆動型でスケジュール済みの大規模ワークロードを実行する

作成者: HARI OHM PRASATH RAJAGOPAL (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら

このパターンでは、AWS Fargate を使用して、Amazon Web Services (AWS) クラウド上で、スケジュール済みでイベント駆動型の大規模なワークロードを実行する方法を説明しています。

このモード設定のユースケースでは、プルリクエストが送信されるたびに、AWS アカウント番号や認証情報などの AWS 機密データを検索するためにコードがスキャンされます。プルリクエストが Lambda 関数を開始します。Lambda 関数は、コードスキャンを処理する Fargate タスクを呼び出します。Lambda は、新たなプルリクエストが発生するたびに開始されます。スキャンで機密情報が検出されると、Amazon Simple Notification Service (Amazon SNS) はスキャン結果を E メールメッセージで送信します。

このパターンは、以下に示すビジネスユースケースで役立ちます。

- ランタイム (15 分という制限) やメモリの制限が原因で、AWS Lambda によって実行できないスケジュールされたイベント駆動型のワークロードを多数実行する必要がある場合
- これらのワークロード向けにプロビジョニングされたインスタンスを AWS に管理させる場合

このパターンを使用すると、新しい仮想プライベートクラウド (VPC)を作成することができます。このパターンでは も使用します AWS CodeCommit。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS CodeCommit でコードベースをホストし、プルリクエストを作成している
- macOS、Linux、または Windows にインストールおよび設定されている AWS コマンドラインインターフェイス (AWS CLI) バージョン 1.7 以降
- コンテナでワークロードを実行している
- クラスパスに Apache Maven 実行ファイルが設定されている

アーキテクチャ

以下のステップが全体のフローに含まれます。

1. CodeCommit で新しいプルリクエストが送信されるたびに Lambda 関数が開始されます。Lambda 関数は、Amazon EventBridge を介して`CodeCommit Pull Request State Change`イベントをリッスンします。
2. Lambda関数は、コードをチェックおよびスキャンするために次の環境パラメータを持つ新しい Fargate タスクを送信します。

```
RUNNER # <<TaskARN>>
SNS_TOPIC # <<SNSTopicARN>>
SUBNET # <<Subnet in which Fargate task gets launched>>
```

   スキャンによってコード内の機密データが検出された場合、Fargate は Amazon SNS トピックに新しいメッセージをプッシュします。
3. SNS サブスクライバーはトピックからメッセージを読み取り、メールメッセージを送信します。

テクノロジー

- AWS CodeCommit
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon SNS
- Docker

ツール

ツール

- AWS CLI – AWS コマンドラインインターフェイス (CLI) は、AWS のサービスを管理するための統合ツールです。
- AWS CodeCommit – AWS CodeCommit は、セキュアな Git ベースのリポジトリをホストするフルマネージド型のソース管理サービスです。CodeCommit を使用すると、セキュアでスケーラブルな環境で、チームが共同でコードを作成できます。
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) は、フルマネージド型の Docker コンテナレジストリで、Docker コンテナイメージの保存、管理、デプロイを行うことができます。
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) は、非常にスケーラブルで高速なコンテナ管理サービスです。Amazon ECS を使用して、クラスター上のコンテナを実行、停止、管理できます。
- 「AWS Fargate」— AWS Fargate は、Amazon ECS で使用できるテクノロジーであり、サーバーや Amazon EC2 インスタンスのクラスターを管理することなくコンテナを実行できます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバー (または生産者から消費者) へのメッセージ配信を提供するマネージドサービスです。パブリッシャーは、論理アクセスポイントおよび通信チャネルであるトピックにメッセージを送信することで、受信者と非同期的に通信します。SNS トピックにサブスクライブしているクライアントは、サポートされているプロトコルを使用して、Lambda、E メール、モバイル プッシュ通知、モバイル テキストメッセージ (SMS) などのパブリッシングメッセージを受信します。
- Docker は、コンテナと呼ばれるパッケージでアプリケーションをビルド、テスト、配信するのに役立ちます。
- Git クライアント — 必要なアーティファクトをチェックアウトするためのコマンドラインまたはデスクトップツール
- Maven — Apache Maven は、プロジェクトのビルド、レポート、文書化を一元管理するためのプロジェクト管理ツールです。

エピック

ローカルリポジトリをセットアップする

タスク	説明	必要なスキル
コードをダウンロードします。	[添付ファイル] セクションで、.zip ファイルをダウンロードし、ファイルを抽出します。	開発者、AWS システム管理者
リポジトリを設定します。	ルートフォルダから `mvn clean install` を実行します。	開発者、AWS システム管理者

Amazon ECR イメージを作成し、イメージをプッシュする

タスク	説明	必要なスキル
Amazon ECR でリポジトリを作成して、ログインします。	Amazon EMR コンソールを開きます。ナビゲーションペインで、[リポジトリ] を選択し、[リポジトリの作成] を選択します。このストーリーやその他のストーリーに関するヘルプは、「関連リソース」セクションを参照してください。	開発者、AWS システム管理者
コンテナイメージをプッシュします。	リポジトリを開き、[プッシュコマンドの表示] を選択して、Docker にログインします。ログインしたら、[追加情報] セクションの [コンテナイメージをプッシュする] にあるコマンドを、必要な置換を加えて実行します。	開発者、AWS システム管理者

タスク	説明	必要なスキル
	これにより、コードスキャンに使用される Docker コンテナイメージがアップロードされます。アップロードの完了後、Amazon ECR リポジトリに最新ビルドの URL をコピーします。	

CodeCommit リポジトリを作成する

タスク	説明	必要なスキル
CodeCommit リポジトリを作成します。	新しい AWS CodeCommit リポジトリを作成するには、[追加情報] セクションの [CodeCommit リポジトリの作成] にあるコマンドを実行します。	開発者、AWS システム管理者

VPC の作成 (オプション)

タスク	説明	必要なスキル
VPC を作成します。	既存の VPC ではなく新規の VPC を使用する場合は、[追加情報] セクションの [VPC の作成] にあるコマンドを実行します。AWS Cloud Development Kit (AWS CDK) スクリプトは、作成された VPC とサブネットの ID を出力します。	開発者、AWS システム管理者

Amazon ECS クラスターと Fargate タスクを作成する

タスク	説明	必要なスキル
クラスターとタスクを作成します。	Amazon ECS クラスターおよび Fargate タスク定義を作成するには、[追加情報] セクションの [クラスターおよびタスクの作成] の下にあるコマンドを実行します。シェルスクリプトを実行するときに、正しい VPC ID と Amazon ECR リポジトリ URI がパラメータとして渡されていることを確認してください。このスクリプトは、(スキャンを行う責任のある) Docker イメージを指す Fargate タスク定義を作成します。次に、スクリプトは、ジョブおよびそれに関連付けられた実行ロールを作成します。	開発者、AWS システム管理者
Amazon EKS クラスターを検証します。	Amazon ECS コンソールを開きます。ナビゲーションペインで、[クラスター] を選択し、新しく作成した Fargate-Job-Cluster という名前の Amazon ECS クラスターを選択します。その後、ナビゲーションペインで [タスク定義] を選択し、`awscdkfargateecsTaskDef` プレフィックスの付いた新しいタスク定義があることを確認します。	開発者、AWS システム管理者

SNS トピックとサブスクライバーを作成する

タスク	説明	必要なスキル
SNS トピックを作成します。	SNS トピックを作成するには、[追加情報] セクションの [SNS トピックの作成] にあるコマンドを実行します。作成に成功したら、次の手順で使用する SNS ARN を書き留めます。	開発者、AWS システム管理者
SNS トピックとサブスクライバーを作成します。	SNS トピックのメールサブスクライバーを作成するには、[追加情報] セクションの [SNS サブスクライバーの作成] にあるコマンドを実行します。CLI コマンドで使用されている TopicARN と Email address は必ず置き換えてください。メール通知を受信するには、サブスクライバーとして使用しているメールアドレスを確認してください。	開発者、AWS システム管理者

Lambda 関数と CodeCommit トリガーを作成する

タスク	説明	必要なスキル
関数とトリガーを作成します。	CodeCommit トリガーを使用して Lambda 関数を作成するには、[追加情報] セクションにある「Lambda 関数と CodeCommit トリガー」コマンドを実行します コマンドを実行する前に、パラメータを	開発者、AWS システム管理者

タスク	説明	必要なスキル
	適切な値に置き換えてください。このスクリプトは、Lambda 関数を作成し、新しいプルリクエストが行われたときに呼び出されるように設定します。	

アプリケーションをテストする

タスク	説明	必要なスキル
アプリケーションをテストします。	AWS の機密データを CodeCommit リポジトリにチェックインする場合は、Lambda 関数を開始する必要があります。Lambda 関数は Fargate タスクを開始します。Fargate タスクはコードをスキャンし、スキャン結果をメールで送信します。	開発者、AWS システム管理者

関連リソース

- 「Amazon ECR リポジトリの作成」
- 「Amazon ECR に Docker イメージをプッシュする」

追加情報

「コンテナイメージをプッシュ」

```
> cd 1-ecr-image-push
> ./run.sh <<ecr-repository>>
```

CodeCommit リポジトリを作成する

```
aws codecommit create-repository --repository-name test-repo --repository-description
 "My Test repository"
```

「VPC を作成する」

```
> cd 2-create-vpc
> ./run.sh
```

出力

```
aws-batch-cdk-vpc-efs-launch-template.privatesubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.publicsubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.vpcid = vpc-<<id>>
```

「クラスターとタスクを作成する」

```
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 3-create-ecs-task
> ./run.sh <<vpc-id>> <<ecr-repo-uri>>
```

出力

```
aws-cdk-fargate-ecs.CLUSTERNAME = Fargate-Job-Cluster
aws-cdk-fargate-ecs.ClusterARN = <<cluster_arn>>
aws-cdk-fargate-ecs.ContainerARN = Fargate-Container
aws-cdk-fargate-ecs.TaskARN = <<task_arn>>
aws-cdk-fargate-ecs.TaskExecutionRole = <<execution_role_arn>>
aws-cdk-fargate-ecs.TaskRole = <<task_role_arn>>
```

「SNS トピックを作成する」

```
aws sns create-topic --name code-commit-topic
```

「SNS サブスクライバーを作成する」

```
aws sns subscribe \
```

```
    --topic-arn <<topic_arn>> \
    --protocol email \
    --notification-endpoint <<email_address>>
```

「Lambda 関数と CodeCommit トリガー」

```
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 5-Lambda-CodeCommit-Trigger
> ./run.sh <<taskarn>> <<snstopicarn>> subnet-<<id>> <<codecommitarn>>
```

出力

```
aws-cdk-fargate-lambda-event.Cloudwatchrule = <<cloudwatchrule>>
aws-cdk-fargate-lambda-event.CodeCommitLambda = AWS-Code-Scanner-Function
aws-cdk-fargate-lambda-event.LambdaRole = <<lambdaiamrole>>
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

C# と AWS CDK を使用するサイロモデル用の SaaS アーキテクチャでのテナントオンボーディング

作成者:Tabby Ward (AWS), Susmitha Reddy Gankidi (AWS), and Vijai Anand Ramalingam (AWS)

概要

Software as a service (SaaS) アプリケーションは、さまざまなアーキテクチャモデルで構築できます。サイロモデルとは、テナントに専用のリソースを提供するアーキテクチャを指します。

SaaS アプリケーションは、新しいテナントを環境に導入する際に摩擦のないモデルに依存しています。新しいテナントを作成するのに必要なすべての要素を正常にプロビジョニングして構成するには、多くの場合、多数のコンポーネントのオーケストレーションが必要になります。SaaS アーキテクチャでは、このプロセスをテナントオンボーディングと呼びます。オンボーディングは、オンボーディングプロセスのコードとしてのインフラストラクチャを活用して、すべての SaaS 環境で完全に自動化する必要があります。

このパターンでは、Amazon Web Services (AWS) でテナントを作成し、そのテナントの基本インフラストラクチャをプロビジョニングする例を紹介します。パターンは、C# と AWS Cloud Development Kit (AWS CDK) を使用します。

このパターンでは課金アラームが発生するため、スタックを米国東部 (バージニア北部) または us-east-1、AWS リージョンにデプロイすることをお勧めします。詳細については、AWS ドキュメントを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- このパターンの AWS リソースを作成するのに十分な IAM アクセス権を持つ AWS Identity and Access Management (IAM) プリンシパル。詳細については、「IAM ロール」を参照してください。
- 「Amazon Command Line Interface (AWS CLI) をインストール」し、AWS CDK のデプロイを実行するように「AWS CLI を設定」します。
- 「Visual Studio 2022」をダウンロードしてインストールするか、「Visual Studio Code」をダウンロードしてインストールしました。
- 「AWS Toolkit for Visual Studio」のセットアップ。

- 「.NET Core 3.1 またはそれ以降」(C# AWS CDK アプリケーションには必須)
- 「Amazon.Lambda.Tools」がインストールされました。

制限事項

- AWS CDK は「AWS CloudFormation」を使用しているため、AWS CDK アプリケーションには CloudFormation サービスクォータが適用されます。詳細については、「AWS CloudFormation のクォータ」を参照してください。
- テナントの CloudFormation スタックは CloudFormation サービスロール `infra-cloudformation-role` を使用して作成され、アクション (`sns*` と `sqs*`) にはワイルドカード文字が使用されますが、リソースは `tenant-cluster` プレフィックスにロックされます。実稼働環境での使用では、この設定を評価し、このサービスロールへの必要なアクセス権のみを提供してください。また、`InfrastructureProvision` Lambda 関数はワイルドカード文字 (`cloudformation*`) を使用して CloudFormation スタックをプロビジョニングしますが、リソースは `tenant-cluster` プレフィックスにロックダウンされます。
- このサンプルコードの docker ビルドは、`linux/amd64` ベースイメージを強制するために `--platform=linux/amd64` に使用されています。これは、最終的なイメージアーティファクトが、デフォルトで x86-64 アーキテクチャを使用する Lambda に適したものになるようにするためです。ターゲット Lambda アーキテクチャを変更する必要がある場合は、必ず Dockerfiles と AWS CDK コードの両方を変更してください。詳細については、ブログ記事「AWS Lambda 関数を ARM ベースの AWS Graviton2 プロセッサーに移行する」を参照してください。
- スタックの削除プロセスでは、スタックによって生成された CloudWatch Logs (ロググループとログ) はクリーンアップされません。AWS マネジメントコンソール、Amazon CloudWatch コンソール、または API を使用して、ログを手動でクリーンアップする必要があります。

このパターンは例として設定されています。本番環境で使用する場合は、以下の設定を評価し、ビジネス要件に基づいて変更を加えます。

- この例では、「AWS Simple Storage Service (Amazon S3)」バケットでは、わかりやすくするためにバージョニングが有効になっていません。セットアップを評価し、必要に応じて更新してください。
- この例では、簡単にするために、認証、認可、スロットリングなしで Amazon API Gateway REST API エンドポイントを設定します。本番環境での使用には、システムをビジネスセキュリティインフラストラクチャと統合することをお勧めします。この設定を評価し、必要に応じて必要なセキュリティ設定を追加してください。

- このテナントインフラストラクチャの例では、「Amazon Simple Notification Service (Amazon SNS)」と「Amazon Simple Queue Service (Amazon SQS)」のセットアップは最小限です。各テナントの「AWS Key Management Service (AWS KMS)」は、「AWS KMS キーポリシー」に基づいて使用するアカウントの「Amazon CloudWatch」サービスと Amazon SNS サービスが利用できるように開放されます。セットアップは単なるプレースホルダーです。ビジネスユースケースに基づいて、必要に応じて設定を調整してください。
- API エンドポイントと AWS CloudFormation を使用したバックエンドテナントのプロビジョニングと削除を含むがこれらに限定されないセットアップ全体は、基本的なハッピーパスの場合のみを対象としています。ビジネスニーズに基づいて、必要な再試行ロジック、追加のエラー処理ロジック、セキュリティロジックを使用してセットアップを評価し、更新してください。
- このサンプルコードは、この記事の執筆時点でポリシーを確認するために最新の「cdk-nag」でテストされています。将来、新しいポリシーが施行される可能性があります。これらの新しいポリシーでは、スタックをデプロイする前に、推奨事項に基づいてスタックを手動で変更する必要がある場合があります。既存のコードを見直して、ビジネス要件に合っていることを確認してください。
- このコードは、作成されるほとんどのリソースに静的に割り当てられた物理名に頼るのではなく、AWS CDK を使用してランダムなサフィックスを生成します。この設定は、これらのリソースが一意であり、他のスタックと競合しないようにするためです。詳細については、「AWS CDK ドキュメント」を参照してください。ビジネス要件に基づいて調整してください。
- このサンプルコードは、.NET Lambda アーティファクトを Docker ベースのイメージにパッケージ化し、Lambda が提供する「コンテナイメージランタイム」で実行されます。コンテナイメージランタイムには、標準的な転送および保存メカニズム (コンテナレジストリ) と、より正確なローカルテスト環境 (コンテナイメージを使用) という利点があります。「Lambda が提供する .NET ランタイム」を使用するようにプロジェクトを切り替えて Docker イメージのビルド時間を短縮することもできますが、その場合は転送メカニズムと保存メカニズムを設定し、ローカルセットアップが Lambda セットアップと一致することを確認する必要があります。ユーザーのビジネス要件に合わせてコードを調整してください。

製品バージョン

- AWS CDK バージョン 2.45.0 またはそれ以降
- Visual Studio 2022

アーキテクチャ

テクノロジースタック

- Amazon API Gateway
- AWS CloudFormation
- Amazon CloudWatch
- Amazon DynamoDB
- AWS Identity and Access Management (IAM)
- AWS KMS
- AWS Lambda
- Amazon S3
- Amazon SNS
- Amazon SQS

アーキテクチャ

次の図に、テナントスタックの作成フローを示します。コントロールプレーンとテナントテクノロジースタックの詳細については、「追加情報」セクションを参照してください。

テナントスタックの作成フロー

1. ユーザーは、新しいテナントペイロード (テナント名、テナントの説明) を含む POST API リクエストを JSON 形式で Amazon API Gateway がホストする REST API に送信します。API Gateway はリクエストを処理し、バックエンドの Lambda テナントオンボーディング機能に転送します。この例では、認可も認証もありません。実稼働環境では、この API を SaaS インフラストラクチャセキュリティシステムと統合する必要があります。
2. テナントオンボーディング機能がリクエストを検証します。次に、テナント名、生成されたテナントユニバーサルユニーク識別子 (UUID)、テナントの説明を含むテナントレコードを Amazon DynamoDB テナントオンボーディングテーブルに保存しようとします。
3. DynamoDB がレコードを保存すると、DynamoDB ストリームはダウンストリームの Lambda テナントインフラストラクチャ機能を開始します。
4. テナントインフラストラクチャ Lambda 関数は、受信した DynamoDB ストリームに基づいて動作します。ストリームが INSERT イベント用である場合、関数はストリームの NewImage セ

クション (最新の更新レコード、テナント名フィールド) を使用して CloudFormation を呼び出し、S3 バケットに保存されているテンプレートを使用して新しいテナントインフラストラクチャを作成します。CloudFormation テンプレートには、テナント名パラメーターが必要です。

5. AWS CloudFormation は、CloudFormation テンプレートと入力パラメータに基づいてテナントインフラストラクチャを作成します。
6. 各テナントインフラストラクチャ設定には、CloudWatch アラーム、請求アラーム、およびアラームイベントがあります。
7. アラームイベントは SNS トピックへのメッセージになり、テナントの AWS KMS キーによって暗号化されます。
8. SNS トピックは、受信したアラームメッセージを SQS キューに転送し、その SQS キューは、テナントの AWS KMS によって暗号化キーとして暗号化されます。

他のシステムを Amazon SQS と統合して、キュー内のメッセージに基づいてアクションを実行できます。この例では、コードを汎用的に保つため、受信メッセージはキューに残り、手動で削除する必要があります。

テナントスタックの削除フロー

1. ユーザーは、新しいテナントペイロード (テナント名、テナントの説明) を含む DELETE API リクエストを JSON 形式で Amazon API Gateway がホストする REST API に送信します。このリクエストは Amazon API Gateway がホストする REST API で処理され、テナントオンボーディング機能に転送されます。この例では、認可も認証もありません。実稼働環境では、この API は SaaS インフラストラクチャセキュリティシステムと統合されます。
2. テナントオンボーディング機能はリクエストを確認し、テナントオンボーディングテーブルからテナントレコード (テナント名) を削除しようとします。
3. DynamoDB がレコードを正常に削除すると (レコードはテーブルに存在し、削除された)、DynamoDB ストリームはダウンストリーム Lambda テナントインフラストラクチャ関数を開始します。
4. テナントインフラストラクチャ Lambda 関数は、受信した DynamoDB ストリームレコードに基づいて動作します。ストリームが REMOVE イベント用である場合、関数はレコードの OldImage セクション (最新の変更 (削除) の前のレコード情報と Tenant Name フィールド) を使用して、そのレコード情報に基づいて既存のスタックの削除を開始します。
5. AWS CloudFormation は、入力に従ってターゲットテナントスタックを削除します。

ツール

AWS サービス

- 「Amazon API Gateway」は、任意のスケールで REST、HTTP、WebSocket API を作成、公開、維持、監視、保護する上で役立ちます。
- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK Toolkit は、AWS Cloud Development Kit (AWS CDK) アプリケーションの操作に役立つコマンドラインクラウド開発キットです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- 「Amazon Simple Queue Service (Amazon SQS)」は、分散したソフトウェアシステムとコンポーネントの統合と分離を支援し、安全で耐久性があり、利用可能なホスト型キューを提供します。
- 「AWS Toolkit for Visual Studio」は Visual Studio 統合開発環境 (IDE) 用のプラグインです。Toolkit for Visual Studio は、AWS サービスを使用する.NET アプリケーションの開発、デバッグ、およびデプロイをサポートします。

その他のツール

- 「Visual Studio」は、コンパイラー、コード補完ツール、グラフィカルデザイナー、およびソフトウェア開発をサポートするその他の機能を備えた IDE です。

コード

このパターンのコードは、「SaaS Architecture for Silo Model APG 例のテナントオンボーディング」リポジトリにあります。

エピック

AWS CDK のセットアップ

タスク	説明	必要なスキル
Node.js がインストールされていることを確認してください。	Node.js がローカルマシンにインストールされていることを確認するには、次のコマンドを実行します。 `node --version`	AWS 管理者、AWS DevOps
AWS CDK Toolkit をインストールします。	AWS CDK Toolkit をローカルマシンにインストールするには、次のコマンドを実行します。 `npm install -g aws-cdk` npm がインストールされていない場合は、「Node.js サイト」からインストールできます。	AWS 管理者、AWS DevOps
AWS CDK ツールキットのバージョンを確認します。	AWS CDK Toolkit のバージョンがマシンに正しくインストールされていることを確認す	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	るには、次のコマンドを実行します。 `cdk --version`	

テナントオンボーディング (コントロールプレーン) のコードを確認してください。

タスク	説明	必要なスキル
リポジトリをクローン作成します。	「リポジトリ」をクローンし、`\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example` フォルダに移動します。 Visual Studio 2022 で `\src\TenantOnboardingInfra.sln` ソリューションファイルを開きます。`TenantOnboardingInfraStack.cs` ファイルを開き、コードを確認します。 このスタックの一部として、以下のリソースが作成されます。 • DynamoDB テーブル • S3 バケット (CloudFormation テンプレートを S3 バケットにアップロードします。) • Lambda 実行ロール	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	• Lambda function • API ゲートウェイ API • Lambda 関数へのイベントソース	
CloudFormation のテンプレートを確認してください。	`\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\template` フォルダーで、`infra.yaml` を開き、CloudFormation テンプレートを確認します。このテンプレートは、テナントオンボーディング DynamoDB テーブルから取得したテナント名でハイドレイトされます。 このテンプレートはテナント固有のインフラストラクチャをプロビジョニングします。この例では、AWS KMS キー、Amazon SNS、Amazon SQS、および CloudWatch アラームのプロビジョニングを行います。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
テナントオンボーディング機能を確認してください。	`Function.cs` を開き、テナントオンボーディング関数のコードを確認してください。この関数は Visual Studio AWS Lambda Project (.NET Core-C#) テンプレートと.NET 6 (コンテナイメージ) ブループリントを使用して作成されています。 `Dockerfile` を開き、コードを確認します。`Dockerfile` は、Lambda コンテナイメージを構築するための手順を含むテキストファイルです。 次の NuGet パッケージが依存関係として `TenantOnboardingFunction` プロジェクトに追加されることに注意してください。 • `Amazon.Lambda.APIGatewayEvents` • `AWSSDK.DynamoDBv2` • `Newtonsoft.Json`	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
テナントのインフラストラクチャプロビジョニング機能を確認してください。	`\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\InfraProvisioningFunction` に移動します。 `Function.cs` を開き、テナントインフラストラクチャプロビジョニング関数のコードを確認してください。この関数は Visual Studio AWS Lambda Project (.NET Core-C#) テンプレートと.NET 6 (コンテナイメージ) ブループリントを使用して作成されています。 `Dockerfile` を開き、コードを確認します。 次の NuGet パッケージが依存関係として `InfraProvisioningFunction` プロジェクトに追加されることに注意してください。 • `Amazon.Lambda.DynamoDBEvents` • `AWSSDK.DynamoDBv2` • `AWSSDK.Cloudformation`	アプリ開発者、AWS DevOps

AWS リソースをデプロイする

タスク	説明	必要なスキル
ソリューションをビルドします。	ソリューションを構築するには、以下のステップを実行します。 1. Visual Studio 2022 で `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\TenantOnboardingInfra.sln` ソリューションファイルを開きます。 2. ソリューションのコンテキスト (右クリック) メニューを開き、[ソリューションの構築] を選択します。 **Note** ソリューションを構築する前に、`Amazon.CDK.Lib` NuGet パッケージを`\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\TenantOnboardingInfra` プロジェクトの最新バー	アプリ開発者

タスク	説明	必要なスキル
	ジョンに更新してください。	

AWS CDK 環境をブートストラップします。	Windows コマンドプロンプトを開き、cdk.json ファイルがある AWS CDK アプリケーションのルートフォルダに移動します (\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example)。ブートストラップの場合は、次のコマンドを実行します。
	`cdk bootstrap`
	認証情報用の AWS プロファイルを作成した場合は、プロファイルでコマンドを使用してください。
	`cdk bootstrap --profile <profile name>`
		AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

AWS CDK スタックを一覧表示します。	このプロジェクトの一部として作成されるスタックをすべて一覧表示するには、次のコマンドを実行します。
	`cdk ls`
	`cdk ls --profile <profile name>`
	認証情報用の AWS プロファイルを作成した場合は、プロファイルでコマンドを使用してください。
	`cdk ls --profile <profile name>`
		AWS 管理者、AWS DevOps

どの AWS リソースが作成されるかを確認してください。	このプロジェクトの一部として作成されるすべての AWS リソースを確認するには、以下のコマンドを実行します。
	`cdk diff`
	認証情報用の AWS プロファイルを作成した場合は、プロファイルでコマンドを使用してください。
	`cdk diff --profile <profile name>`
		AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

AWS CDK を使用してすべての AWS リソースをデプロイします。	すべての AWS リソースをデプロイするには、次のコマンドを実行します。
	`cdk deploy --all --require-approval never`
	認証情報用の AWS プロファイルを作成した場合は、プロファイルでコマンドを使用してください。
	`cdk deploy --all --require-approval never --profile <profile name>`
	デプロイが完了したら、次の例に示すように、コマンドプロンプトの出力セクションから API URL をコピーします。
	`Outputs: TenantOnboardingInfraStack.TenantOnboardingAPIEndpoint42E526D7 = https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/`
		AWS 管理者、AWS DevOps

機能性の検証

タスク	説明	必要なスキル

新しいテナントを作成する。	新しいテナントを作成するには、次の curl リクエストを送信します。
	
	`curl -X POST <TenantOnboardingAPIEndpoint* from CDK Output>tenant -d '{"Name":"Tenant123", "Description":"Stack for Tenant123"}'`
	
	次の例に示すように、プレースホルダー `<TenantOnboardingAPIEndpoint* from CDK Output>` を AWS CDK の実際の値に変更します。
	
	`curl -X POST https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/tenant -d '{"Name": "Tenant123", "Description":"test12"}'`
	
	以下の例は出力を示しています。
	
	`{"message": "A new tenant added - 5/4/2022 7:11:30 AM"}`
		アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
DynamoDB で新しく作成されたテナントの詳細を確認します。	DynamoDB で新しく作成されたテナントの詳細を確認するには、次のステップを実行します。 1. AWS マネジメントコンソールを開き、Amazon DynamoDB サービスに移動します。 2. 左側のナビゲーションで [項目の探索] を選択し、`TenantOnboarding` テーブルを選択します。 **Note** テナント名には が付加されますtenantcluster- 。詳細については、「追加情報」セクションをご覧ください。 3. テナントの詳細を含む新しいアイテムが作成されていることを確認します。	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
新しいテナントのスタックの作成を確認します。	新しいスタックが正常に作成され、CloudFormation テンプレートに従って新しく作成されたテナントのインフラストラクチャでプロビジョニングされたことを確認します。 1. CloudFormation コンソールを開きます。 2. 左側のナビゲーションで [スタック] を選択し、テナント名のスタックが正常に作成されたことを確認します。 3. 新しく作成したテナントスタックを選択し、[リソース] タブを選択します。アラームリソースと Amazon SQS リソースをメモします。 4. AWS 認証情報を設定して新しいターミナルを開き、正しいリージョンを指定します。テストアラームを発生させるには、次のコードを入力し、`<alarm resource name>` をステップ 3 で書き留めたアラームリソース名に置き換えます。 `aws cloudwatch set-alarm-state --alarm-name <alarm resource name> --state-value`	アプリ開発者、AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	ALARM --state-reason 'Test setup' 次の例は、アラームリソース名を含むコードを示します。 aws cloudwatch set-alarm-state --alarm-name tenantcluster-tenant123-alarm --state-value ALARM --state-reason 'Test setup' 5. コンソールを開き、Amazon SQS コンソールに移動します。ステップ 3 で特定した Amazon SQS リソース名を選択します。「AWS ドキュメントの指示」に従って、ステップ 4 で発生したアラームからテストメッセージを受信して削除します。	

タスク	説明	必要なスキル

テナントスタックを削除します。	テナントスタックを削除するには、次の curl リクエストを送信します。
	
	`curl -X DELETE <TenantOnboardingAPIEndpoint* from CDK Output>tenant/<Tenant Name from previous step>`
	
	次の例のように、プレースホルダー `<TenantOnboardingAPIEndpoint* from CDK Output>` を AWS CDK の実際の値に変更し、`<Tenant Name from previous step>` を前のテナント作成ステップの実際の値に変更します。
	
	`curl -X DELETE https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/tenant/Tenant123`
	
	以下の例は出力を示しています。
	
	`{"message": "Tenant destroyed - 5/4/2022 7:14:48 AM"}`
		アプリ開発者、AWS DevOps、AWS 管理者

タスク	説明	必要なスキル
既存のテナントのスタックの削除を確認します。	既存のテナントスタックが削除されたことを確認するには、次のステップを実行します。 1. コンソールを開き、CloudFormation コンソールに移動します。 2. 左側のナビゲーションで、テナント名の既存のスタックがコンソールに存在しない (CloudFormation コンソールが [アクティブ] スタックのみを表示するように設定されている場合) か、削除中であることを確認します。スタックが CloudFormation コンソールにない場合は、ドロップダウンリストを使用してコンソールの設定を [アクティブ] から [削除] に変更し、削除されたスタックを確認し、スタックが正常に削除されたことを確認します。	アプリ開発者、AWS 管理者、AWS DevOps

クリーンアップ

タスク	説明	必要なスキル
環境を破壊する。	スタックをクリーンアップする前に、次の点を確認してください。	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
	• DynamoDB のすべてのレコードは、前回のテナント削除操作、または DynamoDB コンソールまたは API によって削除されます。テナントレコードを削除するたびに、対応する AWS CloudFormation のクリーンアップが開始されます。 • テナントベースの AWS CloudFormation スタックはすべて、(DynamoDB トリガーのクリーンアップロジックが失敗した場合に備えて) AWS CloudFormation コンソールでクリーンアップされます。 テストが完了したら、次のコマンドを実行することで、AWS CDK を使用してすべてのスタックと関連リソースを破棄できます。 `cdk destroy --all;` 認証情報用の AWS プロファイルを作成した場合は、そのプロファイルを使用してください。	

タスク	説明	必要なスキル
	スタックの削除プロンプトを確認して、スタックを削除します。	
Amazon CloudWatch Logs をクリーンアップします。	スタックの削除プロセスでは、スタックによって生成された CloudWatch Logs (ロググループとログ) はクリーンアップされません。CloudWatch コンソールまたは API を使用して、CloudWatch リソースを手動でクリーンアップします。	アプリ開発者、AWS DevOps、AWS 管理者

関連リソース

- 「AWS CDK .NET ワークショップ」
- 「C# で AWS CDK を操作する」
- 「CDK .NET リファレンス」

追加情報

コントロールプレーンテクノロジースタック

.NET で記述された CDK コードは、以下のリソースで構成されるコントロールプレーンインフラストラクチャのプロビジョニングに使用されます。

1. API Gateway

   コントロールプレーンスタックの REST API エントリポイントとして機能します。
2. テナントオンボーディング Lambda 関数

   この Lambda 関数は、m メソッドを使用して API Gateway によって開始されます。

   POST メソッドの API リクエストにより、(`tenant name`、`tenant description`) が DynamoDB `Tenant Onboarding` テーブルに挿入されます。

このコード例では、テナント名はテナントスタック名とそのスタック内のリソース名の一部としても使用されています。これは、これらのリソースを識別しやすくするためです。このテナント名は、競合やエラーを避けるため、セットアップで一意である必要があります。入力検証の詳細な設定については、「IAM ロールのドキュメント」と [制限事項] セクションで説明されています。

DynamoDB テーブルへの永続化プロセスは、テナント名がテーブル内の他のレコードで使用されていない場合にのみ成功します。

`PutItem` 条件式として使用できるのはパーティションキーだけなので、この場合のテナント名がこのテーブルのパーティションキーになります。

テナント名が以前に記録されたことがなければ、レコードはテーブルに正常に保存されます。

ただし、テナント名がテーブル内の既存のレコードですでに使用されている場合、操作は失敗し、DynamoDB `ConditionalCheckFailedException` 例外が開始されます。この例外は、テナント名が既に存在することを示す失敗メッセージ (`HTTP BadRequest`) を返すために使用されます。

`DELETE` メソッド API リクエストは、`Tenant Onboardin g` テーブルから特定のテナント名のレコードを削除します。

この例の DynamoDB レコードの削除は、レコードが存在しなくても成功します。

ターゲットレコードが存在して削除されると、DynamoDB ストリームレコードが作成されます。それ以外の場合は、ダウンストリームレコードは作成されません。

3. Amazon DynamoDB Streams を有効にしたテナントによる DynamoDB のオンボーディング

   これによりテナントのメタデータ情報が記録され、レコードを保存または削除すると、ストリームが下流の `Tenant Infrastructure` Lambda 関数に送信されます。

4. テナントインフラストラクチャ Lambda 関数

   この Lambda 関数は、前のステップの DynamoDB ストリームレコードによって開始されます。レコードが `INSERT` イベントの場合は、AWS CloudFormation を呼び出して、S3 バケットに保存されている CloudFormation テンプレートを使用して新しいテナントインフラストラクチャを作成します。レコードが `REMOVE` の場合、ストリームレコードの `Tenant Name` フィールドに基づいて既存のスタックの削除を開始します。

5. S3 バケット

これは CloudFormation テンプレートを保存するためのものです。

6. 各 Lambda 関数の IAM ロールと CloudFormation のサービスロール

   各 Lambda 関数には、そのタスクを実行するための「最小特権」アクセス許可を持つ固有の IAM ロールがあります。たとえば、`Tenant On-boarding` Lambda 関数には DynamoDB への読み取り/書き込みアクセス権があり、`Tenant Infrastructure` Lambda 関数は DynamoDB ストリームのみを読み取ることができます。

   テナントスタックのプロビジョニング用にカスタム CloudFormation サービスロールが作成されます。このサービスロールには、CloudFormation スタックプロビジョニングの追加権限 (AWS KMS キーなど) が含まれています。これにより、ロールが Lambda と CloudFormation に分割され、1 つのロール (インフラストラクチャ Lambda ロール) にすべてのアクセス許可が割り当てられることがなくなります。

   強力なアクション (CloudFormation スタックの作成や削除など) を許可する権限はロックされ、`tenantcluster-` で始まるリソースにのみ許可されます。リソースの命名規則のため、例外は AWS KMS です。API から取り込まれたテナント名は、他の検証チェックと一緒に `tenantcluster-` で付加されます (ハイフン付きの英数字のみ。ほとんどの AWS リソース命名に当てはまるように 30 文字未満に制限されています)。これにより、テナント名によってコアインフラストラクチャスタックやリソースが誤って中断されることがなくなります。

テナントテクノロジースタック

CloudFormation のテンプレートは S3 バケットに保存されます。テンプレートは、テナント固有の AWS KMS キー、CloudWatch アラーム、SNS トピック、SQS キュー、および「SQS ポリシー」をプロビジョニングします。

AWS KMS キーは、Amazon SNS と Amazon SQS によるメッセージのデータ暗号化に使用されます。「AWSSolutions-SNS2 と AWSSolutions-SQS2」のセキュリティプラクティスでは、Amazon SNS と Amazon SQS を暗号化して設定することを推奨しています。ただし、AWS マネージドキーを使用する場合、CloudWatch アラームは Amazon SNS では機能しないため、この場合はカスタマー管理のキーを使用する必要があります。詳細については、「AWS ナレッジセンター」を参照してください。

SQS ポリシーは Amazon SQS キューで使用され、作成された SNS トピックがメッセージをキューに配信できるようにします。SQS ポリシーがないと、アクセスは拒否されます。詳細については、「Amazon SNS ドキュメント」を参照してください。

CQRS とイベントソーシングを使用してモノリスをマイクロサービスに分解する

ロドルフォ ジュニアによって作成されました。 セラダ(AWS)、ドミトリー グリン(AWS)、タビー ウォード(AWS)

概要

このパターンは、コマンドクエリ責任分離 (CQRS) パターンとイベントソーシングパターンの両方を使用する 2 つのパターンを組み合わせたものです。CQRS パターンは、コマンドモデルとクエリモデルの責任を分離します。イベントソーシングパターンは、非同期のイベント駆動型通信を利用して、全体的なユーザーエクスペリエンスを向上させます。

CQRS とAmazon Web Services(AWS)サービスを使用して、モノリスアプリケーションをマイクロサービスアーキテクチャにリファクタリングしながら、各データモデルを個別に維持およびスケーリングできます。その後、イベントソーシングパターンを使用して、コマンドデータベースのデータをクエリデータベースに同期できます。

このパターンでは、最新バージョンの Visual Studio を使用して開くことができるソリューション (*.sln) ファイルを含むサンプルコードを使用しています。この例には、AWS のサーバーレスアプリケーション、従来型アプリケーション、オンプレミスアプリケーションで CQRS とイベントソーシングがどのように機能するかを紹介する Reward API コードが含まれています。

CQRS とイベントソーシングの詳細については、追加情報 セクションを参照してください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon CloudWatch
- Amazon DynamoDB テーブル
- Amazon DynamoDB Streams
- AWS Identity and Access Management (IAM) アクセスキーとシークレットキー。詳細については、関連リソースセクションのビデオを参照してください。
- AWS Lambda
- Visual Studio に精通していること

- AWS Toolkit for Visual Studio に精通していること。詳細については、関連リソースセクションのAWS Toolkit for Visual Studio のデモビデオを参照してください。

製品バージョン

- Visual Studio 2019 コミュニティエディション。
- AWS Toolkit for Visual Studio。
- .NET Core 3.1 このコンポーネントは Visual Studio インストレーションのオプションです。インストール中に.NET Core を含めるには、NET Core クロスプラットフォーム開発を選択します。

制約事項

- 従来のオンプレミスアプリケーション (ASP.NET Core Web API とデータアクセスオブジェクト) のサンプルコードにはデータベースは付属していません。ただし、このコードにはモックデータベースとして機能する `CustomerData` インメモリオブジェクトが付属しています。用意されているコードはパターンをテストするのに十分です。

アーキテクチャ

ソーステクノロジースタック

- ASP.NET Core Web API プロジェクト
- IIS Web サーバー
- データアクセスオブジェクト
- CRUD モデル

ソースアーキテクチャ

ソースアーキテクチャでは、CRUD モデルには 1 つのアプリケーションにコマンドインターフェースとクエリインターフェースの両方が含まれています。コード例については、`CustomerDAO.cs` (添付) を参照してください。

ターゲット テクノロジー スタック

- Amazon DynamoDB

- Amazon DynamoDB Streams
- AWS Lambda
- (オプション) Amazon API Gateway API ゲートウェイ
- Amazon Simple Notification Service (Amazon SNS)

ターゲット アーキテクチャ

ターゲットアーキテクチャでは、コマンドインターフェイスとクエリインターフェイスは分離されています。次の図に示されているアーキテクチャは、API ゲートウェイと Amazon SNS を使用して拡張できます。詳細については、追加情報 セクションを参照してください。

1. コマンド Lambda 関数は、データベースに対して作成、更新、削除などの書き込み操作を実行します。
2. クエリ Lambda 関数は、データベースに対して get や select などの読み取り操作を実行します。
3. この Lambda 関数は、Command データベースからの DynamoDB ストリームを処理し、変更があった場合はクエリデータベースを更新します。

ツール

ツール

- Amazon DynamoDB – Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。
- Amazon DynamoDB Streams – Amazon DynamoDB Streams は、Amazon DynamoDB テーブル内の項目レベルでの変更を時系列順にキャプチャする のサービスです。このサービスは、この情報を最大 24 時間ログに保存します。保管時の暗号化では、DynamoDB Streams のデータが暗号化されます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- AWS Management Console – AWS マネジメントコンソールは、AWS のサービスを管理するための広範なサービスコンソールコレクションへのアクセスを提供するウェブアプリケーションです。

- Visual Studio 2019 コミュニティエディション — Visual Studio 2019 は統合開発環境 (IDE) です。コミュニティエディションは、オープンソースのコントリビューターには無料で提供されます。このパターンでは、Visual Studio 2019 コミュニティエディションを使用してサンプルコードを開き、コンパイル、実行します。表示のみの場合は、任意のテキストエディターまたは Visual Studio Code を使用できます。
- AWS Toolkit for Visual Studio – AWS Toolkit for Visual Studio は Visual StudioIDE へのプラグインです。AWS Toolkit for Visual Studio を使用すると、AWS サービスを使用する .NET アプリケーションの開発、デバッグ、および展開が容易になります。

コード

サンプルコードは添付されています。サンプルコードをデプロイする手順については、エピックセクションを参照してください。

エピック

ソリューションのオープンと構築

タスク	説明	必要なスキル
セクションを開きます。	1. 添付ファイルセクションからサンプルソースコード (CQRS-ES Code.zip) をダウンロードし、ファイルを抽出します。 2. Visual Studio IDE で、ファイル、開く、プロジェクトソリューションを選択し、ソースコードを抽出したフォルダーに移動します。 3. AWS.APG.CQRSES.sln を選択し、開くを選択します：ソリューション全体が Visual Studio に読み込まれます。	アプリ開発者

タスク	説明	必要なスキル
ソリューションを構築します。	ソリューションのコンテキスト (右クリック) メニューを開き、ソリューションをビルドする を選択します。これにより、ソリューション内のすべてのプロジェクトがビルドされ、コンパイルされます。正常にコンパイルされるはずです。 Visual Studio ソリューションエクスプローラーにはディレクトリ構造が表示されるはずです。 • `CQRS On-Premises Code Sample` CQRS をオンプレミスで使用する例が含まれています。 • `CQRS AWS Serverless` には、AWS サーバーレスサービスを使用するすべての CQRS とイベントソーシングのサンプルコードが含まれています。	アプリ開発者

DynamoDB テーブルを構築する

タスク	説明	必要なスキル
認証情報の提供	アクセスキーをまだお持ちでない場合は、関連リソースセクションの動画をご覧ください。	アプリケーション開発者、データエンジニア、DBA

タスク	説明	必要なスキル
	1. ソリューションエクスプローラーで、CQRS AWS サーバーレスを展開し、ビルドソリューションフォルダーを展開します。 2. AwS.APG.CQRSES.Build プロジェクトを展開し、`Program.cs` ファイルを表示します。 3. `Program.cs` の一番上までスクロールして `Program()` を探してください。 4. アカウントアクセスキーに `YOUR ACCESS KEY` を置き換え、アカウントのシークレットキーに `YOUR SECRET KEY` を置き換えてください。実稼働環境では、キーをハードコーディングしないことに注意してください。代わりに、AWS Secrets Manager を使用して認証情報を保存および取得できます。	
プロジェクトをビルドします。	プロジェクトをビルドするには、AwS.APG.CQRSES.Build プロジェクトのコンテキスト (右クリック) メニューを開き、 ビルドを選択します。	アプリケーション開発者、データエンジニア、DBA

タスク	説明	必要なスキル
テーブルを作成してデータを入力します。	テーブルを作成してシードデータを入力するには、AwS.APG.CQRSES.Build プロジェクトのコンテキスト (右クリック) メニューを開き、 デバッグ、新規インスタンスの開始を選択します。	アプリケーション開発者、データエンジニア、DBA
テーブルコンストラクトとデータを検証します。	確認するには、AWS Explorer, ￥に移動し、Amazon DynamoDB を展開します。テーブルが表示されるはずです。各テーブルを開いて、サンプルデータを表示します。	アプリケーション開発者、データエンジニア、DBA

ローカルテストを実行する

タスク	説明	必要なスキル
Go プロジェクトを構築します。	1. ソリューションを開き、CQRS AWS Services/CQRS/Tests ソリューションフォルダに移動します。 2. AWS.APG.CQRSES.CQRSLambda.Tests プロジェクトで BaseFunctionTest.cs を開き、AccessKey と SecretKey を、作成した IAM キーに置き換えます。 3. 変更を保存します。 4. プロジェクトをビルドするには、aws.apg.CQRSES.Build プロジェクトのコンテキスト (右クリック) メ	アプリ開発者、テストエンジニア

タスク	説明	必要なスキル
	ニューを開き、ビルドを選択します。	
イベントソーシングプロジェクトを構築います。	1. CQRS AWS Services/EventSource/Testsソリューションフォルダに移動します。 2. AWS.APG.CQRSES.EventSourceLambda.Tests プロジェクトで BaseFunctionTest.cs を開き、AccessKey と SecretKey を、作成した IAM キーに置き換えます。 3. 変更を保存します。 4. プロジェクトをビルドするには、aws.apg.CQRSES.Build プロジェクトのコンテキスト (右クリック) メニューを開き、ビルドを選択します。	アプリ開発者、テストエンジニア
テストの実行	すべてのテストを実行するには、表示、テストエクスプローラー、すべてのテストを表示で実行の順に選択します。すべてのテストに合格するはずです。合格すると緑色のチェックマークアイコンが表示されます。	アプリ開発者、テストエンジニア

CQRS Lambda 関数を AWS に公開する

タスク	説明	必要なスキル
Lambda 関数を発行します。	1. Solution Explorer で、AWS.APG.CQRSES.CommandCreateLambdプロジェクトのコンテキスト(右クリック)メニューを開き、AWS Lambdaへの発行先 を選択します。 2. 使用するプロファイル、Lambda 関数をデプロイする AWS リージョン、および関数名を選択します。 3. 残りのフィールドはデフォルトのままにして、次へ を選択します。 4. ロール名ドロップダウンリストで AWSLambda FullAccessを選択します。 5. アカウントキーを指定するには、追加を選択し、`AcessKey` 変数として入力し、アクセスキーを値として入力します。次に、もう一度追加を選択し、`SecretKey` 変数として入力し、値としてシークレットキーを入力します。 6. 残りのフィールドはデフォルトのままにして、保存を選択します。Lambda テスト関数がアップロードされると、その関数は Visual	アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル
	Studio に自動的に表示されます。 7. 次のプロジェクトに対して、ステップ 1～6 を繰り返します。 • aws.apg.cqrses.commandDeleteLambda • aws.apg.cqrses.commandUpdate Lambda • aws.apg.cqrses.CommandAddRewardLambda • aws.apg.cqrses.CommandRewardLambda • aws.apg.cqrses.queryCustomerListLambda • aws.apg.cqrses.queryrewardLambda	
関数のアップロードを検証します。	(オプション) 関数が正常にロードされたことを確認するには、AWS Explorer に移動して AWS Lambda を展開します。テストウィンドウを開くには、Lambda 関数を選択します (ダブルクリック)。	アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル
Lambda 関数をテストします。	1. 追加情報 セクションのリクエストデータを入力するか、テストデータからサンプルリクエストデータをコピーします。テストする機能のデータを必ず選択してください。 2. テストを実行するには、呼び出し を選択します。レスポンスとエラーはレスポンステキストボックスに表示され、ログは ログテキストボックスまたは [CloudWatch Logs] に表示されます。 3. データを確認するには、AWS Explorerで DynamoDB テーブルを選択します (ダブルクリック)。 すべての CQRS Lambda プロジェクトは、`CQRS AWS Serverless\CQRS \Command Microserv ice` および `CQRS AWS Serverless\CQRS\Co mmand Microservice` ソリューションフォルダーにあります。ソリューションディレクトリとプロジェクトについては、追加情報 セクションのソースコードディレクトリを参照してください。	アプリ開発者、DevOps エンジニア

タスク	説明	必要なスキル
残りの関数を公開します。	次のプロジェクトに対して、前のステップを繰り返します。 • aws.apg.cqrses.commandDeleteLambda • aws.apg.cqrses.commandUpdate Lambda • aws.apg.cqrses.CommandAddRewardLambda • aws.apg.cqrses.CommandRewardLambda • aws.apg.cqrses.queryCustomerListLambda • aws.apg.cqrses.queryrewardLambda	アプリ開発者、DevOps エンジニア

イベントリスナーとしての Lambda 関数の設定

タスク	説明	必要なスキル
顧客 Lambda イベントハンドラーとリワード Lambda イベントハンドラーを公開します。	各イベントハンドラーを公開するには、前のエピックの手順に従います。 プロジェクトは `CQRS AWS Serverless\Event Source\Customer Event` および `CQRS AWS Serverless\Event Source\Reward Event` ソリューションフォルダーにあります。詳細については、追加情報セクションのソース	アプリ開発者

タスク	説明	必要なスキル
	コードディレクトリを参照してください。	

タスク	説明	必要なスキル
イベントソーシング Lambda イベントリスナーをアタッチします。	1. Lambda プロジェクトを公開するときに使用したのと同じアカウントを使用して AWS マネジメントコンソールにログインします。 2. リージョンには、米国東部 1 または前のエピックで Lambda 関数をデプロイしたリージョンを選択します。 3. Lambda サービスに移動します。 4. `EventSourceCustomer` Lambda 関数を選択する 5. トリガを追加を選択します。 6. トリガー設定 ドロップダウンリストで、DynamoDB を選択します。 7. DynamoDB table テーブルのドロップダウンリストで cqrses-customer-cmd を選択します。 8. 開始位置ドロップダウンリストで水平線のトリム元を選択します トリムホライズンとは、DynamoDB トリガーが最後の (トリムされていない) ストリームレコード、つまりシャード内で最も古いレコードから読み取りを開始することを意味します。	アプリ開発者

タスク	説明	必要なスキル
	9. 有効 チェックボックスを選択します。 10残りのフィールドはデフォルトのままにして、追加を選択します。 リスナーが DynamoDB テーブルに正常にアタッチされると、Lambda デザイナーページに表示されます。	
EventSourceReward Lambda 関数をパブリッシュしてアタッチします。	`EventSourceReward` Lambda 関数をパブリッシュしてアタッチするには、DynamoDB テーブル のドロップダウンリストから cqrses-reward-cmd を選択して、前の 2 つのストーリーの手順を繰り返します。	アプリ開発者

DynamoDB ストリームと Lambda トリガーのテストと検証

タスク	説明	必要なスキル
ストリームと Lambda トリガーをテストします。	1. Visual Studio で、AWS Explorer に移動します。 2. [AWS Lambda] を展開し、CommandRedeemReward 機能を選択します (ダブルクリック)。表示される関数ウィンドウで、関数をテストできます。	アプリ開発者

タスク	説明	必要なスキル
	3. リクエストテキストボックスに、JavaScript Object Notation (JSON) 形式で、リクエストデータを入力します。リクエストの例については、追加情報 セクションのテストデータを参照してください。 4. 呼び出しを選択します。	
DynamodDB 報酬クエリテーブルを使用して検証します。	1. cqrses-reward-queryテーブルを開きます。 2. 特典を利用した顧客のポイントを確認します。交換したポイントは、お客様の合計ポイントから差し引いてください。	アプリ開発者
CloudWatch Logs を使用して検証します。	1. CloudWatch に移動し、ロググループ を選択します。 2. /aws/lambda/EventSourceReward ロググループには `EventSourceReward` トリガーのログが含まれています。Lambdaの呼び出しはすべてログに記録され、Lambdaコードの `context.Logger.LogLine` と `Console.Writeline` に配置したメッセージも含まれます。	アプリ開発者

タスク	説明	必要なスキル
EventSourceCustomer トリガーを検証します。	`EventSourceCustomer` トリガーを検証するには、`EventSourceCustomer` トリガーのそれぞれの顧客テーブルと CloudWatch ログを使用して、このエピックの手順を繰り返します。	アプリ開発者

関連リソース

リファレンス

- Visual Studio 2019 Community Editionのダウンロード。
- AWS Toolkit for Visual Studioのダウンロード
- AWS Toolkit for Visual Studio ユーザーガイド
- AWS でのサーバーレス
- DynamoDB のユースケースとデザインパターン
- マーティン ファウラー (CQRS)
- マーティン ファウラー イベント ソーシング

動画

- AWS Toolkit for Visual Studioのデモビデオ
- 新しい IAM ユーザーのアクセスキー ID を作成する方法を教えてください。

追加情報

CQRS とイベントソーシング

CQRS

CQRS パターンは、データアクセスオブジェクトの単一 CRUD (作成、読み取り、更新、削除) モデルなどの単一の概念操作モデルを、コマンド操作モデルとクエリ操作モデルに分離します。コマンド

モデルとは、作成、更新、削除など、状態を変更するあらゆる操作を指します。クエリモデルとは、値を返すあらゆる操作を指します。

1. Customer CRUD モデルには以下のインターフェースが含まれます。
   - `Create Customer()`
   - `UpdateCustomer()`
   - `DeleteCustomer()`
   - `AddPoints()`
   - `RedeemPoints()`
   - `GetVIPCustomers()`
   - `GetCustomerList()`
   - `GetCustomerPoints()`

要件が複雑になれば、この単一モデルのアプローチから移行できます。CQRS はコマンドモデルとクエリモデルを使用して、データの書き込みと読み取りの責任を分離します。そうすることで、データを独立して維持管理できます。責任が明確に分離されていれば、各モデルを強化しても他のモデルには影響しません。この分離によってメンテナンスとパフォーマンスが向上し、アプリケーションが大きくなるにつれて複雑さが軽減されます。

1. 顧客 コマンド モデルのインターフェース :
   - `Create Customer()`
   - `UpdateCustomer()`
   - `DeleteCustomer()`
   - `AddPoints()`
   - `RedeemPoints()`
2. 顧客 コマンド モデルのインターフェース :
   - `GetVIPCustomers()`
   - `GetCustomerList()`
   - `GetCustomerPoints()`
   - `GetMonthlyStatement()`

コード例については、ソースコードディレクトリを参照してください。

次に、CQRS パターンによってデータベースが切り離されます。この分離は、マイクロサービスアーキテクチャの主要要素である各サービスの完全な独立性につながります。

AWS クラウドで CQRS を使用すると、各サービスをさらに最適化できます。たとえば、さまざまなコンピューティング設定を設定したり、サーバーレスまたはコンテナベースのマイクロサービスのどちらかを選択できます。オンプレミスのキャッシュを Amazon ElastiCache に置き換えることができます。オンプレミスでメッセージングをパブリッシュ/サブスクライブメッセージングしている場合は、そのメッセージを Amazon Simple Notification Service (Amazon SNS) に置き換えることができます。さらに、従量制料金の料金設定や、使用した分だけ支払うさまざまな AWS サービスを利用できます。

CQRS には次の利点があります。

- 独立スケーリング — 各モデルのスケーリング戦略は、サービスの要件と需要に合わせて調整できます。高性能アプリケーションと同様に、読み取りと書き込みを分離することで、それぞれの需要に合わせてモデルを個別にスケーリングできます。また、コンピュートリソースを追加または削減して、一方のモデルのスケーラビリティ要求に応えることもできますが、もう一方のモデルには影響しません。
- 独立したメンテナンス — クエリモデルとコマンドモデルを分離することで、モデルの保守性が向上します。一方のモデルにコードを変更したり拡張したりしても、もう一方のモデルには影響しません。
- セキュリティ — 権限とポリシーを別々のモデルに適用して読み取りと書き込みを行う方が簡単です。
- 読み込みの最適化 — クエリ用に最適化されたスキーマを定義できます。たとえば、集計データにはスキーマを定義し、ファクトテーブルには別のスキーマを定義できます。
- 統合 — CQRS はイベントベースのプログラミングモデルによく合います。
- 複雑さの管理 — クエリモデルとコマンドモデルへの分離は、複雑な分野に適しています。

Local を使用する場合は、次の点に留意してください。

- CQRS パターンはアプリケーションの特定の部分にのみ適用され、アプリケーション全体には適用されません。パターンに合わないドメインに実装すると、生産性が低下し、リスクが高まり、複雑さが増す可能性があります。

- このパターンは、読み取りと書き込みの操作が不均衡な、頻繁に使用されるモデルに最適です。
- 処理に時間がかかる大規模なレポートなど、読み取りの多いアプリケーションでは、CQRS では適切なデータベースを選択し、集約データを保存するスキーマを作成することができます。これにより、レポートデータを 1 回だけ処理して集計テーブルにダンプすることで、レポートの読み取りと表示の応答時間を短縮できます。
- 書き込みの多いアプリケーションでは、書き込み操作用にデータベースを設定し、書き込みの需要が高まったときにコマンド microservice が独立してスケーリングできるようにすることができます。例については、`AWS.APG.CQRSES.CommandRedeemRewardLambda` および `AWS.APG.CQRSES.CommandAddRewardLambda` マイクロサービスを参照してください。

イベントソーシング

次のステップは、コマンドが実行されたときに、イベントソーシングを使用してクエリデータベースを同期することです。例えば、次の事項を検討します。

- 顧客報酬ポイントが追加され、クエリデータベース内の顧客の合計または集計された報酬ポイントを更新する必要があります。
- 顧客の姓がコマンドデータベースで更新されるため、クエリデータベース内の代理顧客情報を更新する必要があります。

従来の CRUD モデルでは、トランザクションが完了するまでデータをロックすることでデータの一貫性を確保していました。イベントソーシングでは、一連のイベントをパブリッシュすることでデータが同期され、サブスクライバーはそのイベントを消費してそれぞれのデータを更新します。

イベントソーシングパターンでは、データに対して実行された一連のアクションをすべて確認して記録し、一連のイベントを通じて公開します。これらのイベントはデータに加えられた一連の変更を表しており、そのイベントの購読者は記録を最新の状態に保つために処理する必要があります。これらのイベントはサブスクライバーによって処理され、サブスクライバーのデータベース上のデータが同期されます。この場合は、それがクエリデータベースです。

次の図は、AWS 上の CQRS で使用されるイベントソーシングを示しています。

1. コマンド Lambda 関数は、データベースに対して作成、更新、削除などの書き込み操作を実行します。
2. クエリ Lambda 関数は、データベースに対して get や select などの読み取り操作を実行します。

3. この Lambda 関数は、Command データベースからの DynamoDB ストリームを処理し、変更があった場合はクエリデータベースを更新します。また、この機能を使用して Amazon SNS にメッセージを発行し、サブスクライバー がデータを処理できるようにすることもきます。
4. (オプション) Lambda イベントサブスクライバーは Amazon SNS によって発行されたメッセージを処理し、クエリデータベースを更新します。
5. (オプション) Amazon SNS は書き込みオペレーションの E メール通知を送信します。

AWS では、クエリデータベースは DynamoDB Streams によって同期できます。DynamoDB は、DynamoDB テーブル内の項目レベルの変更に関するシーケンスを時間順にキャプチャし、その情報を 24 時間以内に永続的に保存します。

DynamoDB Streams をアクティブ化すると、データベースはイベントソーシングパターンを可能にする一連のイベントを公開できます。イベントソーシングパターンによってイベントサブスクライバーが追加されます。イベントサブスクライバーアプリケーションはイベントを消費し、サブスクライバーの責任に応じて処理します。前の図では、イベントサブスクライバーが変更を Query DynamoDB データベースにプッシュして、データの同期を維持しています。Amazon SNS、メッセージブローカー、イベントサブスクライバーアプリケーションを使用することで、アーキテクチャは切り離された状態に保たれます。

イベントソーシングには次の利点があります。

- トランザクションデータの一貫性
- データに加えられたアクションのモニタリングに使用できる、信頼性の高い監査証跡とアクション履歴
- マイクロサービスなどの分散アプリケーションが、環境全体でデータを同期できるようにします。
- 状態が変化しても、確実にイベントを発行できます。
- 過去の状態の再構築または再生
- モノリシックなアプリケーションからマイクロサービスへの移行のためのイベントを交換する疎結合エンティティ
- 同時更新による競合の軽減。イベントソーシングにより、データストア内のオブジェクトを直接更新する必要がなくなります。
- タスクとイベントを切り離すことによる柔軟性と拡張性
- 外部システム更新
- 1 回のイベントで複数のタスクを管理

イベントソーシングを使用する場合は、次の点に注意してください。

- ソースサブスクライバーデータベース間のデータの更新には多少の遅延があるため、変更を取り消す唯一の方法は、イベントストアに補償イベントを追加することです。
- イベントソーシングの実装はプログラミングスタイルが異なるため、習得には時間がかかります。

テストデータ

デプロイが成功したら、次のテストデータを使用して Lambda 関数をテストします。

CommandCreate Customer

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Done",  "CompanyName":"AnyCompany",
 "Address": "USA",  "VIP":true }
```

CommandUpdate Customer

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Doe",  "CompanyName":"Example Corp.",
 "Address": "Seattle, USA",  "VIP":true }
```

CommandDelete Customer

顧客 ID をリクエストデータとして入力します。たとえば、顧客 ID が 151 の場合、リクエストデータとして 151 と入力します。

```
151
```

QueryCustomerList

これは空白です。呼び出されると、すべての顧客が返されます。

CommandAddReward

これにより、ID 1 (リチャード) のお客様 (リチャード) に 40 ポイント加算されます。

```
{
  "Id":10101,
  "CustomerId":1,
  "Points":40
```

```
}
```

CommandRedeemReward

これにより、ID 1 (リチャード) の顧客に 15 ポイント差し引れます。

```
{
  "Id":10110,
  "CustomerId":1,
  "Points":15
}
```

クエリー/リワード

顧客の ID を入力します。たとえば、リチャードには 1、アーナビには 2、シャーリーの場合は 3 と入力します。

```
2
```

ソースコードディレクトリ

Visual Studio ソリューションのディレクトリ構造のガイドとして次の表を使用してください。

CQRS オンプレミスコードサンプルソリューションディレクトリ

顧客 CRUD モデル

CQRS オンプレミスコードサンプル\ CRUD モデル\ AWS.APG.CQRSES.DAL プロジェクト

顧客 CRUD モデルの CQRS バージョン

- 顧客コマンド：CQRS On-Premises Code Sample\CQRS Model\Command Microservice \AWS.APG.CQRSES.Command プロジェクト
- 顧客クエリ：CQRS On-Premises Code Sample\CQRS Model\Query Microservice \AWS.APG.CQRSES.Query プロジェクト

Command and Query microservices

Command マイクロサービスはソリューションフォルダー CQRS On-Premises Code Sample \CQRS Model\Command Microservice にあります。

- AWS.APG.CQRSES.CommandMicroservice ASP.NET Core API プロジェクトは、コンシューマーがサービスとやり取りするエントリポイントとして機能します。
- AWS.APG.CQRSES.Command .NET Core プロジェクトは、コマンド関連のオブジェクトとインターフェイスをホストするオブジェクトです。

Command マイクロサービスはソリューションフォルダー CQRS On-Premises Code Sample \CQRS Model\Query Microservice にあります。

- AWS.APG.CQRSES.QueryMicroservice ASP.NET Core API プロジェクトは、コンシューマーがサービスとやり取りするエントリポイントとして機能します。
- AWS.APG.CQRSES.Query .NET Core プロジェクトは、コマンド関連のオブジェクトとインターフェイスをホストするオブジェクトです。

CQRS AWS サーバーレスコードソリューションディレクトリ

このコードは、AWS サーバーレスサービスを使用するオンプレミスコードの AWS バージョンです。

C# .NET Core では、各 Lambda 関数は 1 つの.NET Core プロジェクトによって表されます。このパターンのサンプルコードでは、コマンドモデルとクエリモデルのインターフェースごとに個別のプロジェクトがあります。

AWS のサービスの使用

AWS サーバーレスサービスを使用するCQRSのルートソリューションディレクトリは、この CQRS AWS Serverless\CQRS フォルダーにあります。この例には、顧客と報酬という 2 つのモデルが含まれています。

顧客とリワードのLambda関数のコマンドは、CQRS\Command Microservice\Customer および CQRS\Command Microservice\Reward フォルダーにあります。これらには以下の Lambda プロジェクトが含まれています。

- 顧客コマンド：CommandCreateLambda、CommandDeleteLambda、CommandUpdateLambda
- リワードコマンド：CommandAddRewardLambda と CommandRedeemRewardLambda

顧客とリワードのクエリ Lambda 関数は、CQRS\Query Microservice\Customer および CQRS\QueryMicroservice\Reward フォルダーにあります。これらには以下の QueryCustomerListLambda と QueryRewardLambda Lambda プロジェクトが含まれています。

CQRS テストプロジェクト

テストプロジェクトは CQRS\Tests フォルダーの下にあります。このプロジェクトには、CQRS Lambda 関数のテストを自動化するテストスクリプトが含まれています。

AWS サービスを使用したイベントソーシング

次の Lambda イベントハンドラーは、Customer と Reward DynamoDB ストリームによって開始され、クエリテーブル内のデータを処理および同期します。

- EventSourceCustomer Lambda 関数は顧客テーブル (cqrses-customer-cmd) の DynamoDB ストリームにマップされます。
- EventSourceReward Lambda 関数は顧客テーブル (cqrses-reward-cmd) の DynamoDB ストリームにマップされます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

その他のパターン

- AWS PrivateLink と Network Load Balancer を使用して、Amazon EKS 上のコンテナアプリケーションにプライベートアクセスします。
- AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する
- DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する
- CI/CD パイプラインを使用して Amazon EKS へ Java アプリケーションを自動的にビルドし、デプロイする
- AWS CDK を使用してマイクロサービス用の CI/CD パイプラインと Amazon ECS クラスターを自動的に構築する
- BMC AMI Cloud Data を使用してメインフレームデータを Amazon S3 にバックアップおよびアーカイブする
- Amazon EC2 Auto Scaling と Systems Manager を搭載した Micro Focus Enterprise Server PAC を構築する
- Amazon DataZone を使用してエンタープライズデータメッシュを構築する AWS CDKAWS CloudFormation
- サーバーレスアプローチを使用して AWS サービスを連結する
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化
- AWS CodeCommit リポジトリから最新の AWS Amplify ウェブアプリケーションを継続的にデプロイする
- Python を使用して EBCDIC データを AWS 上の ASCII に変換およびアンパックします
- Micro Focusを使用して複雑なレコードレイアウトのメインフレームデータファイルを変換
- CodePipeline と HashiCorp Packer を使用してパイプラインと AMI を作成
- CodePipeline を使用してパイプラインを作成し、アーティファクトの更新をオンプレミスの EC2 インスタンスにデプロイ
- 、Angular AWS Amplify、および Module Federation を使用してマイクロフロントエンド用のポータルを作成する
- Amazon EKS クラスターをデプロイおよびデバッグ
- Elastic Beanstalk を使用してコンテナをデプロイする
- PostgreSQL 互換の Aurora グローバルデータベースを使用して Oracle DR をエミュレート
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する

- QuickSight で AWS Mainframe Modernization と Amazon Q を使用して Db2 z/OS データインサイトを生成する
- Amazon ECR リポジトリに移行するときに、重複するコンテナイメージを自動的に特定する
- Amazon API Gateway でカスタムドメインを使用してパスベースの API バージョニングを実装する
- Oracle SQL Developer と AWS SCT を使用して Amazon RDS for Oracle から Amazon RDS for PostgreSQL に段階的に移行
- Stonebranch ユニバーサルコントローラーと AWS Mainframe Modernizationを統合
- 複数の AWS アカウントと AWS リージョンで AWS Service Catalog 製品を管理
- AWS メンバーアカウントを AWS Organizations から AWS Control Tower に移行する
- Precisely からのConnect を使用して VSAM ファイルを Amazon RDS または Amazon MSK に移行およびレプリケート
- AWS DMS を使用して SAP ASE から Amazon RDS for SQL Server) に移行する
- Oracle 外部テーブルを Amazon Aurora PostgreSQL 互換に移行
- Amazon Q Developer を使用して CardDemo メインフレームアプリケーションをモダナイズする
- Rocket Enterprise Server と LRS VPSX/MFI を使用して、 AWS でメインフレームのバッチ印刷ワークロードをモダナイズする
- Micro Focus Enterprise ServerとLRS VPSX/MFIを使用して、AWS 上のメインフレームのオンライン印刷ワークロードを最新化
- Rocket Enterprise Server と LRS PageCenterX AWS を使用してメインフレーム出力管理をモダナイズする
- Transfer Family を使用して、メインフレームファイルを Amazon S3 に直接移動する
- App2Container が生成した Docker イメージを最適化する
- AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する
- IaC 原則を使用して Amazon Aurora グローバルデータベースのブルー/グリーンデプロイを自動化する
- Precisely Connect を使用してメインフレームデータベースを AWS にレプリケート
- Amazon ECS Anywhere を使用して Amazon WorkSpaces で Amazon ECS タスクを実行する
- から OpenSearch AWS Lambda にテレメトリデータを送信して、リアルタイムの分析と視覚化を行う
- Amazon S3 に Helm v3 チャートリポジトリを設定する

- マルチリージョン、マルチアカウント組織で AWS CloudFormation ドリフト検出を設定する
- AWS Lambda を使用して六角形アーキテクチャで Python プロジェクトを構築する
- LocalStack および Terraform Tests を使用して AWS インフラストラクチャをテストする
- SAP ペースメーカークラスターを ENSA1 から ENSA2 にアップグレード
- Amazon Q Developer をコーディングアシスタントとして使用して生産性を高める
- Account Factory for Terraform (AFT) のコードをローカルで検証する

メインフレーム

トピック

- をインストールして IBM z/OS AWS のサービス からアクセスする AWS CLI
- BMC AMI Cloud Data を使用してメインフレームデータを Amazon S3 にバックアップおよびアーカイブする
- AWS Mainframe Modernization と を使用して COBOL Db2 プログラムを構築する AWS CodeBuild
- Amazon EC2 Auto Scaling と Systems Manager を搭載した Micro Focus Enterprise Server PAC を構築する
- AWS クラウドで高度なメインフレームファイルビューアを構築
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化
- Python を使用して EBCDIC データを AWS 上の ASCII に変換およびアンパックします
- AWS Lambda を使用して Amazon S3 のメインフレームファイルを EBCDIC 形式から文字区切りの ASCII 形式に変換します
- Micro Focusを使用して複雑なレコードレイアウトのメインフレームデータファイルを変換
- Terraform を使用して、コンテナ化された Blu Age アプリケーションの環境をデプロイする
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用して Db2 z/OS データインサイトを生成する
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する
- Stonebranch ユニバーサルコントローラーと AWS Mainframe Modernizationを統合
- Precisely からのConnect を使用して VSAM ファイルを Amazon RDS または Amazon MSK に移行およびレプリケート
- Rocket Enterprise Server と LRS PageCenterX AWS を使用してメインフレーム出力管理をモダナイズする
- Amazon Q Developer を使用して CardDemo メインフレームアプリケーションをモダナイズする
- Rocket Enterprise Server と LRS VPSX/MFI を使用して、 AWS でメインフレームのバッチ印刷ワークロードをモダナイズする
- メインフレームのモダナイゼーション: Rocket Software Enterprise Suite AWS を使用した での DevOps
- Micro Focus Enterprise ServerとLRS VPSX/MFIを使用して、AWS 上のメインフレームのオンライン印刷ワークロードを最新化

- Transfer Family を使用して、メインフレームファイルを Amazon S3 に直接移動する
- トラステッドコンテキストを使用して、AWS の Db2 フェデレーションデータベースのユーザーアクセスを保護し、合理化する
- 大規模な Db2 z/OS データを CSV ファイルで Amazon S3 に転送する
- その他のパターン

をインストールして IBM z/OS AWS のサービス からアクセスする AWS CLI

作成者: Souma Ghosh (AWS)、Phil de Valence (AWS)、Paulo Vitor Pereira (AWS)

概要

AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルでコマンドを使用して複数の AWS サービスを管理するためのオープンソースツールです。最小限の設定で、コマンドプロンプト、ターミナル、bash シェルなどのコマンドラインセッションからコマンドを実行して、ブラウザベースで提供される機能と同じ機能を実装できます AWS Management Console。

のサービスとしての AWS インフラストラクチャ (IaaS) の管理、管理、およびアクセス機能はすべて AWS Management Console 、 AWS API および で利用できます AWS CLI。IBM z/OS メインフレーム AWS CLI に をインストールして、z/OS AWS のサービス から直接アクセス、管理、操作できます。 AWS CLI を使用すると、ユーザーやアプリケーションは次のようなさまざまなタスクを実行できます。

- z/OS と Amazon Simple Storage Service (Amazon S3) オブジェクトストレージ間のファイルまたはデータセットの転送とバケットのコンテンツの表示
- AWS Mainframe Modernization 環境内のバッチジョブの開始など、さまざまな AWS リソースの開始と停止
- AWS Lambda 関数を呼び出して一般的なビジネスロジックを実装する
- 人工知能と機械学習 (AI/ML) および分析サービスとの統合

このパターンでは、z/OS AWS CLI で をインストール、設定、使用する方法について説明します。グローバルにインストールできるため、すべての z/OS ユーザー、またはユーザーレベルで使用できます。このパターンでは、z/OS Unix System Services (USS) からのインタラクティブなコマンドラインセッション AWS CLI で、またはバッチジョブとして を使用する方法についても詳しく説明します。

前提条件と制限

前提条件

- z/OS から へのネットワーク通信 AWS

デフォルトでは、 は TCP ポート 443 で HTTPS AWS のサービス を使用して にリクエスト AWS CLI を送信します。を AWS CLI 正常に使用するには、TCP ポート 443 でアウトバウンド接続を作成できる必要があります。次のいずれかの z/OS USS コマンド (環境にはインストールされていない場合があります) を使用して、z/OS から へのネットワーク接続をテストできます AWS。

```
ping amazonaws.com
dig amazonaws.com
traceroute amazonaws.com
curl -k https://docs.aws.amazon.com/cli/v1/userguide/cli-chap-welcome.html
```

- AWS credentials

  z/OS の AWS クラウド サービスと通信するには、 でターゲットにアクセスするための権限を持つ認証情報を設定 AWS CLI する必要があります AWS アカウント。プログラムによる コマンドでは AWS、アクセスキーを使用できます。アクセスキーは、アクセスキー ID とシークレットアクセスキーで構成されます。アクセスキーがない場合は、 AWS Management Consoleから作成することができます。ベストプラクティスとして、 AWS アカウント ルートユーザーが必須でない限り、ルートユーザーのアクセスキーをタスクに使用しないでください。代わりに、新しい管理者 IAM ユーザーを作成し、アクセスキーを使用してユーザーをセットアップするための最小特権のアクセス許可を準備 します。ユーザーを作成したら、このユーザーのアクセスキー ID とシークレットアクセスキーを作成できます。

  > **Warning**
  >
  > AWS Identity and Access Management (IAM) ユーザーには、セキュリティリスクをもたらす長期的な認証情報があります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

- IBM Python for z/OS

  には Python 3.8 以降 AWS CLI が必要です。IBM では、IBM Open Enterprise Python for z/OS を使用して Python を z/OS で実行できるようになりました。IBM Open Enterprise Python は、Shopz SMP/E から無料で利用できます。または、IBM ウェブサイトから PAX ファイルをダウンロードすることもできます。手順については、IBM Open Enterprise Python for z/OS のインストールと設定のドキュメントを参照してください。

制約事項

- このパターンで提供されるインストール手順は、AWS CLI バージョン 1 にのみ適用されます。の最新バージョン AWS CLI はバージョン 2 です。ただし、このパターンでは、バージョン 2 のインストール方法が異なり、バージョン 2 で使用可能なバイナリ実行可能ファイルは z/OS システムと互換性がないため、古いバージョンを使用します。

製品バージョン

- AWS CLI バージョン 1
- Python 3.8 以降

アーキテクチャ

テクノロジースタック

- z/OS を実行するメインフレーム
- Mainframe z/OS UNIX System Services (米国)
- Mainframe Open MVS (OMVS) – z/OS UNIX シェル環境コマンドインターフェイス
- ダイレクトアクセスストレージデバイス (DASD) などのメインフレームディスク
- AWS CLI

ターゲット アーキテクチャ

次の図は、IBM z/OS での AWS CLI デプロイを示しています。SSH や telnet セッションなどのインタラクティブなユーザーセッション AWS CLI から を呼び出すことができます。バッチジョブから呼び出すには、ジョブ制御言語 (JCL)、または z/OS Unix シェルコマンドを呼び出すことができる任意のプログラムを使用します。

は、TCP/IP ネットワーク経由で AWS のサービス エンドポイントと AWS CLI 通信します。このネットワーク接続は、インターネット経由で、またはカスタマーデータセンターからデータセンターへのプライベート AWS Direct Connect 接続を介して行うことができます AWS クラウド 。通信は AWS 認証情報で認証され、暗号化されます。

自動化とスケール

AWS のサービス を使用して の機能を調べ AWS CLI 、USS シェルスクリプトを開発して z/OS から AWS リソースを管理できます。z/OS バッチ環境から AWS CLI コマンドとシェルスクリプトを実行することもできます。また、メインフレームスケジューラと統合することで、バッチジョブを特定のスケジュールで実行するように自動化できます。 AWS CLI コマンドまたはスクリプトはパラメータ (PARMs) とプロシージャ (PROCs) 内でコーディングでき、パラメータが異なるバッチジョブから PARM または PROC を呼び出す標準的なアプローチに従ってスケーリングできます。

ツール

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

ベストプラクティス

- セキュリティ上の理由から、アクセス許可は、アクセス AWS キーの詳細が保存されている USS ディレクトリに制限します。を使用するユーザーまたはプログラムのみへのアクセスを許可します AWS CLI。
- AWS アカウント ルートユーザーアクセスキーをタスクに使用しないでください。代わりに、新しい管理者 IAM ユーザーを作成し、アクセスキーで設定します。

> **Warning**
>
> IAM ユーザーには、セキュリティリスクをもたらす長期的な認証情報があります。このリスクを軽減するために、これらのユーザーにはタスクの実行に必要な権限のみを付与し、不要になったユーザーを削除することをお勧めします。

エピック

z/OS USS に AWS CLI バージョン 1 をインストールする

タスク	説明	必要なスキル
Python 3.8 以降をインストールします。	1. 次のいずれかの方法を使用して、z/OS USS コマンド	メインフレーム z/OS 管理者

タスク	説明	必要なスキル
	プロンプトインターフェイスにログインします。 • Interactive System Productivity Facility (ISPF) パネルからタイム共有オプション (TSO) OMVS コマンドを使用する、または https://www.ibm.com/docs/en/zos-basic-skills?topic=interfaces-what-is-ispf • SSH または telnet を使用して、メインフレーム論理パーティション (LPAR) の IP に接続します。 このパターンでは、`cliuser`が USS 環境へのログインに使用されるユーザー ID であり、`/u/cliuser/` がユーザーのホームディレクトリであることを前提としています。z/OS 環境では、インストール要件に応じてユーザーのホームディレクトリを異なる方法で設定できます。 2. まだインストールされていない場合は、IBM Open Enterprise Python for z/OS のインストールガイドに	

タスク	説明	必要なスキル
	従って Python 3.8 以降をインストールします。	

タスク	説明	必要なスキル

USS 環境変数を設定します。	プロファイルに環境変数を追加します。これらは、個々のユーザーの `/u/cliuser/.profile` ファイル (cliuser) またはすべてのユーザーの `/etc/profile` ファイルに追加できます。
	
	Note
	このパターンは、Python が `/u/awscli/python` ディレクトリにインストールされていることを前提としています。インストールディレクトリが異なる場合は、それに応じてコードを更新します。
	
	`# Python configuration`
	`export BPXKAUTOCVT='ON'`
	`export CEERUNOPTS='FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)'`
	`export TAGREDIR_ERR=txt`
	`export TAGREDIR_IN=txt`
	`export TAGREDIR_OUT=txt`
	
	`# AWS CLI configuration`
		メインフレーム z/OS 管理者

タスク	説明	必要なスキル
	`export PATH=/u/cliuser/python/bin:$PATH` `export PYTHONPATH=/u/cliuser/python:$PYTHONPATH`	
Python のインストールをテストします。	Python コマンドを実行します。 `python --version` 出力は、Python 3.8 以降が正しくインストールされていることを確認します。	メインフレーム z/OS 管理者

タスク	説明	必要なスキル

pip を確認またはインストールします。	1. pip コマンドは通常、IBM ウェブサイトから Python をインストールするときに自動的にインストールされます。確認するには、 コマンドを実行します。
	`pip –-version`
	pip がインストールされている場合、このコマンドにはインストールされているバージョンが表示されます。
	2. pip コマンドが見つからない場合は、次のコマンドを実行して pip をインストールします。
	`python -m ensurepip --upgrade`
	その他のインストールオプションについては、pip ドキュメントを参照してください。
		メインフレーム z/OS 管理者

タスク	説明	必要なスキル

AWS CLI バージョン 1 をインストールします。	1. をインストールするには AWS CLI、 コマンドを実行します。
	
	`python -m pip install awscli`
	
	出力は以下のようになります。
	
	`Successfully installed PyYAML-6.0.1 awscli-1.32.23 botocore-1.34.23 colorama-0.4.4 docutils-0.16 jmespath-1.0.1 pyasn1-0.5.1 python-dateutil-2.8.2 rsa-4.7.2 s3transfer-0.10.0 urllib3-2.0.7`
	
	2. 次のコマンドを実行して、aws 実行可能ファイルのアクセス許可を変更します。必ず、Python のインストールパス<python_installation_dir> でプレースホルダーディレクトリを更新してください。
	
	`chmod 744 <python_installation_dir>/bin/aws`
		メインフレーム z/OS 管理者

タスク	説明	必要なスキル

	3. 次のコマンドを実行して、AWS CLI インストールをテストします。
	`aws --version`
	出力には、次のような AWS CLI、Python、および botocore のバージョンが表示されます。
	`aws-cli/1.32.3 Python/3.9.5 OS/390/27.00 botocore/1.34.3`	

z/OS からの AWS CLI アクセスを設定する

タスク	説明	必要なスキル
AWS アクセスキー、デフォルトのリージョン、出力を設定します。	このAWS CLI ドキュメントでは、 AWS アクセスを設定するためのさまざまなオプションについて説明します。組織の標準に従って設定を選択できます。この例では、短期認証情報設定を使用します。 1. 次のコマンドを使用して を設定します AWS CLI。 `aws configure` 2. プロンプトが表示されたら、以下の項目の詳細を入	AWS 管理者、メインフレーム z/OS 管理者、メインフレーム z/OS 開発者

タスク	説明	必要なスキル

	力します。アクセスキー ID とシークレットアクセスキーの値は、前提条件ステップで AWS 認証情報を設定するときに取得したキーからのものです。
	
	`AWS Access Key ID [None]: ASIAIOSFODNN7EXAMPLE`
	`AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
	`Default region name [None]: us-east-1`
	`Default output format [None]:`
	`aws configure set aws_session_token IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRINGEXAMPLE`
	
	アクセスキーを含むこの設定は、 `/u/cliuser/.aws` フォルダに保存されます。セキュリティ上の理由から、このフォルダは、 を使用するユーザーまたはプログラムにのみアクセスを許可するように制限します AWS CLI。	

タスク	説明	必要なスキル
をテストします AWS CLI。	1. コマンドプロンプトで次のコマンドを実行して、シンプルなコマンド AWS CLI で をテストします。 `aws s3 ls` 出力には、エラー AWS アカウント なしで設定されたのすべての S3 バケットが一覧表示されます。 2. 次の 2 つのエピックの手順に従って、USS から Amazon S3 にデータを転送します。これらの 2 つのオプションのうち 1 つを選択できます。 • オプション 1 (次のエピック): EBCDIC カンマ区切り値 (CSV) ファイルを Amazon S3 にインタラクティブに転送し、Amazon Athena からファイルをクエリします。 • オプション 2: EBCDIC 固定長データセットをバッチジョブとして Amazon S3 に転送します。	Mainframe z/OS 管理者、Mainframe z/OS 開発者

オプション 1 – USS セッションから Amazon S3 にインタラクティブにデータを転送する

タスク	説明	必要なスキル
サンプル CSV ファイルをダウンロードして転送します。	1. 添付ファイルセクションsales-records.csv からダウンロードします。 ???このファイルは、販売レコードを含むサンプル CSV ファイルを提供します。 2. ファイルを z/OS USS に転送します。 3. 任意のテキストエディタを使用して、USS で/u/cliuser/sales-records.csv ファイルが EBCDIC 形式で読み取れることを確認します。	アプリ開発者、メインフレーム z/OS 開発者

S3 バケットを作成し、CSV ファイルをアップロードします。	1. CSV ファイルを保存する S3 バケットを作成します。
	`aws s3 mb s3://<s3_bucket_name>`
	ここで、 <s3_bucket_name> はバケットの一意の名前です。次に例を示します。
	`aws s3 mb s3://DOC-EXAMPLE-BUCKET1`
	2. z/OS USS から S3 バケットに CSV ファイルをアップロードします。
		アプリ開発者、メインフレーム z/OS 開発者

タスク	説明	必要なスキル

	`aws s3 cp <csv_file _path> s3://<s3_ bucket_name>`
	
	以下に例を示します。
	
	`aws s3 cp /u/cliuser/ sales-records.csv s3://DOC-EXAMPLE-B UCKET1`
	
	3. S3 バケットの内容を一覧表示し、アップロードされたファイルが含まれていることを確認します。
	
	`aws s3 ls s3://<s3_ bucket_name>`
	
	以下に例を示します。
	
	`aws s3 ls s3://DOC-EXAMPLE-BUCKET1`	

タスク	説明	必要なスキル
S3 バケットとアップロードされたファイルを表示します。	1. にサインイン AWS Management Console し、Amazon S3 コンソールを開きます。 2. 新しい S3 バケットとアップロードされたオブジェクトに移動します。 オブジェクトのアップロードの詳細については、Amazon S3 ドキュメントの「Amazon S3 の開始方法」を参照してください。 Amazon S3	AWS 全般

タスク	説明	必要なスキル

Amazon Athena テーブルで SQL クエリを実行します。	1. Amazon Athena コンソールを開きます。
	2. Amazon S3 の CSV データを使用して、新しいテーブル (などDOC-EXAMPLE-BUCKET) を作成します。Amazon S3 詳細については、Amazon S3 ドキュメントのAmazon Athena を使用した Amazon S3 インベントリのクエリ」を参照してください。 Amazon S3
	3. テーブルに対してSELECTクエリを実行して、データを表示します。
	`SELECT * FROM <table_name>;`
	以下に例を示します。
	`SELECT * FROM DOC-EXAMPLE-BUCKET;`
	SQL クエリの出力には、CSV ファイルの内容が表示されます。
		AWS 全般、アプリ開発者

オプション 2 – バッチ JCL を使用して USS から Amazon S3 にデータを転送する

タスク	説明	必要なスキル
サンプルファイルをアップロードします。	1. 添付ファイルセクションsales-records-fixed.txt からダウンロードします。 ???これは、販売レコードを含むサンプルファイルです。テキストファイルの名前を に変更します。たとえば、 を に変更しますUSER.DATA.FIXED 。 2. 固定ブロック (FB)、256 レコード長 (LRECL)、物理シーケンシャル (PS) データセットとしてファイルを z/OS に転送します。 3. データセットリストユーティリティを使用して、USER.DATA.FIXED データセットが ISPF オプション 3.4 の EBCDIC 形式で読み取り可能であることを確認します。出力例については、???「追加情報」セクションを参照してください。	メインフレーム z/OS 開発者
バッチ JCL を作成します。	バッチ JCL を次のようにコードして、送信先 S3 バケットを作成し、データセットをアップロードして、バケットのコンテンツを一覧表示しま	メインフレーム z/OS 開発者

タスク	説明	必要なスキル

	す。ディレクトリ名、ファイル名、バケット名は、必ず独自の値に置き換えてください。
	
	`//AWSCLICP JOB`
	` ACTINFO1,'IBMUSER'`
	`,CLASS=A,MSGCLASS=`
	`H,MSGLEVEL=(1,1),`
	`// NOTIFY=&SYSUID,TIM`
	`E=1440`
	`//*--------------`
	`------------------`
	`------------------`
	`-------`
	`//* Sample job for AWS`
	` CLI`
	`//*--------------`
	`------------------`
	`------------------`
	`-------`
	`//USSCMD EXEC PGM=BPXBA`
	`TCH`
	`//STDERR  DD SYSOUT=*`
	`//STDOUT  DD SYSOUT=*`
	`//STDENV  DD *`
	` export PATH=/u/c`
	`liuser/python/bin:`
	`$PATH`
	`//STDPARM DD *`
	`SH`
	` export _BPXK_AUT`
	`OCVT=ON;`
	` aws s3 mb s3://DOC-`
	`EXAMPLE-BUCKET2;`
	` cp "//'USER.DATA.FIXE`
	`D'" /tmp/tmpfile;`
	` aws s3 cp /tmp/tmpf`
	`ile s3://DOC-EXAMPLE-`
	`BUCKET2/USER.DATA.F`
	`IXED;`	

タスク	説明	必要なスキル
	`rm /tmp/tmpfile; aws s3 ls s3://DOC-EXAMPLE-BUCKET2; /*`	
バッチ JCL ジョブを送信します。	1. 前のステップでコーディングした JCL ジョブを送信します。 2. System Display and Search Facility (SDSF) でジョブのステータスを確認します。成功した場合、ジョブはリターンコード 0 で終わるはずです。 3. ジョブログからの標準出力 (STDOUT) には、バケットの作成、データセットのアップロード、バケットの内容のリストが正常に表示されます。画面の図例については、「追加情報」セクションを参照してください。	メインフレーム z/OS 開発者

タスク	説明	必要なスキル
S3 バケットにアップロードされたデータセットを表示します。	1. にサインイン AWS Management Console し、Amazon S3 コンソールを開きます。 2. テストバケットにアップロードされたファイルに移動します。 3. Amazon Redshift などの分析サービスを使用して、`USER.DATA.FIXED` ファイルをさらに処理したり、分析したりできます。	AWS 全般

関連リソース

- AWS CLI バージョン 1 のドキュメント
- AWS Mainframe Modernization CLI コマンドリファレンス
- AWS Mainframe Modernization

追加情報

ISPF オプション 3.4 の USER.DATA.FIXED (データセットリストユーティリティ)

送信されたバッチジョブの SYSOUT

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

BMC AMI Cloud Data を使用してメインフレームデータを Amazon S3 にバックアップおよびアーカイブする

サントッシュ・クマール・シン (AWS)、ミカエル・リーバーマン (Model9 メインフレーム・ソフトウェア)、ジルベルト・ビオンド (AWS)、マギー・リー (AWS) が制作

概要

このパターンは、メインフレームデータを Amazon Simple Storage Service (Amazon S3) に直接バックアップしてアーカイブし、BMC AMI クラウドデータ (以前は Model9 Manager と呼ばれていました) を使用してそのデータをリコールしてメインフレームに復元する方法を示しています。メインフレームのモダナイゼーションプロジェクトの一環としてバックアップおよびアーカイブソリューションをモダナイズする方法、またはコンプライアンス要件を満たす方法をお探しの場合、このパターンはこれらの目標を達成するのに役立ちます。

通常、メインフレームでコアビジネスアプリケーションを実行する組織は、仮想テープライブラリ (VTL) を使用してファイルやログなどのデータストアをバックアップします。この方法は、請求可能な MIPS を消費し、メインフレーム外のテープに保存されているデータにアクセスできないため、コストがかかる可能性があります。これらの問題を回避するには、BMC AMI Cloud Data を使用して、運用および履歴メインフレームデータを迅速かつ費用対効果の高い方法で Amazon S3 に直接転送できます。BMC AMI Cloud Data を使用すると、IBM z 統合情報プロセッサ (zIIP) エンジンを活用してコスト、並列処理、転送時間を短縮 AWS しながら、TCP/IP 経由でデータを にバックアップおよびアーカイブできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 有効なライセンスキーを持つ BMC AMI Cloud Data
- メインフレームと AWS 間の TCP/IP 接続
- S3 バケットへの読み取り/書き込みアクセス用の AWS Identity and Access Management (IAM) ロール
- BMC AMI クラウドプロセスを実行するためのメインフレームセキュリティ製品 (RACF) アクセス
- 使用可能なネットワークポート、S3 バケットへのアクセスを許可するファイアウォールルール、および専用の z/FS ファイルシステムを持つ BMC AMI Cloud z/OS エージェント (Java バージョン 8 64 ビット SR5 FP16 以降)

- BMC AMI Cloud 管理サーバーの要件を満たす

制約事項

- BMC AMI Cloud Data は、管理サーバーと同じ Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで Docker コンテナとして実行される PostgreSQL データベースに運用データを保存します。Amazon Relational Database Service (Amazon RDS) は現在、BMC AMI クラウドデータのバックエンドとしてサポートされていません。最新の製品更新の詳細については、BMC ドキュメントの「最新情報」を参照してください。
- このパターンは、z/OS メインフレームデータのみをバックアップおよびアーカイブします。BMC AMI Cloud Data は、メインフレームファイルのみをバックアップおよびアーカイブします。
- このパターンでは、データを JSON や CSV などの標準のオープン形式に変換しません。BMC AMI Cloud Analytics (以前は Model9 Gravity と呼ばれていました) などの追加の変換サービスを使用して、データを標準のオープン形式に変換します。クラウドネイティブアプリケーションとデータ分析ツールは、クラウドに書き込まれたデータにアクセスできます。

製品バージョン

- BMC AMI クラウドデータバージョン 2.x

アーキテクチャ

ソーステクノロジースタック

- z/OS を実行するメインフレーム
- データセットや z/OS UNIX System Services (USS) ファイルなどのメインフレームファイル
- ダイレクトアクセスストレージデバイス (DASD) などのメインフレームディスク
- メインフレームテープ (仮想または物理テープライブラリ)

ターゲットテクノロジースタック

- Amazon S3
- Virtual Private Cloud (VPC) での Amazon EC2 インスタンス
- AWS Direct Connect
- Amazon Elastic File System (Amazon EFS)

ターゲット アーキテクチャ

次の図は、メインフレーム上の BMC AMI Cloud Data ソフトウェアエージェントがAmazon S3にデータを保存するレガシーデータのバックアップおよびアーカイブプロセスを駆動するリファレンスアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. BMC AMI Cloud Data ソフトウェアエージェントは、メインフレーム論理パーティション (LPARs) で実行されます。ソフトウェアエージェントは、DASD またはテープから TCP/IP 経由で Amazon S3 に直接メインフレームデータの読み取りと書き込みを行います。
2. AWS Direct Connect は、オンプレミスネットワークと の間に物理的に分離された接続を設定します AWS。セキュリティを強化するには、 上でsite-to-site VPN を実行して AWS Direct Connect 、転送中のデータを暗号化します。
3. S3 バケットはメインフレームファイルをオブジェクトストレージデータとして保存し、BMC AMI Cloud Data エージェントは S3 バケットと直接通信します。証明書は、エージェントと Amazon S3 間のすべての通信の HTTPS 暗号化に使用されます。Amazon S3 データ暗号化は、保管中のデータを暗号化して保護するために使用されます。
4. BMC AMI クラウドデータ管理サーバーは、EC2 インスタンスで Docker コンテナとして実行されます。インスタンスはメインフレーム LPAR や S3 バケット上で稼働するエージェントと通信します。
5. Amazon EFS はアクティブ EC2 インスタンスとパッシブ EC2 インスタンスの両方にマウントされ、ネットワークファイルシステム (NFS) ストレージを共有します。これは、フェイルオーバー時に管理サーバーで作成されたポリシーに関連するメタデータが失われないようにするためです。アクティブなサーバーによるフェイルオーバーが発生した場合、パッシブサーバーにはデータ損失なしでアクセスできます。パッシブサーバーに障害が発生した場合、データ損失なしでアクティブサーバーにアクセスできます。

ツール

AWS サービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 にスケーラブルなコンピューティング能力を提供します AWS クラウド。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。

- Amazon Elastic File System (Amazon EFS) は、 で共有ファイルシステムを作成および設定するのに役立ちます AWS クラウド。
- Amazon Simple Storage Service (Amazon S3) は、ほぼあらゆる量のデータを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークで AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。
- AWS Direct Connectは、標準イーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect ロケーションにリンクします。この接続を使用すると、ネットワークパスでインターネットサービスプロバイダーをバイパスしながら、パブリック AWS サービスへの仮想インターフェイスを直接作成できます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理します。

BMC ツール

- BMC AMI クラウド管理サーバーは、Amazon EC2 の Amazon Linux Amazon マシンイメージ (AMI) で Docker コンテナとして実行される GUI アプリケーションです。管理サーバーには、レポート、ポリシーの作成と管理、アーカイブの実行、バックアップ、リコール、復元の実行などの BMC AMI Cloud アクティビティを管理する機能が用意されています。
- BMC AMI Cloud エージェントは、TCP/IP を使用してファイルをオブジェクトストレージに直接読み書きするオンプレミスのメインフレーム LPAR で実行されます。開始されたタスクはメインフレーム LPAR で実行され、Amazon S3 との間でバックアップおよびアーカイブデータの読み取りと書き込みを行います。
- BMC AMI Cloud Mainframe Command Line Interface (M9CLI) には、管理サーバーに依存することなく、TSO/E から直接、またはバッチオペレーションで BMC AMI Cloud アクションを実行するための一連のコマンドが用意されています。

エピック

S3 バケットと IAM ポリシーを作成

タスク	説明	必要なスキル
S3 バケットを作成します。	「S3 バケット」を作成して、メインフレーム環境からバックアップおよびアーカイブするファイルとボリュームを保存します。	AWS 全般
IAM ポリシーを作成します。	すべての BMC AMI Cloud 管理サーバーとエージェントには、前のステップで作成した S3 バケットへのアクセスが必要です。 必要なアクセス権を付与するには、次の IAM ポリシーを作成します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "Listfolder", "Action": [ "s3:ListBucket", "s3:GetBucketLocation", "s3:ListBucketVersions" ], "Effect": "Allow",`	AWS 全般

タスク	説明	必要なスキル
	"Resource": ["arn:aws:s3:::<Bucket Name>"] }, { "Sid": "Objectaccess", "Effect": "Allow", "Action": ["s3:PutObject", "s3:GetObjectAcl", "s3:GetObject", "s3:DeleteObjectVersion", "s3:DeleteObject", "s3:PutObjectAcl", "s3:GetObjectVersion"], "Resource": ["arn:aws:s3:::<Bucket Name>/*"] }] }	

BMC AMI Cloud ソフトウェアライセンスを取得し、ソフトウェアをダウンロードする

タスク	説明	必要なスキル
BMC AMI Cloud ソフトウェアライセンスを取得します。	ソフトウェアライセンスキーを取得するには、BMC AMI クラウドチームにお問い合わせください。ライセンスを生成するには z/OS `D M=CPU` コマンドの出力が必要です。	ビルドリード
BMC AMI Cloud ソフトウェアとライセンスキーをダウンロードします。	インストールファイルとライセンスキーを取得するには、BMC ドキュメントの指示に従います。	メインフレームインフラストラクチャ管理者

メインフレームに BMC AMI Cloud ソフトウェアエージェントをインストールする

タスク	説明	必要なスキル
BMC AMI Cloud ソフトウェアエージェントをインストールします。	1. インストールプロセスを開始する前に、エージェントのソフトウェアとハードウェアの最小要件を満たしていることを確認してください。 2. エージェントをインストールするには、BMC ドキュメントの指示に従います。 3. エージェントがメインフレーム LPAR で実行を開始したら、`ZM91000I MODEL9 BACKUP AGENT INITIALIZED` スプール内のメッセージを確認します。エージェント	メインフレームインフラストラクチャ管理者

タスク	説明	必要なスキル
	の STDOUT で`Object store connectivity has been established successfully` メッセージを検索して、エージェントと S3 バケット間の接続が正常に確立されていることを確認します。	

EC2 インスタンスで BMC AMI Cloud 管理サーバーを設定する

タスク	説明	必要なスキル
Amazon EC2 Linux 2 インスタンスを作成します。	Amazon EC2 ドキュメント」の「ステップ 1: インスタンスを起動する」の手順に従って、異なるアベイラビリティーゾーンで 2 つの Amazon EC2 Linux 2 インスタンスを起動します。 インスタンスは、以下の推奨されるハードウェアおよびソフトウェアの要件を満たしている必要があります。 • CPU — 最低 4 コア • RAM — 8 GB 以上 • ドライブ — 40 GB • 推奨の EC2 インスタンス — C5.xlarge • OS – Linux • ソフトウェア — ドッカー、解凍、VI/Vim	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
	• ネットワーク帯域幅 – 最小 1 GB 詳細については、「BMC ドキュメント」を参照してください。	
Amazon EFS ファイルシステムを作成します。	Amazon EFS ドキュメントの「ステップ 1: Amazon EFS ファイルシステムを作成する」 の指示に従って Amazon EFS ファイルシステムを作成します。 ファイルシステムを作成するときは、次の操作を行います。 • 標準ストレージクラスを選択します。 • 使用された同じな VPC を選択して、EC2 インスタンスを起動します。	クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
Docker をインストールし、管理サーバーを設定します。	EC2 インスタンスに接続します。 Amazon EC2 ドキュメントの「Linux インスタンスへのConnect」の指示に従って EC2 インスタンスに接続します。 EC2 インスタンスを設定します。 各インスタンスタイプでは、以下の作業を行います。 1. Docker をインストールするには、 コマンドを実行します。 ``` sudo yum install docker ``` 2. Docker を起動するには、コマンドを実行します。 ``` sudo service docker start ``` 3. Docker のステータスを検証するには、 コマンドを実行します。 ``` sudo service docker status ``` 4. `/etc/selinux` フォルダー内の `config` ファイルを `SELINUX=p`	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	ermissive に変更します。
	5. model9-v2.x.y_build_build-id-server.zip および VerificationScripts.zip ファイル (以前にダウンロードした) を EC2 インスタンスの一時フォルダ (インスタンスの/var/tmpフォルダなど) にアップロードします。
	6. tmp フォルダに移動するには、 コマンドを実行します。
	`cd/var/tmp`
	7. 検証スクリプトを解凍するには、 コマンドを実行します。
	`unzip VerificationScripts.zip`
	8. ディレクトリを変更するには、 コマンドを実行します。
	`cd /var/tmp/sysutils/PrereqsScripts`
	9. 検証スクリプトを実行するには、 コマンドを実行します。	

タスク	説明	必要なスキル
	`./M9VerifyPrereqs.sh` 10検証スクリプトが入力を求められたら、Amazon S3 URL とポート番号を入力します。次に、z/OS IP/DNS とポート番号を入力します。 **Note** スクリプトはチェックを実行して、EC2 インスタンスがメインフレームで実行されている S3 バケットとエージェントに接続できることを確認します。接続が確立されると、成功メッセージが表示されます。	

タスク	説明	必要なスキル

管理サーバーソフトウェアをインストールします。	1. アクティブサーバーにする予定の EC2 インスタンスのルートディレクトリ (例:/data/model9) にフォルダとサブフォルダを作成します。
	2. パッケージをインストールamazon-efs-utilsし、前に作成した Amazon EFS ファイルシステムをマウントするには、次のコマンドを実行します。
	`sudo yum install -y amazon-efs-utils`
	`sudo mount -t efs -o tls <File System ID>:/ /data/model9`
	3. EC2 インスタンスの/etc/fstab ファイルを Amazon EFS ファイルシステムのエントリで更新するには (Amazon EC2 の再起動時に Amazon EFS が自動的に再マウントされるようにするには) 、 コマンドを実行します。
	`<Amazon-EFS-file-system-id>:/ /data/model9 efs defaults,_netdev 0 0`
		クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル

	4. BMC AMI Cloud のインストールファイルへのパスとターゲットのインストール場所を定義するには、次のコマンドを実行して変数をエクスポートします。
	
	`export MODEL9_HOME=/data/model9`
	`export M9INSTALL=/var/tmp`
	
	Note
	これらの EXPORT コマンドを .bashrcスクリプトに追加することをお勧めします。
	
	5. ディレクトリを変更するには、mkdir diag コマンドを実行し、cd $MODEL9_HOME コマンドを実行して別のサブディレクトリを作成します。
	6. インストールファイルを解凍するには、 コマンドを実行します。
	
	`unzip $M9INSTALL/model9-<v2.x.y>_build_<build-id>-server.zip`	

タスク	説明	必要なスキル

	Note
	x.y (バージョン) と を自分の値build-idに置き換えます。
	7. アプリケーションをデプロイするには、次のコマンドを実行します:
	`docker load -i $MODEL9_HOME/model9-<v2.x.y>_build_<build-id>.docker`
	`docker load -i $MODEL9_HOME/postgres-12.10-x86.docker.gz`
	Note
	v2.x.y (バージョン) と を自分の値build-idに置き換えます。
	8. `$MODEL9_HOME/conf` フォルダー内の `model9-local.yml` ファイルを更新します。
	Note
	一部のパラメータにはデフォルト値があ	

タスク	説明	必要なスキル

	り、他のパラメータは必要に応じて更新できます。詳細については、 model9-local.yml ファイルの手順を参照してください。
	9. という名前のファイルを作成し$MODEL9_HOME/conf、次のパラメータをファイルに追加します。
	`TZ=America/New_York`
	`EXTRA_JVM_ARGS=-Xmx2048m`
	10. Docker ネットワークブリッジを作成するには、コマンドを実行します。
	`docker network create -d bridge model9network`
	11. BMC AMI Cloud の PostgreSQL データベースコンテナを起動するには、次のコマンドを実行します。
	`docker run -p 127.0.0.1:5432:5432 \`
	`-v $MODEL9_HOME/db/data:/var/lib/postgresql/data:z \`	

タスク	説明	必要なスキル
	--name model9db -- restart unless-st opped \ --network model9net work \ -e POSTGRES_PASSWORD= model9 -e POSTGRES_ DB=model9 -d postgres:12.10	
	12PostgreSQL コンテナの実行が開始されたら、次のコマンドを実行してアプリケーションサーバーを起動します。 docker run -d -p 0.0.0.0:443:443 -p 0.0.0.0:80:80 \ --sysctl net.ipv4. tcp_keepalive_time =600 \ --sysctl net.ipv4. tcp_keepalive_intv l=30 \ --sysctl net.ipv4. tcp_keepalive_prob es=10 \ -v $MODEL9_HOME:/mode l9:z -h $(hostname) --restart unless-st opped \ --env-file $MODEL9_H OME/conf/model9.env \ --network model9net work \ --name model9-v2.x.y model9:<v2.x.y>.<b uild-id>	

タスク	説明	必要なスキル
	Note v2.x.y (バージョン) と を自分の値build-idに置き換えます。 13両方のコンテナのヘルスステータスを確認するには、コマンドを実行します。 `docker ps -a` 14パッシブ EC2 インスタンスに管理サーバーをインストールするには、ステップ 1 ~ 4、7、10 ~ 13 を繰り返します。 **Note** 問題をトラブルシューティングするには、/data/model9/logs/ フォルダに保存されているログに移動します。詳細については、「BMC ドキュメント」を参照してください。	

エージェントを追加し、BMC AMI クラウド管理サーバーでバックアップまたはアーカイブポリシーを定義する

タスク	説明	必要なスキル
新しいエージェントを追加します。	新しいエージェントを追加する前に、以下を確認してください。 • BMC AMI Cloud エージェントはメインフレーム LPAR で実行されており、完全に初期化されています。スプール内の`ZM91000I MODEL9 BACKUP AGENT INITIALIZED` 初期化メッセージを検索して、エージェントを特定します。 • 管理サーバーの Docker コンテナは完全に初期化され、実行されます。 バックアップポリシーとアーカイブポリシーを定義する前に、管理サーバーでエージェントを作成する必要があります。エージェントを作成するには、以下の作業を行います。 1. ウェブブラウザを使用して、Amazon EC2 マシンにデプロイされている管理サーバーにアクセスし、メインフレーム認証情報を使用してログインします。	メインフレームストレージ管理者または開発者

タスク	説明	必要なスキル
	2. エージェントタブを選択し、新しいエージェントの追加を選択します。 3. 名前では、エージェン名を入力します。 4. ホスト名/IP アドレスには、メインフレームのホスト名または IP アドレスを入力します。 5. ポートでは、ポート番号を入力します。 6. 接続のテストを選択します。接続が正常に確立されると、成功メッセージが表示されます。 7. CREATE を選択します。 エージェントが作成されると、テーブルに表示される新しいウィンドウに、オブジェクトストレージとメインフレームエージェントに対する接続ステータスが表示されます。	

タスク	説明	必要なスキル
バックアップポリシーまたはアーカイブポリシーを作成する。	1. ポリシーを選択します。 2. ポリシーを作成を選択します。 3. 新しいポリシーの作成ページで、ポリシーの仕様を入力します。 **Note** 使用可能な仕様の詳細については、BMC ドキュメントの「新しいポリシーの作成」を参照してください。 4. [Finish] を選択してください。 5. これで、新しいポリシーが表として一覧表示されました。この表を表示するには、ポリシータブを選択します。	メインフレームストレージ管理者または開発者

管理サーバーからバックアップポリシーまたはアーカイブポリシーを実行します。

タスク	説明	必要なスキル
バックアップポリシーまたはアーカイブポリシーを実行する。	管理サーバーから以前に作成したデータバックアップまたはアーカイブポリシーを手動または自動で (スケジュールに基づいて) 実行します。ポリシーを手動で実行するには：	メインフレームストレージ管理者または開発者

タスク	説明	必要なスキル
	1. ナビゲーションメニューからポリシータブを選択します。 2. 実行するポリシーのテーブルの右側で、3 つのドットのメニューを選択します。 3. 今すぐ実行を選択します。 4. ポップアップ確認ウィンドウではい、今すぐポリシーを実行を選択します。 5. ポリシーが実行されたら、ポリシーアクティビティセクションで実行ステータスを確認します。 6. 実行されたポリシーで、3 ドットメニューを選択し、実行ログの表示を選択してログを表示します。 7. バックアップが作成されたことを確認するには、S3 バケットを確認します。	

タスク	説明	必要なスキル
バックアップポリシーまたはアーカイブポリシーを復元します。	1. ナビゲーションメニューで、ポリシータブを選択します。 2. 復元プロセスを実行するポリシーを選択します。これにより、その特定のポリシーに対して過去に実行されたすべてのバックアップアクティビティまたはアーカイブアクティビティが一覧表示されます。 3. 復元するバックアップを選択するには、日付/時間列を選択します。`file/Volume/Storage` グループ名にはポリシーの実行詳細が表示されます。 4. テーブルの右側で、3 つのドットのメニューを選択し、RESTORE を選択します。 5. ポップアップウィンドウで、ターゲット名、ボリューム、ストレージグループを入力し、RESTORE を選択します。 6. メインフレームの認証情報を入力し、もう一度 RESTORE を選択します。 7. 復元が成功したことを確認するには、ログまたはメインフレームを確認します。	メインフレームストレージ管理者または開発者

メインフレームからバックアップポリシーまたはアーカイブポリシーを実行します。

タスク	説明	必要なスキル
M9CLI を使用してバックアップまたはアーカイブポリシーを実行します。	M9CLI を使用して、TSO/E、REXX、または JCLs 経由でバックアップおよび復元プロセスを実行します。BMC AMI クラウド管理サーバーでルールを設定する必要はありません。 TSO/E の使用： TSO/E を使用する場合は、`M9CLI REXX`が に連結されていることを確認しますTS0。TSO/E を使用してデータセットをバックアップするには、 `TSO M9CLI BACKDSN <DSNAME>` コマンドを使用します。 **Note** M9CLI コマンドの詳細については、BMC ドキュメントの「CLI リファレンス」を参照してください。 JCLs の使用： JCL を使用してバックアップとアーカイブのポリシーを実行するには、`M9CLI` コマンドを実行します。	メインフレームストレージ管理者または開発者

タスク	説明	必要なスキル

	バッチオペレーションの使用：
	
	次の例は、 M9CLI コマンドをバッチで実行してデータセットをアーカイブする方法を示しています。
	
	`//JOBNAME JOB …`
	`//M9CLI EXEC PGM=IKJEF`
	`T01`
	`//STEPLIB DD DISP=SHR,`
	`DSN=<MODEL9 LOADLIB>`
	`//SYSEXEC DD DISP=SHR,`
	`DSN=<MODEL9 EXEC LIB>`
	`//SYSTSPRT DD SYSOUT=*`
	`//SYSPRINT DD SYSOUT=*`
	`//SYSTSIN DD TSO M9CLI`
	` ARCHIVE`
	` M9CLI ARCHIVE <DSNNAME`
	` OR DSN PATTERN>`
	`/`	

タスク	説明	必要なスキル
JCL バッチでバックアップポリシーまたはアーカイブポリシーを実行します。	BMC AMI Cloud には、M9SAPIJ というサンプル JCL ルーチンが用意されています。M9SAPIJ をカスタマイズして、JCL を使用して管理サーバーで作成された特定のポリシーを実行できます。このジョブは、バックアップと復元のプロセスを自動的に実行するバッチスケジューラの一部にすることもできます。 バッチジョブには以下の必須値が必要です。 • 管理サーバの IP アドレス/ホスト名 • ポート番号 • ポリシー ID またはポリシー名 (管理サーバー上で作成される) **Note** サンプルジョブの手順に従って、他の値を変更することもできます。	メインフレームストレージ管理者または開発者

関連リソース

- 「AWS によるメインフレームのモダナイゼーション」(AWS ドキュメント)

- 「メインフレーム向けクラウドBackup が Model9 と AWS でコストを削減する方法」(AWS パートナーネットワークブログ)
- 「Model9 を使用して AWS でメインフレームデータ分析を有効にする方法」(AWS パートナーネットワークブログ)
- 「AWS Direct Connect の耐障害性に関する推奨事項」(AWS ドキュメント)
- BMC AMI Cloud ドキュメント (BMC ウェブサイト)

AWS Mainframe Modernization と を使用して COBOL Db2 プログラムを構築する AWS CodeBuild

作成者: Luis Gustavo Dantas (AWS) と Eduardo Zimelewicz (AWS)

概要

このパターンでは、 AWS Mainframe Modernization Replatform ツールを使用して COBOL Db2 プログラムをプリコンパイルしてバインドするシンプルな AWS CodeBuild プロジェクトを作成する方法について説明します。これにより、 AWS Mainframe Modernization Replatform ランタイム環境でこれらのプログラムのデプロイと実行が可能になります。

ビジネス指向のプログラミング言語である COBOL は、その信頼性と読みやすさにより、多くの重要なアプリケーションを強化します。リレーショナルデータベース管理システムである IBM Db2 は、大量のデータを効率的に管理し、SQL を介して COBOL プログラムと統合します。COBOL と Db2 は、新しいテクノロジーの出現にもかかわらず、金融や政府などの業界でミッションクリティカルな運用のバックボーンを形成します。

COBOL および Db2 コンポーネントをメインフレーム環境から他のプラットフォームに移行すると、プラットフォームの互換性、統合の複雑さ、データ移行、パフォーマンスの最適化などの課題が生じます。これらの重要なコンポーネントを移動するには、信頼性と機能を維持しながら円滑な移行を確保するために、慎重な計画、技術的専門知識、リソースが必要です。

この AWS Mainframe Modernization サービスは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスなどの AWS インフラストラクチャで実行するメインフレームアプリケーションとデータベースをリプラットフォームするためのツールとリソースを提供します。これには、メインフレームワークロードを、大きなコード変更なしでクラウドに移行することが含まれます。

Db2 プリコンパイルおよびバインドプロセスは、データベースアプリケーションのパフォーマンスと信頼性を最適化するために不可欠です。事前コンパイルにより、埋め込み SQL ステートメントが実行可能コードに変換されるため、ランタイムのオーバーヘッドが軽減され、効率が向上します。バインドプロセスは、プリコンパイルされたコードをデータベース構造にリンクし、アクセスパスとクエリの最適化を容易にします。このプロセスにより、データの整合性が確保され、アプリケーションの応答性が向上し、セキュリティの脆弱性から保護されます。適切にプリコンパイルおよびバインドされたアプリケーションは、リソースの消費を最小限に抑え、スケーラビリティを強化し、SQL インジェクション攻撃のリスクを軽減します。

前提条件と制限

前提条件

- AWS アカウント と管理レベルのコンソールアクセス。
- IBM Db2 for z/OS や Db2 for Linux、Unix、Windows (LUW) などの IBM Db2 データベースシステム。
- IBM Data Server Client ソフトウェア。IBM ウェブサイトからダウンロードできます。詳細については、「IBM Data Server クライアントとデータサーバードライバータイプ」を参照してください。
- コンパイルおよびバインドされる COBOL Db2 プログラム。または、このパターンは、使用できる基本的なサンプルプログラムを提供します。
- プライベートネットワーク AWS を持つ 上の仮想プライベートクラウド (VPC)。VPC の作成の詳細については、Amazon Virtual Private Cloud (Amazon VPC) のドキュメントを参照してください。
- GitHub や GitLab などのソースコントロールリポジトリ。

機能制限

- クォータについては、 AWS CodeBuild 「 のクォータ AWS CodeBuild」を参照してください。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ソーステクノロジースタック

ソーススタックには以下が含まれます。

- Db2 データベースを使用してデータを保存する COBOL プログラム
- IBM COBOL コンパイラと Db2 for z/OS プリコンパイラ
- ファイルシステム、トランザクションマネージャー、スプールなど、メインフレーム設定の他の部分

ターゲットテクノロジースタック

このパターンのアプローチは、z/OS の場合は Db2 から LUW の場合は Db2 にデータを移動するか、z/OS の場合は Db2 にとどまるという 2 つのオプションで機能します。ターゲットアーキテクチャには以下が含まれます。

- Db2 データベースを使用してデータを保存する COBOL プログラム
- AWS Mainframe Modernization リプラットフォームコンパイルツール
- AWS CodeBuild アプリケーションをビルドするためのインフラストラクチャとしての
- Amazon Linux などのその他の AWS クラウド リソース

ターゲット アーキテクチャ

この図表は、以下を示すものです：

1. ユーザーは、GitHub や GitLab GitLab などのソースコントロールリポジトリにコードをアップロードします。
2. AWS CodePipeline は変更に気づき、リポジトリからコードを取得します。
3. CodePipeline はコードを起動 AWS CodeBuild して送信します。
4. CodeBuild は、`buildspec.yml`テンプレート (追加情報セクションで提供) の指示に従って、以下を行います。
   a. Amazon Simple Storage Service (Amazon S3) バケットから IBM データサーバークライアントを取得します。
   b. IBM Data Server クライアントをインストールしてセットアップします。
   c. から Db2 認証情報を取得します AWS Secrets Manager。
   d. Db2 サーバーに接続します。
   e. COBOL プログラムをプリコンパイル、コンパイル、バインドします。
   f. が AWS CodeDeploy 使用する S3 バケットに完成した製品を保存します。
5. CodePipeline は CodeDeploy を開始します。
6. CodeDeploy は、ランタイム環境に既にインストールされているエージェントを調整します。エージェントは Amazon S3 からアプリケーションを取得し、「」の手順に基づいてインストールします`appspec.yml`。

モノをシンプルに保ち、ビルドに集中させるために、このパターンの手順ではステップ 1 から 4 までをカバーしますが、COBOL Db2 プログラムのデプロイは含まれません。

自動化とスケール

わかりやすくするために、このパターンではリソースを手動でプロビジョニングする方法について説明します。ただし、 AWS CloudFormation AWS Cloud Development Kit (AWS CDK)や HashiCorp Terraform など、これらのタスクを自動化する自動化オプションは多数あります。詳細については、AWS CloudFormation「」および「」のAWS CDKドキュメントを参照してください。

ツール

AWS のサービス

- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeDeploy は、Amazon EC2 またはオンプレミスインスタンス、 AWS Lambda 関数、または Amazon Elastic Container Service (Amazon ECS) サービスへのデプロイを自動化します。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS Mainframe Modernization には、メインフレームから AWS マネージドランタイム環境への移行とモダナイゼーションの計画と実装に役立つツールとリソースが用意されています。

その他のツール

- リプラットフォームツール用の Amazon ECR AWS Mainframe Modernization イメージ。COBOL アプリケーションをコンパイルするには、 AWS Mainframe Modernization Replatform ツールを含む Amazon Elastic Container Registry (Amazon ECR) イメージを使用して CodeBuild を開始する必要があります。

  ```
  673918848628.dkr.ecr.<your-region>.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1
  ```

  使用可能な ECR イメージの詳細については、 AWS Mainframe Modernization ユーザーガイドのチュートリアルを参照してください。
- IBM Data Server Client ソフトウェアは、CodeBuild で COBOL Db2 プログラムを事前コンパイルおよびバインドするために不可欠です。COBOL コンパイラと Db2 間のブリッジとして機能します。

ベストプラクティス

- すべての COBOL プログラムがデータ永続化レイヤーとして Db2 に依存しているわけではありません。Db2 にアクセスするためのコンパイルディレクティブが、Db2 とやり取りするように特別に設計された COBOL プログラムにのみ適用されます。COBOL Db2 プログラムと Db2 を使用しない COBOL プログラムを区別するロジックを実装しますDb2。
- 変更されていないプログラムのコンパイルは避けることをお勧めします。コンパイルが必要なプログラムを特定するプロセスを実装します。

エピック

クラウドインフラストラクチャを作成する

タスク	説明	必要なスキル
IBM Data Server クライアントとパイプラインアーティファクトをホストする S3 バケットを作成します。	S3 バケットを設定して、(a) IBM Data Server Client をアップロードし、(b) リポジトリからコードを保存し、(c) ビルドプロセスの結果を保存する必要があります。 1. にサインインし AWS Management Console、Amazon S3 コンソールを開きます。 2. 既存の S3 バケットを選択するか、新しいバケットを作成します。バケットの Amazon リソースネーム (ARN) を書き留めておきます。 S3 バケットを作成する方法については、Amazon S3 ドキュメントを参照してください。	AWS 全般

タスク	説明	必要なスキル
IBM Data Server クライアントを S3 バケットにアップロードします。	1. Amazon S3 コンソールで、バケットを選択して開きます。 2. フォルダの作成を選択し、その名前をクライアントとして指定してから、フォルダの作成を選択します。 3. クライアントフォルダを開き、アップロード、ファイルの追加を選択します。 4. IBM https://www.ibm.com/docs/en/db2/11.5?topic=overviews-data-server-clientsウェブサイトからローカルファイルシステムに以前にダウンロードした IBM Data Server Client ファイルを選択します。 ファイル名は `v11.5.8_linuxx64_client.tar.gz` または に似ている必要がありますv11.5.9_linuxx64_client.tar.gz 。 5. Open、Upload を選択し、アップロードが完了するまで待ちます。 6. ファイルとフォルダタブで、データサーバークライアントを選択し、その S3 URI を書き留めます。	AWS 全般

タスク	説明	必要なスキル
Db2 認証情報の AWS Secrets Manager シークレットを作成します。	DB2 認証情報を安全に保存するためのシークレットを作成するには： 1. Secrets Manager コンソールで、新しいシークレットの保存を選択します。 2. シークレットタイプの選択ペインで、別のタイプのシークレットとプレーンテキストを選択します。 3. プレーンテキストボックスに、次の JSON 構造を使用して Db2 認証情報を入力します。 `{ "username": "<your-db2-user-name>", "password": "<your-db2-password>", "db2node": "db2dev", "db2host": "<your-db2-hostname-or-IP>", "db2port": <your-db2-port>, "db2name": "<your-db2-location>", "qualifier": "<your-db2-qualifier>" }`	AWS 全般

タスク	説明	必要なスキル
	4. 次へ を選択し、シークレットに などの名前を付けますdev-db2-cred 。 5. Next、Next、および Store を選択します。 シークレットの作成の詳細については、Secrets Manager のドキュメントを参照してください。	

タスク	説明	必要なスキル
VPC サブネットから Db2 にアクセスできることを確認します。	AWS CodeBuild では、Data Server クライアントが事前コンパイルおよびバインド操作を実行できるように、Db2 サーバーへの接続が必要です。CodeBuild が安全な接続を介して Db2 サーバーに到達できることを確認します。 1. Amazon VPC コンソールを開きます。 2. ナビゲーションペインで、サブネットを選択し、CodeBuild が機能するプライベートサブネットの IDs と IPv4 CIDRs を書き留めます。 3. インバウンドルールを導入して、Db2 システムの現在のネットワークアクセスコントロール設定を更新します。このルールは、CodeBuild プロジェクトに関連付けられているサブネット CIDRs からのみ Db2 ポートへのカスタム TCP アクセスを有効にする必要があります。	ネットワーク管理者、AWS 全般

アプリケーションアーティファクトを作成する

タスク	説明	必要なスキル
COBOL Db2 アセットを作成します。	1. シンプルな COBOL Db2 の例を使用する場合は、次のソースコードを として保存しますCDB2SMP.cbl 。または、この例を既に所有しているプログラムに置き換えることができます。 `IDENTIFICATION DIVISION. PROGRAM-ID. CDB2SMP. DATA DIVISION. WORKING-STORAGE SECTION. 01 WS-NAME PIC X(100). PROCEDURE DIVISION. EXEC SQL SELECT NAME INTO :WS-NAME FROM SYSIBM.SYSTABLES END-EXEC GOBACK.` 2. 変更をコミットし、ファイルをリポジトリにプッシュします。	アプリ開発者
`buildspec.yml` ファイルを作成します。	1. ??? 「追加情報」セクションで提供されている例に基づいて `buildspec.yml` ファイルを生成します。	AWS DevOps

タスク	説明	必要なスキル
	2. 変更をコミットし、ファイルをリポジトリにプッシュします。	
リポジトリを CodePipeline に接続します。	1. 「AWS 開発者向けツールコンソール」を開きます。 2. ナビゲーションペインで、設定、接続を選択します。 3. 選択したソースプロバイダーのデベロッパーツールコンソールドキュメントの指示に従ってください。 後のステップで CodePipeline の (IAM) ポリシーを作成するときに、接続に Amazon リソースネーム AWS Identity and Access Management (ARN) が必要になります。	AWS DevOps

アクセス許可の設定

タスク	説明	必要なスキル
CodeBuild の IAM ポリシーを作成します。	CodeBuild プロジェクトでは、Secrets Manager や Amazon S3 などの一部のリソースにアクセスする必要があります。 必要なアクセス権限を設定するには:	AWS 全般

タスク	説明	必要なスキル
	1. [IAM コンソール] を開きます。 2. ナビゲーションペインで、ポリシー、ポリシーの作成を選択し、CodeBuild サービスを選択します。 3. フォーマットを Visual から JSON に切り替え、追加情報セクションで提供されている CodeBuild ポリシーをポリシーエディタフィールドにコピーします。 4. このポリシーに名前を付け、次のステップで後で参照できるように保存します。 IAM ポリシーの作成の詳細については、IAM ドキュメントを参照してください。	

タスク	説明	必要なスキル
CodeBuild の IAM ロールを作成します。	CodeBuild でセキュリティポリシーを使用できるようにするには、IAM ロールを設定する必要があります。 このロールを作成するには： 1. IAM コンソールのナビゲーションペインで、ロール、ロールの作成を選択します。 3. 信頼されたエンティティタイプの場合、AWS のサービスデフォルト設定のままにします。 4. ユースケースでは、CodeBuild サービスを選択し、次へを選択します。 4. 使用可能な IAM ポリシーのリストで、CodeBuild 用に作成したポリシーを見つけ、次へ を選択してロールにアタッチします。 5. ロールの名前を指定し、ロールの作成を選択して CodeBuild で後で参照できるように保存します。 の IAM ロールの作成の詳細については AWS のサービス、IAM ドキュメントを参照してください。	AWS 全般

タスク	説明	必要なスキル
CodePipeline の IAM ポリシーを作成します。	AWS CodePipeline パイプラインでは、コードリポジトリや Amazon S3 など、一部のリソースにアクセスする必要があります。 CodeBuild に対して前述の手順を繰り返して、CodePipeline の IAM ポリシーを作成します (ステップ 2 では、CodeBuild の代わりに CodePipeline CodePipeline を選択します)。	AWS DevOps

タスク	説明	必要なスキル

CodePipeline の IAM ロールを作成します。	CodePipeline でセキュリティポリシーを使用できるようにするには、IAM ロールを設定する必要があります。
	
	このロールを作成するには：
	
	1. IAM コンソールで、ロール、ロールの作成を選択します。
	2. [Trusted entity type] (信頼されたエンティティのタイプ) で、[Custom trust policy] (カスタム信頼ポリシー) を選択します。
	
	空の `Principal` 要素を持つポリシーが表示されます。
	3. `Principal` 線の中括弧の間に以下を追加します。
	
	`"Service": "codepipeline.amazonaws.com"`
	
	信頼ポリシーは次のようになります。
	
	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "Statement1", "Effect": "Allow",`
		AWS DevOps

タスク	説明	必要なスキル

	<pre>"Principal": {
	"Service":
	"codepipeline.amazonaws.com"
	},
	"Action":
	"sts:AssumeRole"
	}
	]
	}</pre>
	4. [Next (次へ)] を選択します。
	5. 使用可能な IAM ポリシーのリストで、CodePipeline 用に作成したポリシーを見つけ、次へ を選択してロールにアタッチします。
	6. ロールの名前を指定し、ロールの作成 を選択して、後で CodePipeline で参照できるように保存します。	

COBOL Db2 プログラムをコンパイルしてバインドする

タスク	説明	必要なスキル
CodePipeline パイプラインと CodeBuild プロジェクトを作成します。	COBOL Db2 プログラムをコンパイルしてバインドする CodePipeline パイプラインと CodeBuild プロジェクトを作成するには： 1. CodePipeline コンソールを開き、パイプラインの作成	AWS DevOps

タスク	説明	必要なスキル
	、カスタムパイプラインの構築を選択します。 2. パイプラインの名前を指定します。 3. サービスロールで、既存のサービスロールを選択し、CodePipeline 用に作成した IAM ロールの ARN を指定します。 4. 詳細設定を展開し、カスタムロケーションを選択し、以前に作成した S3 バケットを選択し、次へを選択します。 5. ソースプロバイダーで、サードパーティーのソースプロバイダーを選択し、プロバイダーの関連情報を入力します。 a. Connection で、ソースプロバイダー用に作成された接続を選択します。 b. リポジトリ名で、リポジトリを選択します。 c. デフォルトブランチで、COBOL プログラムと を保存するブランチを選択しますbuildspec.yml 。 d. [Next (次へ)] を選択します。 6. ビルドプロバイダー で、その他のビルドプロバイ	

タスク	説明	必要なスキル
	ダー を選択しますAWS CodeBuild。 7. プロジェクト名 で、プロジェクトの作成 を選択します。 コンソールには、ビルドプロジェクトを作成できるCodeBuild ウィンドウが表示されます。このウィンドウでは、次の操作を行います。 a. プロジェクトの名前を入力します。 b. [環境イメージ] で、[カスタムイメージ] を選択します。 c. 環境タイプで、Linux コンテナを選択します。 d. ECR アカウントで、その他の ECR アカウントを選択します。 e. Amazon ECR リポジトリ URI には、 と入力します`673918848628.dkr.ecr.<your-region>.amazonaws.com/m2-enterprise-build-tools:8.0.9.R1` 。 f. サービスロールで、既存のサービスロールを選択	

タスク	説明	必要なスキル
	し、CodeBuild 用に作成したロールを選択します。 g. 追加設定セクションを展開し、このプロジェクトの VPC、プライベートサブネット、セキュリティグループを選択します。 h. [Buildspec] セクションで [buildspec ファイルを使用する] を選択します。 i. ウィンドウの最後に、CodePipeline に進む」を選択します。CodeBuild ウィンドウが閉じて、CodePipeline コンソールに戻ることができます。 8. CodePipeline コンソールに戻り、次へを選択します。 9. デプロイステージの追加ページで、デプロイステージをスキップを選択して確認します。 10パイプラインパラメータを確認し、パイプラインの作成を選択します。	
出力を確認します。	CodePipeline ビルドログを確認して、ビルドの成功を確認します。	AWS DevOps

タスク	説明	必要なスキル

Db2 で結果を確認します。	SYSPLAN テーブルのパッケージバージョンを確認します。
	
	`select CAST(NAME AS VARCHAR(10)) as name, VALIDATE, LAST_BIND_TIME, LASTUSED, CAST(PKGVERSION AS VARCHAR(10)) as PKGVERSION from SYSIBM.SYSPLAN where NAME = 'CDB2SMP' order by LAST_BIND_TIME desc`
	
	バージョンは、CDB2SMPCodeBuild ビルド ID と一致する必要があります。この例は、次のとおりです。
	
	`NAME VALIDATE LAST_BIND_TIME LASTUSED PKGVERSION ---------- -------- ---------- ---------- -------- ---------- CDB2SMP B 2024-05-18-11.53.11.503738 01/01/0001 19`	

トラブルシューティング

問題	ソリューション
サービス間を移動するときに、 AWS コンソールでリージョンが切り替わることがあります。	サービスを切り替える AWS リージョン ときは、必ず選択した を確認してください。 AWS リージョン セレクターは、コンソールウィンドウの右上隅にあります。
CodeBuild から Db2 接続の問題を特定するのは難しい場合があります。	接続の問題をトラブルシューティングするには、次の DB2 connect コマンドを `buildspec.yml` ファイルに追加します。この追加により、接続の問題をデバッグして解決できます。 `db2 connect to $DB_NAME user $DB2USER using $DB2PASS`
IAM コンソールのロールページに、作成した IAM ポリシーがすぐに表示されない場合があります。	遅延が発生した場合は、画面を更新して最新情報を表示します。

関連リソース

IBM ドキュメント

- IBM Data Server クライアントとドライバータイプ
- IBM Data Server クライアントとドライバータイプをダウンロードする

AWS ドキュメント

- Amazon S3 ユーザーガイド
- AWS CodeBuild ユーザーガイド
- AWS Mainframe Modernization ユーザーガイド
- AWS Secrets Manager ユーザーガイド
- AWS CodePipeline ユーザーガイド

- AWS CodeDeploy ユーザーガイド

追加情報

CodeBuild ポリシー

プレースホルダー <RegionID>、<AccountID>、、<BucketARN>、 <SubnetARN>を自分の値<DB2CredSecretARN>に置き換えます。

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": "ecr:GetAuthorizationToken", "Effect": "Allow", "Resource": "*" },
        {"Action": ["ecr:GetDownloadUrlForLayer", "ecr:BatchGetImage",
                    "ecr:BatchCheckLayerAvailability"],
         "Effect": "Allow",
         "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"},
        {"Action": "s3:PutObject", "Effect": "Allow", "Resource": "arn:aws:s3:::aws-m2-
repo-*/*"},
        {"Action": ["logs:PutLogEvents", "logs:CreateLogStream",
 "logs:CreateLogGroup"],
         "Effect": "Allow", "Resource": "arn:aws:logs:<RegionId>:<AccountId>:*"},
        {"Action": ["ec2:DescribeVpcs", "ec2:DescribeSubnets",
                    "ec2:DescribeSecurityGroups", "ec2:DescribeNetworkInterfaces",
                    "ec2:DescribeDhcpOptions", "ec2:DeleteNetworkInterface",
                    "ec2:CreateNetworkInterface"],
         "Effect": "Allow", "Resource": "*"},
        {"Action": "ec2:CreateNetworkInterfacePermission",
         "Effect": "Allow", "Resource": ["<SubnetARN>"]},
        {"Action": "s3:*", "Effect": "Allow", "Resource": ["<BucketARN>/
*","<BucketARN>"]},
        {"Action": "secretsmanager:GetSecretValue",
         "Effect": "Allow", "Resource": "<DB2CredSecretARN>"}
    ]
}
```

CodePipeline ポリシー

プレースホルダー <BucketARN>と を自分の値<ConnectionARN>に置き換えます。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {"Action": ["s3:List*", "s3:GetObjectVersion", "s3:GetObject",
 "s3:GetBucketVersioning" ],
        "Effect": "Allow",
        "Resource": ["<BucketARN>/*", "<BucketARN>"]},
        {"Action": ["codebuild:StartBuild", "codebuild:BatchGetBuilds"],
         "Effect": "Allow", "Resource": "*"},
        {"Action": ["codestar-connections:UseConnection"],
         "Effect": "Allow", "Resource": "<ConnectionARN>"}
        ]
}
```

buildspec.yml

<your-bucket-name> プレースホルダーを実際の S3 バケット名に置き換えます。

```
version: 0.2
phases:
  pre_build:
    commands:
      - /var/microfocuslicensing/bin/mfcesd -no > /var/microfocuslicensing/logs/
mfcesd_startup.log 2>&1 &
      - |
        mkdir $CODEBUILD_SRC_DIR/db2client
        aws s3 cp s3://<your-bucket-name>/v11.5.8_linuxx64_client.tar.gz
 $CODEBUILD_SRC_DIR/db2client/ >> /dev/null 2>&1
        tar -xf $CODEBUILD_SRC_DIR/db2client/v11.5.8_linuxx64_client.tar.gz -C
 $CODEBUILD_SRC_DIR/db2client/
        cd $CODEBUILD_SRC_DIR/db2client/
        ./client/db2_install -f sysreq -y -b /opt/ibm/db2/V11.5 >> /dev/null 2>&1

        useradd db2cli
        /opt/ibm/db2/V11.5/instance/db2icrt -s client -u db2cli db2cli
        DB2CRED=$(aws secretsmanager get-secret-value --secret-id dev-db2-cred | jq -r
 '.SecretString | fromjson')
        read -r DB2USER DB2PASS DB_NODE DB_HOST DB_PORT DB_NAME DB_QUAL <<<$(echo
 $DB2CRED | jq -r
 '.username, .password, .db2node, .db2host, .db2port, .db2name, .qualifier')
        . /home/db2cli/sqllib/db2profile
        db2 catalog tcpip node $DB_NODE remote $DB_HOST server $DB_PORT
        db2 catalog db $DB_NAME as $DB_NAME at node $DB_NODE authentication server
  build:
    commands:
      - |
```

```
        revision=$CODEBUILD_SRC_DIR/loadlib
        mkdir -p $revision; cd $revision
        . /opt/microfocus/EnterpriseDeveloper/bin/cobsetenv
        cob -zU $CODEBUILD_SRC_DIR/CDB2SMP.cbl -C "DB2(DB==${DB_NAME} PASS==${DB2USER}.
${DB2PASS} VERSION==${CODEBUILD_BUILD_NUMBER} COLLECTION==DB2AWSDB"
artifacts:
  files:
    - "**/*"
  base-directory: $revision
```

Amazon EC2 Auto Scaling と Systems Manager を搭載した Micro Focus Enterprise Server PAC を構築する

作成者: Kevin Yung (AWS)、Peter Woods、Abraham Rondon (Micro Focus)、Krithika Palani Selvam (AWS)

概要

このパターンでは、スケールアウトパフォーマンスおよびアベイラビリティークラスター (PAC) で Micro Focus Enterprise Server を使用し、Amazon Web Services () で Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling グループを使用するメインフレームアプリケーション用のスケーラブルなアーキテクチャが導入されていますAWS。このソリューションは、 AWS Systems Manager および Amazon EC2 Auto Scaling ライフサイクルフックで完全に自動化されています。このパターンを使用すると、メインフレームのオンラインアプリケーションとバッチアプリケーションをセットアップして、キャパシティの需要に応じた自動的なスケールインとスケールアウトを行い、高い回復性を実現できます。

> **Note**
>
> このパターンは、Micro Focus Enterprise Server バージョン 6.0 でテストされています。バージョン 8 については、「Micro Focus Runtime のセットアップ (Amazon EC2 上)」を参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Micro Focus Enterprise Server のソフトウェアとライセンス。詳細については、Micro Focus の営業担当にお問い合わせください。
- Micro Focus Enterprise Server で実行するメインフレームアプリケーションの再構築と配信の概念について理解している。大まかな概要については、「Micro Focus Enterprise Server データシート」を参照してください。
- Micro Focus Enterprise Serverのスケールアウトパフォーマンスおよび可用性クラスターの概念について理解している。詳細については、「Micro Focus Enterprise Server ドキュメント」を参照してください。

- 継続的インテグレーション (CI) によるメインフレームアプリケーション DevOps の全体的な概念について理解している。 AWSと Micro Focus によって開発された AWS 規範ガイダンスパターンについては、「メインフレームのモダナイゼーション: Micro Focus AWS を使用した での DevOps」を参照してください。

> **Note**
>
> このパターンは、Micro Focus Enterprise Server バージョン 6 でテストされています。バージョン 8 については、「Micro Focus Runtime のセットアップ (Amazon EC2 上)」を参照してください。

機能制限

- Micro Focus Enterprise Server でサポートされているプラットフォームのリストについては、「Micro Focus Enterprise Serverのデータシート」を参照してください。
- このパターンで使用されるスクリプトとテストは Amazon EC2 Windows Server 2019 に基づいています。他の Windows Server バージョンとオペレーティングシステムでは、このパターンではテストされていません。
- このパターンは Micro Focus Enterprise Server 6.0 for Windows に基づいています。このパターンの開発では、それ以前またはそれ以降のリリースではテストされていません。

製品バージョン

- Micro Focus Enterprise Server 6.0
- [Windows Server 2019]

アーキテクチャ

従来のメインフレーム環境では、アプリケーションと企業データをホストするハードウェアをプロビジョニングする必要があります。季節、月次、四半期、さらには前例のない、または予期しない需要の急増に対応するため、メインフレームユーザーは、ストレージとコンピューティングの容量を追加購入してスケールアウトする必要があります。ストレージとコンピューティングキャパシティのリソースを増やすと全体的なパフォーマンスは向上しますが、スケーリングは線形になりません

Amazon EC2 Auto Scaling と Micro Focus Enterprise Servers を使用して AWS でオンデマンド消費モデルを採用し始めた場合は異なります。以下のセクションでは、Amazon EC2 Auto Scaling グループで Micro Focus Enterprise Server スケールアウトパフォーマンスおよび可用性クラスター (PAC) を使用して、完全に自動スケーリングされたメインフレームアプリケーションのアーキテクチャを構築する方法について詳しく説明します。

Micro Focus Enterprise Server の自動スケーリングアーキテクチャ

最初に Micro Focus Enterprise Server の基本概念を理解することが重要です。従来 IBM メインフレーム上で実行されていたアプリケーションに、メインフレーム互換の x86 デプロイメント環境を提供します。オンラインおよびバッチ実行と、以下の機能をサポートするトランザクション環境を提供します。

- IBM COBOL
- IBM PL/I
- IBM JCL バッチジョブ
- IBM CICS と IMS TM トランザクション
- ウェブサービス
- SORT を含む一般的なバッチユーティリティ

Micro Focus Enterprise Server を使用すると、メインフレームアプリケーションで最小限の変更を実行できます。既存のメインフレームワークロードを x86 プラットフォームに移行してモダナイズし、AWS クラウドネイティブエクステンションを活用して新規市場や地域への迅速な拡大を図ることができます。

AWS 規範ガイダンスパターン「メインフレームのモダナイゼーション: DevOps on AWS with Micro Focus」では、AWS CodePipeline と AWS CodeBuild を搭載した Micro Focus Enterprise Developer と Enterprise Test Server を使用して、AWS でのメインフレームアプリケーションの開発とテストを加速するアーキテクチャを導入しました。このパターンでは、メインフレームアプリケーションを AWS 実稼働環境にデプロイし、高い可用性と回復性を実現することに重点を置いています。

メインフレームの実稼働環境では、メインフレームに IBM Parallel Sysplex を設定して、高パフォーマンスと高可用性を実現している場合があります。Sysplex と同様のスケールアウト アーキテクチャを構築するために、Micro Focus はパフォーマンス/可用性クラスター (PAC) を Enterprise Server に導入しました。PAC は、単一イメージとして管理され、Amazon EC2 インスタンスでスケールアウトされた複数の Enterprise Server リージョンへのメインフレームアプリケーションのデ

プロイをサポートします。PAC は、予測可能なアプリケーションパフォーマンスとオンデマンドのシステムスループットもサポートします。

PAC では、複数の Enterprise Server インスタンスが連携して 1 つの論理エンティティとして機能します。　そのため、1 つの Enterprise Server インスタンスに障害が発生しても、容量が他のリージョンと共有され、新規インスタンスは Amazon EC2 Auto Scaling グループなどの業界標準機能を使用して自動的に起動され、事業継続性が中断されることはありません。これにより、単一障害点が解消され、ハードウェア、ネットワーク、アプリケーションの問題に対する耐性が向上します。　スケールアウトしたエンタープライズサーバーインスタンスを運用および管理するために、Enterprise Server Common Web Administration (ESCWA) API を使用することで、Enterprise Server の運用メンテナンスおよび保守性を簡素化することができます。

> **Note**
>
> Micro Focus では、Enterprise Server リージョンに障害が発生した場合やメンテナンスが必要な場合に可用性が損なわれないように、パフォーマンスクラスターと可用性クラスター (PAC) を少なくとも 3 つの Enterprise Server リージョンで構成することをお勧めします。

PAC 構成でリージョンデータベース、クロスリージョンデータベース、およびオプションのデータストアデータベースを管理するには、サポートされているリレーショナルデータベース管理サービス (RDBMS) が必要です。Virtual Storage Access Method (VSAM) ファイルの管理にはデータストアデータベースを使用し、可用性とスケーラビリティを向上させるには Micro Focus データベースファイルハンドラサポートを使用する必要があります。サポートされる RDBMS には、以下のものが含まれます。

- Microsoft SQL Server 2009 R2 以降
- PostgreSQL 10.x (Amazon Aurora PostgreSQL 互換エディションを含む)
- DB2 10.4 以降

サポートされている RDBMS および PAC の要件の詳細は、「Micro Focus Enterpriseサーバー - 前提条件」および「Micro Focus Enterprise Server - 推奨される PAC 構成」を参照してください。

次の図は、Micro Focus PAC の一般的な AWS アーキテクチャセットアップを示しています。

コンポーネント	説明

1	Enterprise Server インスタンス、自動スケーリンググループ	PAC の Enterprise Server インスタンスと一緒にデプロイされる自動スケーリンググループを設定します。 インスタンスの数は、Amazon CloudWatch アラームを使用して Amazon CloudWatch アラートを介して外部に拡張したり、内部に起動したりすることができます。
2	Enterprise Server ESCWA インスタンス、自動スケーリンググループ	Enterprise Server Common Web Administration (ESCWA) でデプロイされた自動スケーリンググループを設定します。 ESCWA はクラスター管理 API を提供します。 ESCWA 経由のソーシャルサーバーは、Enterprise Server インスタンスの自動スケーリングイベント中に Enterprise Server サーバーを追加または削除したり、PAC 内のエンタープライズサーバー領域を起動または停止したりするコントロールプレーンとして機能します。 ESCWA インスタンスは PAC 管理にのみ使用されるため、トラフィックパターンは予測可能で、自動スケーリングに必要な容量要件は 1 に設定できます。

3	マルチ AZ セットアップでの Amazon Aurora インスタンス	Enterprise Server インスタンス間で共有されるユーザーデータファイルとシステムデータファイルの両方をホストするリレーショナルデータベース管理システム (RDBMS) を設定します。
4	Amazon ElastiCache (Redis OSS) インスタンスとレプリカ	ElastiCache (Redis OSS) プライマリインスタンスと少なくとも 1 つのレプリカをセットアップしてユーザーデータをホストし、Enterprise Server インスタンスのスケールアウトリポジトリ (SOR) として機能します。特定タイプのユーザーデータを保存するスケールアウトリポジトリを 1 つ以上設定できます。 Enterprise Server は、PAC の整合性を維持するための要件である SOR として Redis NoSQL データベースを使用します。
5	Network Load Balancer	ロードバランサーを設定して、アプリケーションのホスト名を指定し、Enterprise Server インスタンスによって提供されるサービスに接続します (たとえば、3270 エミュレータを使用してアプリケーションにアクセスします)。

これらのコンポーネントは、Micro Focus Enterprise Server PAC クラスターの最小要件となります。次のセクションでは、クラスター管理の自動化について説明します。

スケーリング用の AWS Systems Manager Automation を使用する

PAC クラスターが AWS にデプロイされると、PAC は Enterprise Server Common Web Administration (ESCWA) で管理されます。

自動スケーリングイベント中のクラスター管理タスクを自動化するには、Systems Manager Automation ランブックと Amazon EventBridge を搭載した Amazon EC2 Auto Scaling を使用できます。以下の図は、これらの自動化のアーキテクチャを示しています。

	コンポーネント	説明
1	Auto Scaling ライフサイクルフック	Auto Scaling ライフサイクルフックを設定し、自動スケーリンググループ内の新規インスタンスの開始時および既存インスタンスの終了時に、Amazon EventBridge に通知を送信します。
2	Amazon EventBridge	Amazon EventBridge ルールを設定して、自動スケーリングイベントを Systems Manager Automation ランブックターゲットにルーティングします。
3	Automation ランブック	Systems Manager Automation ランブックを設定して Windows PowerShell スクリプトを実行し、ESCWA API を呼び出して PAC を管理します。例については、「追加情報」セクションを参照してください。
4	自動スケーリンググループの Enterprise Server ESCWA インスタンス	自動スケーリンググループで、Enterprise Server ESCWA インスタンスを設定します。ESCWA インスタンスに

は、PAC を管理するための API が用意されています。

ツール

- Micro Focus Enterprise Server – Micro Focus Enterprise Server は、Enterprise Developer のあらゆる統合開発環境 (IDE) バリアントで作成されたアプリケーションの実行環境を提供します。
- Amazon EC2 Auto Scaling – Amazon EC2 Auto Scaling は、アプリケーションの負荷を処理するために適切な数の Amazon EC2 インスタンスを利用できるようにします。Auto Scaling グループと呼ばれる EC2 インスタンスの集合を作成し、インスタンスの最小数と最大数を指定します。
- Amazon ElastiCache (Redis OSS) – Amazon ElastiCache は、クラウド内の分散インメモリデータストアまたはキャッシュ環境をセットアップ、管理、スケーリングするためのウェブサービスです。高性能かつスケーラブルで費用対効果の高いキャッシュソリューションを提供します。
- 「Amazon RDS」— Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケールをより簡単にするウェブサービスです。リレーショナルデータベース向けに、コスト効率に優れ、サイズ変更可能な容量を提供し、一般的なデータベース管理タスクを管理します。
- AWS Systems Manager – AWS Systems Manager は、インフラストラクチャを表示およびコントロールするために使用できる AWS サービスです。Systems Manager コンソールを使用すると、複数の AWS サービスからの運用データを表示し、AWS リソース全体の運用タスクを自動化できます。Systems Manager は、マネージドインスタンスをスキャンし、検出されたポリシー違反を報告 (または是正措置を講じる) して、セキュリティとコンプライアンスを維持することができます。

エピック

Amazon Aurora インスタンスを作成する

タスク	説明	必要なスキル
Amazon Aurora インスタンス用の AWS CloudFormation テンプレートを作成します。	AWS のサンプルコードスニペットを使用して、Amazon Aurora PostgreSQL 互換エディションインスタンスを作	クラウドアーキテクト

タスク	説明	必要なスキル
	成する CloudFormation テンプレートを作成します。	
CloudFormation スタックをデプロイして Amazon Aurora インスタンスを作成します。	CloudFormation テンプレートを使用して、実稼働環境のワークロードでマルチ AZ レプリケーションが有効になっている Aurora PostgreSQL 互換インスタンスを作成します。	クラウドアーキテクト
Enterprise Server のデータベース接続を設定します。	「Micro Focus ドキュメント」の指示に従って、Micro Focus Enterprise Server の接続文字列とデータベース設定を準備します。	データエンジニア、DevOps エンジニア

Redis インスタンス用の Amazon ElastiCache クラスターを作成します。

タスク	説明	必要なスキル
Redis インスタンス用の Amazon ElastiCache クラスター向けに CloudFormation テンプレートを作成します。	AWS のサンプルコードスニペットを使用して、Redis インスタンス用の Amazon ElastiCache クラスターを作成する CloudFormation テンプレートを作成します。	クラウドアーキテクト
Redis インスタンス用の Amazon ElastiCache クラスター向けに CloudFormation テンプレートを作成します。	実稼働ワークロードでマルチ AZ レプリケーションが有効になっている Redis インスタンス用の Amazon ElastiCache クラスターを作成します。	クラウドアーキテクト

タスク	説明	必要なスキル
Enterprise Server PSOR 接続を設定します。	「Micro Focus ドキュメント」の指示に従って、PAC Scale-Out Repository (PSOR) の接続文字列とデータベース設定を準備します。	DevOps エンジニア

Micro Focus Enterprise Server ESCWA 自動スケーリンググループの作成

タスク	説明	必要なスキル
Micro Focus Enterprise Server AMI を作成します。	Amazon EC2 Windows Server インスタンスを作成し、その EC2 インスタンスに Micro Focus Enterprise Server バイナリをインストールします。 EC2 インスタンスの Amazon マシンイメージ (AMI) を作成します。詳細については、「Enterprise Server ドキュメント」を参照してください。	クラウドアーキテクト
Enterprise Server ESCWA 用の CloudFormation テンプレートを作成します。	「AWS サンプルコードスニペット」を使用して、自動スケーリンググループ内に Enterprise Server ESCWA のカスタムスタックを作成するためのテンプレートを作成します。	クラウドアーキテクト
CloudFormation スタックをデプロイして、Enterprise Server ESCWA 用の Amazon EC2 スケーリンググループを作成します。	CloudFormation テンプレートを使用して、前のストーリーで作成した Micro Focus Enterprise Server ESCWA AMI を含む自動スケーリング	クラウドアーキテクト

タスク	説明	必要なスキル
	グループをデプロイします。	

AWS Systems Manager Automation ランブックを作成する

タスク	説明	必要なスキル
Systems Manager Manager Automation ランブック用の CloudFormation テンプレートを作成します。	「追加情報」セクションのサンプルコードスニペットを使用して、PAC 作成、Enterprise Server のスケールイン、Enterprise Server のスケールアウトを自動化するための Systems Manager Automation ランブックを作成する CloudFormation テンプレートを作成します。	クラウドアーキテクト
Systems Manager Automation ランブックを含む CloudFormation スタックをデプロイします。	CloudFormation テンプレートを使用して、PAC 作成、Enterprise Server のスケールイン、Enterprise Server のスケールアウト用の Automation ランブックを含むスタックをデプロイします。	クラウドアーキテクト

Micro Focus Enterprise Server の自動スケーリンググループを作成する

タスク	説明	必要なスキル
Micro Focus Enterprise Server の自動スケーリンググループを設定するための CloudForm	AWS のサンプルコードスニペットを使用して、自動スケーリンググループを作成する CloudFormation テンプ	クラウドアーキテクト

タスク	説明	必要なスキル
ation テンプレートを作成します。	レートを作成します。このテンプレートは、Micro Focus Enterprise Server ESCWA インスタンス用に作成された AMI を再利用します。 次に、AWS サンプルコードスニペットを使用して自動スケーリングライフサイクルイベントを作成し、同じ CloudFormation テンプレートでスケールアウトイベントとスケールインイベントをフィルタリングするように Amazon EventBridge を設定します。	
Micro Focus Enterprise Servers の自動スケーリンググループ用の CloudFormation スタックをデプロイします。	Micro Focus Enterprise Servers の自動スケーリンググループ用の CloudFormation スタックをデプロイします。	クラウドアーキテクト

関連リソース

- 「Micro Focus Enterprise Server パフォーマンス/可用性クラスター (PAC)」
- 「Amazon EC2 Auto Scaling のライフサイクルフック」
- 「EventBridge を使用したトリガーによるオートメーションの実行」

追加情報

PAC クラスターをスケールインまたはスケールアウトするには、以下のシナリオを自動化する必要があります。

PAC の開始または再作成を自動化する

PAC クラスターの開始時に、Enterprise Server ESCWA は ESCWA に API を呼び出して PAC 構成を作成するように要求します。これにより、Enterprise Server リージョンが起動し、PAC に追加されます。　PAC を作成または再作成するには、次の手順に従います：

1. ESCWA で PAC スケールアウトリポジトリ (PSOR) を指定した名前で設定します。

```
POST /server/v1/config/groups/sors
```

2. 指定した名前で PAC を作成し、PSOR をアタッチします。

```
POST /server/v1/config/groups/pacs
```

3. PAC の初回設定時は、リージョンデータベースとクロスリージョンデータベースを設定します。

> **Note**
>
> このステップでは、SQL クエリと Micro Focus Enterprise Suite コマンドライン dbhfhadmin ツールを使用してデータベースを作成し、初期データをインポートします。

4. PAC の定義を Enterprise Server リージョンにインストールします。

```
POST /server/v1/config/mfds
POST /native/v1/config/groups/pacs/${pac_uid}/install
```

5. PAC 内の Enterprise Server リージョンを起動します。

```
POST /native/v1/regions/${host_ip}/${port}/${region_name}/start
```

これまでの手順は、Windows PowerShell スクリプトを使用して実装できます。

以下の手順では、Windows PowerShell スクリプトを再利用して PAC を作成するための自動化を設定する方法について説明します。

1. Amazon EC2 起動テンプレートを作成し、ブートプロセスの一部として Windows PowerShell スクリプトをダウンロードまたは作成します。例えば、EC2 ユーザーデータを使用して、Amazon Simple Storage Service (Amazon S3) バケットからスクリプトをダウンロードできます。
2. Windows PowerShell スクリプトを呼び出す AWS Systems Manager Automation ランブックを作成します。

3. インスタンスタグを使用して、ランブックを ESCWA インスタンスに関連付けます。

4. 起動テンプレートを使用して ESCWA 自動スケーリンググループを作成します。

次の AWS CloudFormation スニペットの例を使用して、Automation ランブックを作成できます。

PAC の作成に使用される Systems Manager Automation ランブックの CloudFormation スニペットの例

```
PACInitDocument:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Command
     Content:
       schemaVersion: '2.2'
       description: Operation Runbook to create Enterprise Server PAC
       mainSteps:
       - action: aws:runPowerShellScript
         name: CreatePAC
         inputs:
           onFailure: Abort
           timeoutSeconds: "1200"
           runCommand:
           - |
             C:\Scripts\PAC-Init.ps1
PacInitAutomation:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Automation
     Content:
       description: Prepare Micro Focus PAC Cluster via ESCWA Server
       schemaVersion: '0.3'
       assumeRole: !GetAtt SsmAssumeRole.Arn
       mainSteps:
         - name: RunPACInitDocument
           action: aws:runCommand
           timeoutSeconds: 300
           onFailure: Abort
           inputs:
             DocumentName: !Ref PACInitDocument
             Targets:
               - Key: tag:Enterprise Server - ESCWA
                 Values:
```

```
                    - "true"
  PacInitDocumentAssociation:
    Type: AWS::SSM::Association
    Properties:
      DocumentVersion: "$LATEST"
      Name: !Ref PACInitDocument
      Targets:
        - Key: tag:Enterprise Server - ESCWA
          Values:
            - "true"
```

詳細については、「Micro Focus Enterprise Server- PAC の設定」を参照してください。

新しい Enterprise Server インスタンスによるスケールアウトの自動化

Enterprise Server インスタンスをスケールアウトする場合、その Enterprise Server リージョンを PAC に追加する必要があります。　以下の手順では、ESCWA API を呼び出して、Enterprise Server リージョンを PAC に追加する方法について説明します。

1. PAC の定義を Enterprise Server リージョンにインストールします。

```
POST '/server/v1/config/mfds'
POST /native/v1/config/groups/pacs/${pac_uid}/install
```

2. PAC 内のリージョンをウォームスタートします。

```
POST /native/v1/regions/${host_ip}/${port}/${region_name}/start
```

3. 自動スケーリンググループをロードバランサーに関連付けて、Enterprise Server インスタンスをロードバランサーに追加します。

これまでの手順は、Windows PowerShell スクリプトを使用して実装できます。　詳細については、「Micro Focus Enterprise Server- PAC の設定」を参照してください。

次の手順は、Windows PowerShell スクリプトを再利用して、新しく起動した Enterprise Server インスタンスを PAC に追加するイベント駆動型の自動化を構築するために使用します。

1. ブートストラップ時に Enterprise Server リージョンをプロビジョニングする Enterprise Server インスタンス用の Amazon EC2 起動テンプレートを作成します。たとえば、Micro Focus Enterprise Server のコマンド mfds を使用してリージョン設定をインポートできます。このコマンドのさらなる詳細とオプションについては、「Enterprise Server リファレンス」を参照してください。

2. 前の手順で作成した起動テンプレートを使用する Enterprise Server 自動スケーリンググループを作成します。
3. Windows PowerShell スクリプトを呼び出す Systems Manager Automation ランブックを作成します。
4. インスタンスタグを使用して、ランブックを ESCWA インスタンスに関連付けます。
5. Enterprise Server 自動スケーリンググループの EC2 インスタンスの起動成功イベントをフィルタリングする Amazon EventBridge ルールを作成し、Automation ランブックを使用するターゲットを作成します。

次の CloudFormation スニペットの例を使用して、Automation ランブックと EventBridge ルールを作成できます。

Enterprise Server インスタンスのスケールアウトに使用される Systems Manager 用の CloudFormation スニペットの例

```
ScaleOutDocument:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Command
     Content:
       schemaVersion: '2.2'
       description: Operation Runbook to Adding MFDS Server into an existing PAC
       parameters:
         MfdsPort:
           type: String
         InstanceIpAddress:
           type: String
           default: "Not-Available"
         InstanceId:
           type: String
           default: "Not-Available"
       mainSteps:
       - action: aws:runPowerShellScript
         name: Add_MFDS
         inputs:
           onFailure: Abort
           timeoutSeconds: "300"
           runCommand:
           - |
             $ip = "{{InstanceIpAddress}}"
```

```
              if ( ${ip} -eq "Not-Available" ) {
                $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output
text --query "Reservations[0].Instances[0].PrivateIpAddress"
              }
              C:\Scripts\Scale-Out.ps1 -host_ip ${ip} -port {{MfdsPort}}

  PacScaleOutAutomation:
    Type: AWS::SSM::Document
    Properties:
      DocumentType: Automation
      Content:
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        description: Scale Out 1 New Server in Micro Focus PAC Cluster via ESCWA
Server
        schemaVersion: '0.3'
        assumeRole: !GetAtt SsmAssumeRole.Arn
        mainSteps:
          - name: RunScaleOutCommand
            action: aws:runCommand
            timeoutSeconds: 300
            onFailure: Abort
            inputs:
              DocumentName: !Ref ScaleOutDocument
              Parameters:
                InstanceIpAddress: "{{InstanceIpAddress}}"
                InstanceId: "{{InstanceId}}"
                MfdsPort: "{{MfdsPort}}"
              Targets:
                - Key: tag:Enterprise Server - ESCWA
                  Values:
                    - "true"
```

新しい Enterprise Server インスタンスによるスケールアウトの自動化

スケールアウトと同様に、Enterprise Server インスタンスをスケールインすると、EC2 インスタンス終了ライフサイクルアクションイベントが開始されます。Micro Focus Enterprise Server インスタンスを PAC から削除するには、以下のプロセスと API 呼び出しが必要になります。

1. 終了する Enterprise Server インスタンスのリージョンを停止します。

```
POST "/native/v1/regions/${host_ip}/${port}/${region_name}/stop"
```

2. Enterprise Server インスタンスを PAC から削除します。

```
DELETE "/server/v1/config/mfds/${uid}"
```

3. Enterprise Server インスタンスの終了を続行するシグナルを送信します。

これまでの手順は、Windows PowerShell スクリプトに実装できます。　このプロセスの詳細については、「Micro Focus Enterprise Server ドキュメント - PAC の管理」を参照してください。

次の手順は、Windows PowerShell スクリプトを再利用して、新しく起動した Enterprise Server インスタンスを PAC から終了する、イベント駆動型の自動化を構築する方法について説明します。

1. Windows PowerShell スクリプトを呼び出す Systems Manager Automation ランブックを作成します。
2. インスタンスタグを使用して、ランブックを ESCWA インスタンスに関連付けます。
3. EC2 インスタンスを終了するための自動スケーリンググループのライフサイクルフックを作成します。
4. Enterprise Server 自動スケーリンググループの EC2 インスタンスのライフサイクルイベントをフィルタリングする Amazon EventBridge ルールを作成し、Automation ランブックを使用するターゲットを作成します。

次の CloudFormation テンプレートの例を使用して、Systems Manager Automation ランブック、ライフサイクルフック、および EventBridge ルールを作成できます。

Enterprise Server インスタンスのスケールインに使用される Systems Manager Automation ランブック用の CloudFormation スニペットの例

```
  ScaleInDocument:
     Type: AWS::SSM::Document
     Properties:
       DocumentType: Command
```

```
      Content:
        schemaVersion: '2.2'
        description: Operation Runbook to Remove MFDS Server from PAC
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        mainSteps:
        - action: aws:runPowerShellScript
          name: Remove_MFDS
          inputs:
            onFailure: Abort
            runCommand:
            - |
              $ip = "{{InstanceIpAddress}}"
              if ( ${ip} -eq "Not-Available" ) {
                $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output
text --query "Reservations[0].Instances[0].PrivateIpAddress"
              }
              C:\Scripts\Scale-In.ps1 -host_ip ${ip} -port {{MfdsPort}}

  PacScaleInAutomation:
    Type: AWS::SSM::Document
    Properties:
      DocumentType: Automation
      Content:
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        description: Scale In 1 New Server in Micro Focus PAC Cluster via ESCWA Server
        schemaVersion: '0.3'
        assumeRole: !GetAtt SsmAssumeRole.Arn
        mainSteps:
```

```
      - name: RunScaleInCommand
        action: aws:runCommand
        timeoutSeconds: "600"
        onFailure: Abort
        inputs:
          DocumentName: !Ref ScaleInDocument
          Parameters:
            InstanceIpAddress: "{{InstanceIpAddress}}"
            MfdsPort: "{{MfdsPort}}"
            InstanceId: "{{InstanceId}}"
          Targets:
            - Key: tag:Enterprise Server - ESCWA
              Values:
                - "true"
      - name: TerminateTheInstance
        action: aws:executeAwsApi
        inputs:
          Service: autoscaling
          Api: CompleteLifecycleAction
          AutoScalingGroupName: !Ref AutoScalingGroup
          InstanceId: "{{ InstanceId }}"
          LifecycleActionResult: CONTINUE
          LifecycleHookName: !Ref ScaleInLifeCycleHook
```

Amazon EC2 自動スケーリングトリガーの自動化

Enterprise Server インスタンスのスケーリングポリシーを設定するプロセスには、アプリケーションの動作を理解する必要があります。ほとんどの場合、ターゲット追跡スケーリングポリシーを設定できます。たとえば、Amazon CloudWatch のメトリクスとして平均 CPU 使用率を使用して、自動スケーリングポリシーを設定できます。詳細については、「Amazon EC2 Auto Scaling のターゲット追跡スケーリング ポリシー」を参照してください。通常のトラフィックパターンを使用するアプリケーションでは、予測的スケーリングポリシーの使用を検討してください。詳細については、「Amazon EC2 Auto Scaling の予測スケーリング」を参照してください。

AWS クラウドで高度なメインフレームファイルビューアを構築

ブーパシー・ゴパルサミー（AWS）とジェレミア・オコナー（AWS）によって作成されました

概要

このパターンでは、手順とコードサンプルを提供して、AWS サーバーレスサービスを使用することでメインフレームファイルを参照および確認するための高度なツールを構築することを支援します。このパターンは、メインフレーム入力ファイルを Amazon OpenSearch Service ドキュメントに変換してブラウズしたり検索したりする方法の例を示しています。ファイルビューアーツールは次の目標達成に役立ちます。

- AWS ターゲット移行環境での一貫性を保持するため、メインフレームのファイル構造とレイアウトを同じにしてください（たとえば、外部にファイルを転送するバッチアプリケーションでも同じファイルレイアウトを維持できます）
- メインフレーム移行中の開発とテストをスピードアップできます。
- 移行後のメンテナンス活動をSupport

前提条件と制限

前提条件

- アクティブなAWS アカウント
- レガシープラットフォームからアクセス可能なサブネットを持つ仮想プライベートクラウド (VPC)
-

> **Note**
>
> 入力ファイルとそれに対応する共通ビジネス指向言語 (COBOL) コピーブック (: 入力ファイルと COBOL コピーブックの例については、GitHub リポジトリのgfs-mainframe-solutions」を参照してください。 COBOL コピーブックの詳細については、IBM ウェブサイトの「Enterprise COBOL for z/OS 6.3 Programming Guide」を参照してください）。

制約事項

- コピーブックの解析は、ネストレベルが 2 つ以下に限られています (OCCURS)。

アーキテクチャ

ソーステクノロジースタック

- [FB (固定ブロック)] 形式の入力ファイル
- COBOL コピーブックのレイアウト

ターゲットテクノロジースタック

- Amazon Athena
- Amazon OpenSearch Service
- Amazon Simple Storage Service (Amazon S3)
- AWS Lambda
- AWS Step Functions

ターゲットアーキテクチャ

次の図は、メインフレーム入力ファイルを解析して OpenSearch Service ドキュメントに変換し、ブラウジングや検索を行うプロセスを示しています。

この図表は、次のワークフローを示しています：

1. 管理者ユーザーまたはアプリケーションは、入力ファイルを 1 つの S3 バケットに、COBOL コピーブックを別の S3 バケットにプッシュします。
2.

> **Note**
>
> 入力ファイルを含む S3 バケットは、サーバーレスの Step Functions ワークフローを開始する Lambda 関数を呼び出します。: このパターンで Step Functions ワークフローを駆動するための S3 イベントトリガーと Lambda 関数の使用はオプションです。このパターンの GitHub コードサンプルにはこれらのサービスの使用は含まれていませんが、要件に応じてこれらのサービスを使用できます。

3. Step Functions ワークフローは、次の Lambda 関数のすべてのバッチプロセスを調整します。
   - この `s3copybookparser.py` 関数はコピーブックのレイアウトを解析し、フィールド属性、データ型、オフセット (入力データ処理に必要) を抽出します。

- この `s3toathena.py` 関数は Athena テーブルレイアウトを作成します。Athena `s3toathena.py` は関数によって処理された入力データを解析し、そのデータを CSV ファイルに変換します。
- この `s3toelasticsearch.py` 関数は S3 バケットから結果ファイルを取り込み、そのファイルを OpenSearch Service にプッシュします。

4. ユーザーは OpenSearch Service を使用して OpenSearch ダッシュボードにアクセスし、さまざまなテーブル形式や列形式のデータを取得し、インデックス化されたデータに対してクエリを実行します。

ツール

AWS サービス

- 「Amazon Athena」 は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析することを支援するインタラクティブなクエリサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。このパターンでは、Lambda を使用して、ファイルの解析、データの変換、OpenSearch Service へのデータのロードなどのコアロジックを実装して、インタラクティブなファイルアクセスが可能になります。
- 「Amazon OpenSearch Service」は、AWS クラウドにおける OpenSearch クラスターのデプロイ、オペレーション、スケーリングを支援するマネージドサービスです。このパターンでは、OpenSearch Service を使用して変換されたファイルにインデックスを付け、インタラクティブな検索機能をユーザーに提供します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- 「AWS Step Functions」 は、Lambda関数と他のサービスを組み合わせてビジネスクリティカルなアプリケーションを構築できるサーバーレスオーケストレーションサービスです。このパターンでは、Step Functions を使用して Lambda 関数をオーケストレーションします。

その他のツール

- 「GitHub」は、コラボレーションツールとバージョン管理を提供するコードホスティングサービスです。
- 「Python」は高水準プログラミング言語です。

コード

このパターンのコードは、GitHub 内の「gfs-mainframe-patterns」リポジトリで利用できます。

エピック

ターゲット環境の準備

タスク	説明	必要なスキル
S3 バケットを作成するには	コピーブック、入力ファイル、出力ファイルを保存するための「S3 バケットを作成します」。S3 バケットには次のフォルダ構造をお勧めします。 • `copybook/` • `input/` • `output/` • `query/` • `results/`	AWS 全般
s3copybook パーサー関数を作成します。	1. 「 という名前の Lambda 関数を作成して」 `s3copybookparser` に呼び出し、「GitHub」リポジトリからソースコード (`s3copybookparser.py` と `copybook.py`) をアップロードします。	AWS 全般

タスク	説明	必要なスキル
	2. Lambda 関数に S3ReadOnly 「IAM ポリシーをアタッチ」します。	
s3toアテナ関数を作成します。	1. という名前の Lambda 関数を作成し s3toathena に呼び出し、「GitHub」リポジトリからソースコード (s3toathena.py) をアップロードします。Lambda タイムアウトを 60 秒以上に設定します。 2. 必要なリソースへのアクセスを提供するには、IAM AmazonAthenaFullAccess ポリシーと S3FullAccess Lambda 関数をアタッチします。	AWS 全般

タスク	説明	必要なスキル
s3toエラスティック検索関数を作成します。	1. **Important** 「Python の依存関係を Lambda 環境に追加します」。: `s3toelasticsearch` 関数を使用するには、Lambda 関数が Python Elasticsearch クライアントの依存関係 (`Elasticsearch==7.9.0` および) を使用するため、Python の依存関係を追加する必要がありますrequests_aws4auth 。 2. という名前の Lambda 関数を作成し `s3toelasticsearch` に呼び出し、「GitHub」 リポジトリからソースコード (`s3toelasticsearch.py`) をアップロードします。 3. Python の依存関係を Lambda レイヤーとしてインポートします。 4. IAM ポリシー `S3ReadOnly` と `AmazonOpenSearchServiceRead`	AWS 全般

タスク	説明	必要なスキル
	`OnlyAccess` を Lambda 関数にアタッチします。	

タスク	説明	必要なスキル
OpenSearch サービスクラスターを作成します。	クラスターを作成します 1. 「OpenSearch Service コネクタを作成」します クラスターを作成する場合、以下の作業を行います。 • **Note** 「OpenSearch ダッシュボード」へのサインインに使用できるクラスターの「マスターユーザーとパスワードを作成します」。: Amazon Cognito で認証を使用する場合、このステップは必要ありません。 • きめ細かなアクセス制御を選択します。これにより、OpenSearch Service のデータへのアクセスをコントロールする追加の方法が提供されます。 2. ドメイン URL をコピーし、環境変数 'HOST' として Lambda 関数 `s3toelasticsearch` に渡します。	AWS 全般

タスク	説明	必要なスキル
	IAM ロールにアクセス許可を付与 Lambda 関数の IAM ロール (`arn:aws:iam::**:role/service-role/s3toelasticsearch-role-**`) にきめ細かくアクセスできるようにするには、以下を実行します。 1. マスターユーザーとして OpenSearch Dashboards にアクセスします。 2. セキュリティタブを選択し、ロール、all_access、マップユーザー、バックエンドロールを選択します。 3. Lambda関数のIAMロールのAmazon リソースネーム (ARN) を追加し、[保存] を選択します。詳細については、OpenSearch Service ドキュメントの 「ユーザーへのロールのマッピング」 を参照してください。	

タスク	説明	必要なスキル
オーケストレーション用の Step Functions を作成します。	1. 標準フローを付けて、「Step Functions ステートマシンを作成」します。定義は「GitHub」リポジトリに含まれています。 2. JSON スクリプトで、Lambda 関数の ARN を、環境内の Lambda 関数の ARN に置き換えます。	AWS 全般

デプロイして実行

タスク	説明	必要なスキル
S3 バケットに入力ファイルとコピーブックをアップロードします。	「GitHub」リポジトリのサンプルフォルダからサンプルファイルをダウンロードし、先ほど作成した S3 バケットにファイルをアップロードします。 1. `<S3_Bucket>/copybook` フォルダに `Mockedcopy.cpy` と `acctix.cpy` をアップロードします。 2. `Modedupdate.txt` と `acctindex.cpy` サンプルの入力ファイルを `<S3_Bucket>/input` フォルダにアップロードします。	AWS 全般

タスク	説明	必要なスキル

手順関数を呼び出します	1. AWS マネジメントコンソールにサインインし、「手順関数コンソール」を開きます。
	2. 左側のナビゲーションで、ステートマシンを選択します。
	3. ステートマシンを選択し、実行開始を選択します。
	4. 入力ボックスに、S3 バケットへの JSON 変数として次のコピーブック/ファイルパスを入力し、[実行開始] を選択します。
	
	`{ "s3_copybook_bucket_name": "<BUCKET NAME>", "s3_copybook_bucket_key": "<COPYBOOK PATH>", "s3_source_bucket_name": "<BUCKET NAME", "s3_source_bucket_key": "INPUT FILE PATH" }`
	
	以下に例を示します。
	
	`{ "s3_copybook_bucket_name": "fileaidtest",`
		AWS 全般

タスク	説明	必要なスキル

	<pre>"s3_copybook_bucket_key": "copybook/acctix.cpy",
	"s3_source_bucket_name": "fileaidtest",
	"s3_source_bucket_key": "input/acctindex"
	}</pre>	

Step Functions でワークフローの実行を検証します。	「Step Functions コンソール」で、グラフインスペクターでワークフローの実行を確認します。実行実行状態は、実行ステータスを表すように色分けされています。たとえば、青は進行中、緑は成功、赤は失敗を示します。実行イベント履歴」セクションの表を参照して、実行イベントに関する詳細情報を確認することもできます。 グラフィカルなワークフロー実行の例については、このパターンの追加情報セクションのStep Functions グラフを参照してください。	AWS 全般

タスク	説明	必要なスキル
Amazon CloudWatch で配信ログを検証します。	1. AWS マネジメントコンソールにサインインし、「CloudWatch console」 を開きます。 2. ナビゲーションペインで、ログを拡張してから、ロググループを選択します。 3. 検索ボックスで、`s3toelasticsearch` 関数のロググループを検索します。 成功した配信ログの例については、このパターンの追加情報セクションにある CloudWatch 配信ログを参照してください。	AWS 全般

タスク	説明	必要なスキル
OpenSearch ダッシュボードでフォーマットされたファイルを検証し、ファイル操作を実行します。	1. AWS マネジメントコンソールにサインインします。[分析] の下で、[Amazon OpenSearch Service] を選択します。 2. ナビゲーションペインで、[ドメイン] を選択します。 3. 検索ボックスに「OpenSearch ダッシュボード」内のドメインの URL を入力します。 4. ダッシュボードを選択し、「マスターユーザーとしてサインインします」。 5. インデックス化されたデータを表形式で参照します。 6. 入力ファイルを OpenSearch Dashboards のフォーマットされた出力ファイル (インデックス付きドキュメント) と比較します。ダッシュボードビューには、フォーマットされたファイルに追加された列ヘッダーが表示されます。未フォーマットの入力ファイルのソースデータが、ダッシュボードビューのターゲットデータと一致することを確認します。 7. インデックス化されたファイルに対して、検索	AWS 全般

タスク	説明	必要なスキル
	(フィールド名、値、式を使用するなど)、フィルター、「DQL」(Dashboard Query Language) 操作などのアクションを実行します。	

関連リソース

リファレンス

- 「COBOL コピーブックの例」(IBM ドキュメンテーション)
- 「BMC コンピュウェアファイルエイド」(BMC ドキュメンテーション)

チュートリアル

- 「チュートリアル: Amazon S3 トリガーを使用して Lambda 関数を呼び出す」(AWS Lambda ドキュメント)
- 「AWS Step Functions と AWS Lambda を使用してサーバーレスワークフローを作成する方法を教えてください」(AWS ドキュメント)
- 「Amazon OpenSearch Service での OpenSearch Dashboards の使用」(AWS ドキュメント)

追加情報

Step Functions・グラフ

以下は、手順関数図の例を示しています。グラフには、このパターンで使用されている Lambda 関数の実行実行ステータスが表示されます。

CloudWatch デリバリログ

次の例は、`s3toelasticsearch` 実行が正常に実行された場合の配信成功ログを示しています。

2022-08-10T 15:53:33.033-05:00	処理中のドキュメント数:100	
	2022-08-10T 15:53:33 .171-05:00	[情報] 2022-08-10T 20:53 .171Z a1b2-90ab-cdef-EXAMPLE 11111EXAMPLE 111EXAMPLE POST https://search-essearch-3h4uqclifeqaj2vg4mphe7ffle.us-east-2.es.amazonaws.com:443/_bulk [ステータス:200 リクエスト:0.100s]
	2022-08-10T 15:53:33.172-05:00	一括書き込み成功:100 件のドキュメント

Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化

作成者: Richard Milner-Watts (AWS)

概要

このパターンでは、「Blu Age」 ツールを使用することで、最新化されたメインフレームワークロードを実行するためのサンプルコンテナ環境を提供します。Blu Age は従来のメインフレームワークロードを最新の Java コードに変換します。このパターンでは、Javaアプリケーションの周りにラッパーを提供し、「Amazon Elastic Container Service (Amazon ECS)」 や 「Amazon Elastic Kubernetes Service (Amazon EKS)」 のような コンテナオーケストレーションを使用することで、それを実行できます。

Blu Age と AWS のサービスを使用して、ワークロードを最新化する方法の詳細については、以下の AWS 規範ガイダンス出版物を参照してください:

- 「最新のBlu AgeメインフレームワークロードをサーバーレスAWS インフラストラクチャ上で実行」
- 「Terraform を使用して、コンテナ化された Blu Age アプリケーション用の環境をデプロイ」

Blu Age によるメインフレームワークロードを最新化についてサポートが必要な場合は、Blu Age Web サイトの [専門家に連絡] から お問い合わせください。最新のワークロードを AWS に移行したり、AWS のサービスと統合したり、それらを本番環境に移行したりするためのサポートが必要な場合は、AWS アカウントマネージャーにお問い合わせいただくか、AWS プロフェッショナルサービスフォームにご記入ください。

前提条件と制限

前提条件

- Blu Age により作成された最新の Java アプリケーション。テスト目的で、このパターンは、コンセプトの証明として使用できるサンプル Java アプリケーションを提供します。
- コンテナの構築に使用できる 「Docker」 環境。

制約事項

使用するコンテナオーケストレーションプラットフォームによっては、コンテナで使用できるリソース (CPU、RAM、ストレージなど) が限定的である場合があります。たとえば、Amazon ECS を AWS Fargate で使う場合、制限と考慮事項については 「Amazon ECS のドキュメント」 を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Blu Age
- Java

ターゲットテクノロジースタック

- Docker

ターゲットアーキテクチャ

以下の図表は、Docker コンテナ内の Blu Age アプリケーションのアーキテクチャを示しています。

1. コンテナのエントリポイントは、ラッパースクリプトです。この Bash スクリプトは、Blu Age アプリケーションおよび出力処理のランタイム環境を準備する役割を担います。
2. コンテナの環境変数は、Amazon Simple Storage Service (Amazon S3) バケット名およびデータベース認証情報などの、ラッパースクリプトの変数を設定するために使用されます。環境変数は、AWS Secrets Manager または AWS Systems Manager の機能であるパラメータストアによって提供されます。Amazon ECS をコンテナオーケストレーションサービスとして使用している場合、Amazon ECS タスク定義に環境変数をハードコーディングすることもできます。
3. ラッパースクリプトは、Blu Age アプリケーションを実行する前に、S3 バケットからあらゆる入力ファイルをコンテナに取り込みます。AWS コマンドラインインターフェイス (AWS CLI) は、コンテナにインストールされます。これは、ゲートウェイ仮想プライベートクラウド (VPC) エンドポイントを介して、Amazon S3 に保存されたオブジェクトにアクセスするためのメカニズムを提供します。
4. Blu Age アプリケーションの Java アーカイブ (JAR) ファイルは、Amazon Aurora などのその他のデータソースと通信する必要がある場合があります。
5. 完了後、ラッパースクリプトは結果の出力ファイルを さらなる処理を行うために S3 バケットに送ります (例、Amazon CloudWatch ログインサービスで)。またこのパターンでは、標準の

CloudWatch ロギングの代替を使っている場合、圧縮されたログファイルの Amazon S3 への配信もサポートします。

ツール

AWS サービス

- 「Amazon Elastic Container Registry (Amazon ECR)」 は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。

ツール

- 「Docker」 は、アプリケーションを構築、テスト、デプロイするためのソフトウェアプラットフォームです。Docker はソフトウェアを「コンテナ」 と呼ばれる標準化されたユニットにパッケージ化します。コンテナには、ライブラリ、システムツール、コード、ランタイムなど、ソフトウェアの実行に必要なものがすべて含まれています。Docker を使用すると、あらゆる環境にアプリケーションをデプロイしスケーリングできます。
- 「Bash」 は GNU オペレーティングシステムのコマンド言語インターフェイス (シェル) です。
- 「Java」 はこのパターンで使用されるプログラミング言語ならびに開発環境です。
- 「Blu Age」 は、アプリケーションコード、依存関係、インフラストラクチャなどの従来のメインフレームワークロードをクラウド用の最新のワークロードに変換する、AWS mainframe modernization ツールです。

コードリポジトリ

このパターンのコードは、GitHub 内の「 Blu Age サンプルコンテナリポジトリ」で利用できます。

ベストプラクティス

- 環境変数を使用して、変数を外部化し、アプリケーションの動作を変更します。これらの変数により、コンテナオーケストレーションソリューションが、コンテナを再構築することなくランタイム環境を変更できるようにします。このパターンには、Blue Age アプリケーションに役立つ環境変数の例が含まれています。
- Blu Age アプリケーションを実行する前に、アプリケーションの依存関係を検証します。たとえば、データベースが使用可能であること、そして認証情報が有効であることを確認します。依存

関係を検証するテストをラッパースクリプトに記述し、また一致しない場合は早い段階で失敗します。

- ラッパースクリプト内の詳細なログ記録を使用します。オーケストレーションプラットフォームやジョブの所要時間によって、実行中のコンテナと直接やり取りするのが難しい場合があります。問題の診断に役立つ出力が STDOUT に書き込まれていることを必ず確認してください。たとえば、アプリケーションの実行前と実行後の両方で、アプリケーションの作業ディレクトリの内容が出力に含まれる場合があります。

エピック

Blue Age アプリケーション JAR ファイルを入手

タスク	説明	必要なスキル
オプション 1 - Blu Age を使用して、アプリケーションの JAR ファイルを取得します。	このパターンのコンテナには Blu Age アプリケーションが必要です。代わりに、このパターンで提供されるサンプル Java アプリケーションをプロトタイプとして使用することもできます。 Blu Age チームの協力で、コンテナに織り込むことができるアプリケーションの JAR ファイルを取得します。JAR ファイルが利用できない場合、次のタスクを見て、代わりにサンプルアプリケーションを使用します。	クラウドアーキテクト
オプション 2 - 提供されたサンプルアプリケーション JAR ファイルをビルドまたは使用します。	このパターンでは、事前ビルドのサンプル JAR ファイルを提供します。このファイルでは、30 秒間スリープして終了する前に、アプリケーション	アプリ開発者

タスク	説明	必要なスキル
	の環境変数を STDOUT に出力します。 このファイルは bluAgeSample.jar という名前が付けられ、GitHub リポジトリの「docker フォルダー」にあります。 コードを変更して独自バージョンの JAR ファイルをビルドする場合、GitHub リポジトリの「/java_sample/src/sample_java_app.java」にあるソースコードを使用します。ビルドスクリプトを「./java_sample/build.sh」で使用して、 Java ソースをコンパイルし、新しい JAR ファイルをビルドします。	

Blu Age コンテナーをビルドする

タスク	説明	必要なスキル
GitHub リポジトリのクローンを作成します。	以下のコマンドを使用してサンプルコードリポジトリをクローンします： `git clone https://github.com/aws-samples/aws-blu-age-sample-container`	AWS DevOps

タスク	説明	必要なスキル
Docker を使用して、コンテナをビルドします。	Amazon ECR などの Docker レジストリにプッシュする前に、Docker を使用してコンテナをビルドしてください： 1. 選択したターミナルから、ローカルの GitHub リポジトリで docker フォルダーにナビゲーションします。 2. このコマンドを使用してコンテナをビルドします： `docker build -t <tag> .` ここで <tag> は、使用するコンテナ名です。	AWS DevOps
Blu Age コンテナをテストします。	(オプション) 必要に応じて、以下のコマンドを使用してコンテナをローカルでテストします： `docker run -it <tag> / bin/bash`	AWS DevOps

タスク	説明	必要なスキル
Docker リポジトリに対し認証を行います。	Amazon ECR を使用する計画がある場合、「Amazon ECR ドキュメント」の手順に従って、 AWS CLI をインストールし、設定し、Docker CLI をデフォルトのレジストリで認証します。 認証のための get-login-password コマンドを使用することを推奨します。 **Note** Amazon ECR コンソールでは、プッシュコマンドの表示ボタンを使用する場合、このコマンドの事前入力済みバージョンが提供されます。詳細については、「Amazon S3 のドキュメント」参照してください。 `aws ecr get-login -password --region <region> | docker login --username AWS --password-stdin <account>.dkr.ecr.<region>.amazonaws.com`	AWS DevOps

タスク	説明	必要なスキル
	Amazon ECR を使用する計画がない場合、コンテナレジストリシステムの指示に従ってください。	
コンテナリポジトリを作成します。	Amazon ECR でリポジトリを作成します。手順については、「Terraform を使用してコンテナ化された Blu Age アプリケーションの環境をデプロイする」パターンを参照してください。 別のコンテナレジストリシステムを使用する場合、そのシステムの指示に従います。	AWS DevOps

タスク	説明	必要なスキル

コンテナにタグを付けて、ターゲットリポジトリにプッシュします。	Amazon ECR を使用する場合:
	1. ローカルの Docker イメージに Amazon ECR のレジストリとリポジトリをタグ付けして、リモートリポジトリにプッシュできるようにします：
	`docker tag <tag>:latest <account>.dkr.ecr.<region>.amazonaws.com/<repository>:<versionNumber>`
	2. リモートリポジトリにイメージをプッシュします：
	`docker push <account>.dkr.ecr.<region>.amazonaws.com/<repository>:<versionNumber>`
	詳細については、Amazon ECR ユーザーガイド の「Dockerイメージをプッシュする」を参照してください。
		AWS DevOps

関連リソース

「AWS リソース」

- 「AWS Blu Ageサンプルコンテナリポジトリ」

- 「最新のBlu AgeメインフレームワークロードをサーバーレスAWS インフラストラクチャ上で実行」
- 「Terraform を使用して、コンテナ化された Blu Age アプリケーション用の環境をデプロイする」
- 「AWS CLI で Amazon ECR を使用」(Amazon ECR ユーザーガイド)
- 「プライベートレジストリ認証」(Amazon ECR ユーザーガイド)
- 「Amazon ECS ドキュメント」
- 「Amazon EKS ドキュメント」

追加リソース

- 「Blue Age ウェブサイト」
- 「Docker ウェブサイト」

Python を使用して EBCDIC データを AWS 上の ASCII に変換およびアンパックします

作成:ルイス・グスタボ・ダンタス (AWS)

概要

メインフレームは通常、重要なビジネスデータをホストするため、Amazon Web Services（AWS）クラウドやその他の米国情報交換標準コード（ASCII）環境にデータを移行する場合、データを最新化することが最も重要なタスクの 1 つです。メインフレームでは、データは通常、拡張バイナリコード 10 進数交換コード (EBCDIC) 形式でエンコードされます。データベース、仮想ストレージアクセスメソッド (VSAM)、またはフラットファイルをエクスポートすると、通常、圧縮されたバイナリ EBCDIC ファイルが生成され、移行がより複雑になります。最も一般的に使用されるデータベース移行ソリューションはチェンジデータキャプチャ (CDC) で、ほとんどの場合、データエンコーディングを自動的に変換します。ただし、これらのデータベース、VSAM、またはフラットファイルには CDC メカニズムが使用できない場合があります。これらのファイルについては、データを最新化するための代替アプローチが必要です。

このパターンは、EBCDIC データを ASCII 形式に変換して最新化する方法を説明しています。変換後、データを分散データベースにロードしたり、クラウド内のアプリケーションにデータを直接処理させることができます。このパターンは、「メインフレームデータユーティリティ」の GitHub リポジトリにある変換スクリプトとサンプルファイルを使用します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- EBCDIC 入力ファイルとそれに対応する共通ビジネス指向言語 (COBOL) コピーブック。サンプルの EBCDIC ファイルと COBOL コピーブックは、「メインフレームデータユーティリティ」の GitHub リポジトリに含まれています。COBOL コピーブックの詳細については、IBM Web サイトの「z/OS 6.4 プログラミング用エンタープライズ COBOL ガイド」を参照してください。

制約事項

- COBOL プログラム内で定義されたファイルレイアウトはサポートされていません。これらは別途利用できるようにする必要があります。

製品バージョン

- Python バージョン 3.8 以降。

アーキテクチャ

ソーステクノロジースタック

- メインフレーム上の EBCDIC データ
- COBOL コピーブック

ターゲットテクノロジースタック

- 仮想プライベートクラウド (VPC)のAmazon Elastic Compute Cloud (Amazon EC2) インスタンス
- Amazon Elastic Block Store (Amazon EBS)
- Python とそれに必要なパッケージ、JavaScript オブジェクト表記 (JSON)、システム、および日時
- 最新のアプリケーションですぐに読み込んだり、リレーショナルデータベーステーブルに読み込んだりできる ASCII フラットファイル

ターゲットアーキテクチャ

アーキテクチャ図は、EC2 インスタンスで EBCDIC ファイルを ASCII ファイルに変換するプロセスを示しています。

1. 「parse_copybook_to_json.py」スクリプトを使用して、COBOL コピーブックを JSON ファイルに変換します。
2. JSON ファイルと「extract_ebcdic_to_ascii.py」スクリプトを使用して、EBCDIC データを ASCII ファイルに変換します。

自動化とスケール

最初の手動ファイル変換に必要なリソースが揃ったら、ファイル変換を自動化できます。このパターンには自動化の指示は含まれていません。変換を自動化する方法は複数あります。以下は、考えられるアプローチの概要です。

1. AWS コマンドラインインターフェイス (AWS CLI) コマンドと Python スクリプトコマンドをシェルスクリプトにカプセル化します。
2. シェルスクリプトジョブを EC2 インスタンスに非同期的に送信する AWS Lambda 関数を作成します。詳細については、「AWS Lambda を使用した SSH ジョブのスケジュール設定」を参照してください。
3. レガシーファイルがアップロードされるたびに Lambda 関数を呼び出す Amazon Simple Storage Service (Amazon S3) トリガーを作成します。詳細については、チュートリアル： Amazon S3 トリガーを使用して Lambda 関数を呼び出すを参照してください。

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。仮想サーバーを必要な数だけ起動して、迅速にスケールアップまたはスケールダウンができます。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

その他のツール

- 「GitHub」は、コラボレーションツールとバージョン管理を提供するコードホスティングサービスです。
- 「Python」は高水準プログラミング言語です。

コードリポジトリ

このパターンのコードは、 GitHub 内の「mainframe-data-utilities」リポジトリで利用できます。

エピック

EC2 インスタンスの準備

タスク	説明	必要なスキル
EC2 インスタンスを起動します。	EC2 インスタンスにはアウトバウンドのインターネットアクセスが必要です。これにより、インスタンスは GitHub にある Python ソースコードにアクセスできるようになります。インスタンスを作成するには 1. https：//console.aws.amazon.com/ec2 で Amazon EC2 コンソールを開きます。 2. EC2 Linux インスタンスを起動します。パブリック IP アドレスを使用し、ポート 22 経由のインバウンドアクセスを許可します。インスタンスのストレージサイズが EBCDIC データファイルのサイズの 2 倍以上であることを確認してください。手順については、Amazon EC2 ドキュメントを参照してください。	AWS 全般
Git をインストールする。	1. Secure Shell (SSH) クライアントを使用して、起動した EC2 インスタンスに接続します。詳細については、「Linux インスタンスへの	AWS 全般、Linux

タスク	説明	必要なスキル

	接続」を参照してください。
	2. Amazon EC2 コンソールで、次のコマンドを実行します。EC2 インスタンスに Git をインストールします。
	`sudo yum install git`
	3. 次のコマンドを実行して、Git が正常にインストールされていることを確認します。
	`git --version`	

タスク	説明	必要なスキル

Python をインストールします。	1. Amazon EC2 コンソールで、次のコマンドを実行します。EC2 インスタンスに Python をインストールします。
	`sudo yum install python3`
	2. Amazon EC2 コンソールで、次のコマンドを実行します。これにより、EC2 インスタンスに Pip3 がインストールされます。
	`sudo yum install python3-pip`
	3. Amazon EC2 コンソールで、次のコマンドを実行します。これにより、AWS SDK for Python (Boto3) が EC2 インスタンスにインストールされます。
	`sudo pip3 install boto3`
	4. Amazon EC2 コンソールで、次のコマンドを実行します。ここで、`<us-east-1>` は AWS リージョンのコードです。リージョンコードの一覧については、Amazon EC2 ユーザーガイドの「使用可能なリー
		AWS 全般、Linux

タスク	説明	必要なスキル
	ジョン」を参照してください。 `export AWS_DEFAULT_REGION=<us-east-1>`	
GitHub リポジトリのクローンを作成します。	1. Amazon EC2 コンソールで、次のコマンドを実行します。これにより、「メインフレームデータユーティリティ」リポジトリが GitHub からクローンされ、デフォルトのコピー場所であるhomeフォルダが開きます。 `git clone https://github.com/aws-samples/mainframe-data-utilities.git` 2. homeフォルダーに、その `mainframe-data-utilities` フォルダーが存在することを確認します。	AWS 全般、GitHub

EBCDIC データから ASCII ファイルを作成します。

タスク	説明	必要なスキル
COBOL コピーブックを解析して JSON レイアウトファイルにします。	`mainframe-data-utilities` フォルダー内で「parse_copybook_to_json.py」スクリプトを実行します。この自動化モジュール	AWS 全般、Linux

タスク	説明	必要なスキル

	は COBOL コピーブックからファイルレイアウトを読み取り、JSON ファイルを作成します。JSON ファイルには、ソースファイルからのデータの解釈と抽出に必要な情報が含まれています。これにより、COBOL コピーブックから JSON メタデータが作成されます。
	
	次のコマンドは、COBOL コピーブックを JSON ファイルに変換します。
	
	`python3 parse_copybook_to_json.py \ -copybook LegacyReference/COBPACK2.cpy \ -output sample-data/cobpack2-list.json \ -dict sample-data/cobpack2-dict.json \ -ebcdic sample-data/COBPACK.OUTFILE.txt \ -ascii sample-data/COBPACK.ASCII.txt \ -print 10000`
	
	このスクリプトは受け取った引数を出力します。
	
	`------------------ ------------------ ------------------ ----------------`	

タスク	説明	必要なスキル

	<pre>Copybook file.....
	.....| LegacyRef
	erence/COBPACK2.cpy
	Parsed copybook (JSON
	 List).| sample-data/
	cobpack2-list.json
	JSON Dict (document
	ation)...| sample-da
	ta/cobpack2-dict.json
	ASCII file.....
	.....| sample-
	data/COBPACK.ASCII.t
	xt
	EBCDIC file.....
	.....| sample-
	data/COBPACK.OUTFILE
	.txt
	Print each.....
	.....| 10000
	---------- 	
	---------- 	
	---------- 	
	----------</pre> 	
	引数に関する詳細は、GitHub リポジトリの「README ファイル」を参照してください。	

タスク	説明	必要なスキル
JSON レイアウトファイルを調べてください。	1. 「parse_copybook_to_json.py」スクリプトで定義されている出力パスに移動します。 2. 「sample-data/cobpack2-list.json」ファイルの作成時間を確認して、適切な JSON レイアウトファイルを選択していることを確認します。 3. JSON ファイルを調べて、内容が次のようになっていることを確認します。 `"input": "extract-ebcdic-to-ascii/COBPACK.OUTFILE.txt", "output": "extract-ebcdic-to-ascii/COBPACK.ASCII.txt", "max": 0, "skip": 0, "print": 10000, "lrecl": 150, "rem-low-values": true, "separator": "|", "transf": [ { "type": "ch", "bytes": 19, "name": "OUTFILE-TEXT" }`	AWS 全般、JSON

タスク	説明	必要なスキル
	JSON レイアウトファイルの最も重要な属性は次のとおりです。 • `input`— 変換する EBCDIC ファイルのパスを含める • `output`— ASCII ファイルが生成されるパスを定義します。 • `lrecl`— 論理レコード長のサイズをバイト単位で指定します • `transf`— すべてのフィールドとそのサイズ (バイト単位) を一覧表示します。 JSON レイアウトファイルの詳細については、GitHub リポジトリの「README ファイル」を参照してください。	

タスク	説明	必要なスキル
ASCII ファイルを作成します。	クローニングされた GitHub リポジトリに含まれている「extract_ebcdic_to_ascii.py」スクリプトを実行します。このスクリプトは EBCDIC ファイルを読み取り、変換されて読み取り可能な ASCII ファイルを書き込みます。 `python3 extract_ebcdic_to_ascii.py -local-json sample-data/cobpack2-list.json` このスクリプトは EBCDIC データを処理するときに、10,000 レコードのバッチごとにメッセージを出力します。次の例を参照してください。 `---------- 2023-05-15 21:21:46.322253 | Local Json file | -local-json | sample-data/cobpack2-list.json 2023-05-15 21:21:47.034556 | Records processed	10000 2023-05-15 21:21:47.736434 | Records processed	20000`	AWS 全般

タスク	説明	必要なスキル
	2023-05-15 21:21:48. 441696 | Records processed | 30000 2023-05-15 21:21:49. 173781 | Records processed | 40000 2023-05-15 21:21:49. 874779 | Records processed | 50000 2023-05-15 21:21:50. 705873 | Records processed | 60000 2023-05-15 21:21:51. 609335 | Records processed | 70000 2023-05-15 21:21:52. 292989 | Records processed | 80000 2023-05-15 21:21:52. 938366 | Records processed | 89280 2023-05-15 21:21:52. 938448 Seconds 6.616232 印刷頻度を変更する方法については、GitHub リポジトリの「README ファイル」を参照してください。	

タスク	説明	必要なスキル
ASCII ファイルを検証します。	1. 「Extract-EBCDIC-to-ASCII/CobPack.ascii.txt」ファイルの作成時刻を確認して、最近作成されたものであることを確認してください。 2. Amazon EC2 コンソールで、次のコマンドを入力します。これにより ASCII ファイルの最初のレコードが開きます。 `head sample-data/COBPACK.ASCII.txt -n 1| xxd` 3. 最初のレコードの内容を調べます。EBCDIC ファイルは通常バイナリなので、キャリッジリターンやラインフィード (CRLF) の特殊文字はありません。「extract_ebcdic_to_ascii.py」スクリプトは、スクリプトパラメータで定義されている列区切り文字としてパイプ文字を追加します。 提供されているサンプル EBCDIC ファイルを使用した場合、ASCII ファイルの最初のレコードは次のようになります。	AWS 全般、Linux

タスク	説明	必要なスキル
	00000000: 2d30 3030 3030 3030 3030 3130 3030 3030 -000000000100000 00000010: 3030 307c 3030 3030 3030 3030 3031 3030 000|00000 0000100 00000020: 3030 3030 3030 7c2d 3030 3030 3030 3030 000000|-0 0000000 00000030: 3031 3030 3030 3030 3030 7c30 7c30 7c31 0100000000| 0|0|1 00000040: 3030 3030 3030 3030 7c2d 3130 3030 3030 00000000| -100000 00000050: 3030 307c 3130 3030 3030 3030 307c 2d31 000|10000 0000|-1 00000060: 3030 3030 3030 3030 7c30 3030 3030 7c30 00000000| 00000|0 00000070: 3030 3030 7c31 3030 3030 3030 3030 7c2d 0000|1000 00000|- 00000080: 3130 3030 3030 3030 307c 3030 3030 3030 100000000| 000000 00000090: 3030 3030 3130 3030 3030 3030 307c 2d30 000010000 0000|-0 000000a0: 3030 3030 3030 3030 3031 3030	

タスク	説明	必要なスキル
	3030 3030 000000000 1000000 000000b0: 3030 7c41 7c41 7c0a 00|A|A|.	

タスク	説明	必要なスキル

EBCDIC ファイルを評価してください。	Amazon EC2 コンソールで、次のコマンドを入力します。EBCDIC ファイルの最初のレコードが開きます。
	`head sample-data/COBPACK.OUTFILE.txt -c 150 | xxd`
	サンプルの EBCDIC ファイルを使用した場合、結果は次のようになります。
	`00000000: 60f0 f0f0 f0f0 f0f0 f0f0 f0f0 f1f0 f0f0 f0f0 `........ .... 00000010: f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f1f0 f0f0 .... 00000020: f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f1f0 .... 00000030: f0f0 f0f0 f0f0 d000 0000 0005 f5e1 00fa .... 00000040: 0a1f 0000 0000 0005 f5e1 00ff ffff fffa .... 00000050: 0a1f 0000 000f 0000 0c10 0000 000f 1000 .... 00000060: 0000 0d00 0000 0000 1000 0000`
		AWS 全般、Linux、EBCDIC

タスク	説明	必要なスキル
	0f00 0000 .... 00000070: 0000 1000 0000 0dc1 c100 0000 0000 0000 .... 00000080: 0000 0000 0000 0000 0000 0000 0000 0000 .... 00000090: 0000 0000 0000	
	ソースファイルとターゲットファイルの同等性を評価するには、EBCDIC に関する包括的な知識が必要です。たとえば、サンプル EBCDIC ファイルの最初の文字はハイフン (-) です。EBCDIC ファイルの 16 進数表記ではこの文字は60で表され、ASCII ファイルの 16 進表記ではこの文字は2Dで表されます。EBCDIC から ASCII への変換表については、IBM ウェブサイトの「EBCDIC から ASCII へ」を参照してください。	

関連リソース

リファレンス

- EBCDIC 文字セット (IBM ドキュメント)
- EBCDIC から ASCII への変換 (IBM ドキュメント)
- COBOL (IBM ドキュメント)

- JCL の基本的な概念 (IBM ドキュメント)
- Amazon Linux EC2 インスタンスに接続 (Amazon EC2 ドキュメント)

チュートリアル

- AWS Lambda を使用して SSH ジョブをスケジューリングする (AWS ブログ記事)
- Amazon S3 トリガーを使用して Lambda 関数を呼び出す (AWS Lambda ドキュメント)

AWS Lambda を使用して Amazon S3 のメインフレームファイルを EBCDIC 形式から文字区切りの ASCII 形式に変換します

作成:ルイス・グスタボ・ダンタス (AWS)

概要

このパターンは、メインフレームの EBCDIC (拡張バイナリコード 10 進交換コード) ファイルを文字区切りの ASCII (米国情報交換標準コード) ファイルに自動的に変換する AWS Lambda 関数を起動する方法を示しています。Lambda 関数は、ASCII ファイルが Amazon Simple Storage Service (Amazon S3) バケットにアップロードされた後に実行されます。ファイル変換後、x86 ベースのワークロードで ASCII ファイルを読み取ったり、最新のデータベースにファイルをロードしたりできます。

このパターンで示されているファイル変換方法は、現代の環境で EBCDIC ファイルを扱う際の課題を克服するのに役立ちます。EBCDIC でエンコードされたファイルには、バイナリ形式またはパックド 10 進形式で表されるデータが含まれていることが多く、フィールドは固定長です。最近の x86 ベースのワークロードや分散環境は通常 ASCII でエンコードされたデータを処理し、EBCDIC ファイルを処理できないため、これらの特性は障害となります。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- S3 バケット
- 管理者のアクセス許可を持つ AWS Identity and Access Management (IAM) ユーザー
- AWS CloudShell
- Python 3.8.0 以降。
- EBCDIC でエンコードされたフラットファイルと、それに対応するデータ構造を共通ビジネス指向言語 (COBOL) のコピーブックでエンコードしたもの

> **Note**
>
> このパターンでは、サンプル EBCDIC ファイル (CLIENT.EBCDIC.txt) とそれに対応する COBOL コピーブック (COBKS05.cpy) を使用します。どちらのファイルも GitHub「メインフレームデータユーティリティ」リポジトリにあります。

制約事項

- COBOL コピーブックには通常、複数のレイアウト定義があります。「mainframe-data-utilities」プロジェクトは、この種のコピーブックを解析することはできますが、データ変換時にどのレイアウトを検討すべきかを推測することはできません。これは、コピーブックにはこのロジックがないからです (代わりに COBOL プログラムに残っています)。そのため、コピーブックを解析した後は、レイアウトを選択するルールを手動で設定する必要があります。
- このパターンには「Lambda クォータ」が適用されます。

アーキテクチャ

ソーステクノロジースタック

- IBM z/OS、IBM i、およびその他の EBCDIC システム
- EBCDIC でエンコードされたデータを含むシーケンシャル・ファイル (IBM Db2 アンロードなど)
- COBOL コピーブック

ターゲットテクノロジースタック

- Amazon S3
- Amazon S3イベント通知
- IAM
- Lambda function
- Python 3.8 以降
- メインフレームデータユーティリティ
- JSON メタデータ
- 文字区切りの ASCII ファイル

ターゲットアーキテクチャ

次の図は、メインフレームの EBCDIC ファイルを ASCII ファイルに変換するアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーはコピーブックパーサースクリプトを実行して COBOL コピーブックを JSON ファイルに変換します。
2. ユーザーは JSON メタデータを S3 バケットにアップロードします。これにより、データ変換 Lambda 関数でメタデータを読み取ることができます。
3. ユーザーまたは自動プロセスが、EBCDIC ファイルを S3 バケットにアップロードします。
4. S3 通知イベントは、データ変換 Lambda 関数をトリガーします。
5. AWS は Lambda 関数の S3 バケットの読み取り/書き込み権限を検証します。
6. Lambda は S3 バケットからファイルを読み取り、ファイルを EBCDIC から ASCII にローカルに変換します。
7. Lambda はプロセスのステータスを Amazon CloudWatch に記録します。
8. Lambda は、ASCII ファイルを Amazon S3 に書き込みます。

> **Note**
>
> コピーブックパーサースクリプトは、メタデータを JSON に変換し、そのデータを S3 バケットにアップロードした後、1 回だけ実行されます。最初の変換後、S3 バケットにアップロードされた同じ JSON ファイルを使用する EBCDIC ファイルはすべて、同じメタデータを使用します。

ツール

AWS ツール

- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムに監視します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「AWS CloudShell」はブラウザベースのシェルで、AWS コマンドラインインターフェイス (AWS CLI) とプリインストールされたさまざまな開発ツールを使用して、AWS のサービスを管理できます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、自動的にスケーリングします。

その他のツール

- 「GitHub」は、コラボレーションツールとバージョン管理を提供するコードホスティングサービスです。
- 「Python」は高水準プログラミング言語です。

コード

このパターンのコードは、GitHub 内の「mainframe-data-utilities」リポジトリで利用できます。

ベストプラクティス

以下のベストプラクティスを考慮します。

- Amazon リソースネーム (ARN) レベルで設定します。
- IAM ポリシーには、常に最小権限のアクセス権限を付与してください。詳細については、IAM ドキュメントの「IAM のセキュリティのベストプラクティス」を参照してください。

エピック

環境変数と作業フォルダーの作成

タスク	説明	必要なスキル

環境変数を作成します。	以下の環境変数をテキストエディターにコピーし、<placeholder>次の例の値を自分のリソースの値に置き換えます。
	
	`bucket=<your_bucket_name>`
	`account=<your_account_number>`
		AWS 全般

タスク	説明	必要なスキル
	`region=<your_region_code>` **Note** S3 バケット、AWS アカウント、AWS リージョンへの参照は後で作成します。 環境変数を定義するには、「CloudShell コンソール」を開き、更新した環境変数をコピーしてコマンドラインに貼り付けます。 **Note** CloudShell セッションが再起動するたびに、このステップを繰り返す必要があります。	

タスク	説明	必要なスキル
作業フォルダを作成します。	後でリソースをクリーンアップするプロセスを簡略化するには、次のコマンドを実行して CloudShell に作業フォルダを作成します。 `mkdir workdir; cd workdir` **Note** CloudShell セッションへの接続が失われるたびに、ディレクトリを作業ディレクトリ (`workdir`) に変更する必要があります。	AWS 全般

IAM ポリシーと IAM ロールを作成する

タスク	説明	必要なスキル
トリガーのための Lambda 関数を作成します。	Lambda 関数で EBCDIC コンバーターが実行されます。関数には IAM ロールが必要です。IAM ロールを作成する前に、リソースがそのポリシーを引き継ぐことを可能にする信頼ポリシードキュメントを定義する必要があります。 CloudShell 作業フォルダから、次のコマンドを実行して	AWS 全般

タスク	説明	必要なスキル

	ポリシードキュメントを作成します。
	
	`E2ATrustPol=$(cat <<EOF`
	`{`
	`  "Version":`
	`  "2012-10-17",`
	`  "Statement": [`
	`  {`
	`  "Effect":`
	`  "Allow",`
	`  "Principa`
	`l": {`
	`  "Service": "lambda.a`
	`mazonaws.com"`
	`  },`
	`  "Action":`
	`  "sts:AssumeRole"`
	`  }`
	`  ]`
	`}`
	`EOF`
	`)`
	`printf "$E2ATrustPol" >`
	`  E2ATrustPol.json`	

Lambda 変換用に IAM ロールを作成する	IAM ロールを作成するには、CloudShell 作業フォルダから以下の AWS CLI コマンドを実行します。
	
	`aws iam create-role`
	`  --role-name E2AConvLa`
	`mbdaRole --assume-`
	`role-policy-docume`
	`nt file://E2ATrustPol`
	`.json`
		AWS 全般

タスク	説明	必要なスキル

Lambda 関数の IAM ポリシードキュメントを作成します。	Lambda 関数には S3 バケットへの読み取り/書き込みアクセス権限と Amazon CloudWatch Logs の書き込み権限が必要です。
	
	IAM ポリシーを作成するには、CloudShell 作業フォルダから次のコマンドを実行します。
	
	`E2APolicy=$(cat <<EOF`
	`{`
	`  "Version": "2012-10-17",`
	`  "Statement": [`
	`  {`
	`  "Sid": "Logs",`
	`  "Effect": "Allow",`
	`  "Action": [`
	`  "logs:PutLogEvents",`
	`  "logs:CreateLogStream",`
	`  "logs:CreateLogGroup"`
	`  ],`
	`  "Resource": [`
	`  "arn:aws:logs:*:*:log-group:*",`
	`  "arn:aws:logs:*:*:log-group:*:log-stream:*"`
	`]
		AWS 全般

タスク	説明	必要なスキル

	<pre> },
	        {
	            "Sid": "S3",
	            "Effect":
	"Allow",
	            "Action": [
	
	"s3:GetObject",
	
	"s3:PutObject",
	
	"s3:GetObjectVersion"
	            ],
	            "Resource":
	[
	
	"arn:aws:s3:::%s/*",
	
	"arn:aws:s3:::%s"
	            ]
	        }
	    ]
	}
	EOF
	)
	printf "$E2APolicy"
	"$bucket" "$bucket"
	> E2AConvLambdaPolic
	y.json</pre>	

タスク	説明	必要なスキル
IAM ポリシーを IAM ロールにアタッチします。	IAM ポリシーを IAM ロールにアタッチするには、Cloud Shell 作業フォルダから次のコマンドを実行します。 `aws iam put-role-policy --role-name E2AConvLambdaRole --policy-name E2AConvLambdaPolicy --policy-document file://E2AConvLambdaPolicy.json`	AWS 全般

EBCDIC 変換用のLambda 関数を作成する

タスク	説明	必要なスキル
EBCDIC 変換ソースコードをダウンロードしてください。	CloudShell 作業フォルダから次のコマンドを実行して、メインフレームデータユーティリティのソースコードを GitHub からダウンロードします。 `git clone https://github.com/aws-samples/mainframe-data-utilities.git mdu`	AWS 全般
ZIP パッケージを作成します。	CloudShell 作業フォルダから次のコマンドを実行して、EBCDIC 変換用の Lambda 関数を作成する ZIP パッケージを作成します。	AWS 全般

タスク	説明	必要なスキル
	`cd mdu; zip ../mdu.zip *.py; cd ..`	
Lambda 関数を作成します。	CloudShell 作業フォルダから次のコマンドを実行して、EBCDIC 変換用の Lambda 関数を作成します。 `aws lambda create-function \ --function-name E2A \ --runtime python3.9 \ --zip-file fileb://mdu.zip \ --handler extract_ebcdic_to_ascii.lambda_handler \ --role arn:aws:iam::$account:role/E2AConvLambdaRole \ --timeout 10 \ --environment "Variables={layout=$bucket/layout/}"` **Note** 環境変数レイアウトは、JSON メタデータが存在する Lambda 関数に指示します。	AWS 全般

タスク	説明	必要なスキル

Lambda 関数のリソースベースのポリシーを作成します。	CloudShell 作業フォルダから次のコマンドを実行して、Amazon S3 イベント通知が EBCDIC 変換のための Lambda 関数をトリガーできるようにします。
	`aws lambda add-permission \`
	`--function-name E2A \`
	`--action lambda:InvokeFunction \`
	`--principal s3.amazonaws.com \`
	`--source-arn arn:aws:s3:::$bucket \`
	`--source-account $account \`
	`--statement-id 1`
		AWS 全般

Amazon S3イベント通知の作成

タスク	説明	必要なスキル
Amazon S3 イベント通知の設定ドキュメントを作成します。	Amazon S3 イベント通知は、ファイルが入力フォルダに配置されると EBCDIC 変換 Lambda 関数を開始します。 CloudShell 作業フォルダから、次のコマンドを実行して Amazon S3 イベント通知用の JSON ドキュメントを作成します。 `{`	AWS 全般

タスク	説明	必要なスキル

	"LambdaFunctionC
	onfigurations": [
	{
	"Id": "E2A",
	"LambdaFu
	nctionArn": "arn:aws:
	lambda:%s:%s:funct
	ion:E2A",
	"Events":
	["s3:ObjectCreated:
	Put"],
	"Filter": {
	"Key": {
	"FilterRu
	les": [
	{
	"Name":
	"prefix",
	"Value":
	"input/"
	}
	]
	}
	}
	}
	]
	}
	EOF
	)
	printf "$S3E2AEvent"
	"$region" "$account" >
	S3E2AEvent.json	

タスク	説明	必要なスキル
Amazon S3イベント通知を作成します。	CloudShell 作業フォルダから次のコマンドを実行して、Amazon S3 イベント通知を作成します。 `aws s3api put-bucket-notification-configuration --bucket $bucket --notification-configuration file://S3E2AEvent.json`	AWS 全般

JSON メタデータの作成とアップロード

タスク	説明	必要なスキル

COBOL コピーブックを解析します。	CloudShell 作業フォルダーから以下のコマンドを実行して、サンプル COBOL コピーブックを JSON ファイルに解析します (これにより、データファイルを正しく読み取り、スライスする方法が定義される)。
	`python3 mdu/parse_copybook_to_json.py \`
	`-copybook mdu/LegacyReference/COBKS05.cpy \`
	`-output CLIENT.json \`
	`-output-s3key CLIENT.ASCII.txt \`
		AWS 全般

タスク	説明	必要なスキル

	<pre>-output-s3bkt $bucket \
	-output-type s3 \
	-print 25</pre>	

変換ルールを追加します。	サンプルデータファイルとそれに対応する COBOL コピーブックはマルチレイアウトファイルです。つまり、変換では特定のルールに基づいてデータをスライスする必要があります。この場合、各行の 3 番目と 4 番目にあるバイトがレイアウトを定義します。
	
	CloudShell 作業フォルダからCLIENT.json ファイルを編集し、内容を"transf-rule": [], から次のように変更します。
	
	<pre>"transf-rule": [
	{
	"offset": 4,
	"size": 2,
	"hex": "0002",
	"transf": "transf1"
	},
	{
	"offset": 4,
	"size": 2,
	"hex": "0000",
	"transf": "transf2"
	}
	],</pre>
		AWS 全般、IBMメインフレーム、コボル

タスク	説明	必要なスキル
JSON メタデータを S3 バケットにアップロードします。	CloudShell 作業フォルダから次の AWS CLI コマンドを実行して JSON メタデータを S3 バケットにアップロードします。 `aws s3 cp CLIENT.json s3://$bucket/layout/CLIENT.json`	AWS 全般

EBCDIC ファイルを変換してください。

タスク	説明	必要なスキル
EBCDIC ファイルを S3 バケットに送信します。	CloudShell 作業フォルダから次のコマンドを実行して EBCDIC ファイルを S3 バケットに送信します。 `aws s3 cp mdu/sample-data/CLIENT.EBCDIC.txt s3://$bucket/input/` **Note** ASCII ファイルが S3 バケットにアップロードされたときに Lambda 変換関数が再度呼び出されないように、入力 (EBCDIC) ファイルと出力 (ASCII) ファイルに対して異なるフォル	AWS 全般

タスク	説明	必要なスキル
	ダを設定することをお勧めします。	

出力を確認してください。	CloudShell 作業フォルダから次のコマンドを実行して、ASCII ファイルが S3 バケットに生成されているかどうかを確認します。
	`awss3 ls s3://$bucket/`
	Note
	データ変換には数秒かかる場合があります。ASCII ファイルを確認することをお勧めします。
	ASCII ファイルが使用可能になったら、次のコマンドを実行して S3 バケットから現在のフォルダーにファイルをダウンロードします。
	`aws s3 cp s3://$bucket/ CLIENT.ASCII.txt .`
	ASCII ファイルの内容を確認します。
	`head CLIENT.ASCII.txt`
		AWS 全般

環境をクリーンアップする

タスク	説明	必要なスキル

(オプション) 変数とフォルダを準備します。	CloudShell との接続が失われた場合は、再接続してから次のコマンドを実行して、ディレクトリを作業フォルダーに変更します。
	`cd workdir`
	環境変数が定義されていることを確認します。
	`bucket=<your_bucket_name>`
	`account=<your_account_number>`
	`region=<your_region_code>`
		AWS 全般

バケットの通知設定を削除します。	CloudShell 作業フォルダから次のコマンドを実行して、Amazon S3 イベント通知設定を削除します。
	`aws s3api put-bucket-notification-configuration \`
	`--bucket=$bucket \`
	`--notification-configuration="{}"`
		AWS 全般

Lambda 関数を削除する	CloudShell 作業フォルダから次のコマンドを実行して、EBCDIC コンバーターの Lambda 関数を削除します。	AWS 全般

タスク	説明	必要なスキル
	`awslambdadelete-function--function-nameE2A`	

IAM ポリシーと IAM ロールを削除します。	CloudShell 作業フォルダから次のコマンドを実行して、EBCDIC コンバータロールとポリシーを削除します。
	`aws iam delete-role-policy --role-name E2AConvLambdaRole --policy-name E2AConvLambdaPolicy`
	`aws iam delete-role --role-name E2AConvLambdaRole`
		AWS 全般

S3 バケットで生成されたファイルを削除します。	CloudShell 作業フォルダから次のコマンドを実行して、S3 バケットで生成されたファイルを削除します。
	`aws s3 rm s3://$bucket/layout --recursive`
	`aws s3 rm s3://$bucket/input --recursive`
	`aws s3 rm s3://$bucket/CLIENT.ASCII.txt`
		AWS 全般

タスク	説明	必要なスキル
作業フォルダを削除します。	CloudShell 作業フォルダから次のコマンドを実行して、workdirとその内容を削除します。 `cd ..; rm -Rf workdir`	AWS 全般

関連リソース

- メインフレームデータユーティリティのREADME (GitHub)
- EBCDIC 文字セット (IBM ドキュメント)
- EBCDIC から ASCII への変換 (IBM ドキュメント)
- COBOL (IBM ドキュメント)
- Amazon S3 トリガーを使用して Lambda 関数を呼び出す (AWS Lambda ドキュメント)

Micro Focusを使用して複雑なレコードレイアウトのメインフレームデータファイルを変換

作成者：ピーター・ウェスト

概要

このパターンは、テキスト以外のデータや複雑なレコードレイアウトを含むメインフレームデータファイルを、Micro Focus 構造ファイルを使用して EBCDIC (拡張バイナリコード 10 進数交換コード) 文字エンコーディングから ASCII (米国情報交換標準コード) 文字エンコーディングに変換する方法を示しています。ファイルの変換を完了するには、以下の手順を実行する必要があります。

1. メインフレーム環境のすべてのデータ項目とレコードレイアウトを記述した単一のソースファイルを準備します。
2. Micro Focus Classic データファイルツールまたはデータファイルツールの一部として Micro Focus データファイルエディタを使用して、データのレコードレイアウトを含む構造ファイルを作成します。構造ファイルはテキスト以外のデータを識別するので、メインフレームファイルをEBCDICからASCIIに正しく変換できます。
3. クラシックデータファイルツールまたはデータファイルツールを使用して構造ファイルをテストします。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Windows 用 Micro Focus エンタープライズデベロッパー、「AWS Mainframe Modernization」を通じて利用可能

製品バージョン

- Micro Focus エンタープライズサーバー 7.0 以降

ツール

- 「Micro Focus エンタープライズデベロッパー」は、エンタープライズデベロッパーのあらゆる統合開発環境 (IDE) バリアントで作成されたアプリケーションの実行環境を提供します。

- Micro Focus 「クラシックデータファイルツール」は、データファイルの変換、ナビゲート、編集、作成に役立ちます。クラシックデータファイルツールには、「データファイルコンバータ」、「レコードレイアウトエディタ」、および「データファイルエディタ」が含まれています。
- Micro Focus 「データファイルツール」はデータファイルの作成、編集、移動に役立ちます。データファイルツールには、「データファイルエディタ」、「ファイル変換ユーティリティ」、および「データファイル構造コマンドラインユーティリティ」が含まれます。

エピック

ソースファイルの準備

タスク	説明	必要なスキル
ソースコンポーネントを特定する。	テキスト以外のデータを含む再定義を含め、ファイルで使用できるすべてのレコードレイアウトを特定します。 再定義を含むレイアウトがある場合は、それらのレイアウトを、データ構造の各組み合わせを説明する固有のレイアウトに因数分解する必要があります。通常、データファイルのレコードレイアウトは次の原型で記述できます。 • テキストデータのみを含むレコードレイアウト • テキスト以外のデータを含むレコードレイアウト • REDEFINES句に従属する非テキストデータを含むレコードレイアウト	アプリ開発者

タスク	説明	必要なスキル
	複雑なレコードレイアウトを含むファイルのフラット化されたレコードレイアウトの作成について詳しくは、「メインフレーム移行のためのASCII 環境でのEBCDICアプリケーションの再ホスト」を参照してください。	

タスク	説明	必要なスキル

レコードレイアウトの条件を特定する。	複数のレコード レイアウトを持つファイル、またはリデファイン節を持つ複雑なレイアウトを含むファイルでは、変換中に使用するレイアウトを定義するために使用できるレコード内のデータと条件を特定します。この作業については、これらのファイルを処理するプログラムを理解している対象分野の専門家 (SME) と話し合うことをお勧めします。
	
	たとえば、1 つのファイルに、テキスト以外のデータを含む 2 つのレコードタイプが含まれている場合があります。ソースを調べると、以下のようなコードが見つかる可能性があります。
	
	`MOVE "M" TO PART-TYPE`
	`MOVE "MAIN ASSEMBLY" TO`
	`PART-NAME`
	`MOVE "S" TO PART-TYPE`
	`MOVE "SUB ASSEMBLY 1"`
	`TO PART-NAME`
	
	このコードは、以下の識別に役立ちます。
	
	• 「パート-タイプ」フィールドはレコードタイプを決定するために使用されます。
		アプリ開発者

タスク	説明	必要なスキル
	• 「M」という値は「M-パート-レコード」に使用されます • 「S」という値は「S-パート-レコード」に使用されます このフィールドで使用される値を文書化して、レコードレイアウトをファイル内の正しいデータレコードに関連付けることができます。	

タスク	説明	必要なスキル
ソースファイルをビルドします。	ファイルが複数のソースファイルにまたがって記述されている場合、またはレコードレイアウトに REDEFINES 句に従属する非テキストデータが含まれている場合は、レコードレイアウトを含む新しいソースファイルを作成します。新しいプログラムでは、SELECT ステートメントと FD ステートメントを使用してファイルを記述する必要はありません。このプログラムでは、レコードの説明をワーキングストレージ内に 01 レベルとして格納するだけで済みます。 **Note** データファイルごとにソースファイルを作成するか、すべてのデータファイルを記述するマスターソースファイルを作成できます。	アプリ開発者

タスク	説明	必要なスキル
ソースコードをコンパイルします。	ソースファイルをコンパイルしてデータディクショナリを構築します。EBCDIC 文字セットを使用してソースファイルをコンパイルすることをお勧めします。IBMCOMP ディレクティブまたは ODOSLIDE ディレクティブを使用している場合は、ソースファイルでもこれらのディレクティブを使用する必要があります。 **Note** IBMCOMP は COMP フィールドのバイトストレージに影響し、ODOSLIDE は OCCURS VARYING 構造体のパディングに影響します。これらのディレクティブが正しく設定されていないと、変換ツールはデータレコードを正しく読み込めません。その結果、変換されたファイルのデータが不正になります。	アプリ開発者

(オプション A) クラシックデータファイルツールを使用して構造ファイルを作成します。

タスク	説明	必要なスキル
ツールを起動し、辞書をロードします。	1. Windows の「スタート」メニューアイコンを選択し、「マイクロフォーカスエンタープライズデベロッパー」を検索して選択し、「クラシックデータファイルツール」を選択します。 2. 「ファイル」を選択し、「レコードレイアウト」を選択します。 3. 「レイアウトを作成するファイルを選択」ダイアログボックスの「ファイル名」で、ソースファイルを先にコンパイルしたときに作成された IDY (`.idy`) ファイルを選択します。次に、[開く] を選択します。 4. クラシックデータファイルツールが EBCDIC を使用していることを確認するには、「データファイルツール」ダイアログボックスで IDY ファイルが「EBCDIC」に設定され、「データツール」が「ANSI」に設定されている場合は「はい」を選択します。	アプリ開発者
デフォルトのレコードレイアウトを作成します。	どの条件付きレイアウトにも一致しないすべてのレコード	アプリ開発者

タスク	説明	必要なスキル
	には、デフォルトのレコードレイアウトを使用してください。 1. 「レイアウト」ウィンドウでデータ構造を展開し、デフォルトレイアウトに使用されている 01 レベルを探します。 2. 01 項目を右クリックし、「新しいレイアウト」を選択します。 3. 「新規レコードレイアウトウィザード」ダイアログボックスで、「デフォルトレイアウト」を選択し、「次へ」を選択します。 4. [Finish] を選択してください。 デフォルトレイアウトは「レイアウト」ペインに表示され、赤いフォルダアイコンで識別できます。	

タスク	説明	必要なスキル
条件付きレコードレイアウトを作成します。	条件付きレコードレイアウトは、1 つのファイルに複数のレコードレイアウトがある場合に使用します。 1. 「レイアウト」ペインでデータ構造を展開し、条件付きレイアウトに使用する 01 レベルを探します。 2. 01 項目を右クリックし、「新しいレイアウト」を選択します。 3. 「新規レコードレイアウトウィザード」ダイアログボックスで「条件付きレイアウト」を選択し、「次へ」を選択します。 4. [Finish] を選択してください。条件付きレイアウトは「レイアウト」ペインに表示され、黄色のフォルダーアイコンで識別できます。 5. 条件付きレイアウトを展開し、条件を設定する必要があるフィールドを右クリックして、「プロパティ」を選択します。 6. 「フィールドプロパティ」ダイアログボックスに、条件を入力します。文字セットが「EBCDIC」に設定されていることを確認し、「OK」を選択します。条件が設定されているフィール	アプリ開発者

タスク	説明	必要なスキル
	ドの横にチェックマークが表示されます。 7. このレイアウトの条件を必要とする他のフィールドについては、手順 5 ~ 6 を繰り返します。 8. その他の条件付きレイアウトを追加する必要がある場合は、手順 1 ~ 6 を繰り返します。 9. 「ファイル」を選択し、「名前を付けて保存」を選択し、構造ファイルをディスクに保存します。	

(オプション B) データファイルツールを使用して構造ファイルを作成します。

タスク	説明	必要なスキル
ツールを起動し、辞書をロードします。	1. Windows のスタートメニューアイコンを選択し、「Micro Focus Enterprise Developer」を検索して選択し、「データファイルツール」を選択します。 2. 「ファイル」、「新規」、「構造」「ファイル」を選択します。 3. 「開く」ダイアログの「ファイル名」で、ソースファイルを先にコンパイルしたときに作成された IDY (`.idy`) ファイルを選択しま	アプリ開発者

タスク	説明	必要なスキル
	す。次に、[開く] を選択します。 4. データファイルツールが EBCDIC を使用していることを確認するには、「デバッグファイル」セクションのドロップダウンメニューが「EBCDIC」に設定されていることを確認します。	
デフォルトのレコードレイアウトを作成します。	どの条件付きレイアウトにも一致しないすべてのレコードには、デフォルトのレコードレイアウトを使用してください。 1. 左ペインの「使用可能なレイアウト」セクションでデータ構造を展開し、デフォルトレイアウトに使用されている 01 レベルを探します。 2. 01 項目を右クリックし、「デフォルトレイアウトの作成」を選択します。 デフォルトレイアウトは「レイアウト」ペインに表示され、青い「D」アイコンで識別できます。	アプリ開発者

タスク	説明	必要なスキル
条件付きレコードレイアウトを作成します。	条件付きレコードレイアウトは、1 つのファイルに複数のレコードレイアウトがある場合に使用します。 1. 右側のペインの「選択されたレイアウト」セクションでデータ構造を展開し、条件付きレイアウトに使用されている 01 レベルを探します。 2. 01 項目を右クリックし、「条件付きレイアウトの作成」を選択します。条件付きレイアウトは右側の「レイアウト」ペインに表示され、緑色の「C」アイコンで識別できます。 3. 条件付きレイアウトを展開し、条件を設定する必要があるフィールドを右クリックして、「プロパティ」を選択します。 4. 「フィールドプロパティ」ダイアログボックスに、条件を入力します。文字セットが「EBCDIC」に設定されていることを確認し、「OK」を選択します。 条件が設定されているフィールドの横には、赤い「IF」アイコンが表示されます。 5. このレイアウトの条件を必要とする他のフィールドに	アプリ開発者

タスク	説明	必要なスキル
	ついては、手順 3 ~ 4 を繰り返します。 6. その他の条件付きレイアウトを追加する必要がある場合は、手順 1 ~ 4 を繰り返します。 7. 「ファイル」を選択し、「名前を付けて保存」を選択し、構造ファイルをディスクに保存します。	

(オプション A) クラシックデータファイルツールを使用して構造ファイルをテストします。

タスク	説明	必要なスキル
EBCDIC データファイルをテストする。	構造ファイルを使用して EBCDIC テストデータファイルを正しく表示できることを確認します。 1. Windows のスタートメニューアイコンを選択し、「マイクロフォーカスエンタープライズデベロッパー」を見つけて選択し、「クラシックデータツール」を選択します。 2. [ファイル]を選択し、[開く]を選択します。 3. [開く] ダイアログボックスの「ファイル名」で EBCDIC データセットを選択し、「開く」を選択します。	アプリ開発者

タスク	説明	必要なスキル
	4. 「ファイル」、「データファイルエディター」、「レコードレイアウトの読み込み」を選択します。 5. 「開く」ダイアログの「ファイル名」で、構造ファイルを選択し、「開く」を選択します。 6. 文字セットモードが EBCDIC に設定されていることを確認するには、ドロップダウンメニューが「EBCDIC」に設定されていることを確認します。左側のペインには未加工レコードデータ、右のペインにはフォーマット済みデータが表示されます。 7. さまざまなレコードを選択して、すべてのフォーマットが正しいレイアウトでレンダリングされるようにします。	

(オプション B) データファイルツールを使用して構造ファイルをテストします。

タスク	説明	必要なスキル
EBCDIC データファイルをテストする。	構造ファイルを使用して EBCDIC テストデータファイルを正しく表示できることを確認します。	アプリ開発者

タスク	説明	必要なスキル
	1. Windows の「スタート」メニューアイコンを選択し、「マイクロフォーカスエンタープライズデベロッパー」を見つけて選択し、「データファイルツール」を選択します。 2. 「ファイル」、「開く」、「データファイル」を選択します。 3. 「データファイルを開く」ダイアログの「ローカル」タブの「ファイル名」で、「ブラウズ」を選択して EBCDIC テストファイルの場所を検索します。 4. 「構造ファイル (オプション)」では、「ブラウズ」を選択して構造ファイルの場所を検索します。 5. 「ファイルの詳細」セクションで、ファイルの詳細を入力し、「エンコーディング」が「EBCDIC」に設定されていることを確認します。 6. 要件に応じて、「共有的に」「開く」または「排他的に」「開く」モードを選択します。 7. ツールバーの「表示」セクションのドロップダウンメニューが「EBCDIC」に設定されていることを確認し	

タスク	説明	必要なスキル
	ます。左側のペインには未加工レコードデータが表示され、右のペインにはフォーマットされたデータが表示されます。 8. さまざまなレコードを選択して、すべてのフォーマットが正しいレイアウトでレンダリングされるようにします。	

データファイルの変換をテストします。

タスク	説明	必要なスキル
EBCDIC ファイルの変換をテストします。	1. Windows のスタートメニューアイコンを選択し、「マイクロフォーカスエンタープライズデベロッパー」を見つけて選択し、「クラシックデータツール」を選択します。 2. [ツール] を選択し、次に [変換] を選択します。 3. 「データファイル変換」ダイアログボックスの「入力ファイル」セクションの「ファイル名」で、「ブラウズ」を選択して EBCDIC 入力ファイルを検索して選択します。「文字セット」が「EBCDIC」に設定されていることを確認します。	アプリ開発者

タスク	説明	必要なスキル
	4. 「文字セット変換」セクションで、「文字セットを変換」チェックボックスと「レコードに非テキストデータ項目を含む」チェックボックスを選択します。「変換するレイアウトを選択」を選択し、「ブラウズ」を選択して構造ファイルを検索して選択します。 5. 「新規ファイル」セクションの「ファイル名」に、作成する ASCII 出力ファイルのパスとファイル名を入力します。デフォルトでは、変換ツールのデフォルトは入力ファイルと同じ形式です。テストでは、オプションはデフォルト設定値のままにしておきます。 6. 「変換」を選択します。 7. 「(オプション A) クラシックデータファイルツールを使用して構造ファイルをテストする」、または「(オプション B)データファイルツールを使用して構造ファイルをテストする」セクションの手順に従い、EBCDIC ファイルの代わりに ASCII 出力ファイルをロードします。 8. EBCDIC ファイルと ASCII ファイルの両方をデータ	

タスク	説明	必要なスキル
	ファイルエディタに読み込み、ファイルを並べて比較して変換の正確性を確認します。	

関連リソース

- マイクロフォーカス (Micro Focus ドキュメンテーション)
- メインフレームとレガシーコード (AWS ブログ投稿)
- AWS 規範ガイダンス (AWS ドキュメント)
- AWS ドキュメント (AWS ドキュメント)
- AWS ジェネラルリファレンス (AWS ドキュメント)
- AWS 用語集 (AWS ドキュメント)

Terraform を使用して、コンテナ化された Blu Age アプリケーションの環境をデプロイする

作成者: Richard Milner-Watts (AWS)

概要

従来のメインフレームワークロードを最新のクラウドアーキテクチャに移行することで、メインフレームを維持するコストを削減できます。このコストは、環境が古くなるにつれて増加する一方です。ただし、メインフレームからジョブを移行することには固有の課題があります。社内リソースはジョブのロジックに慣れていないかもしれませんし、こうした特殊なタスクでメインフレームの高いパフォーマンスを発揮することは、市販の一般的な CPU と比較すると再現するのが難しい場合があります。これらのジョブを書き直すのは大変な作業で、多大な労力を必要とします。

Blu Age は従来のメインフレームワークロードを最新の Java コードに変換し、それをコンテナとして実行できます。

このパターンは、Blu Age ツールでモダナイズされたコンテナ化されたアプリケーションを実行するサンプルサーバーレスアーキテクチャです。付属の HashiCorp Terraform ファイルは、バッチタスクとリアルタイムサービスの両方をサポートし、Blu Age コンテナのオーケストレーションの安全なアーキテクチャを構築します。

Blu Age と AWS のサービスを使用してワークロードをモダナイズする方法の詳細については、以下の AWS 規範ガイダンス出版物を参照してください。

- Blu Age でモダナイズされたメインフレームワークロードを AWS のサーバーレスインフラストラクチャ上で実行する
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化する

Blu Ageを使用してメインフレームワークロードをモダナイズする方法については、Blu Age の Web サイトで [Contact our experts (専門家に連絡)] を選択して、Blu Age チームにお問い合わせください。最新のワークロードを AWS に移行したり、AWS のサービスと統合したり、それらを本番環境に移行したりするためのサポートが必要な場合は、AWS アカウントマネージャーにお問い合わせいただくか、AWS プロフェッショナルサービスフォームにご記入ください。

前提条件と制限

前提条件

- Blu Age によってモダナイズされたメインフレームのコンテナ化パターンによって提供された、コンテナ化された Blu Age アプリケーションのサンプル。サンプルアプリケーションには、最新化されたアプリケーションの入出力処理を処理するロジックが用意されており、このアーキテクチャと統合できます。
- これらのリソースをデプロイするには Terraform が必要です。

制限

- Amazon Elastic Container Service (Amazon ECS) は、コンテナで使用可能なタスクリソースに制限を設けます。これらのリソースには CPU、RAM、ストレージが含まれます。たとえば、Amazon ECS を AWS Fargate で使用する場合は、タスクリソースの制限が適用されます。

製品バージョン

このソリューションは次のバージョンでテスト済みです。

- Terraform 1.3.6
- Terraform AWS Provider 4.46.0

アーキテクチャ

ソーステクノロジースタック

- Blu Age
- Terraform

ターゲットテクノロジースタック

- Amazon Aurora PostgreSQL 互換エディション
- AWS Backup
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- AWS Secrets Manager

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)
- AWS Step Functions
- AWS Systems Manager

ターゲットアーキテクチャ

ソリューションアーキテクチャを次の図に示します。

1. このソリューションは次の IAM ロールをデプロイします。
   - バッチタスクロール
   - バッチタスク実行ロール
   - サービスタスクロール
   - サービスタスク実行ロール
   - 段階関数ロール
   - AWS Backup ロール
   - RDS 拡張モニタリングロール

   ロールは最小特権アクセスの原則に準拠しています。
2. Amazon ECR は、このパターンによってオーケストレーションされたコンテナイメージの保存に使用されます。
3. AWS Systems Manager Parameter Store は、実行時に各環境に関する設定データを Amazon ECS タスク定義に提供します。
4. AWS Systems Manager は、ランタイムに各環境に関する機密設定データを Amazon ECS タスク定義に提供します。データは AWS KMS によって暗号化されています。
5. Terraform モジュールは、すべてのリアルタイムタスクとバッチタスクの Amazon ECS タスク定義を作成します。
6. Amazon ECS は、コンピュートエンジンとして AWS Fargate を使用してバッチタスクを実行します。これは短寿命のタスクで、必要に応じて AWS Step Functions によって開始されます。
7. Amazon Aurora PostgreSQL 互換は、モダナイズされたアプリケーションをサポートするデータベースを提供します。これは IBM Db2 または IBM IMS DB などのメインフレームデータベースに代わるものです。

8. Amazon ECS は長寿命のサービスを実行して、最新のリアルタイムワークロードを提供します。これらのステートレスアプリケーションは、アベイラビリティーゾーンに分散されたコンテナで永続的に実行されます。
9. Network Load Balancer は、リアルタイムワークロードへのアクセス権付与に使用されます。Network Load Balancer は、IBM CICS などの旧プロトコルをサポートしています。または、HTTP ベースのワークロードで Application Load Balancer を使用できます。
10. Amazon S3 は、ジョブの入出力用のオブジェクトストレージを提供します。コンテナは Amazon S3 へのプル操作とプッシュ操作を処理して、Blu Age アプリケーションの作業ディレクトリを準備する必要があります。
11. AWS Step Functions サービスは、Amazon ECS タスクの実行を調整してバッチワークロードを処理するために使用されます。
12. 各バッチワークロードの SNS トピックは、モダナイズされたアプリケーションを E メールなどの他のシステムと統合、Amazon S3 から FTP への出力オブジェクトの配信などの追加アクションの開始に使用されます。

> **Note**
>
> デフォルトでは、ソリューションはインターネットにアクセスできません。このパターンは、仮想プライベートクラウド (VPC) が AWS Transit Gateway などのサービスを使用して他のネットワークに接続されることを前提としています。そのため、ソリューションが使用する AWS サービスへのアクセスを付与するために、複数のインターフェイス VPC エンドポイントがデプロイされます。インターネットへの直接アクセスを有効にするために、Terraform モジュールのトグルを使用して VPC エンドポイントをインターネットゲートウェイと関連リソースに置き換えできます。

自動化とスケール

このパターン全体でサーバーレスリソースを使用することで、スケールアウトによって設計の規模にほとんど制限がないことを確認できます。これにより、元のメインフレームで体験する可能性のあるコンピュートリソースの競合など、近隣のノイズ懸念が軽減されます。バッチタスクは、必要に応じて同時に実行するようスケジュールできます。

各コンテナは、Fargate がサポートしている最大サイズで制限されます。詳細については、Amazon ECS ドキュメントのタスク CPU とメモリセクションを参照してください。

リアルタイムワークロードを水平方向にスケールするために、コンテナを追加できます。

ツール

AWS のサービス

- 「Amazon Aurora PostgreSQL 互換エディション」は、PostgreSQL デプロイのセットアップ、運用、スケーリングに役立つ、フルマネージド型のACID準拠のリレーショナルデータベースエンジンです。
- AWS Backup は、データ保護をクラウド、オンプレミス AWS のサービス、および 間で一元化および自動化するのに役立つフルマネージドサービスです。
- 「Amazon Elastic Container Registry (Amazon ECR)」 は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージレジストリサービスです。
- 「Amazon Elastic Container Service (Amazon ECS)」 は、クラスターでのコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。
- AWS Identity and Access Management (IAM) は、リソースの使用を認証および認可されたユーザーを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータの保存、保護、取得に役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functions は、 AWS Lambda 関数やその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。
- AWS Systems Manager Parameter Store は、設定データ管理とシークレット管理用の安全な階層型ストレージを提供します。

その他のサービス

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。このパターンでは、Terraform を使用してサンプルアーキテクチャを作成します。

コードリポジトリ

このパターンのソースコードは、GitHub Blu Age サンプル ECS インフラストラクチャ (Terraform) リポジトリにあります。

ベストプラクティス

- テスト環境では、最新のアプリケーションを設定する `forceDate` オプションなどの機能を使用して、既知の期間に常に実行することで一貫したテスト結果を生成します。
- 各タスクを個別に調整して、最適な量のリソースを消費します。Amazon CloudWatch Container Insights を使用すると、潜在的なボトルネックに関するガイダンスを取得できます。

エピック

デプロイの環境を準備する

タスク	説明	必要なスキル
ソリューションソースコードを複製します。	GitHub プロジェクトからソリューションコードを複製します。	DevOps エンジニア
Terraform の状態を保存するリソースをデプロイして、環境をブートストラップします。	1. ターミナルウィンドウを開き、Terraform がインストールされ、 AWS 認証情報が使用可能であることを確認します。 2. `bootstrap-terraform` フォルダに移動します。 3. S3 バケット (`<accountId>-terraform-backend`) と Amazon DynamoDB テーブル (`terraform-lock`) の名前を変更する場合は、`main.tf` ファイルを編集します。	DevOps エンジニア

タスク	説明	必要なスキル
	4. `terraform apply` コマンドを実行して、リソースをデプロイします。S3 バケットと DynamoDB テーブル名を書き留めます。	

ソリューションインフラストラクチャをデプロイする

タスク	説明	必要なスキル
Terraform の設定を確認および更新します。	ルートディレクトリで `main.tf`, ファイルを開き、内容を確認し、以下の更新を検討します。 1. 文字列を検索し、使用するリージョン`eu-west-1` に置き換え AWS リージョンて、 を更新します。 2. 前のエピックでデフォルトが変更された場合は、`Terraform Backend` ブロックのバケット名を更新します。 3. 前のエピックでデフォルトが変更された場合は、`dynamodb_table` 値を更新します。 4. `stack_prefix` 変数の値を目的の文字列に更新します。この文字列は、このパターンによって作成されたすべてのリソースの名前の前に追加されます。	DevOps エンジニア

タスク	説明	必要なスキル
	5. vpc_cidr の値を更新します。少なくとも /24 のアドレス範囲である必要があります。 6. Locals セクションを確認します。これはデプロイされる Blu Age タスクの定義に使用されます。ソリューションではリスト bluage_batch_modules オブジェクトを繰り返し、リストの各要素に関連するリソース (Step Functions ステートマシン、タスク定義、SNS トピック) を作成します。場合によっては、環境ごとに変数の調整が必要になる場合があります。たとえば、テスト環境でランタイムを強制する場合は、force_execution_time 変数の値を変更できます。 7. インターネットアクセスを有効にするには、direct_internet_access_required の値を false から true に変更します。これにより、インフラストラクチャのパブリックインターネットアクセスを有効にする NAT ゲートウェイと	

タスク	説明	必要なスキル
	ルートテーブルとともに、インターネットゲートウェイがデプロイされます。デフォルトでは、ソリューションはインターネットに直接アクセスせずにインターフェイス VPC エンドポイントを VPC にデプロイします。 8. Elastic Load Balancing で提供されるすべてのクライアント/サーバーワークロードにアクセスを付与するには、許可されるべき CIDR ネットワークを使用して `additional_nlb_ingress_cidrs` の値を更新します。	
Terraform ファイルをデプロイします。	ターミナルから、`terraform apply` コマンドを実行してすべてのリソースをデプロイします。Terraform によって生成された変更を確認し、はいと入力してビルドを開始します。 このインフラストラクチャのデプロイには 15 分以上かかる場合がある点に注意してください。	DevOps エンジニア

(オプション) 有効な Blu Age コンテナ化アプリケーションをデプロイする

タスク	説明	必要なスキル
Blu Age コンテナイメージを Amazon ECR にプッシュします。	前のエピックで作成した Amazon ECR リポジトリにコンテナをプッシュします。手順については、Amazon ECR ドキュメントを参照してください。 コンテナイメージの URI を書き留めます。	DevOps エンジニア
Blu Age のコンテナイメージを参照する Terraform を更新します。	アップロードしたコンテナイメージを参照する、`main.tf` ファイルを更新します。	DevOps エンジニア
Terraform ファイルを再デプロイします。	ターミナルから `terraform apply` を実行して、すべてのリソースをデプロイします。Terraform から提案された更新を確認してから、はいと入力してデプロイを続行します。	DevOps エンジニア

関連リソース

- Blu Age
- Blu Age でモダナイズされたメインフレームワークロードを AWS のサーバーレスインフラストラクチャ上で実行する
- Blu Age によってモダナイズされたメインフレームワークロードをコンテナ化する

QuickSight で AWS Mainframe Modernization と Amazon Q を使用して Db2 z/OS データインサイトを生成する

作成者: Shubham Roy (AWS)、Roshna Razack (AWS)、Santosh Kumar Singh (AWS)

概要

組織が IBM Db2 メインフレーム環境でビジネスクリティカルなデータをホストしている場合、そのデータからインサイトを取得することは、成長とイノベーションを促進するために不可欠です。メインフレームデータをロック解除することで、迅速、安全、スケーラブルなビジネスインテリジェンスを構築して、Amazon Web Services (AWS) クラウドでデータ駆動型の意思決定、成長、イノベーションを加速できます。

このパターンは、ビジネスインサイトを生成し、IBM Db2 for z/OS テーブルのメインフレームデータから共有可能な説明文を作成するためのソリューションを示しています。メインフレームのデータ変更は、AWS Mainframe Modernization Precisely によるデータレプリケーションを使用して Amazon Managed Streaming for Apache Kafka (Amazon MSK) トピックにストリーミングされます。Amazon Redshift ストリーミング取り込みを使用すると、Amazon MSK トピックデータは Amazon QuickSight の分析のために Amazon Redshift Serverless データウェアハウステーブルに保存されます。

QuickSight でデータが利用可能になったら、Amazon Q in QuickSight で自然言語プロンプトを使用して、データの概要の作成、質問、データストーリーの生成を行うことができます。SQL クエリを記述したり、ビジネスインテリジェンス (BI) ツールを学習したりする必要はありません。

ビジネスコンテキスト

このパターンは、メインフレームデータ分析とデータインサイトのユースケースのソリューションを示しています。パターンを使用して、会社のデータのビジュアルダッシュボードを構築します。このソリューションを実証するために、このパターンでは、米国のメンバーに医療、歯、ビジョンプランを提供する医療会社を使用しています。この例では、メンバー属性とプラン情報が IBM Db2 for z/OS データテーブルに保存されます。ビジュアルダッシュボードには、以下が表示されます。

- リージョン別のメンバーディストリビューション
- 性別によるメンバー分布
- 年齢別のメンバーディストリビューション
- プランタイプ別のメンバーディストリビューション
- 予防イミュニゼーションを完了していないメンバー

リージョン別のメンバーディストリビューションの例と、予防イミュニゼーションを完了していないメンバーの例については、「追加情報」セクションを参照してください。

ダッシュボードを作成したら、前の分析からのインサイトを説明するデータストーリーを生成します。データストーリーは、予防的な免除を完了したメンバーの数を増やすための推奨事項を提供します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。このソリューションは、Amazon Elastic Compute Cloud (Amazon EC2) の Amazon Linux 2 で構築およびテストされました。
- メインフレームシステムがアクセスできるサブネットを持つ Virtual Private Cloud (VPC)。
- ビジネスデータを含むメインフレームデータベース。このソリューションの構築とテストに使用されるデータの例については、「添付ファイル」セクションを参照してください。
- Db2 z/OS テーブルで有効になっている変更データキャプチャ (CDC)。Db2 z/OS で CDC を有効にするには、IBM のドキュメントを参照してください。
- ソースデータベースをホストしている z/OS システムにインストールされている z/OS 用 CDC を正確に接続します。Precisely Connect CDC for z/OS イメージは、 AWS Mainframe Modernization - Data Replication for IBM z/OS Amazon Machine Image (AMI) 内の zip ファイルとして提供されます。Precisely Connect CDC for z/OS をメインフレームにインストールするには、Precisely インストールドキュメントを参照してください。

制約事項

- メインフレーム Db2 データは、Precisely Connect CDC でサポートされているデータ型である必要があります。サポートされているデータ型のリストについては、Precisely Connect CDC ドキュメントを参照してください。
- Amazon MSK のデータは、Amazon Redshift でサポートされているデータ型である必要があります。サポートされているデータ型のリストについては、Amazon Redshift のドキュメントを参照してください。
- Amazon Redshift には、データ型ごとに異なる動作とサイズ制限があります。詳細については、Amazon Redshift のドキュメントを参照してください。
- QuickSight のほぼリアルタイムのデータは、Amazon Redshift データベースに設定された更新間隔によって異なります。

- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。QuickSight の Amazon Q は現在、QuickSight をサポートするすべてのリージョンで利用できるわけではありません。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

製品バージョン

- AWS Mainframe Modernization Precisely バージョン 4.1.44 によるデータレプリケーション
- Python バージョン 3.6 以降
- Apache Kafka バージョン 3.5.1

アーキテクチャ

ターゲット アーキテクチャ

次の図は、AWS Mainframe Modernization Precisely と QuickSight の Amazon Q でデータレプリケーションを使用してメインフレームデータからビジネスインサイトを生成するためのアーキテクチャを示しています。 QuickSight

この図表は、次のワークフローを示しています：

1. Precisely Log Reader Agent は、Db2 ログからデータを読み取り、メインフレーム上の OMVS ファイルシステムの一時ストレージにデータを書き込みます。
2. パブリッシャーエージェントは、一時ストレージから raw Db2 ログを読み取ります。
3. オンプレミスコントローラーデーモンは、オペレーションを認証、認可、モニタリング、管理します。
4. Apply Agent は、事前設定された AMI を使用して Amazon EC2 にデプロイされます。TCP/IP を使用して、コントローラーデーモンを介してパブリッシャーエージェントに接続します。Apply Agent は、複数のワーカーを使用してデータを Amazon MSK にプッシュし、高スループットを実現します。
5. ワーカーは、データを JSON 形式で Amazon MSK トピックに書き込みます。レプリケートされたメッセージの中間ターゲットとして、Amazon MSK は高可用性と自動フェイルオーバー機能を提供します。

6. Amazon Redshift ストリーミング取り込みは、Amazon MSK から Amazon Redshift Serverless データベースへの低レイテンシーで高速なデータ取り込みを提供します。Amazon Redshift のストアドプロシージャは、メインフレーム変更データ (insert/update/deletes) の調整を Amazon Redshift テーブルに実行します。これらの Amazon Redshift テーブルは、QuickSight のデータ分析ソースとして機能します。
7. ユーザーは、分析とインサイトのために QuickSight のデータにアクセスします。QuickSight の Amazon Q を使用して、自然言語プロンプトを使用してデータを操作できます。

ツール

AWS のサービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動し、すばやくスケールアウトまたはスケールインできます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- 「Amazon Managed Streaming for Apache Kafka (Amazon MSK)」 は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションを、構築および実行することを支援するフルマネージドサービスです。
- Amazon QuickSight は、視覚化、分析、レポート生成に使用できるクラウドスケールのビジネスインテリジェンス (BI) サービスです。このパターンでは、QuickSight の Amazon Q の生成 BI 機能を使用します。
- Amazon Redshift Serverless は Amazon Redshift のサーバーレスオプションであり、データウェアハウスインフラストラクチャを設定および管理しなくても、数秒で分析の実行とスケーリングをより効率的に行うことができます。
- AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。

その他のツール

- Precisely Connect CDC は、レガシーシステムからデータを収集し、クラウドおよびデータプラットフォームに統合します。

コードリポジトリ

このパターンのコードは、GitHub Mainframe_DataInsights_change_data_reconciliation リポジトリで入手できます。コードは Amazon Redshift のストアドプロシージャです。このストアドプロシージャは、Amazon MSK から Amazon Redshift テーブルへのメインフレームデータの変更 (挿入、更新、削除) を調整します。これらの Amazon Redshift テーブルは、QuickSight のデータ分析ソースとして機能します。

ベストプラクティス

- Amazon MSK クラスターをセットアップする際は、ベストプラクティスに従ってください。
- Amazon Redshift データ解析のベストプラクティスに従ってパフォーマンスを向上させます。
- Precisely セットアップ用の AWS Identity and Access Management (IAM) ロールを作成するときは、最小特権の原則に従い、タスクの実行に必要な最小限のアクセス許可を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

Amazon EC2 で Precisely を使用して AWS Mainframe Modernization データレプリケーションを設定する

タスク	説明	必要なスキル

セキュリティグループを設定します。	コントローラーデーモンと Amazon MSK クラスターに接続するには、EC2 インスタンスのセキュリティグループを作成します。次のインバウンドルールとアウトバウンドルールを追加します。
	• インバウンドルール 1:
	  • Type で、カスタム TCP を選択します。
	  • [プロトコル] で [TCP] を選択します。
		DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	• ポート範囲には、2626 (Precisely コントローラーデーモンのデフォルトポート) またはメインフレームで実行されているコントローラーデーモンのポート番号を選択します。 • Source で、CIDR ブロックを選択します。 • インバウンドルール 2: • [タイプ] で [カスタム TCP] を選択します。 • Protocol で、SSH を選択します。 • ポート範囲 で、22 を選択します。 • ソース で、IP アドレスまたはプレフィックスリストを選択します。 • インバウンドルール 3: • [タイプ] で [カスタム TCP] を選択します。 • [プロトコル] で [TCP] を選択します。 • ポート範囲 で、9092-9098 を選択します。 • Source で CIDR ブロックを選択します。 • アウトバウンドルール 1:	

タスク	説明	必要なスキル
	• [タイプ] で [カスタム TCP] を選択します。 • [プロトコル] で [TCP] を選択します。 • ポート範囲 で、9092-9098 を選択します。 • Source で CIDR ブロックを選択します。 • アウトバウンドルール 2: • [タイプ] で [カスタム TCP] を選択します。 • [プロトコル] で [TCP] を選択します。 • ポート範囲には、2626 (Precisely コントローラーデーモンのデフォルトポート) またはメインフレームで実行されているコントローラーデーモンのポート番号を選択します。 • Source で CIDR ブロックを選択します。 セキュリティグループの名前を書き留めます。EC2 インスタンスを起動し、Amazon MSK クラスターを設定するときに、名前を参照する必要があります。	

タスク	説明	必要なスキル
IAM ポリシーと IAM ロールを作成します。	1. IAM ポリシーと IAM ロールを作成するには、AWS ドキュメントの指示に従います。 IAM ポリシーは、Amazon MSK クラスターでトピックを作成し、それらのトピックにデータを送信するアクセスを許可します。 2. IAM ロールを作成したら、ポリシーを関連付けます。 IAM ロール名を書き留めます。このロールは、EC2 インスタンスを起動するときに IAM インスタンスプロファイルとして使用されます。	DevOps エンジニア、AWS システム管理者

タスク	説明	必要なスキル
EC2 インスタンスをプロビジョニングします。	Precisely CDC を実行して Amazon MSK に接続するように EC2 インスタンスをプロビジョニングするには、次の手順を実行します。 1. AWS Marketplace にサインインし、AWS Mainframe Modernization – IBM z/OS のデータレプリケーションにサブスクライブします。 2. マネージドサブスクリプションから AMI を選択し、新しいインスタンスを起動で を選択します。 3. インスタンス名、インスタンスタイプ、キーペア、VPC、サブネットなど、その他の設定の詳細を入力します。詳細については、 Amazon EC2 ドキュメントを参照してください。 4. ドロップダウンリストで、前に作成したセキュリティグループを選択します。 5. 詳細 IAM インスタンスプロファイルで、前に作成したロールを選択する必要があります。 6. [インスタンスを起動] を選択します。	AWS 管理者、DevOps エンジニア

Amazon MSK のセットアップ

タスク	説明	必要なスキル
Amazon MSK クラスターを作成します。	Amazon MSK クラスターを作成するには、次の を実行します。 1. にサインインし AWS Managemen t Console、https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。 2. [クラスターを作成] を選択してください。 3. クラスター作成方法ではカスタム作成を選択し、クラスタータイプではプロビジョンドを選択します。 4. クラスターの名前を入力します。 5. 必要に応じてクラスター設定を更新し、他の設定のデフォルトを維持します。 6. < Kafka バージョン > を書き留めます。 Kafka クライアントのセットアップ時に必要になります。 7. [次へ] を選択します。 8. Precisely EC2 インスタンスに使用したのと同じ VPC とサブネットを選択し、前に作成したセキュリティグループを選択します。	AWS DevOps、クラウド管理者

タスク	説明	必要なスキル
	9. セキュリティ設定セクションで、SASL/SCRAM と IAM ロールベースの認証の両方を有効にします。Precisely Connect CDC は SASL/SCRAM (Simple Authentication and Security Layer/ Salted Challenge Response Mechanism) を使用し、Amazon Redshift への接続には IAM が必要です。 10[次へ] を選択します。 11確認するには、モニタリングとブローカーのログ配信方法を選択します。 12次へ を選択し、クラスターの作成 を選択します。 一般的なプロビジョニングされたクラスターの作成には最大 15 分かかります。クラスターが作成されると、そのステータスは作成中からアクティブに変わります。	

タスク	説明	必要なスキル
SASL/SCRAM 認証を設定します。	Amazon MSK クラスターの SASL/SCRAM 認証を設定するには、次の手順を実行します。 1. Secrets Manager でシークレットを設定するには、AWS ドキュメントの指示に従います。 2. Amazon MSK コンソールを開き、前に作成した Amazon MSK クラスターを選択します。 3. [プロパティ] タブを選択します。 4. シークレットの関連付け、シークレットの選択、作成したシークレットキーの選択、シークレットの関連付けを選択します。 次のような成功メッセージが表示されます。 `Successfully associated 1 secret for cluster <chosen cluster name>` 5. クラスター名を選択します。 6. クラスターの概要で、クライアント情報の表示を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
	7. 認証タイプ SASL/SCRAM のプライベートエンドポイント接続文字列を書き留めます。	

タスク	説明	必要なスキル

Amazon MSK トピックを作成します。	Amazon MSK トピックを作成するには、以下を実行します。
	
	1. 前に作成した EC2 インスタンスに接続し、次のコマンドを実行して最新の更新をインストールします。
	
	`sudo yum update -y`
	
	2. 次のコマンドを実行して、Java および Kafka ライブラリをインストールします。
	
	`sudo yum install -y java-11 librdkafka librdkafka-devel`
	
	3. kafka で という名前のフォルダを作成するには/home/ec2-user 、そのフォルダに移動し、次のコマンドを実行します。
	
	`mkdir kafka;cd kafka`
	
	4. kafka クライアントライブラリを kafkaフォルダにダウンロードし、 を Amazon MSK クラスターの作成時にメモした Kafka バージョン<YOUR MSK VERSION>に置き換えます。
		クラウド管理者

タスク	説明	必要なスキル
	``` wget https://a rchive.apache.org/ dist/kafka//kafka_ 2.13-<YOUR MSK VERSION>.tgz ``` 5. ダウンロードしたファイルを抽出するには、次のコマンドを実行し、 を置き換えますYOUR MSK VERSION>。 ``` tar -xzf kafka_2.13- <YOUR MSK VERSION>. tgz ``` 6. kafka libs ディレクトリに移動して Java IAM 認証 Java Archive (JAR) ファイルをダウンロードするには、 を置き換えて次のコマンドを実行します<YOUR MSK VERSION>。 ``` cd kafka_2.13-<YOUR MSK VERSION>/libs wget https://g ithub.com/aws/aws- msk-iam-auth/relea ses/download/v1.1. 1/aws-msk-iam-auth -1.1.1-all.jarkafka ``` 7. Kafka bin ディレクトリに移動して client.properties ファイルを作	

タスク	説明	必要なスキル

	成するには、次のコマンドを実行します。
	
	`cd /home/ec2-user/kafka/kafka_2.13-<YOUR MSK VERSION>/bin`
	`cat >client.properties`
	
	8. 次の内容で client.properties ファイルを更新します。
	
	`security.protocol=SASL_SSL`
	`sasl.mechanism=AWS_MSK_IAM`
	`sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;`
	`sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler`
	
	9. Kafka トピックを作成するには、Kafka ビンに移動して次のコマンドを実行し、 を Amazon MSK クラスターの作成時にメモした IAM ブートストラップサーバーのプライベートエンドポイント<kafka broker>に置き換えます。
	
	`./kafka-topics.sh --bootstrap-server`	

タスク	説明	必要なスキル
	<kafka broker> --command-config client.properties --create --replication-factor 3 –partitions 6 --topic <topic name> 10.メッセージCreated topic <topic name>が表示されたら、トピック名を書き留めます。	

Amazon EC2 で Precisely Apply エンジンを設定する

タスク	説明	必要なスキル
データ変更をレプリケートするように Precisely スクリプトを設定します。	Precisely Connect CDC スクリプトを設定して、変更されたデータをメインフレームから Amazon MSK トピックにレプリケートするには、次の手順を実行します。 1. フォルダ名を正確に作成し、そのフォルダに変更するには、次のコマンドを実行します。 `mkdir /home/ec2-user/precisely;cd /home/ec2-user/precisely` 2. と scriptsという名前のフォルダを 2 つ作成しddls、 scriptsフォル	アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル

	ダに変更するには、次のコマンドを実行します。
	
	`mkdir scripts;mkdir ddls;cd scripts`
	
	3. scripts フォルダsqdata_kafka_producer.conf に という名前のファイルを作成するには、次のコマンドを実行します。
	
	`cat >sqdata_kafka_producer.conf`
	
	4. 次の内容でsqdata_kafka_producer.conf ファイルを更新します。
	
	`builtin.features=SASL_SCRAM`
	`security.protocol=SASL_SSL`
	`sasl.mechanism=SCRAM-SHA-512`
	`sasl.username=<User Name>`
	`sasl.password=<Password>`
	`metadata.broker.list=<SASL/SCRAM Bootstrap servers>`
	
	Note
	以前に設定した Amazon MSK	

タスク	説明	必要なスキル
	SASL/SCRAM ブローカーリスト<SASL/SCRAM Bootstrap servers>で を更新します。Secrets Manager で以前に設定したユーザー名とパスワード<Password> で <User Name>と を更新します。	
	5. scripts フォルダにscript.sqd ファイルを作成します。 `cat >script.sqd` Apply Engine は script.sqd を使用してソースデータを処理し、ソースデータをターゲットにレプリケートします。Apply Engine スクリプトの例については、「追加情報」セクションを参照してください。 6. ddls フォルダに変更し、Db2 テーブルごとに .ddl ファイルを作成するには、次のコマンドを実行します。	

タスク	説明	必要なスキル
	``` cd /home/ec2-user/pre cisely/ddls cat >mem_details.ddl cat >mem_plans.ddl ``` .ddl ファイルの例については、「追加情報」セクションを参照してください。	

タスク	説明	必要なスキル
ネットワーク ACL キーを生成します。	ネットワークアクセスコントロールリスト (ネットワーク ACL) キーを生成するには、次の手順を実行します。 1. sqdata インストールパスをエクスポートするには、次のコマンドを実行します。 ``` export PATH=$PATH:/usr/sbin:/opt/precisely/di/sqdata/bin ``` 2. /home/ec2-user ディレクトリに変更してネットワーク ACL キーを生成するには、次のコマンドを実行します。 ``` cd /home/ec2-user sqdutil keygen --force ``` パブリックキーとプライベートキーが生成されると、次のメッセージが表示されます。 ``` SQDUT04I Generating a private key in file /home/ec2-user/.nacl/id_nacl SQDC017I sqdutil(pid=27344) terminated successfully ```	クラウドアーキテクト、AWS DevOps

タスク	説明	必要なスキル
	3. `.nacl` フォルダに保存されている生成されたパブリックキーを書き留めます。	

メインフレームソース環境を準備する

タスク	説明	必要なスキル
ISPF 画面でデフォルトを設定します。	Interactive System Productivity Facility (ISPF) でデフォルト設定を設定するには、Precisely ドキュメントの指示に従います。	メインフレームシステム管理者
コントローラーデーモンを設定します。	コントローラーデーモンを設定するには、以下を実行します。 1. SQData z/OS メインメニュー画面で、オプション 2 を選択します。 2. Add Daemon to List 画面で、Daemon Name フィールドにデーモンの名前を入力し、Enter キーを押します。	メインフレームシステム管理者
パブリッシャーを設定します。	パブリッシャーを設定するには、以下を実行します。 1. SQData z/OS メインメニュー画面で、オプション 3 を選択します。これにより、キャプチャ/パブリッ	メインフレームシステム管理者

タスク	説明	必要なスキル
	シャーの概要画面が表示されます。 2. CAB ファイルを追加するオプションを選択します。これにより、一覧表示する CAB ファイルの追加画面が表示されます。 3. Name フィールドに、CAB ファイルの名前を入力します。Db2 の場合は、Type を と入力しますD。 4. [Enter] キーを押します。これにより、新しい Db2 キャプチャ CAB ファイルの作成画面が表示されます。 5. zFS Dir フィールドで、ストレージマウントポイントを指定します。 6. Enter キーを押して保存し、続行します。	

タスク	説明	必要なスキル
デーモン設定ファイルを更新します。	コントローラーデーモン設定ファイルのパブリッシャーの詳細を更新するには、次の手順を実行します。 1. SQData z/OS メインメニュー画面で、オプション 2 を選択します。 2. 作成したデーモンSの近くに と入力して、デーモンの詳細を表示します。 3. と入力し1、Enter キーを押してエージェントファイルを編集します。 4. CAB ファイルの詳細を追加します。次の例は、という名前の CAB ファイルの詳細を示していますDB2ZTOMSK 。の代わりにメインフレームユーザー ID を使用します<userid>。 `ÝDB2ZTOMSK¨ type=capture cab=/u/<userid>/sqdata/DB2ZTOMSK.cab` 5. F3 を押します。 6. 2 を入力して ACL ファイルを編集します。次の例に示すようにuserid、 acl を設定ファイルに追加します。	メインフレームシステム管理者

タスク	説明	必要なスキル
	Ýacls¨ prod=admin,<userid> 7. F3 を押して保存し、終了します。	
コントローラーデーモンを開始するジョブを作成します。	ジョブを作成するには、以下を実行します。 1. オプションで、「」と入力しますG。 2. JOB カード、ジョブライブラリと proc ライブラリ、Db2 loadライブラリの詳細を入力します。 3. ネットワーク ACL ファイルの詳細を入力し、オプション 2 を入力して、指定されたジョブライブラリにジョブコントロール言語 (JCL) ファイルを生成します。	メインフレームシステム管理者

タスク	説明	必要なスキル
キャプチャパブリッシャー JCL ファイルを生成します。	キャプチャパブリッシャー JCL ファイルを生成するには、次の手順を実行します。 1. SQData z/OS メインメニュー画面で、オプション 3 を選択します。これにより、キャプチャ/パブリッシャーの概要画面が表示されます。 2. CAB ファイルSの横にあるを入力して選択します。これにより、Db2 Capture/Publisher Detail 画面が表示されます。 3. オプションGで、`capture/publisher` ジョブを生成するオプションを入力します。 4. JOB カード、ジョブ、プロシージャライブラリ、Db2 ロードライブラリの詳細を入力します。 5. ジョブを作成するには、オプション 4 を選択します。ジョブは、ジョブライブラリで指定されたジョブライブラリに作成されます。	メインフレームシステム管理者

タスク	説明	必要なスキル

CDC を確認して更新します。	1. 次のクエリを実行して Db2 テーブルの DATACAPTURE フラグを確認し、Db2 テーブル名<table name>に変更します。
	`SELECT DATACAPTURE FROM SYSIBM.SYSTABLES WHERE NAME='<table name>';`
	結果が DATACAPTURE と表示されることを確認しますY。
	2. DATACAPTURE がでない場合はY、次のクエリを実行して Db2 テーブルで CDC を有効にし、Db2 テーブル名<table name>に変更します。
	`ALTER TABLE <table name> DATA CAPTURE CHANGES;`
		メインフレームシステム管理者

タスク	説明	必要なスキル
JCL ファイルを送信します。	前のステップで設定した次の JCL ファイルを送信します。 • コントローラーデーモンを起動する JCL ファイル • キャプチャと公開を開始する JCL ファイル JCL ファイルを送信したら、EC2 インスタンスで Precisely で Apply Engine を起動できます。	メインフレームシステム管理者

CDC の実行と検証

タスク	説明	必要なスキル
Apply Engine を起動し、CDC を検証します。	EC2 インスタンスで Apply Engine を起動し、CDC を検証するには、次の手順を実行します。 1. EC2 インスタンスに接続するには、AWS ドキュメントの指示に従います。 2. `script.sqd` ファイルを含むディレクトリに変更します。 `cd /home/ec2-user/precisely/scripts`	クラウドアーキテクト、アプリ開発者

タスク	説明	必要なスキル
	3. Apply Engine を起動するには、次の sqdeng start コマンドを実行します。 `sqdeng -s script.sqd --identity=/home/ec2-user/.nacl/id_nacl` Apply Engine は、メインフレームソースからの更新の待機を開始します。 4. CDC をテストするには、Db2 テーブルでレコードの挿入または更新を行います。 5. Apply Engine ログに、キャプチャされてターゲットに書き込まれたレコードの数が表示されていることを確認します。	

タスク	説明	必要なスキル

Amazon MSK トピックのレコードを検証します。	Kafka トピックからメッセージを読み取るには、次の手順を実行します。
	
	1. EC2 インスタンスの Kafka クライアントインストールパスのbinディレクトリを変更するには、次のコマンドを実行し、 をお使いのバージョン<Kafka version>に置き換えます。
	
	`cd /home/ec2-user/kafka/kafka_2.13-<Kafka version>/bin`
	
	2. Kafka トピックでメッセージとして書き込まれた Db2 CDC を検証するには、次のコマンドを実行し、 <kafka broker>とを前に作成したトピック<Topic Name>に置き換えます。
	
	`./kafka-console-consumer.sh --bootstrap-server <kafka broker>:9098 --topic <Topic Name> --from-beginning --consumer.config client.properties`
	
	3. メッセージが Db2 テーブルで更新されたレコードの数
		アプリ開発者、クラウドアーキテクト

タスク	説明	必要なスキル
	と一致することを確認します。	

Amazon Redshift Serverless データウェアハウスにメインフレーム変更データを保存する

タスク	説明	必要なスキル
Amazon Redshift Serverless をセットアップします。	Amazon Redshift Serverless データウェアハウスを作成するには、AWS ドキュメントの指示に従います。 Amazon Redshift Serverless ダッシュボードで、名前空間とワークグループが作成され、使用可能であることを確認します。このパターン例では、プロセスに 2～5 分かかる場合があります。	データエンジニア
ストリーミング取り込みに必要な IAM ロールと信頼ポリシーを設定します。	Amazon MSK からの Amazon Redshift Serverless ストリーミング取り込みを設定するには、次の手順を実行します。 1. Amazon Redshift が Amazon MSK にアクセスするための IAM ポリシーを作成します。 `[region]` を AWS リージョン for Amazon MSK、`[account-id]` を AWS アカウント ID、 を Amazon MSK クラスター名`[msk-cluster-name]` に置き	データエンジニア

タスク	説明	必要なスキル
	換えて、次のコードを実行します。 `{"Version": "2012-10-17","Statement": [{"Sid": "MSKIAMpolicy","Effect": "Allow","Action": ["kafka-cluster:ReadData","kafka-cluster:DescribeTopic","kafka-cluster:Connect"],"Resource": ["arn:aws:kafka:[region]:[account-id]:cluster/[msk-cluster-name]/*","arn:aws:kafka:[region]:[account-id]:topic/[msk-cluster-name]/*"]},{"Effect": "Allow","Action": ["kafka-cluster:AlterGroup","kafka-cluster:DescribeGroup"],"Resource": ["arn:aws:kafka:[region]:[account-id]:group/[msk-cluster-name]/*"]}]}` クラスター名と Amazon リソースネーム (ARN)	

タスク	説明	必要なスキル
	は、Amazon MSK コンソールで確認できます。コンソールで、クラスターの概要を選択し、ARN を選択します。 2. IAM ロールを作成してポリシーをアタッチするには、AWS ドキュメントの指示に従います。 3. IAM ロールを Amazon Redshift Serverless 名前空間にアタッチするには、次の手順を実行します。 a. コンソールにサインインし、https://console.aws.amazon.com/redshiftv2/ で Amazon Redshift コンソールを開きます。 b. [Serverless dashboard] (サーバーレスダッシュボード) を選択します。 c. 名前空間を選択します。 d. セキュリティと暗号化タブを選択します。 e. アクセス許可を選択し、作成した IAM ロールをアタッチします。 4. Amazon Redshift Serverless セキュリティグループで、以下の詳細を含むインバウンドルールを作成します。	

タスク	説明	必要なスキル
	• [タイプ] で [カスタム TCP] を選択します。 • [プロトコル] で [TCP] を選択します。 • ポート範囲 で、9098、9198 を選択します。 • Source で、Amazon MSK セキュリティグループを選択します。 5. Amazon MSK セキュリティグループで、以下の詳細を含むインバウンドルールを作成します。 • [タイプ] で [カスタム TCP] を選択します。 • [プロトコル] で [TCP] を選択します。 • ポート範囲 で、9098、9198 を選択します。 • Source で、Amazon Redshift セキュリティグループを選択します。 このパターンでは、Amazon Redshift と Amazon MSK の両方の設定の IAM 認証にポートを使用します。詳細については、AWS ドキュメント (ステップ 2) を参照してください。	

タスク	説明	必要なスキル
	6. Amazon Redshift Serverless ワークグループの拡張 VPC ルーティングを有効にします。詳細については、AWS のドキュメントを参照してください。	
Amazon Redshift Serverless を Amazon MSK に接続します。	Amazon MSK トピックに接続するには、Amazon Redshift Serverless で外部スキーマを作成します。Amazon Redshift クエリエディタ v2 で、次の SQL コマンドを実行し、 を以前に作成したロール`'iam_role_arn'` に置き換え、 `'MSK_cluster_arn` をクラスターの ARN に置き換えます。 `CREATE EXTERNAL SCHEMA member_schema FROM MSK IAM_ROLE 'iam_role_arn' AUTHENTICATION iam URI 'MSK_cluster_arn';`	移行エンジニア

タスク	説明	必要なスキル
マテリアライズドビューを作成します。	Amazon Redshift Serverless で Amazon MSK トピックのデータを使用するには、マテリアライズドビューを作成します。Amazon Redshift クエリエディタ v2 で、次の SQL コマンドを実行し、 を Amazon MSK トピックの名前<MSK_Topic_name> に置き換えます。 `CREATE MATERIALIZED VIEW member_view AUTO REFRESH YES AS SELECT kafka_partition, kafka_offset, refresh_time, json_parse(kafka_value) AS Data FROM member_schema.<MSK_Topic_name> WHERE CAN_JSON_PARSE(kafka_value);`	移行エンジニア

タスク	説明	必要なスキル

Amazon Redshift でターゲットテーブルを作成します。	Amazon Redshift テーブルは QuickSight の入力を提供します。このパターンでは、メインフレームのソース Db2 テーブル`member_plans` と一致するテーブル`member_dtls` と を使用します。
	
	Amazon Redshift で 2 つのテーブルを作成するには、Amazon Redshift クエリエディタ v2 で次の SQL コマンドを実行します。
	
	`-- Table 1: members_dtls`
	`CREATE TABLE members_dtls (`
	`memberid INT ENCODE AZ64,`
	`member_name VARCHAR(100) ENCODE ZSTD,`
	`member_type VARCHAR(50) ENCODE ZSTD,`
	`age INT ENCODE AZ64,`
	`gender CHAR(1) ENCODE BYTEDICT,`
	`email VARCHAR(100) ENCODE ZSTD,`
	`region VARCHAR(50) ENCODE ZSTD`
	`) DISTSTYLE AUTO;`
	
	`-- Table 2: member_plans`
	`CREATE TABLE member_plans (`
	`memberid INT ENCODE AZ64,`
		移行エンジニア

タスク	説明	必要なスキル

	<pre>medical_plan CHAR(1)
	ENCODE BYTEDICT,
	dental_plan CHAR(1)
	ENCODE BYTEDICT,
	vision_plan CHAR(1)
	ENCODE BYTEDICT,
	preventive_immuniz
	ation VARCHAR(50)
	ENCODE ZSTD
	) DISTSTYLE AUTO;</pre>	

Amazon Redshift でストアドプロシージャを作成します。	このパターンでは、ストアドプロシージャを使用してINSERT、QuickSight で分析するために、ソースメインフレームからターゲット Amazon Redshift データウェアハウステーブルに変更データ (UPDATE、、DELETE) を同期します。 Amazon Redshift でストアドプロシージャを作成するには、クエリエディタ v2 を使用して、GitHub リポジトリにあるストアドプロシージャコードを実行します。	移行エンジニア

タスク	説明	必要なスキル
ストリーミングマテリアライズドビューから読み取り、ターゲットテーブルにロードします。	ストアドプロシージャは、ストリーミングマテリアライズドビューからデータ変更を読み取り、データ変更をターゲットテーブルにロードします。ストアドプロシージャを実行するには、次のコマンドを使用します。 `call SP_Members_Load();` Amazon EventBridge を使用して、Amazon Redshift データウェアハウス内のジョブをスケジュールし、データレイテンシー要件に基づいてこのストアドプロシージャを呼び出すことができます。EventBridge は、一定の間隔でジョブを実行します。プロシージャへの前回の呼び出しが完了したかどうかをモニタリングするには、 AWS Step Functions ステートマシンなどのメカニズムを使用する必要がある場合があります。詳細については、以下のリソースを参照してください。 • スケジュールに従って実行する Amazon EventBridge ルールの作成 • AWS Step Functions と Amazon Redshift Data API	移行エンジニア

タスク	説明	必要なスキル
	を使用して ELT プロセスのオーケストレーションを高速化します。 もう 1 つのオプションは、Amazon Redshift クエリエディタ v2 を使用して更新をスケジュールすることです。詳細については、「クエリエディタ v2 を使用したクエリのスケジュール」を参照してください。	

Amazon Redshift のデータに QuickSight を接続する

タスク	説明	必要なスキル
QuickSight をセットアップします。	QuickSight を設定するには、AWS ドキュメントの指示に従います。	移行エンジニア
QuickSight と Amazon Redshift 間の安全な接続を設定します。	QuickSight と Amazon Redshift 間の安全な接続を設定するには、次の手順を実行します。 1. QuickSight から Amazon Redshift への接続を承認するには、Amazon Redshift コンソールを開き、Amazon Redshift セキュリティグループにインバウンドルールを追加します。このルールでは、QuickSight をセット	移行エンジニア

タスク	説明	必要なスキル
	アップした CIDR 範囲からポート 5439 (デフォルトの Redshift ポート) へのトラフィックを許可する必要があります。 AWS リージョン とその IP アドレスのリストについては、「Supported AWS リージョン for QuickSight」を参照してください。 2. Amazon Redshift コンソールで、ワークグループ、データアクセス、ネットワークとセキュリティを選択し、パブリックアクセスを有効にします。	

タスク	説明	必要なスキル
QuickSight のデータセットを作成します。	Amazon Redshift から QuickSight のデータセットを作成するには、次の手順を実行します。 1. QuickSight コンソールのナビゲーションペインで、データセットを選択します。 2. [Datasets (データセット)] ページで、[New data set (新しいデータセット)] を選択します。 3. Redshift 手動接続を選択します。 4. 新しい Redshift データソースウィンドウで、接続情報を入力します。 • データソース名に、Amazon Redshift データソースの名前を入力します。 • データベースサーバーには、Amazon Redshift クラスターのエンドポイントを入力します。Amazon Redshift Serverless ダッシュボードのクラスターワークグループの一般情報セクションのエンドポイントフィールドからエンドポイント値を取得できます。次の例に示すよう	移行エンジニア

タスク	説明	必要なスキル
	に、サーバーアドレスはコロンの前のエンドポイントの最初の部分です。 `mfdata-insights.NNNNNNNN.us-east-1.redshift-serverless.amazonaws.com:5439/dev` • ポートには、 5439 (Amazon Redshift のデフォルトポート) と入力します。 • データベースの名前を入力します (エンドポイントのスラッシュの後) 。この場合、データベース名は ですdev。 • ユーザー名とパスワードには、Amazon Redshift データベースのユーザー名とパスワードを入力します。 5. 接続の検証を選択します。成功すると、検証を示す緑色のチェックマークが表示されます。検証に失敗した場合は、「トラブルシューティング」セクションを参照してください。 6. [データソースを作成] を選択します。	

タスク	説明	必要なスキル
データセットを結合します。	QuickSight で分析を作成するには、 AWS ドキュメントの手順に従って 2 つのテーブルを結合します。 結合設定ペインで、結合タイプの左を選択します。Join 句で、 を使用しますmemberid from member_plans = memberid from members_details 。	移行エンジニア

QuickSight で Amazon Q を使用してメインフレームデータからビジネスインサイトを取得する

タスク	説明	必要なスキル
QuickSight で Amazon Q をセットアップします。	QuickSight Generative BI 機能で Amazon Q を設定するには、 AWS ドキュメントの指示に従います。	移行エンジニア
メインフレームデータを分析し、ビジュアルダッシュボードを構築します。	QuickSight でデータを分析および視覚化するには、以下を実行します。 1. メインフレームデータ分析を作成するには、 AWS ドキュメントの指示に従います。データセットで、作成したデータセットを選択します。 2. 分析ページで、ビジュアルの構築を選択します。	移行エンジニア

タスク	説明	必要なスキル
	3. 分析用のトピックの作成ウィンドウで、既存のトピックの更新を選択します。 4. トピックの選択ドロップダウンリストで、前に作成したトピックを選択します。 5. トピックのリンクを選択します。 6. トピックをリンクしたら、ビジュアルの構築を選択して Amazon Q Build a Visual ウィンドウを開きます。 7. プロンプトバーに、分析の質問を書き込みます。このパターンに使用される質問の例を次に示します。 • リージョン別にメンバーディストリビューションを表示する • 年齢別にメンバーディストリビューションを表示する • 性別でメンバーディストリビューションを表示する • プランタイプ別にメンバーディストリビューションを表示する • メンバーに予防イミュナイゼーションを完了していないことを示す	

タスク	説明	必要なスキル
	質問を入力したら、ビルドを選択します。QuickSight の Amazon Q はビジュアルを作成します。 8. ビジュアルダッシュボードにビジュアルを追加するには、分析に追加を選択します。 完了したら、ダッシュボードを公開して、組織内の他のユーザーと共有できます。例については、「追加情報」セクションの「メインフレームビジュアルダッシュボード」を参照してください。	

メインフレームデータから QuickSight で Amazon Q を使用してデータストーリーを作成する

タスク	説明	必要なスキル
データストーリーを作成します。	前の分析からのインサイトを説明するデータストーリーを作成し、メンバーの予防的免除を増やすレコメンデーションを生成します。 1. データストーリーを作成するには、AWS ドキュメントの指示に従います。 2. データストーリープロンプトには、以下を使用します。	移行エンジニア

タスク	説明	必要なスキル
	`Build a data story about Region with most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunization. Include 4 points of supporting data for this pattern.` また、独自のプロンプトを構築して、他のビジネスインサイトのデータストーリーを生成することもできます。 3. ビジュアルの追加を選択し、データストーリーに関連するビジュアルを追加します。このパターンでは、前に作成したビジュアルを使用します。 4. [Build] を選択します。 5. データストーリー出力の例については、「追加情報」セクションの「データストーリー出力」を参照してください。	

タスク	説明	必要なスキル
生成されたデータストーリーを表示します。	生成されたデータストーリーを表示するには、データストーリーページでそのストーリーを選択します。	移行エンジニア
生成されたデータストーリーを編集します。	データストーリーのフォーマット、レイアウト、またはビジュアルを変更するには、AWS ドキュメントの指示に従います。	移行エンジニア
データストーリーを共有します。	データストーリーを共有するには、 AWS ドキュメントの指示に従います。	移行エンジニア

トラブルシューティング

問題	ソリューション
QuickSight から Amazon Redshift へのデータセットを作成するために、 `Validate Connection` はフェイルしました。	1. Amazon Redshift Serverless インスタンスにアタッチされたセキュリティグループが、QuickSight をセットアップしたリージョンに関連付けられた IP アドレス範囲からのインバウンドトラフィックを許可していることを確認します。 2. Amazon Redshift Serverless がデプロイされている VPC が公開されていることを確認します。 3. Amazon Redshift に正しいユーザー名とパスワードを使用していることを確認します。Amazon Redshift コンソールでユーザー名とパスワードをリセットできます。

問題	ソリューション
EC2 インスタンスで Apply エンジンを起動しようとすると、次のエラーが返されます。 `-bash: sqdeng: command not found`	次のコマンドを実行してsqdata、インストールパスをエクスポートします。 `export PATH=$PATH:/usr/sbin:/opt/precisely/di/sqdata/bin`
Apply Engine を起動しようとすると、次のいずれかの接続エラーが返されます。 • `SQDD018E Cannot connect to transfer socket(rc==0x18468). Agent:<Agent Name > Socket:/u /./sqdata/.DB2ZTOMSK.cab.data` • `SQDUR06E Error opening url cdc:// <VPC end point name>:2626/ DB2ZTOMSK/DB2ZTOMSK : errno:112 8 (Unknown error 1128)`	メインフレームスプールをチェックして、コントローラーデーモンジョブが実行されていることを確認します。

関連リソース

- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してインサイトを生成する (パターン)
- QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する (デモ)
- AWS Mainframe Modernization - IBM z/OS のデータレプリケーション
- マテリアライズドビューへの Amazon Redshift ストリーミング取り込み

追加情報

.ddl ファイルの例

members_details.ddl

```
CREATE TABLE MEMBER_DTLS (
```

```
memberid INTEGER NOT NULL,
member_name VARCHAR(50),
member_type VARCHAR(20),
age INTEGER,
gender CHAR(1),
email VARCHAR(100),
region VARCHAR(20)
);
```

member_plans.ddl

```
CREATE TABLE MEMBER_PLANS (
memberid INTEGER NOT NULL,
medical_plan CHAR(1),
dental_plan CHAR(1),
vision_plan CHAR(1),
preventive_immunization VARCHAR(20)
);
```

.sqd ファイルの例

を Amazon MSK トピック名に置き換え <kafka topic name>ます。

script.sqd

```
-- Name: DB2ZTOMSK: DB2z To MSK JOBNAME DB2ZTOMSK;REPORT EVERY 1;OPTIONS
 CDCOP('I','U','D');-- Source Descriptions
JOBNAME DB2ZTOMSK;
REPORT EVERY 1;
OPTIONS CDCOP('I','U','D');

-- Source Descriptions
BEGIN GROUP DB2_SOURCE;
DESCRIPTION DB2SQL /var/precisely/di/sqdata/apply/DB2ZTOMSK/ddl/mem_details.ddl AS
 MEMBER_DTLS;
DESCRIPTION DB2SQL /var/precisely/di/sqdata/apply/DB2ZTOMSK/ddl/mem_plans.ddl AS
 MEMBER_PLANS;
END GROUP;
-- Source Datastore
DATASTORE cdc://<zos_host_name>/DB2ZTOMSK/DB2ZTOMSK
OF UTSCDC
AS CDCIN
DESCRIBED BY GROUP DB2_SOURCE ;
```

```
-- Target Datastore(s)
DATASTORE 'kafka:///<kafka topic name>/key'
OF JSON
AS TARGET
DESCRIBED BY GROUP DB2_SOURCE;
PROCESS INTO TARGET
SELECT
{
REPLICATE(TARGET)
}
FROM CDCIN;
```

メインフレームビジュアルダッシュボード

次のデータビジュアルは、分析の質問 のために QuickSight の Amazon Q によって作成されましたshow member distribution by region。

次のデータビジュアルは、質問 について QuickSight の Amazon Q によって作成されましたshow member distribution by Region who have not completed preventive immunization, in pie chart。

データストーリーの出力

次のスクリーンショットは、プロンプト の QuickSight で Amazon Q によって作成されたデータストーリーのセクションを示していますBuild a data story about Region with most numbers of members. Also show the member distribution by age, member distribution by gender. Recommend how to motivate members to complete immunization. Include 4 points of supporting data for this pattern。

この概要では、データストーリーでは、メンバー数が最も多いリージョンを選択して、イミュニゼーションの取り組みから最大の影響を与えることを推奨しています。

データストーリーは、4 つのリージョンのメンバー番号の分析を提供します。メンバーが最も多いのは、北東部、南西部、および南東部リージョンです。

データストーリーは、年齢別のメンバーの分析を示しています。

データストーリーは、米国中西部でのイミュニゼーションの取り組みに焦点を当てています。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

QuickSight で AWS Mainframe Modernization と Amazon Q を使用してデータインサイトを生成する

作成者: Shubham Roy (AWS)、Roshna Razack (AWS)、Santosh Kumar Singh (AWS)

概要

組織がビジネスクリティカルなデータをメインフレーム環境でホストしている場合、そのデータからインサイトを取得することは、成長とイノベーションを促進するために不可欠です。メインフレームデータを ロック解除することで、迅速、安全、スケーラブルなビジネスインテリジェンスを構築して、Amazon Web Services (AWS) クラウドでデータ駆動型の意思決定、成長、イノベーションを加速できます。

このパターンは、QuickSight の BMC と Amazon Q でAWS Mainframe Modernization ファイル転送を使用して、ビジネスインサイトを生成し、メインフレームデータから共有可能な説明文を作成するためのソリューションを示しています。メインフレームデータセットは、AWS Mainframe Modernization File Transfer with BMC を使用して Amazon Simple Storage Service (Amazon S3) に転送されます。 AWS Lambda 関数は、Amazon QuickSight にロードするためのメインフレームデータファイルをフォーマットして準備します。

QuickSight でデータが利用可能になったら、QuickSight の Amazon Q で自然言語プロンプトを使用して、データの概要の作成、質問、データストーリーの生成を行うことができます。SQL クエリを記述したり、ビジネスインテリジェンス (BI) ツールを学習したりする必要はありません。

ビジネスコンテキスト

このパターンは、メインフレームデータ分析とデータインサイトのユースケースのソリューションを示しています。パターンを使用して、会社のデータのビジュアルダッシュボードを構築します。このソリューションを実証するために、このパターンでは、米国のメンバーに医療、歯、ビジョンプランを提供する医療会社を使用しています。この例では、メンバー属性とプラン情報がメインフレームデータセットに保存されます。ビジュアルダッシュボードには、以下が表示されます。

- リージョン別のメンバーディストリビューション
- 性別によるメンバー分布
- 年齢別のメンバーディストリビューション
- プランタイプ別のメンバーディストリビューション
- 予防イミュニゼーションを完了していないメンバー

ダッシュボードを作成したら、前の分析からのインサイトを説明するデータストーリーを生成します。データストーリーは、予防的な免除を完了したメンバーの数を増やすための推奨事項を提供します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- ビジネスデータを含むメインフレームデータセット
- メインフレームにファイル転送エージェントをインストールするアクセス

制約事項

- メインフレームデータファイルは、QuickSight でサポートされているファイル形式のいずれかである必要があります。サポートされているファイル形式のリストについては、Amazon QuickSight のドキュメントを参照してください。

  このパターンでは、Lambda 関数を使用してメインフレームファイルを QuickSight でサポートされている形式に変換します。

アーキテクチャ

次の図は、QuickSight で BMC と Amazon Q で AWS Mainframe Modernization ファイル転送を使用してメインフレームデータからビジネスインサイトを生成するためのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ビジネスデータを含むメインフレームデータセットは、BMC での AWS Mainframe Modernization ファイル転送を使用して Amazon S3 に転送されます。
2. Lambda 関数は、ファイル転送先 S3 バケットにあるファイルをカンマ区切り値 (CSV) 形式に変換します。
3. Lambda 関数は、変換されたファイルをソースデータセット S3 バケットに送信します。
4. ファイル内のデータは QuickSight によって取り込まれます。

5. ユーザーは QuickSight のデータにアクセスします。QuickSight の Amazon Q を使用して、自然言語プロンプトを使用してデータを操作できます。

ツール

AWS サービス

- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Mainframe Modernization BMC を使用したファイル転送は、メインフレームのモダナイゼーション、移行、拡張のユースケースのために、メインフレームデータセットを変換して Amazon S3 に転送します。
- Amazon QuickSight は、単一のダッシュボードでデータを視覚化、分析、レポートするのに役立つクラウドスケールの BI サービスです。このパターンでは、QuickSight の Amazon Q の生成 BI 機能を使用します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

ベストプラクティス

- BMC と Lambda 関数を使用して AWS Mainframe Modernization ファイル転送用の AWS Identity and Access Management (IAM) ロールを作成するときは、最小特権の原則に従います。
- ソースデータセットに QuickSight でサポートされているデータ型があることを確認します。ソースデータセットにサポートされていないデータ型が含まれている場合は、サポートされているデータ型に変換します。サポートされていないメインフレームデータ型と、QuickSight で Amazon Q がサポートするデータ型に変換する方法については、関連リソースセクションを参照してください。

エピック

BMC で AWS Mainframe Modernization ファイル転送を設定する

タスク	説明	必要なスキル
ファイル転送エージェントをインストールします。	メインフレームに AWS Mainframe Modernization File Transfer Agent をインストールするには、AWS ドキュメントの指示に従います。	メインフレームシステム管理者
メインフレームファイル転送用の S3 バケットを作成します。	S3 バケットを作成して、BMC による AWS Mainframe Modernization ファイル転送から出力ファイルを保存します。アーキテクチャ図では、これはファイル転送先バケットです。	移行エンジニア
データ転送エンドポイントを作成します。	1. S3 バケットを作成して、BMC による AWS Mainframe Modernization ファイル転送の入力メインフレームファイルをステージングします。 2. メインフレームデータ転送エンドポイントを作成するには、AWS ドキュメントの指示に従います。	AWS Mainframe Modernization スペシャリスト

QuickSight 統合のメインフレームファイル名拡張子を変換する

タスク	説明	必要なスキル
S3 バケットを作成する。	Lambda 関数の S3 バケットを作成して、変換されたメイン	移行エンジニア

タスク	説明	必要なスキル
	フレームファイルをソースから最終的な送信先バケットにコピーします。	

タスク	説明	必要なスキル
Lambda 関数を作成する。	ファイル拡張子を変更し、メインフレームファイルをレプリケート先バケットにコピーする Lambda 関数を作成するには、次の手順を実行します。 1. にサインインし AWS Management Console、AWS Lambda コンソールに移動します。 2. 関数の作成を選択し、最初から Author を選択します。 3. 関数名に、関数の名前を入力します。 4. Runtime ドロップダウンリストで、Python.3.X を選択します。 5. Change default execution role を展開し、Create a new role with basic Lambda permissions を選択します。 6. [関数の作成] を選択してください。 7. Code タブを選択し、追加情報セクションで提供されている `S3CopyLambda.py` Python コードを貼り付けます。Python コードは、Microsoft Visual Studio 統合開発環境 (IDE) で Amazon Q Developer を使用して生成されました。	移行エンジニア

タスク	説明	必要なスキル
	8. `destination_bucket_name` を以前に作成した S3 バケットの名前に、をメインフレームファイル名`change destination_file_key` に編集します。 9. Lambda 関数をデプロイします。	

タスク	説明	必要なスキル
Lambda 関数を呼び出す Amazon S3 トリガーを作成します。	Lambda 関数を呼び出すトリガーを設定するには、次の手順を実行します。 1. Lambda コンソールで 「関数ページ」を開きます。 2. Lambda 関数を選択します。 3. 関数の概要で、トリガーの追加を選択します。 4. トリガー設定ドロップダウンリストで、S3 を選択します。 5. Bucket フィールドに、ソースバケットの名前を入力します。 6. イベントタイプのドロップダウンリストで、すべてのオブジェクト作成イベントを選択します。 7. 入力と出力の両方に同じ S3 バケットを使用することは推奨されないことを承認し、追加 を選択します。 詳細については、チュートリアル: Amazon S3 トリガーを使用して Lambda 関数を呼び出すを参照してください。	移行リード

タスク	説明	必要なスキル
Lambda 関数の IAM アクセス許可を指定します。	Lambda 関数がファイル転送先とソースデータセット S3 バケットにアクセスするには、IAM アクセス許可が必要です。ファイル転送先 S3 バケットの `s3:GetObject` および `s3:DeleteObject` アクセス許可とソースデータセット S3 バケットの `s3:PutObject` アクセスを許可して 、Lambda 関数実行ロールに関連付けられたポリシーを更新しますS3。 詳細については、「チュートリアル: Amazon S3 トリガーを使用して Lambda 関数を呼び出す」の「アクセス許可ポリシーの作成」セクションを参照してください。	移行リード

メインフレームデータ転送タスクを定義する

タスク	説明	必要なスキル
メインフレームファイルを S3 バケットにコピーする転送タスクを作成します。	メインフレームファイル転送タスクを作成するには、AWS Mainframe Modernization ドキュメントの指示に従います。 **Note** ソースコードページのエンコーディングを	移行エンジニア

タスク	説明	必要なスキル
	IBM1047 として指定し、ターゲットコードページのエンコーディングを UTF-8 として指定します。	
転送タスクを確認します。	データ転送が成功したことを確認するには、AWS Mainframe Modernization ドキュメントの指示に従います。メインフレームファイルがファイル転送先 S3 バケットにあることを確認します。	移行リード
Lambda コピー関数を確認します。	Lambda 関数が開始され、ファイルがソースデータセット S3 バケットに .csv 拡張子でコピーされていることを確認します。 Lambda 関数によって作成された .csv ファイルは、QuickSight の入力データファイルです。データの例については、添付ファイルセクションの `Sample-data-member-healthcare-APG` ファイルを参照してください。	移行リード

QuickSight をメインフレームデータに接続する

タスク	説明	必要なスキル
QuickSight をセットアップします。	QuickSight を設定するには、AWS ドキュメントの指示に従います。	移行リード
QuickSight のデータセットを作成します。	QuickSight のデータセットを作成するには、AWS ドキュメントの指示に従います。入力データファイルは、メインフレームデータ転送タスクを定義したときに作成された変換されたメインフレームファイルです。	移行リード

QuickSight で Amazon Q を使用してメインフレームデータからビジネスインサイトを取得する

タスク	説明	必要なスキル
QuickSight で Amazon Q を設定します。	この機能には Enterprise Edition が必要です。QuickSight で Amazon Q を設定するには、以下を実行します。 1. Amazon Q アドオンを取得するには、AWS ドキュメントの「ステップ 1: Q アドオンを取得する」の手順に従います。 2. Amazon Q で生成 BI 機能を使用するには、ユーザーのアカウントをアップグレードします。AWS ドキュメ	移行リード

タスク	説明	必要なスキル
	ントの指示に従ってください。 3. 前に作成したデータセットを使用して Amazon Q トピックを作成します。AWS ドキュメントの指示に従ってください。 4. 自然言語がわかりやすいようにトピックメタデータを設定するには、 AWS ドキュメントの指示に従います。	

タスク	説明	必要なスキル
メインフレームデータを分析し、ビジュアルダッシュボードを構築します。	QuickSight でデータを分析および視覚化するには、以下を実行します。 1. メインフレームデータ分析を作成するには、 AWS ドキュメントの指示に従います。データセットで、前のステップで作成したデータセットを選択します。 2. 分析ページで、ビジュアルの構築を選択します。 3. 分析用のトピックの作成ウィンドウで、既存のトピックの更新を選択します。 4. トピックの選択ドロップダウンリストで、前に作成したトピックを選択します。 5. トピックリンクを選択します。 6. トピックをリンクしたら、ビジュアルの構築を選択して Amazon Q Build a Visual ウィンドウを開きます。 7. プロンプトバーに、分析の質問を書き込みます。このパターンに使用される質問の例を次に示します。 • リージョン別にメンバーディストリビューションを表示する	移行エンジニア

タスク	説明	必要なスキル
	• 年齢別のメンバーディストリビューションを表示する • 性別でメンバーディストリビューションを表示する • プランタイプ別にメンバーディストリビューションを表示する • メンバーの予防イミュナイゼーションが完了していないことを示す 質問を入力したら、ビルドを選択します。QuickSight の Amazon Q はビジュアルを作成します。 8. ビジュアルダッシュボードにビジュアルを追加するには、分析に追加を選択します。 完了したら、ダッシュボードを公開して、組織内の他のユーザーと共有できます。例については、「追加情報」セクションの「メインフレームビジュアルダッシュボード」を参照してください。	

メインフレームデータから QuickSight で Amazon Q を使用してデータストーリーを作成する

タスク	説明	必要なスキル
データストーリーを作成します。	前の分析からのインサイトを説明するデータストーリーを作成し、メンバーの予防的免除を増やすレコメンデーションを生成します。 1. データストーリーを作成するには、 AWS ドキュメントの指示に従います。 2. データストーリープロンプトには、以下を使用します。 `Build a data story about Region with most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunization. Include 4 points of supporting data for this pattern.` また、独自のプロンプトを構築して、他のビジネスインサイトのデータストーリ	移行エンジニア

タスク	説明	必要なスキル
	ーを生成することもできます。 3. ビジュアルを追加を選択し、データストーリーに関連するビジュアルを追加します。このパターンでは、前に作成したビジュアルを使用します。 4. [Build] を選択します。 5. データストーリー出力の例については、「追加情報」セクションの「データストーリー出力」を参照してください。	
生成されたデータストーリーを表示します。	生成されたデータストーリーを表示するには、AWS ドキュメントの指示に従います。	移行リード
生成されたデータストーリーを編集します。	データストーリーのフォーマット、レイアウト、またはビジュアルを変更するには、AWS ドキュメントの指示に従います。	移行リード
データストーリーを共有します。	データストーリーを共有するには、AWS ドキュメントの指示に従います。	移行エンジニア

トラブルシューティング

問題	ソリューション
BMC を使用したファイル転送で転送タスクを作成するのデータセットの検索条件に入力されたメインフレーム AWS Mainframe Modernization ファイルまたはデータセットを検出できません。	1. まず、 AWS Mainframe Modernization Transfer with BMC コンソールでデータ転送エンドポイントを選択して接続を確認します。最後のハートビート時間が 2 分を超える場合、ファイル転送の接続は確立されていません。メインフレームで実行されているエージェントの最後のハートビート時間が 2 分未満の場合、エージェントへの接続は成功します。ステップ 2 に進みます。 2. AWS Secrets Manager 設定を確認します。シークレットキーは、Secrets Manager で、メインフレームのユーザー ID の値を持つキー `userId` (大文字 I) と、メインフレームパスワードの値`password`を持つキーを設定する必要があります。`userId` および `password`シークレットキーでは大文字と小文字が区別されるため、そのまま入力する必要があります。

関連リソース

PACKED-DECIMAL (COMP-3) や BINARY (COMP または COMP-4) などのメインフレームデータ型を QuickSight でサポートされているデータ型に変換するには、次のパターンを参照してください。

- Python を使用して で EBCDIC データを ASCII に変換および解凍 AWS する
- を使用して Amazon S3 でメインフレームファイルを EBCDIC 形式から文字区切りの ASCII 形式に変換する AWS Lambda

追加情報

S3CopyLambda.py

次の Python コードは、IDE の Amazon Q Developer でプロンプトを使用して生成されました。

```
#Create a lambda function triggered by S3. display the S3 bucket name and key
import boto3
s3 = boto3.client('s3')
def lambda_handler(event, context):
print(event)
bucket = event['Records'][0]['s3']['bucket']['name']
key = event['Records'][0]['s3']['object']['key']
print(bucket, key)
#If key starts with object_created, skip copy, print "copy skipped". Return lambda with
 key value.
if key.startswith('object_created'):
print("copy skipped")
return {
'statusCode': 200,
'body': key
}
# Copy the file from the source bucket to the destination bucket.
 Destination_bucket_name = 'm2-filetransfer-final-opt-bkt'. Destination_file_key =
 'healthdata.csv'
copy_source = {'Bucket': bucket, 'Key': key}
s3.copy_object(Bucket='m2-filetransfer-final-opt-bkt', Key='healthdata.csv',
 CopySource=copy_source)
print("file copied")
#Delete the file from the source bucket.
s3.delete_object(Bucket=bucket, Key=key)
return {
'statusCode': 200,
'body': 'Copy Successful'
}
```

メインフレームビジュアルダッシュボード

次のデータビジュアルは、分析の質問 のために QuickSight の Amazon Q によって作成されましたshow member distribution by region。

次のデータビジュアルは、質問 について QuickSight の Amazon Q によって作成されましたshow member distribution by Region who have not completed preventive immunization, in pie chart。

データストーリーの出力

次のスクリーンショットは、プロンプト用に QuickSight の Amazon Q によって作成されたデータストーリーのセクションを示しています。`Build a data story about Region with most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunization. Include 4 points of supporting data.`

この概要では、データストーリーでは、メンバー数が最も多いリージョンを選択して、イミュニゼーションの取り組みから最大の影響を与えることを推奨しています。

データストーリーは、上位 3 つのリージョンのメンバー番号の分析を提供し、Southwest を、イミュナイゼーションの取り組みに重点を置いた主要なリージョンとして指定します。

> **Note**
>
> 南西部および北東部の各リージョンには、それぞれ 8 人のメンバーがいます。ただし、Southwest には完全には普及していないメンバーが多数存在するため、イミュニゼーション率を高めるためのイニシアチブから恩恵を受ける可能性が高くなります。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Stonebranch ユニバーサルコントローラーと AWS Mainframe Modernizationを統合

制作：ベイディ・サンカラン (AWS)、ロバート・ルミュー (ストーンブランチ)、フセイン・ゴムレクシゾグル (ストーンブランチ)、パブロ・アロンソ・プリエト (AWS)

概要

このパターンは、「Stonebranch ユニバーサルオートメーションセンター (UAC) のワークロードオーケストレーション」を「Amazon Web Services AWS Mainframe Modernizationサービス」と統合する方法を説明しています。AWS Mainframe Modernizationサービスは、メインフレームアプリケーションを AWS クラウドに移行して最新化します。AWS Mainframe Modernization Replatform with Micro Focus Enterprise テクノロジーと AWS Mainframe Modernization Automated Refactor with AWS Blu Age の 2 つのパターンを提供します。

Stonebranch UAC は、リアルタイムの IT 自動化およびオーケストレーションプラットフォームです。UAC は、オンプレミスから AWS までのハイブリッド IT システム全体のジョブ、アクティビティ、ワークフローを自動化および調整するように設計されています。メインフレームシステムを使用する企業クライアントは、クラウド中心の最新のインフラストラクチャとアプリケーションに移行しつつあります。Stonebranch のツールとプロフェッショナルサービスは、既存のスケジューラーと自動化機能の AWS クラウドへの移行を容易にします。

AWS Mainframe Modernizationサービスを使用してメインフレームプログラムを AWS クラウドに移行または最新化すると、この統合を使用してバッチスケジューリングを自動化し、敏捷性を高め、メンテナンスを改善し、コストを削減できます。

このパターンでは、Stonebranch スケジューラを AWS Mainframe Modernization サービス Micro Focus Enterprise ランタイムに移行されたメインフレームアプリケーションと統合する手順を示します。このパターンは、ソリューションアーキテクト、開発者、コンサルタント、移行スペシャリスト、および移行、モダナイゼーション、運用、またはDevOpsに携わる他の人々を対象としています。

ターゲットを絞った成果

このパターンは、以下の目標となる成果を提供することに重点を置いています。

- 「Stonebranch ユニバーサルコントローラー」から「AWS Mainframe Modernizationサービス (Microfocus ランタイム)」で実行されているメインフレームバッチジョブをスケジュール、自動化、実行する機能。

- Stonebranch ユニバーサルコントローラからアプリケーションのバッチプロセスを監視します。
- Stonebranch ユニバーサルコントローラからバッチプロセスを自動または手動で開始/再開/再実行/停止します。
- AWS Mainframe Modernizationのバッチプロセスの結果を取得します。
- Stonebranch ユニバーサルコントローラーのバッチジョブの「AWS CloudWatch」ログをキャプチャします。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ジョブコントロール言語 (JCL) ファイルと AWS Mainframe Modernization サービス (Micro Focus ランタイム) 環境にデプロイされたバッチプロセスを備えた Micro Focus Bankdemo アプリケーション
- Micro Focus 「Enterprise Server」上で動作するメインフレームアプリケーションを構築してデプロイする方法に関する基本的な知識
- 「Stonebranch ユニバーサルコントローラー」の基礎知識
- ストーンブランチトライアルライセンス (「ストーンブランチ」にお問い合わせください)
- 4 コア最低、8 GB のメモリ、2 GB のディスク容量を備えた Windows または Linux Amazon Elastic Compute Cloud (Amazon EC2) インスタンス (たとえば、xlarge)
- Apache Tomcat バージョン 8.5.x または 9.0.x
- Oracle Java ランタイム環境 (JRE) または OpenJDK バージョン 8 または 11
- Amazon Aurora MySQL-Compatible Edition
- エクスポートリポジトリ用「Amazon Simple Storage Service (Amazon S3)」バケット
- 高可用性 (HA) を実現するエージェントストーンブランチユニバーサルメッセージサービス (OMS) 接続用の 「Amazon Elastic File System (Amazon EFS)」
- Stonebranch ユニバーサルコントローラー 7.2 ユニバーサルエージェント 7.2 インストールファイル
- AWS Mainframe Modernization「タスクスケジューリングテンプレート」 (.zip ファイルの最新リリースバージョン)

制約事項

- 製品とソリューションはOpenJDK 8と11でのみテストされ、互換性が検証されています。
- 「aws-mainframe-modernization-stonebranch-integration」タスクスケジューリングテンプレートは、AWS Mainframe Modernizationサービスでのみ機能します。
- このタスクスケジューリングテンプレートは、Stonebranch エージェントの UNIX、Linux、または Windows エディションでのみ機能します。
- 一部の AWS サービスは、すべての AWS リージョンで利用できるわけではありません。リージョンの可用性については、「リージョン別の AWS のサービス」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

ターゲットアーキテクチャ

次の図表は、このパイロットに必要な AWS 環境の例を示しています。

1. Stonebranch ユニバーサルオートメーションセンター (UAC) には、ユニバーサルコントローラーとユニバーサルエージェントの 2 つの主要コンポーネントがあります。Stonebranch OMS はコントローラと個々のエージェント間のメッセージバスとして使用されます。
2. Stonebranch UAC データベースはユニバーサルコントローラによって使用されます。データベースは、MySQL、Microsoft SQL Server、Oracle、または Aurora MySQL と互換性があります。
3. AWS Mainframe Modernizationサービス — 「BankDemo アプリケーションがデプロイされた」Micro Focus ランタイム環境。BankDemo アプリケーションファイルは S3 バケットに保存されます。このバケットにはメインフレーム JCL ファイルも含まれています。
4. Stonebranch UAC は、バッチ実行時に以下の機能を実行できます。
    a. AWS Mainframe Modernizationサービスにリンクされている S3 バケットにある JCL ファイル名を使用してバッチジョブを開始します。
    b. バッチジョブのステータスを実行させます。
    c. バッチジョブの実行が完了するまでお待ちください。
    d. バッチジョブ実行のログを取得します。
    e. 失敗したバッチジョブを再実行します。
    f. ジョブの実行中にバッチジョブをキャンセルします。
5. Stonebranch UAC はアプリケーションに対して以下の機能を実行できます。

a. アプリケーションの開始
b. アプリケーションのステータスを取得
c. アプリケーションが起動または停止するまでお待ちください。
d. アプリケーションの停止
e. アプリケーション操作のログを取得

ストーンブランチのジョブ変換

次の図は、近代化の過程におけるストーンブランチの転職プロセスを表しています。ジョブスケジュールとタスク定義を、AWS Mainframe Modernizationバッチタスクを実行できる互換性のある形式に変換する方法について説明します。

1. 変換プロセスでは、ジョブ定義が既存のメインフレームシステムからエクスポートされます。
2. JCL ファイルはメインフレームモダナイゼーションアプリケーションの S3 バケットにアップロードできます。これにより、これらの JCL ファイルを AWS Mainframe Modernizationサービスでデプロイできます。
3. 変換ツールは、エクスポートされたジョブ定義を UAC タスクに変換します。
4. すべてのタスク定義とジョブスケジュールが作成されると、これらのオブジェクトは Universal Controller にインポートされます。変換されたタスクは、メインフレームで実行するのではなく、AWS Mainframe Modernizationサービス内のプロセスを実行します。

ストーンブランチ UAC アーキテクチャ

以下のアーキテクチャ図は、高可用性 (HA) ユニバーサルコントローラのアクティブ-アクティブ-パッシブモデルを表しています。Stonebranch UAC は複数のアベイラビリティーゾーンにデプロイされ、高可用性を提供し、ディザスタリカバリ (DR) をサポートします。

ユニバーサルコントローラー

2 台の Linux サーバーがユニバーサルコントローラーとしてプロビジョニングされます。どちらも同じデータベースエンドポイントに接続します。各サーバーにはユニバーサルコントローラアプリケーションと OMS が格納されています。ユニバーサルコントローラーの最新バージョンは、プロビジョニング時に使用されます。

ユニバーサルコントローラーは Tomcat Web アプリにドキュメント ROOT としてデプロイされ、ポート 80 で提供されます。このデプロイにより、フロントエンドロードバランサーの設定が容易になります。

Stonebranch ワイルドカード証明書 (例：`https://customer.stonebranch.cloud`) を使用して TLS または HTTPS 経由の HTTP を有効にします。これにより、ブラウザとアプリケーション間の通信が保護されます。

OMS

ユニバーサルエージェントと OMS (Opswise メッセージサービス) は各ユニバーサルコントローラサーバに常駐しています。顧客側からデプロイされたすべてのユニバーサルエージェントは、両方の OMS サービスに接続するように設定されます。OMS はユニバーサルエージェントとユニバーサルコントローラ間の共通のメッセージングサービスとして機能します。

Amazon EFS は各サーバーにスプールディレクトリをマウントします。OMS は、この共有スプールディレクトリを使用して、コントローラやエージェントからの接続情報やタスク情報を保持します。OMS は高可用性のモードで動作します。アクティブな OMS がダウンした場合、パッシブ OMS はすべてのデータにアクセスでき、アクティブな操作を自動的に再開します。ユニバーサルエージェントはこの変更を検出し、新しいアクティブ OMS に自動的に接続します。

データベース

Amazon Relational Database Service (Amazon RDS) には、Amazon Aurora MySQL 互換のエンジンを使用する UAC データベースが格納されています。Amazon RDS は、定期的なバックアップの管理と提供に役立ちます。ユニバーサルコントローラのインスタンスは両方とも同じデータベースエンドポイントに接続します。

ロードバランサー

Application Load Balancer はインスタンスごとに設定されます。ロードバランサーはいつでもトラフィックをアクティブコントローラーに転送します。インスタンスドメイン名はそれぞれのロードバランサーエンドポイントを指します。

URL

次の例に示しているように、各インスタンスには URL があります。

環境	インスタンス

本番稼働用	`customer.stonebranch.cloud`
開発 (ノンプロダクション)	`customerdev.stonebranch.cloud`
テスト (ノンプロダクション)	`customertest.stonebranch.cloud`

> **Note**
>
> 本番稼働用以外のインスタンス名は、必要に応じて設定できます。

高可用性

ハイアベイラビリティ (HA) とは、指定された期間、システムを障害なく継続的に運用できることです。このような障害には、ストレージ、CPU やメモリの問題によるサーバー通信応答の遅延、ネットワーク接続などが含まれますが、これらに限定されません。

HA 要件を満たすには：

- EC2 インスタンス、データベース、その他の設定はすべて、同じ AWS リージョン内の 2 つの別々のアベイラビリティーゾーンにミラーリングされます。
- コントローラーは、2 つのアベイラビリティーゾーンにある 2 つの Linux サーバーで Amazon マシンイメージ (AMI) を介してプロビジョニングされます。たとえば、ヨーロッパ eu-west-1 リージョンでプロビジョニングされている場合、アベイラビリティーゾーン eu-west-1a とアベイラビリティーゾーン eu-west-1c にユニバーサルコントローラーがあります。
- アプリケーションサーバー上でジョブを直接実行することはできず、データをこれらのサーバーに保存することもできません。
- Application Load Balancer は、各ユニバーサルコントローラーのヘルスチェックを実行してアクティブなコントローラーを特定し、トラフィックをそのユニバーサルコントローラーに転送します。1 つのサーバーに問題が発生した場合、ロードバランサーはパッシブユニバーサルコントローラーを自動的にアクティブ状態に昇格させます。その後、ロードバランサーはヘルスチェックから新しいアクティブな Universal Controller インスタンスを識別し、トラフィックの転送を開始します。フェイルオーバーは 4 分以内に行われ、ジョブが失われることはありません。フロントエンド URL は変わりません。
- Aurora MySQL 互換のデータベースサービスには、ユニバーサルコントローラーのデータが保存されます。本番環境では、1 つのAWSリージョン内の 2 つの異なるアベイラビリティゾーンに

ある 2 つのデータベースインスタンスでデータベースクラスターを構築します。どちらのユニバーサルコントローラも、単一のデータベースクラスターエンドポイントを指す Java Database Connectivity (JDBC) インターフェイスを使用します。1 つのデータベースインスタンスで問題が発生した場合、データベースクラスターエンドポイントは正常なインスタンスを動的に参照します。手動による介入は必要ありません。

バックアップとパージ

Stonebranch ユニバーサルコントローラは、表に示されているスケジュールに従って古いデータをバックアップおよび消去するように設定されています。

Type	スケジュール
アクティビティ	7 日間
監査	90 日間
履歴	60 日

表示されている日付より古いBackup データは、.xml 形式にエクスポートされ、ファイルシステムに保存されます。バックアッププロセスが完了すると、古いデータはデータベースから削除され、本番インスタンスでは最大 1 年間 S3 バケットにアーカイブされます。

このスケジュールはユニバーサルコントローラーのインターフェースで調整できます。ただし、これらの時間枠を長くすると、メンテナンス中のダウンタイムが長くなる可能性があります。

ツール

AWS サービス

- AWS Mainframe Modernization サービスは、メインフレームアプリケーションを AWS クラウドネイティブなマネージドランタイム環境にモダナイズするのに役立ちます。移行とモダナイズの計画および実装に役立つツールとリソースを提供します。
- Amazon Elastic Block Store (Amazon EBS) は、Amazon EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。
- Amazon Elastic File System (Amazon EFS) は、 AWS クラウドでの共有ファイルシステムの作成と設定に役立ちます。

- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。このパターンでは Amazon Aurora MySQL–Compatible Edition を使用しています。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「Elastic Load Balancing (ELB)」は、受信するアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散します。たとえば、1 つ以上のアベイラビリティーゾーンの Amazon EC2 インスタンス、コンテナ、および IP アドレスにトラフィックを分散できます。このパターンでは、Application Load Balancer を使用します。

ストーンブランチ

- 「ユニバーサルオートメーションセンター (UAC)」は、エンタープライズワークロード自動化製品のシステムです。このパターンでは次の UAC コンポーネントを使用します。
  - Tomcat ウェブコンテナで実行される Java ウェブアプリケーションであるユニバーサルコントローラーは、ユニバーサルオートメーションセンターのエンタープライズジョブスケジューラおよびワークロードオートメーションブローカーソリューションです。Controller は、コントローラー情報を作成、モニタリング、設定するためのユーザーインターフェイスを提供します。スケジューリングロジックを処理し、ユニバーサルエージェントとの間で送受信されるすべてのメッセージを処理し、ユニバーサルオートメーションセンターの高可用性オペレーションの多くを同期します。
  - 「Universal Agent」はベンダーに依存しないスケジューリングエージェントで、レガシーおよび分散型を問わず、すべての主要なコンピューティングプラットフォーム上の既存のジョブスケジューラと連携します。z/シリーズ、I/シリーズ、UNIX、Linux、または Windows で動作する全てのスケジューラーがサポートされています。
- 「Universal Agent」はベンダーに依存しないスケジューリングエージェントで、レガシーおよび分散型を問わず、すべての主要なコンピューティングプラットフォーム上の既存のジョブスケジューラと連携します。z/シリーズ、I/シリーズ、UNIX、Linux、または Windows で動作する全てのスケジューラーがサポートされています。
- 「Stonebranch aws-メインフレーム-モダナイゼイション-stonebranch-インテグレーション AWS Mainframe Modernizationユニバーサルエクステンション」は、AWS Mainframe Modernizationプラットフォームでバッチジョブを実行、監視、再実行するための統合テンプレートです。

コード

このパターンのコードは、GitHub 内の「aws-mainframe-modernization-stonebranch-integration」リポジトリで利用できます。

エピック

Amazon EC2 へのユニバーサルコントローラとユニバーサルエージェントのインストール

タスク	説明	必要なスキル
インストールファイルをダウンロードします。	Stonebranch サーバーからインストールをダウンロードします。インストールファイルを入手するには、Stonebranch に連絡してください。	クラウドアーキテクト
EC2 インスタンスを起動します。	ユニバーサルコントローラとユニバーサルエージェントのインストールには、約 3 GB の追加スペースが必要です。そのため、インスタンスには少なくとも 30 GB のディスクスペースを確保してください。 アクセスできるようにセキュリティグループにポート 8080 を追加します。	クラウドアーキテクト
前提条件をチェックします。	インストールの前に、以下を実行します。 1. 「Java ランタイム環境のダウンロード」の説明に従って Java をインストールします。 `$ sudo yum -y update`	クラウド管理者、Linux 管理者

タスク	説明	必要なスキル

	``
	$ sudo yum install
	 java-11-amazon-cor
	retto
	``
	
	必ず、サポートされている JAVA バージョンのいずれかを使用してください。前のコマンドで java-11 がインストールされるはずです。Java のバージョンを確認し、バージョン 11 を使用していることを確認してから続行してください。
	
	2. 「Apache Tomcat のインストール」で説明されているように、次のコマンドを実行します。
	
	``
	$ sudo yum install
	 tomcat tomcat-admin-
	webapps
	$ sudo systemctl
	 enable tomcat
	$ sudo systemctl start
	 tomcat
	``
	
	3. 「Aurora MySQL DB クラスターの作成と接続」の説明に従って Amazon Aurora データベースを作成します。Amazon Aurora MySQL 互換エディションを使用します。
	
	マスターユーザー名とマスターパスワードを選択します。残りの設定はデフォル	

タスク	説明	必要なスキル
	ト値のままにしておきます。	

タスク	説明	必要なスキル

ユニバーサルコントローラーをインストールします。	1. universal-controller-7.2.0.0.tar インストールファイルを EC2 インスタンスにアップロードします。
	2. インストールファイルをtempフォルダにアーカイブ解除します。
	`$ tar -xvf universal-controller-7.2.0.0.tar`
	3. インストールスクリプトに実行権限を付与します。
	`$ chmod a+x install-controller.sh`
	4. Load Balancer Controller をインストールします。この例では、次のコマンドを使用してユニバーサルコントローラを /usr/share/tomcat にインストールします。前のステップで作成した Amazon Aurora データベースの名前を使用します。
	`$ sudo ./install-controller.sh --tomcat-dir /usr/share/tomcat/ --controller-file universal-controller-7.2.0.0-build.145.war --`
		クラウドアーキテクト、Linux 管理者

タスク	説明	必要なスキル
	dbuser admin --dbpass “****” --dbname uc --rdbms mysql --dburl jdbc:mysql://database-2-instance-1.cih63miincgy.us-east-1.rds.amazonaws.com:3306/ スクリプトの出力の最後の行は「インストール完了」になっているはずです。 5. EC2 インスタンス内の次の URL に移動します。 http://<public_ip>:8080/uc 6. ログイン画面の「ユーザー名」セクションに「ops.admin」と入力し、「パスワード」フィールドは空白のままにします。 7. ops.admin ユーザーのパスワードを設定します。	

タスク	説明	必要なスキル

ユニバーサルエージェントをインストールします。	1. sb-7.2.0.1-linux-3.10-x86_64.tar.Z インストールファイルを EC2 インスタンスにアップロードします。
	2. EC2 インスタンスにログインします。
	3. ユニバーサルエージェントインストールパッケージをアーカイブ解除します。
	`$ zcat sb-7.2.0.1-linux-3.10-x86_64.tar.Z | tar xvf -`
	4. 以下のコマンドを実行してください。
	`$ sudo ./unvinst --oms_servers 7878@localhost --oms_autostart yes --python yes`
	5. PAM ファイルを作成します。
	`$ cp /etc/pam.d/login /etc/pam.d/ucmd`
	6. ユニバーサルエージェントの自動起動を有効にします。
		クラウド管理者、Linux 管理者

タスク	説明	必要なスキル
	`$ /sbin/restorecon -v /etc/rc.d/init.d/ubrokerd`	
OMS をユニバーサルコントローラに追加します。	1. ops.admin ユーザーと一緒にユニバーサルコントローラにログインします。 2. 画面の左上隅にある「サービス」メニューを選択し、「システム」の「OMS サーバー」メニューを選択します。 3. OMS サーバーアドレス フィールドに と入力しlocalhost 、保存します。 4. OMS サーバーのステータスが [接続済み]、「セッションステータス」が「動作可能」と表示されます。	ユニバーサルコントローラー管理者

AWS Mainframe Modernizationユニバーサルエクステンションをインポートしてタスクを作成する

タスク	説明	必要なスキル
統合テンプレートをインポートします。	このステップには、「AWS Mainframe Modernizationユニバーサルエクステンション」が必要です。.zip ファイルの最新リリースバージョンがダウンロードされていることを確認します。	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	1. ops.admin ユーザーと一緒にユニバーサルコントローラにログインします。 2. 「サービス」、「統合テンプレートのインポート」に移動します。 3. 統合テンプレートの.zip ファイル (aws_mainframe_modernization_stonebranch_extension.zip) を選択し、「インポート」を選択します。 統合テンプレートをインポートすると、「利用可能なサービス」の下に「AWS Mainframe Modernizationタスク」が表示されます。	

タスク	説明	必要なスキル
解決可能な認証情報を有効にする。	1. 「サービス」、「AWS Mainframe Modernizationタスク」に移動します。 2. 右側のパネルで、必須フィールドに入力します。 • 名前：新しいメインフレームモダナイゼイションタスク • エージェント：唯一のエージェント (AGNT0001) を選択します。 「AWS Mainframe Modernizationの詳細」では： • アクション：環境を一覧表示する • AWS 認証情報：EC2 インスタンスに AWS Identity and Access Management (IAM) ロールを追加している場合は、このフィールドを空のままにしておくことができます。`AWSAccessKeyID` と`AWSSecretKey` を使用する場合は、フィールドの横にあるアイコン () を選択します。 開いた「認証情報の詳細」ウィンドウに、次の情報を入力して保存します。	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	• 名前：AWS Mainframe Modernization証明書 • ランタイムユーザ：このフィールドに AWS アクセスキー ID を記入します。 • ランタイムパスワード：このフィールドに AWS シークレットキーを記入します。 • エンドポイント： エンドポイントに正しい AWS リージョンがあることを確認してください。デフォルト: `https://m2.us-east-1.amazonaws.com` 。 • リージョン：AWS Mainframe Modernization サービスのリージョンを入力します。デフォルト: `us-east-1` 。 3. 残りのフィールドはデフォルト値のままにして、タスクを保存します。	

タスク	説明	必要なスキル
タスクを起動します。	1. 右側のパネルの上部にある「タスクを起動」を選択します。 2. 確認ウィンドウで、[起動] を選択します。その後、Universal Controller Console に次のようなメッセージが表示されます。 2022-08-24 午前 10：11 時 49 分 タスクインスタンス sys_id 1661291493634146313NC8E38DB8OZJY を使用してユニバーサルタスク「新しいメインフレームモダナイゼイションタスク」を正常に起動しました。 3. 「インスタンス」に移動します。「インスタンス」タブが表示されない場合は、右矢印を選択して右にスクロールします。 4. リスト内のタスクインスタンスのコンテキスト (右クリック) メニューを開き、「出力を取得」を選択し、「出力を取得」で「提出信」を選択します。 5. 「出力を取得」ウィンドウに、STDOUT 内の環境のリストが表示されます。	ユニバーサルコントローラー管理者

バッチジョブの開始をテストする

タスク	説明	必要なスキル
バッチジョブのタスクを作成します。	1. 「サービス」、「AWS Mainframe Modernizationタスク」に移動します。 2. 右側のパネルで、必須フィールドに入力します。 • 名前：新しいメインフレームモダナイゼイションタスク • エージェント：唯一のエージェント (AGNT0001) を選択します。 「AWS Mainframe Modernizationの詳細」では： • アクション：バッチ を開始する(またはBatch を開始してバッチジョブを実行し、AWS でタスクが完了するまで待つ) • AWS 認証情報：EC2 インスタンスに IAM ロールを追加している場合は、このフィールドを空のままにしておくことができます。`AWSAccessKeyID` と`AWSSecretKey` を使用する場合は、フィールドの横にあるアイコン () を選択します。	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	• エンドポイント： エンドポイントに正しい AWS リージョンがあることを確認してください。デフォルトは「https://m2.us-east-1.amazonaws.com」です。 • リージョン：AWS Mainframe Modernization サービスのリージョンを入力します。デフォルト: `us-east-1` 。 • アプリケーション： フィールド () の横にあるアイコンを選択し、「アプリケーション選択肢の更新」で「提出」を選択します。これにより、AWS Mainframe Modernizationサービスに接続され、アプリケーションのリストが返されます。これで、ドロップダウンリストからアプリケーションを選択できます。バッチジョブを実行するアプリケーションを選択します。 • JCL ファイル名：`RUNHELLO.jcl` • 成功または失敗を待つ： このオプションを選択すると、タスクはバッチジョブのステータスが	

タスク	説明	必要なスキル
	「成功」または「失敗」になるまで待機します。 • ポーリング間隔：各ポーリング間の時間です。 • 実行ログを取得：選択すると、バッチジョブが完了すると自動的にログが取得されます。 • ログフォーマット：印刷されるログのフォーマットです。テキスト形式でも JSON 形式でもかまいません。 3. 残りのフィールドはデフォルト値のままにして、タスクを保存します。	

タスク	説明	必要なスキル
タスクを起動します。	1. 右側のパネルの上部にある「タスクを起動」を選択します。 2. 確認ウィンドウで、[起動] を選択します。その後、Universal Controller Console に次のようなメッセージが表示されます。 2022-08-24 午前 11：11：59 タスクインスタンスsys_id <sys id>でユニバーサルタスク「メインフレームモダナイゼイション開始バッチ」を正常に起動しました。 3. 「インスタンス」に移動します。「インスタンス」タブが表示されない場合は、右矢印を選択して右にスクロールします。 4. リスト内のタスクインスタンスのコンテキスト (右クリック) メニューを開き、「出力を取得」を選択し、「出力を取得」で「提出信」を選択します。 5. 「出力を取得」ウィンドウに、STDOUT 内の環境のリストが表示されます。	ユニバーサルコントローラー管理者

複数のタスクのワークフローを作成する

タスク	説明	必要なスキル
タスクをコピーしてください。	1. コピーを作成したいタスクのコンテキスト (右クリック) メニューを開き、「コピー」を選択します。 2. 「AWS Mainframe Modernizationタスクのコピー」ウィンドウで、新しいタスクの新しい名前を「メインフレームモダナイゼーション開始Batch-「RUNAWS2」と入力します。 3. メインフレーム近代化開始Batch-「RUNAWS3」という名前を使用して、タスクをもう一度コピーします。 4. メインフレーム・モダナイゼーション・スタート・Batch-「RUNAWS4」という名前を使用して、タスクをもう一度コピーします。 5. 最後に、メインフレームモダナイゼイション開始Batch-「FOOBAR」という名前でタスクをコピーします。	ユニバーサルコントローラー管理者
タスクを更新します。	1. メインフレーム・モダナイゼーション・スタート・Batch-RUNAWS2タスクを開いて (ダブルクリック)、「JCL ファイル名」フィー	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	ルドをRUNAWS2.jcl に変更して保存します。 2. メインフレーム・モダナイゼーション・スタート・Batch-RUNAWS3 タスクを開き (ダブルクリック)、「JCL ファイル名」フィールドをRUNAWS3.jcl に変更して保存します。 3. メインフレーム・モダナイゼーション・スタート・Batch-RUNAWS4 タスクを開いて (ダブルクリック)、「JCL ファイル名」フィールドをRUNAWS4.jcl に変更して保存します。 4. メインフレーム・モダナイゼーション・スタート・Batch-FOOBARタスクを開き (ダブルクリック)、「JCL ファイル名」フィールドをMISSING.jcl ,に変更して保存します。JCL ファイル名の値が正しくないため、このタスクは失敗します。	

タスク	説明	必要なスキル
ワークフローを作成します。	1. [サービス]、[ワークフロー]に移動します。 2. 右側のパネルの「名前」フィールドに「メインフレーム・モダナイゼーション・ワークフロー」と入力し、保存します。 3. 右側のパネルで、「ワークフローを編集」を選択します。 4. 「ワークフローエディター」タブの「タスクを追加」ボタン (+)。 5. 「タスク検索」ウィンドウで「検索」を選択すると、ユニバーサルコントローラ内のすべてのタスクが表示されます。 6. メインフレーム・モダナイゼーション・スタート・Batch・タスクの横にあるアイコンをクリックし、そのアイコンを「ワークフロー・エディタ」の何もない場所にドラッグします。 7. 他のメインフレーム・モダナイゼーション・タスクについても同じ操作を繰り返し、「追加情報」セクションに示されているように配置します。 8. 「接続 ボタン」() を選択し、タスク同士を接続しま	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	す。タスクを別のタスクと接続するには、タスクの中央をクリックし、目的のタスクにドラッグします。 9. 「追加情報」セクションに示されているようにタスクを接続し、ワークフローを保存します。 10. ワークフローエディタの何もない場所を右クリックして、「ワークフローを起動」を選択し、「OK」を選択します。	

タスク	説明	必要なスキル
ワークフロー実行のステータスを確認します。	1. 左側のメニューで、「アクティビティ」を選択します。 2. ウィンドウの中央にある「開始」を選択します。 リストにタスクインスタンスのリストが表示されます。 3. リストから メインフレームモダナイゼイションワークフローを開く (ダブルクリック) するか、コンテキスト (右クリック) メニューを開いて「ワークフロータスクコマンド」、「ワークフローを表示」を選択します。 追加情報セクションに表示されているタスクが表示されます。2 つ目のタスクは、見つからない JCL ファイルを使用したために失敗することが予想されていました。	ユニバーサルコントローラー管理者

失敗したバッチジョブのトラブルシューティングと再実行

タスク	説明	必要なスキル
失敗したタスクを修正して再実行してください。	1. 失敗したタスクを開きます (ダブルクリック)。タスクのエラーが表示されます。	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	2. 失敗したタスクを修正するには 2 つのオプションがあります。 • JCL ファイル名を修正し、`FOOBAR.jcl` に設定します。 • 正しい JCL ファイル名を「JCL ファイル名 (Temp) 」に追加します。このフィールドは「JCL ファイル名」フィールドを上書きします。 このパイロットでは、2 番目のオプションを選択し、タスクインスタンスを保存します。 3. 「ワークフローモニター」で、失敗したタスクのコンテキスト (右クリック) メニューを開き、「コマンド」、「再実行」を選択します。 4. その後、すべてのタスクは正常に完了します。	

アプリケーションの開始タスクとアプリケーションの停止タスクの作成

タスク	説明	必要なスキル
「アプリケーションを開始」アクションを作成します。	1. 「サービス」、「AWS Mainframe Modernizationタスク」に移動します。	ユニバーサルコントローラー管理者

タスク	説明	必要なスキル
	2. 右側のパネルで、必須フィールドに入力します。 • 名前：メインフレーム近代化開始アプリケーション • エージェント：唯一のエージェント (AGNT0001) を選択します。 「AWS Mainframe Modernizationの詳細」では： • アクション：アプリケーションの起動 • AWS 認証情報：EC2 インスタンスに IAM ロールを追加している場合は、このフィールドを空のままにしておくことができます。`AWSAccessKeyID` と`AWSSecretKey` を使用する場合は、前に作成した認証情報を選択してください。 • エンドポイント： エンドポイントのリージョンが正しいことを確認してください。デフォルト: `https://m2.us-east-1.amazonaws.com` 。 • リージョン：AWS Mainframe Modernization	

タスク	説明	必要なスキル
	サービスのリージョンを入力します。デフォルト: `us-east-1` 。 • アプリケーション：フィールド () の横にあるアイコンを選択し、「アプリケーション選択肢の更新」で「提出」を選択します。これにより、AWS Mainframe Modernizationサービスに接続され、アプリケーションのリストが返されます。これで、ドロップダウンリストからアプリケーションを選択できます。バッチジョブを実行するアプリケーションを選択します。 • 成功または失敗を待つ：このオプションを選択すると、タスクはバッチジョブのステータスが「成功」または「失敗」になるまで待機します。 • ポーリング間隔：各ポーリング間の時間です。 • 実行ログを取得：選択すると、バッチジョブが完了すると自動的にログが取得されます。 • ログフォーマット：印刷されるログのフォーマットです。テキスト形式で	

タスク	説明	必要なスキル
	も JSON 形式でもかまいません。 3. 残りのフィールドはデフォルト値のままにして、タスクを保存します。 4. 次に、このタスクをコピーして Stop Application のタスクを作成します。名前を「メインフレームモダナイゼイションストップアプリケーション」に変更し、アクションを「アプリケーションの停止」に変更します。	

バッチキャンセル実行タスクの作成

タスク	説明	必要なスキル
「バッチをキャンセル」アクションを作成します。	1. 「サービス」、「AWS Mainframe Modernizationタスク」に移動します。 2. 右側のパネルで、必須フィールドに入力します。 • 名前：メインフレームのモダナイゼイション Batch 実行をキャンセル • エージェント：唯一のエージェント (AGNT0001) を選択します。	

タスク	説明	必要なスキル
	「AWS Mainframe Modernizationの詳細」では： • アクション： Batch実行をキャンセル • AWS 認証情報：EC2 インスタンスに IAM ロールを追加している場合は、このフィールドを空のままにしておくことができます。`AWSAccessKeyID` と`AWSSecretKey` を使用する場合は、前に作成した認証情報を選択してください。 • エンドポイント： エンドポイントのリージョンが正しいことを確認してください。デフォルト: `https://m2.us-east-1.amazonaws.com` 。 • リージョン：AWS Mainframe Modernization サービスのリージョンを入力します。デフォルト: `us-east-1` 。 • アプリケーション： フィールド () の横にあるアイコンを選択し、「アプリケーション選択肢の更新」で「提出」を選択します。これに	

タスク	説明	必要なスキル
	より、AWS Mainframe Modernizationサービスに接続され、アプリケーションのリストが返されます。これで、ドロップダウンリストからアプリケーションを選択できます。バッチジョブを実行するアプリケーションを選択します。 • 成功または失敗を待つ：このオプションを選択すると、タスクはバッチジョブのステータスが「成功」または「失敗」になるまで待機します。 • ポーリング間隔：各ポーリング間の時間です。 • 実行ログを取得：選択すると、バッチジョブが完了すると自動的にログが取得されます。 • ログフォーマット：印刷されるログのフォーマットです。テキスト形式でも JSON 形式でもかまいません。 3. 残りのフィールドはデフォルト値のままにして、タスクを保存します。	

関連リソース

- ユニバーサルコントローラー
- ユニバーサルエージェント
- LDAP 設定
- SAML シングルサインオン
- エクスプレス・コンバージョン・ツール

追加情報

ワークフローエディターのアイコン

すべてのタスクが接続されている

ワークフローステータス

Precisely からのConnect を使用して VSAM ファイルを Amazon RDS または Amazon MSK に移行およびレプリケート

プラチ・カンナ（AWS）とブーパシー・ゴパルサミー（AWS）によって作成されました

概要

このパターンは、「Connect」 from Precisely を使用して、メインフレームから AWS クラウドのターゲット環境に仮想ストレージアクセス方法 (VSAM) ファイルを移行および複製する方法を示しています。このパターンで対象となる環境には、Amazon Relational Database Service (Amazon RDS) と Amazon Managed Streaming for Apache Kafka (Amazon MSK) などがあります。Connect は、「変更データキャプチャ (CDC)」を使用してソース VSAM ファイルへの更新を継続的に監視し、これらの更新を 1 つ以上の AWS ターゲット環境に転送します。このパターンを使用して、アプリケーションのモダナイゼーションやデータ分析の目標を達成できます。たとえば、Connect を使用して VSAM アプリケーションファイルを低レイテンシーで AWS クラウドに移行したり、VSAM データを AWS データウェアハウスまたはデータレイクに移行して、アプリケーションのモダナイゼーションに必要以上の同期レイテンシーに耐えられる分析を行うことができます。

前提条件と制限

前提条件

- 「IBM z/OS V2R1」またはそれ以降
- 「z/OS 用 CICS トランザクションサーバ (CICS TS) V5.1」以降 (CICS/VSAM データキャプチャ)
- 「IBM MQ 8.0」またはそれ以降
- z/OS セキュリティ要件への準拠 (SQData ロードライブラリの APF 認可など）
- VSAM リカバリーログがオンになっています
- （オプション）「CDC ログを自動的にキャプチャする CICS VSAM リカバリバージョン（CICS VR）」
- アクティブなAWS アカウント
- レガシープラットフォームからアクセス可能なサブネットを持つ「Amazon 仮想プライベートクラウド (VPC)」
- Precisely が提供する VSAM Connect ライセンス

制約事項

- Connect は、ソース VSAM スキーマまたはコピーブックに基づくターゲットテーブルの自動作成をサポートしていません。ターゲットテーブル構造は初めて定義する必要があります。
- Amazon RDS などの非ストリーミングターゲットの場合は、Apply Engine 設定スクリプトに変換ソースからターゲットへのマッピングを指定する必要があります。
- ロギング、モニタリング、アラート機能は API を通じて実装され、完全に動作するには外部コンポーネント (Amazon CloudWatch など) が必要です。

製品バージョン

- z/OS 用 SQData 40134
- Amazon Elastic Compute Cloud (Amazon EC2) 上の Amazon マシンイメージ (AMI) 用 SQData 4.0.43

アーキテクチャ

ソーステクノロジースタック

- Job コントロール言語 (JCL)
- z/OS UNIX シェルとインタラクティブ・システム・プロダクティビティ・ファシリティ (ISPF)
- VSAM ユーティリティ (IDCAMS)

ターゲットテクノロジースタック

- Amazon EC2
- Amazon MSK
- Amazon RDS
- Amazon VPC

ターゲットアーキテクチャ

VSAM ファイルを Amazon RDS に移行する

次の図は、ソース環境 (オンプレミスのメインフレーム) では CDC エージェント/パブリッシャー、ターゲット環境 (AWS クラウド) では 「Apply Engine」を使用して、VSAM ファイルを Amazon RDS などのリレーショナルデータベースにリアルタイムまたはほぼリアルタイムで移行する方法を示しています。

この図は、次のバッチワークフローを示しています。

1. Connect は、バックアップファイルの VSAM ファイルを比較して変更を特定することでファイルへの変更をキャプチャし、その変更を口グストリームに送信します。
2. パブリッシャーはシステムログストリームのデータを消費します。
3. パブリッシャーは、キャプチャしたデータ変更を TCP/IP 経由でターゲットエンジンに伝えます。Controller デーモンは、ソースとターゲット環境の間の通信を認証します。
4. ターゲット環境のApply Engineは、パブリッシャーエージェントから変更を受け取り、リレーショナルデータベースまたは非リレーショナルデータベースに適用します。

この図は、次のオンラインワークフローを示しています。

1. Connect は、ログ複製を使用してオンラインファイルの変更をキャプチャし、キャプチャした変更を口グストリームにストリーミングします。
2. パブリッシャーはシステムログストリームのデータを消費します。
3. パブリッシャーは、キャプチャしたデータ変更を TCP/IP 経由でターゲットエンジンに伝えます。Controller デーモンは、ソースとターゲット環境の間の通信を認証します。
4. ターゲット環境のApply Engineは、パブリッシャーエージェントから変更を受け取り、リレーショナルデータベースまたは非リレーショナルデータベースに適用します。

VSAM ファイルを Amazon MSK に移行しています。

次の図は、VSAM データ構造をメインフレームから Amazon MSK にハイパフォーマンスモードでストリーミングし、Amazon MSK と統合する JSON または AVRO スキーマ変換を自動的に生成する方法を示しています。

この図は、次のバッチワークフローを示しています。

1. Connect は、CICS VR を使用するか、バックアップファイルの VSAM ファイルを比較して変更を特定することにより、変更をファイルにキャプチャします。キャプチャされた変更は口グストリームに送信されます。
2. パブリッシャーはシステムログストリームのデータを消費します。

3. パブリッシャーは、キャプチャしたデータ変更を TCP/IP 経由でターゲットエンジンに伝えます。Controller デーモンは、ソースとターゲット環境の間の通信を認証します。
4. parallel 処理モードで動作している Replicator Engine は、データをワークキャッシュ単位に分割します。
5. ワーカースレッドは、キャッシュからデータをキャプチャします。
6. データはワーカースレッドから Amazon MSK トピックに公開されます。
7. ユーザーは、コネクタを使用して Amazon MSK の変更を Amazon DynamoDB、Amazon Simple Storage Service (Amazon S3)、または Amazon OpenSearch Service などのターゲットに適用します。

この図は、次のオンラインワークフローを示しています。

1. オンラインファイル内の変更は、ログ複製を使用してキャプチャされます。キャプチャされた変更がログストリームにストリームされます。
2. パブリッシャーはシステムログストリームのデータを消費します。
3. パブリッシャーは、キャプチャしたデータ変更を TCP/IP 経由でターゲットエンジンに伝えます。Controller デーモンは、ソースとターゲット環境の間の通信を認証します。
4. parallel 処理モードで動作している Replicator Engine は、データをワークキャッシュ単位に分割します。
5. ワーカースレッドは、キャッシュからデータをキャプチャします。
6. データはワーカースレッドから Amazon MSK トピックに公開されます。
7. ユーザーは、「コネクタ」を使用して Amazon MSK からの変更を DynamoDB、Amazon S3、OpenSearch Service などのターゲットに適用します。

ツール

- 「Amazon Managed Streaming for Apache Kafka (Amazon MSK)」は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションを、構築および実行することを支援するフルマネージドサービスです。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。

エピック

ソース環境 (メインフレーム) の準備

タスク	説明	必要なスキル
Connect CDC 4.1 をインストールします。	1. 「Precisely Support チーム」に連絡して、ライセンスとインストールパッケージを入手してください。 2. サンプル JCL を使用して Connect CDC 4.1 をインストールします。手順については、Precisely ドキュメントの 「JCL を使用して Connect CDC (SQData) をインストールする」を参照してください。 3. `SETPROG APF` コマンドを実行して、Connect ロードライブラリ SQDATA.V4nnn.LOADLIB を承認します。	IBM メインフレーム開発者/管理者
zFS ディレクトリーをセットアップします。	zFS ディレクトリをセットアップするには、Precisely ドキュメントの 「zFS 変数ディレクトリ」の指示に従ってください。 **Note** コントローラーデーモンとキャプチャ/パブリッシャーエージェント設定は、z/OS UNIX Systems	IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル
	Services ファイルシステム (zFS と呼ばれます) に保存されます。コントローラーデーモン、キャプチャ、ストレージ、パブリッシャーの各エージェントには、少数のファイルを保存するための事前定義済みの zFS ディレクトリ構造が必要です。	
TCP/IP ポートを設定します。	TCP/IP ポートを設定するには、Precisely ドキュメントの「TCP/IP」ポートに記載されている指示に従ってください。 **Note** Controller デーモンには、ソースシステムに TCP/IP ポートが必要です。ポートはターゲットシステム (キャプチャされた変更データが処理される) のエンジンによって参照されます。	IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル
z/OS ログストリームを作成します。	「z/OS ログストリーム」を作成するには、Precisely ドキュメントの「z/OS システムログストリームの作成」の指示に従ってください。 **Note** Connect はログストリームを使用して、移行中にソース環境とターゲット環境間でデータをキャプチャしてストリーミングします。 z/OS LogStream を作成する JCL の例については、Precisely ドキュメントの「z/OS システムログストリームの作成」を参照してください。	IBM メインフレームデベロッパー
zFS ユーザーとスターティッドタスクの ID を識別して承認します。	RACF を使用して OMVS zFS ファイルシステムへのアクセスを許可します。JCL の例については、Precisely ドキュメントの「zFS ユーザー ID とスターティッドタスク ID の識別と承認」を参照してください。	IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル
z/OS 公開鍵/秘密鍵と認証鍵ファイルを生成します。	JCL を実行してkey pair を生成します。例としては、このパターンの追加情報セクションのキーペアの例を参照してください。 手順については、Precisely ドキュメントの「z/OS 公開鍵、秘密鍵、および認証鍵ファイルの生成」を参照してください。	IBM メインフレーム開発者/管理者

CICS VSAM ログ複製をアクティブ化し、ログストリームに添付します。	次の JCL スクリプトを実行します。
	
	`//STEP1 EXEC PGM=IDCAMS`
	`//SYSPRINT DD SYSOUT=*`
	`//SYSIN DD *`
	`ALTER SQDATA.CICS.FILEA -`
	`LOGSTREAMID(SQDATA.VSAMCDC.LOG1) -`
	`LOGREPLICATE`
		IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル

FCT を使用して VSAM ファイル回復ログを有効にします。	次のパラメータ変更を反映するようにファイル制御テーブル (FCT) を変更します。
	
	``
	Configure FCT Parms
	  CEDA ALT FILE(name)
	GROUP(groupname)
	  DSNAME(data set
	name)
	  RECOVERY(NONE|BACK
	OUTONLY|ALL)
	  FWDRECOVLOG(NO|1-9
	9)
	  BACKUPTYPE(STATIC|
	DYNAMIC)
	  RECOVERY PARAMETERS
	  RECOVery : None |
	Backoutonly | All
	  Fwdrecovlog : No |
	1-99
	  BAckuptype : Static
	| Dynamic
	``
		IBM メインフレーム開発者/管理者

パブリッシャーエージェント用に CDCZlog を設定します。	1. CDCZlog パブリッシャー CAB ファイルを作成します。 2. パブリッシュされたデータを暗号化します。 3. CDCZlog パブリッシャーランタイム JCL を準備します。	IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル
コントローラーデーモンを有効にします。	1. ISPF パネルを開き、以下のコマンドを実行して Precisely メニュー EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn' を開きます。 2. コントローラーデーモンを設定するには、メニューからオプション 2 を選択します。	IBM メインフレーム開発者/管理者
パブリッシャーを有効にします。	1. ISPF パネルを開き、以下のコマンドを実行して Precisely メニュー EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn' を開きます。 2. パブリッシャーを設定するには、メニューからオプション 3 を選択し、I を選択して挿入します。	IBM メインフレーム開発者/管理者

タスク	説明	必要なスキル
ログストリームをアクティブ化します。	1. ISPF パネルを開き、以下のコマンドを実行して Precisely メニュー `EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn'` を開きます。 2. パブリッシャーをセットアップするには、メニューからオプション 4 を選択し、I を選択して挿入します。次に、前の手順で作成したログストリームの名前を入力します。	IBM メインフレーム開発者/管理者

ターゲット環境 (AWS) の準備

タスク	説明	必要なスキル
EC2 インスタンスに Precisely をインストールします。	Amazon EC2 用の Amazon Linux AMI に Connect from Precisely をインストールするには、Precisely ドキュメントの「UNIX へのConnect CDC (SQData) のインストール」の手順に従ってください。	AWS 全般
TCP/IP ポートを開きます。	インバウンドアクセスとアウトバウンドアクセス用のコントローラーデーモンポートを含むようにセキュリティグループを変更するには、Precisely ドキュメントの	AWS 全般

タスク	説明	必要なスキル
	「TCP/IP」の指示に従ってください。	
ファイルディレクトリを作成する。	ファイルディレクトリを作成するには、Precisely ドキュメントの「ターゲット適用環境の準備」の指示に従ってください。	AWS 全般

Apply Engine の設定ファイルを作成します。	Apply Engine のワーキングディレクトリに Apply Engine 設定ファイルを作成します。次の例の設定ファイルは、ターゲットとして Apache Kafka を示しています。
	`builtin.features=SASL_SCRAM`
	`security.protocol=SASL_SSL`
	`sasl.mechanism=SCRAM-SHA-512`
	`sasl.username=`
	`sasl.password=`
	`metadata.broker.list=`
	Note
	詳細については、Apache Kafka ドキュメントの「セキュリティ」を参照してください。
		AWS 全般

タスク	説明	必要なスキル
Apply Engine 処理用のスクリプトを作成します。	Apply Engine のスクリプトを作成してソースデータを処理し、ソースデータをターゲットに複製します。詳しくは、Precisely ドキュメントの「適用エンジンスクリプトの作成」を参照してください。	AWS 全般
スクリプトを実行します。	SQDPARSE と SQDENG コマンドを使用して、スクリプトを実行します。詳細については、Precisely ドキュメントの「ZoS 用のスクリプトを解析する」 を参照してください。	AWS 全般

環境を検証します。

タスク	説明	必要なスキル
CDC 処理の対象となる VSAM ファイルとターゲットテーブルのリストを検証します。	1. レプリケーションログ、リカバリログ、FCT パラメータ、ログストリームを含む VSAM ファイルを検証します。 2. 必要なスキーマ定義、テーブルアクセス、その他の基準に従ってテーブルが作成されているかどうかなど、ターゲットデータベーステーブルを検証します。	AWS 全般、メインフレーム
Connect CDC SQData プロダクトがリンクされていることを確認します。	テストジョブを実行し、このジョブからのリターンコード	AWS 全般、メインフレーム

タスク	説明	必要なスキル
	が 0 (成功) であることを確認します。 **Note** Connect CDC SQData Apply Engine のステータスメッセージには、アクティブな接続メッセージが表示されるはずです。	

テストケースの実行と検証 (Batch)

タスク	説明	必要なスキル
メインフレームでバッチジョブを実行します。	変更した JCL を使用してバッチアプリケーションジョブを実行します。変更した JCL には、次の処理を行うステップを含めてください。 1. データファイルのバックアップを作成します。 2. バックアップファイルと変更されたデータファイルを比較し、差分ファイルを生成して、メッセージに含まれるデルタレコード数を記録します。 3. デルタファイルを z/OS ログストリームにプッシュします。	AWS 全般、メインフレーム

タスク	説明	必要なスキル
	4. JCL を実行します。JCL の例については、Precisely ドキュメンテーションの「ファイル比較キャプチャ JCL の準備」を参照してください。	
ステップログを確認します。	ログストリームをチェックして、完了したメインフレームのバッチジョブの変更データが表示されることを確認します。	AWS 全般、メインフレーム
ソースデルタ変更とターゲットテーブルの数を検証します。	レコードが集計されていることを確認するには、以下を実行します。 1. バッチ JCL メッセージからソースデルタ数を収集します。 2. Apply Engine を監視して、VSAM ファイルに挿入、更新、または削除されたレコード数のレコードレベルのカウントを確認します。 3. ターゲットテーブルにレコード数を問い合わせます。 4. さまざまなレコード数をすべて比較して集計します。	AWS 全般、メインフレーム

テストケースの実行と検証 (オンライン)

タスク	説明	必要なスキル
CICS リージョンでオンライントランザクションを実行する。	1. オンライントランザクションを実行してテストケースを検証します。 2. トランザクション実行コードを検証します (RC=0 — 成功)。	IBM メインフレームデベロッパー
ステップログを確認します。	ログストリームに特定のレコードレベルの変更が反映されていることを確認します。	AWS メインフレームデベロッパー
ターゲットデータベースの数を確認します。	Apply Engine でレコードレベルのカウントを監視します。	正確には Linux です。
ターゲットデータベースのレコード数とデータレコードを検証します。	ターゲットデータベースにクエリを実行して、レコード数とデータレコードを検証します。	AWS 全般

関連リソース

- 「VSAM z/OS」 (正確なドキュメンテーション)
- 「適用エンジン」 (Precisely ドキュメンテーション)
- 「レプリケーターエンジン」 (Precisely ドキュメンテーション)
- 「ログストリーム」 (IBM ドキュメント)

追加情報

設定ファイルの例

これは、ソース環境がメインフレーム、ターゲット環境が Amazon MSK であるログストリームの設定ファイルの例です。

```
 -- JOBNAME -- PASS THE SUBSCRIBER NAME
 -- REPORT  progress report will be produced after "n" (number) of Source records
processed.

 JOBNAME VSMTOKFK;
 --REPORT EVERY 100;
 -- Change Op has been ‘I’ for insert, ‘D’ for delete , and ‘R’ for Replace. For RDS
it is 'U' for update
 -- Character Encoding on z/OS is Code Page 1047, on Linux and UNIX it is Code Page
819 and on Windows, Code Page 1252
 OPTIONS
 CDCOP('I', 'U', 'D'),
 PSEUDO NULL = NO,
 USE AVRO COMPATIBLE NAMES,
 APPLICATION ENCODING SCHEME = 1208;

 --       SOURCE DESCRIPTIONS

 BEGIN GROUP VSAM_SRC;
 DESCRIPTION COBOL ../copybk/ACCOUNT AS account_file;
 END GROUP;

 --       TARGET DESCRIPTIONS

 BEGIN GROUP VSAM_TGT;
 DESCRIPTION COBOL ../copybk/ACCOUNT AS account_file;
 END GROUP;

 --       SOURCE DATASTORE (IP & Publisher name)

 DATASTORE cdc://10.81.148.4:2626/vsmcdct/VSMTOKFK
 OF VSAMCDC
 AS CDCIN
 DESCRIBED BY GROUP VSAM_SRC ACCEPT ALL;

 --       TARGET DATASTORE(s) - Kafka and topic name

 DATASTORE 'kafka:///MSKTutorialTopic/key'
 OF JSON
 AS CDCOUT
 DESCRIBED BY GROUP VSAM_TGT FOR INSERT;
```

```
  --       MAIN SECTION

  PROCESS INTO
  CDCOUT
  SELECT
  {
  SETURL(CDCOUT, 'kafka:///MSKTutorialTopic/key')
  REMAP(CDCIN, account_file, GET_RAW_RECORD(CDCIN, AFTER), GET_RAW_RECORD(CDCIN,
 BEFORE))
  REPLICATE(CDCOUT, account_file)
  }
  FROM CDCIN;
```

キーペアの例

JCL を実行してkey pair を生成する方法の例を次に示します。

```
//SQDUTIL EXEC PGM=SQDUTIL //SQDPUBL DD DSN=&USER..NACL.PUBLIC, //
 DCB=(RECFM=FB,LRECL=80,BLKSIZE=21200), // DISP=(,CATLG,DELETE),UNIT=SYSDA, //
 SPACE=(TRK,(1,1)) //SQDPKEY DD DSN=&USER..NACL.PRIVATE, //
 DCB=(RECFM=FB,LRECL=80,BLKSIZE=21200), // DISP=(,CATLG,DELETE),UNIT=SYSDA, //
 SPACE=(TRK,(1,1)) //SQDPARMS DD  keygen //SYSPRINT DD SYSOUT= //SYSOUT DD SYSOUT=* //
SQDLOG DD SYSOUT=* //*SQDLOG8 DD DUMMY
```

Rocket Enterprise Server と LRS PageCenterX AWS を使用してメインフレーム出力管理をモダナイズする

作成者：Shubham Roy (AWS), Abraham Rondon (Micro Focus), and Guy Tucker (Levi, Ray and Shoup Inc)

概要

メインフレームの出力管理を最新化することで、コスト削減を実現し、レガシーシステムを維持することによる技術的負担を軽減し、DevOps とAmazon Web Services (AWS) のクラウドネイティブテクノロジーによる耐障害性と敏捷性を向上させることができます。このパターンは、ビジネスクリティカルなメインフレーム出力管理ワークロードを AWS クラウドで最新化する方法を示しています。このパターンでは、Rocket Enterprise Server を、Levi、Ray & Shoup, Inc. によるモダナイズされたメインフレームアプリケーションのランタイムとして使用します。 (LRS) プリントサーバーとしての VPSX/MFI (Micro Focus Interface)、アーカイブサーバーとしての LRS PageCenterX。LRS PageCenterx は、表示、インデックス作成、検索、アーカイブを行い、ビジネスアウトプットへのアクセスを保護するためのアウトプット管理ソリューションを提供します。

このパターンは、「リプラットフォーム」のメインフレーム近代化アプローチに基づいています。メインフレームアプリケーションは「AWS Mainframe Modernization」により、Amazon Elastic Compute Cloud (Amazon EC2) に移行されます。メインフレームの出力管理ワークロードは Amazon EC2 に移行され、IBM Db2 for z/OS などのメインフレームデータベースは Amazon Relational Database Service (Amazon RDS) に移行されます。LRS Directory Integration Server (LRS/DIS) は、Microsoft Active DirectoryのAWS Directory Serviceと連携して、出力管理ワークフロー認証と認可を行います。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- メインフレームの出力管理ワークロード。
- Rocket Enterprise Server で実行されるメインフレームアプリケーションを再構築して配信する方法に関する基本的な知識。詳細については、Rocket Software ドキュメントの「Rocket Enterprise Server のデータシート」を参照してください。
- LRS クラウド印刷ソリューションとコンセプトに関する基本知識。詳細については、LRS ドキュメントの「Output Modernization」を参照してください。

- Rocket Enterprise Server のソフトウェアとライセンス。詳細については、Rocket Software にお問い合わせください。
- LRS VPSX/MFI、LRS PageCenterx、LRS/Queueおよび LRS/DIS ソフトウェアとライセンス。詳細については、LRS にお問い合わせください。LRS 製品がインストールされる EC2 インスタンスのホスト名を指定する必要があります。

> **Note**
>
> メインフレームの出力管理ワークロードの設定に関する考慮事項の詳細については、このパターンの「追加情報」セクションの「考慮事項」を参照してください。

製品バージョン

- Rocket Enterprise Server 10.0
- 「LRS VPSX/MFI」
- 「LRS PageCenterX」V1R3 以降

アーキテクチャ

ソーステクノロジースタック

- オペレーティングシステム — IBM z/OS
- プログラミング言語 – 一般的なビジネス指向言語 (COBOL)、ジョブコントロール言語 (JCL)、および顧客情報管理システム (CICS)
- データベース — IBM Db2 for z/OS、IBM 情報管理システム (IMS) データベースおよび仮想ストレージアクセス方法 (VSAM)
- セキュリティ — Resource Access Control Facility (RACF)、CA Top Secret for z/OS、Access Control Facility 2 (ACF2)
- 印刷およびアーカイブソリューション — IBM メインフレーム z/OS 出力および印刷製品 (z/OS、LRSおよび CA Deliver 用 IBM InfoPrint サーバー) およびアーカイブソリューション (CA デリバリー、ASG Mobiusまたは CA バンドル)

ソースアーキテクチャ

次の図は、メインフレームの出力管理ワークロードの一般的な現状のアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーは、COBOL で書かれた IBM CICS アプリケーション上に構築されたエンゲージメントシステム (SoE) 上でビジネストランザクションを実行します。
2. SoE はメインフレームサービスを呼び出し、IBM DB2 for z/OS などの記録システム (SoR) データベースにビジネストランザクションデータを記録します。
3. SoR は SoE からのビジネスデータを永続化します。
4. バッチジョブスケジューラーは印刷出力を生成するためにバッチ・ジョブを開始します。
5. バッチジョブはデータベースからデータを抽出します。ビジネスニーズに基づき、データをフォーマットし、請求明細書、ID カード、ローン明細書などのビジネス出力を生成します。最後に、バッチジョブは出力を出力管理にルーティングし、ビジネスニーズに基づき、出力フォーマット、公開、保存を行います。
6. 出力管理はバッチジョブからの出力を受け取ります。出力管理は、LRS PageCenterx ソリューション（このパターンで示されている）や CA View などの出力管理システム内の指定された宛先に出力を索引付け、整と公開を行います。
7. ユーザーは出力の表示、検索、取得を行うことができます。

ターゲットテクノロジースタック

- オペレーティングシステム — Amazon EC2 上で実行される Windows サーバー
- コンピューティング — Amazon EC2
- ストレージ — Amazon Elastic Block Store (Amazon EBS) と Amazon FSx for Windows File Server
- プログラミング言語 — COBOL、JCL と CICS
- データベース— Amazon RDS
- セキュリティ — AWS Managed Microsoft AD
- 印刷とアーカイブ — AWS での LRS 印刷 (VPSX) とアーカイブ (PageCenterX) ソリューション
- メインフレームランタイム環境 – Rocket Enterprise Server

ターゲットアーキテクチャ

次の図は、AWS クラウドにデプロイされるメインフレームのバッチ印刷ワークロードのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. バッチジョブスケジューラーはバッチジョブを開始して、請求明細書、ID カード、ローン明細書などの出力を生成します。
2. メインフレームバッチジョブ (Amazon EC2 にリプラットフォーム) は、Rocket Enterprise Server ランタイムを使用してアプリケーションデータベースからデータを抽出し、データにビジネスロジックを適用して、データをフォーマットします。次に、Rocket Software プリンター終了モジュール (OpenText Micro Focus ドキュメント) を使用して、データを出力先に送信します。
3. アプリケーションデータベース (Amazon RDS 上で実行する SoR) は、印刷出力用のデータを永続化します。
4. LRS VPSX/MFI プリンティングソリューションは Amazon EC2 にデプロイされ、そのオペレーションデータが Amazon EBS に保存されます。LRS VPSX/MFI は、TCP/IP ベースの LRS/キュー送信エージェントを使用して、Rocket Software JES Print Exit API を介して出力データを収集します。

   LRS VPSX/MFI は EBCDIC から ASCII への変換などデータの前処理を行います。また、IBM Advanced Function Presentation (AFP) や Xerox ラインコンディショニングデータストリーム (LCDS) などのメインフレーム専用データストリームを、プリンタコマンド言語 (PCL) や PDF などのより一般的な表示と印刷データストリームに変換するなど、より複雑なタスクも実行します。

   LRS PageCenterx のメンテナンスウィンドウの間、LRS VPSX/MFI は出力キューを保持し、出力キューのバックアップとして使用します。LRS VPSX/MFI は LRS/Queue プロトコルで LRS PageCenterx に接続し、出力を送信します。LRS/Queue はジョブの準備完了と交換完了を実行し、データ転送が確実に行われるようにします。

   注意:

   Rocket Software Print Exit から LRS/Queue および LRS VPSX/MFI でサポートされているメインフレームバッチメカニズムに渡される印刷データの詳細については、「追加情報」セクションの「データキャプチャの印刷」を参照してください。

注: LRS VPSX/MFI はプリンターフリートレベルでもヘルスチェックを実行できます。詳細は、このパターンの「追加情報」セクションの「プリンターフリートのヘルスチェック」を参照してください。

5. LRS PageCenterx 出力管理ソリューションは Amazon EC2 にデプロイされ、その運用データは Amazon FSx for Windows File Server に保存されます。LRS PageCenterX は、LRS PageCenterx にインポートされたすべてのファイルと、そのファイルにアクセスできるすべてのユーザーを一元的に管理するレポート管理システムを提供しています。ユーザーは特定のファイルコンテンツを表示しまたは複数のファイルを検索して条件に一致するものを探すことができます。

   LRS/netX コンポーネントは、LRS PageCenterx アプリケーションと他の LRS アプリケーションに共通のランタイム環境を提供するマルチスレッド Web アプリケーションサーバーです。LRS/Web Connect コンポーネントは Web サーバーにインストールし、Web サーバーから LRS/NetX Web アプリケーションサーバーへのコネクタを提供しています。

6. LRS PageCenterx はファイルシステムオブジェクト用のストレージを提供しています。LRS PageCenterx の運用データは、Amazon FSx for Windows File Server に保存されています。
7. 出力管理認証と認可は、LRS/DIS を使用して AWS が管理する Microsoft AD によって実行されます。

> **Note**
>
> ターゲットソリューションでは、通常、IBM AFP や Xerox LCDS などのメインフレームフォーマット言語に対応するためにアプリケーションを変更する必要はありません。

AWS インフラストラクチャアーキテクチャ

次の図は、メインフレームの出力管理ワークロード用で、その可用性が高く安全な AWS インフラストラクチャアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. バッチスケジューラーはバッチプロセスを開始し、高可用性 (HA) のために複数の「アベイラビリティーゾーン」にわたって Amazon EC2 にデプロイされます。

> **Note**
>
> このパターンは、バッチスケジューラの実装を対象としていません。実装の詳細については、スケジューラーのソフトウェアベンダードキュメントを参照してください。

2. メインフレームのバッチジョブ (JCL や COBOL などのプログラミング言語で書かれる) は、コアビジネスロジックを使用して、請求明細書、ID カード、ローン明細書などの印刷出力を処理し生成します。バッチジョブは、HA 用の 2 つのアベイラビリティーゾーンにわたって Amazon EC2 にデプロイされます。Rocket Software Print Exit API を使用して、印刷出力を LRS VPSX/MFI にルーティングし、データの前処理を行います。
3. LRS VPSX/MFI プリントサーバーは、HA (アクティブ/スタンバイ冗長ペア) 用の 2 つのアベイラビリティーゾーンにわたって Amazon EC2 にデプロイされます。「Amazon EBS」を運用データストアとして使用します。Network Load Balancer は LRS VPSX/MFI EC2 インスタンスのヘルスチェックを実行します。アクティブなインスタンスが異常な状態である場合、ロードバランサーは他のアベイラビリティーゾーンのホットスタンバイインスタンスにトラフィックをルーティングします。印刷リクエストは、各 EC2 インスタンスの LRS Job Queue にローカルに保持されます。障害が生じた場合、LRS サービスが印刷リクエストの処理を再開する前に、障害が生じたインスタンスを再起動する必要があります。

> **Note**
>
> LRS VPSX/MFI は、プリンターフリートレベルでヘルスチェックを実行することもできます。詳細は、このパターンの「追加情報」セクションの「プリンターフリートのヘルスチェック」を参照してください。

4. LRS PageCenterX 出力管理は、HA (アクティブ/スタンバイ冗長ペア) 用の 2 つのアベイラビリティーゾーンにわたって Amazon EC2 にデプロイされます。「Amazon FSx for Windows File Server」「を運用データストアとして使用します。アクティブなインスタンスが異常な状態である場合、ロードバランサーは LRS PageCenterx EC2 インスタンスのヘルスチェックを実行し、トラフィックを他のアベイラビリティーゾーンのスタンバイインスタンスにルーティングします。
5. 「Network Load Balancer」は、LRS VPSX/MFI サーバーを LRS PageCenterx と統合するための DNS 名を提供します。

> **Note**
>
> LRS PageCenterX は Layer 4 ロードバランサーをサポートしています。

6. LRS PageCenterx は、HA の 2 つのアベイラビリティーゾーンにデプロイされた運用データストアとして Amazon FSx for Windows File Server を使用します。LRS PageCenterx は、ファイル共有内のファイルのみを解析し、外部データベース内のファイルは解析しません。
7. AWS Managed Microsoft AD は、LRS/DIS とともに使用して、出力管理ワークフローの認証と認可を実行します。詳細については、「追加情報」セクションの「出力認証と認可の印刷」を参照してください。

ツール

サービス

- AWS Directory Service for Microsoft Active Directory により、ディレクトリ対応型ワークロードと AWS リソースが、AWS クラウドの Microsoft Active Directory を使用できるようになります。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- 「Elastic Load Balancing (ELB)」は、受信したアプリケーションまたはネットワークのトラフィックを複数のターゲットに分散させます。たとえば、1 つ以上のアベイラビリティーゾーンの Amazon EC2 インスタンス、コンテナと IP アドレスにトラフィックを分散できます。このパターンは、Network Load Balancer を使用します。
- 「Amazon FSx」は、業界標準の接続プロトコルをサポートし、AWS リージョン全体で高い可用性とレプリケーションを提供するファイルシステムを提供しています。このパターンは Amazon FSx for Windows File Server を使用します。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。

その他のツール

- 「LRS PageCenterx」ソフトウェアは、スケーラブルなドキュメントおよびレポートコンテンツ管理ソリューションを提供することで、自動インデックス作成、暗号化、高度な検索機能によってユーザーが情報から最大限の価値を引き出せるようにします。
- LRS と Rocket Software によってコード展開された LRS VPSX/MFI (Micro Focus Interface) は、Rocket Software JES スプールからの出力をキャプチャし、指定された印刷先に確実に配信します。
- LRS/キューは TCP/IP に基づいた転送エージェントです。LRS VPSX/MFI は LRS/キューを使用して、Rocket Software JES Print Exit プログラミングインターフェイスを介して印刷データを収集またはキャプチャします。
- LRS ディレクトリ統合サーバー (LRS/DIS) は、印刷ワークフローにおける認証と認可に使用されます。
- Rocket Enterprise Server は、メインフレームアプリケーション用のアプリケーションデプロイ環境です。これは、任意のバージョンの Rocket Enterprise Developer を使用して移行または作成されたメインフレームアプリケーションのランタイム環境を提供します。

エピック

Rocket ランタイムをセットアップし、メインフレームバッチアプリケーションをデプロイする

タスク	説明	必要なスキル
ランタイムを設定し、デモアプリケーションをデプロイします。	Amazon EC2 で Rocket Enterprise Server をセットアップし、Rocket Software BankDemo デモアプリケーションをデプロイするには、AWS Mainframe Modernization ユーザーガイドの指示に従います。 BankDemo アプリケーションは、印刷出力を作成して開始するメインフレームのバッチアプリケーションです。	クラウドアーキテクト

Amazon EC2 で LRS プリントサーバーを設定

タスク	説明	必要なスキル
Amazon EC2 Windows インスタンスを作成します。	Amazon EC2 Windows インスタンスを起動するには、Amazon EC2 ドキュメント」の「Amazon EC2 インスタンスを起動する」の手順に従います。 Amazon EC2 LRS 製品ライセンスに使用されたホスト名と同じものを使用してください。 インスタンスは、LRS VPSX/MFI に対する次のハードウェア要件とソフトウェア要件を満たしている必要があります。 • CPU – Dual Core • RAM — 16 GB • ドライブ — 500 GB • 最小限の EC2 インスタンス — `m5.xlarge` • OS – Windows • ソフトウェア — Internet Information Services (IIS) または Apache **Note** 前述のハードウェアとソフトウェアの要件は、小規模なプリンターフリート (約	クラウドアーキテクト

タスク	説明	必要なスキル
	500-1000「」) を対象としています。すべての要件については、LRS と AWS の担当者にお問い合わせください。 1. Windows インスタンスを作成するときは、EC2 ホスト名が LRS 製品ライセンスに使用されているホスト名と同じであることを確認します。 2. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスへのConnect」の指示に従って EC2 インスタンスに接続します。 3. Windows の [スタート] メニューで、[Server Manager] を見つけて開きます。 4. Server Managerで、ダッシュボード、クイックスタート、ロールと機能の追加とサーバーロールを順に選択します。 5. サーバーロールで WebServer (IIS) とアプリケーション開発を順に選択します。	

タスク	説明	必要なスキル
	6. アプリケーション開発で、CGI チェックボックスを選択します。 7. CGI をインストールするには、Windows Server Manager のロールと機能の追加ウィザードの指示に従ってください。 8. LRS/Queue 通信用のEC2インスタンスの Windows ファイアウォールのポート 5500 を開きます。	

タスク	説明	必要なスキル
EC2 インスタンスに LRS VPSX/MFI をインストールします。	1. EC2 インスタンスに接続します。 2. 受信した LRS 電子メールから製品のダウンロードページへのリンクを開きます。 **Note** LRS 製品は、電子ファイル転送 (EFT) によって配布されます。 3. LRS VPSX/MFI をダウンロードし、ファイル (デフォルトフォルダー:c:\LRS) を解凍します。 4. LRS VPSX/MFI をインストールするには、解凍したフォルダーから LRS 製品インストーラーを起動します。 5. 機能を選択メニューで VPSX® Server を選択し、次へを選択してインストールプロセスを開始します。インストールが完了すると、正常に終了したことを示すメッセージが表示されます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/Queue をインストールします。	1. Rocket Enterprise Server EC2 インスタンスに接続します。 2. 受信したはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/Queue をダウンロードし、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS 製品インストーラーを起動して LRS/Queue をインストールします。 4. LRS 製品インストーラーの指示に従い、インストールプロセスを完了します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS をインストールします。	LLRS/DIS製品は通常、LRS VPSXインストールに含まれています。ただし、LRS/DIS が LRS VPSX と一緒にインストールされなかった場合は、次の手順に従い、インストールしてください。 1. LRS VPSX/MFI EC2 インスタンスに接続します。 2. 受信したはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/DIS をダウンロードして、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、 [LRS Product Installer] を起動します。 4. LRS Product Installer で LRS Misc Tools を展開し、LRS DIS を選択して、次へを選択します。 5. LRS Product Installer の残りの指示に従い、インストールプロセスを完了します。	クラウドアーキテクト

タスク	説明	必要なスキル
ターゲットグループを作成します。	「Network Load Balancer のターゲットグループを作成する」の指示に従って、ターゲットグループを作成します。ターゲットグループを作成したら、LRS VPSX/MFI EC2 インスタンスをターゲットとして登録します。 1. グループ詳細の指定ページで、ターゲットの種類とインスタンスを順に選択します。 2. プロトコルには [TCP] を選択します。 3. ポートは、5500 を選択します。 4. ターゲットの登録ページの使用可能なインスタンスセクションで、LRS VPSX/MFI EC2 インスタンスを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
Network Load Balancer を作成します。	Network Load Balancer を作成するには、「Elastic Load Balancing ドキュメント」の指示に従ってください」。Network Load Balancer は、Rocket Enterprise Server から LRS VPSX/MFI EC2 インスタンスにトラフィックをルーティングします。 Network Load Balancer を作成するときは、リスナーとルーティングページで次の値を選択します 1. [プロトコル] で [TCP] を選択します。 2. ポートは 5500 を選択します。 3. デフォルトアクションには、先ほど作成したターゲットグループに対して [転送先] を選択します。	クラウドアーキテクト

Rocket Enterprise Server を LRS/Queue および LRS VPSX/MFI と統合する

タスク	説明	必要なスキル
LRS/キュー統合用に Rocket Enterprise Server を設定します。	1. Amazon EC2 ドキュメントの手順に従って、Rocket Enterprise Server Amazon EC2 インスタンスに接続します。	クラウドアーキテクト

タスク	説明	必要なスキル
	2. Windows スタートメニューで、Rocket Enterprise Server 管理 UI を開きます。 3. メニューバーでNATIVE を選択します。 4. ナビゲーションペインでディレクトリサーバーを選択し、次に Enterprise Server リージョンで BANKDEMO を選択します。 5. 左側のナビゲーションペインの全般から追加のセクションまでスクロールし、LRSQ を指すように環境変数 (`LRSQ_ADDRESS` 、`LRSQ_PORT` 、`LRSQ_COMMAND`) を設定します。 • [LRSQ_ADDRESS] には、前に作成した Network Load Balancer の IP アドレスまたは DNS 名を入力します。 • LRSQ_PORT には、「VPSX LRSQ Listener Port (5500)」を入力します。 • [LRSQ_COMMAND] には、LRSQ 実行ファイルのパスロケーションを入力します。	

タスク	説明	必要なスキル
	Note LRS は現在、DNS 名の最大文字数 50 をサポートしています。DNS 名が 50 文字を超える場合は、代わりにNetwork Load Balancer の IP アドレスを使用できます。	

タスク	説明	必要なスキル
LRS VPSX/MFI 統合用に Rocket Enterprise Server を設定します。	1. LRS VPSX/MFI インストーラから にある Rocket Enterprise Server の場所にVPSX_MFI_R2 フォルダをコピーしますC\BANKDEMO\print 。 2. Amazon EC2 ドキュメントの手順に従って、Rocket Enterprise Server Amazon EC2 インスタンスに接続します。 3. Windows スタートメニューで、Rocket Enterprise Server 管理 UI を開きます。 4. メニューバーで [印刷] を選択します。 5. ナビゲーションペインで、[ディレクトリサーバー]、[BankDemo] の順に選択します。 6. [BANKDEMO] で [JES] を選択します。 7. [JES プログラムパス] に、C\BANKDEMO\print からの DLL(VPSX_MFI_R2) パスを追加します。	クラウドアーキテクト

印刷キューと印刷ユーザーを設定

タスク	説明	必要なスキル
Rocket Software Print Exit モジュールを Rocket Enterprise Server バッチプリンターのサーバー実行プロセスに関連付けます。	1. Amazon EC2 ドキュメントの手順に従って、Rocket Enterprise Server Amazon EC2 インスタンスに接続します。 2. Windows のスタートメニューで、Rocket Focus Enterprise Server Administration UI を開きます。 3. メニューバーで [印刷] を選択します。 4. ナビゲーションペインで、[ディレクトリサーバー]、[BankDemo] の順に選択します。 5. BANKDEMO で JES を選択し、プリンターまでスクロールします。 6. プリンターで、Rocket Software Print Exit モジュール (LRSPRTE6 for Batch) を Rocket Enterprise Server バッチプリンターのサーバー実行プロセス (SEP) に関連付けます。これにより、LRS VPSX/MFI への印刷出力ルーティングを可能にします。 設定の詳細については、「OpenText Micro Focus ドキュメント」の「Using	クラウドアーキテクト

タスク	説明	必要なスキル
	the Exit」を参照してください。	
LRS VPSX/MFI で印刷出力キューを作成し、LRS PageCenterX と統合します。	1. LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューで VPSX ウェブインターフェースを開きます。 3. ナビゲーションペインで [プリンタ] を選択します。 4. [追加] を選択し、[プリンタの追加] を選択します。 5. プリンタ設定ページでプリンタ名にローカルを入力します。 6. [VPSX ID] には、VPS1 を入力します。 7. [CommType] には、[TCPIP/LRSQ] を選択します。 8. ホスト/IP アドレスには、LRS PageCenterx EC2 インスタンスの前にある Network Load Balancer の IP アドレスを入力します。 9. リモートポートは 5800 を入力します。 10.リモートキューには、出力を保存する LRS PageCenterx ドキュメントフォルダーの名前を入力します。 11[追加] を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS VPSX/MFI でプリントユーザーを作成します。	1. LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューで VPSX ウェブインターフェースを開きます。 3. ナビゲーションペインで [セキュリティ]、[ユーザー] の順に選択します。 4. ユーザー名列で admin とコピーを順に選択します。 5. ユーザープロファイル管理ウィンドウのユーザー名に、ユーザー名 (たとえば、PrintUser) を入力します。 6. 説明には、簡単な説明 ([テストプリント用ユーザー] など) を入力します。 7. [Update] (更新) を選択します。これにより、印刷ユーザー (PrintUser など) を作成します。 8. ナビゲーションペインで、ユーザーで、作成した新しいユーザーを選択します。 9. コマンドメニューでセキュリティを選択します。 10.セキュリティルールページで、該当するプリンタセキュリティとジョブセキュリティオプションをすべて	クラウドアーキテクト

タスク	説明	必要なスキル
	選択して、保存を選択します。 11新しいプリントユーザーを管理者グループに追加するには、ナビゲーションペインでセキュリティを選択して、設定を選択します。 12.セキュリティ設定ウィンドウで、新しいプリントユーザーを管理者列に追加します。	

Amazon EC2 で LRS PageCenterx サーバーを設定

タスク	説明	必要なスキル
Amazon EC2 Windows インスタンスを作成します。	Amazon EC2 ドキュメントの「ステップ 1: インスタンスを起動する」の次の指示に従い、Amazon EC2 Windows インスタンスを起動します。LRS 製品ライセンスに使用したホスト名と同じものを使用してください。 インスタンスは、LRS PageCenterx のため次のハードウェア要件とソフトウェア要件を満たす必要があります。 • CPU – Dual Core • RAM — 16 GB • ドライブ — 500 GB	クラウドアーキテクト

タスク	説明	必要なスキル
	• 最小限の EC2 インスタンス — `m5.xlarge` • OS – Windows • ソフトウェア — IIS または Apache **Note** 上記のハードウェアとソフトウェアの要件は、小規模なプリンターフリート (約 500 ~ 1000) を対象としています。すべての要件については、LRS と AWS の担当者にお問い合わせください。 1. Windows インスタンスを作成するときは、EC2 ホスト名が LRS 製品ライセンスに使用されているホスト名と同じであることを確認してください。 2. 「Amazon EC2 ドキュメント」の指示に従い、EC2 インスタンスに接続します。 3. Windows の [スタート] メニューで、[Server Manager] を見つけて開きます。	

タスク	説明	必要なスキル
	4. Server Manager で、ダッシュボード、クイックスタート、ロールと機能の追加とサーバーロールを順に選択します。 5. サーバーロールで WebServer (IIS) を選択して、アプリケーション開発を選択します。 6. アプリケーション開発で、CGI チェックボックスを選択します。 7. Windows Server Manager のロールと機能の追加ウィザードの指示に従い、CGI をインストールします。 8. EC2 インスタンスの Windows ファイアウォールで、インバウンド TCP/IP トラフィック用のポート 5800 を開きます。LRS VPSX は 5800 ポートの TCPIP/LRSQ プロトコルを通じて LRS PageCenterx と通信します。	

タスク	説明	必要なスキル
EC2 インスタンスに LRS PageCenterX をインストールします。	1. EC2 インスタンスに接続します。 2. 受信した LRS 電子メールから製品のダウンロードページへのリンクを開きます。 **Note** LRS 製品は、電子ファイル転送 (EFT) によって配布されます。 3. LRS PageCenterX をダウンロードして、ファイルを解凍します (デフォルトフォルダー：`c:\LRS`)。 4. 解凍したフォルダーから LRS Product Installerを起動して、LRS PageCenterX をインストールします。 5. Select Features メニューで PageCenterX と次へを順に選択してインストールプロセスを開始します。インストールが完了すると、正常に終了したことを示すメッセージが表示されます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS をインストールします。	LLRS/DIS製品は通常、LRS VPSXインストールに含まれています。ただし、LRS/DIS が LRS VPSX と一緒にインストールされなかった場合は、次の手順に従い、インストールしてください。 1. LRS PageCenterx EC2 インスタンスに接続します。 2. 受信したはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/DIS をダウンロードして、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS Product Installer を起動します。 4. LRS Product Installer で LRS Misc Tools を展開し、LRS DIS を選択して、次へを選択します。 5. LRS Product Installer の残りの指示に従い、インストールプロセスを完了します。	クラウドアーキテクト

タスク	説明	必要なスキル
ターゲットグループを作成します。	「Network Load Balancer のターゲットグループを作成する」の指示に従い、ターゲットグループを作成します。ターゲットグループを作成したら、LRS PageCenterx EC2 インスタンスをターゲットとして登録します。 1. グループ詳細の指定ページのターゲットの種類を選択して、インスタンスを選択します。 2. プロトコルには [TCP] を選択します。 3. ポートは 5800 を選択します。 4. ターゲットの登録ページで使用可能なインスタンスセクションで、LRS PageCenterx EC2 インスタンスを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
Network Load Balancer を作成します。	Network Load Balancer を作成するには、「Elastic Load Balancing ドキュメント」の指示に従ってください」。Network Load Balancer は、LRS VPSX/MFI からのトラフィックを LRS PageCenterx EC2 インスタンスにルーティングします。 Network Load Balancer を作成するときは、リスナーとルーティングページで次の値を選択します 1. [プロトコル] で [TCP] を選択します。 2. ポートは 5800 を選択します。 3. デフォルトアクションには、先ほど作成したターゲットグループに対して [転送先] を選択します。	クラウドアーキテクト

LRS PageCenterX に出力管理機能を設定します。

タスク	説明	必要なスキル
LRS PageCenterX のインポート機能を有効にします。	LRS PageCenterx インポート機能を使用すると、LRS PageCenterx に表示される出力をJob 名やフォーム ID などの基準で認識できます。その後、出力を LRS PageCente	クラウドアーキテクト

タスク	説明	必要なスキル
	rx の特定のフォルダーにルーティングできます。 インポート機能を有効にするには、次の手順を実行します。 1. 「Amazon EC2 ドキュメント」の以下の指示に従い、LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX Web Interface を開きます。 3. フォルダーエクスプローラーで管理者を選択します。 4. 設定ページで、詳細設定、パラメーターのインポートを選択します。 5. パラメーターのインポートセクションで、高度なインポートチェックボックスを選択します。 6. 変更をコミットするには、「更新」を選択します。	

タスク	説明	必要なスキル
ドキュメント保存ポリシーを設定します。	LRS PageCenterx はドキュメント保存ポリシーにより、LRS PageCenterx にドキュメントを保存する期間を決定します。 ドキュメント保存ポリシーを設定するには、以下を実行します。 1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューでPCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで管理者を選択します。 4. 管理者ページでアーカイブグループリスト/一般管理者を選択して、保持ポリシーを選択します。 5. 保持ポリシーセクションで追加を選択して保持ポリシーを作成します。 6. 保持ポリシー情報ページで、保持ポリシー名、説明とキュメント保持期間を入力します。 7. 変更を保存してポリシーを作成するには、[OK] を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
出力ドキュメントを LRS PageCenterx の特定のフォルダーにルーティングするルールを作成します。	LRS PageCenterx では、レポート定義がこの送信先を呼び出したときに、出力が送信されるフォルダパスが決定されます この例では、レポート定義の Form ID フォルダーに基づき、フォルダーを作成し、出力をそのフォルダーに保存します。 1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで管理者、事前インポート、宛先を選択します。 4. 送信先セクションで追加を選択し、送信先メンテナンスフォームを開きます。 5. 送信先管理] ォームに、次の値を入力します。 • 送信先名 — フォーム • 説明 — 送信先の説明 (フォームベースのフォルダー構造など) • Destination type — フォルダー • フォルダパラメータ — フォルダパスをインポート (ドキュメントが到着	クラウドアーキテクト

タスク	説明	必要なスキル
	したときに PageCenterX に作成されるフォルダパス。たとえば、このパス /Test/&FORM/&IMPORTDATE/&IMPORTTIME では、ベース Test フォルダ、STD という名前のForm-IDに基づくサブフォルダ、インポート日に基づくサブフォルダ、インポート日に基づくサブフォルダ、インポート時刻に基づくサブフォルダが作成されます) • ドキュメント名 — ドキュメントがフォルダーに保存されるときにドキュメントに割り当てられる動的な名前。 6. ドロップダウンリストで、保存ポリシーを選択します。たとえば、[Year1] を選択すると、ドキュメントが 1 年間保存されます。 7. 変更を保存するには、[OK] を選択します。	

タスク	説明	必要なスキル
レポート定義を作成します。	1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで、管理者、事前インポート、レポート定義を選択して、追加を選択します。 4. レポート定義メンテナンスページの全般タブで、レポート定義名を入力します。 5. 全般タブのフィールドで、ジョブ名、フォーム、クラスと作成者などの選択条件を指定できます。たとえば、MFIDEMO のジョブ名を入力できます。ジョブ名の値は、印刷出力を生成するバッチジョブの名前になります。 6. 送信先タブの使用可能な送信先で、以前に作成した送信先 (フォーム) を選択します。 7. 追加を選択し、フォームの送信先を割り当てられた送信先として追加します。	クラウドアーキテクト

タスク	説明	必要なスキル
	Note この例では、MFI DEMO によって生成され、LRS PageCenterX にルーティングされた出力が、送信先定義で定義されたフォルダ構造に保存されるレポート定義が含まれています。	

アウトプット管理の認証と認可を設定します。

タスク	説明	必要なスキル
ユーザーとグループを持つ AWS Managed Microsoft AD ドメインを作成します。	1. AWS マネージド Microsoft AD にディレクトリを作成するには、「AWS マネージド Microsoft AD ディレクトリの作成」の指示に従ってください。 2. EC2 インスタンス (Active Directory Manager) をデプロイし、Active Directory ツールをインストールして AWS マネージド Microsoft AD を管理するには、「ステップ 3: EC2 インスタンスをデプロイして Managed Microsoft AD を管理する」の指示に従ってください。	クラウドアーキテクト

タスク	説明	必要なスキル
	3. 「Amazon EC2 ドキュメント」の指示に従い、EC2 インスタンスに接続します。 **Note** EC2 インスタンスに接続するときに、Windows セキュリティウィンドウで、ステップ 1 で作成したディレクトリの管理者認証情報を入力します。 4. Windows スタートメニューから [Windows 管理ツール] を選択し、[Active Directory ユーザーとコンピュータ] を選択します。 5. Active Directory ドメインにプリントユーザーを作成するには、「ユーザーの作成」の指示に従ってください。	
EC2 インスタンスを AWS Managed Microsoft AD ドメインに参加させます。	LRS VPSX/MFI および LRS PageCenterX EC2 インスタンスを AWS マネージド Microsoft AD ドメインに「自動」(AWS ナレッジセンターのドキュメント) または「手動」(AWS Directory Service ドキュメント) で結合します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS PageCenterx EC2 インスタンス用に LRS/DIS を AWS Managed Microsoft AD と設定して統合します。	1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューでPCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで [管理者] を選択します。 4. 設定ページで、 [セキュリティパラメータ] セクションの [セキュリティタイプ] で、LRS/DIS を選択します。 5. セキュリティパラメータセクションの残りのオプションの設定を入力します。 6. Windows のスタートメニューで PageCenterx フォルダを開き、サーバー起動を選択し、サーバー停止を選択します。 7. アクティブディレクトリのユーザー名とパスワードで LRS PageCenterx にログインします。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS VPSX から LRS PageCenterX に出力をインポートするようにインポートグループを設定します。	1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで、管理者、セキュリティ管理者、グループを選択します。 4. グループセクションで追加を選択し、グループ設定フォームを開きます。 5. グループ設定フォームで、グループ名と説明に値を入力します。 6. 汎用オプションを展開し、ンポートチェックボックスを選択します。 7. 変更を保存するには、[OK] を選択します。	クラウドアーキテクト
セキュリティルールをインポートグループに追加します。	1. インポートグループのコンテキスト (右クリック) メニューを開きます。 2. 詳細設定を選択して、 [セキュリティ] を選択します。 3. セキュリティセクションでインポートを選択して、サブフォルダーチェックボックスを選択します。 4. 変更を保存するには、[適用] を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS PageCenterX にユーザーを作成して、LRS VPSX/MFI からの出力インポートを実行します。	LRS PageCenterX でユーザーを作成して出力インポートを実行する場合、ユーザー名は LRS VPSX/MFI の印刷出力キューの VPSX ID と同じようにする必要があります。この例では、VPSX ID は VPS1 です。 1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで、管理者、セキュリティ管理者、ユーザーを選択します。 4. 追加を選択してユーザープロファイル管理フォームを開きます。 5. ユーザープロファイルの管理のユーザー名に VPS1 を入力します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS PageCenterX インポートユーザーをインポート専用グループに追加します。	LRS VPSX から LRS PageCenterx へのドキュメントのインポートに必要な権限を与えるには、次の操作を行います。 1. LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで PCX ウェブインターフェースを開きます。 3. フォルダーエクスプローラーで、管理者、セキュリティ管理者、グループを選択します。 4. グループセクションで、インポートのみグループのコンテキスト (右クリック) メニューを開き、詳細情報、セキュリティを選択します。 5. フォルダセキュリティレコード (ImportOnly)ページで、ユーザタブを選択します。 6. ユーザータブの名前で、ドロップダウンリストからユーザーの VPS1 を選択して、適用を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS VPSX/MFI EC2 インスタンス用に AWS Managed Microsoft AD で LRS/DIS を設定します。	1. LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューで、VPSX ウェブインターフェースを開きます。 3. ナビゲーションペインで [セキュリティ]、[設定] の順に選択します。 4. セキュリティ設定ページのセキュリティパラメータセクションのセキュリティタイプで、LRS/DIS (外部)を選択します。 5. セキュリティパラメータセクションの残りのオプションの設定を入力します。 6. Windows のスタートメニューで、LRS 出力管理フォルダーを開き、サーバー起動を選択し、サーバー停止を選択します。 7. アクティブディレクトリのユーザー名とパスワードで LRS VPSX/MFI にログインします。	クラウドアーキテクト

Amazon FSx for Windows File Server を LRS PageCenterx の運用データストアとして設定します。

タスク	説明	必要なスキル
LRS PageCenterx 用のファイルシステムを作成します。	Multi-AZ 環境で Amazon FSx for Windows File Server を	クラウドアーキテクト

タスク	説明	必要なスキル
	LRS PageCenterx の運用データストアとして使用するには、ステップ 1: ファイルシステムを作成するの指示に従ってください。	
ファイル共有を LRS PageCenterx EC2 インスタンスにマッピングします。	前のステップで作成したファイル共有を LRS PageCenterX EC2 インスタンスにマッピングするには、「ステップ 2: Windows サーバーを実行する EC2 インスタンスへファイル共有をマッピング」の指示に従います。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS PageCenterX Control DirectoryとMaster Folder Directoryを Amazon FSx ネットワーク共有ドライブにマッピングします。	1. 「Amazon EC2 ドキュメント」の指示に従い、LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで、PCX ウェブインターフェイスを開きます。 3. フォルダーエクスプローラーで、管理者、設定を選択します。 4. 設定ページでディレクトリを選択して、制御ディレクトリを選択します。 5. 制御ディレクトリに、`\\FSx file share DNS name\share\cntl` を入力します。 6. マスターフォルダディレクトリに、`\\FSx file share DNS name\share\mstr` を入力します。	クラウドアーキテクト

出力管理ワークフローのテスト

タスク	説明	必要なスキル
Rocket Software BankDemo アプリからバッチ印刷リクエストを開始します。	1. Rocket Enterprise Server EC2 インスタンスで 3270 ターミナルエミュレータを開きます。 2. `connect 127.0.0.1:9278` コマンドを実行し	テストエンジニア

タスク	説明	必要なスキル
	て BankDemo アプリに接続します。 3. BankDemo のコマンドラインインターフェイスで、[ユーザー ID] に [B0001] を入力します。パスワードには、空白以外のキーを入力します。 4. 印刷済みステートメントをリクエストオプションでは、空白行に [X] を入力します。 5. ステートメントの送信元セクションのメールに Y を入力し、F10 を押します。	

タスク	説明	必要なスキル
LRS PageCenterX の印刷出力をチェックします。	1. 「Amazon EC2 ドキュメント」の指示に従い、LRS PageCenterx EC2 インスタンスに接続します。 2. Windows のスタートメニューで、PCX ウェブインターフェイスを開きます。 3. ナビゲーションペインでテストフォルダーを開き、STD フォルダーを開き、`08-03-2023` (MM-DD-YYYY) などのジョブ実行日を含むフォルダーを開きます。 **Note** これは、「Create a rule to route the output document to a specific folder in LRS PageCenterX」で定義されているフォルダ構造と同じです。 4. `formtest-STD.txt` ファイルを開きます。 これで、勘定科目番号、説明、日付、金額および残高の列を含む取引明細書が印刷されます。例としては、このパターンの `batch_print_output`	テストエンジニア

タスク	説明	必要なスキル
	添付ファイルを参照してください。	

関連リソース

- 「LRS」
- 「Advanced Function Presentationデータストリーム」 (IBM ドキュメント)
- 「ラインコンディショニングデータストリーム (LCDS)」(Compart ドキュメント)
- 「Empowering Enterprise Mainframe Workloads on AWS with Micro Focus」 (ブログ記事)
- 「メインフレームのオンライン印刷ワークロードを AWS で最新化」 (AWS 規範ガイダンス)
- 「メインフレームのバッチ印刷ワークロードを AWS で最新化」 (AWS 規範ガイダンス)

追加情報

考慮事項

モダナイゼーションを進める中で、メインフレームのバッチプロセスとオンラインプロセスおよびそれらにより生成される出力について、さまざまな構成を検討する必要があるかもしれません。メインフレームプラットフォームは、印刷に直接影響する特定の要件に合わせて、使用するすべての顧客とベンダーによってカスタマイズされています。たとえば、現在のプラットフォームでは、IBM AFP データストリームや Xerox LCDS が現在のワークフローに組み込まれている場合があります。さらに、「メインフレームのキャリッジコントロール文字」や「チャネルコマンドワード」が印刷ページの外観に影響を与える場合があり、特別な処理が必要な場合もあります。モダナイゼーション計画プロセスの一環として、特定の印刷環境における構成を評価し、理解しておくことをお勧めします。

印刷データキャプチャ

Rocket Software Print Exit は、LRS VPSX/MFI がスプールファイルを効果的に処理するために必要な情報を渡します。この情報は、関連する制御ブロックに渡される次のようなフィールドで構成されています。

- JobName
- 所有者 (ユーザー ID)
- 送信先

- フォーム
- ファイル名
- 書き込み

LRS VPSX/MFI は、Rocket Enterprise Server からデータをキャプチャするための以下のメインフレームバッチメカニズムをサポートしています。

- 標準 z/OS JCL SYSOUT DD/OUTPUT ステートメントを使用して COBOL の印刷/スプール処理をバッチ処理します。
- 標準 z/OS JCL CA-SPOOL SUBSYS DD ステートメントを使用して COBOL 印刷/スプールをバッチ処理します。
- CBLTDLI インターフェースを使用した IMS/COBOL プリント/スプール処理。サポートされているメソッドとプログラミング例の完全なリストについては、「製品ライセンスに含まれている LRS ドキュメント」を参照してください。

プリンターフリートのヘルスチェック

LRS VPSX/MFI (LRS LoadX) は、デバイス管理や運用の最適化など、詳細なヘルスチェックを実行できます。デバイス管理では、プリンターデバイスの障害を検出し、印刷要求を正常なプリンターに転送できます。プリンターデバイスの詳細なヘルスチェックについて、詳しくは、製品ライセンスに付属の「LRSドキュメント」を参照してください。

印刷認証と認可

LRS/DISを使用すると、LRSアプリケーションはMicrosoft Active Directoryサーバーまたは Lightweight Directory Access Protocol(LDAP)サーバーを使用してユーザー ID とパスワードを認証できます LRS/DIS は、基本的な印刷認可に加えて、次のようなユースケースでも粒度レベルの印刷セキュリティ制御を適用できます。

- プリンタジョブを参照できるユーザーを管理します。
- 他のユーザーのジョブの参照レベルを管理します。
- 保留やリリース、削除、変更、コピー、ルート変更などのコマンドレベルのセキュリティなどの運用タスクを管理します。セキュリティは Active Directory セキュリティグループと LDAP グループと同様に、ユーザー ID またはグループのいずれかで設定できます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon Q Developer を使用して CardDemo メインフレームアプリケーションをモダナイズする

作成者: Santosh Kumar Singh (AWS)、Che" du Preez (AWS)

概要

Amazon Q Developer transform for mainframe は、メインフレームアプリケーションのモダナイゼーションを加速する AI を活用したエージェントです。生成 AI を使用してメインフレームのモダナイゼーションプロセスを合理化します。レガシーコード分析、メインフレームドキュメント、モノリシックアプリケーションのビジネスドメインへの分解、コードリファクタリングなどの複雑なタスクを自動化します。アプリケーション分析や移行シーケンス計画などの複雑なタスクを自動化することで、モダナイゼーションプロジェクトを加速します。モノリシックアプリケーションを分解する場合、Amazon Q Developer はメインフレームアプリケーション変換をインテリジェントにシーケンスし、ビジネス機能を並行して変換するのに役立ちます。Amazon Q Developer は、意思決定を加速し、運用の俊敏性と移行効率を高めることができます。

このパターンでは、オープンソースのメインフレームアプリケーションのサンプルである CardDemo を使用して Amazon Q Developer のメインフレーム変換機能をテストするためのstep-by-stepの手順を示します。

> **Note**
>
> Amazon Q Developer の変換機能はプレビューリリースであり、変更される可能性があります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS IAM Identity Center、有効
- 管理者が Amazon Q Developer コンソールを使用できるようにするアクセス許可
- 管理者が Amazon Q Developer 変換ウェブエクスペリエンスの接続リクエストを受け入れることを許可するアクセス許可

機能制限

- Amazon Q Developer は一部の でのみ使用できます AWS リージョン。詳細については、「Amazon Q Developer でサポートされているリージョン」を参照してください。
- メインフレームの Amazon Q Developer 変換は、コード分析、ドキュメント生成、分解のために IBM z/OS メインフレームファイルのみをサポートします。サポートされているファイルタイプのリストについては、「メインフレームアプリケーションの変換でサポートされているファイルタイプ」を参照してください。
- Amazon Q Developer のメインフレーム変換機能にはサービスクォータがあります。詳細については、「メインフレーム変換機能のサービスクォータ」を参照してください。
- 共有ワークスペースでコラボレーションするには、すべてのユーザーが、Amazon Q Developer 変換ウェブエクスペリエンスのインスタンスに関連付けられている IAM Identity Center の同じインスタンスの登録済みユーザーである必要があります。
- Amazon Simple Storage Service (Amazon S3) バケットと Amazon Q Developer サブスクリプションは同じである必要があります AWS アカウント。

アーキテクチャ

次の図は、このパターンで設定したアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. Amazon Q Developer はコネクタを使用して、Amazon S3 バケットに保存されている CardDemo メインフレームアプリケーションにアクセスします。
2. Amazon Q Developer は AWS IAM Identity Center 、 を使用してユーザーアクセスと認証を管理します。システムは、認証、認可、暗号化、アクセス管理のためのセキュリティコントロールの複数のレイヤーを実装し、処理中のコードとアーティファクトを保護します。ユーザーはチャットインターフェイスを介して Amazon Q Developer エージェントとやり取りします。入力目標と目的は英語で入力できます。
3. エージェントは、ユーザーの指示を解釈し、ジョブプランを作成し、ジョブを実行可能なタスクに分割して、自律的にアクションを実行します。ユーザーは変換を確認して承認できます。変換タスクには以下が含まれます。
    - コード分析 – Amazon Q Developer は、各ファイルのコードを分析して、ファイル名、ファイルタイプ、コード行、パスなどの詳細を確認します。エージェントはソースコードを分析し、分類を実行し、依存関係マッピングを作成し、欠落しているアーティファクトを特定します。

- ドキュメントの生成 – Amazon Q Developer はメインフレームアプリケーションのドキュメントを生成します。コードを分析することで、レガシーシステムに存在するビジネスロジック、フロー、統合、依存関係の説明など、アプリケーションプログラムの詳細なドキュメントを自動的に作成できます。
- 分解 — Amazon Q Developer は、プログラムとコンポーネント間の依存関係を考慮するドメインにコードを分解します。これにより、関連するファイルとプログラムが同じドメイン内で適切にグループ化されます。また、分解プロセス中にアプリケーションロジックの整合性を維持するのにも役立ちます。
- 移行ウェーブプランニング – 分解フェーズで作成したドメインに基づいて、Amazon Q Developer は推奨されるモダナイゼーション順序で移行ウェーブプランを生成します。
- コードリファクタリング – Amazon Q Developer は、すべてまたは選択したドメインファイルのコードを Java コードにリファクタリングします。このステップの目的は、アプリケーションの重要なビジネスロジックを保持し、モダナイズされたクラウド最適化 Java アプリケーションにリファクタリングすることです。

4. Amazon Q Developer は、リファクタリングされたコード、およびその他の関連するプランとドキュメントを Amazon S3 バケットに保存します。

ツール

AWS のサービス

- AWS IAM Identity Center は、すべての AWS アカウント およびクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理するために役立ちます。
- Amazon Q Developer は、生成 AI を活用した会話アシスタントであり、 AWS アプリケーションの理解、構築、拡張、運用に役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、データ量にかかわらず、保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

によって作成されたオープンソースの CardDemo メインフレームアプリケーション AWS は、メインフレームのモダナイゼーションを開始するのに役立ちます。

ベストプラクティス

- 小規模から始める – 小規模で複雑ではないコード (15,000 ~ 20,000 行のコード) から始めて、Amazon Q Developer がメインフレームアプリケーションを分析および変換する方法を理解します。
- 人間の専門知識と組み合わせる – Amazon Q Developer をアクセラレーターとして使用し、人間の専門知識を適用して最適な結果を実現します。
- レビューとテストを徹底する – 変換されたコードを注意深く確認し、包括的なテストを実行して、変換後の機能同等性を検証します。
- フィードバックを提供する – 改善のためのフィードバックと提案を提供するには、 のフィードバックを送信するボタンを使用する AWS Management Console か、 でケースを作成しますAWS サポート。詳細については、「サポートケースの作成」を参照してください。入力は、サービスの強化と将来の開発に役立ちます。

エピック

メインフレームアプリケーションを準備する

タスク	説明	必要なスキル
バケットを作成します。	Amazon Q Developer にサブスクライブ AWS アカウントしているのと同じ に Amazon S3 バケットを作成します。このバケットを使用してメインフレームアプリケーションを保存し、Amazon Q Developer はこのバケットを使用してリファクタリングされたコードと、変換に関連するその他のファイルを保存します。手順については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	AWS 全般

タスク	説明	必要なスキル
サンプルメインフレームアプリケーションを準備します。	1. 次のコマンドを入力して、CardDemo リポジトリをローカルワークステーションにクローンします。 `git clone https://github.com/aws-samples/aws-mainframe-modernization-carddemo.git` 2. という名前の新しいフォルダを作成しますcarddemo。 3. メインフレームソースコードを含むappフォルダを、クローンされたリポジトリから carddemoフォルダにコピーします。 4. carddemo フォルダを ZIP ファイルに圧縮します。 5. 作成した Amazon S3 バケットに ZIP ファイルをアップロードします。手順については、Amazon S3 ドキュメントの「オブジェクトのアップロード」を参照してください。	アプリ開発者、DevOps エンジニア

Amazon Q Developer を設定する

タスク	説明	必要なスキル
IAM Identity Center にユーザーを追加します。	IAM Identity Center に見込みユーザーを追加します。詳細については、IAM Identity Center ドキュメントの「Connect workforce users」を参照してください。	AWS 管理者
Amazon Q Developer Pro にユーザーをサブスクライブします。	ターゲットアカウントに Amazon Q Developer Pro を設定し、ユーザーをサブスクライブします。手順については、「Amazon Q Developer Pro にユーザーをサブスクライブする」を参照して、 AWS アカウント 使用している のタイプに対応するオプションを選択します。	AWS 管理者
Amazon Q Developer で変換機能を有効にします。	1. Amazon Q Developer を管理する AWS アカウント AWS Management Console の にサインインします。 2. Amazon Q Developer コンソールを開きます。 3. [設定] を選択します。 4. Amazon Q Developer: Transform settings セクションで、Edit を選択します。 5. Amazon Q Developer 変換を有効に し、保存を選択します。	AWS 管理者

タスク	説明	必要なスキル
	6. アプリケーション URL をコピーします。 7. 新しいブラウザタブに URL を貼り付けます。これにより、Amazon Q Developer Transform ウェブエクスペリエンスが開きます。	
変換ウェブエクスペリエンスへのユーザーアクセスを設定します。	各ユーザーは、Amazon Q Developer Transform ウェブエクスペリエンスにアクセスするには、次のステップを実行する必要があります。 1. E メールで送信された招待を受け入れます。 2. パスワードを作成する。 3. 次を実行して多要素認証を設定します。 • Amazon Q Developer Transform ウェブエクスペリエンスにログインします。 • プロンプトが表示されたら、多要素認証 (MFA) デバイスを登録します。画面上の指示に従ってください。	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
変換ウェブエクスペリエンスにログインします。	1. ターゲット AWS Management Console の にサインインします AWS アカウント。 2. Amazon Q Developer コンソールを開きます。 3. QDevTransform を選択します。これにより、Amazon Q Developer Transform ウェブエクスペリエンスが開きます。	アプリ開発者、アプリオーナー
ワークスペースを設定します。	Amazon Q Developer Pro ウェブエクスペリエンスでユーザーがコラボレーションできるワークスペースを設定します。Amazon Q Developer ドキュメントの「ワークスペースのセットアップ」の手順に従います。	AWS 管理者

メインフレームアプリケーションの変換

タスク	説明	必要なスキル
変換ジョブを作成します。	CardDemo メインフレームアプリケーションをモダナイズするための変換ジョブを作成します。手順については、Amazon Q Developer ドキュメントの「ジョブの作成と開始」を参照してください。目標の設定を求められたら、コードの分析、ド	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	キュメントの生成、コードの分解、移行シーケンスの計画、コードを Java に変換することを選択します。	
コネクタを設定します。	CardDemo メインフレームアプリケーションを含む Amazon S3 バケットを使用してコネクタを確立します。このコネクタにより、Amazon Q Developer はバケット内のリソースにアクセスし、連続する変換関数を実行できます。手順については、Amazon Q Developer ドキュメントの「コネクタのセットアップ」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
コード分析を実行します。	1. アセットの場所を指定ページで、アップロードした ZIP ファイルの Amazon S3 `carddemo` バケットパスを入力します。 2. 承認を選択して Q に送信します。 Amazon Q Developer がコードの分析を開始します。 3. Worklog タブでステータスをモニタリングします。 4. 分析が完了したら、左側のナビゲーションペインのコードの分析で、コード分析結果の表示を選択します。 5. (オプション) ダウンロードを選択して、完全なアセットリスト、欠落しているソースコード、依存関係ファイルをダウンロードします。 詳細については、Amazon Q Developer ドキュメントの「コード分析」を参照してください。	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
ドキュメントを生成します。	1. 左側のナビゲーションペインの「ドキュメントの生成」で、「ファイルの選択」を選択し、設定を行います。 2. COBOL または JCL を展開し、1 つ以上のファイルを選択します。 3. ドキュメントの詳細レベルを選択します。 • 概要 – スコープ内の各ファイルの概要を示します。また、 は各ファイルの 1 行の概要を提供します。 • 詳細な機能仕様 – メインフレームアプリケーション変換スコープ内の各ファイルの包括的な詳細を提供します。詳細には、ロジックとフロー、特定されたビジネスルール、データフロー、依存関係、入出力処理、さまざまなトランザクションの詳細などがあります。 4. 「Q に送信」を選択します。 5. Worklog タブで進行状況をモニタリングします。	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	Note ドキュメントの生成時間は、ファイルの数とコード行に応じて数分から数時間まで異なります。 6. 完了したら、ドキュメント結果の確認を選択して、Amazon S3 バケットの出力を表示します。 7. Amazon S3 バケットから zip ファイルをダウンロードし、生成されたドキュメントを確認します。 詳細については、Amazon Q Developer ドキュメントの「ドキュメントの生成」を参照してください。	

タスク	説明	必要なスキル
コードを分解します。	1. 左側のナビゲーションペインで、コードを分解 を展開し、ドメインに分解 を選択します。 2. Actions リストで、Create domain を選択します。 3. などの新しいドメインの名前を入力しますBill payment。 4. (オプション) 説明を入力します。 5. ファイルの検索バーで を検索しCB00、ファイルを選択します。 6. シードとしてマークを選択します。 7. シードフラグが「いいえ」から「はい」に変わることを確認します。 8. [作成] を選択します。 9. Actions リストで、分解の設定を選択します。 10.ドメインサイズを調整し、保存をクリックします。 11Actions リストで、Decompose を選択します。 12.ドメイン名を選択して分解出力を確認します。 13分解が完了したら、承認を選択して Q に送信します。	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	分解とシードの詳細については、Amazon Q Developer ドキュメントの「分解」を参照してください。	
移行ウェーブを計画します。	CardDemo アプリケーションの移行ウェーブを計画します。Amazon Q Developer ドキュメントの「移行ウェーブ計画」の指示に従って、ウェーブ計画を確認して編集します。	アプリ開発者、アプリオーナー
コードをリファクタリングします。	CardDemo メインフレームアプリケーションコードを、すべてまたは選択したドメインファイルで Java コードにリファクタリングします。Amazon Q Developer ドキュメントの「リファクタリングコード」の指示に従ってください。	アプリ開発者、アプリオーナー

トラブルシューティング

問題	ソリューション
次のメッセージが表示されます。 `You do not have sufficient permission on your user to administer CodeWhisperer. Ask your account administrator to provide you with the required`	Amazon Q Developer にサブスクライブし、組織内のユーザーに Amazon Q Developer へのアクセスを許可するには、管理者アクセス権が必要です。詳細については、「Amazon Q サブスクリプションコンソールを使用してアクセスと必要なポリシーを設定することを管理者に許可する」を参照してください。

問題	ソリューション
`codewhisperer:ListProfiles permission to proceed.`	
サブスクライブオプションは Amazon Q Developer Pro では使用できません。次のメッセージが表示されます。 `Unable to connect to organization instance of IAM Identity Center. Your application must be configured in the same AWS ##### as your organization instance of IAM Identity Center before you can assign users and groups.`	で AWS Management Console、IAM Identity Center が有効になってい AWS リージョン るに変更します。

関連リソース

AWS ドキュメント

- メインフレームアプリケーションの変換 (Amazon Q Developer ドキュメント)
- Amazon Q Developer: Transform for Mainframe (Amazon Q Developer ドキュメント)

その他の AWS リソース

- Amazon Q Developer 生成 AI エージェントを使用してメインフレームのモダナイゼーションを簡素化する (AWS ブログ記事)
- Amazon Q Developer FAQs
- AWS IAM Identity Center よくある質問

動画とチュートリアル

- Amazon Q Developer の概要: 変換 (AWS スキルビルダー)
- AWS re:Invent 2024 - Amazon Q Developer を使用してメインフレームアプリケーションをより迅速にモダナイズする (YouTube)

- AWS re:Invent 2024 - 移行とモダナイゼーションを自動化してトランスフォーメーションを加速する (YouTube)
- AWS re:Invent 2024 - トヨタが gen AI でイノベーションを推進し、運用効率を高める (YouTube)

Rocket Enterprise Server と LRS VPSX/MFI を使用して、 AWS でメインフレームのバッチ印刷ワークロードをモダナイズする

作成者：Shubham Roy (AWS), Abraham Rondon (Micro Focus), Guy Tucker (Levi, Ray and Shoup Inc), and Kevin Yung (AWS)

概要

このパターンは、Rocket Enterprise Server をモダナイズされたメインフレームアプリケーションのランタイムとして使用し、LRS VPSX/MFI (Micro Focus Interface) をプリントサーバーとして使用して、ビジネスクリティカルなメインフレームのバッチ印刷ワークロードをアマゾン ウェブ サービス (AWS) クラウドでモダナイズする方法を示しています。このパターンは、「リプラットフォーム」のメインフレーム近代化アプローチに基づいています。このアプローチでは、メインフレームのバッチジョブを Amazon Elastic Compute Cloud (Amazon EC2) に、メインフレームデータベース (IBM DB2 for z/OS など) を Amazon Relational Database Service (Amazon RDS) に移行します。最新の印刷ワークフローの認証と認可は、AWS Managed Microsoft AD としても知られる Microsoft Active Directory のための AWS Directory Service によって実行されます。LRS Directory Information Server (LRS/DIS) は AWS Managed Microsoft AD と統合されています。バッチ印刷ワークロードを最新化することで、IT インフラストラクチャコストの削減、レガシーシステムの維持に伴う技術的負担の軽減、データサイロの排除、DevOps モデルによる敏捷性と効率性の向上、AWS クラウドのオンデマンドリソースと自動化の活用が可能になります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- メインフレームの印刷または出力管理のワークロード
- Rocket Enterprise Server で実行されるメインフレームアプリケーションを再構築して配信する方法に関する基本的な知識 (詳細については、Rocket ドキュメントの「Rocket Enterprise Server のデータシート」を参照してください)。
- LRS クラウド印刷ソリューションと概念に関する基本的な知識
- Rocket Enterprise Server のソフトウェアとライセンス (詳細については、Rocket sales にお問い合わせください)。
- LRS VPSX/MFI、LRS/Queue、LRS/DIS のソフトウェアとライセンス (詳細については、「LRS の営業担当」にお問い合わせください)。

> **Note**
>
> メインフレームバッチ印刷ワークロードの設定に関する考慮事項の詳細については、このパターンの「追加情報」セクションの「考慮事項」を参照してください。

製品バージョン

- Rocket Enterprise Server 6.0 (製品更新 7)
- 「LRS VPSX/MFI」 V1R3 以上

アーキテクチャ

ソーステクノロジースタック

- オペレーティングシステム — IBM z/OS
- プログラミング言語 – 共通ビジネス指向言語 (COBOL)、ジョブコントロール言語 (JCL)、および顧客情報管理システム (CICS)
- データベース — IBM DB2 for z/OS および仮想ストレージアクセス方法 (VSAM)
- セキュリティ — Resource Access Control Facility (RACF)、CA Top Secret for z/OS、Access Control Facility 2 (ACF2)
- 印刷と出力管理 — IBM メインフレーム z/OS 印刷製品 (IBM Tivoli Output Manager for z/OS、LRS と CA View)

ターゲットテクノロジースタック

- オペレーティングシステム — Amazon EC2 上で実行する Microsoft Windows Server
- コンピューティング — Amazon EC2
- プログラミング言語 — COBOL、JCL と CICS
- データベース — Amazon RDS
- セキュリティ — AWS Managed Microsoft AD
- 印刷と出力管理 — AWS での LRS 印刷ソリューション
- メインフレームランタイム環境 – Rocket Enterprise Server

ソースアーキテクチャ

次の図は、メインフレームのバッチ印刷ワークロードの一般的な現状のアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーは、COBOL で書かれた IBM CICS アプリケーション上に構築されたエンゲージメントシステム (SoE) 上でビジネストランザクションを実行します。
2. SoE はメインフレームサービスを呼び出し、IBM DB2 for z/OS などの記録システム (SoR) データベースにビジネストランザクションデータを記録します。
3. SoR は SoE からのビジネスデータを永続化します。
4. バッチジョブスケジューラは、バッチジョブを開始して印刷出力を生成します。
5. バッチジョブは、データベースからデータを抽出し、ビジネス要件に基づいてデータをフォーマットしてから、請求明細書、IDカード、ローン明細書などのビジネス出力を生成します。最後に、バッチジョブは出力を印刷出力管理にルーティングし、ビジネス要件に基づいて処理と出力配信を行います。
6. 印刷出力管理は、バッチジョブからの印刷出力を受け取り、その出力を電子メール、セキュア FTP を使用するファイル共有、LRS 印刷ソリューション (このパターンで示されている) を使用する物理プリンター、IBM Tivoli などの指定された宛先に配信します。

ターゲットアーキテクチャ

次の図は、AWS クラウドにデプロイされるメインフレームのバッチ印刷ワークロードのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. バッチジョブスケジューラーはバッチジョブを開始して、請求明細書、ID カード、ローン明細書などの印刷出力を作成します。
2. メインフレームバッチジョブ (Amazon EC2 にリプラットフォーム) は、Rocket Enterprise Server ランタイムを使用してアプリケーションデータベースからデータを抽出し、データにビジネスロジックを適用してデータをフォーマットし、Rocket Software Print Exit (Micro Focus ドキュメント) を使用してデータを出力先に送信します。
3. アプリケーションデータベース (Amazon RDS 上で実行する SoR) は、印刷出力用のデータを永続化します。

4. LRS VPSX/MFI 印刷ソリューションは Amazon EC2 にデプロイされ、運用データは Amazon Elastic Block Store (Amazon EBS) に保存されます。LRS VPSX/MFI は TCP/IP ベースの LRS/キュー送信エージェントを使用して、Rocket Software JES Print Exit API を介して印刷データを収集し、指定されたプリンター宛先にデータを配信します。

> **Note**
>
> ターゲットソリューションでは、通常、IBM Advanced Function Presentation (AFP) や Xerox Line Condition Data Stream (LCDS) などのメインフレームフォーマット言語に対応するためにアプリケーションを変更する必要はありません。AWS でのメインフレームアプリケーションの移行とモダナイゼーションに Rocket Software を使用する方法の詳細については、ブログ記事「Empowering Enterprise Mainframe Workloads on AWS with Micro Focus」を参照してください。

AWS インフラストラクチャアーキテクチャ

次の図は、メインフレームのバッチ印刷ワークロードに向けた可用性と安全性に優れた AWS インフラストラクチャアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. バッチスケジューラーはバッチプロセスを開始し、高可用性 (HA) のために複数の「アベイラビリティーゾーン」にわたって Amazon EC2 にデプロイされます。

> **Note**
>
> このパターンは、バッチスケジューラの実装を対象としていません。実装の詳細については、スケジューラーのソフトウェアベンダーのドキュメントを参照してください。

2. メインフレームバッチジョブ (JCL や COBOL などのプログラミング言語で書かれている) は、コアビジネスロジックで、請求明細書、ID カード、ローン明細書などの印刷出力を処理し、生成します。ジョブは HA の 2 つのアベイラビリティーゾーンにまたがって Amazon EC2 にデプロイされ、Rocket Software Print Exit を使用して、エンドユーザーの印刷用に印刷出力を LRS VPSX/MFI にルーティングします。

3. LRS VPSX/MFI は、TCP/IP ベースの LRS/キュー送信エージェントを使用して、Rocket Software JES Print Exit プログラミングインターフェイスから印刷データを収集またはキャプチャします。Print Exit は、LRS VPSX/MFI がスプールファイルを効果的に処理し、LRS/Queue コマンドを動的に作成できるようにするために必要な情報を渡します。コマンドは、Rocket Software の標準組み込み関数を使用して実行されます。

> **Note**
>
> Rocket Software Print Exit から LRS/Queue および LRS VPSX/MFI がサポートするメインフレームバッチメカニズムに渡される印刷データの詳細については、このパターンの???「追加情報」セクションの「印刷データキャプチャ」を参照してください。

4.

> **Note**
>
> Network Load Balancer は、Rocket Enterprise Server を LRS VPSX/MFI と統合するための DNS 名を提供します。: LRS VPSX/MFI はレイヤー 4 ロードバランサーをサポートします。また、Network Load Balancer は LRS VPSX/MFI の基本的なヘルスチェックを行い、登録されている正常なターゲットにトラフィックをルーティングします。

5.

> **Note**
>
> LRS VPSX/MFI プリントサーバーは、高可用性を実現するために Amazon EC2 で 2 つのアベイラビリティーゾーン (AZ) にまたがってデプロイされており、運用データストアとして「Amazon EBS」を使用しています。LRS VPSX/MFI は、アクティブ-アクティブとアクティブ-パッシブの両方のサービスモードをサポートします。このアーキテクチャでは、アクティブ/パッシブペアの複数の AZ をアクティブスなホットスタンバイとして使用します。アクティブなインスタンスの状態が異常な場合、Network Load Balancer は LRS VPSX/MFI EC2 インスタンスのヘルスチェックを実行し、別の AZ のホットスタンバイインスタンスにトラフィックをルーティングします。印刷リクエストは、各 EC2 インスタンスの LRS Job Queue にローカルに保持されます。復旧した場合、LRS サービスが印刷リクエストの処理を再開するには、障害が発生したインスタンスを再起動する必要があります。: LRS VPSX/MFI は、プリンターフリートレベルでヘルスチェックを実行することもできます。詳細は、このパターンの [追加情報] セクションにある [プリンターフリートのヘルスチェック] を参照してください。

6. AWS Managed Microsoft AD は LRS/DIS と統合して、印刷ワークフローの認証と認可を実行します。詳細については、このパターンの???「追加情報」セクションの「認証と認可の印刷」を参照してください。
7. LRS VPSX/MFI はブロックストレージに Amazon EBS を使用しています。アクティブな EC2 インスタンスからAmazon EBS データをポイントインタイムスナップショットとして Amazon S3 にバックアップし、ホットスタンバイ EBS ボリュームに復元できます。Amazon EBS ボリュームスナップショットの作成、保持、削除を自動化するには、「Amazon Data Lifecycle Manager」で自動スナップショットの頻度を設定し、「RTO/RPO 要件」に基づいて復元することができます。

ツール

AWS サービス

- Amazon Elastic Block Store (Amazon EBS) は、EC2 インスタンスで使用するブロックレベルのストレージボリュームを提供します。EBS ボリュームの動作は、未初期化のブロックデバイスに似ています。これらのボリュームは、デバイスとしてインスタンスにマウントできます。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動できます。
- Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケーリングを容易にするウェブサービスです。リレーショナルデータベース向けに、コスト効率に優れ、サイズ変更可能な容量を提供し、一般的なデータベース管理タスクを管理します。
- AWS Managed Microsoft AD とも呼ばれる AWS Directory Service for Microsoft Active Directory を使用すると、ディレクトリ対応のワークロードと AWS リソースが AWS クラウドで Microsoft Active Directory を使用できるようになります。

その他のツール

- LRS と Rocket Software が共同開発した LRS VPSX/MFI (Micro Focus Interface) は、Rocket Enterprise Server JES スプールからの出力をキャプチャし、指定された印刷先に確実に配信します。
- LRS Directory Information Server (LRS/DIS) は、印刷ワークフロー中の認証と認可に使用されます。

- TCP/IP ベースの LRS/キュー送信エージェントは、LRS VPSX/MFI が Rocket Software JES Print Exit プログラミングインターフェイスを介して印刷データを収集またはキャプチャするために使用されます。
- Rocket Enterprise Server は、メインフレームアプリケーション用のアプリケーションデプロイ環境です。これは、任意のバージョンの Rocket Software Enterprise Developer を使用して移行または作成されたメインフレームアプリケーションの実行環境を提供します。

エピック

Amazon EC2 で Rocket Enterprise Server をセットアップし、メインフレームバッチアプリケーションをデプロイする

タスク	説明	必要なスキル
Rocket Enterprise Server をセットアップし、デモアプリケーションをデプロイします。	Amazon EC2 で Rocket Enterprise Server を設定し、Amazon EC2 に Rocket Software BankDemo デモアプリケーションをデプロイします。 BankDemo アプリケーションは、印刷出力を作成して開始するメインフレームのバッチアプリケーションです。	クラウドアーキテクト

Amazon EC2 で LRS プリントサーバーを設定

タスク	説明	必要なスキル
印刷用の LRS 製品ライセンスを取得します。	LRS VPSX/MFI、LRS/Queue と LRS/DIS の LRS 製品ライセンスを取得するには、「LRS Output Management チーム」にお問い合わせください。LRS 製品をインストー	ビルドリード

タスク	説明	必要なスキル
	ルする EC2 インスタンスのホスト名を指定する必要があります。	

タスク	説明	必要なスキル
Amazon EC2 Windows インスタンスを作成して LRS VPSX/MFI を作成します。	Amazon EC2 ドキュメントのAmazon EC2 インスタンスを起動する」の手順に従って、Amazon EC2 Windows インスタンスを起動します。Amazon EC2 インスタンスは、LRS VPSX/MFI に対する次のハードウェア要件とソフトウェア要件を満たしている必要があります。 • CPU – Dual Core • RAM — 16 GB • ドライブ — 500 GB • 最小限の EC2 インスタンス — m5.xlarge • OS – Windows/Linux • ソフトウェア — Internet Information Service (IIS) または Apache **Note** 上記のハードウェアとソフトウェアの要件は、小規模なプリンターフリート (約 500 ~ 1000) を対象としています。すべての要件については、LRS と AWS の担当者にお問い合わせください。	クラウドアーキテクト

タスク	説明	必要なスキル
	Windows インスタンスの作成時に、次のようにしてください。 1. EC2 ホスト名が LRS 製品ライセンスに使用されているホスト名と同じものであることを確認します。 2. 以下の操作を実行して Amazon EC2 で CGI を有効にします。 a. の手順に従って EC2 インスタンスに接続する	

タスク	説明	必要なスキル
EC2 インスタンスに LRS VPSX/MFI をインストールします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の指示に従い、 EC2 インスタンスに接続します。 2. **Note** 受信するはずの LRS 電子メールから製品のダウンロードページへのリンクを開きます。: LRS 製品は電子ファイル転送 (EFT) によって配布されます。 3. LRS VPSX/MFI をダウンロードし、ファイル (デフォルトフォルダー:c:\LRS) を解凍します。 4. 解凍したフォルダーから LRS 製品インストーラーを起動し、LRS VPSX/MFI をインストールします。 5. [機能選択] メニューで [VPSX® サーバー (V1R3.022)] を選択し、[次へ] を選択してインストールプロセスを開始します。インストールが完了すると、正常に終了したことを示すメッセージが表示されます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/Queue をインストールします。	1. Amazon EC2 ドキュメントの「ステップ 2: インスタンスに接続する」の手順に従って、Rocket Enterprise Server EC2 インスタンスに接続します。 Amazon EC2 2. 受信するはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/Queue をダウンロードし、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS 製品インストーラーを起動して LRS/Queue をインストールします。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS をインストールします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. 受信するはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/DIS をダウンロードし、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS Product Installer を起動して、このファイルをダウンロードします。 4. LRS Product Installer で [LRS Misc Tools] を展開し、[LRS DIS] を選択し、[次へ] を選択します。 5. LRS Product Installer の残りの指示に従い、インストールプロセスを完了します。	クラウドアーキテクト

タスク	説明	必要なスキル
ターゲットグループを作成し、LRS VPSX/MFI EC2 をターゲットとして登録します。	Elastic Load Balancing ドキュメントの「Network Load Balancer のターゲットグループを作成する」の次の指示に従い、ターゲットグループを作成します。 ターゲットグループを作成する際には、以下のようにしてください。 1. [グループの詳細を指定] ページの[ターゲットタイプ] を選択し、[インスタンス] を選択します。 2. プロトコルには [TCP] を選択します。 3. ポートは 5500 を選択します。 4. [ターゲットの登録] ページの [利用可能なインスタンス] セクションで、LRS VPSX/MFI EC2 インスタンスを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
Network Load Balancer を作成します。	「Elastic Load Balancing ドキュメント」の「Network Load Balancer を作成する」の次の指示に従います。Network Load Balancer は、Rocket Enterprise Server から LRS VPSX/MFI EC2 にトラフィックをルーティングします。 Network Load Balancer を作成し、[リスナーとルーティング] ページで次の操作を実行します。 1. [プロトコル] で [TCP] を選択します。 2. ポートは 5500 を選択します。 3. デフォルトアクションには、先ほど作成したターゲットグループに対して [転送先] を選択します。	クラウドアーキテクト

Rocket Enterprise Server を LRS VPSX/MFI および LRS/キューと統合する

タスク	説明	必要なスキル
LRS/キュー統合用に Rocket Enterprise Server を設定します。	1. Amazon EC2 ドキュメントの「ステップ 2: インスタンスに接続する」の手順に従って、Rocket Enterprise Server EC2 インスタンスに接続します。 Amazon EC2	クラウドアーキテクト

タスク	説明	必要なスキル
	2. Windows スタートメニューで、Rocket Enterprise Server 管理 UI を開きます。 3. メニューバーで NATIVE を選択します。 4. ナビゲーションペインで、[ディレクトリサーバー]、[BankDemo] の順に選択します。 5. 左側のナビゲーションペインの [全般] から [追加] セクションまでスクロールし、LRSQ を指すように環境変数 (LRSQ_ADDRESS、LRSQ_PORT、LRSQ_COMMAND) を設定します。 6. [LRSQ_ADDRESS] には、前に作成したNetwork Load Balancer の IP アドレスまたは DNS 名を入力します。 7. LRSQ_PORT には、「VPSX LRSQ Listener Port (5500)」を入力します。 8. [LRSQ_COMMAND] には、LRSQ 実行ファイルのパスロケーションを入力します。	

タスク	説明	必要なスキル
	Note LRS は現在、DNS 名の最大文字数 50 をサポートしていますが、これは将来変更される可能性があります。DNS 名が 50文字より大きい場合は、代わりに Network Load Balancer の IP アドレスを使用できます。	

タスク	説明	必要なスキル
LRS VPSX/MFI 統合用に Rocket Enterprise Server を設定します。	1. LRS VPSX/MFI インストーラから にある Rocket Enterprise Server の場所にVPSX_MFI_R2 フォルダをコピーしますC\BANKDEMO\print 。 2. Amazon EC2 ドキュメントの「ステップ 2: インスタンスに接続する」の手順に従って、Rocket Enterprise Server EC2 インスタンスに接続します。 Amazon EC2 3. Windows スタートメニューで、Rocket Enterprise Server 管理 UI を開きます。 4. メニューバーで NATIVE を選択します。 5. ナビゲーションペインで、[ディレクトリサーバー]、[BankDemo] の順に選択します。 6. BANKDEMO で JES を選択します。 7. [JES プログラムパス] に、その C\BANKDEMO\print 位置からの DLL(VPSX_MFI_R2) パスを追加します。	クラウドアーキテクト

Rocket Enterprise Server と LRS VPSX/MFI でプリンターと印刷ユーザーを設定する

タスク	説明	必要なスキル
Rocket Software Print Exit モジュールを Rocket Enterprise Server バッチプリンターのサーバー実行プロセスに関連付けます。	1. Amazon EC2 ドキュメントの「ステップ 2: インスタンスに接続する」の手順に従って、Rocket Enterprise Server EC2 インスタンスに接続します。 Amazon EC2 2. Windows スタートメニューで、Rocket Enterprise Server 管理 UI を開きます。 3. メニューバーで NATIVE を選択します。 4. ナビゲーションペインで、[ディレクトリサーバー]、[BankDemo] の順に選択します。 5. BANKDEMO で JES を選択し、プリンターまでスクロールします。 6. InPrinters で、Rocket Software Print Exit モジュール (LRSPRTE6 for Batch) を Rocket Enterprise Server バッチプリンターのサーバー実行プロセス (SEP) に関連付けます。これにより、LRS VPSX/MFI への印刷出力ルーティングが可能になります。	クラウドアーキテクト

タスク	説明	必要なスキル
	7. エンタープライズサーバー管理 UI にサインインします。 設定の詳細については、「Micro Focus のドキュメント」の「Exit の使用」を参照してください。	

タスク	説明	必要なスキル
LRS VPSX/MFI にプリンターを追加してください。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューから VPSX ウェブインターフェイスを開きます。 3. ナビゲーションペインで [プリンタ] を選択します。 4. [追加] を選択し、[プリンタの追加] を選択します。 5. プリンタ設定ページでプリンタ名にローカルを入力します。 6. [VPSX ID] には、VPS1 を入力します。 7. [CommType] には、[TCPIP/LRSQ] を選択します。 8. [ホスト/IP アドレス] には、追加する物理的プリンターの IP アドレスを入力します。 9. [デバイス] には、デバイスの名前を入力します。 10[Windows Driver] または [Linux/Mac Driver] のいずれかを選択します。 11[追加] を選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS VPSX/MFI でプリントユーザーを作成します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューから VPSX ウェブインターフェイスを開きます。 3. ナビゲーションペインで [セキュリティ]、[ユーザー] の順に選択します。 4. ユーザー名列で管理者を選択して、コピーを選択します。 5. ユーザープロファイル管理ウィンドウのユーザー名に、ユーザー名 (たとえば、PrintUser) を入力します。 6. 説明には、簡単な説明 (たとえば、テスト印刷の場合は User) を入力します。 7. [Update] (更新) を選択します。これにより、印刷ユーザー (PrintUser など) を作成します。 8. ナビゲーションペインの User で、 作成した新しいユーザーを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
	9. コマンドメニューから [セキュリティ] を選択します。 10.セキュリティルールページで、該当するプリンタセキュリティとジョブセキュリティオプションをすべて選択して、保存を選択します。 11新しいプリントユーザーを管理者グループに追加するには、ナビゲーションペインに移動してセキュリティと設定を順に選択します。 12.セキュリティ設定ウィンドウで、新しいプリントユーザーを管理者列に追加します。	

印刷認証と認可の設定

タスク	説明	必要なスキル
ユーザーとグループを持つ AWS Managed Microsoft AD ドメインを作成します。	1. AWS Directory Service ドキュメントの「AWS Managed Microsoft AD ディレクトリの作成」の手順に従い、AWS Managed Microsoft AD にアクティブディレクトリを作成します。 2. AWS Directory Service ドキュメントの「ステッ	クラウドアーキテクト

タスク	説明	必要なスキル
	プ 3:EC2 インスタンスをデプロイして Managed Microsoft AD を管理する」の手順に従い、EC2 インスタンス (Active Directory マネージャー) をデプロイし、Active Directory ツールをインストールして AWS マネージド Microsoft AD を管理します。 3. **Note** 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の指示に従い、 EC2 インスタンスに接続します。: EC2 インスタンスに接続するときは、Windows セキュリティウィンドウに管理者認証情報 (ステップ 1 で作成したディレクトリ用) を入力します。 4. Windows スタートメニューから [Windows 管理ツール] を選択し、[Active Directory ユーザーとコンピュータ] を選択します。 5. 「AWS Directory Service ドキュメント」の「ユーザー	

タスク	説明	必要なスキル
	の作成」の手順に従い、Active Directory ドメインにプリントユーザーを作成します。	
LRS VPSX/MFI EC2 を AWS Managed Microsoft AD ドメインに接続します。	LRS VPSX/MFI EC2 を AWS Managed Microsoft AD ドメイン に「自動」(AWS ナレッジセンターのドキュメント) または 「手動」(AWS Directory Service ドキュメント) で参加させます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS を AWS Managed Microsoft AD と設定し、統合します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows スタートメニューからVPSX Web Interface を開きます。 3. ナビゲーションペインで [セキュリティ]、[設定] の順に選択します。 4. Security Configuration ページの Security Parameters セクションで Security Type に Internal を選択します。 5. Security Parameters セクションに、残りのオプションの設定を入力します。 6. Microsoft Windows のスタートメニューから LRS Output Management フォルダを開き、Server Start と Server Stop を順に選択します。 7. Active Directory のユーザー名とパスワードを使用して LRS VPSX/MFI にログインします。	クラウドアーキテクト

印刷ワークフローのテスト

タスク	説明	必要なスキル
Rocket Software BankDemo アプリからバッチ印刷リクエストを開始します。	1. Rocket Enterprise Server EC2 インスタンスで 3270 ターミナルエミュレータを開きます。 2. 次のコマンド：`connect 127.0.0.1:9278` を実行してBankDemoアプリに接続します。 3. BankDemo のコマンドラインインターフェースで、ユーザー ID に B0001 を入力します。パスワードには、空白以外のキーを入力します。 4. 印刷済みステートメントをリクエストオプションでは、空白行に [X] を入力します。 5. ステートメントの送信元セクションのメールに Y を入力し、F10 を押します。	テストエンジニア
LRS VPSX/MFI のプリント出力を確認します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows スタートメニューからVPSX Web Interface を開きます。	テストエンジニア

タスク	説明	必要なスキル
	3. ナビゲーションペインで [プリンター] を選択し、[出力キュー] を選択します。 4. [Spool ID] 列で、プリンターキュー内のリクエストのスプール ID を選択します。 5. [アクション] タブの [コマンド] 列で、[ブラウザ] を選択します。 これで、勘定科目、説明、日付、金額と残高の列を含む勘定取引明細書が印刷されます。例として、このパターンの [batch_print_output] 添付ファイルを参照してください。	

関連リソース

- 「LRS 出力のモダナイゼーション」(LRS ドキュメント)
- 「ANSI とマシンキャリッジ制御」(IBM ドキュメント)
- 「チャネルコマンドワード」(IBM ドキュメント)
- 「Micro Focus による AWS 上のエンタープライズメインフレームワークロードの強化」(AWS パートナー ネットワークブログ)
- 「Amazon EC2 Auto Scaling とSystems Manager による Micro Focus エンタープライズサーバー PAC の構築」(AWS 規範ガイダンスのドキュメント)
- 「Advanced Function Presentation (AFP) データストリーム」(IBM ドキュメント)
- 「ラインコンディショニングデータストリーム (LCDS)」(Compart ドキュメント)

追加情報

考慮事項

モダナイゼーションを進める中で、メインフレームのバッチプロセスとそれらが生成する出力について、さまざまな構成を検討するかもしれません。メインフレームプラットフォームは、印刷に直接影響する特定の要件に合わせて、使用するすべての顧客とベンダーによってカスタマイズされています。たとえば、現在のプラットフォームでは、IBM Advanced Function Presentation (AFP) や Xerox Line Condition Data Stream (LCDS) が現在のワークフローに組み込まれている場合があります。さらに、「メインフレームのキャリッジコントロール文字」や「チャネルコマンドワード」は、印刷ページの外観に影響を与える可能性があり、特別な処理が必要な場合もあります。モダナイゼーション計画プロセスの一環として、特定の印刷環境における構成を評価し、理解しておくことをお勧めします。

印刷データキャプチャ

Rocket Software Print Exit は、LRS VPSX/MFI がスプールファイルを効果的に処理するために必要な情報を渡します。この情報は、関連する制御ブロックに渡される次のようなフィールドで構成されます。

- JobName
- 所有者 (ユーザー ID)
- 送信先
- フォーム
- ファイル名
- 書き込み

LRS VPSX/MFI は、Rocket Enterprise Server からデータをキャプチャするための以下のメインフレームバッチメカニズムをサポートしています。

- 標準の z/OS JCL SYSOUT DD/OUTPUT ステートメントにより、COBOL 印刷/スプールをバッチ処理します。
- 標準の z/OS JCL CA-SPOOL SUBSYS DD ステートメントにより、BATCH COBOL 印刷/スプールをバッチ処理します。
- CBLTDLI インターフェースを使用した IMS/COBOL 印刷/スプール処理 (サポートされているメソッドとプログラミング例の全リストについては、製品ライセンスに含まれている LRS ドキュメントを参照してください)。

プリンターフリートのヘルスチェック

LRS VPSX/MFI (LRS LoadX) は、デバイス管理や運用の最適化など、詳細なヘルスチェックを実行できます。デバイス管理では、プリンターデバイスの障害を検出し、印刷要求を正常なプリンターに転送できます。プリンターフリートの詳細なヘルスチェックについて、詳細は、製品ライセンスに含まれている LRS のドキュメントを参照してください。

印刷認証と認可

LRS/DIS を使用すると、LRS アプリケーションは Microsoft Active Directory または LDAP サーバーでユーザー ID とパスワードを認証できます。LRS/DIS は、基本的な印刷認可に加えて、次のようなユースケースでも粒度レベルの印刷セキュリティ制御を適用できます。

- プリンタジョブを参照できるユーザーを管理します。
- 他のユーザーのジョブの参照レベルを管理します。
- 運用タスクを管理します。たとえば、保留/リリース、削除、変更、コピー、ルート変更などのコマンドレベルのセキュリティなど。セキュリティは、ユーザー ID またはグループ (AD グループや LDAP グループと同様) のいずれかで設定できます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

メインフレームのモダナイゼーション: Rocket Software Enterprise Suite AWS を使用した での DevOps

作成者：Kevin Yung (AWS)

概要

注意： AWS CodeCommit は、新規のお客様にはご利用いただけません。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

お客様の課題

メインフレームハードウェアでコアアプリケーションを実行する組織では、デジタルイノベーションの要求を満たすためにハードウェアをスケールアップする必要がある場合、通常、いくつかの課題に直面します。これらの課題には、以下の制約があります。

- メインフレームの開発環境とテスト環境は、メインフレームのハードウェアコンポーネントには柔軟性がなく、変更には高いコストがかかるため、拡張することができません。
- 新規開発者は従来のメインフレーム開発ツールに慣れておらず、興味もないため、メインフレーム開発はスキル不足に直面しています。コンテナ、継続的統合/継続的デリバリー (CI/CD) パイプライン、最新のテストフレームワークなどの最新テクノロジーは、メインフレーム開発では利用できません。

パターンアウトカム

これらの課題に対処するために、Amazon Web Services (AWS) と AWS Partner Network (APN) パートナーである Rocket Software Micro Focus が協力してこのパターンを作成しました。このソリューションが以下の成果達成に役立つように設計されています。

- 開発者の生産性が向上されました。開発者は、新しいメインフレーム開発インスタンスを数分で取得できます。
- を使用して AWS クラウド 、実質的に無制限の容量で新しいメインフレームテスト環境を作成します。
- 新しいメインフレーム CI/CD インフラストラクチャの迅速なプロビジョニング。でのプロビジョニングは、 AWS CloudFormation と を使用して 1 時間以内に完了 AWS できます AWS Systems Manager。

- 、、Amazon Elastic Container Registry (Amazon ECR) など AWS CodeBuild AWS CodeCommit AWS CodePipeline AWS CodeDeploy、メインフレーム開発用の AWS DevOps ツールのネイティブ使用。
- 従来のウォーターフォール開発をメインフレームプロジェクトのアジャイル開発に変換します。

テクノロジーの概要

このパターンでは、ターゲットスタックには以下のコンポーネントが含まれます。

論理コンポーネント	実装ソリューション	説明
ソースコードリポジトリ	Rocket Software AccuRev Server、CodeCommit、Amazon ECR	ソースコード管理 – このソリューションでは、次の 2 種類のソースコードを使用します。 • COBOL や JCL などのメインフレームソースコード。 • AWS インフラストラクチャテンプレートと自動化スクリプト どちらのタイプのソースコードもバージョン管理が必要ですが、異なる SCM で管理されます。メインフレームまたは Rocket Software Enterprise Server にデプロイされたソースコードは、Rocket Software Micro Focus AccuRev Server で管理されます。 AWS テンプレートと自動化スクリプトは CodeCommit で管理されます。Amazon ECR は Docker イメージリポジトリに使用されます。

エンタープライズ開発者インスタンス	Amazon Elastic Compute Cloud (Amazon EC2)、Rocket Software Enterprise Developer for Eclipse	メインフレーム開発者は、Rocket Software Enterprise Developer for Eclipse を使用して Amazon EC2 でコードを開発できます。これにより、コードの作成やテストをメインフレームのハードウェアに依存しなくても構わないです。
Rocket Software Enterprise Suite ライセンス管理	Rocket Software Enterprise Suite License Manager	一元化された Rocket Software Enterprise Suite ライセンスの管理とガバナンスのために、このソリューションは Rocket Software Enterprise Suite License Manager を使用して必要なライセンスをホストします。
CI/CD パイプライン	CodePipeline、CodeBuild、CodeDeploy、コンテナ内の Rocket Software Enterprise Developer、コンテナ内の Rocket Software Enterprise Test Server、Rocket Software Micro Focus Enterprise Server	メインフレーム開発チームには、コードのコンパイル、統合テスト、および回帰テストを実行するための CI/CD パイプラインが必要です。では AWS、CodePipeline と CodeBuild は、コンテナ内で Rocket Software Enterprise Developer と Enterprise Test Server をネイティブに操作できます。

前提条件と制限

前提条件

名前	説明

py3270	py3270 は IBM 3270 ターミナルエミュレーター x3270 への Python インターフェースです。x3270 または s3270 サブプロセスに API を提供します。
3270	x3270 は X ウィンドウシステムと Windows の IBM 3270 ターミナルエミュレーターです。 開発者はこれをローカルでのユニットテストに使用できます。
ロボット-フレームワーク-メインフレーム-3270-ライブラリ	メインフレーム3270 は py3270 プロジェクトベースのロボットフレームワークのライブラリです。
Rocket ソフトウェア Verastream	Rocket Software Verastream は、モバイルアプリ、ウェブアプリケーション、SOA ウェブサービスのテスト方法に従ってメインフレームアセットをテストできるようにする統合プラットフォームです。
Rocket Software Unified Functional Testing (UFT) インストーラとライセンス	Rocket Software Unified Functional Testing は、ソフトウェアアプリケーションと環境に機能テストと回帰テストの自動化を提供するソフトウェアです。
Rocket Software Enterprise Server インストーラとライセンス	エンタープライズサーバーはメインフレームアプリケーション用のランタイム環境を提供します。
Rocket Software Enterprise Test Server インストーラとライセンス	Rocket Software Enterprise Test Server は、IBM メインフレームアプリケーションのテスト環境です。
サーバー用の Rocket Software AccuRev インストーラとライセンス、Windows および Linux オペレーティングシステム用の Rocket Software Micro Focus AccuRev インストーラとライセンス	AccuRev はソースコード管理 (SCM) を提供します。AccuRev システムは、複数のファイルを開発するチームによる使用を想定して設計されています。

Rocket Software Enterprise Developer for Eclipse インストーラ、パッチ、ライセンス	エンタプライズ開発者に、メインフレームの中核となるオンラインおよびバッチアプリケーションの開発と保守を行うためのプラットフォームを提供します。

機能制限

- Windows Docker イメージのビルドはCodeBuild に適用されません。この報告された問題には、Windows カーネル/HCS チームと Docker チームからのサポートが必要です。回避策は、システムマネージャーを使用して Docker イメージビルドランブックを作成することです。このパターンでは、回避策を使用して、Rocket Software Enterpise Developer for Eclipse および Rocket Software Micro Focus Enterprise Test Server Container イメージを構築します。
- CodeBuild からの仮想プライベートクラウド (VPC) 接続は Windows ではまだサポートされていないため、このパターンでは、Rocket Software License Manager を使用して OpenText Rocket Software Enterprise Developer および Rocket Software Enterprise Test Server コンテナのライセンスを管理しません。

製品バージョン

- Rocket Software Enterprise Developer 5.5 以降
- Rocket Software Enterprise Test Server 5.5 以降
- Rocket Software Enterprise Server 5.5 以降
- Rocket Software AccuRev 7.x 以降
- Rocket Software Enterprise Developer および Enterprise Test Server 用の Windows Docker ベースイメージ: microsoft/dotnet-framework-4.7.2-runtime
- AccuRev クライアントの Linux Docker ベースイメージ：amazonlinux:2

アーキテクチャ

メインフレーム環境

従来のメインフレーム開発では、開発者はメインフレームのハードウェアを使用してプログラムの開発とテストを行う必要がありました。開発/テスト環境では毎秒100万命令 (MIPS) という制限があるなど、容量の制限に直面して、メインフレーム・コンピューターで利用できるツールに依存する必要があります。

多くの組織では、メインフレーム開発はウォーターフォール型の開発手法に従い、チームは変更をリリースするために長いサイクルに頼っています。通常、これらのリリースサイクルはデジタル製品開発よりも長くなります。

次の図表では、複数のメインフレームプロジェクトがメインフレームハードウェアを共有して開発している様子を示しています。メインフレームハードウェアでは、開発環境とテスト環境をスケールアウトしてより多くのプロジェクトに対応させるにはコストがかかります。

AWS アーキテクチャー

このパターンは、メインフレーム開発を に拡張します AWS クラウド。まず、AccuRev SCM を使用してメインフレームソースコードをホストします AWS。次に、Enterprise Developer と Enterprise Test Server がメインフレームコードの構築とテストに使用できます AWS。

次のセクションでは、パターンの 3 つの主要コンポーネントについて説明します。

1. SCM

このパターンでは AWS、AccuRev を使用して、メインフレームソースコードの SCM ワークスペースとバージョン管理のセットを作成します。ストリームベースのアーキテクチャにより、複数のチームでメインフレームを並行開発できます。AccuRev では、変更をマージするためにプロモートというコンセプトを使用します。AccuRev では、変更を他のワークスペースに追加するには、更新コンセプトを使用します。

プロジェクトレベルでは、各チームが AccuRev に 1 つ以上のストリームを作成して、プロジェクトレベルの変更を追跡できます。これらはプロジェクトストリームと呼ばれます。これらのプロジェクトストリームは同じ親ストリームから継承されます。親ストリームでは、異なるプロジェクトストリームからの変更をマージするために使用されます。

各プロジェクトストリームはコードを AccuRev に昇格させることができ、昇格後トリガーが AWS CI/CD パイプラインを開始するように設定されます。プロジェクトストリーム変更のビルドが成功した場合、親ストリームにプロモートして、さらに回帰テストを行うことができます。

通常、親ストリームはシステム統合トリームと呼ばれます。プロジェクトストリームからシステム統合ストリームへのプロモーションが行われると、プロモーション後のトリガーが別の CI/CD パイプラインを起動して回帰テストを実行します。

メインフレームコードに加えて、このパターンには AWS CloudFormation テンプレート、Systems Manager Automation ドキュメント、スクリプトが含まれます。infrastructure-as-code のベストプラクティスに従って、CodeCommit でバージョン管理されます。

デプロイのためにメインフレームコードをメインフレーム環境に同期する必要がある場合、Rocket Software は AccuRev SCM からメインフレーム SCM にコードを同期する Enterprise Sync ソリューションを提供します。

2. 開発者とテスト環境

大規模な組織では、メインフレーム開発者を100人以上、あるいは1000人以上規模に拡大することは困難です。この制約に対処するため、このパターンでは、開発に Amazon EC2 Windows インスタンスを使用します。インスタンスには、Enterprise Developer for Eclipse ツールがインストールされます。開発者はすべてのメインフレームコードのテストとデバッグをインスタンス上でローカルに実行できます。

AWS Systems Manager ステートマネージャーとオートメーションのドキュメントは、開発者インスタンスのプロビジョニングを自動化するために使用されます。開発者インスタンスの作成にかかる平均時間は 15 分以内です。以下のソフトウェアと設定が準備されています。

- AccuRev にソースコードをチェックアウトして AccuRev にコミットするための Windows クライアント
- メインフレームコードをローカルで記述、テスト、デバッグするための Enterprise Developers for Eclipse ツール
- オープンソースのテストフレームワーク Python 動作駆動開発 (BDD) テストフレームワーク Behave、py3270、およびアプリケーションをテストするためのスクリプト作成のx3270 エミュレーター
- エンタープライズテストサーバー Docker コンテナーでエンタプライズテストサーバーイメージを構築し、アプリケーションをテストするための Docker 開発者ツール。

開発サイクルでは、開発者は EC2 インスタンスを使用して、メインフレームコードをローカルで開発およびテストします。ローカルの変更が正常にテストされる場合、開発者は変更を AccuRev サーバーにプロモートします。

3. CI/CD パイプライン

このパターンでは、CI/CD パイプラインは本番環境にデプロイする前の統合テストや回帰テストに使用されます。

SCM セクションで説明したように、AccuRev はプロジェクトストリームと統合ストリームの 2 種類のストリームを使用します。各ストリームが CI/CD パイプラインに接続されます。AccuRev サーバーと の統合を実行するために AWS CodePipeline、このパターンは AccuRev 昇格後スクリプトを使用して CI/CD を開始するイベントを作成します。

たとえば、開発者が AccuRev のプロジェクトストリームへの変更を推進する場合、AccuRev サーバーで実行される推進後のスクリプトが開始されます。次に、スクリプトは変更のメタデータを Amazon Simple Storage Service (Amazon S3) バケットにアップロードし、Amazon S3 イベントを作成します。このイベントでは、CodePipeline で設定されたパイプラインの実行を開始します。

同じイベント開始メカニズムが統合ストリームとそれに関連するパイプラインに使用されます。

CI/CD パイプラインでは、CodePipeline は CodeBuild を AccuRev Linux クライアントコンテナとともに使用して、AccuRev ストリームから最新のコードをチェックアウトします。次に、パイプラインは CodeBuild を起動して Enterprise Developer Windows コンテナを使用してソースコードをコンパイルし、CodeBuild の Enterprise Test Server Windows コンテナを使用してメインフレームアプリケーションをテストします。

CI/CD パイプラインは CloudFormation テンプレートを使用して構築され、ブループリントは新しいプロジェクトに使用されます。テンプレートを使用すると、プロジェクトが新しい CI/CD パイプラインを作成するのに 1 時間もかかりません AWS。

メインフレームのテスト機能をスケールするために AWS、このパターンは Rocket Software DevOps テストスイート、Verastream、UFT サーバーを構築します。最新の DevOps ツールを使用すると、 AWS 必要な数のテストを で実行できます。

Rocket Software を使用したメインフレーム開発環境の例を次の図 AWS に示します。

ターゲットテクノロジースタック

このセクションでは、パターンの各コンポーネントのアーキテクチャを詳しく見ていきます。

1. ソースコードリポジトリ – AccuRev SCM

AccuRev SCM は、メインフレームソースコードのバージョンを管理するように設定されています。高可用性を実現するため、AccuRev にプライマリモードとレプリカモードが適用されます。オペレータは、プライマリノードでメンテナンスを行う際に、レプリカにフェイルオーバーできます。

CI/CD パイプラインの応答速度を上げるために、このパターンでは Amazon CloudWatch Eventsを使用してソースコードの変更を検出し、パイプラインの開始を開始します。

1. パイプラインは Amazon S3 ソースを使用するように設定されています。
2. CloudWatch イベントルールは、ソース S3 バケットから S3 イベントをキャプチャするように設定されています。
3. CloudWatch イベントルールは、パイプラインにターゲットを設定します。
4. AccuRev SCM は、プロモーションの完了後にプロモーション後のスクリプトをローカルで実行するように設定されています。
5. AccuRev SCM はプロモーションのメタデータを含む XML ファイルを生成し、スクリプトはその XML ファイルをソース S3 バケットにアップロードします。
6. アップロード後、ソース S3 バケットは CloudWatch Events ルールに一致するイベントを送信し、CloudWatch Events ルールはパイプラインの実行を開始します。

パイプラインが実行されると、AccuRev Linux クライアントコンテナを使用して、関連する AccuRev ストリームから最新のメインフレームコードをチェックアウトする CodeBuild プロジェクトが開始されます。

次の図表では、AccuRev サーバーのセットアップを示しています。

2. エンタープライズデベロッパーテンプレート

このパターンでは、Amazon EC2 テンプレートを使用して開発者インスタンスを簡単に作成できます。ステータスマネージャーを使用して、ソフトウェアとライセンスの設定を EC2 インスタンスに一貫して適用できます。

Amazon EC2 テンプレートには VPC コンテキスト設定とデフォルトインスタンス設定がビルドインされ、企業のタグ付け要件に準拠しています。テンプレートを使用して、チームは独自の新しい開発インスタンスを作成できます。

開発者インスタンスが起動する時、タグと関連付けることで、システムマネージャーはステータスマネージャーを使用して自動化を適用します。自動化には以下の一般的なステップが含まれます。

1. Enterprise Developer ソフトウェアをインストールし、パッチをインストールします。
2. Windows 用の AccuRev クライアントをインストールします。

3. 開発者が AccuRev ストリームに参加できるように、事前設定済みのスクリプトをインストールします。Eclipse ワークスペースを初期化します。
4. x3270、py3270、Docker などの開発ツールをインストールします。
5. License Manager ロードバランサーを指すようにライセンス設定を構成します。

次の図は、Amazon EC2 テンプレートによって作成されたエンタープライズ開発者インスタンスと、State Manager によってインスタンスに適用されたソフトウェアと設定を示しています。エンタープライズ開発者インスタンスは に接続 AWS License Manager してライセンスをアクティブ化します。

3. CI/CD パイプライン

AWS アーキテクチャセクションで説明したように、パターンにはプロジェクトレベルの CI/CD パイプラインとシステム統合パイプラインがあります。メインフレームの各プロジェクトチームは、プロジェクトで開発しているプログラムを構築するためのパイプラインを 1 つまたは複数の CI/CD パイプラインを作成します。これらのプロジェクトの CI/CD パイプラインは、関連する AccuRev ストリームからソースコードをチェックアウトします。

プロジェクトチームでは、開発者は関連の AccuRev ストリームでコードを宣伝します。次に、プロモーションによってプロジェクトパイプラインが開始され、コードが構築され、統合テストが実行されます。

各プロジェクト CI/CD パイプラインは、Enterprise Developer ツールの Amazon ECR イメージと Enterprise Test Server ツールの Amazon ECR イメージで CodeBuild プロジェクトを使用します。

CodePipeline と CodeBuild は、CI/CD パイプラインの作成に使用されます。CodeBuild と CodePipeline には前払い料金や契約がないため、お支払いは実際に使用した分のみです。メインフレームハードウェアと比較して、この AWS ソリューションはハードウェアプロビジョニングのリードタイムを大幅に短縮し、テスト環境のコストを削減します。

現代の開発では、複数のテスト方法論が使用されています。たとえば、テスト駆動開発 (TDD)、BDD、ロボットフレームワークなどです。このパターンでは、開発者はこれらの最新ツールをメインフレームテストに使用できます。たとえば、x3270、py3270、および Behas python テストツールを使用することで、オンラインアプリケーションの動作を定義できます。これらの CI/CD パイプラインでは、ビルドメインフレーム 3270 ロボットフレームワークを使用することもできます。

次の図表は、チームストリーム CI/CD パイプラインを示しています。

次の図は、CodePipeline がメインフレーム3270ロボットフレームワークで作成したプロジェクトCI/CDテストレポートを示します。

次の図は、CodePipeline によって Py3270 と Behave BDD で作成されたプロジェクト CI/CD テストレポートを示します。

プロジェクトレベルのテストに合格した後、テストされたコードは手動で AccuRev SCM の統合ストリームにプロモートされます。チームがプロジェクトパイプラインのテスト対象範囲に確信が持てたら、このステップを自動化できます。

コードがプロモートされる場合、システム統合 CI/CD パイプラインはマージされたコードをチェックアウトし、回帰テストを実行します。マージされたコードは、すべての並行プロジェクトストリームからプロモートされます。

テスト環境の粒度に応じて、お客様は UAT、本番稼働前など、別の環境により多くのシステム統合 CI/CD パイプラインを持つことができます。

このパターンでは、システム統合パイプラインで使用されるツールは、エンタープライズテストサーバー、UFT サーバー、Verastream です。これらのツールはすべて Docker コンテナにデプロイして CodeBuild で使用できます。

メインフレームプログラムのテストに成功する場合、アーティファクトはバージョン管理を行って S3 バケットに保存されます。

次の図は、システム統合 CI/CD パイプラインを示しています。

アーティファクトがシステム統合 CI/CD パイプラインで正常にテストされた後に、本番環境へのデプロイに移行できます。

ソースコードをメインフレームにデプロイする必要がある場合、Rocket Software は AccuRev から Mainframe Endeavour にソースコードを同期するための Enterprise Sync ソリューションを提供します。

次の図は、アーティファクトを Enterprise Server にデプロイする本稼働 CI/CD パイプラインを示しています。この例では、CodeDeploy はテスト済みのメインフレームアーティファクトを Enterprise Server にデプロイします。

CI/CD パイプラインのアーキテクチャチュートリアルに加えて、CodeBuild および CodePipeline でのメインフレームアプリケーションのテストの詳細については、 AWS DevOps ブログ記事「Automate thousands of mainframe tests on with the Micro Focus Enterprise Suite」を参照してください。 AWS(Micro Focus は Rocket Software になりました) 。メインフレームテストのベストプラクティスと詳細については、ブログ記事を参照してください AWS。

ツール

AWS オートメーションツール

- AWS CloudFormation
- Amazon CloudWatch Events
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline
- Amazon ECR
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager

Rocket ソフトウェアツール

- Eclipse 用 Rocket Enterprise Developer
- Rocket Enterprise テストサーバー
- Rocket Enterprise Server (本番デプロイ)
- Rocket ソフトウェア AccuRev

- Rocket Software Enterprise Suite License Manager
- Rocket Software Verastream ホストインテグレーター
- ロケットソフトウェア UFT One

その他のツール

- x3270
- py3270
- Robot-Framework-Mainframe-3270-Library

エピック

AccuRev SCM インフラストラクチャの作成

タスク	説明	必要なスキル
CloudFormation を使用してプライマリ AccuRev SCM サーバーをデプロイします。		AWS CloudFormation
AccuRev 管理者ユーザーを作成します。	AccuRev SCM サーバにログインし、CLI コマンドを実行して管理者ユーザを作成します。	AccuRev SCM サーバ管理者
AccuRev ストリームを作成します。	プロダクション、システム統合、チームストリームの順に、上位ストリームから継承する AccuRev ストリームを作成します。	AccuRev SCM 管理者
開発者の AccuRev ログインアカウントを作成します。	AccuRev SCM CLI コマンドを使用して、メインフレーム開発者用の AccuRev ユーザーログインアカウントを作成します。	AccuRev SCM 管理者

エンタープライズデベロッパー Amazon EC2 起動テンプレートを作成

タスク	説明	必要なスキル
CloudFormation を使用して Amazon EC2 起動テンプレートをデプロイします。	CloudFormation を使用して、Enterprise Developer インスタンス用の Amazon EC2 起動テンプレートをデプロイします。テンプレートには、Rocket Enterprise Developer インスタンスの Systems Manager Automation ドキュメントが含まれています。	AWS CloudFormation
Amazon EC2 テンプレートからエンタープライズ開発者インスタンスを作成します。		AWS コンソールログインとメインフレーム開発者スキル

Enterprise Developer ツールの Docker イメージを作成する

タスク	説明	必要なスキル
Enterprise Developer ツールの Docker イメージを作成します。	Docker コマンドと Enterprise Developer ツールの Dockerfile を使用して Docker イメージを作成します。	Docker
Amazon ECR で、Docker リポジトリを作成します。	Amazon ECR コンソールで、Enterprise Developer Docker イメージのリポジトリを作成します。	Amazon ECR
Enterprise Developer ツールの Docker イメージを Amazon ECR にプッシュします。	Docker プッシュコマンドを実行して、エンタープライズデベロッパーツールの Docker イメージをプッシュ	Docker

タスク	説明	必要なスキル
	し、Amazon ECR の Docker リポジトリに保存します。	

Enterprise Test Server Docker イメージを作成する

タスク	説明	必要なスキル
Enterprise Test Server Docker イメージを作成します。	Docker コマンドと Enterprise Test Server Dockerfile を使用して Docker イメージを作成します。	Docker
Amazon ECR で、Docker リポジトリを作成します。	Amazon ECR コンソールで、Enterprise Test Server Docker イメージの Amazon ECR リポジトリを作成します。	Amazon ECR
Enterprise Test Server Docker イメージを Amazon ECR にプッシュします。	Docker プッシュコマンドを実行して、エンタープライズテストサーバーの Docker イメージを Amazon ECR にプッシュして保存します。	Docker

チームストリーム CI/CD パイプラインの作成

タスク	説明	必要なスキル
CodeCommit リポジトリを作成します。	CodeCommit コンソールで、インフラストラクチャと CloudFormation コード用の Git ベースのリポジトリを作成します。	AWS CodeCommit

タスク	説明	必要なスキル
CloudFormation テンプレートとオートメーションコードを CodeCommit リポジトリにアップロードします。	Git push コマンドを実行して、CloudFormation テンプレートとオートメーションコードをリポジトリにアップロードします。	Git
CloudFormation を使用してチームストリーム CI/CD パイプラインをデプロイします。	準備済みの CloudFormation テンプレートを使用して、チームストリーム CI/CD パイプラインをデプロイします。	AWS CloudFormation

システム統合 CI/CD パイプラインの作成

タスク	説明	必要なスキル
UFT Docker イメージを作成します。	Docker コマンドと UFT Dockerfile を使用して Docker イメージを作成します。	Docker
UFT イメージの Amazon ECR に Docker リポジトリを作成します。	Amazon ECR コンソールで、UFT イメージの Docker リポジトリを作成します。	Amazon ECR
UFT Docker イメージを Amazon ECR にプッシュします。	Docker プッシュコマンドを実行して、エンタープライズテストサーバーの Docker イメージを Amazon ECR にプッシュして保存します。	Docker
Verastream Docker イメージを作成します。	Docker コマンドと Verastream Dockerfile を使用して Docker イメージを作成します。	Docker

タスク	説明	必要なスキル
Verastream イメージの Amazon ECR に Docker リポジトリを作成します。	Amazon ECR コンソールで、Verastream イメージの Docker リポジトリを作成します。	Amazon ECR
CloudFormation を使用してシステム統合 CI/CD パイプラインをデプロイします。	準備済みの CloudFormation テンプレートを使用して、システム統合 CI/CD パイプラインをデプロイします。	AWS CloudFormation

本番デプロイCI/CD パイプラインの作成

タスク	説明	必要なスキル
AWS クイックスタートを使用して Enterprise Server をデプロイします。	CloudFormation を使用して Enterprise Server をデプロイするには、 AWS クイックスタートで Enterprise Server を起動します。	AWS CloudFormation
本番デプロイCI/CD パイプラインをデプロイします。	CloudFormation コンソールで、CloudFormation テンプレートを使用して本番デプロイ CI/CD パイプラインをデプロイします。	AWS CloudFormation

関連リソース

リファレンス

- AWS DevOps ブログ - Micro Focus Enterprise Suite AWS を使用して、 で何千ものメインフレームテストを自動化します (Micro Focus は Rocket Software になりました)。
- py3270/py3270 GitHub リポジトリ
- Altran-PT-GDC/Robot-Framework-Mainframe-3270-ライブラリ GitHub リポジトリ

- ビヘイブようこそ！
- APN パートナーブログ - タグ: Micro Focus (Micro Focus が Rocket Software になりました)
- 起動テンプレートからのインスタンスの起動

AWS Marketplace

- ロケットソフトウェア UFT One

AWS クイックスタート

- の Rocket Enterprise Server AWS

Micro Focus Enterprise ServerとLRS VPSX/MFIを使用して、AWS 上のメインフレームのオンライン印刷ワークロードを最新化

作成者：Shubham Roy (AWS), Abraham Rondon (Micro Focus), Guy Tucker (Levi, Ray and Shoup Inc), and Kevin Yung (AWS)

概要

このパターンは、最新のメインフレームアプリケーションのランタイムとしてMicro Focus Enterprise Serverを使用し、プリントサーバーとしてLRS VPSX/MFI（Micro Focus Interface）を使用することで、Amazon Web Services（AWS）クラウド上のビジネスクリティカルなメインフレームのオンライン印刷ワークロードを最新化する方法を示しています。このパターンは、「リプラットフォーム」のメインフレーム近代化アプローチに基づいています。このアプローチでは、メインフレームのオンラインアプリケーションを Amazon Elastic Compute Cloud (Amazon EC2) に、メインフレームデータベース (IBM DB2 for z/OS など) を Amazon Relational Database Service (Amazon RDS) に移行します。最新の印刷ワークフローの認証と認可は、AWS Managed Microsoft AD としても知られる Microsoft Active Directory のための AWS Directory Service によって実行されます。LRS Directory Information Server (LRS/DIS) は AWS Managed Microsoft AD と統合されており、印刷ワークフローの認証と認可を行います。オンライン印刷ワークロードを最新化することで、IT インフラストラクチャコストの削減、レガシーシステムの維持に伴う技術的負担の軽減、データサイロの排除、DevOps モデルによる敏捷性と効率性の向上、AWS クラウドのオンデマンドリソースと自動化の活用が可能になります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- メインフレームのオンライン印刷または出力管理のワークロード
- Micro Focus Enterprise Server上で動作するメインフレームアプリケーションを再構築して配信する方法に関する基本知識 (詳細については、Micro Focus ドキュメントの「Enterprise Server」を参照してください)。
- LRS クラウドプリンティングソリューションとコンセプトに関する基本知識 (詳細については、LRS ドキュメントの「Output Modernization」を参照してください)。
- Micro Focus Enterprise Serverのソフトウェアとライセンス (詳細については、「Micro Focusの営業担当」にお問い合わせください)。

- LRS VPSX/MFI、LRS/Queue、LRS/DIS のソフトウェアとライセンス (詳細については、「LRS の営業担当」にお問い合わせください)。

> **Note**
>
> メインフレームのオンライン印刷ワークロードの設定に関する考慮事項の詳細については、このパターンの「追加情報」セクションの「考慮事項」を参照してください。

製品バージョン

- Micro Focus Enterprise Server 8.0 以降
- 「LRS VPSX/MFI V1R3」以降

アーキテクチャ

ソーステクノロジースタック

- オペレーティングシステム — IBM z/OS
- プログラミング言語 — 共通ビジネス指向言語 (COBOL) と顧客情報管理システム (CICS)
- データベース — IBM DB2 for z/OS IBM 情報管理システム (IMS) と仮想ストレージアクセス方法 (VSAM)
- セキュリティ — Resource Access Control Facility (RACF)、CA Top Secret for z/OS、Access Control Facility 2 (ACF2)
- 印刷と出力管理 — IBM メインフレーム z/OS 印刷製品 (IBM Infoprint Server for z/OS、LRS と CA View)

ターゲットテクノロジースタック

- オペレーティングシステム — Amazon EC2 上で実行する Microsoft Windows Server
- コンピューティング — Amazon EC2
- プログラミング言語 — COBOL と CICS
- データベース — Amazon RDS
- セキュリティ — AWS Managed Microsoft AD

- 印刷と出力管理 — AWS での LRS 印刷ソリューション
- メインフレームランタイム環境 — Micro Focus Enterprise Server

ソースアーキテクチャ

次の図は、メインフレームのオンライン印刷ワークロードの一般的な現在の状態アーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーは、COBOL で書かれた IBM CICS アプリケーション上に構築されたエンゲージメントシステム (SoE) 上でビジネストランザクションを実行します。
2. SoE はメインフレームサービスを呼び出し、IBM DB2 for z/OS などの記録システム (SoR) データベースにビジネストランザクションデータを記録します。
3. SoR は SoE からのビジネスデータを永続化します。
4. CICS SoE は印刷要求を処理する印刷トランザクションアプリケーションを起動しますが、ユーザーは CICS SoE から印刷出力を生成する要求を開始します。
5. 印刷トランザクションアプリケーション (CICS や COBOL プログラムなど) は、データベースからデータを抽出し、ビジネス要件に従い、データをフォーマットし、請求明細書、ID カード、ローン明細書などのビジネス出力 (印刷データ) を生成します。次に、アプリケーションは Virtual Telecommunications Access Method (VTAM) を使用して印刷要求を送信します。z/OS 印刷サーバー (IBM Infoprint Server など) は NetSpool または類似の VTAM コンポーネントで印刷要求をインターセプトし、JES 出力パラメーターで JES スプールに印刷出力データセットを作成します。JES 出力パラメータは、印刷サーバーが特定のネットワークプリンターに出力を送信するために使用されるルーティング情報を指定します。VTAMという用語は z/OS Communications Server と System Network Architecture (SNA) サービス要素を指します。
6. 印刷出力送信コンポーネントは、JES スプールからの出力印刷データセットを、LRS (このパターンで示されている)、IBM InfoPrint Server、または電子メールの宛先などのリモートプリンタまたは印刷サーバーに送信します。

ターゲットアーキテクチャ

次の図は、AWS クラウドにデプロイされるメインフレームのオンライン印刷ワークロードのアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. ユーザーはオンライン (CICS) ユーザーインターフェイスから印刷リクエストを開始し、請求明細書、ID カード、ローン明細書などの印刷出力を作成します。
2. メインフレームオンラインアプリケーション (「Amazon EC2にリプラットフォーム」) は、Micro Focus Enterprise Serverランタイムを使用してアプリケーションデータベースからデータを抽出し、ビジネスロジックをデータに適用し、データをフォーマットしてから、「Micro Focus CICS Print Exit」(DFHUPRNT) を使用してデータを印刷先に送信します。
3. アプリケーションデータベース (Amazon RDS 上で実行する SoR) は、印刷出力用のデータを永続化します。
4. LRS VPSX/MFI 印刷ソリューションは Amazon EC2 にデプロイされ、運用データは Amazon Elastic Block Store (Amazon EBS) に保存されます。LRS VPSX/MFI は TCP/IP ベースの LRS/キュートランスミッションエージェントで、Micro Focus CICS Print Exit API (DFHUPRNT) を介して印刷データを収集し、そのデータを指定されたプリンター宛先に配信します。最新の CICS アプリケーションで使用される元の TERMID (TERM) が VPSX/MFI Queue 名として使用されます。

> **Note**
>
> ターゲットソリューションでは、通常、IBM Advanced Function Presentation (AFP) や Xerox Line Condition Data Stream (LCDS) などのメインフレームフォーマット言語に対応するためにアプリケーションを変更する必要はありません。Micro Focus を使用して AWS 上のメインフレームアプリケーションの移行とモダナイズを行う方法の詳細については、AWS ドキュメントの「Micro Focus による AWS で Empowering Enterprise Mainframe Workloads の強化」を参照してください。

AWS インフラストラクチャアーキテクチャ

次の図は、メインフレームのオンライン印刷ワークロードに向けた可用性と安全性に優れた AWS インフラストラクチャアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. メインフレームオンラインアプリケーション (CICS や COBOL などのプログラミング言語で書かれている) は、コアビジネスロジックで、請求明細書、ID カード、ローン明細書などの印刷出力を処理し、生成します。オンラインアプリケーションは、高可用性 (HA) を実現するために Amazon EC2 で 2 つの「アベイラビリティーゾーン」(AZ) にまたがってデプロイされます。また、Micro Focus CICS Print Exit で印刷出力を LRS VPSX/MFI にルーティングし、エンドユーザーが印刷できるようにします。
2. LRS VPSX/MFI は TCP/IP ベースの LRS/Queue 送信エージェントで、Micro Focus オンライン Print Exitプログラミングインターフェイスから印刷データを収集またはキャプチャします。オンライン Print Exit は、LRS VPSX/MFI が印刷ファイルを効果的に処理し、LRS/Queue コマンドを動的に作成できるようにするために必要な情報を渡します。

> **Note**
>
> 印刷用のさまざまな CICS アプリケーションプログラミング方法と、Micro Focus Enterprise サーバーおよび LRS VPSX/MFI でサポートされる方法の詳細については、このパターンの追加情報セクションの「データキャプチャの印刷」を参照してください。

3.

> **Note**
>
> 「Network Load Balancer」は、Micro Focus Enterprise ServerをLRS VPSX/MFI と統合するための DNS 名を提供しています。: LRS VPSX/MFI はレイヤー 4 ロードバランサーをサポートします。また、Network Load Balancer は LRS VPSX/MFI の基本的なヘルスチェックを行い、登録されている正常なターゲットにトラフィックをルーティングします。

4. LRS VPSX/MFI プリントサーバーは、高可用性を実現するために Amazon EC2 で 2 つのアベイラビリティーゾーン (AZ) にまたがってデプロイされており、運用データストアとして「Amazon EBS」を使用しています。LRS VPSX/MFI は、アクティブ-アクティブとアクティブ-パッシブの両方のサービスモードをサポートします。このアーキテクチャでは、アクティブ/パッシブペアの複数の Availability Zones をアクティブスなホットスタンバイとして使用します。アクティブなインスタンスの状態が異常な場合、Network Load Balancer は LRS VPSX/MFI EC2 インスタンスのヘルスチェックを実行し、別の Availability Zones のホットスタンバイインスタンスにトラフィックをルーティングします。印刷リクエストは、各 EC2 インスタンスの LRS Job Queue にローカルに保持されます。復旧した場合、LRS サービスが印刷リクエストの処理を再開するには、障害が発生したインスタンスを再起動する必要があります。

> **Note**
>
> LRS VPSX/MFI は、プリンターフリートレベルでヘルスチェックを実行することもできます。詳細は、このパターンの [追加情報] セクションにある [プリンターフリートのヘルスチェック] を参照してください。

5. AWS Managed Microsoft AD は LRS/DIS と統合して、印刷ワークフローの認証と認可を実行します。詳細については、このパターンの「追加情報」セクションの「認証と認可の印刷」を参照してください。
6. LRS VPSX/MFI はブロックストレージに Amazon EBS を使用しています。アクティブな EC2 インスタンスからAmazon EBS データをポイントインタイムスナップショットとして Amazon S3 にバックアップし、ホットスタンバイ EBS ボリュームに復元できます。Amazon EBS ボリュームスナップショットの作成、保持、削除を自動化するには、「Amazon Data Lifecycle Manager」で自動スナップショットの頻度を設定し、「RTO/RPO 要件」に基づいて復元することができます。

ツール

AWS サービス

- Amazon Elastic Block Store (Amazon EBS) は、Amazon EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。EBS ボリュームの動作は、未初期化のブロックデバイスに似ています。これらのボリュームは、デバイスとしてインスタンスにマウントできます。
- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- 「Microsoft Active Directory (AD) 用 AWS Directory Service」は、AWS Managed Microsoft Active Directory とも呼ばれ、ディレクトリ対応のワークロードと AWS リソースが AWS のマネージド型 Active Directory を使用できるようにします。

その他のツール

- 「LRS VPSX/MFI (Micro Focus Interface)」は、LRS と Micro Focusと連携して、Micro Focus Enterprise Server JES スプールからの出力をキャプチャし、指定された印刷先に確実に配信します。
- LRS Directory Information Server (LRS/DIS) は、印刷ワークフロー中の認証と認可に使用されます。
- LRS/Queue は TCP/IP ベースの LRS/Queue 送信エージェントで、LRS VPSX/MFIに使用され、Micro Focus オンラインPrint Exitプログラミングインターフェイスから印刷データを収集またはキャプチャします。
- 「Micro Focus Enterprise Server」は、メインフレームアプリケーション用アプリケーションのデプロイです。Micro Focus Enterprise Developer の任意のバージョンで移行または作成されたメインフレームアプリケーションの実行環境を提供します。

エピック

Amazon EC2 で Micro Focus Enterprise Server を設定し、メインフレームオンラインアプリケーションをデプロイします。

タスク	説明	必要なスキル
Micro Focus Enterprise Server を設定し、デモオンラインアプリケーションをデプロイします。	Amazon EC2 で Micro Focus Enterprise Server を設定し、「Micro Focus ドキュメント」の「Tutorial: CICS Support」の指示に従い、Micro Focus Account Demo アプリケーション (ACCT Demo) を Amazon EC2 にデプロイします。 ACCT Demo アプリケーションは、印刷出力を作成して開始するメインフレームオンライン (CICS) アプリケーションです。	クラウドアーキテクト

Amazon EC2 で LRS プリントサーバーを設定

タスク	説明	必要なスキル
印刷用の LRS 製品ライセンスを取得します。	LRS VPSX/MFI、LRS/Queue と LRS/DIS の LRS 製品ライセンスを取得するには、「LRS Output Management チーム」にお問い合わせください。LRS 製品をインストールする EC2 インスタンスのホスト名を指定する必要があります。	ビルドリード
Amazon EC2 Windows インスタンスを作成して LRS VPSX/MFI を作成します。	Amazon EC2 ドキュメントの「ステップ 1: インスタンスを起動する」の次の指示に従い、Amazon EC2 Windows インスタンスを起動します。インスタンスは、LRS VPSX/MFI に対する次のハードウェア要件とソフトウェア要件を満たしている必要があります。 • CPU – Dual Core • RAM — 16 GB • ドライブ — 500 GB • 最小限の EC2 インスタンス — m5.xlarge • OS – Windows/Linux • ソフトウェア — Internet Information Service (IIS) または Apache	クラウドアーキテクト

タスク	説明	必要なスキル
	Note 上記のハードウェアとソフトウェアの要件は、小規模なプリンターフリート (約 500 ~ 1000) を対象としています。すべての要件については、LRS と AWS の担当者にお問い合わせください。 Windows インスタンスの作成時に、次のようにしてください。 1. EC2 ホスト名が LRS 製品ライセンスに使用されているホスト名と同じものであることを確認します。 2. 以下の操作を実行して Amazon EC2 で CGI を有効にします。 a. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の指示に従い、 EC2 インスタンスに接続します。 b. Windows スタートメニューで、[Server Manager] を見つけて開きます。	

タスク	説明	必要なスキル
	c. サーバーマネージャーで、[ダッシュボード]、[クイックスタート]、[ロールと機能の追加] を選択します。次に、「サーバーロール」を選択します。 d. [サーバーロール] で [WebServer (IIS)] を選択し、[アプリケーション開発] を選択します。 e. [アプリケーション開発] で、[CGI] チェックボックスを選択します。 f. Windows Server Manager の役割と機能の追加ウィザードの指示に従い、CGI をインストールします。 g. LRS/Queue 通信用のEC2 インスタンスの Windows ファイアウォールのポート 5500 を開きます。	

タスク	説明	必要なスキル
EC2 インスタンスに LRS VPSX/MFI をインストールします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の指示に従い、 EC2 インスタンスに接続します。 2. **Note** 受信するはずの LRS 電子メールから製品のダウンロードページへのリンクを開きます。: LRS 製品は電子ファイル転送 (EFT) によって配布されます。 3. LRS VPSX/MFI をダウンロードし、ファイル (デフォルトフォルダー:c:\LRS) を解凍します。 4. 解凍したフォルダーから LRS 製品インストーラーを起動し、LRS VPSX/MFI をインストールします。 5. [機能選択] メニューで [VPSX® サーバー (V1R3.022)] を選択し、[次へ] を選択してインストールプロセスを開始します。インストールが完了すると、正常に終了したことを示すメッセージが表示されます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/Queue をインストールします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い、Micro Focus Enterprise Server EC2 インスタンスに接続します。 2. 受信するはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/Queue をダウンロードし、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS 製品インストーラーを起動して LRS/Queue をインストールします。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS をインストールします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. 受信するはずの LRS 電子メールから LRS 製品のダウンロードページへのリンクを開き、LRS/DIS をダウンロードし、ファイルを解凍します。 3. ファイルをダウンロードした位置に移動し、LRS Product Installer を起動して、このファイルをダウンロードします。 4. LRS Product Installer で [LRS Misc Tools] を展開し、[LRS DIS] を選択し、[次へ] を選択します。 5. LRS Product Installer の残りの指示に従い、インストールプロセスを完了します。	クラウドアーキテクト

タスク	説明	必要なスキル
ターゲットグループを作成し、LRS VPSX/MFI EC2 をターゲットとして登録します。	Elastic Load Balancing ドキュメントの「Network Load Balancer のターゲットグループを作成する」の次の指示に従い、ターゲットグループを作成します。 ターゲットグループを作成する際には、以下のようにしてください。 1. [グループの詳細を指定] ページの[ターゲットタイプ] を選択し、[インスタンス] を選択します。 2. プロトコルには [TCP] を選択します。 3. ポートは 5500 を選択します。 4. [ターゲットの登録] ページの [利用可能なインスタンス] セクションで、LRS VPSX/MFI EC2 インスタンスを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
Network Load Balancer を作成します。	「Elastic Load Balancing ドキュメント」の「Network Load Balancer を作成する」の次の指示に従います。Network Load Balancer は、Micro Focus Enterprise Server から LRS VPSX/MFI EC2 にトラフィックをルーティングします。 Network Load Balancer を作成し、[リスナーとルーティング] ページで次の操作を実行します。 1. [プロトコル] で [TCP] を選択します。 2. ポートは 5500 を選択します。 3. デフォルトアクションには、先ほど作成したターゲットグループに対して [転送先] を選択します。	クラウドアーキテクト

Micro Focus Enterprise Serverを LRS VPSX/MFI と LRS/Queue と統合

タスク	説明	必要なスキル
LRS/LRS/Queue 統合用に Micro Focus Enterprise Server を設定します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い、Micro Focus Enterprise Server EC2 インスタンスに接続します。	クラウドアーキテクト

タスク	説明	必要なスキル
	2. Windows のスタートメニューで、Micro Focus Enterprise Server Administration UI を開きます。 3. メニューバーで NATIVE を選択します。 4. ナビゲーションペインで Directory Server を選択して、BANKDEMO または Enterpriseサーバーリージョンを選択します。 5. 左側のナビゲーションペインの全般から追加セクションまでスクロールし、LRSQ を指すように環境変数 (LRSQ_ADDRESS、LRSQ_PORT、LRSQ_COMMAND) を設定します。 6. [LRSQ_ADDRESS] には、前に作成したNetwork Load Balancer の IP アドレスまたは DNS 名を入力します。 7. LRSQ_PORT には、「VPSX LRSQ Listener Port (5500)」を入力します。 8. [LRSQ_COMMAND] には、LRSQ 実行ファイルのパスロケーションを入力します。	

タスク	説明	必要なスキル
	9. **Note** LRS は現在、DNS 名の最大文字数制限 50 をサポートしていますが、これは将来変更される可能性があります。DNS 名が 50文字 より大きい場合は、代わりに Network Load Balancer の IP アドレスを使用できます。	

タスク	説明	必要なスキル
CICS Print Exit (DFHUPRNT) を Micro Focus Enterprise Server の初期化に使用できるようにします。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い、Micro Focus Enterprise Server EC2 インスタンスに接続します。 2. **Note** CICS Print Exit (DFHUPRNT) を LRS VPSX/MFI 実行フォルダ (VPSX_MFI_R2 という) から Micro Focus Enterprise Server EC2 インスタンスの位置にコピーします。32 ビットシステムの場合、その位置は `C:\Program Files (x86)\Micro Focus\Enterprise Server\bin` です。64 ビットシステムの場合、その位置は `C:\Program Files (x86)\Micro Focus\Enterprise Server\bin64` です。: コピーDFHUPRNT.	クラウドアーキテクト

タスク	説明	必要なスキル
	dll 時にDFHUPRNT_64.dll ファイルの名前を に変更する必要があります。 Micro Focus Enterprise Server が CICS Print Exit (DFHUPRNT) を検出したことを確認します。 1. Micro Focus Enterprise Server を停止して起動します。 2. Micro Focus Enterprise ServerのAdministration パネルで、モニター、ログ、コンソールログを開きます。 3. コンソールログに「3270 プリンターのユーザー終了 DFHUPRNT が正常にインストールされました」というメッセージがあるか確認します。	

タスク	説明	必要なスキル
CICS プリンタの端末 ID (TERMID) を Micro Focus Enterprise Server として定義します。	MMicro Focus Enterprise Server で 3270 印刷を有効にします。 1. Micro Focus Enterprise ServerのAdministrationパネルで、CICS 、Resources 、By Group を開きます。 2. 左側のナビゲーションパネルから SIT (システム初期化テーブル)を選択して、BNKCICV を選択します。 3. 全般セクションで 3270 までスクロールし、3270 印刷チェックボックスを選択します。 CICS プリンタの端末を Micro Focus Enterprise Server として定義します。 1. Micro Focus Enterprise ServerのAdministrationパネルで、CICS 、Resources 、By Type を開きます。 2. 左のナビゲーションペインから、[期間] を選択し、[新規] を選択します。Create Terminal Resource フォームが開きます。 3. Name にLRS Print Queue の名前を入力します。(注：このパターンでは、CICS	クラウドアーキテクト

タスク	説明	必要なスキル
	プリンターの端末 ID と LRS VPSX Print Queue として「P275」を使用します)。 4. Group に BANKTERM を入力します。 5. 自動インストール — モデルに NO を入力します。 6. Terminal Identifiers - Terminal タイプに DFHPRT32 を入力します。 7. ネット名に VTAMP275 を入力します。 8. Terminal Usage では、In Service チェックボックスを選択します。 9. ページ上部をスクロールし、Save を選択します。 10[インストール] を選択します。ポップアップメッセージに、インストールが成功したことを示すメッセージが表示されます。	

Micro Focus Enterprise Server と LRS VPSX/MFI にプリンターとプリントユーザーを設定します。

タスク	説明	必要なスキル
LRS VPSX にプリントキューを作成します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/	クラウドアーキテクト

タスク	説明	必要なスキル
	MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューから VPSX ウェブインターフェイスを開きます。 3. ナビゲーションペインで [プリンタ] を選択します。 4. [追加] を選択し、[プリンタの追加] を選択します。 5. プリンタ設定ページのリンタ名に P275 を入力します。 6. [VPSX ID] には、VPS1 を入力します。 7. [CommType] には、[TCPIP/LRSQ] を選択します。 8. [ホスト/IP アドレス] には、追加する物理的プリンターの IP アドレスを入力します。 9. [デバイス] には、デバイスの名前を入力します。 10[Windows Driver] または [Linux/Mac Driver] のいずれかを選択します。 11[追加] を選択します。 Note 印刷キューは、Micro Focus Enterprise	

タスク	説明	必要なスキル
	Server で作成された印刷 TERMIDs と同等である必要があります。	

タスク	説明	必要なスキル
LRS VPSX/MFI でプリントユーザーを作成します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows のスタートメニューから VPSX ウェブインターフェイスを開きます。 3. ナビゲーションペインで [セキュリティ]、[ユーザー] の順に選択します。 4. ユーザー名列で管理者を選択して、コピーを選択します。 5. ユーザープロファイル管理ウィンドウのユーザー名に、ユーザー名 (たとえば、PrintUser) を入力します。 6. 説明には、簡単な説明 (たとえば、テスト印刷の場合は User) を入力します。 7. [Update] (更新) を選択します。これにより、印刷ユーザー (PrintUser など) を作成します。 8. ナビゲーションペインの User で、 作成した新しいユーザーを選択します。	クラウドアーキテクト

タスク	説明	必要なスキル
	9. コマンドメニューから [セキュリティ] を選択します。 10.セキュリティルールページで、該当するプリンタセキュリティとジョブセキュリティオプションをすべて選択して、保存を選択します。 11新しいプリントユーザーを管理者グループに追加するには、ナビゲーションペインに移動してセキュリティと設定を順に選択します。 12.セキュリティ設定ウィンドウで、新しいプリントユーザーを管理者列に追加します。	

印刷認証と認可の設定

タスク	説明	必要なスキル
ユーザーとグループを持つ AWS Managed Microsoft AD ドメインを作成します。	1. AWS Directory Service ドキュメントの「AWS Managed Microsoft AD ディレクトリの作成」の手順に従い、AWS Managed Microsoft AD にアクティブディレクトリを作成します。 2. AWS Directory Service ドキュメントの「ステッ	クラウドアーキテクト

タスク	説明	必要なスキル
	プ 3:EC2 インスタンスをデプロイして Managed Microsoft AD を管理する」の手順に従い、EC2 インスタンス (Active Directory マネージャー) をデプロイし、Active Directory ツールをインストールして AWS マネージド Microsoft AD を管理します。 3. **Note** 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の指示に従い、 EC2 インスタンスに接続します。: EC2 インスタンスに接続するときは、Windows セキュリティウィンドウに管理者認証情報 (ステップ 1 で作成したディレクトリ用) を入力します。 4. Windows スタートメニューから [Windows 管理ツール] を選択し、[Active Directory ユーザーとコンピュータ] を選択します。 5. 「AWS Directory Service ドキュメント」の「ユーザー	

タスク	説明	必要なスキル
	の作成」の手順に従い、Active Directory ドメインにプリントユーザーを作成します。	
LRS VPSX/MFI EC2 を AWS Managed Microsoft AD ドメインに接続します。	LRS VPSX/MFI EC2 を AWS Managed Microsoft AD ドメイン に「自動」(AWS ナレッジセンターのドキュメント) または 「手動」(AWS Directory Service ドキュメント) で参加させます。	クラウドアーキテクト

タスク	説明	必要なスキル
LRS/DIS を AWS Managed Microsoft AD と設定し、統合します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows の [スタート] メニューから「VPSX Web Interface」を開きます。 3. ナビゲーションペインで [セキュリティ]、[設定] の順に選択します。 4. Security Configuration ページの Security Parameters セクションで Security Type に Internal を選択します。 5. Security Parameters セクションに、残りのオプションの設定を入力します。 6. Microsoft Windows のスタートメニューから LRS Output Management フォルダを開き、Server Start と Server Stop を順に選択します。 7. Active Directory のユーザー名とパスワードを使用して LRS VPSX/MFI にログインします。	クラウドアーキテクト

オンライン印刷ワークフローのテスト

タスク	説明	必要なスキル
Micro Focus ACCT Demo アプリからオンライン印刷リクエストを開始します。	1. **Note** Micro Focus Enterprise Server EC2 インスタンスで TN3270 ターミナルエミュレータを開きます。(: このパターンでは、3270 個のターミナルエミュレータを使用します) 。 2. TN3270 ターミナルエミュレータ (Rumba) Cに接続します。ホスト名アドレスには 127.0.0.1 を使用します。Telnet Port には 9270 を使用します。 3. 3270 画面に接続したら、Ctrl+Shift+Z を押して画面をクリアします。 4. ACCT Demoアプリケーションを起動するには、クリア画面で「ACCT」を入力します。ACCT Demo オンライン (CICS) アプリケーションのメイン画面が開きます。注：メイン画面には、アカウントファイル、名前で検索するには、リクエストタイプ、アカウ	クラウドアーキテクト

タスク	説明	必要なスキル
	ント、プリンターなどのメニューオプションがあります。 5. ACCT Demo online (CICS) アプリケーションから印刷リクエストを送信するには、リクエストタイプフィールドに P、アカウントフィールドに 11111、プリンターフィールドに P275 を入力します。プリンターフィールドの値は、必ず CICS プリンタの端末 ID の値に設定してください。 6. [Enter] キーを押します。 画面下部に、「Print Request Scheduled」(印刷要求がスケジュールされています) というメッセージが表示されます。これにより、ACCT Demo アプリケーションからオンライン印刷要求が生成され、印刷処理のために LRS VPS/MFI に送信されたことが確認されます。	

タスク	説明	必要なスキル
LRS VPSX/MFI のプリント出力を確認します。	1. 「Amazon EC2 ドキュメント」の「ステップ 2: インスタンスに接続する」の次の指示に従い 、LRS VPSX/MFI EC2 インスタンスに接続します。 2. Windows スタートメニューから[VPSX Web Interface] を開きます。 3. ナビゲーションペインで [プリンター] を選択し、[出力キュー] を選択します。前にオンライン印刷用に作成した P275 印刷キューを探します。 4. プリントキュー (P275) の場合、Spool ID 列で、プリンターキュー内のリクエストのスプール ID を選択します。 5. [アクション] タブの [コマンド] 列で、[ブラウザ] を選択します。 これで、Account No.、SURNAME、FIRST、ADDRESS、TELEPHONE No.、Cards Issued、Date issued、Amount と Balance の各列を含む勘定取引明細書の印刷出力を確認できるようになりました。	テストエンジニア

タスク	説明	必要なスキル
	例として、このパターンの online_print_output 添付ファイルを参照してください。	

関連リソース

- 「LRS 出力のモダナイゼーション」(LRS ドキュメント)
- 「VTAM ネットワークの概念」(IBM ドキュメント)
- 「論理ユニット (LU) タイプの概要」 (IBM ドキュメント)
- 「ANSI とマシンキャリッジ制御」(IBM ドキュメント)
- 「Micro Focus による AWS 上のエンタープライズメインフレームワークロードの強化」(AWS パートナー ネットワークブログ)
- 「Amazon EC2 Auto Scaling とSystems Manager による Micro Focus エンタープライズサーバー PAC の構築」(AWS 規範ガイダンスのドキュメント)
- 「Advanced Function Presentation (AFP) データストリーム」(IBM ドキュメント)
- 「ラインコンディショニングデータストリーム (LCDS)」(Compart ドキュメント)

追加情報

考慮事項

モダナイゼーションを進める中で、メインフレームのオンラインプロセスとそれらが生成する出力について、さまざまな構成を検討するかもしれません。メインフレームプラットフォームは、印刷に直接影響する特定の要件に合わせて、使用するすべての顧客とベンダーによってカスタマイズされています。たとえば、現在のプラットフォームでは、IBM Advanced Function Presentation (AFP) や Xerox Line Condition Data Stream (LCDS) が現在のワークフローに組み込まれている場合があります。さらに、「メインフレームのキャリッジコントロール文字」や「チャネルコマンドワード」は、印刷ページの外観に影響を与える可能性があり、特別な処理が必要な場合もあります。モダナイゼーション計画プロセスの一環として、特定の印刷環境における構成を評価し、理解しておくことをお勧めします。

印刷データキャプチャ

このセクションでは、IBM メインフレーム環境で印刷に使用できる CICS アプリケーションプログラミング方法をまとめています。LRS VPSX/MFI コンポーネントは、同じアプリケーションプログラムが同じ方法でデータを作成できるようにする技術を提供しています。次の表は、AWS と LRS VPSX/MFI プリントサーバーを備えた Micro Focus Enterprise Server で実行される最新の CICS アプリケーションで、各アプリケーションプログラミング方法がどのようにサポートされているかを示しています。

[メソッド]	説明	モダナイズされた環境でのメソッドに対するサポート
EEXEC CICS SEND TEXT.. or EXEC CICS SEND MAP..	これらの CICS メソッドと VTAM メソッドは、3270/SCS 印刷データストリームを作成して LUTYPE0、LUTYPE1 と LUTYPE3 プリントデバイスに配信する役割を果たします。	Micro Focus オンライン Print Exit (DFHUPRNT) アプリケーションプログラムインターフェイス (API) を使用すると、これらの方法のいずれかを使用して 3270/SCS 印刷データストリームが作成されたときに、VPSX/MFI で印刷データを処理できます。
EEXEC CICS SEND TEXT.. or EXEC CICS SEND MAP.. (第三者のIBMメインフレームソフトウェアとの連携)	この CICS メソッドと VTAM メソッドは、3270/SCS 印刷データストリームを作成して LUTYPE0、LUTYPE1 と LUTYPE3 プリントデバイスに配信する役割を果たします。第三者のソフトウェア製品は印刷データをインターセプトし、そのデータを ASA/MCH 制御文字を含む標準印刷形式のデータに変換し、データを JES スプールに配置して、JES を使用するメインフレームベースの印刷システムで処理します。	Micro Focus オンライン Print Exit (DFHUPRNT) API を使用すると、これらの方法のいずれかを使用して 3270/SCS 印刷データストリームが作成されたときに、VPSX/MFI で印刷データを処理できます。

EXEC CICS SPOOLOPEN	このメソッドは CICS アプリケーションプログラムが JES スプールに直接データを書き込むことに使用します。その後、このデータは JES を使用するメインフレームベースの印刷システムで処理できるようになります。	Micro Focus Enterprise Server はデータをエンタープライズサーバースプールにスプールし、そこでデータをVPSXにスプールするVPSX/MFI Batch Print Exit (LRSPRTE6) で処理できます。
DRS/API	印刷データを JES に書き込むには、LRS 提供のプログラムインターフェースが使用されます。	VPSX/MFI は、印刷データを VPSX に直接スプールする代替インターフェースを提供しています。

プリンターフリートのヘルスチェック

LRS VPSX/MFI (LRS LoadX) は、デバイス管理や運用の最適化など、詳細なヘルスチェックを実行できます。デバイス管理では、プリンターデバイスの障害を検出し、印刷要求を正常なプリンターに転送できます。プリンターフリートの詳細なヘルスチェックについて、詳細は、製品ライセンスに含まれている LRS のドキュメントを参照してください。

印刷認証と認可

LRS/DIS を使用すると、LRS アプリケーションは Microsoft Active Directory または LDAP サーバーでユーザー ID とパスワードを認証できます。LRS/DIS は、基本的な印刷認可に加えて、次のようなユースケースでも粒度レベルの印刷セキュリティ制御を適用できます。

- プリンタジョブを参照できるユーザーを管理します。
- 他のユーザーのジョブの参照レベルを管理します。
- 運用タスクを管理します。たとえば、保留/リリース、削除、変更、コピー、ルート変更などのコマンドレベルのセキュリティなど。セキュリティは、ユーザー ID またはグループ (AD グループまたは LDAP グループと同様) のいずれかで設定できます。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Transfer Family を使用して、メインフレームファイルを Amazon S3 に直接移動する

作成:ルイス・グスタボ・ダンタス (AWS)

概要

モダナイゼーションの一環として、オンプレミスサーバーとAmazon Web Services (AWS) クラウド間でファイルを転送するという課題に直面する可能性があります。メインフレームからのデータ転送は、重要な課題となり得るのは、メインフレームは通常、Amazon Simple Storage Service (Amazon S3)、Amazon Elastic Block Store (Amazon EBS)、Amazon Elastic File System (Amazon EFS)のような最新のデータストアにアクセスできないからです。

多くのお客様は、オンプレミスの Linux、Unix、Windows サーバーなどの中間ステージングリソースを使用して、AWS クラウドにファイルを転送します。AWS Transfer Family とSecure Shell (SSH File Transfer Protocol (SFTP) を使用してメインフレームのファイルを Amazon S3 に直接アップロードすることで、この間接的な方法を回避できます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- レガシープラットフォームからアクセス可能なサブネットを持つ仮想プライベートクラウド (VPC)
- VPC のTransfer Family エンドポイント
- メインフレーム仮想ストレージアクセス方法 (VSAM) 「ファイルをシーケンシャルの固定長ファイルに変換しました」 (IBM ドキュメント)

機能制限

- SFTP はデフォルトでバイナリモードでファイルを転送します。つまり、ファイルは EBCDIC エンコーディングを保持したまま Amazon S3 にアップロードされます。ファイルにバイナリデータやパックデータが含まれていない場合は、sftp 「ascii サブコマンド」 (IBM ドキュメント) を使用して転送中にファイルをテキストに変換できます。
- ターゲット環境でこれらのファイルを使用するには、パックされた「バイナリコンテンツを含むメインフレームファイル」 (AWS 規範ガイダンス) を解凍する必要があります。

- Amazon S3 オブジェクトのサイズは最低 0 バイトから最大 5 TB までの範囲に設定することができます。Amazon S3 の諸機能について詳しくは、「Amazon S3 よくある質問」 を参照してください。

アーキテクチャ

ソーステクノロジースタック

- Job 制御言語 (JCL)
- z/OS UNIX シェルと ISPF
- SFTP
- VSAM とフラットファイル

ターゲットテクノロジースタック

- Transfer Family
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)

ターゲットアーキテクチャ

次の図は、Transfer Family と SFTP を使用してメインフレームファイルを S3 バケットに直接アップロードするためのリファレンスアーキテクチャを示しています。

この図表は、次のワークフローを示しています：

1. JCL ジョブを使用して、Direct Connect を通じてメインフレームファイルをレガシーメインフレームから AWS クラウドに転送します。
2. Direct Connect を使用すると、ネットワークトラフィックを AWS グローバルネットワークに留め、パブリックインターネットをバイパスできます。Direct Connect では、ネットワーク速度も向上し、50 Mbps から始まり、100 Gbps までスケールアップできます。
3. VPC エンドポイントは、パブリックインターネットを使用せずに VPC リソースとサポートされているサービス間の接続を可能にします。Transfer Family と Amazon S3 へのアクセスは、2 つのプライベートサブネットとアベイラビリティーゾーンにある伸縮自在なネットワークインターフェイスを通じて行われるため、高可用性が実現されます。

4. Transfer Family は、ユーザーを認証し、SFTP を使用してレガシー環境からファイルを受け取り、S3 バケットに移動します。

自動化とスケール

Transfer Family サービスの導入後は、JCL ジョブを SFTP クライアントとして使用して、メインフレームから Amazon S3 に無制限の数のファイルを転送できます。メインフレームファイルを転送する準備ができたら、メインフレームのバッチジョブスケジューラを使用して SFTP ジョブを実行することで、File Transfer を自動化することもできます。

ツール

- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。
- 「AWS Transfer Family」 を使用すると、SFTP、FTPS、および FTP プロトコルを使用して、Amazon S3 と Amazon EFS への定期的な企業間 File Transfer を安全にスケールすることができます。

エピック

S3 バケットとアクセスポリシーを作成します。

タスク	説明	必要なスキル
S3 バケットを作成する。	「S3 バケットを作成して」、レガシー環境から転送するファイルをホストします。	AWS 全般
IAM ロールとポリシーを作成します。	Transfer Family は、AWS Identity and Access Management (IAM) ロールを使用して、前の手順で作成し	AWS 全般

タスク	説明	必要なスキル
	た S3 バケットへのアクセスを許可します。 以下の「IAM ポリシー」を含む「IAM ロールを作成」します。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "UserFolderListing", "Action": [ "s3:ListBucket", "s3:GetBucketLocation" ], "Effect": "Allow", "Resource": [ "arn:aws:s3:::<your-bucket-name>" ] }, { "Sid": "HomeDirObjectAccess", "Effect": "Allow", "Action": [ "s3:PutObject", "s3:GetObjectAcl",`	

タスク	説明	必要なスキル
	"s3:GetObject", "s3:DeleteObjectVersion", "s3:DeleteObject", "s3:PutObjectAcl", "s3:GetObjectVersion"], "Resource": "arn:aws:s3:::<your-bucket-name>/*" }] } **Note** IAM ロールを作成するときは、Transfer ユースケースを選択する必要があります。	

転送サービスの定義

タスク	説明	必要なスキル
SFTP サーバを作成します。	1. AWS マネジメントコンソールにサ「インインし、Transfer Family コンソールを開いて」、サーバーの作成を選択します。	AWS 全般

タスク	説明	必要なスキル
	2. SFTP (SSH File Transfer Protocol) - Secure シェル プロトコルによるFile Transferのみを選択し、次へを選択します。 3. ID プロバイダーでサービス管理を選択し、次へを選択します。 4. Edit endpoint type (エンドポイントタイプ) で VPC hosted (VPC ホステッド) を選択します。 5. アクセスには内部を選択します。 6. [VPC] で、ユーザーの VPC を選択します。 7. アベイラビリティーゾーンセクションでは、アベイラビリティーゾーンとサブネットを選択します。 8. セキュリティグループセクションでは、セキュリティグループを選択して、次へを選択します。 9. ドメインで Amazon S3 を選択し、次へを選択します。 10. 追加詳細の設定ページのオプションはデフォルトのままにして、次へを選択します。 11. Create server (サーバーの作成) を選択します。	

タスク	説明	必要なスキル
	Note SFTP サーバーのセットアップ方法の詳細については、「SFTP 対応サーバーの作成」(AWS Transfer Family ユーザーガイド) を参照してください。	
サーバーのアドレスを取得します。	1. 「Transfer Family コンソールを開き」、サーバー ID 列でサーバー ID を選択します。 2. エンドポイントの詳細セクションのエンドポイントタイプで、エンドポイント ID を選択します。これにより、Amazon EC2 コンソールにガイドします。 3. Amazon VPC コンソールの詳細タブで、DNS 名の横にある DNS 名を探します。	AWS 全般
SFTP クライアント key pair を作成します。	「Microsoft Windows」 または 「macOS/Linux/UNIX」 用の SSH key pair を作成します。	AWS 全般、SSH

タスク	説明	必要なスキル
SFTP ユーザーを作成します。	1. 「Transfer Family コンソールを開き」、ナビゲーションペインからサーバーを選択し、サーバーを選択します。 2. サーバー ID 列で、サーバーのサーバー ID を選択し、ユーザーを追加を選択します。 3. [ユーザー名] には、SSH key pair ユーザー名と一致するユーザー名を入力します。 4. ロールに対して、前に作成したIAM ロールを選択します。 5. ホームディレクトリには、前の手順で作成した S3 バケットを選択します。 6. SSH 公開鍵の場合は、前に作成したkey pair を入力します。 7. [追加] を選択します。	AWS 全般

メインフレームファイルの転送

タスク	説明	必要なスキル
SSH 秘密鍵をメインフレームに送信します。	SFTP または SCP を使用して SSH プライベートキーをレガシー環境に送信します。 Transfer for SFTP	メインフレーム、z/OS UNIX シェル、FTP、SCP

タスク	説明	必要なスキル

	`sftp [USERNAME@mainframeIP]`
	`[password]`
	`cd [/u/USERNAME]`
	`put [your-key-pair-file]`
	
	SCP の例
	
	`scp [your-key-pair-file] [USERNAME@MainframeIP]:/[u/USERNAME]`
	
	次に、後でFile Transferバッチジョブ (例: `/u/CONTROLM`) を実行するユーザー名で SSH キーを z/OS UNIX ファイルシステムに保存します。
	
	Note
	z/OS Unix シェルの詳細については、「z/OS シェルの概要」(IBM ドキュメント) を参照してください。	

タスク	説明	必要なスキル

JCL SFTP クライアントを作成します。	メインフレームにはネイティブ SFTP クライアントがないため、BPXBATCH ユーティリティを使用して z/OS Unix シェルから SFTP クライアントを実行する必要があります。
	
	ISPF エディターで JCL SFTP クライアントを作成します。以下に例を示します。
	
	`//JOBNAM JOB ...`
	`//************`
	`******************`
	`******************`
	`******************`
	`****`
	`//SFTP EXEC PGM=BPXBA`
	`TCH,REGION=0M`
	`//STDPARM DD *`
	`SH cp "//'MAINF`
	`RAME.FILE.NAME'"`
	` filename.txt;`
	`echo 'put filename.txt'`
	` > uplcmd;`
	`sftp -b uplcmd -i`
	` ssh_private_key_fi`
	`le ssh_username@<tran`
	`sfer service ip or`
	` DNS>;`
	`//SYSPRINT DD SYSOUT=*`
	`//STDOUT DD SYSOUT=*`
	`//STDENV DD *`
	`//STDERR DD SYSOUT=*`
		JCL、メインフレーム、z/OS UNIX シェル

タスク	説明	必要なスキル
	Note z/OS Unix シェルでコマンドを実行する方法の詳細については、「BPXBATCH ユーティリティ」(IBM ドキュメント) を参照してください。z/OS で JCL ジョブを作成または編集する方法の詳細については、「ISPF とは」 および「ISPF エディター」(IBM ドキュメンテーション)を参照してください。	

タスク	説明	必要なスキル
JCL SFTP クライアントを実行します。	1. ISPF エディタで SUB と入力し、JCL ジョブが作成されたら Enter キーを押します。 2. SDSF でメインフレームの File Transferバッチジョブのアクティビティを監視します。 **Note** バッチジョブのアクティビティを確認する方法の詳細については、「z/OS SDSF ユーザーガイド (IBM ドキュメント)」を参照してください。	メインフレーム、JCL、ISPF
File Transferを検証します。	1. AWS マネジメントコンソールにサインインし、「Amazon S3 コンソール」を開いて、ナビゲーションペインから Buckets を選択します。 2. Transfer Family に関連付けられているバケットを選択します。 3. オブジェクトタブのオブジェクトセクションで、メインフレームから転送したファイルを見つけます。	AWS 全般

タスク	説明	必要なスキル
JCL SFTP クライアントを自動化します。	ジョブスケジューラを使用して JCL SFTP クライアントを自動的にトリガーします。 **Note** BMC Control-M や CA ワークロードオートメーションなどのメインフレームジョブスケジューラを使用して、時間やその他のバッチジョブの依存関係に基づいてファイル転送のバッチジョブを自動化できます。	ジョブスケジューラー

関連リソース

- 「AWS Transfer Family の仕組み」
- 「AWS によるメインフレームの近代化」

トラステッドコンテキストを使用して、AWS の Db2 フェデレーションデータベースのユーザーアクセスを保護し、合理化する

作成者: Sai Parthasaradhi (AWS)

概要

多くの企業がレガシーメインフレームワークロードをAmazon Web Services (AWS) に移行しています。この移行には、IBM Db2 for z/OS データベースを Amazon Elastic Compute Cloud (Amazon EC2) 上の Linux、Unix、および Windows (LUW) 用の Db2 (LUW) に移行することが含まれます。オンプレミスから AWS への段階的移行中、すべてのアプリケーションとデータベースが Db2 LUW に完全に移行されるまで、ユーザーは IBM Db2 z/OS と Amazon EC2 上の Db2 LUW のデータにアクセスする必要がある場合があります。このようなリモートデータアクセスのシナリオでは、プラットフォームが異なれば使用する認証メカニズムも異なるため、ユーザー認証は難しい場合があります。

このパターンは、Db2 for z/OS をリモートデータベースとして Db2 for LUW 上にフェデレーションサーバーをセットアップする方法を網羅しています。このパターンでは、トラステッドコンテキストを使用して、リモートデータベースで再認証することなく、ユーザーの ID を Db2 LUW から Db2 z/OS に伝達します。トラステッドコンテキストについて詳しくは、「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon EC2 インスタンスで実行されている Db2 インスタンス
- オンプレミスで実行されているリモート Db2 for z/OS データベース
- AWS Site-to-Site VPN または AWS Direct Connect を介して AWS に接続されたオンプレミスネットワーク

アーキテクチャ

ターゲット アーキテクチャ

ツール

サービス

- 「Amazon Elastic Compute Cloud (Amazon EC2)」は、AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- AWS Site-to-Site VPN は、AWS で起動するインスタンスと独自のリモートネットワーク間でトラフィックを渡すのに役立ちます。

その他のツール

- db2cli は Db2 のインタラクティブコマンドラインインターフェイス (CLI)コマンドです。

エピック

AWS で実行されている Db2 LUW データベースでフェデレーションを有効にする

タスク	説明	必要なスキル
DB2 LUW データベースでフェデレーションを有効にします。	DB2 LUW でフェデレーションを有効にするには、次のコマンドを実行します。 `update dbm cfg using federated YES`	DBA

データベースを再起動します。	データベースを再起動するには、次のコマンドを実行します。
	`db2stop force;`
	`db2start;`
		DBA

リモートデータベースをカタログ化する

タスク	説明	必要なスキル
リモート Db2 z/OS サブシステムをカタログ化します。	AWS で実行されている Db2 LUW 上のリモート Db2 z/OS データベースをカタログ化するには、以下のサンプルコマンドを使用します。 `catalog TCPIP NODE tcpnode REMOTE mainframehost SERVER mainframeport`	DBA
リモートデータベースをカタログ化します。	リモートデータベースをカタログするには、次のコマンド例を使用します。 `catalog db dbnam1 as ndbnam1 at node tcpnode`	DBA

リモートサーバー定義を作成する

タスク	説明	必要なスキル
リモート Db2 z/OS データベースのユーザー認証情報を収集します。	次の手順に進む前に、以下の情報を収集します。 • Db2 z/OS サブシステム名 – 前のステップで LUW にカタログ化された Db2 z/OS 名 (例: ndbnam1) • Db2 z/OS バージョン – Db2 z/OS サブシステムバージョン (例: 12)	DBA

タスク	説明	必要なスキル
	• Db2 z/OS ユーザー ID – サーバー定義のみを作成するために必要な BIND 権限を持つユーザー (例: `dbuser1`) • Db2 z/OS パスワード – `dbuser1` のパスワード (例: `dbpasswd`) • Db2 z/OS プロキシユーザー – 信頼できる接続を確立するために使用されるプロキシユーザーの ID (例: `zproxy`) • Db2 z/OS プロキシパスワード – `zproxy` ユーザーのパスワード (例: `zproxy`)	
DRDA ラッパーを作成します。	DRDA ラッパーを作成するには、次のコマンドを実行します。 `CREATE WRAPPER DRDA;`	DBA

タスク	説明	必要なスキル
サーバー定義を作成します。	サーバー定義を作成するには、次のコマンド例を実行します。 `CREATE SERVER ndbserver TYPE DB2/ZOS VERSION 12 WRAPPER DRDA AUTHORIZATION "dbuser1" PASSWORD "dbpasswd " OPTIONS ( DBNAME 'ndbnam1',FED_PROX Y_USER 'ZPROXY' );` この定義では、FED_PROXY_USER が Db2 z/OS データベースへの信頼できる接続を確立するために使用するプロキシユーザーを指定します。認可ユーザー ID とパスワードは、Db2 LUW データベースにリモートサーバーオブジェクトを作成する場合にのみ必要です。これらは後のランタイムには使用されません。	DBA

ユーザーマッピングの作成

タスク	説明	必要なスキル
プロキシユーザーのユーザーマッピングを作成します。	プロキシユーザーのユーザーマッピングを作成するには、次のコマンドを実行します。 `CREATE USER MAPPING FOR ZPROXY SERVER ndbserver OPTIONS (REMOTE_AUTHID`	DBA

タスク	説明	必要なスキル
	`'ZPROXY', REMOTE_PA SSWORD 'zproxy');`	
Db2 LUW で各ユーザーのユーザーマッピングを作成します。	プロキシユーザーを介してリモートデータにアクセスする必要がある AWS 上の Db2 LUW データベース上のすべてのユーザーのユーザーマッピングを作成します。ユーザーマッピングを作成するには、次のコマンドを実行します。 `CREATE USER MAPPING FOR PERSON1 SERVER ndbserver OPTIONS (REMOTE_AUTHID 'USERZID', USE_TRUST ED_CONTEXT 'Y');` このステートメントは、Db2 LUW (PERSON1) 上のユーザーがリモート Db2 z/OS データベース (USE_TRUST ED_CONTEXT 'Y') への信頼できる接続を確立できることを明記しています。プロキシユーザーを介して接続が確立されると、ユーザーは Db2 z/OS ユーザー ID (REMOTE_AUTHID 'USERZID') を使用してデータにアクセスできます。	DBA

トラステッドコンテキストオブジェクトを作成する

タスク	説明	必要なスキル
トラステッドコンテキストオブジェクトを作成します。	リモート Db2 z/OS データベースにトラステッドコンテキストオブジェクトを作成するには、以下のサンプルコマンドを使用します。 `CREATE TRUSTED CONTEXT CTX_LUW_ZOS BASED UPON CONNECTIO N USING SYSTEM AUTHID ZPROXY ATTRIBUTES ( ADDRESS '10.10.10.10' ) NO DEFAULT ROLE ENABLE WITH USE FOR PUBLIC WITHOUT AUTHENTIC ATION;` この定義では、CTX_LUW_ZOS はトラステッドコンテキストオブジェクトの任意の名前です。このオブジェクトには、信頼できる接続の発信元となるサーバーのプロキシユーザー ID と IP アドレスが含まれます。この例では、サーバーは AWS 上の Db2 LUW データベースです。IP アドレスの代わりにドメイン名を使用できます。この条項は、信頼できる接続でのユーザー ID の切り替えがすべてのユーザー ID で	DBA

タスク	説明	必要なスキル
	許可されていることを WITH USE FOR PUBLIC WITHOUT AUTHENTICATION 示しています。パスワードを入力する必要はありません。	

関連リソース

- IBM Resource Access Control Facility (RACF)
- IBM Db2 LUW Federation
- Trusted contexts

追加情報

Db2 トラステッドコンテキスト

トラステッドコンテキストは、フェデレーションサーバーとリモートデータベースサーバー間の信頼関係を定義する Db2 データベースオブジェクトです。信頼できる関係を定義するために、トラステッドコンテキストは信頼属性を指定します。信頼属性には、3 つのタイプがあります。

- 最初のデータベース接続要求を行うシステム許可 ID
- 接続元の IP アドレスまたはドメイン名
- データベースサーバーとデータベースクライアント間のデータ通信の暗号化設定

接続要求のすべての属性が、サーバー上で定義されているトラステッドコンテキストオブジェクトで指定されている属性と一致すると、信頼できる接続が確立されます。信頼接続には、暗黙的な接続と明示的な接続の 2 つのタイプがあります。暗黙的な信頼接続が確立されると、その信頼された接続定義の範囲外では使用できないロールがユーザーに継承されます。明示的な信頼された接続が確立されると、ユーザーは認証の有無にかかわらず、同じ物理接続に切り替えることができます。さらに、Db2 ユーザーには、信頼できる接続内でのみ使用できる権限を指定するロールを付与できます。このパターンでは、明示的な信頼できる接続を使用します。

このパターンのトラステッドコンテキスト

パターンが完成すると、Db2 LUW 上の PERSON1 は、フェデレーテッドトラステッドコンテキストを使用して Db2 z/OS のリモートデータにアクセスします。PERSON1 への接続がトラステッドコンテキスト定義で指定されている IP アドレスまたはドメイン名から発信されている場合、PERSON1 への接続はプロキシユーザーを介して確立されます。接続が確立されると、PERSON1 の対応する Db2 z/OS ユーザー ID は再認証なしで切り替えられ、ユーザーはそのユーザーに設定された Db2 権限に基づいてデータまたはオブジェクトにアクセスできます。

フェデレーションされたトラステッドコンテキストの利点

- このアプローチでは、すべてのユーザーが必要とするすべての特権のスーパーセットを必要とする共通のユーザー ID やアプリケーション ID を使用する必要がなくなるため、最小特権という原則が維持されます。
- フェデレーテッドデータベースとリモートデータベースの両方でトランザクションを実行するユーザーの実際の ID は常に知られており、監査することができます。
- フェデレーションサーバーによる再認証を必要とせずに物理接続がユーザー間で再利用されるので、パフォーマンスが向上します。

大規模な Db2 z/OS データを CSV ファイルで Amazon S3 に転送する

作成者:Bruno Sahinoglu (AWS), Ivan Schuster (AWS), and Abhijit Kshirsagar (AWS)

概要

メインフレームは今でも多くの企業の記録システムであり、現在および過去のビジネストランザクションの記録を持つマスターデータエンティティを含む大量のデータを含んでいます。多くの場合、サイロ化されており、同じ企業内の分散システムからは簡単にはアクセスできません。クラウドテクノロジーの出現とビッグデータの民主化に伴い、企業はメインフレームデータに隠された洞察を利用して新しいビジネス機能を開発することに関心を持っています。

この目標を掲げて、エンタプライズはメインフレームの Db2 データをAmazon Web Services (AWS) クラウド環境に公開しようとしています。ビジネス上の理由はいくつかあり、転送方法はケースごとに異なります。アプリケーションをメインフレームに直接接続したい場合もあれば、データをほぼリアルタイムで複製したい場合もあります。ユースケースがデータウェアハウスまたはデータレイクへのフィードである場合、最新のコピーを用意することはもはや問題ではなく、特にサードパーティ製品のライセンスコストを避けたい場合は、このパターンで説明されている手順で十分かもしれません。別の使用例としては、移行プロジェクトのメインフレームデータ転送が挙げられます。移行シナリオでは、機能同等性テストを実行するにはデータが必要です。この記事で説明するアプローチは、Db2 データをAWS クラウド環境に転送するための費用対効果の高い方法です。

Amazon Simple Storage Service (Amazon S3) は最も統合された AWS サービスの 1 つであるため、そこからデータにアクセスし、Amazon Athena、AWS Lambda 関数、Amazon QuickSight などの他の AWS サービスを使用して直接洞察を収集できます。AWS Glue または AWS Database Migration Service (AWS DMS) を使用して Amazon Aurora または Amazon DynamoDB にデータをロードすることもできます。その目的を念頭に置いて、メインフレームで ASCII 形式の CSV ファイルの Db2 データをアンロードし、ファイルを Amazon S3 に転送する方法について説明します。

この目的のために、必要な数の Db2 テーブルをアンロードして転送するためのジョブ制御言語 (JCL) を生成するのに役立つ「メインフレームスクリプト」が開発されました。

前提条件と制限

前提条件

- 再構築拡張エグゼキューター (REXX) と JCL スクリプトを実行する権限を持つ IBM z/OS オペレーティングシステムユーザー。
- z/OS Unix システムサービス (USS) にアクセスして SSH (セキュアシェル) のプライベートキーとパブリックキーを生成します。

- 書き込み可能な S3 バケット。詳細については、Amazon S3 ドキュメントの「最初の S3 バケットの作成」を参照してください。
- AWS Transfer Family SSH ファイル転送プロトコル (SFTP) 対応サーバー。ID プロバイダーとして [サービスマネージド]、AWS ストレージサービスとして Amazon S3 を使用しています。詳細については、AWS Transfer Family ドキュメントの「SFTP 対応サーバーの作成」を参照してください。

制約事項

- この方法は、ほぼリアルタイムまたはリアルタイムのデータ同期には適していません。
- データはDb2 z/OSからAmazon S3 にのみ移動でき、その逆はできません。

アーキテクチャ

ソーステクノロジースタック

- z/OS で Db2 を実行するメインフレーム

ターゲットテクノロジースタック

- AWS Transfer Family
- Amazon S3
- Amazon Athena
- Amazon QuickSight
- AWS Glue
- Amazon Relational Database Service (Amazon RDS)
- Amazon Aurora
- Amazon Redshift

ソースアーキテクチャとターゲットアーキテクチャ

次の図は、Db2 z/OS データを ASCII CSV 形式で生成、抽出、S3 バケットに転送するプロセスを示しています。

1. Db2 カタログからデータを移行するテーブルのリストが選択されます。
2. このリストは、外部フォーマットの数値列とデータ列を含むアンロードジョブの生成に使用されます。
3. その後、データは AWS Transfer Family を使用して Amazon S3 に転送されます。
4. AWS Glue の抽出、変換、ロード (ETL) ジョブでは、データを変換し、指定された形式で処理済みのバケットにロードできます。また、AWS Glue はデータをデータベースに直接フィードできます。
5. Amazon Athena と Amazon QuickSight を使用してデータをクエリしてレンダリングし、分析に役立てることができます。

次の図は、プロセス全体の論理的な流れを示しています。

1. 最初の JCL は TABNAME と呼ばれ、Db2 ユーティリティ DSNTIAUL を使用して、Db2 からアンロードする予定のテーブルのリストを抽出し、生成します。テーブルを選択するには、SQL 入力を手動で調整して選択し、フィルター条件を追加して 1 つ以上の Db2 スキーマを含める必要があります。
2. REXXEXEC と呼び出された 2 番目の JCL は、JCL スケルトンと REXX プログラムを使用して JCL TABNAME によって作成されたテーブルリストを処理し、テーブル名ごとに 1 つの JCL を生成します。各 JCL には、テーブルをアンロードするステップと、SFTP プロトコルを使用して S3 バケットにファイルを送信するステップが 1 つずつ含まれます。
3. 最後のステップは、JCL を実行してテーブルをアンロードし、ファイルを AWS に転送することです。プロセス全体をオンプレミスまたは AWS のスケジューラーを使用して自動化できます。

ツール

AWS サービス

- 「Amazon Athena」は、標準 SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析できるようにするインタラクティブなクエリサービスです。
- 「Amazon Aurora」はクラウド用に構築されたフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。
- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でデータを確実に分類、整理、強化、移動できます。

- 「Amazon QuickSight」は、クラウドスケールのビジネスインテリジェンス (BI) サービスで、データを単一のダッシュボードで可視化、分析、レポートすることができます。
- 「Amazon Redshift」は、AWS クラウド内でのマネージド型、ペタバイトスケールのデータウェアハウスサービスです。
- Amazon Relational Database Service (Amazon RDS) を使用して、AWS クラウドでリレーショナルデータベース (DB) をセットアップ、運用、スケーリングできます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- 「AWS Transfer Family」は、AWS ストレージサービスとの間でファイルを送受信できる安全な転送サービスです。

メインフレームツール

- 「SSH ファイル転送プロトコル (SFTP)」は、サーバーへのリモートログインとサーバー間のファイル転送を可能にする安全なファイル転送プロトコルです。SSH はすべてのトラフィックを暗号化することでセキュリティを確保します。
- 「DSNTIAUL」は IBM が提供するデータをアンロードするためのサンプルプログラムです。
- 「DSNUTILB」は IBM が提供するユーティリティー・バッチ・プログラムで、DSNTIAUL のさまざまなオプションを使用してデータをアンロードします。
- 「z/OS OpenSSH」は、IBM オペレーティングシステム z/OS の Unix システムサービス上で動作するオープンソースソフトウェア SSH のポートです。SSH は TCP/IP ネットワーク上で稼働している 2 台のコンピューター間の、安全で暗号化された接続プログラムです。ssh-keygen を含む複数のユーティリティが用意されています。
- 「REXX (再構築拡張エグゼキューター)」スクリプトは、Db2 アンロードおよび SFTP ステップによる JCL 生成を自動化するために使用されます。

コード

このパターンのコードは GitHub 内の「unloaddb2」リポジトリで利用できます。

ベストプラクティス

最初のアンロードでは、生成された JCL がテーブルデータ全体をアンロードする必要があります。

最初の完全アンロード後は、パフォーマンスを向上させコスト削減のために段階的にアンロードを実行します。アンロードプロセスに加えられた変更に対応できるように、テンプレート JCL デックの SQL クエリを更新します。

スキーマは手動で変換することも、Db2 SYSPUNCH を入力として Lambda のスクリプトを使用して変換することもできます。工業プロセスでは、「AWS Schema Conversion Tool (SCT)」が推奨オプションです。

最後に、メインフレームベースのスケジューラーを使用するか、メインフレームにエージェントを配置したAWS のスケジューラーを使用して、プロセス全体の管理と自動化を支援します。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを作成する。	手順については、「最初の S3 バケットを作成する」を参照してください。	AWS 全般

Transfer Family サーバーを設定します。

タスク	説明	必要なスキル
SFTP 対応サーバーを作成します。	「AWS Transfer Family コンソール」で SFTP サーバーを開いて作成するには、次の手順を実行します。 1. プロトコルの選択ページで、SFTP (SSH File Transfer Protocol) — セキュアシェル経由のファイル転送チェックボックスを選択します。	AWS 全般

タスク	説明	必要なスキル
	2. ID プロバイダーには、[サービスマネージド] を選択します。 3. エンドポイントには [パブリックにアクセス可能] を選択します。 4. ドメインには [Amazon S3] を選択してください。 5. [追加の情報を設定] ページで、デフォルトの設定を維持します。 6. サーバーを作成します。	
Transfer Family 用の IAM ロールを作成します。	Transfer Family が Amazon S3 にアクセスするための AWS Identity and Access Management (IAM) ロールを作成するには、「IAM ロールおよびポリシーを作成する」の指示に従ってください。	AWS 管理者
Amazon S3 サービスマネージドユーザーをサーバーに追加します。	Amazon S3 サービスマネージドユーザーを追加するには、「AWS ドキュメント」の指示に従い、メインフレームユーザー ID を使用してください。	AWS 全般

通信プロトコルの保護

タスク	説明	必要なスキル
SSH キーを作成します。	メインフレーム USS 環境で、次のコマンドを実行します。	メインフレーム開発者

タスク	説明	必要なスキル
	`ssh-keygen -t rsa` **Note** パスフレーズの入力を求められたら、空のままにします。	

SSH フォルダとキーファイルに適切な権限レベルを設定します。	デフォルトでは、パブリックキーとプライベートキーはユーザーディレクトリ `/u/home/username/.ssh` に保存されます。
	
	キーファイルには 644、フォルダーには 700 の権限を与える必要があります。
	
	`chmod 644 .ssh/id_rsa`
	`chmod 700 .ssh`
		メインフレーム開発者

タスク	説明	必要なスキル
Amazon S3 サービスマネージドユーザーにパブリックキーの内容をコピーします。	USS で生成されたパブリックキーの内容をコピーするには、「AWS Transfer Family コンソール」を開きます。 1. ナビゲーションペインで [Servers] (サーバー) を選択します。 2. サーバーの詳細を表示するには、[サーバーID] 列の識別子を選択します。 3. [ユーザー] で、ユーザー名を選択すると [ユーザーの詳細] ページが表示されます。 4. [SSH パブリックキー] で [SSH パブリックキーの追加] を選択して、パブリックキーをユーザーに追加します。SSH パブリックキーには、パブリックキーを入力します。新しいユーザーを追加する前に、サービスによってキーが検証されます。 5. [Add key] (キーの追加) を選択します。	メインフレーム開発者

JCL を生成する

タスク	説明	必要なスキル
対象範囲内の Db2 テーブルリストを生成します。	入力 SQL を指定して、データ移行の対象となるテーブルのリストを作成します。このステップでは、SQL WHERE 句を使用して Db2 カタログテーブル SYSIBM.SYSTABLES をクエリする選択基準を指定する必要があります。フィルターは、特定のプレフィックスで始まる特定のスキーマ名やテーブル名を含むように、またはインクリメンタルアンロードのタイムスタンプに基づいてカスタマイズできます。出力はメインフレーム上のフィジカルシーケンシャル (PS) データセットに取り込まれます。このデータセットは、JCL 生成の次のフェーズの入力として機能します。 JCL TABNAME (必要に応じて名前を変更できます) を使用する前に、次の変更を行います。 1. <Jobcard>をジョブクラスと Db2 ユーティリティを実行する権限を持つユーザーに置き換えます。 2. <HLQ1>出力データセットの名前を、サイトの標準に	メインフレーム開発者

タスク	説明	必要なスキル

	合わせて置き換えたり、カスタマイズしたりします。
	3. サイトの標準に従って、PDS の STEPLIB スタック (パーティションデータセットの拡張) を更新します。このパターンの例では IBM のデフォルトを使用しています。
	4. PLAN 名と LIB は、インストール固有の値に置き換えてください。
	5. <Schema> と <Prefix> は、Db2 カタログの選択基準に応じて置き換えてください。
	6. 結果の JCL を PDS (パーティションデータセット) ライブラリに保存します。
	7. JCL を送信します。
	
	Db2 テーブルリスト抽出ジョブ
	
	`<Jobcard>`
	`//*`
	`//* UNLOAD ALL THE TABLE`
	`NAMES FOR A PARTICULAR`
	`SCHEMA`
	`//*`
	`//STEP01 EXEC PGM=IEFBR`
	`14`
	`//*`
	`//DD1 DD DISP=(MOD`
	`,DELETE,DELETE),`	

タスク	説明	必要なスキル

	``
	// UNIT=SYSDA,
	// SPACE=(1000,
	(1,1)),
	// DSN=<HLQ1
	>.DSN81210.TABLIST
	//*
	//DD2 DD DISP=(MOD
	,DELETE,DELETE),
	// UNIT=SYSDA,
	// SPACE=(1000,
	(1,1)),
	// DSN=<HLQ1
	>.DSN81210.SYSPUNCH
	//*
	//UNLOAD EXEC PGM=IKJEF
	T01,DYNAMNBR=20
	//SYSTSPRT DD
	 SYSOUT=*
	//STEPLIB DD
	 DISP=SHR,DSN=DSNC1
	0.DBCG.SDSNEXIT
	// DD DISP=SHR,
	DSN=DSNC10.SDSNLOAD
	// DD DISP=SHR,
	DSN=CEE.SCEERUN
	// DD DISP=SHR,
	DSN=DSNC10.DBCG.RU
	NLIB.LOAD
	//SYSTSIN DD *
	  DSN SYSTEM(DBCG)
	  RUN PROGRAM(D
	SNTIAUL) PLAN(DSNT
	IB12) PARMS('SQL') -
	       LIB('DSNC
	10.DBCG.RUNLIB.LOAD')
	  END
	//SYSPRINT DD SYSOUT=*
	//*
	//SYSUDUMP DD SYSOUT=*
	//*
	//SYSREC00 DD DISP=(NEW
	,CATLG,DELETE),
	``	

タスク	説明	必要なスキル

	<pre>// UNIT=SYSD
	A,SPACE=(32760,(10
	00,500)),
	// DSN=<HLQ1
	>.DSN81210.TABLIST
	//*
	//SYSPUNCH DD DISP=(NEW
	,CATLG,DELETE),
	// UNIT=SYSD
	A,SPACE=(32760,(10
	00,500)),
	// VOL=SER=S
	CR03,RECFM=FB,LREC
	L=120,BLKSIZE=12
	// DSN=<HLQ1
	>.DSN81210.SYSPUNCH
	//*
	//SYSIN DD *
	   SELECT CHAR(CREA
	TOR), CHAR(NAME)
	     FROM SYSIBM.SY
	STABLES
	    WHERE OWNER =
	 '<Schema>'
	      AND NAME LIKE
	 '<Prefix>%'
	      AND TYPE = 'T';
	/*</pre>	

タスク	説明	必要なスキル
JCL テンプレートを変更します。	このパターンで提供される JCL テンプレートには、一般的なジョブカードとライブラリ名が含まれています。ただし、ほとんどのメインフレームサイトには、データセット名、ライブラリ名、ジョブカードに独自の命名規則があります。たとえば、Db2 ジョブを実行するには特定のジョブクラスが必要な場合があります。JES2 と JES3 のJob 入力サブシステム実装では、さらに変更が加えられる可能性があります。標準ロード・ライブラリーには、IBM のデフォルトである SYS1 とは異なる第 1 修飾子が付いている場合があります。そのため、実行する前に、サイト固有の標準に合わせてテンプレートをカスタマイズしてください。 スケルトン JCL UNLDSKEL に以下の変更を加えます。 1. Db2 ユーティリティを実行する権限を持つジョブクラスとユーザーを使用してジョブカードを変更します。 2. 出力データセットの名前をサイトの標準に合わせてカスタマイズします。	メインフレーム開発者

タスク	説明	必要なスキル
	3. サイトの標準に従って PDS の STEPLIB スタックを更新します。このパターンの例では IBM のデフォルトを使用しています。 4. <DSN> を Db2 サブシステム名と相関 ID に置き換えてください。 5. 結果の JCL を ISPSLIB スタックの一部である PDS ライブラリ (ISPF の標準スケルトン・テンプレート・ライブラリー) に保存します。 JCL スケルトンをアンロードして SFTP 送信します。 `//&USRPFX.U JOB (DB2UNLOAD),'JOB', CLASS=A,MSGCLASS=A, //  TIME=1440 ,NOTIFY=&USRPFX //* DELETE DATASETS //STEP01  EXEC PGM=IEFBR14 //DD01  DD DISP=(MOD ,DELETE,DELETE), //  UNIT=SYSD A, //  SPACE=(TR K,(1,1)), // DSN=&USRPFX..DB2.P UNCH.&JOBNAME //DD02  DD DISP=(MOD ,DELETE,DELETE),`	

タスク	説明	必要なスキル

	`//  UNIT=SYSDA,`
	`//  SPACE=(TRK,(1,1)),`
	`// DSN=&USRPFX..DB2.UNLOAD.&JOBNAME`
	`//*`
	`//* RUNNING DB2 EXTRACTION BATCH JOB FOR AWS DEMO`
	`//*`
	`//UNLD01  EXEC PGM=DSNUTILB,REGION=0M,`
	`// PARM='<DSN>,UNLOAD'`
	`//STEPLIB  DD DISP=SHR,DSN=DSNC10.DBCG.SDSNEXIT`
	`//  DD  DISP=SHR,DSN=DSNC10.SDSNLOAD`
	`//SYSPRINT DD  SYSOUT=*`
	`//UTPRINT  DD  SYSOUT=*`
	`//SYSOUT  DD  SYSOUT=*`
	`//SYSPUN01 DD DISP=(NEW,CATLG,DELETE),`
	`//  SPACE=(CYL,(1,1),RLSE),`
	`// DSN=&USRPFX..DB2.PUNCH.&JOBNAME`
	`//SYSREC01 DD DISP=(NEW,CATLG,DELETE),`
	`//  SPACE=(CYL,(10,50),RLSE),`
	`// DSN=&USRPFX..DB2.UNLOAD.&JOBNAME`
	`//SYSPRINT DD SYSOUT=*`
	`//SYSIN  DD *`
	`  UNLOAD`
	`  DELIMITED COLDEL ','`
	`  FROM TABLE &TABNAME`	

タスク	説明	必要なスキル

	<pre> UNLDDN SYSREC01
	  PUNCHDDN SYSPUN01
	  SHRLEVEL CHANGE
	 ISOLATION UR;
	 /*
	//*
	//* FTP TO AMAZON S3
	 BACKED FTP SERVER IF
	 UNLOAD WAS SUCCESSFUL
	//*
	//SFTP EXEC PGM=BPXBA
	TCH,COND=(4,LE),RE
	GION=0M
	//STDPARM DD *
	 SH cp "//'&USRP
	FX..DB2.UNLOAD.&JO
	BNAME'"
	   &TABNAME..csv;
	 echo "ascii
	 " >> uplcmd;
	 echo "PUT &TABNAME.
	.csv " >>>> uplcmd;
	 sftp -b uplcmd -i .ssh/
	id_rsa &FTPUSER.
	@&FTPSITE;
	 rm &TABNAME..csv;
	 //SYSPRINT DD SYSOUT=*
	 //STDOUT DD SYSOUT=*
	 //STDENV DD *
	 //STDERR DD SYSOUT=*</pre>	

タスク	説明	必要なスキル
一括アンロード JCL を生成します。	このステップでは、JCL を使用して ISPF 環境で REXX スクリプトを実行します。TABLIST DD 名前に対して JCL を一括生成するための入力として、最初のステップで作成したスコープ内テーブルのリストを指定します。JCL は、ISPFILE DD 名前に対して指定されたユーザー指定の分割データセット内のテーブル名ごとに 1 つの新しい JCL を生成します。このライブラリは事前に割り当ててください。新しい JCL にはそれぞれ 2 つのステップがあります。1 つは Db2 テーブルをファイルにアンロードするステップ、もう 1 つはファイルを S3 バケットに送信するステップです。 JCL REXXEXEC で次の変更を行います (名前は変更できます)。 1. Job card user ID をテーブルのアンロード権限を持つメインフレームユーザ ID に置き換えてください。SYSPROC、ISPPLIB、I および <HLQ1> の値を代用するか、またはサイトの標準に合わせて DSN をカスタマイズしてください。イン	メインフレーム開発者

タスク	説明	必要なスキル
	ストール固有の値を確認するには、TSO ISRDDN コマンドを使用します。 2. <MFUSER> をインストール環境内のジョブ実行権限を持つユーザ ID に置き換えてください。 3. <FTPUSER> をインストール環境で USS 権限と FTP 権限を持つユーザー ID に置き換えてください。このユーザー ID と SSH セキュリティキーは、メインフレーム上の適切な UNIX Systems Services ディレクトリにあることを前提としています。 4. <AWS TransferFamily IP> を AWS Transfer Family の IP アドレスまたはドメイン名に置き換えてください。このアドレスは SFTP ステップに使用されます。 5. サイトの標準対応を適用し、下記の説明に従って REXX プログラムを更新してから JCL を送信してください。 JCL の一括生成ジョブ `//RUNREXX JOB (CREATEJCL),'RUNS ISPF TABLIST',`	

タスク	説明	必要なスキル

	CLASS=A,MSGCLASS=A,
	
	// TIME=1440
	,NOTIFY=&SYSUID
	//* Most of the values
	 required can be updated
	 to your site specific
	//* values using the
	 command 'TSO ISRDDN' in
	 your ISPF session.
	//* Update all the lines
	 tagged with //update
	 marker to desired
	//* site specific
	 values.
	//ISPF EXEC PGM=IKJEF
	T01,REGION=2048K,D
	YNAMNBR=25
	//SYSPROC DD
	 DISP=SHR,DSN=USER.
	Z23D.CLIST
	//SYSEXEC DD
	 DISP=SHR,DSN=<HLQ1
	>.TEST.REXXLIB
	//ISPPLIB DD
	 DISP=SHR,DSN=ISP.S
	ISPPENU
	//ISPSLIB DD
	 DISP=SHR,DSN=ISP.S
	ISPSENU
	// DD
	 DISP=SHR,DSN=<HLQ1
	>.TEST.ISPSLIB
	//ISPMLIB DD
	 DSN=ISP.SISPMENU,D
	ISP=SHR
	//ISPTLIB DD
	 DDNAME=ISPTABL
	// DD DSN=ISP.S
	ISPTENU,DISP=SHR	

タスク	説明	必要なスキル

	``
	//ISPTABL DD
	 LIKE=ISP.SISPTENU,
	UNIT=VIO
	//ISPPROF DD
	 LIKE=ISP.SISPTENU,
	UNIT=VIO
	//ISPLOG DD
	 SYSOUT=*,RECFM=VA,
	LRECL=125
	//SYSPRINT DD SYSOUT=*
	//SYSTSPRT DD
	 SYSOUT=*
	//SYSUDUMP DD
	 SYSOUT=*
	//SYSDBOUT DD
	 SYSOUT=*
	//SYSTSPRT DD
	 SYSOUT=*
	//SYSUDUMP DD
	 SYSOUT=*
	//SYSDBOUT DD
	 SYSOUT=*
	//SYSHELP DD
	 DSN=SYS1.HELP,DISP
	=SHR
	//SYSOUT DD SYSOUT=*
	//* Input list of
	 tablenames
	//TABLIST DD
	 DISP=SHR,DSN=<HLQ1
	>.DSN81210.TABLIST
	//* Output pds
	//ISPFILE DD
	 DISP=SHR,DSN=<HLQ1
	>.TEST.JOBGEN
	//SYSTSIN DD *
	ISPSTART CMD(ZSTEPS
	 <MFUSER> <FTPUSER> <AWS
	 TransferFamily IP>)
	/*
	``	

タスク	説明	必要なスキル
	REXX スクリプトを使用する前に、以下の点を変更します。 1. 前のステップで ZSTEPS をメンバー名として編集した JCL REXXEXEC の SYSEXEC スタックで定義されている PDS ライブラリに REXX スクリプトを保存します。名前を変更する場合は、JCL を必要に応じて更新する必要があります。 2. このスクリプトは trace オプションを使用して、エラーが発生した場合に備えて追加情報を出力します。代わりに EXECIO、ISPEXEC、および TSO ステートメントの後にエラー処理コードを追加し、トレースラインを削除できます。 3. このスクリプトは、最大 100,000 人のメンバーをサポートできる L0Dnnnnn 命名規則を使用してメンバー名を生成します。テーブルが 100,000 個を超える場合は、プレフィックスを短くし、tempjob ステートメント内の番号を調整してください。	

タスク	説明	必要なスキル
	STEPS REX スクリプト /*REXX - - -

タスク	説明	必要なスキル

	<pre>ADDRESS ISPEXEC
	"FTCLOSE NAME("tem
	pjob")"
	END
	
	ADDRESS TSO "FREE
	F(TABLIST) "
	ADDRESS TSO "FREE
	F(ISPFILE) "
	
	exit 0</pre>	

JCL を実行します。

タスク	説明	必要なスキル
Db2 アンロードステップを実行する。	JCL の生成後は、アンロードする必要のあるテーブルと同じ数の JCL が作成されます。 このストーリーでは、JCL で生成された例を使用して、構造と最も重要なステップについて説明します。 ユーザー操作は必要はありません。以下の情報は参考情報です。前のステップで生成した JCL を送信する場合は、スキップして [LoDnnnnn JCL の送信] タスクに進んでください。 IBM が提供する DSNUTILB Db2 ユーティリティで JCL を使用して Db2 データをアンロードする場合は、アンロー	メインフレーム開発者、システムエンジニア

タスク	説明	必要なスキル
	ドされたデータに圧縮された数値データが含まれていないことを確認する必要があります。これを行うには、DSNUTILB DELIMITED パラメーターを使用します。 DELIMITED パラメーターでは、テキストフィールドの区切り文字と二重引用符として文字を追加したり、VARCHAR 列のパディングを削除したり、DATE フィールドを含むすべての数値フィールドを EXTERNAL FORMAT に変換したりすることで、CSV 形式のデータをアンロードできます。 次の例は、コンマ文字を区切り文字として使用して、生成された JCL のアンロードステップがどのように表示されるかを示しています。 `UNLOAD DELIMITED COLDEL ',' FROM TABLE SCHEMA_NAME.TBNAME UNLDDN SYSREC01 PUNCHDDN SYSPUN01 SHRLEVEL CHANGE ISOLATION UR;`	

タスク	説明	必要なスキル
SFTP ステップを実行します。	JCL から SFTP プロトコルを使用するには、BPXBATCH ユーティリティを使用してください。 SFTP ユーティリティは MVS データセットに直接アクセスすることはできません。copy コマンド (cp) を使用して、シーケンシャルファイル `&USRPFX..DB2.UNLOAD.&JOBNAME` を USS ディレクトリにコピーし、それを `&TABNAME..csv` に変更できます。 プライベートキー (`id_rsa`) を使用し、RACF ユーザー ID をユーザー名として使用して `sftp` コマンドを実行し、AWS Transfer Family IP アドレスに接続します。 `SH cp "//'&USRPFX..DB2.UNLOAD.&JOBNAME'" &TABNAME..csv; echo "ascii " >> uplcmd; echo "PUT &TABNAME..csv " >>>> uplcmd; sftp -b uplcmd -i .ssh/id_rsa &FTPUSER.@&FTP_TF_SITE; rm &TABNAME..csv;`	メインフレーム開発者、システムエンジニア

タスク	説明	必要なスキル
LoDnnnnn JCL を送信してください。	以前の JCL では、アンロードして CSV に変換し、S3 バケットに転送する必要があるすべての LoDnnnnn JCL テーブルが生成されています。 生成されたすべての JCL で `submit` コマンドを実行します。	メインフレーム開発者、システムエンジニア

関連リソース

このドキュメントで使用されているさまざまなツールおよびソリューションの詳細については、次の内容を参照してください。

- 「z/OS OpenSSH ユーザーズガイド」
- 「Db2 z/OS — アンロード制御ステートメントのサンプル」
- 「Db2 z/OS — 区切り文字付きファイルのアンロード」
- 「Transfer Family — SFTP 対応サーバーの作成」
- 「Transfer Family — サービス管理対象ユーザーとの連携」

追加情報

Amazon S3 に Db2 データを保存した後は、さまざまな方法で新しい分析情報を得ることができます。Amazon S3 は AWS データ分析サービスと統合されているため、このデータを分散側で自由に使用または公開できます。例えば、次のオペレーションを実行できます。

- 「Amazon S3 上にデータレイク」を構築し、データを移動せずにその場でのクエリ、分析、機械学習ツールを使って貴重なインサイトを引き出します。
- AWS Transfer Family と統合されたアップロード後の処理ワークフローを設定して「Lambda 関数」を開始します。
- 「AWS Glue」を使用して、Amazon S3 または「フルマネージドデータベース」のデータにアクセスするための新しいマイクロサービスを開発します。AWS Glue は、分析、機械学習、アプリ

ケーション開発のためのデータの検出、準備、結合を容易にするサーバーレスデータ統合サービスです。

移行のユースケースでは、メインフレームから S3 にあらゆるデータを転送できるため、次のことが可能になります。

- Amazon S3 Glacier Deep Archive を使用して、物理インフラストラクチャを廃止し、費用対効果の高いデータアーカイブ戦略を構築します。
- Amazon S3 や S3 Glacier や Amazon Elastic File System (Amazon EFS) などの他の AWS サービスを使用して、スケーラブルで耐久性があり、安全なバックアップおよび復元ソリューションを構築し、既存のオンプレミス機能を強化または置き換えます。

その他のパターン

- Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする
- Precisely Connect を使用してメインフレームデータベースを AWS にレプリケート

管理

トピック

- コスト管理
- 高性能コンピューティング
- ハイブリッドクラウド
- 管理とガバナンス
- メッセージとコミュニケーション

コスト管理

トピック

- AWS Cost Explorer を使用して、AWS Glue ジョブの詳細なコストと使用状況レポートを作成します。
- AWS Cost Explorer を使用して Amazon EMR クラスターの詳細なコストと使用状況レポートを作成する
- その他のパターン

AWS Cost Explorer を使用して、AWS Glue ジョブの詳細なコストと使用状況レポートを作成します。

パリジャット・ビード（AWS）とアロマル・ラージ・ジャヤラジャン（AWS）によって作成された

概要

このパターンは、「ユーザー定義のコスト配分タグ」を設定して、AWS Glue データ統合ジョブの使用コストを追跡する方法を示しています。これらのタグを使用して、複数のディメンションにわたるジョブの詳細なコストと使用状況レポートを AWS Cost Explorer で作成できます。たとえば、チーム、プロジェクト、またはコストセンターレベルで使用コストを追跡できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ユーザー定義タグが有効になっている 1 つ以上の AWS Glue ジョブ

アーキテクチャ

ターゲットテクノロジースタック

- AWS Glue
- AWS Cost Explorer

次の図は、タグを適用して AWS Glue ジョブの使用コストを追跡する方法を示しています。

この図表は、次のワークフローを示しています：

1. データエンジニアまたは AWS 管理者が AWS Glue ジョブ用のユーザー定義のコスト配分タグを作成します。
2. AWS 管理者がタグを有効化します。
3. タグはメタデータを AWS Cost Explorer に報告します。

ツール

- AWS Glue は、フルマネージド型の抽出、変換、ロード (ETL) サービスです。これにより、データストアとデータストリーム間でデータを確実に分類、整理、強化、移動できます。
- AWS Cost Explorer を使用すると、コストと使用状況を表示および分析できます。

エピック

AWS Glue ジョブのタグの作成と有効化

タスク	説明	必要なスキル
AWS Glue ジョブ用のユーザー定義のコスト配分タグを作成します。	既存の AWS Glue ジョブにタグを追加するには 1. AWS マネジメントコンソールにサインインし、「AWS Glue コンソール」を開きます。 2. 左のナビゲーションペインの ETL で、[ジョブ] を選択します。 3. 「お客様のジョブ」セクションで、タグ付けするジョブの名前を選択します。 4. [Job details] (ジョブの詳細) タブを選択します。次に、「詳細プロパティ」セクションを展開します。 5. 「タグ」で「新しいタグを追加」を選択します。 6. キーには、タグの名前を入力します。	データエンジニア

タスク	説明	必要なスキル
	7. (オプション) 「値」には、キーに関連付けたい値を入力します。 8. (オプション) ジョブ用に作成するタグごとに手順 5 から 7 を繰り返します。 9. [Save] を選択します。 新しい AWS Glue ジョブにタグを追加するには 1. ユースケースの要件に基づいて新しい AWS Glue ジョブを作成します。手順については、「AWS Glue 開発者ガイド」の「AWS Glue コンソールでジョブを操作」を参照してください。 2. 「Job の詳細」設定を行うときは、このタスクの「既存の AWS Glue ジョブにタグを追加するには」セクションのステップ 4 ~ 9 に従います。 **Note** 詳細については、AWS Glue デベロッパーガイドの「AWS Glue の AWS タグ」を参照してください。	

タスク	説明	必要なスキル
ステップ 1： ユーザー定義のコスト配分タグを有効にする	「AWS 請求ユーザーガイド」の「ユーザー定義のコスト配分タグの有効化」の手順に従ってください。	AWS 管理者

AWS Glue ジョブのコストと使用状況のレポートを作成する

タスク	説明	必要なスキル
AWS Cost Explorer のタグフィルタを使用して、AWS Glue ジョブのコストと使用状況のレポートを作成します。	1. AWS マネジメントコンソールにサインインし、「AWS コストマネジメントコンソール」を開きます。 2. 左のナビゲーションペインの [レポート] を選択します。 3. [Create New Report (新しいレポートを作成)] を選択します。 4. 「レポートタイプを選択」で「コストと使用量 (推奨)」を選択します。次に、[レポートを生成] を選択します。 5. 「フィルター」には「サービス」を選択します。「サービス」ドロップダウンが表示されます。 6. [Glue] のチェックボックスをオンにします。次に、「フィルターを適用」を選択します。	一般的な AWS、AWS 管理者

タスク	説明	必要なスキル
	7. 「フィルター」で「タグ」を選択します。「タグ」ドロップダウンが表示されます。 8. 「チーム」を選択します。次に、タグを割り当てたチームの横にあるチェックボックスを選択します。タグを割り当てていないチームはすべて除外します。次に、「フィルターを適用」を選択します。 9. グラフの上部にある「Tag」を選択します。次に、レポートを作成する AWS Glue ジョブのタグを選択します。 10.グラフの上部にある「過去 3 か月」ドロップダウンを選択し、レポートの対象とする期間を選択します。次に、「月次」ドロップダウンを選択し、レポート内の明細項目を期間に基づいてどのように集計するかを選択します。 11[名前を付けて保存] を選択します。次に、レポートのタイトルを入力します。 12[レポートの保存] を選択します。	

タスク	説明	必要なスキル
	詳細については、「AWS コスト管理ユーザーガイド」の「Cost Explorer を使用してデータを探索する」を参照してください。	

AWS Cost Explorer を使用して Amazon EMR クラスターの詳細なコストと使用状況レポートを作成する

パリジャット・ビード (AWS) とアロマル・ラージ・ジャヤラジャン (AWS) によって作成された

概要

このパターンは、「ユーザー定義のコスト配分タグ」を設定して Amazon EMR クラスターの使用コストを追跡する方法を示しています。これらのタグを使用して、複数のディメンションにわたるクラスターの詳細なコストと使用状況レポートを AWS Cost Explorer で作成できます。たとえば、チーム、プロジェクト、またはコストセンターレベルで使用コストを追跡できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ユーザー定義タグが有効になっている 1 つ以上の「EMR クラスター」

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EMR
- AWS Cost Explorer

ターゲットアーキテクチャ

次の図は、タグを適用して特定の Amazon EMR クラスターの使用コストを追跡する方法を示しています。

この図表は、次のワークフローを示しています：

1. データエンジニアまたは AWS 管理者が Amazon EMR クラスター用のユーザー定義のコスト配分タグを作成します。
2. AWS 管理者がタグを有効化します。
3. タグはメタデータを AWS Cost Explorer に報告します。

ツール

ツール

- 「Amazon EMR」は、AWS でビッグデータフレームワークの実行を簡素化して、ビッグデータを処理および分析するマネージドクラスタープラットフォームです。
- 「AWS Cost Explorer」を使用すると、AWS コストと使用状況を表示および分析できます。

エピック

Amazon EMR クラスターのタグの作成と有効化

タスク	説明	必要なスキル
Amazon EMR クラスター用のユーザー定義のコスト配分タグを作成します。	既存の Amazon EMR クラスターにタグを追加するには [Amazon EMR 管理ガイド] の「既存のクラスターへのタグの追加」の指示に従います。 新しい Amazon EMR クラスターにタグを追加するには [Amazon EMR 管理ガイド] の「新しいクラスターにタグの追加」の指示に従います。 Amazon EMR クラスターの設定方法の詳細については、[Amazon EMR 管理ガイド] の「クラスターの計画と設定」を参照してください。	データエンジニア
ステップ 1： ユーザー定義のコスト配分タグを有効にする	「AWS 請求ユーザーガイド」の「ユーザー定義のコスト配分タグの有効化」の手順に従ってください。	AWS 管理者

Amazon EMR クラスターのコストと使用状況レポートの作成

タスク	説明	必要なスキル
AWS Cost Explorer のタグフィルタを使用して、Amazon EMR クラスターのコストと使用状況のレポートを作成します。	1. AWS マネジメントコンソールにサインインし、「AWS コストマネジメントコンソール」を開きます。 2. 左のナビゲーションペインの [レポート] を選択します。 3. [Create New Report (新しいレポートを作成)] を選択します。 4. 「レポートタイプを選択」で「コストと使用量 (推奨)」を選択します。次に、[レポートを生成] を選択します。 5. 「フィルター」には「サービス」を選択します。[サービス] ドロップダウンが表示されます。 6. EMR (Elastic MapReduce) と EC2 インスタンス (Elastic コンピュートクラウド — コンピュート) の横にあるチェックボックスを選択します。次に、[フィルターを適用] を選択します。 7. 「フィルター」で「タグ」を選択します。「タグ」ド	一般的な AWS、AWS 管理者

タスク	説明	必要なスキル
	ロップダウンが表示されます。 8. 「チーム」を選択します。次に、タグを割り当てたチームの横にあるチェックボックスを選択します。タグを割り当てていないチームはすべて除外します。次に、「フィルターを適用」を選択します。 9. グラフの上部にある [タグ] を選択します。次に、レポートを作成する Amazon EMR クラスターのタグを選択します。 10.グラフの上部にある [過去 3 か月] ドロップダウンを選択し、レポートの対象とする期間を選択します。次に、「月次」ドロップダウンを選択し、レポート内の明細項目を期間に基づいてどのように集計するかを選択します。 11[名前を付けて保存] を選択します。次に、レポートのタイトルを入力します。 12[レポートの保存] を選択します。 詳細については、「AWS コスト管理ユーザーガイド」の「Cost Explorer を使用して	

タスク	説明	必要なスキル
	データを探索する」を参照してください。	

その他のパターン

- Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する
- 複数のアカウントとリージョンの AWS リソースを自動的にインベントリする
- AWS CloudFormation を使用して AppStream 2.0 リソースの作成を自動化する
- DynamoDB TTL を使用して項目を Amazon S3 に自動的にアーカイブする
- AWS Systems Manager メンテナンスウィンドウを使用して Amazon RDS DB インスタンスを自動的に停止および起動する
- Amazon RDS と Amazon Aurora の詳細なコストと使用状況レポートを作成する
- AWS Config と を使用して未使用の Amazon EBS ボリュームを削除する AWS Systems Manager
- Amazon DynamoDB テーブルのストレージコストを推定
- DynamoDB テーブルのコストをオンデマンドで見積る
- Amazon EKS Pod Identity と KEDA を使用して Amazon EKS でイベント駆動型自動スケーリングを設定する
- AWS Fargate WaitCondition フックコンストラクトを使用してリソースの依存関係とタスク実行を調整する

高性能コンピューティング

トピック

- Terraform と DRA を使用して高性能データ処理用の Lustre ファイルシステムをデプロイする
- AWS ParallelCluster 用の Grafana モニタリングダッシュボードを設定する
- NICE EnginFrame と NICE DCV セッションマネージャーを使用して自動スケーリング仮想デスクトップインフラストラクチャを設定する

Terraform と DRA を使用して高性能データ処理用の Lustre ファイルシステムをデプロイする

作成者: Arun Bagal (AWS) と Ishwar Chauthaiwale (AWS)

概要

このパターンでは、Lustre ファイルシステムを に自動的にデプロイ AWS し、Amazon Elastic Compute Cloud (Amazon EC2) および Amazon Simple Storage Service (Amazon S3) と統合します。

このソリューションは、統合されたストレージ、コンピューティングリソース、Amazon S3 データアクセスを備えたハイパフォーマンスコンピューティング (HPC) 環境を迅速にセットアップするのに役立ちます。Lustre のストレージ機能と Amazon EC2 が提供する柔軟なコンピューティングオプション、および Amazon S3 のスケーラブルなオブジェクトストレージを組み合わせることで、機械学習、HPC、ビッグデータ分析でデータ集約型のワークロードに取り組むことができます。

このパターンでは、HashiCorp Terraform モジュールと Amazon FSx for Lustre を使用して、次のプロセスを合理化します。

- Lustre ファイルシステムのプロビジョニング
- FSx for Lustre と S3 バケット間のデータリポジトリ関連付け (DRA) を確立して、Lustre ファイルシステムを Amazon S3 オブジェクトにリンクする
- EC2 インスタンスの作成
- Amazon S3 にリンクされた DRA を使用して Lustre ファイルシステムを EC2 インスタンスにマウントする

このソリューションの利点は次のとおりです。

- モジュラー設計。このソリューションの個々のコンポーネントを簡単に維持および更新できます。
- スケーラビリティ。整合性のある環境を AWS アカウント または リージョンにすばやくデプロイできます。
- 柔軟性。特定のニーズに合わせてデプロイをカスタマイズできます。
- ベストプラクティス。このパターンでは、 AWS ベストプラクティスに従う事前設定されたモジュールを使用します。

Lustre ファイルシステムの詳細については、Lustre ウェブサイトを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 最小特権 AWS Identity and Access Management (IAM) ポリシー (手順を参照)

制約事項

FSx for Lustre は、Lustre ファイルシステムを単一のアベイラビリティーゾーンに制限します。これは、高可用性要件がある場合に懸念される可能性があります。ファイルシステムを含むアベイラビリティーゾーンに障害が発生した場合、復旧するまでファイルシステムへのアクセスは失われます。高可用性を実現するには、DRA を使用して Lustre ファイルシステムを Amazon S3 にリンクし、アベイラビリティーゾーン間でデータを転送できます。

製品バージョン

- Terraform バージョン 1.9.3 以降
- HashiCorp AWS プロバイダーバージョン 4.0.0 以降

アーキテクチャ

次の図は、FSx for Lustre のアーキテクチャと、 AWS のサービス の補完的なアーキテクチャを示しています AWS クラウド。

このアーキテクチャには、以下の項目が含まれます。

- S3 バケットは、データの耐久性、スケーラビリティ、コスト効率に優れたストレージ場所として使用されます。FSx for Lustre と Amazon S3 の統合により、Amazon S3 とシームレスにリンクされた高性能ファイルシステムが提供されます。
- FSx for Lustre は Lustre ファイルシステムを実行および管理します。
- Amazon CloudWatch Logs は、ファイルシステムからログデータを収集してモニタリングします。これらのログは、Lustre ファイルシステムのパフォーマンス、ヘルス、アクティビティに関するインサイトを提供します。
- Amazon EC2 は、オープンソースの Lustre クライアントを使用して Lustre ファイルシステムにアクセスするために使用されます。EC2 インスタンスは、同じ仮想プライベートクラウド (VPC) 内

の他のアベイラビリティーゾーンからファイルシステムにアクセスできます。ネットワーク設定により、VPC 内のサブネット間でアクセスできるようになります。Lustre ファイルシステムがインスタンスにマウントされたら、ローカルファイルシステムを使用するのと同じように、そのファイルとディレクトリを操作できます。

- AWS Key Management Service （ AWS KMS) は、保管中のデータの暗号化を提供することで、ファイルシステムのセキュリティを強化します。

自動化とスケール

Terraform を使用すると、複数の環境にまたがる Lustre ファイルシステムのデプロイ、管理、スケーリングが容易になります。FSx for Lustre では、1 つのファイルシステムにサイズ制限があるため、複数のファイルシステムを作成して水平方向にスケーリングする必要がある場合があります。Terraform を使用して、ワークロードのニーズに基づいて複数の Lustre ファイルシステムをプロビジョニングできます。

ツール

AWS のサービス

- Amazon CloudWatch Logs は、すべてのシステム、アプリケーション、および からのログを一元化 AWS のサービス し、ログをモニタリングして安全にアーカイブするのに役立ちます。
- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon FSx for Lustre を使用すると、高性能な Lustre ファイルシステムの起動、実行、スケーリングを簡単かつ費用対効果の高い方法で行うことができます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、データ量にかかわらず、保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このパターンのコードは、Terraform リポジトリを使用した GitHub Provision FSx for Lustre Filesystem で使用できます。 FSx

ベストプラクティス

- 次の変数は、Lustre ファイルシステムを定義します。「エピック」セクションの指示に従って、環境に基づいてこれらを正しく設定してください。
  - `storage_capacity` – Lustre ファイルシステムのストレージ容量、GiBs。最小およびデフォルト設定は 1200 GiB です。
  - `deployment_type` – Lustre ファイルシステムのデプロイタイプ。2 つのオプションPERSISTENT_1と PERSISTENT_2 (デフォルト) の説明については、FSx for Lustre のドキュメントを参照してください。
  - `per_unit_storage_throughput` – 読み取りおよび書き込みスループット。1 TiB あたり MBs秒。
  - `subnet_id` – FSx for Lustre をデプロイするプライベートサブネットの ID。
  - `vpc_id` – FSx for Lustre をデプロイする AWS 仮想プライベートクラウドの ID。
  - `data_repository_path` – Lustre ファイルシステムにリンクされる S3 バケットへのパス。
  - `iam_instance_profile` – EC2 インスタンスの起動に使用する IAM インスタンスプロファイル。
  - `kms_key_id` – データ暗号化に使用される AWS KMS キーの Amazon リソースネーム (ARN)。
- 変数と `vpc_id`変数を使用して、VPC 内の適切なネットワークアクセス`security_group`と配置を確保します。
- 「エピック」セクションの説明に従って `terraform plan` コマンドを実行し、変更を適用する前にプレビューおよび検証します。これにより、潜在的な問題を把握し、デプロイされる内容を確実に把握できます。
- 「エピック」セクションで説明されているように `terraform validate` コマンドを使用して構文エラーをチェックし、設定が正しいことを確認します。

エピック

環境をセットアップします。

タスク	説明	必要なスキル
Terraform をインストールします。	ローカルマシンに Terraform をインストールするに	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	は、Terraform ドキュメントの指示に従います。	
AWS 認証情報を設定します。	アカウントの AWS Command Line Interface （ AWS CLI) プロファイルを設定するには、AWS ドキュメントの指示に従います。	AWS DevOps、DevOps エンジニア
GitHub リポジトリのクローンを作成します。	GitHub リポジトリのクローンを作成するには、 コマンドを実行します。 `git clone https://github.com/aws-samples/provision-fsx-lustre-with-terraform.git`	AWS DevOps、DevOps エンジニア

FSx for Lustre の設定とデプロイ

タスク	説明	必要なスキル
デプロイ設定を更新します。	1. ローカルマシンのクローンされたリポジトリで、`fsx_deployment` ディレクトリに移動します。 `cd fsx_deployment` 2. `terraform.tfvars` ファイルを開き、次の変数の値を更新します。 • `vpc_id` • `subnet_id`	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル
	• data_repository_path • iam_instance_profile • kms_key_id これらの変数の詳細については、「ベストプラクティス」セクションを参照してください。 3. 同じディレクトリで、locals.tf ファイルを開き、fsx_inress および fsx_egress セキュリティグループ変数の CIDR 範囲を更新します。 4. 必要に応じて、variables.tf ファイルを開き、これらの変数のデフォルト値を更新します。 • storage_capacity • deployment_type • per_unit_storage_throughput これらの変数の詳細については、「ベストプラクティス」セクションを参照してください。	

タスク	説明	必要なスキル
Terraform 環境を初期化します。	環境を初期化して Terraform `fsx_deployment` モジュールを実行するには、以下を実行します。 `terraform init`	AWS DevOps、DevOps エンジニア
Terraform 構文を検証します。	構文エラーをチェックし、設定が正しいことを確認するには、以下を実行します。 `terraform validate`	AWS DevOps、DevOps エンジニア
Terraform 設定を検証します。	Terraform 実行プランを作成し、デプロイをプレビューするには、以下を実行します。 `terraform plan -var-file terraform.tfvars`	AWS DevOps、DevOps エンジニア
Terraform モジュールをデプロイします。	FSx for Lustre リソースをデプロイするには、以下を実行します。 `terraform apply -var-file terraform.tfvars`	AWS DevOps、DevOps エンジニア

AWS リソースをクリーンアップする

タスク	説明	必要なスキル
AWS リソースを削除します。	FSx for Lustre 環境の使用が終了したら、Terraform によってデプロイされた AWS リソー	AWS DevOps、DevOps エンジニア

タスク	説明	必要なスキル

	スを削除して、不要な料金が発生しないようにできます。コードリポジトリで提供される Terraform モジュールは、このクリーンアップを自動化します。
	
	1. ローカルリポジトリで、`fsx_deployment` ディレクトリに移動します。
	
	`cd fsx_deployment`
	
	2. コマンドを実行します。
	
	`terraform destroy -var-file terraform.tfvars`	

トラブルシューティング

問題	ソリューション
FSx for Lustre はエラーを返します。	FSx for Lustre の問題に関するヘルプについては、FSx for Lustre ドキュメントの「Amazon FSx for Lustre のトラブルシューティング」を参照してください。 FSx

関連リソース

- Terraform を使用した Amazon FSx for Lustre の構築 (Terraform ドキュメントのAWS プロバイダーリファレンス)
- Amazon FSx for Lustre の開始方法 (FSx for Lustre ドキュメント)
- AWS Amazon FSx for Lustre に関するブログ投稿

AWS ParallelCluster 用の Grafana モニタリングダッシュボードを設定する

作成者: Dario La Porta (AWS)、William Lu (AWS)

概要

AWS ParallelCluster は、ハイパフォーマンスコンピューティング(HPC)クラスターのデプロイと管理をサポートします。AWS Batch と Slurm のオープンソースジョブスケジューラーをサポートしています。AWS ParallelCluster はログ記録とメトリクス用に Amazon CloudWatch と統合されていますが、ワークロードのモニタリングダッシュボードは提供されていません。

AWS ParallelCluster (GitHub) の Grafana ダッシュボードは AWS ParallelCluster のモニタリングダッシュボードです。ジョブスケジューラーの分析情報とオペレーティングシステム (OS) レベルでの詳細なモニタリングメトリクスを提供します。このソリューションに含まれるダッシュボードの詳細については、GitHub リポジトリの「ダッシュボードの例」を参照してください。これらのメトリクスは、HPC ワークロードとそのパフォーマンスを詳しく理解するために役立ちます。ただし、ダッシュボードコードは最新バージョンの AWS ParallelCluster やソリューションで使用されているオープンソースパッケージに対しては更新されません。このパターンにより、ソリューションが強化され、以下の利点が得られます。

- AWS ParallelCluster v3 をサポート
- Prometheus、Grafana、Prometheus Slurm Exporter、NVIDIA DCGM-Exporter など、最新バージョンのオープンソースパッケージを使用しています。
- Slurm ジョブが使用する CPU コアと GPU の数を増やします。
- ジョブモニタリングダッシュボードを追加する
- 4 つまたは 8 つのグラフィックプロセッシングユニット (GPU) を搭載したノードの GPU ノードモニタリングダッシュボードを強化します。

このバージョンの拡張ソリューションは、AWS のお客様の HPC 実稼働環境で実装および検証されています。

前提条件と制限

前提条件

- AWS ParallelCluster CLI がインストールして設定済み。
- AWS ParallelCluster でサポートされているネットワーク設定。このモードでは、AWS ParallelCluster を使用した 2 つのサブネット 設定を使用します。これには、1 つのパブリック サ

ブネット、プライベート サブネット、インターネットゲートウェイ、および NAT ゲートウェイが必要です。

- すべての AWS ParallelCluster クラスターノードがインターネットにアクセスできる必要があります。これは、インストールスクリプトがオープンソースソフトウェアと Docker イメージをダウンロードできるようにするためです。
- Amazon Elastic Compute Cloud (Amazon EC2) のキーペア このキーペアを持つリソースは、ヘッドノードへの Secure Shell (SSH) アクセス権があります。

機能制限

- このパターンは Ubuntu 20.04 LTS をサポートするように設計されています。別のバージョンの Ubuntu を使用している場合、または Amazon Linux や CentOS を使用している場合は、このソリューションで提供されているスクリプトを変更する必要があります。　これらの変更は、このパターンには含まれていません。

製品バージョン

- Ubuntu 20.04 LTS
- ParallelCluster 3.X

請求とコストに関する考慮事項

- このパターンでデプロイされるソリューションは無料利用枠の対象外です。Amazon EC2、Amazon FSx for Lustre、Amazon VPC の NAT ゲートウェイ、Amazon Route 53 には料金がかかります。

アーキテクチャ

ターゲット アーキテクチャ

以下の図では、ユーザーがヘッドノードで AWS ParallelCluster のモニタリングダッシュボードにアクセスする方法を示しています。ヘッドノードは NICE DCV、Prometheus、Grafana、Prometheus Slurm Exporter、Prometheus Node Exporter、NGINX Open Source を実行します。　コンピュートノードは Prometheus Node Exporter を実行します。ノードに GPU が含まれている場合は NVIDIA DCGM-Exporter も実行します。ヘッドノードはコンピュートノードから情報を取得し、そのデータを Grafana ダッシュボードに表示します。

ほとんどの場合、ジョブスケジューラは大量の CPU やメモリを必要としないので、ヘッドノードの負荷は大きくありません。ユーザーはポート 443 から SSL を使用してヘッドノードのダッシュボードにアクセスします。

権限のある閲覧者はすべて、モニタリングダッシュボードを匿名で閲覧できます。ダッシュボードを変更できるのは Grafana 管理者のみです。 `aws-parallelcluster-monitoring/docker-compose/docker-compose.head.yml` ファイルで Grafana 管理者のパスワードを設定します。

ツール

AWS サービス

- NICE DCV は、さまざまなネットワーク条件下で、任意のクラウドまたはデータセンターから任意のデバイスに、リモートデスクトップやアプリケーションストリーミングを配信するのに役立つ高性能リモート表示プロトコルです。
- AWS ParallelCluster は、ハイパフォーマンスコンピューティング(HPC)クラスターのデプロイと管理をサポートします。AWS Batch と Slurm のオープンソースジョブスケジューラーをサポートしています。
- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。

その他のツール

- Docker は、オペレーティングシステムレベルの仮想化を使用してソフトウェアをコンテナで配信するサービスとしての Platform as a Service (PaaS) 製品のセットです。
- Grafana は、メトリクス、ログ、トレースのクエリ、可視化、アラート表示、探索に役立つオープンソースソフトウェアです。
- NGINX Open Source は、オープンソースのウェブサーバーで、リバースプロキシでもあります。
- NVIDIA データセンター GPU マネージャー (DCGM) は、クラスター環境で NVIDIA データセンターのグラフィックプロセッシングユニット (GPU) を管理およびモニタリングするための一連のツールです。このパターンでは、Prometheus から GPU メトリクスをエクスポートするのに役立つ DCGM-Exporter を使用します。

- Prometheus はオープンソースのシステム監視ツールキットで、ラベルと呼ばれる関連するキーと値のペアを含む、時系列データとしてメトリクスを収集して保存します。このパターンでは、Prometheus Slurm Exporter を使用してメトリクスを収集およびエクスポートし、Prometheus Node Exporter を使用してコンピュートノードからメトリクスをエクスポートします。
- Ubuntu はオープンソースの Linux ベースのオペレーティングシステムで、エンタープライズサーバー、デスクトップ、クラウド環境、IoT 向けに設計されています。

コードリポジトリ

このパターンのコードは、GitHub 内の「pcluster-monitoring-dashboard」リポジトリで利用できます。

エピック

必要なリソースを作成する

タスク	説明	必要なスキル
S3 バケットを作成する。	Amazon S3 バケットを作成する。このバケットを使用して設定スクリプトを保存します。手順については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	AWS 全般
リポジトリをクローン作成します。	以下のコマンドを実行して GitHub pcluster-monitoring-dashboard リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/parallelcluster-monitoring-dashboard.git`	DevOps エンジニア

タスク	説明	必要なスキル
管理者のパスワードを作成します。	1. `aws-parallelcluster-monitoring` フォルダーを選択し、`docker-compose` フォルダーを選択してから、docker-compose.head.yml ファイルを開きます。 2. `GF_SECURITY_ADMIN_PASSWORD` 変数で、`Grafana4PC!` を任意のパスワードに置き換えます。これは Grafana アカウントを管理するための管理パスワードです。 3. docker-compose.head.yml ファイルを保存して閉じます。	Linux シェルスクリプト
必要なファイルを S3 バケットにコピーします。	post_install.sh スクリプトと aws-parallelcluster-monitoring フォルダーを、作成した S3 バケットにコピーします。手順については、Amazon S3 ドキュメントの「オブジェクトのアップロード」を参照してください。	AWS 全般

タスク	説明	必要なスキル
ヘッドノードに追加のセキュリティグループを設定します。	1. ヘッドノードのセキュリティグループを作成します。このセキュリティグループは、ヘッドノードのモニタリングダッシュボードへのインバウンドトラフィックを許可します。手順については、Amazon VPC ユーザーガイドの「セキュリティグループの作成」を参照してください。 2. インバウンドルールをセキュリティグループに追加します。手順については、Amazon VPC ドキュメントの「セキュリティグループにルールを追加する」を参照してください。ルールには次のパラメータを使用します。 • タイプ – HTTPS • プロトコル – TCP • ポート範囲 – 443 • ソース — IP アドレスを入力します。 • 説明 — ユーザーがモニタリングダッシュボードにアクセスできるようにします。	AWS 管理者

タスク	説明	必要なスキル
ヘッドノードの IAM ポリシーを設定します。	ヘッドノードの ID ベースのポリシーを作成します。このポリシーにより、ノードは Amazon CloudWatch からメトリクスデータを取得できます。 GitHub リポジトリには、サンプル ポリシーが含まれています。手順については、AWS Identity and Access Management (IAM) ドキュメントの「IAM ポリシーの作成」を参照してください。	AWS 管理者
コンピューティングノードの IAM ポリシーを設定します。	コンピューティングノードの ID ベースのポリシーを作成します。このポリシーを使用すると、ノードはジョブ ID とジョブ所有者を含むタグを作成できます。GitHub リポジトリには、サンプル ポリシーが含まれています。詳細については、IAM ドキュメントの「IAM ポリシーの作成」を参照してください。 提供されているサンプルファイルを使用する場合は、次の値を置き換えます： • <REGION> – クラスターがホストされている AWS リージョン • <ACCOUNT_ID> – AWS アカウント ID	AWS 管理者

クラスターを作成する

タスク	説明	必要なスキル
提供されたクラスターテンプレートファイルを変更します。	AWS ParallelCluster を作成します。提供されている cluster.yaml AWS CloudFormation テンプレートファイルを開始点として使用して、クラスターを作成します。提供されたテンプレートの次の値を置換します： • <REGION> – クラスターがホストされている AWS リージョン。 • <HEADNODE_SUBNET> — VPC のパブリックサブネット。 • <ADDITIONAL_HEAD_NODE_SG> — ヘッドノード用に作成したセキュリティグループの名前。 • <KEY_NAME> — 既存の Amazon EC2 キーペアの名前を入力します。このキーペアを持つリソースは、ヘッドノードへの Secure Shell (SSH) アクセス権があります。 • <ALLOWED_IPS> —— ヘッドノードへの SSH 接続を許可する CIDR 形式の IP アドレス範囲を入力します。	AWS 管理者

タスク	説明	必要なスキル
	• <ADDITIONAL_HEAD_NODE_POLICY> — ヘッドノード用に作成した IAM ポリシーの名前を入力します。 • <BUCKET_NAME> – 作成した S3 バケットの名前を入力します。 • <COMPUTE_SUBNET>— VPC 内のプライベートサブネットの名前を入力します。 • <ADDITIONAL_COMPUTE_NODE_POLICY> — コンピューティングノード用に作成した IAM ポリシーの名前を入力します。	
クラスターを作成します。	AWS ParallelCluster CLI で、以下のコマンドを入力します。これで、CloudFormation テンプレートがデプロイされ、クラスターが作成されます。このコマンドの詳細については、AWS ParallelCluster ドキュメントの pcluster create-cluster を参照してください。 `pcluster create-cluster -n <cluster_name> -c cluster.yaml`	AWS 管理者

タスク	説明	必要なスキル
クラスターの作成をモニタリングします。	以下のコマンドを入力して、クラスターの作成を監視します。このコマンドの詳細については、AWS ParallelCluster ドキュメントの pcluster describe-cluster を参照してください。 `pcluster describe-cluster -n <cluster_name>`	AWS 管理者

Grafana ダッシュボードを使用する

タスク	説明	必要なスキル
Grafana ポータルにアクセスします。	1. 以下のコマンドを入力して、ヘッドノードのパブリック IP アドレスを取得します。 `pcluster describe-cluster -n <cluster_name> --query headNode.publicIpAddress` 2. ウェブブラウザで、次の URL に移動して Grafana ダッシュボードにアクセスします。 https://<head_node_public_ip_address>	AWS 管理者

タスク	説明	必要なスキル
	3. Grafana のフロントページで、左側のメニューにある 4 つの四角いダッシュボードアイコンを選択し、[全般] を選択します。そうすると、設定済みダッシュボードのリストが表示されます。Grafana では、以下のダッシュボードを使用できます。 • クラスターコスト — クラスターのコストに関する情報が含まれます。 • クラスターログ — クラスターのログに関する情報が含まれます。 • コンピューティングノードの詳細 — コンピューティングノードの使用統計に関する情報が含まれます。 • コンピューティングノードのリスト — クラスターのコンピューティングノードのリストが含まれます。 • GPU ノード — GPU ノードの使用統計に関する情報が含まれます。 • ジョブの詳細 — ジョブのリソース使用率に関する情報が含まれます。	

タスク	説明	必要なスキル
	• ヘッドノードの詳細 — ヘッドノードの使用統計に関する情報が含まれます。 • ParallelCluster の概要 — クラスターの使用状況に関する情報が含まれます	

ソリューションをクリーンアップして、関連コストの発生を防ぎましょう。

タスク	説明	必要なスキル
クラスターを削除します。	クラスターを削除するには、次のコマンドを入力します。このコマンドの詳細については、AWS ParallelCluster ドキュメントの pcluster delete-cluster を参照してください。 `pcluster delete-cluster -n <cluster_name>`	AWS 管理者
IAM ポリシーを削除します。	ヘッドノードとコンピューティングノード用に作成したポリシーを削除します。ポリシーの削除の詳細については、IAM ドキュメントの「IAM ポリシーの削除」を参照してください。	AWS 管理者
セキュリティグループとルールを削除するには	ヘッドノード用に作成したセキュリティグループを削除します。　詳細については、Amazon VPC ドキュメントの	AWS 管理者

タスク	説明	必要なスキル
	「セキュリティグループのルール」と「セキュリティグループの削除」を参照してください。	
S3 バケットを削除します。	設定スクリプトを保存するために作成した S3 バケットを削除します。 詳細については、Amazon S3 ドキュメントの「バケットの削除」を参照してください。	AWS 全般

トラブルシューティング

問題	ソリューション
ブラウザからヘッドノードにアクセスできません。	セキュリティグループをチェックし、インバウンドポート 443 がオープンになっていることを確認します。
Grafana が開かない。	ヘッドノードで、`docker logs Grafana` のコンテナログを確認してください。
一部のメトリクスにデータがありません。	ヘッドノードで、すべてのコンテナのコンテナログを確認します。

関連リソース

AWS ドキュメント

- 「Amazon EC2 の IAM ポリシー」

その他の AWS リソース

- AWS ParallelCluster

- 「AWS ParallelCluster のモニタリングダッシュボード」 (AWS ブログ記事)

その他のリソース

- Prometheus 監視システム
- Grafana

NICE EnginFrame と NICE DCV セッションマネージャーを使用して自動スケーリング仮想デスクトップインフラストラクチャを設定する

作成者: Dario La Porta (AWS) と Salvatore Maccarone (AWS)

概要

NICE DCV は、高性能なリモートディスプレイプロトコルで、さまざまなネットワーク条件の中で、あらゆるデバイスにリモートデスクトップやアプリケーションを安全にストリームする方法を提供します。NICE DCV と Amazon Elastic Compute Cloud (Amazon EC2) を使用すると、グラフィックを多用するアプリケーションを Amazon EC2 インスタンスでリモートで実行し、ユーザーインターフェイスをよりシンプルでリモートのクライアントマシンにストリーミングできます。これにより、高価な専用ワークステーションが不要になり、クラウドとクライアントマシン間で大量のデータを転送する必要がなくなります。

このパターンでは、ウェブベースのユーザーインターフェイスからアクセス可能な、フル機能で自動スケール可能なLinuxおよびWindows仮想デスクトップインフラストラクチャ（VDIをセットアップします。VDIソリューションにより、研究開発（R&D）ユーザーは、グラフィックスを多用する分析リクエストを送信したり、結果をリモートで確認したりするための、アクセスしやすくパフォーマンスの高いユーザーインターフェースを利用できます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 管理者権限と一連のアクセスキー。
- AWS Cloud Development Kit (AWS CDK) Toolkit、インストールおよび設定済み。詳細については、「AWS CDKをインストールする」を参照してください。
- AWS Command Line Interface （ AWS CLI ）、 用にインストールおよび設定されています AWS アカウント。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。
- Python、インストールおよび設定。詳細については、「ソースリリース」 (Python ウェブサイト) を参照してください。
- 1 つ以上の 仮想プライベートクラウド (VPC) が利用可能です。
- 2 つ以上の Elastic IP アドレスを使用できます。デフォルトの制限について詳しくは、「Elastic IP アドレスの制限」 を参照してください。

- Linux Amazon EC2 インスタンスの場合は、Secure Shell (SSH) キーペアを設定します。詳細については、「キーペアおよび Linux インスタンス」 を参照してください。

製品バージョン

- AWS CDK バージョン 2.26.0 以降
- Python バージョン 3.8 以降。

アーキテクチャ

ターゲットアーキテクチャ

次の図は、この VDI ソリューションのさまざまなコンポーネントを示しています。ユーザーは NICE EnginFrame と対話して、Windows と Linux の NICE DCV インスタンスの Amazon EC2 Auto Scaling グループに従って Amazon EC2 インスタンスを起動します。

自動化とスケール

このパターンに含まれるコードは、カスタム VPC、パブリックサブネットとプライベートサブネット、インターネットゲートウェイ、NAT ゲートウェイ、Application Load Balancer、セキュリティグループ、および AWS Identity and Access Management (IAM) ポリシーを作成します。 AWS CloudFormation は、Linux および Windows NICE DCV サーバーのフリートの作成にも使用されます。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。
- NICE DCV は、さまざまなネットワーク条件下で、任意のクラウドまたはデータセンターから任意のデバイスに、リモートデスクトップやアプリケーションストリーミングを配信するのに役立つ高性能リモート表示プロトコルです。このパターンでは、ハイパフォーマンスコンピューティング

(HPC) の 3D グラフィックをリモートでストリーミングする帯域幅効率の高いエクスペリエンスを提供します。

- 「NICE DCV セッションマネージャー」は、NICE DCV サーバーフリートの NICE DCV セッションのライフサイクルを管理するのに役立ちます。
- 「NICE EnginFrame」 は、クラウド内の技術および科学アプリケーションにアクセスするための高度なフロントエンド Web インターフェイスです。

コードリポジトリ

このパターンのコードは、「NICE EnginFrame と NICE DCV セッションマネージャーリポジトリを備えた自動スケーリング VDI」ソリューションで利用できます。

エピック

仮想デスクトップインフラストラクチャをデプロイ

タスク	説明	必要なスキル
リポジトリをクローン作成します。	コードを含むリポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/elastic-vdi-infrastructure.git`	クラウドアーキテクト

必要な AWS CDK ライブラリをインストールします。	AWS CDK ライブラリをインストールします。
	`cd elastic-vdi-infrastructure`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	`pip3 install -r requirements.txt`
		クラウドアーキテクト

タスク	説明	必要なスキル
パラメータを更新します。	1. 選択のテキストエディタで、 app.py ファイルを開きます。 2. 次の必須パラメータに対して、 CHANGE_ME の値を置き換えます。 • region – ターゲット AWS リージョン。詳細な一覧については、「AWS リージョン」を参照してください。 • account – ターゲットの ID AWS アカウント。詳細については、AWS アカウント 「ID の検索」を参照してください。 • key_name – Linux Amazon EC2 インスタンスへのアクセスに使用されるキーペア。 3. (オプション) 以下のパラメータの値を変更して、環境に合わせてソリューションをカスタマイズします。 • ec2_type_enginframe — EnginFrame インスタンスタイプ • ec2_type_broker — セッションマネージャーブローカーのインスタンスタイプ • ebs_enginframe_size — EnginFrame インス	クラウドアーキテクト

タスク	説明	必要なスキル
	タンスの Amazon Elastic Block Store (Amazon EBS) ボリュームのサイズ • `ebs_broker_size` – Session Manager ブローカーインスタンスの Amazon EBS ボリュームのサイズ • `TagName and TagValue` — リソースの請求タグ。 • `efadmin_uid` — EnginFrame 管理者 (efadmin) ユーザーの一意の識別子 • `linux_shared_storage_size` — ギガバイナリバイト (GiB) 単位の OpenZFS サイズ • `Shared_Storage_Linux` — 共有ストレージのマウントポイント • `Enginframe_installer` — EnginFrame ダウンロードリンク • `Session_Manager_Broker_Installer` — セッションマネージャーブローカーのダウンロードリンク。 4. app.py ファイルを保存して閉じます。	

タスク	説明	必要なスキル

ソリューションをデプロイします。	次のコマンドを順番に実行します。
	
	`cdk bootstrap`
	`cdk deploy Assets-Stack Parameters-Stack`
	`cdk deploy Elastic-Vdi-Infrastructure`
	
	デプロイが完了すると、次の 2 つの出力が返されます。
	
	- `Elastic-Vdi-Infrastructure.EnginFrameURL` — EnginFrame ポータルの HTTPS アドレス
	- `Elastic-Vdi-InfrastruSecretEFadminPassword` — efadmin ユーザーのパスワードを含むシークレットの Amazon リソースネーム (ARN)
	
	これらの値をメモしておきます。これらはこのパターンの後半で使用します。
		クラウドアーキテクト

タスク	説明	必要なスキル
多数の Linux サーバーをデプロイします。	1. にサインインし AWS Management Console、CloudFormation コンソールを開きます。 2. スタックの作成を選択し、新しいリソースを使用(標準)を選択します。 3. cloudformation_files フォルダーで dcv-linux-fleet.yaml ファイルを選択します。 4. スタックの詳細の指定ページで、以下のパラメータを定義します。 • スタック名 – スタック名です。 • DCV Fleet — NICE DCV フリートの名前。この値を空白のままにしたり、スペースを使用したりしないでください。 • InstanceType — フリートのインスタンスタイプ。 • RootVolumeSize – Linux Amazon EC2 インスタンスのルートボリュームサイズ。 • MinSize — 使用可能で DCV セッションを実行していない必要があるノードの最小数。たとえば、2 と入力した場合、ソリューションはノード	クラウドアーキテクト

タスク	説明	必要なスキル
	が 2 つから始まります。ユーザーがセッションを作成すると、使用可能なノードの数が 1 に減り、ソリューションは最小限を維持するために別のノードを作成します。 • MaxSize — フリート内の最大ノード数。最大数に達した場合、ユーザーは新しいセッションを開始できません。 • BillingTagName — 請求に使用されるタグ名。このタグ名は Windows スタックに使用されているものとは異なる必要があります。 • 請求タグ値 — 請求に使用されるタグ値。 5. スタック作成ウィザードを完了し、Submit を選択してスタックの作成を開始します。	

タスク	説明	必要なスキル
フリートな Windows サーバーをデプロイしましょう。	1. にサインインし AWS Management Console、CloudFormation コンソールを開きます。 2. スタックの作成を選択し、新しいリソースを使用(標準)を選択します。 3. cloudformation_files フォルダーで dcv-windows-fleet.yaml ファイルを選択します。 4. スタックの詳細の指定ページで、以下のパラメータを定義します。 • スタック名 – スタック名です。 • DCV Fleet — NICE DCV フリートの名前。この値を空白のままにしたり、スペースを使用したりしないでください。 • InstanceType — フリートのインスタンスタイプ。 • RootVolumeSize – Windows Amazon EC2 インスタンスのルートボリュームサイズ。 • MinSize — 使用可能で DCV セッションを実行していない必要があるノードの最小数。	クラウドアーキテクト

タスク	説明	必要なスキル
	• MaxSize — フリート内の最大ノード数。 • BillingTagName — 請求に使用されるタグ名。このタグ名は Linux スタックで使用されているものとは異なる必要があります。 • 請求タグ値 — 請求に使用されるタグ値。 5. スタック作成ウィザードを完了し、Submit を選択してスタックの作成を開始します。	

デプロイされた環境にアクセスする

タスク	説明	必要なスキル
EnginFrame 管理者パスワードを取得します。	EnginFrame 管理アカウントには efadmin という名前が付けられ、パスワードはシークレット AWS Secrets Manager として に保存されます。シークレットの ARN は動的に生成され、 AWS CDK デプロイの出力に表示されます。 1. 前のエピックのソリューションのデプロイ」ストーリーで、`Elastic-Vdi-Infrastructure.SecretEFadminPassword` 出力の下にある生成	クラウドアーキテクト

タスク	説明	必要なスキル
	されたシークレットの ARN を見つけてください。 2. シークレットを取得するには、次のいずれかの操作をします。 • 「Secrets Manager コンソール」を使用します。詳細については、「シークレットの取得」を参照してください。 • 「get-secret-value」コマンドを入力します。 `aws secretsmanager get-secret-value \ --secret-id <secret_arn> \ --query SecretString \ --output text`	
EnginFrame ポータルにアクセスします。	1. 前のエピックのソリューションのデプロイ」ストーリーで、`Elastic-Vdi-Infrastructure.EnginFrameURL` 出力の下にある EnginFrame ポータルの HTTPS アドレスを検出してください。 2. ウェブブラウザで、ポータルの HTTPS アドレスを入力します。 3. efadmin ユーザーの認証情報を入力します。	クラウドアーキテクト

タスク	説明	必要なスキル
Windows セッションを開始します。	1. EnginFrame ポータルのメニューで、Windows デスクトップ」を選択します。 2. Windows 管理者としてサインインするように求められたら、efadmin ユーザーと同じパスワードを入力します。 3. Windows セッションが正常に開始されることを確認します。	クラウドアーキテクト
Linux セッションを開始します。	1. EnginFrame ポータルのメニューで、Linux デスクトップ」を選択します。 2. サインインするように求められたら、efadmin ユーザーの認証情報を入力します。 3. Linux セッションが正常に開始されることを確認します。	クラウドアーキテクト

クリーンアップ

タスク	説明	必要なスキル
スタックを削除します。	CloudFormation コンソールで、Windows および Linux サーバーフリートのスタックを削除します。詳細について、「スタックの削除」を参照してください。	クラウドアーキテクト

タスク	説明	必要なスキル
インフラストラクチャを削除します。	次の AWS CDK コマンドを使用して、デプロイされたインフラストラクチャを削除します。 `cdk destroy --all`	クラウドアーキテクト

トラブルシューティング

問題	ソリューション
デプロイは中断されたため完了しませんでした。	クリーンアップエピックの指示に従い、このパターンを繰り返して環境を再度デプロイします。

関連リソース

- NICE DCV
- 「素敵なEnginFrame」

ハイブリッドクラウド

トピック

- Hybrid Linked Mode を使用して VMware Cloud on AWS へのデータセンター拡張を構成する
- VMware vRealize Automation を VMware Cloud on AWSにプロビジョニングするように設定
- VMware Cloud on AWS を使用して VMware SDDC on AWS をデプロイする
- VMware vRealize Network Insight を VMware Cloud on AWS と統合する
- HCX OS アシストの移行を使用して、 VMware Cloud on AWSに VM を移行
- VMware Aria Operations for Logs を使用して VMware Cloud on から Splunk AWS にログを送信する
- AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する
- その他のパターン

Hybrid Linked Mode を使用して VMware Cloud on AWS へのデータセンター拡張を構成する

作成者: Deepak Kumar (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 の AWS 担当者にお問い合わせください。

このパターンでは、単一の VMware vSphere Client インターフェイスを使用して、オンプレミスデータセンターと VMware Cloud on AWS ソフトウェア定義データセンター (SDDC) のインベントリを Hybrid Linked Mode を使って表示および管理する方法について説明します。

Hybrid Linked Mode を構成することで、オンプレミス仮想マシン (VM) とアプリケーションをクラウド SDDC に移行できます。これにより、IT チームは使い慣れた VMware ツールを使用して、新しいツールを使わずにクラウドベースのリソースを管理できます。また、VMware Cloud Gateway アプライアンスを使用することで、一貫した運用とシンプルな管理を実現できます。

このパターンでは、Hybrid Linked Mode を構成するためのオプションが 2 つありますが、一度に使用できるのは 1 つのオプションだけです。第 1 のオプションでは、Cloud Gateway アプライアンスをインストールし、それを使用してオンプレミスの vCenter Server からクラウド SDDC にリンクします。第 2 のオプションでは、クラウド SDDC から Hybrid Linked Mode を構成します。

前提条件と制限

前提条件 (両方のオプション)

- 既存のオンプレミスデータセンターとクラウド SDDC。
- AWS Direct Connect、VPN、またはその両方を使用した、オンプレミスデータセンターとクラウド SDDC との既存の接続。
- オンプレミスデータセンターとクラウド SDDC が、ネットワークタイムプロトコル (NTP) またはその他の信頼できるタイムソースと同期されていること。
- オンプレミスデータセンターとクラウド SDDC 間の往復時間の最大レイテンシーが 100 ミリ秒を超えないこと。
- オンプレミス環境にアクセスできるクラウド管理者。

- vCenter Server の完全修飾ドメイン名 (FQDN) は、プライベート IP アドレスに変換される必要があります。

オプション 1 の前提条件

- オンプレミス環境は vSphere 6.5.0d 以降で実行する必要があります。
- Cloud Gatewayアプライアンスと vCenter Server が、AWS Direct Connect、VPN、またはその両方を介して通信できること。
- Cloud Gateway アプライアンスがハードウェア要件を満たしていること。
- ファイアウォールポートが開いていること。

オプション 2 の前提条件

- オンプレミスの vCenter サーバーを vSphere 6.0 アップデート 3 以降、または vSphere 6.5.0d 以降で実行していること。
- オンプレミスの vSphere シングルサインオン (SSO)ドメインにログインするための認証情報があること。
- オンプレミス環境のユーザーに、ベース識別名 (ベース DN) への読み取り専用アクセス権があること。
- オンプレミスのドメインネームシステム (DNS) サーバーは VMware 管理ゲートウェイ用に構成されていること。
- ネットワーク接続テストは、VMware 接続検証ツールを使用して実施してください。
- ファイアウォールポートが開いていること。

制約事項

- Hybrid Linked Mode は、オンプレミスの vCenter Server 拡張リンクモードドメインを 1 つだけ接続できます。
- Hybrid Linked Mode は、バージョン 6.7 以降を実行するオンプレミスの vCenter サーバーのみをサポートします。

アーキテクチャ

次の図は、Hybrid Linked Mode を構成するための両方のオプションを示しています。

Hybrid Linked Mode を使用してさまざまなワークロードタイプを移行する

Hybrid Linked Mode は、コールドマイグレーションまたは VMware vSphere vMotion によるライブマイグレーションのいずれかを使用して、オンプレミスデータセンターとクラウド SDDC 間のワークロードの移行をサポートします。移行方法を選択する際に考慮すべき要素には、仮想スイッチの種類とバージョン、クラウド SDDC への接続タイプ、仮想ハードウェアのバージョンなどがあります。

コールドマイグレーションは、ダウンタイムが発生する VM に適しています。VM をシャットダウンして移行し、再起動させてもよいでしょう。アクティブなメモリをコピーする必要がないため、移行時間が短縮されます。ダウンタイムを許容するアプリケーション (Tier 3 アプリケーション、開発およびテストのワークロードなど) には、コールドマイグレーションをお勧めします。VM でダウンタイムが発生しない場合は、ミッションクリティカルなアプリケーションに vMotion を使用したライブマイグレーションを検討してください。

次の図は、Hybrid Linked Mode を使用するさまざまなワークロード移行タイプの概要を示しています。

ツール

- VMware Cloud on AWSは、AWS と VMware が共同開発した統合クラウドサービスです。
- VMware Cloud Gateway アプライアンス は、オンプレミスのリソースをクラウドリソースに接続するハイブリッドクラウドのさまざまなユースケースを可能にします。
- VMware vSphere は VMware の仮想化プラットフォームであり、データセンターを CPU、ストレージ、ネットワークリソースを含む集約コンピューティングインフラストラクチャに変換します。

エピック

オプション 1 - Cloud Gateway アプライアンスで Hybrid Linked Mode を使用する

タスク	説明	必要なスキル
Cloud Gateway アプライアンスを構成します。	1. VMware Cloud on AWS コンソールにログイン	クラウド管理者

タスク	説明	必要なスキル
	し、Cloud Gateway アプライアンスをダウンロードします。 2. 次の 2 つのステップで Cloud Gateway アプライアンスをオンプレミス環境にインストールします。 • [構成の開始] を選択して Cloud Gateway アプライアンスを構成し、デプロイします。 • Hybrid Linked Mode を構成します。 詳細と詳しい手順については、「VMware ドキュメント」の「vCenter Cloud Gateway アプライアンスを使用した Hybrid Linked Mode の構成」を参照してください。	

オプション 2 - クラウド SDDC から Hybrid Linked Mode を使用する

タスク	説明	必要なスキル
クラウド SDDC からHybrid Linked Mode を構成します。	1. VMware Cloud on AWS コンソールにログインし、接続検証ツールを使用して、必要なすべてのネットワーク接続を確認します。詳細については、「VMware ドキュメント」の「Hybrid Linked Mode のネットワー	クラウド管理者

タスク	説明	必要なスキル
	ク接続の検証」を参照してください。 2. クラウド SDDC の vSphere クライアントにログインして[メニュー] を選択し、[管理] を選択して、[ドメイン] を選択します。 3. [ハイブリッドクラウド] セクションで、[リンクされているドメイン] を選択し、オンプレミスの vCenter Server に接続します。 4. クラウド SDDC Lightweight Directory Access Protocol (LDAP) ドメインにアイデンティティソースを追加します。詳細については、「VMware ドキュメント」の「SDDC LDAP ドメインへの ID ソースの追加」を参照してください。	

関連リソース

- Hybrid Linked Mode の構成
- VMware Cloud on AWS のHybrid Linked Mode の構成

VMware vRealize Automation を VMware Cloud on AWSにプロビジョニングするように設定

作成者: Deepak Kumar (AWS)

概要

注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

「VMware vRealize Automation」 は、IT リソースのリクエストと管理に使用できる自動化ソフトウェアです。vRealize Automation を VMware Cloud on AWS で構成することを選択すると、複数のデータセンターやクラウド環境にわたる仮想マシン (VM)、アプリケーション、および IT サービスの配信を自動化できます。

次に、IT チームはカタログアイテムを作成して、サービスのプロビジョニングと運用機能を設定できます。それによりユーザーはリクエストをして、既存の vRealize Automation ツールを使って使用できます。また、VMware Cloud on AWS を 「vRealize Automation クラウドアセンブリ」 と統合することで、IT のアジリティと効率性を向上させることもできます。

このパターンでは、VMware Cloud on AWS で VM またはアプリケーション機能を自動的に構築するように、VMware vRealize Automation を構成する方法について説明します。

前提条件と制限

前提条件

- 既存のオンプレミスデータセンターと VMware Cloud on AWS ソフトウェア定義データセンター (SDDC)。クラウド SDCC の詳細については、VMware ドキュメントの「ソフトウェア定義データセンターについて」 を参照してください。
- AWS Direct Connect、VPN (ルートまたはポリシーベース)、またはその両方を使用した、オンプレミスデータセンターとクラウド SDDC 間の既存の接続です。
- オンプレミスデータセンターとクラウド SDDC は、ネットワークタイムプロトコル (NTP) または別の信頼できるタイムソースと同期されます。
- オンプレミスデータセンターとクラウド SDDC 間の往復時間の最大待ち時間は、100ms を超えてはなりません。

- vCenter Server の完全に指定されたドメイン名 (FQDN) は、プライベート IP アドレスに変換する必要があります。
- オンプレミス環境にアクセスできるクラウド SDDC ユーザー。
- vRealize Automation クラウドアセンブリサービスロールにおける組織オーナーアクセス。
- vRealize Automation サービスブローカで、サービスを利用する権限を持つエンドユーザー。
- VMware Cloud on AWS コンソールから API トークンを生成するには、オンプレミスデータセンターのClassless Inter-Domain Routing (CIDR) の範囲が開いている必要があります。API トークンの生成に必要な最低限のロールを以下に示します：
  - 組織のメンバー
  - 組織の所有者
  - サービスロール - VMware Cloud on AWS
  - 管理者
  - NSX クラウド管理者
  - NSX Cloud Auditor

この詳細については、AWS パートナーネットワークブログの「VMware Cloud on AWS SDDC の接続オプション」を参照してください。

機能制限

- 1 つの vRealize Automation では、パブリックエンドポイントを持つ VMware クラウドアカウントを 20 個までしか構成できません。この詳細については、VMware ドキュメントの「スケーラビリティと同時実行数の上限」を参照してください。

製品バージョン

- vRealize Automation バージョン 8.x 以降
- VMware vRealize ID マネージャバージョン 3.x 以降
- VMware vRealize Suite ライフサイクルマネージャバージョン 8.x 以降

アーキテクチャ

次の図表は、オンプレミス環境と VMware Cloud on AWS 環境の両方のインフラストラクチャを使用できる vRealize Automation サービスを示しています。

VMware クラウドアセンブリコンポーネント

VMware クラウドアセンブリは vRealize Automation のコアコンポーネントであり、これを使用して VM やコンピューティングリソースをデプロイし、プロビジョニングすることができます。次の表は、VMware Cloud on AWS で VM をプロビジョニングするために設定する必要がある VMware クラウドアセンブリコンポーネントについて説明しています。

コンポーネント	定義
クラウドアカウント	クラウドアカウントは、接続情報 (サーバー名、ユーザー名とパスワード、アクセスキー、API トークンなど) を提供します。VMware Cloud Assembly は、クラウドアカウントを使用してリソースのインベントリを収集します。
クラウドゾーン	クラウドゾーンは、クラウドアカウントのリソースの境界 (AWS リージョンやクラウド SDDC など) を識別します。クラウドゾーンは、コンピューティンリソースを Cloud Assembly プロジェクトに関連付けます。
プロジェクト	プロジェクトは、ユーザーとクラウドゾーンなどのリソースで構成される論理要素です。また、VM を構築する際に使用されるリソースクォータと VM 命名ポリシーも含みます
フレーバーマッピング	フレーバーマッピングは、クラウドテンプレートで使用されている VM の容量 (例、CPU の数やメモリ量) に関する情報を提供します。
イメージマッピング	イメージマッピングは、クラウドテンプレートで使用される VMware vSphere VM テンプレートとAmazon Web Services (AWS) イメージをマッピングします。これに関する詳細は、VMware ドキュメントの「vRealize Automation のイ

	メージマッピングについて理解を深める」を参照してください。
ネットワークプロファイル	ネットワークプロファイルは、 VM のプロビジョニング中にネットワークを選択するための配置決定を制御します。
ストレージプロファイル	ストレージプロファイルは、VM のプロビジョニングにストレージを選択するための配置決定を制御します。
クラウドテンプレート	VMware クラウドテンプレートは vRealize Automation の重要なコンポーネントです。理由は、クラウドインフラストラクチャのプロビジョニングとオーケストレーションを定義するためです。クラウドテンプレートはリソースの仕様であり、リソースタイプ、リソースプロパティ、およびユーザーから収集される入力が含まれます。

ツール

- 「VMware vRealize Automation」 — vRealize Automation は、イベント主導型の状態管理とコンプライアンスを備えたインフラストラクチャ自動化プラットフォームです。組織がセルフサービスクラウド、ガバナンスによるマルチクラウド自動化、DevOps ベースのインフラストラクチャ配信を制御および保護できるように設計されています。
- 「VMware Cloud on AWS」 — VMware Cloud on AWS は、AWSとVMwareが共同開発した統合クラウド製品です。

エピック

APIトークンを生成します

タスク	説明	必要なスキル
VMware Cloud on AWS アカウントからAPIトークンを生成します。	1. VMware クラウドコンソールにサインインします。 2. VMware クラウドサービスのツールバーでマイアカウントを選択し、次に API トークンを選択します。 3. API トークンの名前を入力し、必要な有効期間を指定し、トークンのスコープを定義します。 4. Open ID チェックボックスを選択し、次に生成 を選択します。 5. API トークンの認証情報を記録します。 詳細については、VMware ドキュメントの 「API トークンを生成する方法」 を参照してください。	クラウド管理者

オンプレミスデータセンターに vRealize Automation をインストールします

タスク	説明	必要なスキル
必要なソフトウェアをダウンロードします。	マイ VMware ポータルから VMware vRealize Suite ISO ファイルをダウンロードします。このパッケージには	クラウド管理者

タスク	説明	必要なスキル
	vRealize Suite ライフサイクルマネージャー、VMware ID マネージャー、vRealize Automation が含まれています。	
ソフトウェアをインストールします。	VMware ドキュメントの「vRealize Automation VMware ID マネージャーの簡単インストーラを搭載した vRealize 適合ライフサイクルマネージャーのインストール」の手順に従って、ソフトウェアをインストールしてクラウド SDCC に接続します。 **⚠ Important** インストールで以下が使用可能であることを確認します。 • オンプレミスの VMware vCenter サーバのセットアップとログイン認証情報 • vRealize Automation IP とサブネットのネットワーク詳細 • vRealize Automation のライセンスキー	クラウド管理者、クラウドアーキテクト

VMware Cloud on AWS と VMware クラウドアセンブリを接続

タスク	説明	必要なスキル
クラウドアカウントを設定します。	1. VMware Cloud コンソールで、[インフラストラクチャ] タブを開き、[管理 - クラウドアカウント] を選択し、次に [クラウドアカウントを追加] 」を選択します。 2. タイプとして「VMware Cloud on AWS」を選択します。 3. 先に記録した API トークン情報を貼り付けます。これにより、VMware Cloud on AWS 組織に、使用可能なすべてのクラウド SDDC が入力されます。 4. 必要なクラウド SDCC を選択し、次にSDDC の vCenter ユーザー名とパスワードを入力します。 5. 認証に成功すると、統合された VMware Cloud on AWS アカウントが OK ステータスで表示されます。 詳細については、VMware ドキュメントの「vRealize Automation で VMware Cloud on AWS クラウドアカウントを作成する」を参照してください。	クラウドアーキテクト、クラウド管理者

タスク	説明	必要なスキル
プロジェクトを設定します。	1. VMware Cloud コンソールで [プロジェクト] タブを開き、次に新規プロジェクトを選択します。 2. プロジェクトの名前を入力します。 3. [クラウドゾーン] タブを開き、[デフォルトの VMware Cloud on AWS クラウドアカウント] を選択します。	クラウド管理者
クラウドゾーンを設定します。	1. VMware Cloud コンソールで 「クラウドゾーン」を開き、SDDC データセンターの クラウドゾーンを選択します。 2. デフォルトでは、`cloudadmin@vmc.local` （これはクラウド SDDC vCenter のデフォルトのローカルユーザー ID です) は、`Compute-ResourcePool` のプロビジョニングにのみアクセスできます。 3. クラウドゾーンの配下の [コンピューティング] タブを開き、[Compute-ResourcePool] を選択します。	クラウド管理者

タスク	説明	必要なスキル
フレーバーマッピングを設定します。	1. [フレーバーマッピング] タブを開き、新規のフレーバーマッピングを作成します。 2. フレーバー名を入力し、VMware Cloud on AWS アカウントを選択してから、vCPUs の数とメモリ容量を指定します。	クラウド管理者
イメージマッピングを設定します。	1. [イメージマッピング] を開き、新しいイメージマッピングを作成します。 2. イメージ名を入力します。 3. VMware Cloud on AWS アカウントを選択し、必要なクラウドアカウントテンプレートを提供します。	クラウド管理者
ネットワークプロファイルを設定します。	1. [ネットワークプロファイル] を開き、新規ネットワークプロファイルを作成します。 2. ネットワークプロファイル名を入力します。 3. [ネットワーク] タブを開き、プロビジョニングに使用する既存のネットワークを選択します。	クラウド管理者

タスク	説明	必要なスキル
ストレージプロファイルを設定します。	1. [ストレージプロファイル] を開き、[新規ストレージプロファイル] を選択します。 2. ストレージプロファイルの名前を入力します。 3. ポリシーセクションで、新規ポリシーを作成します。 4. [ワークロードデータストア] を選択します。デフォルトでは、 `cloudadmin@vmc.local` のみがワークロードのデータストアのプロビジョニングにアクセスできます。	クラウド管理者

タスク	説明	必要なスキル
クラウドテンプレートを作成します。	1. [デザイン] タブを開き、[クラウドテンプレート] を選択し、[新規作成元] と [空白のキャンバス] を選択します。 2. クラウドテンプレートの名前と説明を入力します。 3. 先に作成したプロジェクトを選択します。 4. クラウドテンプレートリソースのデザインページから、要件に応じて、コンポーネントを空白のキャンバスにドラッグします。 5. [テスト] を選択してテンプレートをテストし、問題があれば修正します。 6. [デプロイ] を選択し、デプロイ名前を提供して、VM をデプロイします。 詳細については、VMware ドキュメントの「基本的なクラウドテンプレートの作成」を参照してください。	クラウド管理者

関連リソース

- 「vRealize Automation バージョン 8.x を、SSDDCに接続」
- 「VMware Cloud on AWS コンソールから SDDC をデプロイ」
- 「AWS Direct Connect を VMware Cloud on AWS に統合」

VMware Cloud on AWS を使用して VMware SDDC on AWS をデプロイする

作成者: Deepak Kumar (AWS) と Derek Cox (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

このパターンでは、Amazon Web Services (AWS) クラウドでホストされる VMware ベースのソフトウェア定義データセンター (SDDC) を作成する方法を説明します。SDDC をデプロイして VMware vSphere ベースのワークロードを AWS クラウドに移行し、既存の VMware ツールとスキルを使用しながら AWS のサービスを活用できます。この SDDC を使用すると、VMware vSphere ベースのプライベート、パブリック、ハイブリッドクラウド環境全体で本稼働アプリケーションを実行でき、AWS サービスへのアクセスが最適化されます。たとえば、SDDC をディザスタリカバリのセカンダリサイトとして使用、またはデータセンターを地理的に異なる場所に拡張できます。

VMware Cloud on AWS は、すべての規模の企業が、さまざまな AWS サービスを使用して VMware vSphere ベースのクラウド環境でワークロードを実行できるようにする従量制料金 (オンデマンド) サービスです。SDDC クラスターあたり最低 2 台のホストから開始し、本番環境ではクラスターあたり 16 台のホストまでスケールアップできます。詳細については、VMware Cloud on AWS の Web サイトを参照してください。SDDC の詳細については、VMware ドキュメントのソフトウェア定義のデータセンターについてを参照してください。

前提条件と制限

前提条件

- MyVMware アカウントにサインアップし、すべてのフィールドに入力します。
- AWS アカウントにサインアップします。手順については、AWS ナレッジセンターを参照してください。
- MyVMware Cloud on AWS アカウントにサインアップします。アクティベーションリンクは、サインアップ時に指定したメールアドレスに送信されます。

制限

- VMware Web サイトの VMware Cloud on AWS 設定制限ページを参照してください。

製品バージョン

- VMware ドキュメントの VMware Cloud on AWS リリースノートを参照してください。

アーキテクチャ

ターゲットテクノロジースタック

次の図は、AWS ベアメタル専用インフラストラクチャで実行中の vSphere、vCenter、vSAN、NSX-T など、VMware ソフトウェアスタックを示しています。Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、AWS Direct Connect、AWS Direct Connect、Amazon Relational Database Service (Amazon RDS)、Amazon DynamoDB など、その他の AWS サービスとシームレスに統合する AWS の VMware ベースのリソースとツールを管理できます。

VMware Cloud on AWS の基本的なエンティティは SDDC で、これには以下のコンポーネントが含まれます。

- コンピュート: コンピュートコンポーネントは VMware Cloud on AWS SDDC の最下層です。VMware Cloud on AWS は、Amazon EC2 ベアメタルインスタンスタイプで実行されます。`i3.metal`、 `i3en.metal`、`i4i.metal` があり、プロセッサとメモリなどの物理リソースに直接アクセスできます。

  > **Important**
  >
  > VMware VMware Cloud on AWS の`i3.metal`インスタンスタイプは、1 年および 3 年の期間のオンデマンドオプションとサブスクリプションオプションを含め、2026 年 12 月 31 日にサポート終了とサポート終了に達するように設定されています。また、現在、新規のお客様は `i3.metal` インスタンスをリクエストできません。詳細については、VMware クラウドブログのお知らせを参照してください。

- ストレージ: SDDC クラスターは、高速で高性能のストレージを提供する不揮発性メモリエクスプレス (NVMe) フラッシュストレージを使用するオールフラッシュ設定の VMware vSAN をサポートしています。 SDDC バージョン 1.20 以降、VMware Cloud on AWS は、Amazon FSx for

NetApp ONTAP と VMware Cloud Flex Storage の 2 つのタイプの外部ストレージをサポートしていいます。

- ネットワーク: ネットワーク機能とポリシーは SDDC クラスター内の VMware NSX-T を使用して管理されます。SDDC クラスターでは多層仮想ネットワークが作成され、ネットワークリソースと物理機器が分離されます。これにより、VMware Cloud on AWS ユーザーは、論理的なソフトウェア定義ネットワークを構築できます。

ツール

- VMware Cloud on AWS は、AWS と VMware が共同開発した統合クラウドです。

エピック

AWS アカウントに VPC とサブネットを作成する

タスク	説明	必要なスキル
AWS アカウントにサインインします。	管理者権限がある認証情報を使用して、AWS アカウントにサインインします。	クラウド管理者
新しい VPC を作成します。	この手順では、SDDC にリンクされた仮想プライベートクラウド (VPC) を定義します。SDDC に使用する VPC がすでにある場合は、この手順をスキップします。 1. AWS リージョンを選択して VMware Cloud on AWS SDDC をデプロイします。 2. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。	クラウド管理者

タスク	説明	必要なスキル
	3. ナビゲーションペインで、[Your VPCs (お使いの VPC)] を選択します。 4. [Create VPC (VPC の作成)] を選択します。 5. VPC の名前タグ、IPv4 CIDR ブロック、テナンシー (デフォルトとして保持) などの VPC 設定を指定してから、[Create VPC (VPC の作成)] を選択します。 6. VPC が作成されたら、[Close (閉じる)] を選択します。 詳細については、AWS ドキュメントの VPC の作成と設定を参照してください。	

タスク	説明	必要なスキル
プライベートサブネットを作成します。	次に、各アベイラビリティーゾーンの Elastic Network Interface (ENI) のプライベートサブネットを作成します。インターネットゲートウェイが接続されていないサブネットの使用を推奨します。 1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。 2. ナビゲーションペインで、[Subnets (サブネット)] を選択します。 3. [Create Subnet (サブネットの作成)] を選択します。 4. [Create Subnet (サブネットの作成)] ページで、前の手順で作成した VPC を選択します。 5. サブネット名、アベイラビリティーゾーン、IPv4 CIDR ブロックなど、サブネットの設定を完了します。 6. [Create Subnet (サブネットの作成)] を選択します。 これらの手順を繰り返して、リージョンの各アベイラビリ	クラウド管理者

タスク	説明	必要なスキル
	ティーゾーンにサブネットを作成します。	

VMware Cloud on AWS をアクティブ化する

タスク	説明	必要なスキル
サービスをアクティブ化します。	MyVMware アカウントにサインアップすると、VMware は指定したメールアドレスにウェルカムメールとアクティベーションリンクを送信します。 1. ウェルカムメールにあるサービスのアクティブ化リンクをブラウザで開きます。 2. MyVMware 認証情報でログインします。 3. サービスの使用に関する利用規約を確認し、同意します。 4. アカウントのアクティベーションプロセスを完了します。VMware Cloud on AWS コンソールにリダイレクトされます。(注：VMware Cloud on AWS アカウントは、アカウントにサブスクライブしているグループまたは事業部門を表す組織に基づいています。この組織	クラウド管理者

タスク	説明	必要なスキル
	は AWS Organizations とは何の関係もありません。) 5. [Select or Create Organization (組織の選択または作成)] ページで、MyVMware アカウントにリンクされた組織を作成します。 6. 論理的に区別できるように、組織名と住所を入力します。 7. [Create Organization (組織の作成)] を選択して、プロセスを完了します。 このプロセスの詳細については、AWS ドキュメントの AWS での SDDC デプロイとベストプラクティスガイドを参照してください。	

タスク	説明	必要なスキル
IAM ロールを割り当てます。	組織が作成されたら、クラウドサービスと SDDC コンソール、SDDC、NSX コンポーネントにアクセスする特権アクセスを特定のユーザーに割り当てます。手順については、VMware ドキュメントの組織のメンバーに VMC サービスロールを割り当てるを参照してください。 次の 2 つのタイプの組織ロールがあります。 • 組織所有者は、ユーザーを追加、削除、変更、およびすべてのクラウドリソースにアクセスできます。 • 組織メンバーはクラウドリソースにのみアクセスできます。	クラウド管理者

SDDC をデプロイする

タスク	説明	必要なスキル
SDDC を VMware Cloud on AWS アカウントにデプロイします。	⚠ Important AWS アカウントが登録販売者として VMware Organization に関連付けられると、AWS アカウント番号を更新することは	クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	できません。VMware Organization ごとに登録できる AWS 販売者は 1 つだけです。 SDDC をデプロイするには： 1. VMC コンソール（https://vmc.vmware.com）にログインします。 2. 利用可能なサービスから、VMware Cloud on AWS Service を選択します。 3. [Create SDDC（ロールの作成）] を選択します。 4. AWS リージョン、デプロイ (シングルホスト、マルチホスト、またはストレッチクラスター)、ホストタイプ、SDDC 名、ホスト数、ホスト容量、合計容量などの SDDC プロパティを入力してから、[Next（次へ）] を選択します。 5. AWS アカウントに接続し、[Next（次へ）] を選択します。 6. 以前に設定した VPC とサブネットを選択してから、[Next（次へ）] を選択します。 7. SDDC の管理サブネット CIDR ブロックを入力して	

タスク	説明	必要なスキル
	から、[Next (次へ)] を選択します。詳細については、VMware Cloud ブログの SDDC の IP サブネットと接続性の選択を参照してください。 8. この 2 つのチェックボックスを選択して SDDC のデプロイ費用を負担することを承諾してから、[Deploy SDDC (SDDC のデプロイ)] を選択します。 [Deploy SDDC (SDDC のデプロイ)] を選択すると請求されます。デプロイプロセスは一時停止またはキャンセルできないため、完了するまでに時間がかかります。 SDDC の作成の詳細については、VMware ドキュメントの VMC コンソールからの SDDC のデプロイを参照してください。	

関連リソース

- ソフトウェア定義のデータセンターのデプロイと管理」(VMware ドキュメント)
- VMware Cloud on AWS 機能 (AWS Web サイト)
- VMware Cloud on AWS でクラウドへの移行とモダナイゼーションを加速する (動画)

VMware vRealize Network Insight を VMware Cloud on AWS と統合する

作成者: Deepak Kumar (AWS)、Pioter Pitera (AWS)、Sachin Trivedi (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

このパターンでは、VMware vRealize Network Insight を VMware Cloud on と統合 AWS し、仮想マシンからのトラフィックフローを調べる方法について説明します。この統合は、VMware Cloud on へのアプリケーションの移行を計画するのにも役立ちます AWS。

vRealize Network Insight は、ネットワークインフラストラクチャを可視化します。セキュリティの向上、移行リスクの軽減、パフォーマンスの最適化のためのネットワークモニタリングおよび分析機能を提供します。このツールを使用して、仮想マシンからのトラフィックフローをモニタリングし、観測されたトラフィックに基づいて推奨されるセキュリティルールを表示できます。vRealize Network Insight の詳細については、VMware のドキュメントを参照してください。

VMware Cloud on AWS は従pay-as-you-go (オンデマンド) サービスで、あらゆる規模の企業が幅広い を使用して VMware vSphere ベースのクラウド環境全体でワークロードを実行できます AWS のサービス。SDDC クラスターあたり最低 2 台のホストから開始し、本番環境ではクラスターあたり 16 台のホストまでスケールアップできます。詳細については、VMware Cloud on AWSウェブサイトを参照してください。SDDC の詳細については、VMware ドキュメントのソフトウェア定義のデータセンターについてを参照してください。

前提条件と制限

前提条件

- VMware Cloud on AWS SDDC、デプロイ済み

制約事項

- 既知の制限については、VMware のドキュメントを参照してください。

製品バージョン

- vRealize Network Insight バージョン 5.0.0
- VMware Cloud on AWS SDDC バージョン 1.24

アーキテクチャ

ソーステクノロジースタック

- vRealize ネットワークインサイト

ターゲットテクノロジースタック

- VMware Cloud on AWS

ターゲット アーキテクチャ

次の図は、VMware Cloud on AWS と vRealize Network Insight on オンプレミス間の接続を示しています。

ツール

- VMware Cloud on AWS は、 AWS と VMware によって共同開発された統合クラウドサービスです。
- VMware vRealize Network Insight は、セキュリティの計画とトラブルシューティングのためにネットワークインフラストラクチャを可視化するモニタリングおよび分析ツールです。

エピック

vRealize Network Insight の環境をセットアップする

タスク	説明	必要なスキル
VMware ユーザーアカウントを作成します。	VMware ユーザーアカウントを作成するか、既存の VMware アカウントにログインします。	クラウド管理者

タスク	説明	必要なスキル
	新しいアカウントを開くには： 1. 登録フォームに記入して、VMware Customer Connect アカウントにサインアップします。 新しいユーザーには、アカウントをアクティブ化するための E メールが送信されます。 2. E メールの認証コードを入力します。 3. Customer Connect にログインします。	
vRealize Network Insight の OVA ファイルをダウンロードします。	vRealize Network Insight の OVA ファイルをダウンロードします。 1. 「httpshttps://my.vmware.com/group/vmware/home.com で VMware 製品ダウンロードページに移動します。 2. vRealize Network Insight を検索します。 3. 最新の vRealize Network Insight バージョン 5.0.0 プラットフォームおよびコレクター OVA ファイルをダウンロードします。	クラウド管理者

タスク	説明	必要なスキル
vRealize Network Insight をデプロイします。	デプロイ手順については、VMware のドキュメントを参照してください。	クラウド管理者

データソースとコレクターを追加する

タスク	説明	必要なスキル
データソースを追加します。	1. vRealize Network Insight にログインします。 2. 設定、アカウント、データソース、ソースの追加を選択します。 3. Type で、オンプレミス vCenter サーバーを選択します。 詳細については、VMware のドキュメントを参照してください。	クラウド管理者
データソースのコレクターを設定します。	手順については、VMware のドキュメントを参照してください。	クラウド管理者

アプリケーションの依存関係を分析する

タスク	説明	必要なスキル
アプリケーションを作成します。	vRealize Network Insight に既存のアプリケーションがない場合は、VMware ドキュメ	クラウド管理者

タスク	説明	必要なスキル
	ントの手順に従って作成します。	
アプリケーションを検出して分析します。	1. vRealize Network Insight を使用してアプリケーションを検出します。手順については、VMware のドキュメントを参照してください。 2. アプリケーションを分析します。手順については、VMware のドキュメントを参照してください。	クラウド管理者

関連リソース

- VMware Cloud on を使用して VMware SDDC on AWS をデプロイ AWSする (AWS 規範ガイダンス)
- Hybrid Linked Mode AWS を使用して VMware Cloud on へのデータセンター拡張を設定する (AWS 規範ガイダンス)
- VMware HCX AWS を使用して VMware SDDC を VMware Cloud on に移行する (AWS 規範ガイダンス)
- VMware vRealize Network Insight ドキュメント (VMware ウェブサイト)

HCX OS アシストの移行を使用して、 VMware Cloud on AWSに VM を移行

作成者：Deepak Kumar (AWS) と Himanshu Gupta (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 AWS 担当者にお問い合わせください。

このパターンでは、OS アシストマイグレーション (OSAM) を使用して、仮想マシン (VM) を 非 vSphere の環境からAmazon Web Services (AWS) 上の VMware Cloud に移行する方法について説明します。

OSAM は VMware Cloud on AWS に含まれている VMware ハイブリッドクラウド拡張(HCX)の一部です。OSAM を使用して、VMware KVM や Hyper-V などの非 vSphere の環境を VMware Cloud on AWS に移行できます。OSAM は、Windows または Linux のゲスト VM にインストールされる Sentinel ソフトウェアを使用して、VM をオンプレミス環境から VMware Cloud on AWS のソフトウェア定義のデータセンター (SDDC) に複製することをサポートします。

このパターンでは、OSAM を有効にする方法、Windows VM に Sentinel ソフトウェアをインストールする方法、移行元サイトで HCX Sentinel ゲートウェイ(SGW)アプライアンスに接続して登録する方法、移行先サイトで HCX Sentinel データレシーバー(SDR)アプライアンスとの転送接続を確立して移行を開始する方法を説明しています。

OSAM の詳細については、「 VMware のドキュメント」を参照してください。

前提条件と制限

前提条件

- HCX をソース環境とターゲット環境にインストールします。HCX の前提条件については、AWS 規範ガイダンスの「VMware HCX を使用して VMware SDDC を VMware Cloud on AWS に移行する」 を参照してください。
- 「OSAM の前提条件については、VMware ドキュメントのインストールチェックリスト」を参照してください。

- OSAM ポート情報については、VMware のポートとプロトコルの Webサイトの「VMware HCX ポート要件」を参照してください。

機能制限

- 「VMware HCX 4.2.0 の構成に関する制限事項」
- 「OSAM のデプロイに関する考慮事項」
- 「サポートされるオペレーティングシステム」
- 「ゲストオペレーションシステムの考慮事項」

製品バージョン

- VMware HCX 4.2.0
- VMware SDDC 1.12

アーキテクチャ

次の図表では、どのようにHCX OSAM が Sentinel ソフトウェアと連携して、オンプレミス環境から非 vSphere のVMを VMware Cloud on AWS に複製するかを示しています。

OSAM は 3 つのコンポーネントで構成されています：

- Sentinel ゲートウェイ (SGW) アプライアンスは、ソースの VMware ベース環境のワークロードとアプリケーションを接続して、転送するために使用されます。
- Sentinel データレシーバー (SDR)は、移行されたワークロードをソースから受信するために、AWS 環境での移行先の VMware Cloud on AWSで使用されます。
- Sentinel ソフトウェアは、移行する各ゲスト VM にインストールする必要があります。

OSAM では、 オンプレミスから VMware SDDC にVM を複製するために、Windows または Linux のゲスト VM にインストールされている Sentinel ソフトウェアが使用されます。ゲスト VM にインストールする Sentinel ソフトウェアは、ゲスト VM からシステム構成を収集し、データ複製を支援します。この情報は、移行のためのゲスト VM のインベントリの作成にも使用され、レプリカ VM でディスクを複製や移行するための準備の際にも役立ちます。

ツール

- VMware HCX 4.2.0
- VMware Cloud on AWS SDDC

エピック

HCX を設定

タスク	説明	必要なスキル
HCX クラウドと HCX コネクタをデプロイします。	VMware ドキュメントの「HCX コネクタと HCX クラウドのインストール」の手順に従います。	クラウド管理者、システム管理者

OSAM の設定と VM の移行

タスク	説明	必要なスキル
HCX Sentinel をインストールします。	Linux に Sentinel をインストールするには: 1. HCX コネクタの vCenter サーバーで、インターコネクト、マルチサイトサービスメッシュ、センチネル管理を選択します。 2. Linux バンドルをダウンロードを選択します。 3. Linux マシンに Sentinel エージェントをインストールします。 詳細については、VMware ドキュメントの「HCX Sentinel	クラウド管理者

タスク	説明	必要なスキル
	エージェントソフトウェアのダウンロードとインストール」 を参照してください。	

タスク	説明	必要なスキル
VM を移行します。	VM をグループ (モビリティグループと呼ばれる) で移行するには、以下の手順に従います： 1. vSphere Client で、HCX プラグインから、サービス、移行 を選択します。 2. 移行 を選択します。 3. 非 vSphere のインベントリ、リモート接続を選択します。これにより、HCX Sentinel をインストールした VM が一覧表示されます。 4. グループ名では、VM のために作成するモビリティグループの名前を入力します。 5. 移行する VM を選択し、次にAdd を選択してモビリティグループに追加します。 6. 各 VM について: a. 移行先のコンピュートコンテナを選択します。 b. 移行先のストレージを選択します。 c. 移行プロファイルを選択します。 d. 移行先フォルダーを選択します。	クラウド管理者

タスク	説明	必要なスキル
	7. 移行プロセスを開始するには、 [進む] を選択します。 HCX では、移行を開始する前に VM の選択を検証します。 詳細については、VMware ドキュメントの「モビリティグループによる仮想マシンの移行」と「モビリティグループによる移行の監視と見積り」を参照してください。	

関連リソース

VMware ドキュメント：

- 「VMware HCX User Guide」
- 「VMC SDDC 移行先環境が付いたチェックリスト B - HCX をインストール」
- 「VMware Cloud on AWS の VMware HCX」
- 「VMware Cloud on AWS の HCX OS アシストマイグレーション」
- 「VMware HCX 4.2.1 リリースノート」

VMware Aria Operations for Logs を使用して VMware Cloud on から Splunk AWS にログを送信する

作成者: Deepak Kumar (AWS) と Piotr Pitera (AWS)

概要

> 注意: 2024 年 4 月 30 日をもって、VMware Cloud on AWS は AWS またはそのチャネルパートナーによって再販されなくなります。このサービスは、引き続き Broadcom を通じて利用できます。詳細については、 の AWS 担当者にお問い合わせください。

このパターンでは、VMware Aria Operations for Logs を使用して、VMware Cloud on AWS イベントまたはログを syslog または Splunk などの HTTP エンドポイントに転送する方法について説明します。

VMware Aria Operations for Logs は、VMware Cloud on AWS 環境で可視性を高め、トラブルシューティングを高速化するログ分析ツールです。このツールは、VMware Cloud on のログまたはイベントの全部または一部 AWS を syslog または HTTP エンドポイントに送信するように設定できます。エンドポイントは、Software as a Service (SaaS) エンドポイントでも、Splunk などのオンプレミスエンドポイントでもかまいません。(このパターンは Splunk の手順を示しています)。VMware Aria Operations for Logs の詳細については、VMware のドキュメントを参照してください。

VMware Cloud on AWS は従pay-as-you-go (オンデマンド) サービスで、あらゆる規模の企業が幅広い を使用して VMware vSphere ベースのクラウド環境全体でワークロードを実行できます AWS のサービス。Software-Defined Data Center (SDDC) クラスターごとに最低 2 つのホストから開始し、本番環境のクラスターごとに最大 16 のホストをスケールアップできます。詳細については、VMware Cloud on AWSウェブサイトを参照してください。SDDC の詳細については、VMware ドキュメントのソフトウェア定義のデータセンターについてを参照してください。

前提条件と制限

前提条件

- Splunk、オンプレミスで設定

機能制限

VMware Aria Operations for Logs の無料トライアルサブスクリプションにサインアップできます。このサブスクリプションは 30 日間有効で、以下の制限があります。

- 転送できるログの最大サイズ: 1 日あたり 50 GB のログ
- 作成できるログ転送設定の最大数: 10
- アクティブ化できるログ転送設定の最大数: 5

すべてのサービス機能にアクセスするには、プレミアムサブスクリプションにアップグレードする必要があります。

トライアルサブスクリプションとプレミアムサブスクリプションの詳細については、VMware ドキュメントの「VMware Aria Operations for Logs (SaaS) Subscriptions and Billing」を参照してください。 VMware 使用制限の詳細については、VMware ドキュメントの「 機能の使用制限」を参照してください。

製品バージョン

- VMware Cloud on AWS SDDC バージョン 1.24
- VMware Aria Operations for Logs バージョン 8.10
- オンプレミス Splunk バージョン 9.x

アーキテクチャ

ソーステクノロジースタック

- VMware Cloud on AWS
- VMware Aria Operations for Logs

ターゲットテクノロジースタック

- オンプレミス Splunk

ターゲット アーキテクチャ

次の図は、企業のデータセンターと VMware Cloud on の VMware Aria Operations for Logs 間の接続を示しています AWS。

ツール

- VMware Cloud on AWS は、 AWS と VMware によって共同開発された統合クラウドサービスです。
- VMware Aria Operations for Logs は、VMware Cloud on のログ分析およびトラブルシューティングツールです AWS。

エピック

SDDC をデプロイし、VMware VMware Aria Operation for Logs を有効にする

タスク	説明	必要なスキル
VMware Cloud on AWS SDDC をデプロイします。	AWS 「 規範ガイダンス」の VMware Cloud on AWS を使用して に VMware SDDC をデプロイ AWSする」の手順に従います。	クラウドアーキテクト、クラウド管理者
VMware Aria Operations for Logs にサインアップします。	手順については、VMware のドキュメントを参照してください。	クラウドアーキテクト

クラウドプロキシをデプロイする

タスク	説明	必要なスキル
クラウドプロキシをデプロイします。	Splunk のオンプレミスインスタンスにログを転送するには、VMware Aria Operations for Logs のクラウドプロキシを追加する必要があります。このプロキシは、オンプレミスのデータセンターから情報を受け取り、分析のため	クラウド管理者、クラウドアーキテクト

タスク	説明	必要なスキル
	に VMware Aria Operations for Logs に送信します。 クラウドプロキシをダウンロードしてインストールするには： 1. ポート 443、22、514 がオンプレミス環境と VMware Cloud on の間で開いていることを確認します AWS。追加のポートには、1514/TCP または 6514/TCP を使用できます。ポートの詳細については、VMware ドキュメントの「VMware Aria Operations for Logs Firewall Recommendations」を参照してください。 VMware 2. VMware Aria Operations for Logs にログインします。 3. ホームページで、ウィジェットでコレクターの追加を選択します。 4. クラウドプロキシ仮想アプライアンス画面で、トークンキーをコピーします。次のステップを完了するには、24 時間以内にこのキーを使用する必要があります。 5. OVA ファイルのダウンロードリンクを選択します。	

タスク	説明	必要なスキル
	6. VMware vSphere ウェブクライアントに移動し、クラスターを選択し、OVF テンプレートのデプロイを選択します。 7. キーの入力を求められたら、ステップ 4 でコピーしたトークンキーを貼り付けます。 8. 終了 を選択してクラウドプロキシをインストールします。	

オンプレミスの Splunk エンドポイントにログを転送する

タスク	説明	必要なスキル
ログ転送を設定します。	Splunk エンドポイントにログを転送するには： 1. VMware Aria Operations for Logs にログインします。 2. ログ管理に移動します。 3. ログ転送を選択します。 4. 新しい設定を選択し、次の設定を完了します。 • ログ転送設定の名前を指定します。 • 送信先 で、オンプレミスを選択します。 • Cloud Proxy で、前にインストールしたクラウドプロキシを選択します。	

タスク	説明	必要なスキル
	• エンドポイントタイプで、TCP を選択します。 • エンドポイント URL には、オンプレミスの Splunk URL を次の形式で指定します。 `tcp://x.x.x.x (your Splunk IP address): 514` • (オプション) タグでは、クエリを容易にするためにタグ名と値を指定できます。 • すべてのログに適用するまたは特定のログに適用する を選択します。すべての VMware Cloud on AWS ログを Splunk に送信する場合は、すべてのログに適用する を選択します。 5. 確認を選択します。 6. [Save] を選択します。 詳細については、VMware ドキュメントの「Forward Logs from VMware Aria Operations for Logs」を参照してください。 VMware	

関連リソース

- VMware Cloud on AWS ウェブサイト
- ソフトウェア定義データセンターについて (VMware ドキュメント)
- VMware Cloud on AWS を使用して に VMware SDDC をデプロイする AWS (AWS 規範ガイダンス)
- VMware HCX AWS を使用してワークロードを VMware Cloud on に移行する (AWS 規範ガイダンス)
- Hybrid Linked Mode AWS を使用して VMware Cloud on へのデータセンター拡張を設定する (AWS 規範ガイダンス)

AWS CDK と GitLab を使用して、Amazon ECS Anywhere 上のハイブリッドワークロード用に CI/CD パイプラインを設定する

作成者：Dr. Rahul Sharad Gaikwad (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら。

Amazon ECS Anywhere は、Amazon Elastic Container Service (Amazon ECS) の拡張機能です。オンプレミスサーバーや仮想マシン (VM) などの外部インスタンスを Amazon ECS クラスターに登録するためのサポートを提供します。この機能は、コストを削減し、ローカルコンテナの複雑なオーケストレーションやオペレーションを軽減します。ECS Anywhere を使用して、オンプレミス環境とクラウド環境の両方でコンテナアプリケーションをデプロイして実行できます。これにより、チームが複数のドメインやスキルセットを習得したり、複雑なソフトウェアを独自に管理したりする必要がなくなります。

このパターンは、Amazon Web Services (AWS) Cloud Development Kit (AWS CDK) スタックを使用して Amazon ECS Anywhere インスタンスで Amazon ECS クラスターをプロビジョニングする段階的なアプローチを説明しています。 次に、AWS CodePipeline を使用して継続的な統合と継続的なデプロイ (CI/CD) パイプラインを設定します。次に、GitLab コードリポジトリを AWS CodeCommit に複製し、コンテナ化されたアプリケーションを Amazon ECS クラスターにデプロイします。

このパターンは、オンプレミスインフラストラクチャを使用してコンテナアプリケーションを実行したり、GitLab を使用してアプリケーションコードベースを管理するユーザーに役立つように設計されています。これらのワークロードは、既存のオンプレミスインフラストラクチャに影響を与えずに、AWS クラウドサービスで管理できます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- オンプレミスインフラストラクチャ上で実行されるコンテナアプリケーション。
- アプリケーションコードベースを管理する GitLab リポジトリ。 詳細については、「リポジトリ」 (GitLab) を参照してください。

- AWS コマンドラインインターフェイス (AWS CLI) をインストールして設定済み。詳細については、「AWS CLI の最新バージョンをインストールまたはアップデート」(AWS CLIのドキュメント) を参照してください。
- AWS CDK ツールキットがインストール済みおよびグローバルに設定済み 詳細については、「AWS CDK をインストールする」(AWS CDK ドキュメント) を参照してください。
- npm、TypeScript で AWS CDK 用にインストールおよび設定されています。詳細については、「Node.js と npm のダウンロードとインストール」(npm ドキュメント) を参照してください。

機能制限

- 制限と考慮事項については、Amazon ECS ドキュメントの「外部インスタンス (Amazon ECS Anywhere)」を参照してください。

製品バージョン

- AWS CDK ツールキットバージョン 2.27.0 以降
- npm バージョン 7.20.3 以降
- Node.js バージョン 16.6.1 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS CDK
- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- Amazon ECS Anywhere
- Amazon Elastic Container Registry (Amazon ECR)
- AWS Identity and Access Management (IAM)
- AWS Systems Manager
- GitHub リポジトリ

ターゲット アーキテクチャ

この図は、このパターンで説明されている 2 つの主なワークフロー、つまり Amazon ECS クラスターのプロビジョニングと、CI/CD パイプラインをセットアップしてデプロイする CI/CD パイプラインの設定を以下のように表しています。

1. Amazon ECS クラスターのプロビジョニング
   a. AWS CDK スタックをデプロイする場合、AWS に CloudFormation スタックが作成されます。
   b. この CloudFormation スタックは Amazon ECS クラスターと関連する AWS リソースをプロビジョニングします。
   c. Amazon ECS クラスターに外部インスタンスを登録するには、仮想マシン (VM) に AWS Systems Manager Agent (SSM Agent) をインストールし、その VM を AWS Systems Manager 管理型のインスタンスとして登録する必要があります。
   d. また、Amazon ECS コンテナエージェントと Docker を VM にインストールして、Amazon ECS クラスターの外部インスタンスとして登録する必要があります。
   e. 外部インスタンスを Amazon ECS クラスターに登録して設定すると、外部インスタンスとして登録された VM 上で複数のコンテナを実行できます。
   f. Amazon ECS クラスターはアクティブで、コンテナを介してアプリケーションワークロードを実行できます。 Amazon ECS Anywhere コンテナインスタンスはオンプレミス環境で実行されますが、クラウドの Amazon ECS クラスターに関連付けられています。
2. CI/CD パイプラインのセットアップとデプロイ
   a. 2 つ目の AWS CDK スタックをデプロイする場合、AWS に CloudFormation スタックが作成されます。
   b. この CloudFormation スタックは、CodePipeline および関連のある AWS リソースにパイプラインをプロビジョニングします。
   c. アプリケーションコードの変更をオンプレミスの GitLab リポジトリにプッシュしてマージします。
   d. GitLab リポジトリは CodeCommit リポジトリに自動的に複製されます。
   e. CodeCommit リポジトリを更新すると、CodePipeline が自動的に起動します。
   f. CodePipeline は CodeCommit からコードをコピーし、CodeBuild にデプロイ可能なアプリケーションビルドを作成します。
   g. CodePipeline は CodeBuild ビルド環境の Docker イメージを作成し、Amazon ECR リポジトリにプッシュします。

h. CodePipeline は、Amazon ECR リポジトリからコンテナイメージをプルする CodeDeploy アクションを開始します。

i. CodePipeline は Amazon ECS クラスターにコンテナイメージをデプロイします。

自動化とスケール

このパターンでは、AWS CDK をコードとしての Infrastructure as Code (IaC) ツールとして使用して、このアーキテクチャを設定してデプロイします。AWS CDK は、AWS リソースのオーケストレーションと Amazon ECS Anywhere と CI/CD パイプラインのセットアップに役立ちます。

ツール

AWS サービス

- AWS Cloud Development Kit (AWS CDK) は、AWS クラウドインフラストラクチャをコードで定義してプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CodeCommit は、独自のソースコントロールシステムを管理しなくても、Git リポジトリを非公開で保存および管理できるバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまな段階を迅速にモデル化および設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ、信頼性を備えたマネージドコンテナイメージのレジストリサービスです。
- Amazon Elastic Container Service (Amazon ECS) は、クラスターでコンテナの実行、停止、管理を支援する、高速でスケーラブルなコンテナ管理サービスです。このパターンは、オンプレミスサーバーまたは VM を Amazon ECS クラスターに登録するためのサポートを提供する Amazon ECS Anywhere も使用します。

その他のツール

- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- npm は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。

- Vagrant は、ポータブルな仮想ソフトウェア開発環境を構築して保守するためのオープンソースユーティリティです。デモンストレーション用に、このパターンでは Vagrant を使用してオンプレミスの VM を作成します。

コードリポジトリ

このパターンのコードは、GitHub 内の「AWS CDK を使用した Amazon ECS Anywhere の CI/CD パイプライン」リポジトリで利用できます。

ベストプラクティス

このパターンをデプロイする場合は、以下のベストプラクティスを考慮してください。

- 「AWS CDK でクラウドインフラストラクチャを開発およびデプロイするためのベストプラクティス」
- 「AWS CDK でクラウドアプリケーションを開発するためのベストプラクティス」(AWS ブログ記事)

エピック

AWS CDK の設定を検証する

タスク	説明	必要なスキル
AWS CDK のバージョンを検証します。	次のコマンドを実行して、AWS CDK Toolkit のバージョンを検証します。 `cdk --version` このパターンには、バージョン 2.27.0 以降が必要です。以前のバージョンを使用している場合は、「AWS CDK ドキュメント」の指示に従って更新してください。	DevOps エンジニア

タスク	説明	必要なスキル
npm バージョンを検証します。	次のコマンドを実行して、npm のバージョンを検証します。 `npm --version` このパターンには、バージョン 7.20.3 以降が必要です。以前のバージョンを使用している場合は、「npm ドキュメント」の指示に従って更新してください。	DevOps エンジニア

AWS 認証情報を設定します。	認証情報を設定するには、aws configure のコマンドを実行し、プロンプトに従ってください。
	`$aws configure`
	`AWS Access Key ID [None]: <your-access-key-ID>`
	`AWS Secret Access Key [None]: <your-secret-access-key>`
	`Default region name [None]: <your-Region-name>`
	`Default output format [None]:`
		DevOps エンジニア

AWS CDK 環境のブートストラップ

タスク	説明	必要なスキル

AWS SAM コードリポジトリを複製します。	1. 次のコマンドを実行することで、AWS CDK リポジトリを使用して、このパターンの Amazon ECS Anywhere の CI/CD パイプラインのクローンを作成します。
	`git clone https://github.com/aws-samples/amazon-ecs-anywhere-cicd-pipeline-cdk-sample.git`
	2. 次のコマンドを実行して、クローンされたディレクトリに移動します。
	`cd amazon-ecs-anywhere-cicd-pipeline-cdk-sample`
		DevOps エンジニア

環境を起動します。	次ののコマンドを実行して、使用したいアカウントと AWS リージョンに AWS CloudFormation テンプレートをデプロイします。 `cdk bootstrap <account-number>/<Region>` 詳細については、AWS CDK ドキュメントの「ブートスト	DevOps エンジニア

タスク	説明	必要なスキル
	ラップ」を参照してください。	

Amazon ECS Anywhere 向けインフラストラクチャの構築とデプロイ

タスク	説明	必要なスキル

パッケージの依存関係をインストールし、TypeScript ファイルをコンパイルします。	パッケージの依存関係をインストールし、次のコマンドを実行して TypeScript ファイルをコンパイルします。
	`$cd EcsAnywhereCdk`
	`$npm install`
	`$npm fund`
	これらのコマンドは、すべてのパッケージをサンプルリポジトリからインストールします。詳細については、npm ドキュメントの「npm ci」と「npm install」を参照してください。これらのコマンドを入力したときに、紛失したパッケージに関するエラーが表示された場合は、このパターンの[トラブルシューティング]のセクションを参照してください。
		DevOps エンジニア

プロジェクトをビルドします。	プロジェクトコードをビルドするには、以下のコマンドを入力します。 `npm run build`	DevOps エンジニア

タスク	説明	必要なスキル
	プロジェクトの構築とデプロイの詳細については、AWS CDK ドキュメントの「初めての AWS CDK アプリケーション」を参照してください。	

Amazon ECS Anywhere 向けインフラストラクチャスタックをデプロイします	1. 次のコマンドを実行して、スタックを一覧表示します。
	`$cdk list`
	2. 出力が `EcsAnywhereInfraStack` スタックと `ECSAnywherePipelineStack` スタックを返すことを確認します。
	3. 次のコマンドを実行して、`EcsAnywhereInfraStack` スタックをデプロイします。
	`$cdk deploy EcsAnywhereInfraStack`
		DevOps エンジニア

タスク	説明	必要なスキル
スタックの作成と出力を検証します。	1. AWS マネジメントコンソールにサインインした後、CloudFormation コンソール (「https://console.aws.amazon.com/cloudformation/」) を開きます。 2. [スタック] ページで、`EcsAnywhereInfraStack` スタックを選択します。 3. スタックの状態が `CREATE_IN_PROGRESS` または `CREATE_COMPLETE` であることを確認します。 Amazon ECS クラスターの設定に時間を要する場合があります。スタックの作成が完了するまで、先に進まないでください。	DevOps エンジニア

オンプレミス VM の設定

タスク	説明	必要なスキル
VM をセットアップします	Vagrantfile が配置されているルートディレクトリから `vagrant up` コマンドを実行して Vagrant VM を作成します。詳細については、	DevOps エンジニア

タスク	説明	必要なスキル
	「Vagrant ドキュメント」を参照してください。	

タスク	説明	必要なスキル
VM を外部インスタンスとして登録します。	1. `vagrant ssh` コマンドを使用して Vagrant VM にログインします。詳細については、「Vagrant ドキュメント」を参照してください。 2. 「AWS CLIのインストール手順」に従い、次のコマンドを実行して仮想マシン上で AWS CLI を停止します。 `$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" \ > -o "awscliv2.zip" $sudo apt install unzip $unzip awscliv2.zip $sudo ./aws/install $aws configure AWS Access Key ID [None]: <your-access-key-ID> AWS Secret Access Key [None]: <your-secret-access-key> Default region name [None]: <your-Region-name> Default output format [None]:` 1. VM を AWS Systems Manager に登録するか、または外部インスタンスを	DevOps エンジニア

タスク	説明	必要なスキル

	アクティブ化するために使用できるアクティベーションコードと ID を作成します。このコマンドの出力には、アクティベーション ID とアクティベーションコード値が含まれます。
	
	`aws ssm create-activation \ > --iam-role EcsAnywhereInstanceRole \ > | tee ssm-activation.json`
	
	このコマンドを実行したときにエラーが発生した場合は、「トラブルシューティング」セクションを参照してください。
	2. アクティベーション ID とコード値を出力します。
	
	`export ACTIVATION_ID=<activation-ID> export ACTIVATION_CODE=<activation-code>`
	
	3. インストールスクリプトを仮想マシンにダウンロードします。	

タスク	説明	必要なスキル
	``` curl --proto "https" - o "ecs-anywhere-inst all.sh" \ > "https://amazon-ec s-agent.s3.amazona ws.com/ecs-anywher e-install-latest.s h" ``` 4. インストールスクリプトを実行します。 ``` sudo bash ecs-anywh ere-install.sh \ --cluster EcsAnywhe reCluster \ --activation-id $ACTIVATION_ID \ --activation-code $ACTIVATION_CODE \ --region <region-n ame> ``` これにより、VM が Amazon ECS Anywhere の外部インスタンスとして設定され、Amazon ECS クラスターに登録されます。詳細については、Amazon ECS のドキュメントの「クラスターへの外部インスタンスの登録」を参照してください。問題が発生した場合は、「トラブルシューティング」セクションを参照してください。	

タスク	説明	必要なスキル

Amazon ECS Anywhere と外部 VM のステータスを検証してください。	VM が Amazon ECS コントロールプレーンに接続され、稼働していることを検証するには、以下のコマンドを使用します。
	`$aws ssm describe-instance-information`
	`$aws ecs list-container-instances --cluster $CLUSTER_NAME`
		DevOps エンジニア

CI/CD パイプラインをデプロイする

タスク	説明	必要なスキル

AWS CodeCommit リポジトリにブランチを作成します。	最初のリポジトリのコミットを作成して、CodeCommit リポジトリに `main` というブランチを作成します。AWS のドキュメントに従って、CodeCommit にコミットを作成することができます。コマンドの例を次に示します。
	`aws codecommit put-file \`
	`--repository-name EcsAnywhereRepo \`
	`--branch-name main \`
	`--file-path README.md \`
	`--file-content "Test" \`
	`--name "Dev Ops" \`
		DevOps エンジニア

タスク	説明	必要なスキル

	<pre>--email "devops@example.com" \
	--commit-message "Adding README."</pre>	

リポジトリのミラーリングを設定します。	GitLab リポジトリを外部ソースとの間でミラーリングできます。 ソースとして使用するリポジトリを選択できます。ブランチ、タグ、コミットは自動的に同期されます。アプリケーションをホストする GitLab リポジトリと CodeCommit リポジトリの間でプッシュミラーを設定します。手順については、「GitLab から CodeCommit へのプッシュミラーの設定」（GitLab ドキュメント）を参照してください。 **Note** デフォルトでは、ミラーリングはリポジトリを自動的に同期します。リポジトリを手動で更新する場合は、「ミラーの更新」(GitLab ドキュメント) を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル
CI/CD パイプラインスタックをデプロイする	次のコマンドを実行して、EcsAnywherePipelineStack スタックをデプロイします。 `$cdk deploy EcsAnywherePipelineStack`	DevOps エンジニア

タスク	説明	必要なスキル
CI/CD パイプラインをテストします。	1. アプリケーションコードを変更し、ソースコード、オンプレミス GitLab リポジトリにプッシュします。詳細については、「プッシュ オプション」(GitLab ドキュメント) を参照してください。たとえば、`../application/index.html` ファイルを編集してアプリケーションのバージョン値を更新します。 2. これにより、コードが CodeCommit リポジトリにコピーされると、CI/CD コンジットが開始されます。次のいずれかを行います : • 自動ミラーリングを使用して GitLab リポジトリと CodeCommit リポジトリを同期している場合は、次の手順に進んでください。 • 手動ミラーリングを使用している場合は、「ミラーの更新」(GitLab ドキュメント) の指示に従って、アプリケーションコードの変更を CodeCommit リポジトリにプッシュします。 3. ローカルマシンのウェブブラウザーで http://localhost:80 と入力します。ポー	DevOps エンジニア

タスク	説明	必要なスキル
	ト 80 が Vagrantfile の localhost に転送されるので、NGINX のウェブページが開きます。更新されたアプリケーションバージョン値が表示されることを確認します。これにより、パイプラインとイメージのデプロイが検証されます。 4. (オプション) AWS マネジメントコンソールでデプロイを確認する場合は、次の手順を実行します。 a. Amazon ECS コンソール (「https://console.aws.amazon.com/ecs/」) を開きます。 b. ナビゲーションバーから、使用するリージョンを選択します。 c. ナビゲーションペインで [クラスター] を選択します。 d. [クラスター] ページで、EcsAnywhereCluster を選択します。 e. [タスク定義] を選択します。 f. コンテナが稼働していることを確認します。	

クリーンアップ

タスク	説明	必要なスキル

リソースをクリーンアップして削除します。	このパターンが完了したら、作成した概念実証リソースを削除する必要があります。クリーンアップするには、次のコマンドを実行します。
	`$cdk destroy EcsAnywherePipelineStack`
	`$cdk destroy EcsAnywhereInfraStack`
		DevOps エンジニア

トラブルシューティング

問題	ソリューション

パッケージ依存関係のインストール中にパッケージが見つからない場合のエラー。	次のコマンドのいずれかを実行して、見つからないパッケージを解析します。
	`$npm ci`
	or
	`$npm install -g @aws-cdk/<package_name>`

VM 上で `aws ssm create-activation` コマンドを実行すると、次のエラーが発生します。 `An error occurred (ValidationException) when calling the CreateActivation operation: Nonexistent role or missing ssm`	`EcsAnywhereInfraStack` スタックは完全にデプロイされておらず、このコマンドを実行するために必要な IAM ロールもまだ作成されていません。CloudFormation コンソールでスタックの状態を確認します。ステータスが `CREATE_COMPLETE` に変更したら、コマンドを再試行します。

問題	ソリューション
`service principal in trust policy: arn:aws:iam::000000000000:role/EcsAnywhereInstanceRole`	

Amazon ECS ヘルスチェックで UNHEALTHY が返され、Amazon ECS コンソール内でクラスターの [サービス] セクションに次のエラーが表示されます。
 	
`service EcsAnywhereService was unable to place a task because no container instance met all of its requirements. Reason: No Container Instances were found in your cluster.`
	次のコマンドを実行して、Vagrant VM の Amazon ECS エージェントを再起動します。
	
	`$vagrant ssh`
	`$sudo systemctl restart ecs`
	`$sudo systemctl status ecs`

関連リソース

- 「Amazon ECS Anywhere マーケティングページ」
- 「Amazon ECS Anywhere のドキュメント」
- 「Amazon ECS Anywhere デモ」(ビデオ)
- 「Amazon ECS Anywhere ワークショップサンプル」
- 「リポジトリ ミラーリング」(GitLab ドキュメント)

その他のパターン

- AWS Transit Gateway によるリージョン間ピアリングの設定を自動化する
- AWS CDK で Amazon ECS Anywhere を設定して、オンプレミスコンテナアプリケーションを管理します。
- WANdisco LiveData Migrator を使用して Hadoop データを Amazon S3 に移行する
- PowerCLI を使用して HCX オートメーションで VMware 仮想マシンを移行
- VMware HCX を使用して VMware Cloud on AWS ワークロードを移行
- F5 から AWS のApplication Load Balancer に移行するときの HTTP ヘッダーを変更
- AWS Cloud でオンプレミスワークロードをリホストする: 移行チェックリスト
- BMC ディスカバリークエリを使用して移行計画のために移行データを抽出

管理とガバナンス

トピック

- Amazon Data Firehose リソースが AWS KMS キーで暗号化されていない場合の識別とアラート
- AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する
- Python を使用して RFC を自動的に作成する
- AWS Systems Manager メンテナンスウィンドウを使用して Amazon RDS DB インスタンスを自動的に停止および起動する
- Terraform を使用して AWS Organizations でソフトウェアパッケージの配布を一元化する
- NLog を使用して Amazon CloudWatch Logs 内の.NET アプリケーションのロギングを設定します
- AWS Service Catalog 製品を異なる AWS アカウントと AWS リージョンにコピー
- クラウド運用モデルの RACI または RASCI マトリックスを作成
- Amazon CloudWatch 異常検知により、カスタムメトリクスのアラームを作成
- デフォルトの暗号化で Amazon EBS ボリュームを使用する AWS Cloud9 IDE を作成
- タグベースの Amazon CloudWatch ダッシュボードを自動的に作成する
- AWS ランディングゾーンの設計を文書化する
- AWS CDK を使用して複数の AWS リージョン、アカウント、OU にわたって Amazon DevOps Guru を有効にすることで、運用パフォーマンスを向上させます。
- ブートストラップパイプラインを使用して Account Factory for Terraform (AFT) を実装する
- 複数の AWS アカウントと AWS リージョンで AWS Service Catalog 製品を管理
- AWS メンバーアカウントを AWS Organizations から AWS Control Tower に移行する
- AWS のサービスを使用して SAP RHEL Pacemaker クラスターをモニタリングする
- 複数の で共有 Amazon マシンイメージの使用をモニタリングする AWS アカウント
- AWS Organizations のプログラムによるアカウント閉鎖のアラートを設定する
- AWS アカウントまたは組織の EBS スナップショットの詳細を表示
- その他のパターン

Amazon Data Firehose リソースが AWS KMS キーで暗号化されていない場合の識別とアラート

作成者: Ram Kandaswamy (AWS)

概要

コンプライアンスのために、一部の組織では Amazon Data Firehose などのデータ配信リソースで暗号化を有効にする必要があります。このパターンは、リソースがコンプライアンス違反になったときに監視、検出、通知する方法を示しています。

暗号化要件を維持するために、このパターンを で使用 AWS して、 AWS Key Management Service (AWS KMS) キーで暗号化されていない Amazon Data Firehose 配信リソースの自動モニタリングと検出を提供できます。このソリューションはアラート通知を送信し、拡張して自動修復を実行することもできます。このソリューションは、ラン AWS ディングゾーンまたは を使用する環境など、個々のアカウントまたはマルチアカウント環境に適用できます AWS Control Tower。

前提条件と制限

前提条件

- Amazon Data Firehose 配信ストリーム
- このインフラストラクチャの自動化 AWS CloudFormationで使用される十分なアクセス許可と への精通

機能制限

- このソリューションは、検出に AWS CloudTrail イベントを使用するため、リアルタイムではなく、暗号化されていないリソースが作成されてから通知が送信されるまでに遅延が生じます。

アーキテクチャ

ターゲットテクノロジースタック

このソリューションはサーバーレステクノロジーと以下のサービスを使用します。

- AWS CloudTrail
- Amazon CloudWatch
- AWS Command Line Interface (AWS CLI)

- AWS Identity and Access Management (IAM)
- Amazon Data Firehose
- AWS Lambda
- Amazon Simple Notification Service (Amazon SNS)

ターゲット アーキテクチャ

この図は、これらのステップを示しています。

1. ユーザーは Amazon Data Firehose を作成または変更します。
2. CloudTrail イベントが検出され、照合されます。
3. AWS Lambda が呼び出されます。
4. 非準拠のリソースが特定されます。
5. メールが送信されます。

自動化とスケール

AWS CloudFormation StackSets を使用して、1 つのコマンドで複数の AWS リージョン または アカウントにこのソリューションを適用できます。

ツール

- AWS CloudTrail は、 のガバナンス、コンプライアンス、運用およびリスク監査を可能にする AWS のサービス です AWS アカウント。ユーザー、ロール、または によって実行されたアクション AWS のサービス は、イベントとして CloudTrail に記録されます。イベントには AWS Management Console、、 AWS CLI、 AWS SDKs、および API オペレーションで実行されたアクションが含まれます。
- Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルでコマンド AWS のサービス を使用して とやり取りできるオープンソースツールです。
- AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。

- Amazon Data Firehose は、リアルタイムのストリーミングデータを配信するためのフルマネージドサービスです。Firehose では、アプリケーションを記述したり、リソースを管理したりする必要はありません。Firehose にデータを送信するデータプロデューサーを作成すると、それにより、指定した送信先にデータが自動配信されます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートするコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。料金は消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合は料金は発生しません。
- Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバー (プロデューサーおよびコンシューマーとも呼ばれます) へのメッセージ配信を提供するマネージドサービスです。

エピック

コンプライアンスのために暗号化を強制

タスク	説明	必要なスキル
AWS CloudFormation StackSets をデプロイします。	で AWS CLI、`firehose-encryption-checker.yaml` テンプレート (添付) を使用して、次のコマンドを実行してスタックセットを作成します。 パラメータには有効な Amazon SNS トピック Amazon リソースネーム (ARN) を指定します。デプロイでは、 テンプレートで説明されているように、CloudWatch Events ルール、Lambda 関数、および必要なアクセス許可を持つ IAM ロールが正常に作成されている必要があります。	クラウドアーキテクト、システム管理者

タスク	説明	必要なスキル
	`aws cloudformation create-stack-set --stack-set-name my-stack-set --template-body file://firehose-encryption-checker.yaml`	
がスタックインスタンスを作成する。	スタックは、 AWS リージョン 選択した および 1 つ以上のアカウントで作成できます。 スタックインスタンスを作成するには、次のコマンドを実行します。スタック名、アカウント番号、リージョンを独自の に置き換えます。 `aws cloudformation create-stack-instances --stack-set-name my-stack-set --accounts 123456789012 223456789012 --regions us-east-1 us-east-2 us-west-1 us-west-2 --operation-preferences FailureToleranceCount=1`	クラウドアーキテクト、システム管理者

関連リソース

- AWS CloudFormation StackSets の使用
- Amazon CloudWatch Events とは

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する

作成者: Appasaheb Bagali (AWS)

概要

AWS Systems Manager は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのリモート管理ツールです。Systems Manager は、Amazon Web Services のインフラストラクチャの可視性と制御を提供します。この汎用性の高いツールを使用すると、セキュリティ脆弱性スキャンレポートで脆弱性と特定された Windows レジストリの変更を修正できます。

このパターンでは、環境の安全のために推奨されるレジストリの変更を自動化することで、Windows オペレーティングシステムを実行している EC2 インスタンスを安全に保つ手順について説明します。このパターンでは Run コマンドを使用してコマンドドキュメントを実行します。コードは添付されており、その一部はコードセクションに含まれます。

前提条件と制限

- アクティブなAWS アカウント
- EC2 インスタンスと Systems Manager にアクセスする権限

アーキテクチャ

ターゲットテクノロジースタック

- 2 つのサブネットとネットワークアドレス変換 (NAT) ゲートウェイがある仮想プライベートクラウド (VPC)
- レジストリ名と値を追加または更新する Systems Manager Command ドキュメント
- 指定された EC2 インスタンス上で コマンドドキュメントを実行する Systems Manager Run コマンド

ターゲット アーキテクチャ

ツール

ツール

- IAM ポリシーとロール – AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスのセキュアな制御に役立つ Web サービスです。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- Amazon Simple Storage Service — Amazon Simple Storage Service (Amazon S3) は、インターネット用のストレージです。Web スケールのコンピューティングを開発者が容易にできるように設計されています。このパターンでは、S3 バケットを使用して Systems Manager ログを保存します。
- AWS Systems Manager – AWS Systems Manager は、 AWS でインフラストラクチャの表示と制御に使用できる AWS サービスです。Systems Manager は、マネージドインスタンスをスキャンし、検出されるポリシー違反を報告 (または是正措置を講じる) して、セキュリティとコンプライアンスを維持できます。
- AWS Systems Manager Command ドキュメント — AWS Systems Manager コマンドドキュメントは Run Command により使用されます。ほとんどのコマンドドキュメントは、Systems Manager でサポートされているすべての Linux および Windows Server オペレーティングシステムでサポートされています。
- AWS Systems Manager Run Command — AWS Systems Manager Run Command では、マネージドインスタンスの設定を安全にリモートで管理する方法を得られます。Run Command を使用すると、一般的な管理タスクを自動化し、一度限りの設定変更を大規模に実行できます。

コード

次のコード例を使用して、Microsoft Windows レジストリ名を `Version` に、レジストリパスを `HKCU:\Software\ScriptingGuys\Scripts` に、値を 2 に追加または更新できます。

```
#Windows registry path which needs to add/update
$registryPath ='HKCU:\\Software\\ScriptingGuys\\Scripts'
#Windows registry Name  which needs to add/update
$Name = 'Version'
#Windows registry value  which needs to add/update
$value = 2
# Test-Path cmdlet to see if the registry key exists.
IF(!(Test-Path $registryPath))
        {
           New-Item -Path $registryPath -Force | Out-Null
           New-ItemProperty -Path $registryPath -Name $name -Value     $value ` -
PropertyType DWORD -                 Force | Out-        Null
        } ELSE {
```

```
                    New-ItemProperty -Path $registryPath -Name $name -Value $value `
 -PropertyType            DWORD        -Force | Out-Null
           }
echo 'Registry Path:'$registryPath
 echo 'Registry Name:'$registryPath
 echo 'Registry Value:'(Get-ItemProperty -Path $registryPath -Name $Name).version
```

完全な Systems Manager コマンドドキュメントの JavaScript Object Notation (JSON) コード例が添付されています。

エピック

VPC をセットアップする

タスク	説明	必要なスキル
VPC を作成します。	AWS マネジメントコンソールで、パブリックおよびプライベートのサブネットと NAT ゲートウェイがある VPC を作成します。詳細については、AWS ドキュメントを参照してください。	クラウド管理者
セキュリティグループを作成します。	各セキュリティグループが、ソース IP アドレスからのリモートデスクトッププロトコル (RDP) へのアクセスを許可していることを確認します。	クラウド管理者

IAM ポリシーと IMA ロールを作成する

タスク	説明	必要なスキル
IAM ポリシーを作成します。	Amazon S3、Amazon EC2、および Systems Manager へのアクセスを提供する IAM ポリシーを作成します。	クラウド管理者

タスク	説明	必要なスキル
IAM ロールを作成します。	IAM ロールを作成して、Amazon S3、Amazon EC2、および Systems Manager へのアクセスを提供する IAM ポリシーをアタッチします。	クラウド管理者

自動化を実行する

タスク	説明	必要なスキル
Systems Manager コマンドドキュメントを作成します。	Microsoft Windows レジストリの変更をデプロイして追加または更新する Systems Manager コマンドドキュメントを作成します。	クラウド管理者
Systems Manager Run Command を実行します。	Systems Manager Run Command を実行し、コマンドドキュメントと Systems Manager ターゲットインスタンスを選択します。これにより、選択したコマンドドキュメント内の Microsoft Windows レジストリの変更がターゲットインスタンスにプッシュされます。	クラウド管理者

関連リソース

- AWS Systems Manager
- AWS Systems Manager ドキュメント
- AWS Systems Manager Run Command

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Python を使用して RFC を自動的に作成する

作成者: Gnanasekaran Kailasam (AWS)

概要

AWS Managed Services (AMS) は、Amazon Web Services (AWS) インフラストラクチャを継続的に管理することで、クラウドベースのインフラストラクチャをより効率的かつ安全に運用できるよう支援します。管理環境に変更するには、特定のオペレーションまたはアクションの変更タイプ (CT) ID を含む新しい変更リクエスト (RFC) を作成して送信する必要があります。

ただし、RFC を手動で作成するには約 5 分かかる場合があり、組織内のチームは毎日複数の RFC を提出する必要がある場合があります。このパターンは、RFC 作成プロセスを自動化し、各 RFC の作成時間を短縮し、手作業によるエラーを排除する上で役立ちます。

このパターンでは、Python コードを使用して、AMS アカウントで Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを停止する `Stop EC2 instance` RFCを自動的に作成する方法を説明します。その後、このパターンのアプローチと Python オートメーションを他の RFC タイプに適用できます。

前提条件と制限

前提条件

- AMS 詳細アカウント。詳細については、AWS Managed Services ドキュメントの「AMS オペレーションプラン」を参照してください。
- AMS アカウントの 1 つ以上の既存の EC2 インスタンス。
- AMS で RFC を作成して提出する方法の理解。
- Python に精通していること。

制限

- RFC は AMS アカウントの変更にのみ使用できます。AWS アカウントは、同様の変更するためにさまざまなプロセスを使用します。

アーキテクチャ

テクノロジースタック

- AMS
- AWS コマンドラインインターフェイス (AWS CLI)
- 「AWS SDK for Python (Boto3)」
- Python とそれに必要なパッケージ (JSON および Boto3)

自動化とスケール

このパターンは `Stop EC2 instance` RFC を自動化するためのサンプルコードを提供しますが、このパターンのサンプルコードとアプローチを他の RFC にも使用できます。

ツール

- AWS Managed Services – AMS は AWS インフラストラクチャをより効率的かつ安全に運用する上で役立ちます。
- AWS CLI – AWS コマンドラインインターフェイス (AWS CLI) は、AWS のサービスを管理するための統合ツールです。AMS では、変更管理 API が RFC を作成および管理するための操作を提供します。
- AWS SDK for Python (Boto3) — Python 用 SDK を使用すると、Python アプリケーション、ライブラリ、またはスクリプトを AWS のサービスと簡単に統合できます。

コード

`AMS Stop EC2 Instance.zip` ファイル (添付) には、`Stop EC2 instance` RFC を作成するための Python コードが含まれています。複数の EC2 インスタンスに対して 1 つの RFC を送信するようにこのコードを設定することもできます。

エピック

オプション 1 — macOS または Linux 用の環境をセットアップする

タスク	説明	必要なスキル
Python をインストールして検証します。	1. ターミナルウィンドウを開き、`brew install python3` コマンドを実行します。	AWS システム管理者

タスク	説明	必要なスキル
	2. `python --version` コマンドを実行して Python が正しくインストールされていることを確認します。 3. `pip --version` コマンドを実行して、pip が正しくインストールされていることを確認します。	
AWS CLI をインストールします。	`pip install awscli --upgrade –user` コマンドを実行して、AWS CLI をインストールします。	AWS システム管理者
Boto3 をインストールします。	`pip install boto3` コマンドを実行して、Boto3 をインストールします。	AWS システム管理者
JSON をインストールします。	`pip install json` コマンドを実行して、JSON をインストールします。	AWS システム管理者

タスク	説明	必要なスキル
AMS CLI をセットアップします。	AWS マネジメントコンソールにサインインし、AMS コンソールを開いてから、[Documentation (ドキュメント)] を選択します。AMS CLI を含む.zip ファイルをダウンロードし、解凍してから、ローカルマシンにインストールします。 AMS CLI をインストールしたら、 `aws amscm help` コマンドを実行します。出力では、AMS 変更管理プロセスに関する情報が提供されます。	AWS システム管理者

オプション 2 — Windows 用の環境をセットアップする

タスク	説明	必要なスキル
Python をインストールして検証します。	1. Windows 用の Python リリースページを開き、最新バージョンをダウンロードしてから、Python をインストールします。 2. `python --version` コマンドを実行して Python が正しくインストールされていることを確認します。 3. `pip --version` コマンドを実行して、pip が正しくインストールされていることを確認します。	AWS システム管理者

タスク	説明	必要なスキル
AWS CLI をインストールします。	`pip install awscli --upgrade –user` コマンドを実行して、AWS CLI をインストールします。	AWS システム管理者
Boto3 をインストールします。	`pip install boto3` コマンドを実行して、Boto3 をインストールします。	AWS システム管理者
JSON をインストールします。	`pip install json` コマンドを実行して、JSON をインストールします。	AWS システム管理者
AMS CLI をセットアップします。	AWS マネジメントコンソールにサインインし、AMS コンソールを開いてから、[Documentation (ドキュメント)] を選択します。AMS CLI を含む.zip ファイルをダウンロードし、解凍してから、ローカルマシンにインストールします。 AMS CLI をインストールしたら、 `aws amscm help` コマンドを実行します。出力では、AMS 変更管理プロセスに関する情報が提供されます。	AWS システム管理者

RFC の CT ID と実行パラメータを抽出する

タスク	説明	必要なスキル
RFC の CT ID、バージョン、実行パラメータを抽出します。	各 RFC には異なる CT ID、バージョン、実行パラメーターがあります。この情報は、次のいずれかのオプションを使用して抽出できます。 1. AWS Managed Services ドキュメントの「RFC 使用例」のCLI による変更要求 (RFC) の検索セクションの指示に従います。 2. AMS コンソールで、類似タイプの既存の RFC を開く、またはテストとして新しい RFC を作成します。RFC の CT ID と実行パラメータを使用します。詳細については、AWS Managed Services ドキュメントの「コンソールでの RFC の検索」を参照してください。 **Note** このパターンの Python オートメーションを他の RFCs に適応させるには、`AMS Stop EC2 Instance.zip` ファイル (添付) の Python	AWS システム管理者

タスク	説明	必要なスキル
	コードファイルの CT ams_stop_ec2_instance タイプとパラメータ値を抽出した値に置き換えます。	

Python オートメーションを実行します。

タスク	説明	必要なスキル
Python オートメーションを実行します。	1. AMS Stop EC2 Instance.zip ファイル (添付) をローカルマシンにダウンロードし、ファイルを抽出します。 2. EC2 インスタンス情報で input_instances を更新します。 3. ターミナルを開き、抽出したコードのパスにナビゲートします。 4. pythonams_stop_ec2_instance.py コマンドを実行します。	AWS システム管理者

関連リソース

- 変更タイプとは?
- CLI チュートリアル: 高可用性 2 層スタック (Linux/RHEL)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Systems Manager メンテナンスウィンドウを使用して Amazon RDS DB インスタンスを自動的に停止および起動する

作成者: Ashita Dsilva (AWS)

概要

このパターンは、 AWS Systems Manager メンテナンスウィンドウを使用して、特定のスケジュールで Amazon Relational Database Service (Amazon RDS) DB インスタンスを自動的に停止および開始する方法 (例: 営業時間外に DB インスタンスをシャットダウンしてコストを削減する) を示しています。

AWS Systems Manager Automation は、`AWS-StopRdsInstance`Amazon RDS DB インスタンスを停止および起動するための および `AWS-StartRdsInstance`ランブックを提供します。つまり、AWS Lambda 関数を使用してカスタムロジックを記述したり、Amazon CloudWatch Events ルールを作成したりする必要はありません。

Systems Manager には、ステートマネージャーとメンテナンスウィンドウの 2 つのタスクをスケジュールする機能があります。ステートマネージャーは、Amazon Web Services (AWS) アカウントのリソースに必要な状態設定を 1 回または特定のスケジュールで設定および維持します。Maintenance Windows は、特定の時間枠にアカウント内のリソースに対してタスクを実行します。ステートマネージャーまたはメンテナンスウィンドウではこのパターンのアプローチを使用できますが、割り当てられた優先度に基づいて 1 つ以上のタスクを実行でき、 AWS Lambda 関数と AWS Step Functions タスクを実行できるため、メンテナンスウィンドウを使用することをお勧めします。ステートマネージャーとメンテナンスウィンドウの詳細については、Systems Manager ドキュメントの「ステートマネージャーとメンテナンスウィンドウの選択」を参照してください。

このパターンでは、cron 式を使用して Amazon RDS DB インスタンスを停止してから起動する 2 つのメンテナンスウィンドウを個別に設定する詳細な手順を示しています。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- 特定のスケジュールで停止して開始したい既存の Amazon RDS DB インスタンス。
- 必要なスケジュールの Cron 式。例えば、式は毎週月曜日、火曜日、水曜日、木曜日、金曜日の 9:00 にタスク`cron(0 9 ? * MON-FRI *)`を実行します。詳細については、Systems Manager ドキュメントのメンテナンスウィンドウの Cron 式と rate 式を参照してください。

- Systems Manager に精通しています。
- RDS インスタンスを起動および停止するアクセス許可。詳細については、「エピック」セクションを参照してください。

制約事項

- Amazon RDS DB インスタンスは一度に最大 7 日間停止できます。7 日後、DB インスタンスは自動的に再起動し、必要なメンテナンスアップデートを確実に受け取ることができます。
- リードレプリカである DB インスタンス、またはリードレプリカを持つ DB インスタンスを停止することはできません。
- マルチ AZ 設定では Amazon RDS for SQL Server DB インスタンスを停止できません。
- Service Quotas は、Maintenance Windows と Systems Manager Automation に適用されます。サービスクォータの詳細については、 AWS 全般のリファレンス ドキュメントのAWS Systems Manager 「エンドポイントとクォータ」を参照してください。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

アーキテクチャ

次の図では、Amazon RDS DB インスタンスを自動的に停止して開始するワークフローを示します。

ワークフローには次の手順があります。

1. メンテナンスウィンドウを作成し、cron 式を使用して Amazon RDS DB インスタンスの停止と開始のスケジュールを定義します。

2. `AWS-StopRdsInstance` または `AWS-StartRdsInstance` ランブックを使用して Systems Manager 自動化タスクをメンテナンスウィンドウに登録します。

3. Amazon RDS DB インスタンスのタグベースのリソースグループを使用して、メンテナンスウィンドウにターゲットを登録します。

テクノロジースタック

- AWS CloudFormation

- AWS Identity and Access Management (IAM)
- Amazon RDS
- Systems Manager

自動化とスケール

必要な Amazon RDS DB インスタンスにタグを付け、タグ付けされたすべての DB インスタンスを含むリソースグループを作成し、このリソースグループをメンテナンスウィンドウのターゲットとして登録することで、複数の Amazon RDS DB インスタンスを同時に停止および起動できます。

ツール

- AWS CloudFormation は、 AWS リソースのモデル化とセットアップに役立つサービスです。
- AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。
- Amazon Relational Database Service (Amazon RDS) は、 でリレーショナルデータベースを簡単にセットアップ、運用、スケーリングできるウェブサービスです AWS クラウド。
- AWS Resource Groups は、 AWS リソースをグループに整理し、リソースにタグを付け、グループ化されたリソースのタスクを管理、モニタリング、自動化するのに役立ちます。
- AWS Systems Manager は、インフラストラクチャの表示と制御 AWS のサービス に使用できるです AWS。このパターンでは、Systems Manager の以下の機能を使用します。
  - AWS Systems Manager 自動化により、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスやその他の AWS リソースの一般的なメンテナンスとデプロイタスクが簡素化されます。
  - AWS Systems Manager メンテナンスウィンドウは、インスタンスで破壊的になり得るアクションを実行するスケジュールを定義するのに役立ちます。

エピック

Systems Manager Automation の IAM サービスロールを作成および設定する

タスク	説明	必要なスキル
Systems Manager Automation の IAM サービスロールを設定します。	にサインイン AWS Management Console し、Systems Manager	AWS 管理者

タスク	説明	必要なスキル
	Automation のサービスロールを作成します。次の 2 つのメソッドのいずれかを使用して、このサービスロールを作成できます。 • AWS CloudFormation を使用して Systems Manager Automation のサービスロールを設定する • IAM を使用して、オートメーションのロールを設定する Systems Manager 自動化ワークフローは、サービスロールを使用して Amazon RDS DB インスタンスで開始アクションと停止アクションを実行することで Amazon RDS を呼び出します。 サービスロールは、Amazon RDS DB インスタンスを起動および停止する権限を持つ以下のインラインポリシーで設定する必要があります。 `{ "Version": "2012-10-17", "Statement": [ { "Sid": "RdsStartStop",`	

タスク	説明	必要なスキル
	"Effect": "Allow", "Action": ["rds:StopDBInstance", "rds:StartDBInstance"], "Resource": "<RDS_Instance_ARN>" }, { "Sid": "RdsDescribe", "Effect": "Allow", "Action": "rds:DescribeDBInstances", "Resource": "*" }] } を Amazon RDS DB インスタンスの Amazon リソースネーム (ARN) <RDS_Instance_ARN> に置き換えてください。 IAM ポリシーとロールの使用に慣れていない場合は、Amazon RDS の停止と使用開始のスケジュール AWS Systems Managerのソリューション概要セクションの指示に従ってください。	

タスク	説明	必要なスキル
	Important サービスロールの ARN を必ず記録してください。	

リソースグループの作成

タスク	説明	必要なスキル
Amazon RDS DB インスタンスにタグを付けます	Amazon RDS コンソールを開き、リソースグループに追加する Amazon RDS DB インスタンスにタグを付けます。タグは AWS リソースに割り当てられたメタデータで、キーと値のペアで構成されます。[アクション] を [タグキー] として使用し、[StartStop] を [値] として使用することをお勧めします。 詳細については、Amazon RDS ドキュメントの「タグの追加、リスト化、削除」を参照してください。	AWS 管理者
タグを付加した Amazon RDS DB インスタンス用のリソースグループを作成します。	AWS Resource Groups コンソールを開き、Amazon RDS DB インスタンス用に作成したタグに基づいてリソースグループを作成します。 [グループ化基準] で、必ずリソースタイプに [AWS:: RDS::	AWS 管理者

タスク	説明	必要なスキル
	DBInstance] を選択し、タグのキーと値のペア (「Action-StartStop」など) を指定してください。これにより、サービスは Amazon RDS DB インスタンスのみをチェックし、このタグを持つその他のリソースはチェックしないことが保証されます。リソースグループの名前は必ず記録してください。 詳細と詳細な手順については、 AWS Resource Groups ドキュメントの「タグベースのクエリの構築」および「グループの作成」を参照してください。	

Amazon RDS DB インスタンスを停止するメンテナンスウィンドウを設定する

タスク	説明	必要なスキル
メンテナンスウィンドウを作成します。	1. Systems Manager コンソールを開き、メンテナンスウィンドウを選択し、メンテナンスウィンドウの作成を選択します。メンテナンスウィンドウの名前 (「StopRdsInstance」など) を入力し、説明を入力して、[未登録のターゲットを許可する] のチェックを外します。	AWS 管理者

タスク	説明	必要なスキル
	2. [CRON/Rate 式] を選択し、Amazon RDS DB インスタンスを停止するタイミングを定義するスケジュール式を指定します。[期間] に 1 を入力し、[タスクの開始を停止する] に 0 を入力します。デフォルトでは、タイムゾーンは UTC です。cron 式で定義したタイムスタンプに基づいて、タイムゾーンを変更してメンテナンスウィンドウを開始できます。 3. [Create maintenance window] を選択します。メンテナンスウィンドウのページに戻り、メンテナンスウィンドウの状態は [有効] になります。 ⚠ Important DB インスタンスを停止するタスクは、開始するとほぼ瞬時に実行され、メンテナンスウィンドウの期間全体にわたるものではありません。このパターンでは、タスクの開始期間と開始タスクの停止の最小値が提供されます。これは、これら	

タスク	説明	必要なスキル
	がメンテナンス時間枠に必要なパラメータであるためです。 詳細と詳細な手順については、Systems Manager ドキュメントの「メンテナンスウィンドウの作成 (コンソール)」を参照してください。	
ターゲットをメンテナンスウィンドウに割り当てます。	1. Systems Manager コンソールで、メンテナンスウィンドウを選択し、アクションを選択し、ターゲットの登録を選択します。 2. [ターゲット] 領域で [リソースグループを選択] を指定し、アカウント内の既存のリソースグループの名前を選択します。 3. リソースタイプについては、AWS:: RDS:: DBInstance を選択し、次に [ターゲットを登録] を選択します。 詳細と詳細な手順については、Systems Manager ドキュメントの「メンテナンスウィンドウにターゲットを割り当てる (コンソール)」を参照してください。	AWS 管理者

タスク	説明	必要なスキル
メンテナンスウィンドウにタスクを割り当てるには	1. Systems Manager コンソールで、メンテナンスウィンドウを選択し、メンテナンスウィンドウを選択します。アクションを選択し、自動化タスクの登録を選択します。 2. [ドキュメント] には、[AWS-StopRDS インスタンス] を選択します。 3. ターゲット セクションで、登録されたターゲットグループの選択を選択し、現在のメンテナンスウィンドウに登録したメンテナンスウィンドウターゲットを選択します。 4. レートコントロールでは、[同時実行数] と [エラーしきい値] に 100% を指定します。レートコントロールの値は、タスクの同時実行性とエラーしきい値の要件に応じて変更できます。詳細については、Systems Manager ドキュメントの「同時実行とエラーのしきい値について」を参照してください。 5. IAM サービスロールセクションのサービスロールでは、このボックスを空白のままにするか、独自のカスタムロールを作成し	AWS 管理者

タスク	説明	必要なスキル
	ます。ボックスを空白のままにすると、Systems Manager はサービスにリンクされたロール AWSServiceRoleForAmazonSSM を自動的に作成し、ロールをタスクに関連付けます。独自のカスタムロールを作成するには、「メンテナンスウィンドウのカスタムサービスロールを作成する (コンソール)」を参照してから、そのカスタムロールをタスクに関連付けます。 6. [入力パラメータ] セクションで、ランブックの次のパラメータを指定します。 • InstanceId: {{RESOURCE_ID}} **Note** InstanceId の場合、擬似パラメータを使用して ARN から Amazon RDS DB リソース ID を抽出します。擬似パラメータの詳細については、Systems Manager ドキュメントの「擬似パラメータにつ	

タスク	説明	必要なスキル
	いて」を参照してください。 • AutomationAssumeRole: Systems Manager オートメーション用に作成したサービスロールの ARN を指定します。 7. [オートメーションの登録] タスクを選択します。 ⚠ Important サービスロールオプションは、メンテナンスウィンドウがタスクを実行するために必要なサービスロールを定義します。ただし、このロールは、以前に Systems Manager Automation 用に作成したサービスロールと同じではありません。 詳細と詳細な手順については、Systems Manager ドキュメントの「メンテナンスウィンドウにタスクを割り当てる (コンソール) 」を参照してください。	

Amazon RDS DB インスタンスを起動するメンテナンスウィンドウを設定する

タスク	説明	必要なスキル
Amazon RDS DB インスタンスを起動するメンテナンスウィンドウを設定します。	Amazon RDS DB インスタンスを停止するには「メンテナンスウィンドウの設定」の手順を繰り返し、スケジュールされた時間に Amazon RDS DB インスタンスを起動する別のメンテナンスウィンドウを設定します。 **Important** DB インスタンスを起動するようにメンテナンスウィンドウを設定するときは、次の変更を行う必要があります。 • メンテナンスウィンドウには新しい名前 (「StartRDSInstance」など) を使用してください。 • cron 式を DB インスタンスの起動に使用したい cron 式に置き換えてください。 • [タスク] の `AWS-StopRdsInstance` ランブックを `AWS-StartRdsInstance` に置き換えます。	AWS 管理者

関連リソース

- Systems Manager Automation ドキュメントを使用してインスタンスを管理し、時間外にコストを削減する (AWS ブログ記事)

Terraform を使用して AWS Organizations でソフトウェアパッケージの配布を一元化する

作成者: Pradip kumar Pandey (AWS)、Aarti Rajput (AWS)、Chintamani Aphale (AWS)、T.V.R.L.Phani Kumar Dadi (AWS)、Mayuri Shinde (AWS)、Pratap Kumar Nanda (AWS)

概要

多くの場合、ワークロード間に強力な分離障壁を構築 AWS リージョン するために、企業は複数のにまた AWS アカウント がる複数の を維持します。セキュリティとコンプライアンスを維持するために、管理チームはセキュリティスキャン用の CrowdStrike、SentinelOne、TrendMicro ツールなどのエージェントベースのツールと、モニタリング用の Amazon CloudWatch エージェント、Datadog エージェント、AppDynamics エージェントをインストールします。これらのチームは、この大規模な環境全体でソフトウェアパッケージの管理と配布を一元的に自動化したい場合に、課題に直面することがよくあります。

の一機能である Distributor は、単一の簡素化されたインターフェイスを通じてAWS Systems Manager、クラウドおよびオンプレミスサーバー全体でマネージド Microsoft Windows および Linux インスタンスにソフトウェアをパッケージ化して公開するプロセスを自動化します。このパターンは、Terraform を使用してソフトウェアのインストール管理のプロセスをさらに簡素化し、最小限の労力 AWS Organizations で 内の多数のインスタンスとメンバーアカウントでスクリプトを実行する方法を示しています。

このソリューションは、Systems Manager によって管理される Amazon、Linux、および Windows インスタンスで機能します。

前提条件と制限

- インストールするソフトウェアがあるディストリビューターパッケージ
- Terraform バージョン 0.15.0 以降
- Systems Manager によって管理され、ターゲットアカウントの Amazon Simple Storage Service (Amazon S3Amazon S3) にアクセスするための基本的なアクセス許可を持つ Amazon Elastic Compute Cloud (Amazon EC2) インスタンス
- を使用してセットアップされた組織のランディングゾーン AWS Control Tower
- （オプション) Account Factory for Terraform (AFT)

アーキテクチャ

リソースの詳細

このパターンでは、Account Factory for Terraform (AFT) を使用して、必要なすべての AWS リソースを作成し、コードパイプラインを使用してデプロイアカウントにリソースをデプロイします。コードパイプラインは 2 つのリポジトリで実行されます。

- グローバルカスタマイズには、AFT に登録されているすべてのアカウントで実行される Terraform コードが含まれています。
- アカウントのカスタマイズには、デプロイアカウントで実行される Terraform コードが含まれています。

アカウントカスタマイズフォルダで Terraform コマンドを実行することで、AFT を使用せずにこのソリューションをデプロイすることもできます。

Terraform コードは、次のリソースをデプロイします。

- AWS Identity and Access Management (IAM) ロールとポリシー
  - SystemsManager-AutomationExecutionRole は、ターゲットアカウントでオートメーションを実行するアクセス許可をユーザーに付与します。
  - SystemsManager-AutomationAdministrationRole は、複数のアカウントと組織単位 (OUs。
- パッケージの圧縮ファイルと manifest.json
  - Systems Manager では、パッケージにはソフトウェアまたはインストール可能なアセットの .zip ファイルが少なくとも 1 つ含まれています。
  - JSON マニフェストには、パッケージコードファイルへのポインタが含まれています。
- S3 バケット
  - 組織全体で共有されている分散パッケージは、Amazon S3 バケットに安全に保存されます。
- AWS Systems Manager ドキュメント (SSM ドキュメント)
  - `DistributeSoftwarePackage` には、メンバーアカウントのすべてのターゲットインスタンスにソフトウェアパッケージを配布するロジックが含まれています。
  - `AddSoftwarePackageToDistributor` には、インストール可能なソフトウェアアセットをパッケージ化し、その機能である Automation に追加するロジックが含まれています AWS Systems Manager。
- Systems Manager の関連付け

- Systems Manager の関連付けを使用してソリューションをデプロイします。

アーキテクチャとワークフロー

この図表は以下のステップを示しています。

1. 一元化されたアカウントからソリューションを実行するには、パッケージまたはソフトウェアをデプロイステップとともに S3 バケットにアップロードします。
2. カスタマイズされたパッケージは、「Owned by me」タブの「Systems Manager コンソールのドキュメント」セクションで利用できるようになります。
3. Systems Manager の一機能であるステートマネージャーは、組織全体でパッケージの関連付けを作成、スケジュール、実行します。関連付けでは、ソフトウェアパッケージをターゲットノードにインストールする前に、マネージドノードにインストールして実行する必要があります。
4. 関連付けは、ターゲットノードにパッケージをインストールするように Systems Manager に指示します。
5. それ以降のインストールまたは変更の場合、ユーザーは同じ関連付けを 1 つの場所から定期的または手動で実行して、アカウント間でデプロイを実行できます。
6. メンバーアカウントでは、自動化はデプロイコマンドをディストリビューターに送信します。
7. ディストリビューターは、 インスタンス間でソフトウェアパッケージを配布します。

このソリューションでは、 内の管理アカウントを使用しますが AWS Organizations、組織に代わってこれを管理するためのアカウント (委任管理者) を指定することもできます。

ツール

AWS サービス

- Amazon Simple Storage Service (Amazon S3) は、量にかかわらず、データを保存、保護、取得するのに役立つクラウドベースのオブジェクトストレージサービスです。このパターンでは、Amazon S3 を使用して分散パッケージを一元化し、安全に保存します。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理を簡素化し、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できるようにします。このパターンでは、次の Systems Manager 機能を使用します。

  - Distributor は、ソフトウェアをパッケージ化して Systems Manager マネージドインスタンスに公開するのに役立ちます。
  - 自動化は、多くの AWS サービスの一般的なメンテナンス、デプロイ、修復タスクを簡素化します。
  - ドキュメントは、組織とアカウント全体で Systems Manager マネージドインスタンスに対してアクションを実行します。
- AWS Organizations は、作成して一元管理する組織に複数の AWS アカウントを統合するのに役立つアカウント管理サービスです。

その他のツール

- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンの手順とコードは、GitHub Centralized Package Distribution リポジトリで入手できます。

ベストプラクティス

- 関連付けにタグを割り当てるには、 AWS Command Line Interface (AWS CLI) または を使用しますAWS Tools for PowerShell。Systems Manager コンソールを使用して関連付けにタグを追加することはできません。詳細については、Systems Manager ドキュメントの「Systems Manager リソースのタグ付け」を参照してください。
- 別のアカウントから共有されたドキュメントの新しいバージョンを使用して関連付けを実行するには、ドキュメントのバージョンを に設定しますdefault。
- ターゲットノードにのみタグを付けるには、1 つのタグキーを使用します。複数のタグキーを使用してノードをターゲットにする場合は、リソースグループオプションを使用します。

エピック

ソースファイルとアカウントを設定する

タスク	説明	必要なスキル

リポジトリをクローン作成します。	1. GitHub 集中型パッケージディストリビューションリポジトリのクローンを作成します。
	`git clone https://github.com/aws-samples/aws-organization-centralised-package-distribution`
	2. Terraform コードリポジトリには、AFT によって管理される 2 つのカスタマイズフォルダが必要です。リポジトリのローカルコピーに次のフォルダが含まれていることを確認します。
	`$ cd centralised-package-distribution`
	`$ ls`
	`global-customization account-customization`
		DevOps エンジニア

グローバル変数を更新します。	`global-customization/variables.tf` ファイルで次の入力パラメータを更新します。これらの変数は、AFT によって作成および管理されるすべてのアカウントに適用されます。	DevOps エンジニア

タスク	説明	必要なスキル
	• account_id ：Distributor ソリューションをデプロイするアカウントの ID。 • aws_region ：関連付けがデプロイされる AWS リージョン 。	
アカウント変数を更新します。	account-customization/variables.tf ファイルで次の入力パラメータを更新します。これらの変数は、AFT によって作成および管理される特定のアカウントにのみ適用されます。 • package_bucket_name ：パッケージディストリビューションファイルを含む S3 バケットの名前。 • package_name ：パッケージディストリビューションファイルの名前。 • package_version ：インストーラのパッケージバージョン。	DevOps エンジニア

パラメータとデプロイファイルのカスタマイズ

タスク	説明	必要なスキル
ステートマネージャーの関連付けの入力パラメータを更新します。	account-customization/association.tf ファイル内の次の入力パラメータを更新して、インスタ	DevOps エンジニア

タスク	説明	必要なスキル

	ンスで維持する状態を定義します。デフォルトのパラメータ値は、ユースケースをサポートしている場合に使用できます。
	
	• `targetAccounts` ：ディストリビューションのターゲットインスタンスを持つアカウントを表す AWS Organizations 内の組織単位 (OU) IDs。OU IDs「ou」で始まります。
	• `targetRegions` ：ターゲットインスタンスが実行されている AWS リージョン（たとえば、「us-east-1」または「ap-southeast-2」）。
	• `action`： パッケージをインストールするかアンインストールするかを指定します。
	• `installationType` ： 次のいずれかのインストールタイプ。
	  • `uninstall` ：パッケージはアンインストールされます。
	  • `reinstall` ：再インストールプロセスが完了するまで、アプリケーションはオフラインになります。	

タスク	説明	必要なスキル
	• In-place update: 新規または更新されたファイルがインストールに追加されている間、アプリケーションを使用できます。 • name: インストールまたはアンインストールするパッケージの名前。 • version： インストールまたはアンインストールするパッケージのバージョン。パッケージのバージョンがインストールされていない場合、システムはエラーを返します。 • bucketName ： パッケージがデプロイされた S3 バケット名。このバケットは、パッケージとマニフェストファイルのみで構成される必要があります。 • bucketPrefix ： パッケージアセットが保存される S3 プレフィックス。 • AutomationAssumeRole ：の Amazon リソースネーム (ARN)SystemsManager-AutomationAdministrationRole 。	

タスク	説明	必要なスキル
圧縮ファイルと manifest.json ファイルをパッケージ用に準備します。	このパターンは、 account-customization/package フォルダにインストールおよびアンインストールスクリプトを含むサンプル PowerShell インストール可能ファイル (Windows の場合は .msi、Linux の場合は .rpm) を提供します。 1. PowerShell のインストール可能なファイルを独自のファイルに置き換えるか、インストール可能なファイル、インストールおよびアンインストールスクリプト、マニフェストファイルを提供して、アカウントの account-customization フォルダにパッケージを作成します。 2. 要件に応じて、Terraform が account-customization フォルダで生成するデフォルトmanifest.json ファイルをカスタマイズします。	DevOps エンジニア

Terraform コマンドを実行してリソースをプロビジョニングする

タスク	説明	必要なスキル
Terraform の設定を初期化します。	AFT を使用してソリューションを自動的にデプロイするに	DevOps エンジニア

タスク	説明	必要なスキル

	は、コードを次の宛先にプッシュします AWS CodeCommit。
	
	`$ git add *`
	`$ git commit -m "message"`
	`$ git push`
	
	`account-customization` フォルダから Terraform コマンドを実行して、AFT を使用せずにこのソリューションをデプロイすることもできます。Terraform ファイルを含む作業ディレクトリを初期化するには、以下を実行します。
	
	`$ terraform init`	

変更をプレビューします。	Terraform がインフラストラクチャに加える変更をプレビューするには、 コマンドを実行します。 `$ terraform plan` このコマンドは、Terraform 設定を評価して、宣言されたリソースの望ましい状態を決定します。また、ワークスペース内でプロビジョニングする実際のインフラストラクチャと目的の状態を比較します。	DevOps エンジニア

タスク	説明	必要なスキル
変更を適用します。	次のコマンドを実行して、variables.tf ファイルに加えた変更を実装します。 `$ terraform apply`	DevOps エンジニア

リソースを検証する

タスク	説明	必要なスキル
SSM ドキュメントの作成を検証します。	1. Systems Manager コンソールの左側のナビゲーションペインで、ドキュメントを選択します。 2. [Owned by me (自分が所有)] タブを選択します。 DistributeSoftwarePackage および AddSoftwarePackageToDistributor パッケージが表示されます。	DevOps エンジニア
オートメーションのデプロイが成功したことを検証します。	1. Systems Manager コンソールの左側のナビゲーションペインで、オートメーションを選択します。 2. 自動化実行リストには、最新の DistributeSoftwarePackage および AddSoftwarePackageToDistrib	DevOps エンジニア

タスク	説明	必要なスキル
	utor デプロイが表示されます。 3. 実行 ID を選択して、正常に完了したことを確認します。	
ターゲットのメンバーアカウントインスタンスにパッケージがデプロイされたことを確認します。	1. Systems Manager コンソールのナビゲーションペインで、コマンドの実行を選択します。 2. コマンド履歴には、各呼び出しとそのステータスが表示されます。 3. コマンド ID を選択すると、各ターゲットインスタンスのデプロイ履歴が表示されます。 4. インスタンス ID を選択し、ディストリビューションの出力セクションを確認します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
ステートマネージャーの関連付けが失敗したか、保留中のステータスのままです。	AWS ナレッジセンターのトラブルシューティング情報を参照してください。
スケジュールされた関連付けの実行に失敗しました。	スケジュール仕様が無効である可能性があります。ステートマネージャーは、現在、関連付けの cron 式での月の指定をサポートしていません。cron 式または rate 式を使用してスケジュールを確認します。

関連リソース

- 一元化されたパッケージディストリビューション (GitHub リポジトリ)
- Account Factory for Terraform (AFT)
- ユースケースとベストプラクティス (AWS Systems Manager ドキュメント)

NLog を使用して Amazon CloudWatch Logs 内の.NET アプリケーションのロギングを設定します

作成者:Bibhuti Sahu (AWS) and Rob Hill (AWS) (AWS)

概要

このパターンでは、NLog オープンソースのロギングフレームワークを使用して、.NET アプリケーションの使用状況とイベントを「Amazon CloudWatch Logs」に記録する方法について説明します。CloudWatch コンソールでは、アプリケーションのログメッセージをほぼリアルタイムで見ることができます。メトリックスを設定し、「メトリクス」のしきい値を超えた場合に通知するように「アラーム」を設定することもできます。CloudWatch Application Insights を使用すると、監視対象アプリケーションの潜在的な問題を示す自動またはカスタムのダッシュボードを表示できます。CloudWatch Application Insights は、アプリケーションとインフラストラクチャに関する進行中の問題をすばやく切り分けることができるように設計されています。

CloudWatch Logs にログメッセージを書き込むには、`AWS.Logger.NLog` NuGet パッケージを.NET プロジェクトに追加します。次に、CloudWatch Logs をターゲットとして使用するように `NLog.config` ファイルを更新します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- .NET ウェブアプリケーションまたはコンソールアプリケーション:
  - サポートされている.NET Framework バージョンまたは.NET Core バージョンを使用します。詳細については、「製品バージョン」を参照してください。
  - NLog を使用してログデータをアプリケーションインサイトのに送信します。
- AWS サービスの IAM ロールを作成するアクセス許可。詳細については、「サービスロールのアクセス許可」を参照してください。
- AWS サービスにロールを渡すためのアクセス許可。詳細については、Granting a user permissions to pass a role to an AWS service を参照してください。

製品バージョン

- .NET Framework バージョン 3.5 またはそれ以降

- .NET Core バージョン 1.0.1、2.0.0、またはそれ以降

アーキテクチャ

ターゲットテクノロジースタック

- NLog
- Amazon CloudWatch Logs

ターゲットアーキテクチャ

1. .NET アプリケーションは NLog ロギングフレームワークにログデータを書き込みます。
2. NLog は CloudWatch Logs にログデータを書き込みます。
3. CloudWatch アラームとカスタムダッシュボードを使用して .NET アプリケーションをモニタリングします。

ツール

AWS サービス

- 「Amazon CloudWatch Application Insights」は、アプリケーションと、その基盤となる AWS のリソースに対するオブザーバビリティを実現します。
- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、 からのログを一元化するのに役立ちます。一元化により、ログを監視して安全にアーカイブできます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- 「AWS Tools for PowerShell」は PowerShell モジュールのセットで、PowerShell コマンドラインから AWS リソースに対するオペレーションのスクリプトを作成できます。

その他のツール

- 「Logger.nLog」は、ログデータを CloudWatch Logs に記録する NLog ターゲットです。
- 「NLog」は .NET プラットフォーム用のオープンソースのロギングフレームワークで、データベース、ログファイル、コンソールなどのターゲットにログデータを書き込むのに役立ちます。

- 「PowerShell」は Windows、Linux、および macOS で動作するMicrosoft の自動化および構成管理プログラムです。
- 「Visual Studio」は、コンパイラー、コード補完ツール、グラフィカルデザイナー、およびソフトウェア開発をサポートするその他の機能を含む統合開発環境 (IDE) です。

ベストプラクティス

- ターゲットロググループの「保存ポリシー」を設定します。これは NLog 設定の外部で行う必要があります。デフォルトでは、ログデータは CloudWatch Logs に無期限に保存されます。
- 「AWS アクセスキーを管理するためのベストプラクティス」を遵守します。

エピック

アクセスとツールのセットアップ

タスク	説明	必要なスキル
IAM ポリシーを作成します。	IAM ドキュメントの「JSON エディタを使ったポリシーの作成」の指示に従ってください。次の JSON ポリシーを入力します。このポリシーには、CloudWatch Logs にログの読み取りと書き込みを許可するのに必要な最小特権のアクセス権限があります。 `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "logs:CreateLogGroup",`	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル

	<pre>"logs:CreateLogStream",
	"logs:GetLogEvents",
	"logs:PutLogEvents",
	"logs:DescribeLogGroups",
	"logs:DescribeLogStreams",
	"logs:PutRetentionPolicy"
	],
	"Resource":
	[
	"*"
	]
	}
	]
	}</pre>	

IAM ロールを作成します。	IAM ドキュメントの「AWS のサービスにアクセス許可を委任するロールの作成」の手順に従ってください。前に作成したポリシーを選択します。これは、CloudWatch Logs がロギングアクションを実行するために引き受けるロールです。	AWS 管理者、AWS DevOps

タスク	説明	必要なスキル
AWS Tools for PowerShellのセットアップ。	1. 「AWS Tools for PowerShell のインストール」で使用しているオペレーティングシステムの手順に従ってください。 2. AWS Tools for PowerShell コマンドレットを使用して、アクセスキーとシークレットキーをプロファイルに保存します。手順については、AWS Tools for PowerShell ドキュメントの「プロファイルの管理」を参照してください。	AWS 全般

NLog を設定

タスク	説明	必要なスキル
NuGet パッケージをインストールします。	1. Visual Studio で [ファイル] を選択し、[プロジェクトまたはソリューションを開く] を選択します。 2. NLog をインストールするプロジェクトを選択します。 3. Visual Studio で [Tools] (ツール) メニューから、[Package Manager Console] (NuGet パッケージマネージャー)、[Package Manager Console] (パッ	アプリ開発者

タスク	説明	必要なスキル
	ケージマネージャーコンソール) を選択します。 4. 次のコマンドを入力して、AWS.Logger.NLog NuGet パッケージをインストールします。 `Install-Package AWS.Logger.NLog -Version 3.1.0`	

タスク	説明	必要なスキル
ロギングターゲットを設定します。	1. `NLog.config` ファイルを開きます。 2. ターゲット `type` には `AWSTarget` を入力します。 3. ターゲット `logGroup` には、使用する「ロググループ」の名前を入力します。ロググループがまだ存在しない場合、指定した名前の新しいロググループが自動的に作成されます。 4. ターゲット `region` には、CloudWatch Logs が設定されている AWS リージョンを入力します。 5. ターゲット `profile` には、アクセスキーとシークレットキーを保存するために以前に作成したプロファイルの名前を入力します。 6. `NLog.config` ファイルを保存して閉じます。 設定ファイルのサンプルについては、このパターンの「追加情報」セクションを参照してください。アプリケーションを実行すると、NLog がログメッセージを書き込んで CloudWatch Logs に送信します。	アプリ開発者

ログの検証とモニタリング

タスク	説明	必要なスキル
ロギングを検証する。	CloudWatch Logs ドキュメントの「CloudWatch Logs に送信されたログデータを表示する」の指示に従ってください。.NET アプリケーションのログイベントが記録されていることを確認します。ログイベントが記録されていない場合は、このパターンの「トラブルシューティング」セクションを参照してください。	AWS 全般
.NET アプリケーションスタックを監視します。	ユースケースの必要に応じて CloudWatch でモニタリングを設定します。「CloudWatch ログインサイト」、「CloudWatch メトリクスインサイト」、および「CloudWatch アプリケーションインサイト」を使用して、.NET ワークロードをモニタリングできます。アラートを受信できるように「アラーム」を設定したり、1 つのビューからワークロードを監視するためのカスタム「ダッシュボード」を作成したりすることもできます。	AWS 全般

トラブルシューティング

問題	ソリューション
ログデータは CloudWatch Logs には表示されません。	IAM ポリシーが CloudWatch Logs が引き受ける IAM ロールにアタッチされていることを確認します。手順については、「エピック」セクションの「アクセスとツールのセットアップ」セクションを参照してください。

関連リソース

- 「ロググループとログストリームの操作」(CloudWatch Logsドキュメント)
- 「Amazon CloudWatch Logs と.NET ロギングフレームワーク」(AWS ブログ記事)

追加情報

次に、サンプル NLog.config ファイルを示します。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="nlog" type="NLog.Config.ConfigSectionHandler, NLog" />
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <nlog>
    <extensions>
      <add assembly="NLog.AWS.Logger" />
    </extensions>
    <targets>
      <target name="aws" type="AWSTarget" logGroup="NLog.TestGroup" region="us-east-1"
 profile="demo"/>
    </targets>
    <rules>
      <logger name="*" minlevel="Info" writeTo="aws" />
    </rules>
  </nlog>
```

```
</configuration>
```

AWS Service Catalog 製品を異なる AWS アカウントと AWS リージョンにコピー

作成者：Sachin Vighe (AWS) と Santosh Kale (AWS)

概要

AWS Service Catalog はリージョナルサービスです。つまり、AWS Service Catalog のポートフォリオと製品 は、それらが作成された AWS リージョンでのみ表示されます。新しいリージョンに AWS Service Catalog ハブ をセットアップする場合、既存の製品を再作成する必要があり、これには時間がかかる場合があります。

このパターンのアプローチは、ソース AWS アカウントまたはリージョンの AWS Service Catalog ハブからターゲットアカウントまたはリージョンの新しいハブに製品をコピーする方法を説明することで、このプロセスを簡素化するのに役立ちます。AWS Service Catalog のハブとスポークモデルの詳細については、AWS 管理とガバナンスブログのAWS Service Catalog のハブとスポークモデル：AWS Service Catalog の多数のアカウントへのデプロイと管理を自動化する方法を参照してください。

このパターンでは、AWS Service Catalog 製品をアカウント間でコピーしたり、他のリージョンにコピーしたりするのに必要な個別のコードパッケージも用意されています。このパターンを使用することにより、組織は時間を節約し、既存および以前の製品バージョンを新しい AWS Service Catalog ハブで利用できるようにし、手動エラーのリスクを最小限に抑え、複数のアカウントまたはリージョンにわたってアプローチを拡大できます。

> **Note**
>
> このパターンのエピックセクションには、製品をコピーするための 2 つのオプションがあります。オプション 1 を使用してアカウントをまたいで製品をコピーするか、オプション 2 を使用してリージョン間で製品をコピーできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- ソースアカウントまたはリージョンの既存の AWS Service Catalog 製品。
- 移行先のアカウントまたはリージョンの既存の AWS Service Catalog ハブ。

- アカウントをまたいで製品をコピーする場合は、製品を含む AWS Service Catalog ポートフォリオを共有してから宛先アカウントにインポートする必要があります。詳細については、AWS Service Catalog ドキュメントのポートフォリオの共有とインポート を参照してください。

機能制限

- リージョンまたはアカウント間でコピーする AWS Service Catalog 製品は、複数のポートフォリオに属することはできません。

アーキテクチャ

次の図は、移行元アカウントから移行先アカウントへの AWS Service Catalog 製品のコピーを示しています。

次の図は、ソースリージョンからターゲットリージョンへの AWS Service Catalog 製品のコピーを示しています。

テクノロジースタック

- Amazon CloudWatch
- AWS Identity and Access Management (IAM)
- AWS Lambda
- AWS Service Catalog

自動化とスケール

このパターンのアプローチは、受信したリクエストの数やコピーする必要がある AWS Service Catalog 製品の数に応じてスケーリングできる Lambda 関数を使用することでスケーリングできます。このプロパティに関する詳細については、AWS Lambda 開発者ガイドのAWS Lambda 関数スケーリングを参照してください。

ツール

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Service Catalog では、では、AWS で承認された IT サービスのカタログを一元管理できます。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。

コード

cross-account-copy パッケージ (添付済み) を使用して、アカウント間で AWS Service Catalog 製品をコピーするか、 cross-region-copy パッケージ (添付) を使用してリージョン間で製品をコピーできます。

cross-account-copy パッケージには以下のファイルが含まれています。

- copyconf.properties — アカウント間で製品をコピーするためのリージョンと AWS アカウント ID パラメータを含む設定ファイル。
- scProductCopyLambda.py — アカウント間で製品をコピーするための Python 関数。
- createDestAccountRole.sh — 宛先アカウントに IAM ロールを作成するスクリプト。
- createSrcAccountRole.sh — ソースアカウントに IAM ロールを作成するスクリプト。
- copyProduct.sh — アカウント間で製品をコピーするための Lambda 関数を作成し、呼び出すスクリプトです。

cross-region-copyパッケージには以下のファイルが含まれています。

- copyconf.properties — リージョン間で製品をコピーするためのリージョンと AWS アカウント ID パラメータを含む設定ファイル。
- scProductCopyLambda.py — リージョン間で製品をコピーするための Python 関数。
- copyProduct.sh — IAM ロールを作成し、リージョン間で製品をコピーするための Lambda 関数を作成して呼び出すスクリプト。

エピック

オプション 1 — アカウント間で AWS Service Catalog 製品をコピー

タスク	説明	必要なスキル
設定ファイルを作成します。	1. `cross-account-copy` パッケージ (添付済み) をローカルマシンにダウンロードします。 2. `copyconf.properties` 設定ファイルを以下の値に更新します。 • `srcRegion` — 製品を含むソースリージョンを指定します。 • `destRegion` — 商品の移行先のリージョンを指定します。 • `sourceAccountId` — 移行元アカウントの AWS アカウント ID を指定します。 • `destAccountId` — 移行先のアカウントの AWS アカウント ID を指定します。	AWS 管理者、クラウド管理者、AWS システム管理者
ソースアカウントで AWS CLI の認証情報を設定します。	`aws configure` コマンドを実行して次の値を指定して、宛先アカウントで AWS CLI にアクセスするための認証情報を設定します。 `$aws configure`	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル

	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]: Region`
	`Default output format [None]:`
	
	詳細については、AWS コマンドラインインターフェイスドキュメントの設定の基本 を参照してください。	

ソースアカウントで AWS CLI の認証情報を設定します。	`aws configure` コマンドを実行して次の値を指定して、ソースアカウントで AWS CLI にアクセスするための認証情報を設定します。
	
	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]: Region`
	`Default output format [None]:`
	
	詳細については、AWS コマンドラインインターフェイスドキュメントの設定の基本 を参照してください。
		AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル
宛先アカウントで Lambda 実行ロールを作成します。	`createDestAccountRole.sh` 移行先アカウントでスクリプトを実行します。スクリプトには、以下のアクションが含まれます。 • 移行先アカウントで Lambda 実行ロールを作成します。 • Lambda 実行ロールの IAM ポリシーを作成してアタッチします。	AWS 管理者、クラウド管理者、AWS システム管理者
ソースアカウントにクロスアカウント IAM ロールを作成します。	`createSrcAccountRole.sh` ソースアカウントでスクリプトを実行します。スクリプトには、以下のアクションが実施されます。 • 製品をコピーするために宛先アカウントの Lambda 実行ロールが引き受けるクロスアカウント IAM ロールをソースアカウントに作成します。 • ソースアカウントのクロスアカウントロールの IAM ポリシーを作成してアタッチします。	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル
移行先のアカウントで CopyProduct スクリプトを実行します。	`copyProduct.sh` 移行先アカウントでスクリプトを実行します。スクリプトには、以下のアクションが実施されます。 • Lambda 関数を作成して呼び出して、ソースアカウントからターゲットアカウントに製品をコピーします。	AWS 管理者、クラウド管理者、AWS システム管理者

オプション 2 — ソースリージョンからターゲットリージョンに AWS Service Catalog 製品をコピー

タスク	説明	必要なスキル
設定ファイルを作成します。	1. `cross-region-copy` パッケージ (添付済み) をローカルマシンにダウンロードします。 2. `copyconf.properties` 設定ファイルを以下の値に更新します。 • `srcRegion` — 製品を含むソースリージョンを指定します。 • `destRegion` — 商品の移行先のリージョンを指定します。 • `accountId` — AWS アカウント ID を指定します。	AWS 管理者、クラウド管理者、AWS システム管理者
AWS 認証情報で CLI を設定します。	`aws configure` コマンドを実行して次の値を指定して、	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル

	ソースアカウントで AWS CLI にアクセスするための認証情報を設定します。
	
	`$aws configure`
	`AWS Access Key ID`
	` [None]: <your_acc`
	`ess_key_id>`
	`AWS Secret Access Key`
	` [None]: <your_sec`
	`ret_access_key>`
	`Default region name`
	` [None]: Region`
	`Default output format`
	` [None]:`
	
	詳細については、AWS コマンドラインインターフェイスドキュメントの設定の基本 を参照してください。	

CopyProductスクリプトを実行します。	移行先リージョンで `copyProduct.sh` スクリプトを実行します。スクリプトには、以下のアクションが実施されます。 • Lambda 実行ロールを作成する • Lambda 実行ロールの IAM ポリシーを作成してアタッチします。 • Lambda 関数を作成して呼び出して、ソースリージョンからターゲットリージョンに製品をコピーします。	AWS 管理者、クラウド管理者、AWS システム管理者

関連リソース

- Lambda 実行ロールを作成 (AWS Lambda ドキュメント)
- ラムダ関数を作成 (AWS Lambda ドキュメント)
- AWS Service Catalog API リファレンス
- AWS Service Catalog ドキュメント

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

クラウド運用モデルの RACI または RASCI マトリックスを作成

作成者：Teddy Germade (AWS), Jerome Descreux (AWS), Josselin LE MINEUR (AWS), と Florian Leroux (AWS))

概要

Cloud Center of Excellence (CCoE) またはCEE (クラウドイネーブルメントエンジン) は、クラウドの運用準備にフォーカスして、権限を与えられ、説明責任を果たせるチームです。彼らの主なフォーカスは、情報IT組織をオンプレミスの運用モデルからクラウドの運用モデルに変革することです。CCoE は、インフラストラクチャ、アプリケーション、運用、セキュリティの代表者を含む部門横断型のチームです。

クラウド運用モデルの主要コンポーネントの 1 つは RACI マトリックス または RASCI マトリックス.です。このデータを使用して、移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義します。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任(R)、説明責任(A)、サポート(S)、協議(C)、情報提供(I)に由来します。サポートタイプは任意です。これを含める場合、 RASCI マトリックスと呼ばれ、サポートを除外すると、 RACI マトリックスと呼ばれます。

添付のテンプレートから始めれば、CCoE チームは組織の RACI または RASCI マトリックスを作成できます。テンプレートには、クラウド運用モデルに共通するチーム、役割、タスクが含まれています。このマトリックスの基礎は、運用統合と CCoE 機能に関連するタスクです。ただし、組織の構造やユースケースのニーズに合わせてこのテンプレートをカスタマイズできます。

RACI マトリックスの実装には制限はありません。このアプローチでは、大規模な組織、新興企業、およびその間のあらゆる組織に有効です。小規模な組織では、同じリソースが複数の役割を担うことがあります。

エピック

マトリックスを作成

タスク	説明	必要なスキル
主要な利害関係者を識別します。	クラウド運用モデルの戦略的目標に関連する主要なサービスマネージャーとチームマネージャーを識別します。	プロジェクトマネージャー

タスク	説明	必要なスキル

マトリックステンプレートをカスタマイズします。	添付ファイル セクションでテンプレートをダウンロードし、RACI または RASCI マトリックスを次のように更新します。
	• Cloud Teams のワークシートでは、組織の必要に応じて CCoE ストリーム名、チーム名、チームの説明を更新します。
	• Cloud Rolesワークシートで、組織の必要に応じて役割、チーム名、役割の説明を更新します。
	• RASCI ワークシートで、組織の必要に応じて以下を更新します。
	  • 行 1 と列 A で、CCoE ストリームを更新します。
	  • 行 2 では、チーム名を更新します。
	  • 行 3 の役割名を更新します。
	  • D 列と E 列で、RASCI チャートに含めたい一般的な項目とアクティビティを更新します。
		プロジェクトマネージャー

タスク	説明	必要なスキル
会議の計画を立てます。	1. RASCIの目的をすべての利害関係者に伝えます。 2. 各チームから権限を与えられた代表者が出席できるように、1 回以上の会議を計画します。	プロジェクトマネージャー

タスク	説明	必要なスキル
マトリックスを完成します。	すべての利害関係者との会議では、以下を実行します。 1. 各チームの代表者が出席していることを確認します。各タスクに責任タイプを正確に割り当てるには、チームの参加が必須です。 2. RASCIマトリックスとは何か、またその目的を参加者と確認します。 3. 参加者と責任分担モデル をレビューし、参加者がクラウドのセキュリティに対する組織の責任範囲を理解できるようにします。 4. RASCI ワークシートの各タスクまたはアクティビティについて、F列からAN列に記入して、以下の責任タイプをアサインします。 • 責任 (R) – この役割では、タスクを完了するための作業を実行する責任があります。 • 説明責任 (A) – この役割には、タスクを確実に完了させる責任があります。この役割は、前提条件が満たされていることを確認し、責任者にタスクを委任する責任もあります。	プロジェクトマネージャー

タスク	説明	必要なスキル
	• サポート(S) – この役割は、担当者がタスクを完了できるよう支援します。この責任タイプは任意ですが、従来の RACI マトリックスを作成するために除外することもできます。 • 協議(C) – タスクに関する意見や専門知識については、この職種に相談する必要があります。タスクによっては、この責任タイプは必要ない場合もあります。 • 情報提供(I) – この役割にはタスクの進捗状況を常に了解し、タスクが完了したら通知を受信する必要があります。 • 空白 – この役割はアクティビティやタスクには関与していません。	
RASCI マトリックスを共有します。	RACI または RASCI マトリックスが完成した後、経営陣の承認を取得します。すべての利害関係者がアクセスできる共有リポジトリまたは一元的な場所に保存します。マトリックスの改訂を記録して承認するには、標準の文書管理プロセスを使用することを推薦します。	プロジェクトマネージャー

関連リソース

- AWS責任共有モデル

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon CloudWatch 異常検知により、カスタムメトリクスのアラームを作成

作成者：Ram Kandaswamy (AWS) と Raheem Jiwani (AWS)

概要

Amazon Web Services (AWS) クラウドでは、Amazon CloudWatch を使用して、メトリクスをモニタリングし、閾値を超えたときに通知を送信し、またはアラームを自動的に変更するアラームを作成できます。

「静的な閾値」による制限を避けるため、過去のパターンに基づいてアラームを作成し、特定のメトリックスが通常の運用時間外になった場合に通知するようにできます。たとえば、Amazon API Gateway から API の応答時間をモニタリングし、サービスレベルアグリーメント (SLA) を満たすことを妨げる異常に関する通知を受信することができます。

このパターンは、CloudWatch 異常検出により、カスタムメトリックスを使用する方法を説明します。このパターンは、Amazon CloudWatch Logs Insights でカスタムメトリクスを作成する方法、または AWS Lambda 関数によりカスタムメトリックスを公開する方法を示しています。次に、Amazon Simple Notification Service (Amazon SNS) により、異常検出を設定し、通知を作成する方法を示しています。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- SNS トピックは、電子メール通知を送信するように設定できます。詳細については、Amazon SNS ドキュメントの「Amazon SNS の使用開始」を参照してください。
- 「CloudWatch Logs」で設定された既存のアプリケーション。

機能制限

- CloudWatch メトリックスはミリ秒の時間間隔をサポートしていません。レギュラーメトリクスとカスタムメトリクスの粒度の詳細については、「Amazon CloudWatchのよくある質問」を参照してください。

アーキテクチャ

この図表は、次のワークフローを示しています：

1. CloudWatch Logs により作成し、更新されたメトリックスを使用するログは CloudWatch にストリーミングされます。
2. アラームは閾値に基づき開始され、SNS トピックにアラートを送信します。
3. Amazon SNS からメール通知が送信されます。

テクノロジースタック

- CloudWatch
- AWS Lambda
- Amazon SNS

ツール

- Amazon CloudWatch は、信頼性、スケーラビリティ、柔軟性に優れたモニタリングソリューションを提供します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードを実行するのに役立つコンピューティングサービスです。
- Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバーへのメッセージ配信を提供するマネージドサービスです。

エピック

カスタムメトリクスの異常検知を設定

タスク	説明	必要なスキル
オプション 1-Lambda 関数により、カスタムメトリクスを作成します。	`lambda_function.py` ファイル (添付) をダウンロードして、「AWS Documentation GitHub」で「aws-lambda-developer-guide」リポジト	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	リにあるサンプル `lambda_function.py` ファイルを置き換えます。これにより、カスタムメトリクスを CloudWatch Logs に送信するサンプル Lambda 関数が得られます。Lambda 関数は Boto3 API により、CloudWatch と統合します。 Lambda 関数を実行した後、AWS マネジメントコンソールにサインインして CloudWatch コンソールを開くと、公開されたメトリクスは公開された名前空間で使用できます。	

タスク	説明	必要なスキル
オプション 2 — CloudWatch ロググループからカスタムメトリックスを作成します。	AWS マネジメントコンソールにサインインし、CloudWatch コンソールを開いて、[ロググループ] を選択します。メトリックを作成するロググループを選択します。 [アクション]、[メトリクスフィルターの作成] の順に選択します。[フィルターパターン] に、使用するフィルターパターンを入力します。詳細については、CloudWatch ドキュメントの「フィルターとパターンの構文」を参照してください。 フィルター パターンをテストするには、[テストパターン] に 1 つ以上のログ イベントを入力します。[ログイベントメッセージ] ボックスのログイベントを区切るために改行が使用されるため、各ログイベントは 1 行以内である必要があります。パターンをテストしたら、メトリックの詳細にメトリックスの名前と値を入力できます。 カスタムメトリクスを作成する手順と詳細については、「CloudWatch のドキュメント」の「ロググループのメト	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
	リクスフィルターの作成」を参照してください。	
カスタムメトリクスのアラームを作成します。	CloudWatch コンソールでアラームとアラームの作成を順に選択します。メトリクスの選択を選択し、前に作成したメトリクスの名前を検索ボックスに入力します。グラフ化したメトリクスタブを選択し、要件に応じてオプションを設定します。 条件で、静的閾値の代わりに異常検知を選択します。これにより、2 つの標準デフォルト偏差に基づくバンドが表示されます。要件に応じて、閾値を設定できます。 [Next (次へ)] を選択します。 **Note** バンドは動的であり、データポイントの品質によって異なります。さらにデータを集約し始めると、バンドと閾値は自動的に更新されます。	DevOps エンジニア、AWS DevOps

タスク	説明	必要なスキル
SNS 通知のセットアップ	[通知] で、アラームが ALARM 状態、OK 状態、または INSUFFICIENT_DATA 状態のときに通知する SNS トピックを選択します。 同じアラーム状態または複数の異なるアラーム状態について複数の通知を送信するには、[Add notification (通知の追加)] を選択します。[Next (次へ)] を選択します。アラームの名前と説明を入力します。名前には ASCII 文字のみを含める必要があります。次いで、[次へ] を選択します。 [プレビューと作成] で、情報と条件が正しいことを確認し、[アラームの作成] を選択します。	DevOps エンジニア、AWS DevOps

関連リソース

- 「カスタムメトリクスの CloudWatch への公開」
- 「CloudWatch 異常検出の使用」
- 「アラームイベントと Amazon EventBridge」
- 「Cloud Watch にカスタムメトリクスをプッシュする際に従うべきベストプラクティスにはどのようなものですか?」(ビデオ)
- 「CloudWatch Application Insightsの紹介」(ビデオ)
- 「CloudWatch による異常検知」(ビデオ)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

デフォルトの暗号化で Amazon EBS ボリュームを使用する AWS Cloud9 IDE を作成

作成者: Janardhan Malyala (AWS)、Dhrubajyoti Mukherjee (AWS)

概要

注意: AWS Cloud9 は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS Cloud9 引き続き使用できます。詳細はこちら

「デフォルトで暗号化」を使用して、Amazon Elastic Block Store (Amazon EBS) ボリュームと Amazon Web Services (AWS) クラウド上のスナップショットコピーを強制的に暗号化できます。

デフォルトで暗号化された EBS ボリュームを使用する AWS Cloud9 統合開発環境 (IDE) を作成できます。ただし、AWS Cloud9 の AWS Identity and Access Management (IAM)「サービスにリンクされたロールには」、これらの EBS ボリュームの AWS Key Management Service (AWS KMS) キーにアクセスする必要があります。アクセスが提供されない場合、AWS Cloud9 IDE の起動に失敗し、デバッグが困難になる場合があります。

このパターンでは、EBS ボリュームで使用される AWS KMS キーにAWS Cloud9のサービスにリンクされたロールを追加する手順を提供します このパターンでデフォルトで暗号化された EBS ボリュームを使用する IDE を正常に作成し、起動するための設定を説明します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- EBS ボリュームではデフォルトの暗号化が有効になっています。デフォルトの暗号化の詳細については、「Amazon Elastic Compute Cloud (Amazon EC2) のドキュメント」の「Amazon EBS 暗号化」を参照してください。
- EBS ボリュームを暗号化するための既存「カスタマーマネージド KMS キー」。

> **Note**
>
> AWS Cloud9 のサービスにリンクされたロールを作成する必要はありません。AWS Cloud9 開発環境を作成すると、AWS Cloud9 によってサービスにリンクされたロールが作成されます。

アーキテクチャ

テクノロジースタック

- AWS Cloud9
- IAM
- AWS KMS

ツール

- 「AWS Cloud9」は、ソフトウェアのコーディング、ビルド、実行、テスト、およびデバッグを支援する統合開発環境 (IDE) です。また、ソフトウェアを AWS クラウドにリリースするのにも役立ちます。
- Amazon Elastic Block Store (Amazon EBS) は、 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで使用するブロックレベルストレージのボリュームを提供します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、データの保護に役立つ暗号キーを作成および管理する上で役立ちます。

エピック

デフォルトの暗号化キー値を検索

タスク	説明	必要なスキル
EBS ボリュームのデフォルトの暗号化キー値を記録します。	AWS マネジメントコンソールにサインインし、Amazon SNS コンソールを開きます。EC2 ダッシュボードを選択し、アカウント属性でデータ保護とセキュリティを選択します。EBS 暗号化セクションで、値をデフォルトの暗号	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	化キーにコピーして記録します。	

AWS KMS キーへのアクセスを提供

タスク	説明	必要なスキル
AWS Cloud9 に EBS ボリュームの KMS キーへのアクセスを提供します。	1. AWS KMS コンソールを開き、カスタマーマネージドキーを選択します。Amazon EBS 暗号化に使用する AWS KMS キーを選択して、キーを表示を選択します。 2. キーポリシータブで、キーポリシーがテキスト形式で表示されることを確認します。テキストフォームが表示されない場合は、ポリシービューに切り替えを選択します。 3. [編集] を選択します。追加情報セクションのコードをポリシーに追加し、[変更を保存] を選択します。ポリシーの変更により、AWS Cloud9 `AWSServiceRoleForAWSCloud9` のサービスにリンクされたロールがキーにアクセスできるようになります。	クラウドアーキテクト、DevOps エンジニア

タスク	説明	必要なスキル
	キーポリシー更新の詳細については、「キーポリシーを変更する方法」(AWS KMS のドキュメント)を参照してください。 ⚠ Important AWS Cloud9 のサービスにリンクされたロールは、最初の IDE を起動すると自動的に作成されます。詳細については、「AWS Cloud9 のドキュメント」の「サービスにリンクされたロールの作成」を参照してください。	

IDE を作成して起動

タスク	説明	必要なスキル
AWS Cloud9 IDE を作成し、起動します。	AWS Cloud9 コンソールを開き、環境を作成を選択します。「AWS Cloud9 のドキュメント」の「EC2 環境の作成」手順に従い、必要に応じて IDE を設定します。	クラウドアーキテクト、DevOps エンジニア

関連リソース

- 「AWS Cloud9 に使用される EBS ボリュームを暗号化する」
- 「AWS Cloud9 のサービスにリンクされたロールの編集」
- 「AWS Cloud9 に EC2 環境の作成」

追加情報

AWS KMS キーポリシーの更新

<aws_accountid> は自分の AWS アカウント ID に置き換えます。

```
{
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<aws_accountid>:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<aws_accountid>:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
```

クロスアカウントキーの使用

クロスアカウント KMS キーを使用する場合は、KMS キーポリシーと組み合わせて許可を使用する必要があります。これにより、キーへのクロスアカウントアクセスが可能になります。Cloud9 環境の作成に使用したのと同じアカウントで、ターミナルで次のコマンドを実行します。

```
aws kms create-grant \
 --region <Region where Cloud9 environment is created> \
 --key-id <The cross-account KMS key ARN> \
 --grantee-principal arn:aws:iam::<The account where Cloud9 environment is
 created>:role/aws-service-role/cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9 \
 --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

このコマンドを実行したら、別のアカウントのキーで EBS 暗号化を使用して Cloud9 環境を作成できます。

タグベースの Amazon CloudWatch ダッシュボードを自動的に作成する

作成者: Janak Vadaria (AWS)、RAJNEESH TYAGI (AWS)、Vinodkumar Mandalapu (AWS)

概要

さまざまな Amazon CloudWatch ダッシュボードを手動で作成すると、特に環境を自動的にスケーリングするために複数のリソースを作成および更新する必要がある場合、時間がかかることがあります。CloudWatch ダッシュボードを自動的に作成および更新するソリューションを使用すると、時間を節約できます。このパターンは、タグ変更イベントに基づいて AWS リソースの CloudWatch ダッシュボードを作成および更新する完全自動化された AWS Cloud Development Kit (AWS CDK) パイプラインをデプロイして、Golden Signals メトリクスを表示するのに役立ちます。

サイト信頼性エンジニアリング (SRE) では、Golden Signals は、ユーザーまたはコンシューマーの観点からサービスの広範なビューを提供する包括的なメトリクスのセットを指します。これらのメトリクスは、レイテンシー、トラフィック、エラー、飽和度で構成されます。詳細については、 AWS ウェブサイトの「サイト信頼性エンジニアリング (SRE) とは」を参照してください。

このパターンが提供するソリューションはイベント駆動型です。デプロイ後、タグ変更イベントを継続的にモニタリングし、CloudWatch ダッシュボードとアラームを自動的に更新します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- v2 AWS CDK の前提条件
- のブートストラップされた環境 AWS
- Python バージョン 3
- AWS SDK for Python (Boto3)、インストール済み
- Node.js バージョン 18 以降
- ノードパッケージマネージャー (npm)、 用にインストールおよび設定 AWS CDK
- AWS CDK および について中程度 (レベル 200) に精通している AWS CodePipeline

制約事項

このソリューションは現在、次の AWS サービスのみの自動ダッシュボードを作成します。

- Amazon Relational Database Service (Amazon RDS)
- AWS Auto Scaling
- Amazon Simple Notification Service (Amazon SNS)
- Amazon DynamoDB
- AWS Lambda

アーキテクチャ

ターゲットテクノロジースタック

- CloudWatch ダッシュボード
- CloudWatch アラーム

ターゲット アーキテクチャ

1. 設定されたアプリケーション AWS タグまたはコード変更のタグ変更イベントは、 でパイプラインを開始し AWS CodePipeline 、更新された CloudWatch ダッシュボードを構築およびデプロイします。
2. AWS CodeBuild は Python スクリプトを実行してタグを設定したリソースを検索し、リソース IDsのローカルファイルに保存します。 CodeBuild
3. CodeBuild は cdk 同期を実行して、CloudWatch ダッシュボードとアラームをデプロイする AWS CloudFormation テンプレートを生成します。
4. CodePipeline は、指定された AWS アカウント およびリージョンに AWS CloudFormation テンプレートをデプロイします。
5. AWS CloudFormation スタックが正常にデプロイされると、CloudWatch ダッシュボードとアラームを表示できます。

自動化とスケール

このソリューションは、 を使用して自動化されています AWS CDK。コードは、Amazon CloudWatch リポジトリの GitHub Golden Signals Dashboards にあります。 Amazon CloudWatch

追加のスケーリングとカスタムダッシュボードを作成するには、複数のタグキーと値を設定できます。

ツール

Amazon サービス

- Amazon EventBridge は、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなど、さまざまなソースからのリアルタイムデータにアプリケーションを接続するのに役立つサーバーレスイベントバスサービスです AWS アカウント。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンドを通じて AWS のサービスを操作するのに役立つオープンソースツールです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

ベストプラクティス

セキュリティのベストプラクティスとして、パイプラインに接続するソースリポジトリに暗号化と認証を使用できます。その他のベストプラクティスについては、CodePipeline ドキュメントの「CodePipeline のベストプラクティスとユースケース」を参照してください。 CodePipeline

エピック

サンプルアプリケーションを設定してデプロイする

タスク	説明	必要なスキル
サンプルアプリケーションを設定してデプロイします。	1. コマンドを使用してGitHub サンプルコードリポジト	AWS DevOps

タスク	説明	必要なスキル
	リのクローンを作成します。 `git clone https://github.com/aws-samples/golden-signals-dashboards-sample-app` 2. コンピュータのクローンされたリポジトリに移動し、任意の工ディタで `src/project-settings.ts` ファイルを開きます。 3. AWS リソースタグとアプリケーションマッピングに従って`projectSettings` 定数値を変更します。 4. `AWS_ACCOUNT` 、`AWS_REGION` 、および `GS_DASHBOARD_INSTANCE` 環境変数を設定します。 • を AWS アカウントのアカウント ID `AWS_ACCOUNT` に設定します。 • サンプルアプリケーションをデプロイするリージョン`AWS_REGION` に設定します。 • 開発環境に応じて`test`、`GS_DASHBOARD_INSTANCE` を `devprod`、、ま	

タスク	説明	必要なスキル
	たは に設定します。(このパターンで説明されているテスト手順testにはをお勧めします)。 5. AWS 認証情報 AWS CLI を使用して をセットアップします。詳細については、 AWS CLI ドキュメントの「 コマンドを使用した設定の設定と表示」を参照してください。 6. 次のコマンドを実行して、Golden Signals ダッシュボードサンプルアプリケーションをデプロイします。 `sh deploy.sh`	

タスク	説明	必要なスキル
ダッシュボードとアラームを自動的に作成します。	サンプルアプリケーションをデプロイしたら、このソリューションがサポートする任意のリソースを、指定されたダッシュボードとアラームを自動的に作成する想定タグ値で作成できます。 このソリューションをテストするには、 AWS Lambda 関数を作成します。 1. サンプルアプリケーションをデプロイ AWS リージョン した AWS Management Console の にサインインします。 2. Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。 3. 関数の作成 を選択し、関数名を入力します。 4. 詳細設定ペインで、タグを有効にするを選択し、新しいタグの追加を選択します。次のキーと値を入力します。 • キー: `AutoDashboard` • 値: `True` 5. [Create function (関数の作成)] を選択します。 Lambda 関数はコードパイプラインをすぐに開始し、	AWS DevOps

タスク	説明	必要なスキル
	その特定の Lambda 関数のダッシュボードとアラームを自動的に作成します。 6. 自動ダッシュボードとアラームを表示するには、「https://https://console.aws.amazon.com/cloudwatch/.com」で CloudWatch コンソールを開きます。`projectSettings` 定数 (デフォルトでは APP1-lambda) で指定した関数のカスタムダッシュボードとアラームを表示できます。 7. Lambda 関数のダッシュボードを選択すると、このソリューションの一部として作成された追加の自動ダッシュボードが表示されます。 8. Amazon RDS、Amazon SNS、DynamoDB などの他の サービスに対してこれらのステップを繰り返し AWS Auto Scalingて、関連付けられたダッシュボードを生成します。Amazon RDS の例については、「追加情報」セクションを参照してください。	

サンプルアプリケーションを削除する

タスク	説明	必要なスキル
golden-signals-dashboard コンストラクトを削除します。	1. サンプルアプリケーションによって作成されたすべての AWS CloudFormation スタックを削除するには、AWS_ACCOUNT 、AWS_REGION 、および GS_DASHBOARD_INSTANCE 環境変数を再設定する必要があります。destroy.sh コマンドには、これらの設定が必要です。 • AWS_ACCOUNT は、アカウントの AWS アカウント ID です。 • AWS_REGION は、サンプルアプリケーションをデプロイしたリージョンです。 • GS_DASHBOARD_INSTANCE はdev、以前の設定に基づいて prod、、testまたは です。 2. AWS 認証情報 AWS CLI を使用して をセットアップします。 3. 次のコマンドを実行して、サンプルアプリケーションおよび関連するすべての AWS CloudFormation スタックを削除します。	AWS DevOps

タスク	説明	必要なスキル
	`sh destroy.sh`	

トラブルシューティング

問題	ソリューション
Python コマンドが見つかりません (`findresources.sh` 「」、8 行目を参照) 。	Python インストールのバージョンを確認します。Python バージョン 3 をインストールしている場合は、 を`resources.sh` ファイルの 8 行`python3`目の `python`に置き換え、`sh deploy.sh` コマンドを再度実行してソリューションをデプロイします。

関連リソース

- ブートストラップ (AWS CDK ドキュメント)
- 名前付きプロファイルの使用 (AWS CLI ドキュメント)
- AWS CDK ワークショップ

追加情報

次の図は、このソリューションの一部として作成された Amazon RDS のサンプルダッシュボードを示しています。

AWS ランディングゾーンの設計を文書化する

作成者: Michael Daehnert (AWS)、Florian Langer (AWS)、および Michael Lodemann (AWS)

概要

ランディングゾーンは、セキュリティとコンプライアンスのベストプラクティスに基づく、適切に設計されたマルチアカウント環境です。これは、すべての組織単位 (OUs)、ユーザー AWS アカウント、その他のリソースを保持するエンタープライズ全体のコンテナです。ランディングゾーンは、あらゆる規模のエンタープライズのニーズに合わせてスケールできます。 AWS には、 を使用するサービスベースのランディングゾーンAWS Control Towerと、構築するカスタマイズされたランディングゾーンの 2 つのオプションがあります。オプションごとに異なるレベルの AWS 知識が必要です。

AWS ランディングゾーンのセットアップを自動化することで時間を節約 AWS Control Tower するために作成された 。 AWS Control Tower は によって管理 AWS され、 のベストプラクティスとガイドラインを使用して基盤環境を作成します。 は、 AWS Service Catalogや などの統合サービス AWS Control Tower を使用してランディングゾーンにアカウントをプロビジョニングしAWS Organizations、それらのアカウントへのアクセスを管理します。

AWS ランディングゾーンプロジェクトは、要件、実装の詳細、運用アクション項目によって異なります。ランディングゾーンの実装ごとに処理する必要があるカスタマイズの側面があります。これには、アクセス管理の処理方法、使用されるテクノロジースタック、運用上の優秀性のモニタリング要件が含まれます (ただし、これらに限定されません)。このパターンは、ランディングゾーンプロジェクトを文書化するのに役立つテンプレートを提供します。テンプレートを使用すると、プロジェクトをより迅速に文書化し、開発チームと運用チームがランディングゾーンを理解するのに役立ちます。

前提条件と制限

制約事項

このパターンでは、ランディングゾーンとは何か、またはランディングゾーンを実装する方法を説明しません。これらのトピックの詳細については、「関連リソース」セクションを参照してください。

エピック

設計ドキュメントを作成する

タスク	説明	必要なスキル
主要な利害関係者を識別します。	ランディングゾーンにリンクされている主要なサービスとチームマネージャーを特定します。	プロジェクトマネージャー
テンプレートをカスタマイズします。	添付ファイルセクションでテンプレートをダウンロードし、次のようにテンプレートを更新します。 1. 組織のランディングゾーンまたはプロセスに適用されないセクションをすべて削除します。 2. 組織に固有のセクションを追加します。	プロジェクトマネージャー
テンプレートを完了します。	ステークホルダーとの会議で、またはwrite-and-reviewのプロセスを使用して、次のようにテンプレートを完了します。 1. 青いボックスのガイダンスと情報を使用して、各セクションを完了します。 2. 黄色のフィールドを組織のカスタム値に置き換えるか削除します。 3. イメージフィールドをカスタムアーキテクチャまたは	プロジェクトマネージャー

タスク	説明	必要なスキル
	フロー図に置き換えるか削除します。 4. テンプレートのリビジョン履歴と寄稿者セクションに入力します。	
設計ドキュメントを共有します。	ランディングゾーン設計ドキュメントが完了したら、すべての利害関係者がアクセスできる共有リポジトリまたは一元的な場所に保存します。標準ドキュメント管理プロセスを使用して、設計ドキュメントのリビジョンを記録および承認することをお勧めします。	プロジェクトマネージャー

関連リソース

- AWS Control Tower ドキュメント
  - AWS Control Tower ランディングゾーンを計画する
  - AWSAWS Control Tower ランディングゾーンのマルチアカウント戦略
  - ランディングゾーンのセットアップに関する管理上のヒント
  - ランディングゾーン設定の期待
- （AWS ソリューションライブラリ) のカスタマイズ AWS Control Tower
- 安全でスケーラブルなマルチアカウント AWS 環境のセットアップ (AWS 規範ガイダンス)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS CDK を使用して複数の AWS リージョン、アカウント、OU にわたって Amazon DevOps Guru を有効にすることで、運用パフォーマンスを向上させます。

作成者：Dr. Rahul Sharad Gaikwad (AWS)

概要

このパターンは、TypeScript の AWS Cloud Development Kit (AWS CDK) を使用して、複数の Amazon Web Services (AWS) リージョン、アカウント、および組織単位 (OU) にわたって Amazon DevOps Guru サービスを有効にする手順を示しています。各アカウントにログインしてアカウントごとに個別に DevOps Guru を有効にする代わりに、AWS CDK スタックを使用して管理者 (プライマリ) AWS アカウントから AWS CloudFormation StackSets をデプロイし、複数のアカウントで Amazon DevOps Guru を有効にすることができます。

Amazon DevOps Guru は、アプリケーションの可用性を向上させ、運用上の問題をより迅速に解決するのに役立つ人工知能運用 (AIOps) 機能を提供します。DevOps Guru は、機械学習 (ML) を活用したレコメンデーションを適用することで、機械学習の専門知識を必要とせずに手作業を軽減します。DevOps Guru はリソースと運用データを分析します。異常を検出すると、問題への対処に役立つ指標、イベント、および推奨事項が表示されます。

このパターンでは、Amazon DevOps Guru を有効にするための 3 つのデプロイオプションについて説明します。

- 複数のアカウントとリージョンですべてのスタックリソース用
- OU のすべてのスタックリソース用
- 複数のアカウントとリージョンで特定のスタックリソース用

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS コマンドラインインターフェイス (AWS CLI)は、「インストール」および「設定」されています。(AWS CLI ドキュメントの「AWS CLI のインストール、更新、アンインストール」を参照してください)。

- AWS CDK ツールキット、インストールおよび設定 (AWS CDK ドキュメントの AWS CDK ツールキットを参照してください)。
- ノードPackage マネージャー (npm)。TypeScript で AWS CDK 用にインストールおよび構成されています。詳細については、npm ドキュメントの [Node.js と npm のダウンロードとインストール] を参照してください。
- Python3 をインストールして設定し、Python スクリプトを実行してサンプルのサーバーレスアプリケーションにトラフィックを注入します。(Python ドキュメントの「Python の設定と使用方法」を参照してください)。
- Pip、Python リクエストライブラリをインストールするようにインストールおよび設定されています。(PyPL Web サイトの「pip インストール手順」を参照してください)。

製品バージョン

- AWS CDK ツールキットバージョン 1.107.0 またはそれ以降
- バージョン 7.9.0 以降
- Node.js バージョン 15.3.0 以降

アーキテクチャ

テクノロジー

このパターンのアーキテクチャには以下のサービスが含まれます。

- Amazon DevOps Guru
- AWS CloudFormation
- Amazon API Gateway
- AWS Lambda
- Amazon DynamoDB
- Amazon CloudWatch
- AWS CloudTrail

AWS CDK スタック

このパターンでは、次の AWS CDK スタックを使用します。

- `CdkStackSetAdminRole`— AWS Identity and Access Management (IAM) 管理者ロールを作成して、管理者とターゲットアカウントの間に信頼関係を確立します。
- `CdkStackSetExecRole`— 管理者アカウントを信頼する IAM ロールを作成します。
- `CdkDevopsGuruStackMultiAccReg`— すべてのスタックで DevOps Guru を有効にし、すべてのスタックで DevOps Guru を有効にし、Amazon Simple Notification Service (Amazon SNS) 通知を設定します。
- `CdkDevopsGuruStackMultiAccRegSpecStacks`— 特定のスタックの複数の AWS リージョンとアカウントで DevOps Guru を有効にし、Amazon SNS 通知を設定します。
- `CdkDevopsguruStackOrgUnit`— 複数の OU で DevOps Guru を有効にし、Amazon SNS 通知を設定します。
- `CdkInfrastructureStack`— API ゲートウェイ、Lambda、DynamoDB などのサンプルサーバーレスアプリケーションコンポーネントを管理者アカウントにデプロイして、フォールトインジェクションとインサイト生成を実演します。

サンプルアプリケーションのアーキテクチャ

次の図は、複数のアカウントとリージョンでデプロイされたサンプルサーバーレスアプリケーションのアーキテクチャを示しています。このパターンでは、管理者アカウントを使用してすべての AWS CDK スタックをデプロイします。また、管理者アカウントを DevOps Guru をセットアップするためのターゲットアカウントの 1 つとして使用します。

1. DevOps Guru を有効にすると、まず各リソースの動作のベースラインを設定し、次に CloudWatch が販売するメトリクスから運用データを取り込みます。
2. 異常を検出すると、CloudTrail からのイベントと関連付けてインサイトを生成します。
3. このインサイトでは、相関関係のある一連のイベントと規定された推奨事項が提供されるため、オペレーターは犯人のリソースを特定できます。
4. Amazon SNS は通知メッセージをオペレータに送信します。

自動化とスケール

このパターンで提供される「GitHub リポジトリ」は、AWS CDK をinfrastructure as code (IaC ツールとして使用して、このアーキテクチャの設定を作成します。AWS CDK は、リソースを調整し、複数の AWS アカウント、リージョン、OU にわたって DevOps Guru を有効にするのに役立ちます。

ツール

AWS サービス

- AWS CDK — AWS Cloud Development Kit (AWS CDK) は、サポートされている 5 つのプログラミング言語 (TypeScript、JavaScript、Python、Java、C#) のいずれかのコードとしてクラウドインフラストラクチャを定義するのに役立ちます。
- AWS CLI — AWS コマンドラインインターフェイス (AWS CLI) は、AWS のサービスやリソースを操作するための一貫したコマンドラインインターフェイスを提供する統合ツールです。

コード

このパターンのソースコードは、GitHub の「Amazon DevOps Guru」CDK サンプルリポジトリにあります。AWS CDK コードはTypeScript で記述されています。リポジトリを複製して使用するには、次のセクションの指示に従います。

> **Important**
>
> このパターンのストーリーには、Unix、Linux、macOS 用にフォーマットされた AWS CDK および AWS CLI コマンドの例が含まれます。Windows の場合は、各行末のバックスラッシュ (\) Unix 連結文字をキャレット (^) に置き換えてください。

エピック

デプロイ用の AWS リソースの準備

タスク	説明	必要なスキル
AWS の名前付きプロファイルを設定します。	マルチアカウント環境にスタックをデプロイするには、以下のように AWS の名前付きプロファイルを設定します。 管理者アカウントの変更 `$aws configure --profile administrator`	DevOps エンジニア

タスク	説明	必要なスキル

	<pre>AWS Access Key ID
	 [****]: <your-adm
	inistrator-access-
	key-ID>
	AWS Secret Access
	 Key [****]: <your-adm
	inistrator-secret-
	access-key>
	Default region name
	 [None]: <your-adm
	inistrator-region>
	Default output format
	 [None]: json</pre>
	ターゲットアカウントの場合：
	<pre>$aws configure --profile
	 target
	AWS Access Key ID
	 [****: <your-target-
	access-key-ID>
	AWS Secret Access Key
	 [****]: <your-target-
	secret-access-key>
	Default region name
	 [None]: <your-target-
	region>
	Default output format
	 [None]: json</pre>
	詳細については、AWS CLI ドキュメントの「名前を指定されたプロファイルを使用する」を参照してください。	

タスク	説明	必要なスキル
AWS プロファイル設定を確認します。	(オプション) AWS CLI ドキュメントの「設定の設定と表示」の手順に従って、`credentials` および`config`ファイル内の AWS プロファイル設定を確認できます。	DevOps エンジニア
AWS CDK のバージョンを確認してください。	次のコマンドを実行して、AWS CDK Toolkit のバージョンを確認します。 `$cdk --version` このパターンには、バージョン 1.107.0 以降が必要です。以前のバージョンの AWS CDK を使用している場合は、「AWS CDK ドキュメント」の指示に従って更新してください。	DevOps エンジニア
プロジェクトコードをクローニングします。	以下のコマンドを使用して、このパターンの GitHub リポジトリをクローンします。 `$git clone https://github.com/aws-samples/amazon-devopsguru-cdk-samples.git`	DevOps エンジニア

タスク	説明	必要なスキル

パッケージの依存関係をインストールし、TypeScript ファイルをコンパイルします。	パッケージの依存関係をインストールし、次のコマンドを実行して TypeScript ファイルをコンパイルします。
	`$cd amazon-devopsguru-cdk-samples`
	`$npm install`
	`$npm fund`
	これらのコマンドは、すべてのパッケージをサンプルリポジトリからインストールします。
	Important
	パッケージの欠落に関するエラーが発生した場合は、次のいずれかのコマンドを使用します。
	`$npm ci`
	—または—
	`$npm install -g @aws-cdk/<package-name>`
	パッケージ名とバージョンのリストは、`/amazon-devopsguru-cdk-samples/package.json` ファ
		DevOps エンジニア

タスク	説明	必要なスキル
	イルのDependencies セクションにあります。詳細については、npm ドキュメントの「npm ci」と「npm install」を参照してください。	

AWS CDK スタックをビルド (合成) する

タスク	説明	必要なスキル
Amazon SNS 通知用の E メールアドレスを設定します。	Amazon SNS 通知用の E メールアドレスを指定するには、次の手順に従います。 1. ファイル/amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-multi-acc-reg-stack.ts と/amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-org-uni-stack.ts を編集します. 2. DevOpsGuruTopic 、Subscription セクションで、Endpointパラメーターをメールアドレスで更新します。 3. ファイルを保存して閉じます。	DevOps エンジニア

タスク	説明	必要なスキル

Go プロジェクトを構築します。	以下のコマンドを実行してプロジェクトコードをビルドし、スタックを合成します。
	`npm run build && cdk synth`
	次のような出力が表示されます:
	`$npm run build && cdk synth`
	`> cdk-devopsguru@0.1.0 build`
	`> tsc`
	`Successfully synthesized to ~/amazon-devopsguru-cdk-samples/cdk.out`
	`Supply a stack id (CdkDevopsGuruStackMultiAccReg,CdkDevopsGuruStackMultiAccRegSpecStacks, CdkDevopsguruStackOrgUnit, CdkInfrastructureStack, CdkStackSetAdminRole, CdkStackSetExecRole) to display its template.`
	詳細と手順については、AWS CDK ドキュメントの「初めての AWS CDK アプリケーション」を参照してください。
		DevOps エンジニア

タスク	説明	必要なスキル

AWS CDK スタックを一覧表示します。	以下のコマンドを実行して、全てのAWS CDK スタックをリストアップします。
	`$cdk list`
	コマンドによって、次のリストが表示されます。
	`CdkDevopsGuruStack`
	`MultiAccReg`
	`CdkDevopsGuruSt`
	`ackMultiAccRegSpec`
	`Stacks`
	`CdkDevopsguruStackOr`
	`gUnit`
	`CdkInfrastructureStac`
	`k`
	`CdkStackSetAdminRole`
	`CdkStackSetExecRole`
		DevOps エンジニア

オプション 1-複数のアカウントのすべてのスタックリソースで DevOps Guru を有効にする

タスク	説明	必要なスキル
IAM ロールを作成するための AWS CDK スタックをデプロイします。	このパターンでは、「AWS CloudFormation StackSets」を使用して複数のアカウントにわたってスタックオペレーションを実行します。初めてスタックセットを作成する場合は、次の IAM ロールを作成して、必要な権限を AWS アカウントに設定する必要があります。	DevOps エンジニア

タスク	説明	必要なスキル
	• `AWSCloudFormationStackSetAdministrationRole` • `AWSCloudFormationStackSetExecutionRole` **Note** ロールには、これらの正確な名前が必要です。 1. 次の CLI コマンドを実行して、管理者 (プライマリ) アカウントで IAM `AWSCloudFormationStackSetAdministrationRole` ロールを作成します。 `$cdk deploy CdkStackSetAdminRole --profile administrator` 2. スタックインスタンスを実行したいすべてのターゲットアカウントに IAM `AWSCloudFormationStackSetExecutionRole` ロールを作成します。このロールを作成するには、次の CLI コマンドを実行します。	

タスク	説明	必要なスキル

	<pre>$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile administrator
	
	$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile target</pre>
	詳細については、AWS CloudFormation ドキュメントの「自己管理型のアクセス許可の承認」を参照してください。	

タスク	説明	必要なスキル
複数のアカウントで DevOps Guru を有効にするための AWS CDK スタックをデプロイします。	AWS CDK `CdkDevopsGuruStackMultiAccReg` スタックは、複数のアカウントとリージョンにスタックインスタンスをデプロイするためのスタックセットを作成します。スタックをデプロイするには、指定されたパラメータを使用して次の CLI コマンドを実行します。 `$cdk deploy CdkDevopsGuruStackMultiAccReg \ --profile administrator \ --parameters AdministratorAccountId=<administrator-account-ID> \ --parameters TargetAccountId=<target-account-ID> \ --parameters RegionIds="<region-1>,<region-2>"` 現在、Amazon DevOps Guru は「DevOps Guru に関するよくある質問」に記載されている AWS リージョンでご利用いただけます。	DevOps エンジニア

オプション 2-OU 全体のすべてのスタックリソースで DevOps Guru を有効にする

タスク	説明	必要なスキル
OU ID を抽出します。	「AWS Organizations」コンソールで、DevOps Guru を有効にする組織ユニットの ID を特定します。	DevOps エンジニア
OU のサービス管理権限を有効にします。	アカウント管理に AWS Organizations を使用している場合は、サービス管理のアクセス権限を付与して DevOps Guru を有効にする必要があります。IAM ロールを手動で作成する代わりに、「組織ベースの信頼できるアクセスとサービスにリンクされたロール (SLR」) を使用してください。	DevOps エンジニア
AWS CDK スタックをデプロイして、複数の OU で DevOps の達人を活用しましょう。	AWS CDK `CdkDevopsguruStackOrgUnit` スタックにより、OU 間で DevOps Guru サービスを有効にすることができます。スタックをデプロイするには、指定されたパラメータを使用して次のコマンドを実行します。 `$cdk deploy CdkDevopsguruStackOrgUnit \ --profile administrator \ --parameters RegionIds="<region-1>,<region-2>" \`	DevOps エンジニア

タスク	説明	必要なスキル
	`--parameters OrganizationalUnitIds="<OU-1>,<OU-2>"`	

オプション 3-複数のアカウントにわたる特定のスタックリソースの DevOps Guru を有効にする

タスク	説明	必要なスキル
IAM ロールを作成するための AWS CDK スタックをデプロイします。	最初のオプションで示した必要な IAM ロールをまだ作成していない場合は、まず作成してください。 1. 次の CLI コマンドを実行して、管理者 (プライマリ) アカウントで IAM `AWSCloudFormationStackSetAdministrationRole` ロールを作成します。 `$cdk deploy CdkStackSetAdminRole --profile administrator` 2. スタックインスタンスを実行したいすべてのターゲットアカウントに IAM `AWSCloudFormationStackSetExecutionRole` ロールを作成します。このロールを作成するには、CLI コマンドを実行します。	DevOps エンジニア

タスク	説明	必要なスキル

	<pre>$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile administrator
	
	$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile target</pre>
	詳細については、AWS CloudFormation ドキュメントの「自己管理型のアクセス許可の承認」を参照してください。	

タスク	説明	必要なスキル
既存のスタックを削除します。	最初のオプションを使用してすべてのスタックリソースで DevOps Guru を有効にしている場合は、次のコマンドを使用して古いスタックを削除できます。 `$cdk destroy CdkDevopsGuruStackMultiAccReg --profile administrator` あるいは、スタックを再デプロイするときに `RegionIds` パラメータを変更することで、「スタックはすでに存在している」エラーを回避することができます。	DevOps エンジニア

タスク	説明	必要なスキル
AWS CDK スタックをスタックリストで更新します。	1. /amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-multi-acc-reg-spec-stack.ts ファイルを編集します。 2. Resources 、CloudFormation 、StackNames の下に、DevOps Guru を有効にするスタックを一覧表示します。デモ用にパラメータでCdkInfrastructureStack スタックを指定していますが、このエントリは必要に応じて編集できます。 3. ファイルを保存して閉じます。 4. 以下を実行して、スタックテンプレートを合成して更新する `$cdk synth`	データエンジニア

タスク	説明	必要なスキル
AWS CDK スタックをデプロイして、複数のアカウントにまたがる特定のスタックリソースで DevOps Guru を有効にします。	AWS CDK CdkDevopsGuruStackMultiAccRegSpecStacks スタックにより、DevOps Guru は複数のアカウントにまたがる特定のスタックリソースに対応できるようになります。以下のコマンドを実行して、API をデプロイします。 `$cdk deploy CdkDevopsGuruStackMultiAccRegSpecStacks \ --profile administrator \ --parameters AdministratorAccountId=<administrator-account-ID> \ --parameters TargetAccountId=<target-account-ID> \ --parameters RegionIds="<region-1>,<region-2>"` **Note** オプション 1 用にこのスタックを以前にデプロイした場合は、スタックがすでに存在するエラーを避けるために、 RegionIds パラメータを変更します (使用可能なリージョ	DevOps エンジニア

タスク	説明	必要なスキル
	ンから必ず選択してください）。	

AWS CDK インフラストラクチャスタックをデプロイ

タスク	説明	必要なスキル
サンプルのサーバーレスインフラストラクチャースタックをデプロイします。	AWS CDK `CdkInfrastructureStack` スタックは API ゲートウェイ、Lambda、DynamoDB テーブルなどのサーバーレスコンポーネントをデプロイして、DevOps Guru のインサイトを実証します。以下のコマンドを実行して、API をデプロイします。 `$cdk deploy CdkInfrastructureStack --profile administrator`	DevOps エンジニア
DynamoDB にサンプルレコードを挿入します。	次のコマンドを実行して、DynamoDB テーブルにサンプルレコードを設定します。`populate-shops-dynamodb-table.json` スクリプトの正しいパスを指定します。 `$aws dynamodb batch-write-item \ --request-items file://scripts/pop`	DevOps エンジニア

タスク	説明	必要なスキル

	ulate-shops-dynamodb-table.json \
	  --profile administrator
	
	コマンドによって以下の出力が表示されます。
	
	{
	  "UnprocessedItems": {}
	}	

タスク	説明	必要なスキル
DynamoDB に挿入されたレコードを確認します。	DynamoDB テーブルにpopulate-shops-dynamodb-table.json ファイルのサンプルレコードが含まれていることを確認するには、AWS CDK スタックの出力として公開されている ListRestApiEndpointMonitorOperator API の URL にアクセスします。この URL は、CdkInfrastructureStack スタックの AWS CloudFormation コンソールの「出力」タブにもあります。出力は次の例のようになります： CdkInfrastructureStack.CreateRestApiMonitorOperatorEndpointD1D00045 = https://oure17c5vob.execute-api.<your-region>.amazonaws.com/prod/ CdkInfrastructureStack.ListRestApiMonitorOperatorEndpointABBDB8D8 = https://cdff8icfrn4.execute-api.<your-region>.amazonaws.com/prod/	DevOps エンジニア

タスク	説明	必要なスキル
リソースのベースラインが完了するまでお待ちください。	このサーバーレススタックには、いくつかのリソースがあります。2 時間待ってから次のステップを実行することをお勧めします。このスタックを本番環境にデプロイした場合、DevOps Guru で監視対象として選択したリソースの数によっては、ベースラインが完了するまでに最大 24 時間かかる場合があります。	DevOps エンジニア

DevOps Guru のインサイトの生成

タスク	説明	必要なスキル
AWS CDK インフラストラクチャスタックを更新します。	DevOps Guru のインサイトを試すには、設定を変更して一般的な運用上の問題を再現できます。 1. `/amazon-devopsguru-cdk-samples/lib/infrastructure-stack.ts` ファイルを編集します。 2. `DDB Table`セクションで、DynamoDB テーブルの読み込み容量を 5 から 1 に変更します。 3. ファイルを保存して閉じます。 4. 次のコマンドを実行して、更新された AWS CDK イ	DevOps エンジニア

タスク	説明	必要なスキル

	ンフラストラクチャスタックを合成してデプロイします。
	
	`$cdk synth`
	`$cdk deploy CdkInfrastructureStack --profile administrator`	

タスク	説明	必要なスキル
API に HTTP リクエストを注入します。	HTTP リクエストの形式で ListRestApiMonitorOperatorEndpointxxxx API にイングレストラフィックを注入します。 1. Python スクリプト/amazon-devopsguru-cdk-samples/scripts/sendAPIRequest.py を実行します。 2. ListRestApiMonitorOperatorEndpointxxxx の API リンクでurl変数を更新します。この URL は、AWS CDK 「デプロイ」コマンドの出力または AWS Cloudformation コンソールのスタックの「出力」タブにあります。 3. ファイルを保存して閉じます。 4. 次のコマンドを使用して、Python スクリプトを実行します。 `$python sendAPIRequest.py` 5. 必ず 200 ステータスコードを取得してください。 6. トラフィックを高速で注入するには、複数の (できれば 4 つの) 端末でスクリプ	DevOps エンジニア

タスク	説明	必要なスキル
	トを実行する必要がある場合があります。 7. スクリプトが約 10 分間ループで実行されると、「DevOps Guru」コンソールに運用上のインサイトが表示されます。	
DevOps Guru Insight の表示	標準的な条件下では、DevOps Guru ダッシュボードの継続的なインサイトカウンターにはゼロが表示されます。異常を検出すると、インサイトという形でアラートが生成されます。ナビゲーションペインで「インサイト」を選択すると、概要、集計指標、関連イベント、推奨事項など、異常の詳細が表示されます。インサイトのレビューの詳細については、「Amazon DevOps Guru を使用して AIOps で運用上の洞察を得る」というブログ投稿を参照してください。	DevOps エンジニア

クリーンアップ

タスク	説明	必要なスキル
リソースをクリーンアップして削除する。	このパターンを確認したら、追加料金が発生しないように、作成したリソースを削除	DevOps エンジニア

タスク	説明	必要なスキル

	する必要があります。以下のコマンドを実行します。
	
	`$cdk destroy CdkDevopsGuruStackMultiAccReg --profile administrator`
	`$cdk destroy CdkDevopsguruStackOrgUnit --profile administrator`
	`$cdk destroy CdkDevopsGuruStackMultiAccRegSpecStacks --profile administrator`
	`$cdk destroy CdkInfrastructureStack --profile administrator`
	`$cdk destroy CdkStackSetAdminRole --profile administrator`
	`$cdk destroy CdkStackSetExecRole --profile administrator`
	`$cdk destroy CdkStackSetExecRole --profile target`	

関連リソース

- Amazon DevOps Guru を使用して AIOps による運用上の洞察を得る
- AWS CloudFormation StackSets を使用して、複数のアカウントとリージョンにわたって Amazon DevOps Guru を簡単に設定できます
- DevOps の達人ワークショップ

ブートストラップパイプラインを使用して Account Factory for Terraform (AFT) を実装する

作成者: Vinicius Elias (AWS) と Edgar Costa Filho (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

このパターンは、 の管理アカウントから AWS Control Tower Account Factory for Terraform (AFT) をデプロイするためのシンプルで安全な方法を提供します AWS Organizations。このソリューションの中核となるのは、Terraform パイプラインを作成して AFT 設定を自動化する AWS CloudFormation テンプレートです。このパイプラインは、初期デプロイや後続の更新に簡単に適応できるように設計されています。

セキュリティとデータの整合性は の最優先事項であるため AWS、マネージドインフラストラクチャと設定の状態を追跡する重要なコンポーネントである Terraform 状態ファイルは、Amazon Simple Storage Service (Amazon S3) バケットに安全に保存されます。このバケットには、サーバー側の暗号化やパブリックアクセスをブロックするポリシーなど、いくつかのセキュリティ対策が設定されており、Terraform の状態が不正アクセスやデータ侵害から保護されます。

管理アカウントは環境全体を調整および監視するため、重要なリソースです AWS Control Tower。このパターンは AWS ベストプラクティスに従っており、デプロイプロセスが効率的であるだけでなく、セキュリティとガバナンスの基準にも準拠し、 AWS 環境に AFT をデプロイするための包括的で安全で効率的な方法を提供します。

AFT の詳細については、 AWS Control Tower ドキュメントを参照してください。

前提条件と制限

前提条件

- 少なくとも管理アカウント、ログアーカイブアカウント、監査アカウント、および AFT 管理用の 1 つの追加アカウントを含む基本的な AWS マルチアカウント環境。
- 確立された AWS Control Tower 環境。CloudFormation テンプレートは内部にデプロイされるため、管理アカウントは適切に設定する必要があります。
- AWS 管理アカウントに必要なアクセス許可。S3 バケット、 AWS Lambda 関数、 AWS Identity and Access Management (IAM) ロール、 AWS CodePipeline プロジェクトなどのリソースを作成および管理するには、十分なアクセス許可が必要です。

- Terraform に精通していること。デプロイには Terraform 設定の生成と管理が含まれるため、Terraform の主要な概念とワークフローを理解することは重要です。

制約事項

- アカウントのAWS リソースクォータに注意してください。デプロイでは複数のリソースが作成され、サービスクォータが発生するとデプロイプロセスが妨げられる可能性があります。
- テンプレートは、Terraform および の特定のバージョン用に設計されています AWS のサービス。バージョンのアップグレードまたは変更には、テンプレートの変更が必要になる場合があります。
- テンプレートは、GitHub Enterprise などのセルフマネージドバージョン管理システム (VCS) サービスをサポートしていません。

製品バージョン

- Terraform バージョン 1.6.6 以降
- AFT バージョン 1.11 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- Amazon EventBridge
- IAM
- AWS Lambda
- Amazon S3

ターゲット アーキテクチャ

次の図は、このパターンで説明されている実装を示しています。

ワークフローは、リソースの作成、コンテンツの生成、パイプラインの実行の 3 つの主要なタスクで構成されます。

リソースの作成

このパターンで提供される CloudFormation テンプレートは、テンプレートをデプロイするときに選択したパラメータに応じて、必要なすべてのリソースを作成してセットアップします。少なくとも、テンプレートは次のリソースを作成します。

- AFT を実装するための CodePipeline パイプライン
- AFT 実装に関連付けられている Terraform 状態ファイルを保存する S3 バケット
- Terraform プランを実装し、パイプラインのさまざまな段階でコマンドを適用する 2 つの CodeBuild プロジェクト
- CodeBuild および CodePipeline サービスの IAM ロール
- パイプラインランタイムアーティファクトを保存するための 2 番目の S3 バケット

選択した VCS プロバイダー (CodeCommit または外部 VCS) に応じて、テンプレートは次のリソースを作成します。

- CodeCommit の場合：
  - AFT Terraform ブートストラップコードを保存する CodeCommit リポジトリ
  - `main` ブランチで CodeCommit リポジトリの変更をキャプチャする EventBridge ルール
  - EventBridge ルールの別の IAM ロール
- GitHub などの他の外部 VCS プロバイダーの場合：
  - AWS CodeConnections 接続

さらに、VCS プロバイダーとして CodeCommit を選択すると、 `Generate AFT Files`パラメータを に設定すると`true`、テンプレートはこれらの追加リソースを作成してコンテンツを生成します。

- 生成されたコンテンツを保存し、CodeCommit リポジトリのソースとして使用する S3 バケット
- 指定されたパラメータを処理し、適切なコンテンツを生成する Lambda 関数
- Lambda 関数を実行する IAM 関数
- テンプレートがデプロイされたときに Lambda 関数を実行する CloudFormation カスタムリソース

コンテンツの生成

AFT ブートストラップファイルとそのコンテンツを生成するために、ソリューションは Lambda 関数と S3 バケットを使用します。関数はバケットにフォルダを作成し、フォルダ内に `main.tf`と の 2 つのファイルを作成しますbackend.tf。この関数は、提供された CloudFormation パラメータも処理し、これらのファイルに事前定義されたコードを入力し、それぞれのパラメータ値を置き換えます。

ファイルを生成するテンプレートとして使用されるコードを表示するには、ソリューションの GitHub リポジトリを参照してください。基本的に、ファイルは次のように生成されます。

main.tf

```
module "aft" {
  source = "github.com/aws-ia/terraform-aws-control_tower_account_factory?
ref=<aft_version>"

  # Required variables
  ct_management_account_id  = "<ct_management_account_id>"
  log_archive_account_id    = "<log_archive_account_id>"
  audit_account_id          = "<audit_account_id>"
  aft_management_account_id = "<aft_management_account_id>"
  ct_home_region            = "<ct_home_region>"

  # Optional variables
  tf_backend_secondary_region = "<tf_backend_secondary_region>"
  aft_metrics_reporting       = "<false|true>"

  # AFT Feature flags
  aft_feature_cloudtrail_data_events      = "<false|true>"
  aft_feature_enterprise_support          = "<false|true>"
  aft_feature_delete_default_vpcs_enabled = "<false|true>"

  # Terraform variables
  terraform_version      = "<terraform_version>"
  terraform_distribution = "<terraform_distribution>"

  # VCS variables (if you have chosen an external VCS)
  vcs_provider                                = "<github|githubenterprise|gitlab|
gitlabselfmanaged|bitbucket>"
  account_request_repo_name                   = "<org-name>/aft-account-request"
  account_customizations_repo_name            = "<org-name>/aft-account-
customizations"
  account_provisioning_customizations_repo_name = "<org-name>/aft-account-provisioning-
customizations"
```

```
  global_customizations_repo_name                = "<org-name>/aft-global-
customizations"

}
```

backend.tf

```
terraform {
  backend "s3" {
    region = "<aft-main-region>"
    bucket = "<s3-bucket-name>"
    key    = "aft-setup.tfstate"
  }
}
```

CodeCommit リポジトリの作成中に `Generate AFT Files`パラメータを に設定すると`true`、テンプレートは生成されたコンテンツを`main`ブランチのソースとして S3 バケットを使用してリポジトリに自動的に入力します。

パイプラインの実行

リソースが作成され、ブートストラップファイルが設定されると、パイプラインが実行されます。第 1 ステージ (ソース) はリポジトリのメインブランチからソースコードを取得し、第 2 ステージ (ビルド) は Terraform プランコマンドを実行し、レビューする結果を生成します。3 番目のステージ (承認) では、パイプラインは手動アクションが最後のステージ (デプロイ) を承認または拒否するのを待ちます。最後のステージでは、パイプラインは前の Terraform `apply` コマンドの結果を入力として使用して Terraform `plan` コマンドを実行します。最後に、クロスアカウントロールと管理アカウントのアクセス許可を使用して、AFT 管理アカウントに AFT リソースを作成します。

> **Note**
>
> 外部 VCS プロバイダーを選択した場合は、VCS プロバイダーの認証情報との接続を許可する必要があります。セットアップを完了するには、 AWS 「 デベロッパーツールコンソールドキュメント」の「保留中の接続の更新」のステップに従います。

ツール

AWS サービス

- AWS CloudFormation は、AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じて管理するために役立ちます。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- AWS CodeConnections では、CodePipeline などの AWS リソースとサービスが GitHub などの外部コードリポジトリに接続できます。
- AWS Lambda は、イベントに応じてコードを実行し、コンピューティングリソースを自動的に管理するコンピューティングサービスであり、本番稼働用の最新のサーバーレスアプリケーションをすばやく作成できます。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。

その他のツール

- Terraform は、インフラストラクチャを安全かつ効率的に構築、変更、およびバージョン化できる Infrastructure as Code (IaC) ツールです。これには、コンピューティングインスタンス、ストレージ、ネットワークなどの低レベルのコンポーネント、DNS エントリや SaaS 機能などの高レベルのコンポーネントが含まれます。
- Python は、学習しやすく強力なプログラミング言語です。効率的な高レベルデータ構造を持ち、オブジェクト指向プログラミングにシンプルかつ効果的なアプローチを提供します。

コードリポジトリ

このパターンのコードは、GitHub AFT ブートストラップパイプラインリポジトリで入手できます。

公式 AFT リポジトリについては、GitHub のAWS Control Tower 「Account Factory for Terraform」を参照してください。

ベストプラクティス

提供されている CloudFormation テンプレートを使用して AFT をデプロイする場合は、ベストプラクティスに従って、安全で効率的で正常な実装を確保することをお勧めします。AFT の実装と運用に関する主要なガイドラインと推奨事項は次のとおりです。

- パラメータの詳細なレビュー: CloudFormation テンプレートの各パラメータを慎重に確認し、理解します。AFT を正しくセットアップして機能させるには、正確なパラメータ設定が不可欠です。
- 定期的なテンプレートの更新: テンプレートを最新の AWS 機能と Terraform バージョンで更新します。定期的な更新により、新機能を活用し、セキュリティを維持できます。
- バージョニング: AFT モジュールバージョンを固定し、可能であればテスト用に別の AFT デプロイを使用します。
- スコープ: AFT は、インフラストラクチャガードレールとカスタマイズをデプロイするためにのみ使用します。アプリケーションのデプロイには使用しないでください。
- リントと検証: AFT パイプラインには、リントされ検証された Terraform 設定が必要です。設定を AFT リポジトリにプッシュする前に、lint、検証、テストを実行します。
- Terraform モジュール: 再利用可能な Terraform コードをモジュールとして構築し、常に組織の要件に合わせて Terraform と AWS プロバイダーのバージョンを指定します。

エピック

AWS 環境のセットアップと設定

タスク	説明	必要なスキル
AWS Control Tower 環境を準備します。	AWS 環境 AWS Control Tower 内で をセットアップして設定し、 の集中管理とガバナンスを確保します AWS アカウント。詳細については、AWS Control Tower ドキュメントの「 の開始方法 AWS Control Tower」を参照してください。	クラウド管理者
AFT 管理アカウントを起動します。	AWS Control Tower Account Factory を使用して、AFT 管	クラウド管理者

タスク	説明	必要なスキル
	理アカウント AWS アカウント として機能する新しい を起動します。詳細については、AWS Control Tower ドキュメントのAWS Service Catalog「Account Factory でアカウントをプロビジョニングする」を参照してください。	

管理アカウントに CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
CloudFormation のテンプレートを起動します。	このエピックでは、このソリューションに付属の CloudFormation テンプレートをデプロイして、 AWS 管理アカウントに AFT ブートストラップパイプラインを設定します。パイプラインは、前のエピックで設定した AFT 管理アカウントに AFT ソリューションをデプロイします。 ステップ 1: AWS CloudFormation コンソールを開く • にサインイン AWS Management Console し、AWS CloudFormation コンソールを開きます。正しい AWS Control Tower メインリージョン内で動作していることを確認します。	クラウド管理者

タスク	説明	必要なスキル
	ステップ 2: 新しいスタックを作成する 1. を選択して新しいスタックを作成します。 2. テンプレートファイルをアップロードするオプションを選択し、このパターンで提供されている CloudFormation テンプレートをアップロードします。 ステップ 3: スタックパラメータを設定する • `VCS Provider`: 使用するバージョン管理システム (VCS) プロバイダーを選択します。GitHub などの外部 VCS を選択するか、アカウントでサービスの使用が許可されている場合は CodeCommit を使用できます。 • `Repository Name` : AFT ブートストラップモジュールを保存するためのリポジトリ名を指定します。外部 VCS プロバイダーの場合は、組織名 (など) を含むフルパスを使用しますmy-github-org/my-repo 。	

タスク	説明	必要なスキル

	• Branch Name: ソースリポジトリブランチを指定します。
	• CodeBuild Docker Image: CodeBuild Docker ベースイメージとして使用するファイルを選択します。
	• VCS プロバイダーを CodeCommit 以外のオプションに設定する場合は、ステップ 8 に進みます。
	
	ステップ 4: ファイル生成を決定する
	
	• VCS プロバイダーとして CodeCommit を選択した場合は、 Generate AFT Filesパラメータを使用してデフォルトの AFT デプロイファイルの生成を制御できます。このパラメータを次のように設定します。
	  • true は、指定されたリポジトリに AFT デプロイファイルを自動的に作成して保存します。
	  • false ファイルの作成を手動で処理する場合、またはファイルがすでに配置されている場合。
	• を選択した場合はfalse、ステップ 8 に進みます。	

タスク	説明	必要なスキル

	それ以外の場合は、まずステップ 5～7 に従います。
	
	ステップ 5: アカウントの詳細 AWS Control Tower と AFT アカウントの詳細を入力する
	
	• `Generate AFT Files` パラメータを に設定する場合は`true`、以下の情報 AWS Control Tower と AFT アカウント固有の情報を指定します。
	  • `Log Archive Account ID`: のログアーカイブアカウント ID の ID AWS Control Tower。
	  • `Audit Account ID`: の監査アカウントの ID AWS Control Tower。
	  • `AFT Management Account ID`: 最初の工ピックで作成した AFT 管理アカウントの ID。
	  • `AFT Main Region` および `AFT Secondary Region`: AFT デプロイ AWS リージョン のメインおよびセカンダリ。
	
	ステップ 6: AFT オプションを設定する	

タスク	説明	必要なスキル

	• メトリクスレポートを設定します。
	  • `AFT Enable Metrics Reporting` : AFT メトリクスレポートを有効または無効にします。詳細については、 AWS Control Tower ドキュメントの「運用メトリクス」を参照してください。
	• AFT 機能オプションの設定 :
	  • `Enable AFT CloudTrail Data Events`: すべての AFT マネージドアカウントで CloudTrail データイベントを有効にします。詳細については、 AWS Control Tower ドキュメントAWS CloudTrail のデータイベントを参照してください。
	  • `Enable AFT Enterprise Support` : すべての AFT マネージドアカウントでエンタープライズサポートを有効にします。詳細については、 AWS Control Tower ドキュメントのAWS 「エンタープライズサポート	

タスク	説明	必要なスキル
	プラン」を参照してください。 • `Enable AFT Delete Default VPC`: AFT 管理アカウントのすべての VPCs のみを削除します。詳細については、AWS Control Tower ドキュメントの AWS「デフォルトの VPC を削除する」を参照してください。 ステップ 7: バージョンを指定する • `AFT Terraform Version`: AFT パイプラインで使用する Terraform のバージョンを選択します。 • `AFT Version`: デプロイする AFT バージョンを定義します。最新の AFT バージョンを使用するには、デフォルト設定 (`latest`) のままにします。 ステップ 8: スタックを確認して作成する • すべてのパラメータと設定を確認します。すべてが順不同の場合は、スタックの作成に進みます。	

タスク	説明	必要なスキル
	ステップ 9: スタックの作成をモニタリングする • AWS CloudFormation は、定義したリソースをプロビジョニングして設定します。CloudFormation コンソールでスタック作成プロセスをモニタリングします。このプロセスには数分かかる場合があります。 ステップ 10: デプロイを検証する • スタックのステータスが CREATE_COMPLETE と表示されたら、すべてのリソースが正しく作成されていることを確認します。 • Outputs セクションで、`TerraformBackendBucketName` 値を書き留めます。	

AFT ブートストラップリポジトリとパイプラインの入力と検証

タスク	説明	必要なスキル
オプション 1: 外部 VCS の AFT ブートストラップリポジトリを入力します。	VCS プロバイダーを (CodeCommit ではなく) 外部 VCS に設定する場合は、以下の手順に従います。	クラウド管理者

タスク	説明	必要なスキル
	(オプション) CloudFormation テンプレートをデプロイした後、新しく作成した AFT ブートストラップリポジトリにコンテンツを入力または検証し、パイプラインが正常に実行されたかどうかをテストできます。 ステップ 1: 接続を更新する 1. CodePipeline コンソールのナビゲーションペインで、設定、接続を選択します。 2. `aft-vcs-connection` 接続を選択します。`Pending` ステータスになっている必要があります。 3. 保留中の接続の更新を選択し、「デベロッパーツールコンソールドキュメント」の「保留中の接続の更新」の手順に従います。 4. 接続のステータスが `Available` になったら、次のステップに進みます。 ステップ 2: リポジトリを入力する 1. 外部 VCS 認証情報を使用して、テンプレートで指定したリポジトリをローカルマシンにクローンします。	

タスク	説明	必要なスキル
	デフォルト名を保持した場合、リポジトリは と呼ばれますaft-setup 。 2. リポジトリで、backend.tf と という 2 つの空のファイルを含む という名前terraform のフォルダを作成しますmain.tf。 3. backend.tf ファイルを開き、次のコードスニペットを追加します。 `terraform { backend "s3" { region = "<aft-main-region>" bucket = "<s3-bucket-name>" key = "aft-setup" } }` ファイルで、次の操作を行います。 • をメイン AFT リージョン<aft-main-region>に置き換えます。これは AWS Control Tower メインリージョンと一致する必要があります。 • を Terraform バックエンドバケットの名前<s3-	

タスク	説明	必要なスキル
	bucket-name> に置き換えます。これは、前にデプロイした CloudFormation テンプレートによって生成されたTerraformBackendBucketName 出力にあります。 4. main.tf ファイルを開き、AFT リポジトリで利用可能な例のいずれかを使用して AFT をデプロイします。たとえば、任意の VCS プロバイダー (CodeCommit、GitHub、または Bitbucket) を使用するか、AFT VPC をカスタマイズできます。その他の AFT 入力オプションについては、AFT リポジトリの README ファイルを参照してください。 ステップ 2: 変更をコミットしてプッシュする • フォルダとファイルを作成して入力したら、変更を確認し、コードをリポジトリにアップロードします。パイプラインは自動的に開始され、ソースステージとビルドステージを実行し、デ	

タスク	説明	必要なスキル
	プロイステージの前に承認アクションを待機します。	

タスク	説明	必要なスキル
オプション 2: CodeCommit の AFT ブートストラップリポジトリを入力します。	VCS プロバイダーを CodeCommit に設定する場合は、以下の手順に従います。 (オプション) CloudFormation テンプレートをデプロイした後、新しく作成した AFT ブートストラップリポジトリにコンテンツを入力または検証し、パイプラインが正常に実行されたかどうかをテストできます。 `Generate AFT Files` パラメータを に設定した場合は`true`、次のストーリー (パイプラインの検証) にスキップします。 ステップ 1: リポジトリを入力する 1. AWS CodeCommit コンソールを開き、新しく作成したリポジトリを選択します。デフォルト名を保持した場合、リポジトリ名は である必要があります`aft-setup` 。 2. SSH、HTTPS、または HTTPS (GRC) を使用してリポジトリをローカルマシンにクローンし、エディタで開きます。	クラウド管理者

タスク	説明	必要なスキル

	3. という名前のフォルダterraform と、その中に backend.tf と という 2 つの空のファイルを作成しますmain.tf。
	4. backend.tf ファイルを開き、次のコードスニペットを追加します。
	
	`terraform {`
	`  backend "s3" {`
	`  region = "<aft-main-region>"`
	`  bucket = "<s3-bucket-name>"`
	`  key  = "aft-setup"`
	`  }`
	`}`
	
	ファイルで、次の操作を行います。
	• をメイン AFT リージョン<aft-main-region>に置き換えます。これは AWS Control Tower メインリージョンと一致する必要があります。
	• を Terraform バックエンドバケットの名前<s3-bucket-name> に置き換えます。これは、前にデプロイした CloudFormation テンプレートによって生成されたTerraform	

タスク	説明	必要なスキル
	BackendBucketName 出力にあります。 5. main.tf ファイルを開き、AFT リポジトリで利用可能な例のいずれかを使用して AFT をデプロイします。たとえば、任意のバージョン管理システム (VCS) プロバイダー (CodeCommit、GitHub、または Bitbucket) を使用するか、AFT VPC をカスタマイズできます。その他の AFT 入力オプションについては、AFT リポジトリの README ファイルを参照してください。 ステップ 2: 変更をコミットしてプッシュする • フォルダとファイルを作成して入力したら、変更を確認し、コードをリポジトリにアップロードします。パイプラインは自動的に開始され、ソースステージとビルドステージを実行し、デプロイステージの前に承認アクションを待機します。	

タスク	説明	必要なスキル
AFT ブートストラップパイプラインを検証します。	ステップ 1: パイプラインを表示する • CodePipeline コンソールを開き、`aft-bootstrap-pipeline` パイプラインが正常に開始されたかどうかを確認します。Terraform プランを実行しているか、手動承認アクションを待っている必要があります。 ステップ 2: Terraform 計画結果を承認する • Terraform プランの結果を確認するには、ビルドステージの実行ログを確認し、承認ステージで実行を承認または拒否します。承認すると、パイプラインは指定された AFT 管理アカウントに AFT リソースのデプロイを開始します。 ステップ 3: デプロイを待機する • パイプラインが正常に実行されるまで待ちます。これには約 30 分かかります。発生する可能性のある障害は、多くの場合、API クォータが原因で発生します。このような場合は、パ	クラウド管理者

タスク	説明	必要なスキル
	イプラインを再実行してデプロイを続行できます。 ステップ 4: 作成されたリソースを確認する • AFT 管理アカウントにアクセスし、リソースが作成されていることを確認します。	

トラブルシューティング

問題	ソリューション
CloudFormation テンプレートに含まれるカスタム Lambda 関数は、デプロイ中に失敗します。	Lambda 関数の Amazon CloudWatch logsをチェックして、エラーを特定します。ログは詳細情報を提供し、特定の問題を特定するのに役立ちます。Lambda 関数に必要なアクセス許可があり、環境変数が正しく設定されていることを確認します。
アクセス許可が不十分であるため、リソースの作成または管理に障害が発生します。	Lambda 関数、CodeBuild、およびデプロイに関連するその他のサービスにアタッチされている IAM ロールとポリシーを確認します。必要なアクセス許可があることを確認します。アクセス許可に問題がある場合は、IAM ポリシーを調整して必要なアクセス許可を付与します。
CloudFormation テンプレートの古いバージョンを新しいバージョン AWS のサービス または Terraform バージョンで使用している。	CloudFormation テンプレートを定期的に更新して、最新の AWS および Terraform リリースと互換性を持たせます。バージョン固有の変更や要件については、リリースノートまたはドキュメントを参照してください。

問題	ソリューション
デプロイ中に AWS のサービス クォータに達します。	パイプラインをデプロイする前に、S3 バケット、IAM ロール、Lambda 関数などのリソースの AWS のサービス クォータを確認します。必要に応じて引き上げをリクエストします。詳細については、 AWS ウェブサイトのAWS のサービス クォータを参照してください。
CloudFormation テンプレートの入力パラメータが正しくないため、エラーが発生します。	すべての入力パラメータに誤字や誤った値がないかを再確認します。アカウント IDs やリージョン名などのリソース識別子が正確であることを確認します。

関連リソース

このパターンを正常に実装するには、次のリソースを確認してください。これらのリソースは、 を使用して AFT を設定および管理する際に貴重な追加情報とガイダンスを提供します AWS CloudFormation。

AWS ドキュメント：

- AWS Control Tower ユーザーガイドには、 のセットアップと管理に関する詳細情報が記載されています AWS Control Tower。
- AWS CloudFormation ドキュメントには、CloudFormation テンプレート、スタック、リソース管理に関するインサイトが記載されています。

IAM ポリシーとベストプラクティス：

- IAM のセキュリティのベストプラクティスでは、IAM ロールとポリシーを使用して AWS リソースを保護する方法について説明します。

Terraform オン AWS：

- Terraform AWS Provider ドキュメントには、 での Terraform の使用に関する包括的な情報が記載されています AWS。

AWS のサービス クォータ：

- AWS のサービス クォータには、 AWS のサービス クォータの表示方法と引き上げをリクエストする方法に関する情報が記載されています。

複数の AWS アカウントと AWS リージョンで AWS Service Catalog 製品を管理

作成者: Ram Kandaswamy (AWS)

概要

Amazon Web Services (AWS) Service Catalog は、企業向けの Infrastructure as Code (IaC) テンプレートのガバナンスと配布を簡単かつ迅速にします。AWS CloudFormation テンプレートで、製品に必要な AWS リソース (スタック) のコレクションを定義します。AWS CloudFormation StackSets は、複数のアカウントおよび AWS リージョンにまたがるスタックを単一の操作で作成、更新、削除できるようにすることで、スタックの機能を拡張します。

AWS Service Catalog の管理者は、開発者が作成した CloudFormation テンプレートで製品を作成し、公開します。その後、これらの製品はポートフォリオに関連付けられ、ガバナンスに制約が適用されます。他の AWS アカウントまたは組織単位 (OU) のユーザーが製品を利用できるようにするには、通常、彼らと「ポートフォリオを共有」します。このパターンは、AWS CloudFormation StackSets に基づく AWS Service Catalog 製品の提供を管理するための代替アプローチを示しています。ポートフォリオを共有する代わりに、スタックセット制約により、製品をデプロイして使用できる AWS リージョンとアカウントを設定します。このアプローチを使用すると、ガバナンス要件を満たしながら、AWS Service Catalog 製品を複数のアカウント、OU と AWS リージョンにプロビジョニングし、一元的に管理できます。

この方法の利点:

- 製品はプライマリアカウントからプロビジョニング、管理され、他のアカウントと共有されることはありません。
- この方法では、特定の製品に基づいてプロビジョニングされたすべての製品 (スタック) を統合して表示できます。
- AWS Service Management Connector での設定は、1 つのアカウントのみを対象としているため、より簡単です。
- AWS Service Catalog の製品のクエリと使用が簡単になりました。

前提条件と制限

前提条件

- IaC とバージョニング用の AWS CloudFormation テンプレート

- AWS リソースのプロビジョニングと管理のためのマルチアカウント設定と AWS Service Catalog

機能制限

- このアプローチでは AWS CloudFormation StackSets が使用され、StackSets 制限が適用されます。
  - StackSets は、マクロによる CloudFormation テンプレートのデプロイをサポートしません。マクロでテンプレートを前処理している場合、StackSets ベースのデプロイを使用することはできません。
  - StackSets では、スタックをスタックセットから切り離すことができるため、特定のスタックをターゲットにして問題を解決できます。ただし、関連付けを解除したスタックをスタックセットに再関連付けることはできません。
- AWS Service Catalog は StackSet 名を自動的に生成します。カスタマイズは現在サポートされていません。

アーキテクチャ

ターゲット アーキテクチャ

1. ユーザーは、AWS リソースをプロビジョニングするための AWS CloudFormation テンプレートを JSON 形式または YAML 形式で作成します。
2. CloudFormation テンプレートは AWS Service Catalog に製品を作成し、ポートフォリオに追加します。
3. ユーザーは、ターゲットアカウントに CloudFormation スタックを作成するプロビジョニング済み製品を作成します。
4. 各スタックは、CloudFormationで指定されたリソースをプロビジョニングします。

ツール

AWS サービス

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Service Catalog では、AWS で承認された IT サービスのカタログを一元管理できます。エンドユーザーは、組織によって設定された制約に従って、必要な承認済みの IT サービスのみをすばやくデプロイできます。

エピック

アカウント間での製品のプロビジョニング

タスク	説明	必要なスキル
ポートフォリオを作成します。	ポートフォリオは、特定の基準に基づき、グループ化された 1 つ以上の製品を含むコンテナです。ポートフォリオを製品に使用すると、製品セット全体に共通の制約を適用しやすくなります。 ポートフォリオを作成するには、「AWS Service Catalog ドキュメント」の手順に従ってください。AWS CLI を使用している場合は、コマンドの例を挙げます。 `aws servicecatalog create-portfolio --provider-name my-provider --display-name my-portfolio`	AWS Service Catalog、IAM

タスク	説明	必要なスキル
	詳細については、「AWS CLI ドキュメント」を参照してください。	
CloudFormation テンプレートを作成します。	リソースを説明する CloudFormation テンプレートを作成します。必要に応じて、リソースプロパティをパラメータ化する必要があります	AWS CloudFormation JSON/YAML

タスク	説明	必要なスキル
バージョン情報を使用して製品を作成します。	CloudFormation テンプレートは、AWS Service Catalog に公開すると製品になります。オプションのバージョン詳細パラメータ (バージョンタイトルや説明など) に値を指定することは、後で製品について問い合わせるときに役立ちます。 ポートフォリオを作成するには、「AWS Service Catalog ドキュメント」の手順に従ってください。AWS CLI を使用している場合は、コマンドの例： `aws servicecatalog create-product --cli-input-json file://create-product-input.json` `create-product-input.json` は製品のパラメータを渡すファイルはどこですか。注：ファイルの例として、このパターンの「追加情報」セクションを参照してください。詳細については、「AWS CLI ドキュメント」を参照してください。	AWS Service Catalog

タスク	説明	必要なスキル
制約を適用します。	スタックセットの制約をポートフォリオに適用し、複数の AWS アカウント、リージョン、権限などの製品デプロイオプションを設定します。手順については、「AWS Service Catalog のドキュメント」を参照してください。	AWS Service Catalog
アクセス許可を追加します。	ユーザーがポートフォリオ内の製品を起動できるようにアクセス権限を付与します。コンソールの手順については、「AWS Service Catalog のドキュメント」を参照してください。AWS CLI を使用している場合は、コマンドの例を挙げます。 `aws servicecatalog associate-principal-with-portfolio \ --portfolio-id port-2s6abcdefwdh4 \ --principal-arn arn:aws:iam::444455556666:role/Admin \ --principal-type IAM` 詳細については、「AWS CLI ドキュメント」を参照してください。	AWS Service Catalog、IAM

タスク	説明	必要なスキル
製品をプロビジョニングします。	プロビジョニングされた製品は、製品のリソースインスタンスです。たとえば、CloudFormation テンプレートに基づいて製品をプロビジョニングすると、CloudFormation スタックとその基礎となるリソースが起動します。 スタックセットの制約に基づき、該当する AWS リージョンとアカウントをターゲットにして製品をプロビジョニングします。AWS CLI でのコマンドの例は次のとおりです。 `aws servicecatalog provision-product \ --product-id prod-abcdfz3syn2rg \ --provisioning-artifact-id pa-abc347pcsccfm \ --provisioned-product-name "mytestppname3"` 詳細については、「AWS CLI ドキュメント」を参照してください。	AWS Service Catalog

関連リソース

リファレンス

- 「AWS Service Catalog の概要」

- 「AWS CloudFormation StackSetsの使用」

チュートリアルと動画

- 「AWS re: Invent 2019: すべてを自動化する:オプションとベストプラクティス」(ビデオ)

追加情報

`create-product` コマンドを使用する場合、`cli-input-json` パラメータは製品所有者、サポートメール、CloudFormation テンプレートの詳細などの情報を指定するファイルを指します。以下にそのようなファイルの例を示します。

```
{
   "Owner": "Test admin",
      "SupportDescription": "Testing",
         "Name": "SNS",
            "SupportEmail": "example@example.com",
            "ProductType": "CLOUD_FORMATION_TEMPLATE",
               "AcceptLanguage": "en",
                  "ProvisioningArtifactParameters": {
                     "Description": "SNS product",
                        "DisableTemplateValidation": true,
                           "Info": {
                              "LoadTemplateFromURL": "<url>"
                     },
                           "Name": "version 1"
}
```

AWS メンバーアカウントを AWS Organizations から AWS Control Tower に移行する

作成者: Rodolfo Jr. Cerrada (AWS)

概要

このパターンは、Amazon Web Services (AWS) アカウントを管理アカウントによって管理されるメンバーアカウントである AWS Organizations から AWS Control Tower に移行する方法を説明します。アカウントを AWS Control Tower に登録することで、予防的な検出ガードレールと、アカウントガバナンスを合理化する機能を利用できます。AWS Organizations の管理アカウントが侵害され、メンバーアカウントを AWS Control Tower が管理する新しい組織に移動する場合は、メンバーアカウントも移行する必要があります。

AWS Control Tower は、AWS Organizations を含む複数の AWS サービスの機能を組み合わせて統合するフレームワークを提供し、マルチアカウント環境全体で一貫したコンプライアンスとガバナンスを保証します。AWS Control Tower を使用すると、AWS Organizations の機能を拡張する一連の規定されたルールと定義に従うことができます。たとえば、ガードレールを使用して、セキュリティログと必要なクロスアカウントアクセス許可が作成され、変更されないようにできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Organizations のターゲット組織で AWS Control Tower をセットアップします (手順については、AWS Control Tower のドキュメントのセットアップを参照)。
- AWS Control Tower の管理者認証情報 (AWSControlTowerAdmins グループのメンバー)
- ソース AWS アカウントの管理者認証情報

制限

- AWS Organizations のソース管理アカウントは、AWS Control Tower のターゲット管理アカウントとは異なる必要があります。

製品バージョン

- AWS Control Tower バージョン 2.3 (2020 年 2 月) 以降 (リリースノートを参照)

アーキテクチャ

次の図は移行プロセスとリファレンスアーキテクチャを示しています。このパターンは、AWS アカウントをソース組織から AWS Control Tower が管理するターゲット組織に移行します。

登録プロセスは、3 つの手順があります。

1. アカウントは AWS Organizations のソース組織を離れます。
2. アカウントはスタンドアロンアカウントになります。つまり、アカウントはどの組織にも属さないため、ガバナンスと請求はアカウント管理者によって独立して管理されます。
3. ターゲット組織は組織に参加するようアカウントの招待を送信します。
4. スタンドアロンアカウントは招待を受け入れ、ターゲット組織のメンバーになります。
5. アカウントは AWS Control Tower に登録され、登録済み組織単位 (OU) に移動されます。(AWS Control Tower ダッシュボードをチェックして登録を確認することをお勧めします。) この時点で、登録済み OU で有効化されているすべてのガードレールが有効になります。

ツール

AWS サービス

- AWS Organizations は、作成して一元管理している複数の AWS アカウントを単一エンティティ（組織）へ統合できるようにするアカウント管理サービスです。
- AWS Control Tower は、AWS Organizations、AWS IAM アイデンティティセンター (AWS Single Sign-On の後継)、AWS Service Catalog など、他のサービスの機能を統合することで、AWS クラウドのすべての組織とアカウント全体で大規模にセキュリティ、オペレーション、およびコンプライアンスのガバナンスルールを強制および管理できます。

エピック

ソース組織からメンバーアカウントを削除する

タスク	説明	必要なスキル
メンバーアカウントがスタンドアロンアカウントとして実行できることを確認します。	ソース組織から離れるメンバーアカウントに、スタンドアロンアカウントとしての運用に必要な情報があることを確認します。たとえば、メンバーアカウントに請求情報がない場合、スタンドアロンアカウントとして運用することはできません。これは、AWS は支払情報を使用して、アカウントが組織に関連付けられていない間に発生する請求対象の AWS アクティビティに対して請求するためです。 通常、AWS Organizations コンソール、API、または AWS コマンドラインインターフェイス (CLI) コマンドを使用してメンバーアカウントを作成した場合、スタンドアロンアカウントに必要な情報は自動的に収集されません。この情報を追加するには、アカウントにサインインし、サポートプラン、連絡先情報、支払方法を指定します。 組織からアカウントを削除する前に知っておくべきことの詳細については、AWS Organizations ドキュメント	アカウント管理者

タスク	説明	必要なスキル
	の組織からアカウントを削除する前にを参照してください。	

タスク	説明	必要なスキル
メンバーアカウントをソース組織から削除します。	AWS Organizations ドキュメントの指示に従い、組織からメンバーアカウントを削除します。組織の管理アカウントにサインインして[メンバーアカウントを削除]、またはメンバーアカウントにサインインして[組織を離れる]こともできます。 アカウントを削除または離れる管理者レベルの認証情報がない場合は、組織の管理者に支援を求めてください。 メンバーアカウントにサポートプラン、連絡先情報、または支払情報がない場合は、その情報の提供と確認を求められます。 組織を離れると、AWS Organizations コンソールの [Getting Started (開始)] ページにリダイレクトされ、そこで、他の組織に参加するアカウントの招待を表示できます。 **Important** この時点で、アカウントはスタンドアロンアカウントです。AWS 無料利用枠の対象とならないワークロードを	管理アカウント管理者またはアカウント管理者

タスク	説明	必要なスキル
	実行中の場合は、アカウントに提供した支払情報と請求情報に基づき請求されます。	
メンバーアカウントがソース組織の一部ではなくなったことを確認します。	AWS Organizations コンソールに、[Leave organization (組織を離れる)] ボタンは表示されなくなっているはずです。代わりに、他の組織からの保留中の招待 (ある場合) が表示されるはずです。	アカウント管理者
離れた組織からアカウントへのアクセスを付与する IAM ロールを削除します。	ソース組織からアカウントを削除しても、AWS Organizations または管理者によって作成された AWS ID およびアクセス管理 (IAM) ロールは自動的に削除されません。ソース組織の管理アカウントからアクセスを終了するには、IAM ロールを手動で削除する必要があります。詳細については、IAM ドキュメントのロールまたはインスタンスプロファイルを削除するを参照してください。 メンバーアカウントが組織を離れると、アカウントにアタッチされていたタグはすべて削除されます。スタンドアロンアカウントはタグをサポートしていません。	アカウント管理者

アカウントを AWS Control Tower の組織に参加するように招待します。

タスク	説明	必要なスキル
AWS Control Tower にサインインします。	管理者として AWS Control Tower コンソールにサインインします。 現在、AWS アカウントをソース組織から AWS Control Tower によって管理されている OU 内の組織に直接移動する方法はありません。ただし、AWS Control Tower ガバナンスを AWS Control Tower によって既に管理されている組織単位 (OU) に登録すると、 AWS Control Tower ガバナンスを既存の AWS アカウントに拡張できます。この手順で AWS Control Tower に口グインする必要があるのはそのためです。	AWS Control Tower 管理者
メンバーアカウントを招待します。	1. AWS Organizations コンソールにサインインし、AWS アカウントページに移動します。 2. [Add an AWS account (AWS アカウントを追加)] ページで、[Invite an existing AWS account (既存の AWS アカウントを招待)] を選択します。 3. 12 桁のアカウント番号 (ダッシュなし)、オプションの説明とタグなど、アカ	AWS Control Tower 管理者

タスク	説明	必要なスキル
	ウント情報を入力してから、[Send invitation (招待を送信)] を選択します。 **Important** アカウント移管の影響を受けるアプリケーションやネットワーク接続がないことを確認します。 このアクションで、メンバーアカウントへのリンクを記載した招待メールを送信します。アカウント管理者がリンクに従って招待を受け入れると、アカウントメンバーは [AWS accounts (AWS アカウント)] ページに表示されます。詳細については、AWS アカウントを組織に招待するを参照してください。	

タスク	説明	必要なスキル
アプリケーションと接続性をテストします。	メンバーアカウントが新しい組織に登録されると、ルート内の OU に表示されます。また、AWS Control Tower に登録されている OU にまだ登録されていないため、アカウントに登録されていないというフラグが付けられ、AWS Control Tower コンソールにも表示されます。 以下について確認します。 • AWS Control Tower ダッシュボードをチェックして、ガードレール違反の有無を確認します。 • ネットワーク接続 (VPN または AWS Direct Connect) をチェックして、転送の影響を受けていないことを確認します。 • (アプリケーション所有者) このアカウントに関連付けられているアプリケーションをテストして、アプリケーションが期待どおりに動作し、依存関係がアカウント移管の影響を受けていないことを確認します。	AWS Control Tower 管理者、メンバーアカウント管理者、アプリケーション所有者

登録用アカウントを準備する

タスク	説明	必要なスキル
ガードレールを見直し、違反があれば修正します。	ターゲット OU で定義されているガードレール、特に予防用ガードレールを確認し、違反があれば修正します。 AWS Control Tower ランディングゾーンをセットアップすると、多くの必須の予防ガードレールがデフォルトで有効化されます。これらは無効化できません。アカウントを登録する前に、これらの必須のガードレールを確認し、メンバーアカウントを (手動またはスクリプトを使用して) 修正する必要があります。 **Note** 予防ガードレールは、AWS Control Tower の登録済みアカウントを準拠させ、ポリシー違反を防止します。予防的ガードレールの違反は登録に影響する可能性があります。検出ガードレールの違反は、登録成功後に検出されると、AWS Control Tower ダッシュボードに表示されます。登録プロセス	AWS Control Tower 管理者、メンバーアカウント管理者

タスク	説明	必要なスキル
	には影響しません。詳細については、AWS ドキュメントの AWS Control Tower のガードレールを参照してください。	
ガードレール違反を修正したら、接続の問題を確認します。	場合によっては、ガードレール違反を修正するには、特定のポートを閉じる、またはサービスを無効化する必要があります。アカウントを登録する前に、それらのポートやサービスを使用するアプリケーションが修正されていることを確認します。	アプリ所有者

AWS Control Tower にアカウントを登録する

タスク	説明	必要なスキル
AWS Control Tower コンソールにサインインします。	AWS Control Tower の管理アクセス権限があるサインイン認証情報を使用します。ルートユーザー (管理アカウント) の認証情報を使用して AWS Organizations アカウントを登録しないでください。エラーメッセージが表示されます。	AWS Control Tower 管理者
アカウントを登録します。	1. AWS Control Tower の [Account Factory (アカウントファクトリー)] ページで、[Enroll account (アカ	AWS Control Tower 管理者

タスク	説明	必要なスキル
	ウントの登録)] を選択します。 2. 登録するアカウントに関連付けられたメールアドレス、AWS Control Tower に表示される表示名、IAM Identity Center のメールアドレス、アカウント所有者の姓名、アカウントを登録する OU などの詳細を入力します。IAM ID センターのメールアドレスは、優先ユーザーメールアドレスです。アカウントメールと同じメールアドレスを使用できます。 3. [[Enroll account (アカウントの登録)] を選択します。 詳細については、AWS Control Tower ドキュメントの既存アカウントの登録を参照してください。	

登録後にアカウントを確認する

タスク	説明	必要なスキル
アカウントを検証します。	AWS Control Tower から、[Accounts (アカウント)] を選択します。登録したばかりのアカウントの初期状態は [Enrolling (登録中)] です。	AWS Control Tower 管理者、メンバーアカウント管理者

タスク	説明	必要なスキル
	登録が完了すると、状態は [Enrolled (登録済み)] に変わります。	
ガードレールの違反がないか確認します。	OUで定義されたガードレールは、登録されたメンバーアカウントに自動的に適用されます。AWS Control Tower ダッシュボードで違反がないか監視し、それに応じて修正します。詳細については、AWS ドキュメントの AWS Control Tower のガードレールを参照してください。	AWS Control Tower 管理者、メンバーアカウント管理者

トラブルシューティング

問題	ソリューション
[An unknown error occurred (不明のエラーが発生しました)] というエラーメッセージが表示されます。後で再試行、または AWS サポートに連絡します。	このエラーは、AWS Control Tower のルートユーザー認証情報 (管理アカウント) を使用して新しいアカウントを登録すると発生します。AWS Service Catalog は、アカウントファクトリーポートフォリオまたは製品をルートユーザーにマップできないため、エラーメッセージが表示されます。このエラーを修正するには、ルート以外のフルアクセスユーザー (管理者) 認証情報を使用して新しいアカウントを登録します。管理アクセスを管理ユーザーに割り当てる方法の詳細については、AWS IAM アイデンティティセンター (AWS Single Sign-On の後継) ドキュメントのはじめにを参照してください。

問題	ソリューション
AWS Control Tower の [Activities (アクティビティ)] ページには、Get Catastrophic Drift アクションが表示されます。	このアクションはサービスのドリフトチェックを反映しており、AWS Control Tower のセットアップに問題があることを示すものではありません。アクションは必要ありません。

関連リソース

ドキュメント

- AWS Organizations 用語と概念 (AWS Organizations ドキュメント)
- AWS Control Tower とは (AWS Control Tower ドキュメント)
- 組織からメンバーアカウントを削除する(AWS Organizations ドキュメント)
- AWS Control Tower で管理者アカウントを作成する(AWS Control Tower ドキュメント)

チュートリアルと動画

- AWS Control Tower ワークショップ(セルフペースのワークショップ)
- AWS Control Tower とは (動画)
- AWS Control Tower でユーザーをプロビジョニングする(動画)
- 既存の組織の AWS Control Tower を有効化する (動画)

AWS のサービスを使用して SAP RHEL Pacemaker クラスターをモニタリングする

作成者: Harsh Thoria (AWS)、Randy Germann (AWS)、RAVEENDRA Voore (AWS)

概要

このパターンでは、Amazon CloudWatch と Amazon Simple Notification Service (Amazon SNS) を使用して、SAP アプリケーションおよび SAP HANA データベースサービス用の Red Hat Enterprise Linux (RHEL) ペースメーカークラスターのアラートをモニタリングおよび設定する手順の概要を説明します。

この設定により、CloudWatch ログストリーム、メトリクスフィルター、アラームを使用して、SAP SCS または ASCS、エンキューレプリケーションサーバー (ERS)、SAP HANA クラスターリソースが「停止」状態になったときにモニタリングできます。Amazon SNS は、停止したクラスターのステータスについて、インフラストラクチャまたは SAP Basis チームに E メールを送信します。

このパターンの AWS リソースは、 AWS CloudFormation スクリプトまたは AWS サービスコンソールを使用して作成できます。このパターンは、コンソールを使用していることを前提としています。CloudFormation スクリプトを提供したり、CloudWatch と Amazon SNS のインフラストラクチャデプロイをカバーしたりすることはありません。Pacemaker コマンドは、クラスターアラート設定を設定するために使用されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- Amazon SNS は、E メールまたはモバイル通知を送信するように設定されています。
- SAP ASCS/ERS for ABAP または SCS/ERS for Java、および SAP HANA Database RHEL Pacemaker クラスター。手順については、以下を参照してください。
  - SAP HANA クラスターのセットアップ
  - SAP Netweaver ABAP/Java クラスターのセットアップ

制約事項

- このソリューションは、現在 RHEL バージョン 7.3 以降の Pacemaker ベースのクラスターで動作します。SUSE オペレーティングシステムではテストされていません。

製品バージョン

- RHEL 7.3 以降

アーキテクチャ

ターゲットテクノロジースタック

- RHEL Pacemaker アラートイベント駆動型エージェント
- Amazon Elastic Compute Cloud (Amazon EC2)
- CloudWatch アラーム
- CloudWatch ロググループとメトリクスフィルター
- Amazon SNS

ターゲット アーキテクチャ

次の図は、このソリューションのコンポーネントとワークフローを示しています。

自動化とスケール

- CloudFormation スクリプトを使用して、 AWS リソースの作成を自動化できます。追加のメトリクスフィルターを使用して、複数のクラスターをスケールおよびカバーすることもできます。

ツール

AWS サービス

- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

ツール

- CloudWatch エージェント (統合) は、EC2 インスタンスからシステムレベルのメトリクス、ログ、トレースを収集し、アプリケーションからカスタムメトリクスを取得するツールです。

- Pacemaker アラートエージェント (RHEL 7.3 以降用) は、Pacemaker クラスターでリソースが停止または再起動したときなど、変更があったときにアクションを開始するツールです。

ベストプラクティス

- で SAP ワークロードを使用するためのベストプラクティスについては AWS、「 AWS Well-Architected フレームワークの SAP レンズ」を参照してください。
- SAP HANA クラスターの CloudWatch モニタリングの設定に伴うコストを考慮してください。詳細については、CloudWatch ドキュメントを参照してください。
- Amazon SNS アラートにページャーまたはチケット発行メカニズムを使用することを検討してください。
- pcs、Pacemaker、フェンシングエージェントの RPM パッケージの RHEL 高可用性 (HA) AWS バージョンを必ず確認してください。

エピック

Amazon SNS をセットアップする

タスク	説明	必要なスキル
SNS トピックを作成します。	1. にサインイン AWS Management Console し、Amazon SNS コンソールを https://console.aws.amazon.com/sns/v3/home://www..com で開きます。 2. Amazon SNS ダッシュボードの [Common actions] の下で、[Create topic] を選択します。 3. 「新しいトピックの作成」ダイアログボックスの「タイプ」で、「標準」を選択します。	AWS 管理者

タスク	説明	必要なスキル
	4. トピック名には、トピックの名前を入力します (例: `my-topic`)。 5. [トピックの作成] を選択してください。 これにより、通知を発行できるリソースポリシーを持つ SNS トピックが作成されます。 6. トピック ARN をコピーします (例: `arn:aws:sns:us-east-1:111122223333:my-topic`)。この ARN は後のステップで使用します。	

タスク	説明	必要なスキル
SNS トピックのアクセスポリシーを変更します。	1. Amazon SNS コンソールのナビゲーションペインで、トピックを選択し、作成したトピックを選択します。 2. 編集 を選択し、アクセスポリシーセクションに移動します。 3. アクセスポリシーに、このトピックへの発行が許可されているサービスプリンシパルの 1 つとして CloudWatch が含まれていることを確認します。以下に例を示します。 `{ "Sid": "Allow AWS CloudWatch to Publish to this SNS topic", "Effect": "Allow", "Principal": { "Service": [ "cloudwatch.amazonaws.com" ] }, "Action": "SNS:Publish", "Resource": "arn:aws:sns:us-east-1:111122223333:my-topic" }`	AWS システム管理者

タスク	説明	必要なスキル
	4. [Save changes] (変更の保存) をクリックします。	

タスク	説明	必要なスキル
SNS トピックにサブスクライブします。	1. Amazon SNS コンソールのナビゲーションペインで、サブスクリプション、サブスクリプションの作成を選択します。 2. トピック ARN には、最初のタスクで作成した ARN を貼り付けます。 3. [Protocol (プロトコル)] として [Email (E メール)] を選択してください。 4. Endpoint には、SAP Pacemaker クラスターを担当し、通知を受け取る担当者またはチームの E メールアドレスを入力します。たとえば、SAP Basis またはインフラストラクチャチームのディストリビューションリストの E メールアドレスを指定できます。 5. [Create subscription] を選択します。 6. E メールアプリケーションから、 AWS 通知からのメッセージを開き、サブスクリプションを確認します。 ウェブブラウザに Amazon SNS の確認画面が表示されます。	AWS システム管理者

クラスターの設定を確認する

タスク	説明	必要なスキル
クラスターのステータスを確認します。	pcs status コマンドを使用して、リソースがオンラインであることを確認します。	SAP ベーシス管理者

Pacemaker アラートを設定する

タスク	説明	必要なスキル

プライマリクラスターインスタンスで Pacemaker アラートエージェントを設定します。	ピマリークラスターの EC2 インスタンスにログインし、次のコマンドを実行します。
	`install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample`
	`touch /var/lib/pacemaker/alert_file.sh`
	`touch /var/log/pcmk_alert_file.log`
	`chown hacluster:haclient /var/log/pcmk_alert_file.log`
	`chmod 600 /var/log/pcmk_alert_file.log`
	`pcs alert create id=alert_file description="Log events to a file." path=/var/lib/pacemaker/alert_file.sh`
	`pcs alert recipient add alert_file id=my-alert_logfile value=/va`
		SAP ベーシス管理者

タスク	説明	必要なスキル
	r/log/pcmk_alert_f ile.log	
セカンダリクラスターインスタンスで Pacemaker アラートエージェントを設定します。	セカンダリクラスターのセカンダリクラスター EC2 インスタンスにログインし、次のコマンドを実行します。 install --mode=0755 / usr/share/pacemak er/alerts/alert_fi le.sh.sample touch /var/lib/ pacemaker/alert_fi le.sh touch /var/log/ pcmk_alert_file.log chown hacluster :haclient /var/log/ pcmk_alert_file.log chmod 600 /var/log/ pcmk_alert_file.log	SAP ベーシス管理者

タスク	説明	必要なスキル

RHEL アラートリソースが作成されていることを確認します。	次のコマンドを使用して、アラートリソースが作成されたことを確認します。
	`pcs alert`
	コマンドの出力は次のようになります。
	`[root@xxxxxxx ~]# pcs alert`
	`Alerts:`
	`Alert: alert_file (path=/var/lib/pacemaker/alert_file.sh)`
	`Description: Log events to a file.`
	`Recipients:`
	`Recipient: my-alert_logfile (value=/var/log/pcmk_alert_file.log)`
		SAP ベーシス管理者

CloudWatch エージェントを設定する

タスク	説明	必要なスキル
CloudWatch エージェントをインストールします。	EC2 インスタンスに CloudWatch エージェントをインストールするには、いくつかの方法があります。コマンドラインを使用するには： 1. CloudWatch エージェントパッケージをダウンロードします。	AWS システム管理者

タスク	説明	必要なスキル
	``` wget https://s3.<region>.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm ``` ここで、 <region> は EC2 インスタンス AWS リージョン が配置されている です (例: us-west-2)。 2. オプション) パッケージの署名を確認します。手順については、CloudWatch ドキュメントの「CloudWatch エージェントパッケージの署名の検証」を参照してください。 CloudWatch 3. 最初のインスタンスに パッケージをインストールします。 ``` sudo rpm -U ./amazon-cloudwatch-agent.rpm ``` 4. セカンダリインスタンスに対して繰り返します。 詳細については、CloudWatch ドキュメントを参照してください。	

タスク	説明	必要なスキル
EC2 インスタンスに IAM ロールをアタッチします。	CloudWatch エージェントがインスタンスからデータを送信できるようにするには、IAM CloudWatchAgentServerRole ロールを各インスタンスにアタッチする必要があります。または、CloudWatch エージェントのポリシーを既存の IAM ロールに追加することもできます。詳細については、CloudWatch ドキュメントを参照してください。	AWS 管理者

タスク	説明	必要なスキル
プライマリクラスターインスタンスの Pacemaker アラートエージェントのログファイルをモニタリングするように CloudWatch エージェントを設定します。	1. コマンドを実行して、プライマリクラスターインスタンスを設定します。 `sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard` 2. Linux の場合は 1 を選択し、モニタリング戦略のオプションを選択します。 3. 質問「ログファイルをモニタリングしますか」で、「はい」を選択し、pcs アラートコマンドから Pacemaker ログファイルのパスを指定します。この場合、これは ですvar/log/pcmk_alert_file.log 。 4. ロググループとログストリームの名前を指定します。ログストリームを指定しない場合、 AWS インスタンス ID がデフォルトとして使用されます。 5. セカンダリクラスターインスタンスに対してステップ 1～4 を繰り返します。	AWS 管理者

タスク	説明	必要なスキル
プライマリクラスターインスタンスとセカンダリクラスターインスタンスで CloudWatch エージェントを起動します。	エージェントを起動するには、プライマリクラスターとセカンダリクラスターの EC2 インスタンスで次のコマンドを実行します。 `sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json`	AWS 管理者

CloudWatch リソースの設定

タスク	説明	必要なスキル
CloudWatch ロググループを設定します。	1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。 2. ナビゲーションペインで、ロググループ、ロググループの作成を選択します。 3. ロググループの名前を入力し、ロググループの作成を選択します。 CloudWatch エージェントは、Pacemaker アラートファイルをログストリームとして	AWS 管理者

タスク	説明	必要なスキル
	CloudWatch ロググループに転送します。	

タスク	説明	必要なスキル
CloudWatch メトリクスフィルターを設定します。	メトリクスフィルターは、CloudWatch ログストリーム`stop <cluster-resource-name>` などのパターンを検索するのに役立ちます。このパターンが特定されると、メトリクスフィルターはカスタムメトリクスを更新します。 1. CloudWatch コンソールのナビゲーションペインで、ロググループを選択します。 2. 前のタスクで作成したロググループの名前を選択します。 3. [アクション]、[メトリクスフィルターの作成] の順に選択します。 4. フィルターパターンには、などの使用するフィルターパターンを入力して`stop ABC_scs`、 という名前の SAP SCS クラスターリソースの停止イベントに一致させます`ABC_scs`。 詳細については、CloudWatch ドキュメントの「フィルターパターン構文」を参照してください。 5. (オプション) フィルターパターンをテストするに	AWS 管理者、SAP ベーシス管理者

タスク	説明	必要なスキル
	は、[テストパターン] に、パターンのテストに使用する 1 つまたは複数の口グイベントを入力します。口グイベントメッセージボックスで口グイベントを区切るには改行が使用されるため、各口グイベントは別々の行で指定する必要があります。 6. [次へ] を選択し、フィルターの名前を入力します。 7. メトリクスの詳細のメトリクス名前空間に、メトリクスが公開される CloudWatch 名前空間の名前を入力します (例: `sapcluster_monitoring`)。この名前空間がまだ存在しない場合は、新規作成を選択します。 8. メトリクス名には、新しいメトリクスの名前を入力します (例: 。ここで`sapcluster_<sid>` 、<sid>は SAP システム識別名です)。 9. メトリクス値には、1 と入力します。 または、 などのトークンを入力することもできます`$size`。これにより、size フィールドを含	

タスク	説明	必要なスキル
	むすべての口グイベントについて、size フィールド内の数値だけメトリクスが増加します。 10.デフォルト値には、0 と入力します。 11.[Create metric filter] (メトリクスフィルターの作成) を選択します。 メトリクスフィルターは、ステップ 4 でパターンを識別すると、CloudWatch カスタムメトリクスの値を 1 sapcluster_abc に更新します。 CloudWatch アラームはメトリクスをSAP-Cluster-QA1-ABC モニタリングsapcluster_abc し、メトリクスの値が 1 に変わったときに SNS 通知を送信します。これは、クラスターリソースが停止し、アクションを実行する必要があることを示します。	

タスク	説明	必要なスキル
SAP ASCS/SCS および ERS メトリクスの CloudWatch メトリクスアラームを設定します。	単一のメトリクスに基づいてアラームを作成するには： 1. CloudWatch コンソールのナビゲーションペインで、アラーム、すべてのアラームを選択します。 2. [アラームの作成] を選択します。 3. [メトリクスの選択] を選択します。 4. 前のタスクで`sapcluster_monitoring` 作成されたカスタムメトリクスを検索します。 5. 前のタスクでも作成された SAP SCS のメトリクス名 (など`sapcluster_<abc>`) を選択します。 6. グラフ化されたメトリクスタブで、以下を設定します。 • [統計] で、[Maximum] を選択します。 • [期間] は[1 minute] を選択してください。 • しきい値タイプで、静的を選択し、 のしきい値 `sapcluster_<sid>` を 1 以上の値に設定します。 7. [Next (次へ)] を選択します。	AWS 管理者

タスク	説明	必要なスキル
	8. 通知 で、最初の工ピックで作成した SNS トピックを選択します。 9. 名前と説明 にアラーム名と簡単な説明を入力し、次へ を選択します。 10.アラームの作成(アラームの作成) を選択します。	
SAP HANA メトリクスの CloudWatch メトリクスアラームを設定します。	前のタスクから CloudWatch メトリクスアラームを設定する手順を繰り返し、これらの変更を加えます。 • ステップ 5 では、SAP HANA のメトリクス名 (例:) を選択しますsapcluster_db_<abc> 。 • ステップ 6 では、 のしきい値sapcluster_<sid> を 0 より大きい値に設定します。	AWS 管理者

関連リソース

- クラスターイベントのスクリプトのトリガー (RHEL ドキュメント)
- ウィザードを使用して CloudWatch エージェント設定ファイルを作成する (CloudWatch ドキュメント)
- サーバーへの CloudWatch エージェントのインストールと実行 (CloudWatch ドキュメント)
- 静的しきい値に基づいて CloudWatch アラームを作成する (CloudWatch ドキュメント)CloudWatch
- 高可用性クラスターを使用した AWS への SAP HANA の手動デプロイ (AWS ウェブサイトの SAP ドキュメント)

- SAP NetWeaver ガイド（ AWS ウェブサイトの SAP ドキュメント）

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

複数の で共有 Amazon マシンイメージの使用をモニタリングする AWS アカウント

作成者：Naveen Suthar (AWS) と Sandeep Gawande (AWS)

概要

Amazon マシンイメージ (AMIs) は、Amazon Web Services () 環境に Amazon Elastic Compute Cloud (Amazon EC2 AWS) インスタンスを作成するために使用されます。AMI は、一つの集中管理された AWS アカウントで作成できます。さらにこのパターンでは、作成者アカウントと呼ばれます。その後、同じ にある複数の 間で AMI を共有できます AWS アカウント 。これは AWS リージョン、このパターンではコンシューマーアカウントと呼ばれます。1 つのアカウントから AMI を管理することでスケーラビリティが向上し、ガバナンスを簡単に行うことができます。コンシューマーアカウントでは、Amazon EC2 Auto Scaling「起動テンプレート」と Amazon Elastic Kubernetes Service (Amazon EKS)「ノードグループ」の共有 AMI を参照できます。

コンシューマーアカウントの AMI を参照 AWS のサービス する共有 AMI が廃止、登録解除、または共有解除された場合、この AMI を使用して新しいインスタンスを起動することはできません。自動スケーリングイベントまたは同じインスタンスの再起動は失敗します。これにより、アプリケーションのダウンタイムやパフォーマンスの低下など、本番環境で問題が生じる可能性があります。AMI 共有イベントと使用状況イベントが複数の で発生する場合 AWS アカウント、このアクティビティをモニタリングするのは難しい場合があります。

このパターンは、同じリージョンのアカウント間での共有 AMI の使用状況とステータスをモニタリングするのに役立ちます。Amazon EventBridge AWS のサービス、Amazon DynamoDB、Amazon Simple Email Service (AWS Lambda Amazon SES) などのサーバーレスを使用します。HashiCorp Terraform を使用して、Infrastructure as Code (IaC) としてプロビジョニングします。このソリューションは、コンシューマーアカウントのサービスが登録解除または共有されていない AMI を参照したときにアラートを発行します。

前提条件と制限

前提条件

- 2 つ以上のアクティブ AWS アカウント: 1 つの作成者アカウントと 1 つ以上のコンシューマーアカウント
- クリエーターアカウントからコンシューマーアカウントに共有される 1 つ以上の AMI
- 「インストール済み」Terraform CLI(Terraform のドキュメント)

- Terraform AWS プロバイダー、設定済み (Terraform ドキュメント)
- (オプションであるが、推奨)「設定済み」Terraform バックエンド(Terraform のドキュメント)
- 「インストール済み」Git

制約事項

- このパターンは、アカウント ID を使用して特定のアカウントと共有されている AMI をモニタリングします。このパターンでは、組織 ID を使用して組織と共有されている AMI はモニタリングされません。
- AMIs は、同じ 内のアカウントにのみ共有できます AWS リージョン。このパターンは、単一のターゲットリージョン内の AMI をモニタリングします。複数のリージョンでの AMI の使用状況をモニタリングするには、このソリューションを各リージョンにデプロイします。
- このパターンでは、このソリューションがデプロイされる前に共有されていた AMI はモニタリングされません。以前に共有していた AMI をモニタリングしたい場合は、AMI 共有を解除してからコンシューマーアカウントと再共有できます。

製品バージョン

- Terraform バージョン 1.2.0 以降
- Terraform AWS プロバイダーバージョン 4.20 以降

アーキテクチャ

ターゲットテクノロジースタック

以下のリソースは Terraform を通じて IaC としてプロビジョニングされます。

- Amazon DynamoDB テーブル
- Amazon EventBridge ルール
- AWS Identity and Access Management (IAM) ロール
- AWS Lambda 関数
- Amazon SES

ターゲットアーキテクチャ

この図表は、次のワークフローを示しています：

1. 作成者アカウントの AMI は、同じ のコンシューマーアカウントと共有されます AWS リージョン。
2. AMI を共有すると、作成者アカウントの EventBridge ルールが`ModifyImageAttribute`イベントをキャプチャし、作成者アカウントで Lambda 関数を開始します。
3. Lambda 関数は、作成者アカウントの DynamoDB テーブルに AMI に関連するデータを格納します。
4. コンシューマーアカウントの AWS のサービス が共有 AMI を使用して Amazon EC2 インスタンスを起動する場合、または共有 AMI が起動テンプレートに関連付けられている場合、コンシューマーアカウントの EventBridge ルールは共有 AMI の使用をキャプチャします。
5. EventBridge ルールは、コンシューマーアカウントで Lambda 関数を起動します。Lambda 関数は以下を実行します。
   a. Lambda 関数は、作成者アカウントの DynamoDB テーブルに AMI 関連データを格納します。
   b. Lambda 関数は、作成者アカウントの IAM ロールを引き受け、作成者アカウントの Lambda テーブルを更新します。`Mapping` テーブルでは、インスタンス ID または起動テンプレート ID をそれぞれの AMI ID にマッピングする項目を作成します。
6. 作成者アカウントで一元管理されている AMI は非推奨、登録解除または共有解除されました。
7. 作成者アカウントのEventBridge ルールは、`remove` アクションを含む `ModifyImageAttribute` または `DeregisterImage` イベントをキャプチャし、Lambda関数を起動します。
8. Lambda 関数は DynamoDB テーブルをチェックして、AMI がいずれかのコンシューマーアカウントで使用されているかを判断します。AMI に関連付けられているインスタンス ID または起動テンプレート ID が `Mapping` テーブルにない場合は、プロセスは完了です。
9. インスタンス ID または起動テンプレート ID が `Mapping` テーブル内の AMI に関連付けられている場合、Lambda 関数は Amazon SES を使用して、設定したサブスクライバーにメール通知を送信します。

ツール

AWS のサービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを提供します。

- 「Amazon EventBridge」 は、アプリケーションをさまざまなソースのデータに接続するために支援するサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Identity and Access Management (IAM) は、AWS リソースの使用を認証および認可されたユーザーを制御することで、AWS リソースへのアクセスを安全に管理するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Email Service (Amazon SES)」はユーザー自身のメールアドレスとドメインを使用してメールを送受信する上で役立ちます。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub 内の「cross-account-ami-monitoring-terraform-samples」リポジトリで利用できます。

ベストプラクティス

- AWS Lambda 関数を使用する際のベストプラクティスに従ってください。
- 「AMI 構築のベストプラクティス」に従ってください。
- IAM ロールを作成するときは、最小特権の原則に従い、タスクの実行に必要最小限の権限を付与します。詳細については、IAM ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。
- AWS Lambda 関数のモニタリングとアラートを設定します。詳細については、「Lambda 関数をモニタリングおよびトラブルシューティングする」を参照してください。

エピック

Terraform 設定ファイルをカスタマイズします。

タスク	説明	必要なスキル
AWS CLI 名前付きプロファイルを作成します。	作成者アカウントと各コンシューマーアカウントで、AWS Command Line Interface (AWS CLI) という名前のプロファイルを作成します。手順については、 AWS 「 入門リソースセンター」の「 のセットアップ AWS CLI」を参照してください。	DevOps エンジニア
リポジトリをクローン作成します。	次のコマンドを入力します。これにより、SSH を使用して GitHub から「 cross-account-ami-monitoring-terraform-samples 」リポジトリのクローンが作成されます。 `git clone git@github.com:aws-samples/cross-account-ami-monitoring-terraform-samples.git`	DevOps エンジニア
プロバイダーの.tf ファイルを更新します。	1. 次のコマンドを入力して、複製されたリポジトリ内の `terraform` フォルダーに移動します。 `cd cross-account-ami-monitoring/terraform`	DevOps エンジニア

タスク	説明	必要なスキル
	2. `provider.tf` ファイルを開きます。 3. 次のように、作成者アカウントとコンシューマーアカウントの Terraform AWS プロバイダー設定を更新します。 • `alias` には、プロバイダー設定の名前を入力します。 • に`region`、このソリューションをデプロイ AWS リージョン するターゲットを入力します。 • には`profile`、アカウントにアクセスするための AWS CLI 名前付きプロファイルを入力します。 4. 複数のコンシューマーアカウントを設定する場合は、追加のコンシューマーアカウントごとにプロファイルを作成します。 5. `provider.tf` ファイルを保存して閉じます。 プロバイダーの構成について、詳細は、Terraform ドキュメントの「複数のプロバイダー設定」を参照してください。	

タスク	説明	必要なスキル
テラフォーム.tfvars ファイルを更新します。	1. `terraform.tfvars` ファイルを開きます。 2. `account_email_mapping` パラメーターで、作成者アカウントとコンシューマーアカウントのアラートを次のように設定します。 • `account` には、アカウント ID を入力します。 • `email` のために、アラートを送信するメールアドレスを入力します。各アカウントにつき 1 つのメールアドレスしか入力できません。 3. 複数のコンシューマーアカウントを設定する場合は、コンシューマーアカウントごとにアカウントとメールアドレスを追加して入力します。 4. `terraform.tfvars` ファイルを保存して閉じます。	DevOps エンジニア

タスク	説明	必要なスキル
メイン.tf ファイルを更新します。	この手順は、このソリューションを複数のコンシューマーアカウントにデプロイする場合にのみ実行します。このソリューションを 1 つのコンシューマーアカウントにのみデプロイする場合、このファイルを変更する必要はありません。 1. `main.tf` ファイルを開きます。 2. コンシューマーアカウントを追加するたびに、テンプレート内の `consumer_account_A` モジュールに基づいた新しいモジュールを作成します。各コンシューマーアカウントには、`provider` のために、この値は `provider.tf` ファイルに入力したエイリアスと一致させる必要があります。 3. `main.tf` ファイルを保存して閉じます。	DevOps エンジニア

Terraform を使用してソリューションをデプロイ

タスク	説明	必要なスキル
ソリューションをデプロイします。	Terraform CLI で、次のコマンドを入力して、作成者アカウントとコンシューマーアカウ	DevOps エンジニア

タスク	説明	必要なスキル

	ントに AWS リソースをデプロイします。
	
	1. 以下のコマンドを入力して、Terraform を初期化します。
	
	`terraform init`
	
	2. 以下のコマンドを入力して、Terraform の設定を検証します。
	
	`terraform validate`
	
	3. 以下のコマンドを入力して、Terraform 実行プランを作成します。
	
	`terraform plan`
	
	4. Terraform プランの設定変更を確認し、変更を実装することを確認します。
	5. リソースをデプロイするには、次のコマンドを入力します。
	
	`terraform apply`	

タスク	説明	必要なスキル
メールアドレス ID を検証します。	Terraform プランをデプロイすると、Terraform は Amazon SES のコンシューマーアカウントごとにメールアドレス ID を作成しました。該当メールアドレスに通知を送信する前に、メールアドレスを検証する必要があります。手順については、「Amazon SES ドキュメント」の「メールアドレスアイデンティの検証」を参照してください。	AWS 全般

リソースデプロイを検証

タスク	説明	必要なスキル
クリエイターアカウントによるデプロイを検証します。	1. 作成者アカウントにサインインします。 2. ナビゲーションバーで、がターゲットを表示していることを確認します AWS リージョン。別のリージョンの場合は、現在表示されているリージョン名を選択して、ターゲットリージョンを選択します。 3. DynamoDB コンソールを開きます。 4. ナビゲーションペインで、[Tables (テーブル)] を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	5. テーブルのリストで、目的の AmiShare テーブルが存在することを検証します。 6. Lambdaのコンソールを開きます。 7. ナビゲーションペインで、[関数] を選択します。 8. 関数リストで、その ami-share 関数が存在することを検証します。 9. IAM コンソールを開きます。 10.ナビゲーションペインで [ロール] を選択します。 11.ロールのリストで、目的の external-ddb-role ロールが存在することを検証します。 12.[EventBridge コンソール] を開きます。 13.ナビゲーションペインで [ルール] を選択します。 14.ルールのリストで、目的の modify_image_attribute_event ルールが存在することを検証します。 15.Amazon SES コンソールを開きます。 16.ナビゲーションペインで、[検証済み ID] を選択します。	

タスク	説明	必要なスキル
	17ID のリストで、コンシューマーアカウントごとにメールアドレス ID が登録し、検証されていることを確認します。	

タスク	説明	必要なスキル
コンシューマアカウントによるデプロイを検証します。	1. コンシューマーアカウントにサインインします。 2. ナビゲーションバーで、がターゲットを表示していることを確認します AWS リージョン。別のリージョンの場合は、現在表示されているリージョン名を選択して、ターゲットリージョンを選択します。 3. DynamoDB コンソールを開きます。 4. ナビゲーションペインで、[Tables (テーブル)] を選択します。 5. テーブルのリストで、目的の `Mapping` テーブルが存在することを検証します。 6. Lambdaのコンソールを開きます。 7. ナビゲーションペインで、[関数] を選択します。 8. 関数リストで、 `ami-usage-function` と `ami-deregister-function` 関数が存在することを検証します。 9. [EventBridge コンソール] を開きます。 10. ナビゲーションペインで [ルール] を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
	11ルールのリストに、`ami_usage_events` ルールと `ami_deregister_events` ルールが存在することを検証します。	

モニタリング検証

タスク	説明	必要なスキル
クリエイターアカウントで AMI を作成します。	1. クリエイターアカウントで非公開 AMI を作成します。手順については、Amazon EC2 インスタンスから AMI を作成する」を参照してください。 2. 新しい AMI をいずれかのコンシューマーアカウントと共有します。手順については、「特定の と AMI を共有する AWS アカウント」を参照してください。	DevOps エンジニア
コンシューマーアカウントで AMI を使用します。	コンシューマーアカウントで、共有 AMI を使用して Amazon EC2 インスタンスまたは起動テンプレートを作成します。手順については、「How do I launch an Amazon EC2 instance from a custom AMI (AWS re:Post Knowledge Center)」または「Create a launch template for an Auto	DevOps エンジニア

タスク	説明	必要なスキル
	Scaling group」(Amazon EC2 Auto Scaling ドキュメント) を参照してください。	
モニタリングとアラートを検証します。	1. 作成者アカウントにサインインします。 2. Amazon EC2 コンソールを開きます。 3. ナビゲーションペインで [AMI] を選択してください。 4. リストで AMI を選択し、[アクション]、[AMI 権限の編集] の順に選択します。 5. [共有アカウント] セクションで、コンシューマーアカウントを選択し、[選択したものを削除] を選択します。 6. [Save changes] (変更の保存) をクリックします。 7. コンシューマーアカウント用に定義したターゲットのメールアドレスが、AMI の共有がキャンセルされたという通知を受信することを確認します。	DevOps エンジニア

(オプション) 共有 AMI のモニタリングを停止

タスク	説明	必要なスキル
リソースを削除します。	1. 以下のコマンドを入力して、このパターンでデプロイされたリソースを削除し、共有 AMI のモニタリングを停止します。 `terraform destroy` 2. `yes` を入力して `destroy` コマンドを確認します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
メールアラートを受信していません。	Amazon SES の電子メールが送信されなかった理由は複数考えられます。以下をチェックしてください: 1. エピックセクションで、リソースデプロイの検証エピックを使用して、インフラストラクチャがすべてで適切にプロビジョニングされたことを確認します AWS アカウント。 2. Amazon CloudWatch Logs の Lambda 関数イベントを検証します。手順については、Lambda ドキュメントの「CloudWatch コンソールの使用」を参照してください。アクセス権限に関する問題 (アイデンティティベースのポリシーまたはリソースベースのポリシーでの明示的な拒否など) がないことを確認します。詳細については、IAM ドキュメントの「ポリシーの評価論理」を参照してください。

問題	ソリューション
	3. Amazon SES で、メールアドレス ID のステータスが確認済みになっていることを確認します。詳細については、「メールアドレス ID の検証」を参照してください。

関連リソース

AWS ドキュメント

- 「Python による Lambda 関数の構築」(Lambda のドキュメント)
- 「AMI を作成する」(Amazon EC2 のドキュメント)
- AMI を特定の と共有する AWS アカウント (Amazon EC2 ドキュメント)
- 「AMI の登録の解除」(Amazon EC2 のドキュメント)

Terraformのドキュメント

- 「Terraform のインストール」
- 「Terraform バックエンド設定」
- 「Terraform AWS Provider」
- 「Terraform バイナリのダウンロード」

AWS Organizations のプログラムによるアカウント閉鎖のアラートを設定する

作成者: Richard Milner-Watts (AWS)、Debojit Bhadra (AWS)、Manav Yadav (AWS)

概要

CloseAccount API for AWS Organizations を使用すると、ルート認証情報を使用してアカウントにログインしなくても、組織内のメンバーアカウントをプログラムで閉鎖できます。RemoveAccountFromOrganization API は AWS Organizations 内の組織からアカウントをプルするので、スタンドアロンアカウントになります。

これらの API により、AWS アカウントを閉鎖または削除できるオペレーターの数が増加する可能性があります。AWS Organizations 管理アカウントの AWS Identity and Access Management (IAM) を通じて組織にアクセスできるすべてのユーザーは、これらの API を呼び出すことができるため、関連する多要素認証 (MFA) デバイスを持つアカウントのルート メールの所有者にアクセスが限定されません。

このパターンでは、`CloseAccount` および `RemoveAccountFromOrganization` API が呼び出されるとアラートが実装されるため、これらのアクティビティを監視できます。アラートは、「Amazon Simple Notification Service (Amazon SNS)」のトピックを使用します。ウェブフック経由で Slack 通知を設定することもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Organizations 内の組織
- 組織のルート下にある組織管理アカウントにアクセスして、必要なリソースを作成します。

機能制限

- 「AWS Organizations API リファレンス」で説明されているように、`CloseAccount` API では 30 日以内に閉鎖できるのは、アクティブなメンバーアカウントの 10% のみです。
- AWS アカウントが閉鎖されると、ステータスは SUSPENDED に変わります。このステータスに移行した後 90 日間、AWS サポートはアカウントを再開できます。保留になったアカウントは 90 日後に完全に削除されます。

- AWS Organizations 管理アカウントと API にアクセスできるユーザーには、これらのアラートを無効にする権限もあります。誤った削除ではなく悪意のある行為が主な懸念事項である場合は、このパターンで作成されたリソースを「IAM のアクセス許可の境界」で保護することを検討してください。
- API `CloseAccount` および `RemoveAccountFromOrganization` は、米国東部 (バージニア北部) リージョン (`us-east-1`) で呼び出します。したがって、イベントを観察するには、このソリューションを `us-east-1` でデプロイする必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Organizations
- AWS CloudTrail
- Amazon EventBridge
- AWS Lambda
- Amazon SNS

ターゲット アーキテクチャ

このパターンのソリューションアーキテクチャを次の図に示します。

1. AWS Organizations は `CloseAccount` または `RemoveAccountFromOrganization` のリクエストを処理します。
2. Amazon EventBridge は AWS CloudTrail と統合されており、これらのイベントをデフォルトのイベントバスに渡します。
3. カスタム Amazon EventBridge ルールは AWS Organizations のリクエストを照合し、AWS Lambda 関数を呼び出します。
4. Lambda 関数は、ユーザーがメールでアラートを受信したり、さらに処理するためにサブスクライブできる SNS のトピックにメッセージを渡します。
5. Slack 通知が有効になっている場合、Lambda 関数は Slack ウェブフックにメッセージを配信します。

ツール

AWS サービス

- AWS CloudFormation では、インフラストラクチャをコードとして扱うことで、関連する AWS リソースやサードパーティリソースのコレクションをモデル化し、迅速かつ一貫したプロビジョニングを行い、ライフサイクル全体を通じて管理できます。
- Amazon EventBridge は、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。EventBridge は、環境の変化の指標であるイベントを受信し、イベントをターゲットにルーティングするルールを適用します。ルールは、イベントパターンと呼ばれるイベントの構造、またはスケジュールのいずれかに基づいて、イベントをターゲットにマッチングさせます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。支払いは、使用したコンピューティング時間に対する料金のみになります。コードが実行されていないときに料金は発生しません。
- AWS Organizations は、AWS リソースの成長や拡張に伴い、環境の一元管理およびガバナンスを支援します。AWS Organizations を使用すると、プログラムによる AWS アカウントの新規作成、リソースの割り当て、ワークロードを整理するためのアカウントのグループ化、ガバナンスのアカウントまたはアカウントグループへのポリシーの適用、すべてのアカウントに単一の支払い方法を使用した請求の簡素化が可能になります。
- AWS CloudTrail は、AWS インフラストラクチャ全体のアカウントアクティビティをモニタリングして記録し、ストレージ、分析、修復アクションを制御できるようにします。
- Amazon Simple Notification Service (Amazon SNS) は、アプリケーション間 (A2A) およびアプリケーションと人間の間 (A2P) の通信の両方に対応するフルマネージド型のメッセージングサービスです。

その他のツール

- AWS Lambda Powertools for Python ライブラリ は、Lambda 関数のトレース、ロギング、メトリクス、およびイベント処理機能を提供するユーティリティのセットです。

コード

このパターンのコードは、GitHub 内の「AWS アカウント閉鎖通知」リポジトリで利用できます。

このソリューションには、このパターンのアーキテクチャをデプロイする CloudFormation テンプレートが含まれています。AWS Lambda Powertools for Python ライブラリを使用してログ記録とトレースを行います。

エピック

アーキテクチャをデプロイする

タスク	説明	必要なスキル
ソリューションスタック用の CloudFormation テンプレートを起動します。	このパターンの CloudFormation テンプレートは GitHub リポジトリのメインブランチにあります。IAM ロール、EventBridge ルール、Lambda 関数、および SNS トピックをデプロイします。 テンプレートを起動するには: 1. GitHub リポジトリをクローンして、ソリューションコードのコピーを取得します。 2. AWS Organizations 管理アカウントで、AWS マネジメントコンソールを開きます。 3. 米国東部 (バージニア北部) リージョン (`us-east-1`) を選択してから、CloudFormation コンソールを開きます。 4. `account-closure-notifier.yml` テンプレートを使用し、以下の値を指	AWS 管理者

タスク	説明	必要なスキル
	定することで、スタックを作成します。 • スタック名: `aws-account-closure-notifier-stack` • `ResourcePrefix` パラメータ: `aws-account-closure-notifier` • `SlackNotification` パラメータ: Slack 通知が必要な場合は、この設定を `true` に変更します。 • `SlackWebhookEndpoint` パラメーター: Slack 通知が必要な場合は、ウェブフック URL を指定します。 CloudFormation スタックの起動に関する詳細については、AWS ドキュメントを参照してください。	

タスク	説明	必要なスキル
ソリューションが正常に起動したことを検証します。	1. CloudFormation スタックが CREATE_COMPLETE ステータスになるまで待ちます。 2. `us-east-1` で、EventBridge コンソールを開きます。 3. `aws-account-closure-notifier-event-rule` という名前で新しいルールが作成されていることを検証します	AWS 管理者

タスク	説明	必要なスキル
SNS トピックにサブスクライブします。	(オプション) SNS トピックをサブスクライブする場合: 1. us-east-1 で Amazon SNS コンソールを開き、aws-account-closure-notifier-sns-topic という名前のトピックを見つけます。 2. トピック名を選択し、[サブスクリプションの作成] を選択します。 3. プロトコルには、[Eメール] を選択します。 4. エンドポイントに通知の受取先として使用するメールアドレスを入力し、[サブスクリプションの作成] を選択します。 5. メールの受信トレイに AWS Notifications からのメッセージが届いていることを確認してください。メール内のリンクを使用して、サブスクリプションを確認します。 SNS 通知をセットアップする詳しい方法については、「Amazon SNS ドキュメント」を参照してください。	AWS 管理者

ソリューションを検証する

タスク	説明	必要なスキル
デフォルトのイベントバスにテストイベントを送信します。	GitHub リポジトリには、テスト用に EventBridge のデフォルトイベントバスに送信できるサンプルイベントが用意されています。EventBridge ルールは、カスタムイベントソース `account.closure.notifier` を使用するイベントにも反応します。 **Note** CloudTrail イベントソースを使用してこのイベントを送信することはできません。これは、イベントを AWS サービスとして送信できないためです。 テストイベントを送信するには 1. `us-east-1` で、EventBridge コンソールを開きます。 2. ナビゲーションペインの [バス] で、[イベントバス] を選択してから、デフォルトのイベントバスを選択します。	AWS 管理者

タスク	説明	必要なスキル
	3. [イベントの送信] を選択します。 4. [イベントソース] には、`account.closure.notifier` と入力します。 5. [詳細タイプ] には、`AWS API Call via CloudTrail` と入力します。 6. [イベントの詳細] には、GitHub リポジトリから `tests/dummy-event.json` の内容をコピーしてテキストボックスに貼り付けます。 7. [送信] を選択して通知ワークフローを開始します。	
メール通知を受信したことを確認します。	SNS トピックをサブスクライブしているメールボックスに通知が届いていることを確認します。閉鎖されたアカウントと API コールを実行したプリンシパルの詳細が記載されたメールが届きます。	AWS 管理者

タスク	説明	必要なスキル
Slack 通知を受信したことを検証します。	(オプション) CloudFormation テンプレートをデプロイした際に `SlackWebhookEndpoint` パラメータにウェブフック URL を指定した場合は、ウェブフックにマッピングされている Slack チャネルを確認してください。閉鎖されたアカウントと API コールを実行したプリンシパルの詳細が記載されたメッセージが表示されます。	AWS 管理者

関連リソース

- 「CloseAccountアクション」 (AWS Organizations API リファレンス)
- 「RemoveAccountFromOrganization アクション」 (AWS Organizations API リファレンス)
- 「AWS Lambda Powertools for Python」

AWS アカウントまたは組織の EBS スナップショットの詳細を表示

作成者：Arun Chandapillai (AWS) と Parag Nagwekar (AWS)

概要

このパターンでは、Amazon Web Services (AWS) アカウントまたは 組織の組織単位 (OU) にあるすべての Amazon Elastic Block Store (Amazon EBS) スナップショットのオンデマンドレポートを自動的に生成する方法を示します。

Amazon EBS は、Amazon Elastic Compute Cloud (Amazon EC2) のために設計された、使いやすくスケーラブルな高性能ブロックストレージサービスです。EBS ボリュームは、耐久性に優れた永続的なストレージで、EC2 インスタンスに添付することができます。EBS ボリュームをデータのプライマリストレージとして使用し、スナップショットを作成することで EBS ボリュームのポイントインタイムバックアップを作成します。AWS マネジメントコンソールまたは AWS コマンドラインインターフェイス (AWS CLI) を使用して、特定の EBS スナップショットの詳細を表示します。このパターンでは、AWS アカウントまたは OU 内のすべての EBS スナップショットに関する情報をプログラムで取得する方法を提供します。

このパターンで提供されるスクリプトを使用して、各スナップショットに関するアカウント ID、スナップショット ID、ボリューム ID とサイズ、スナップショットの作成日、インスタンス ID、説明などの情報を含むカンマ区切り値 (CSV) ファイルを生成できます。EBS スナップショットにタグが付いている場合、レポートには所有者属性とチーム属性も含まれます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- インストール済み 、及び設定済み」のAWS CLI バージョン2
- 適切な権限 (特定のアカウント、または AWS Organizations からスクリプトを実行する予定の場合、 OU のすべてのアカウントのアクセス許可) を持つ AWS アイデンティティおよびアクセス管理 (IAM) ロール

アーキテクチャ

次の図表には、OU の複数の AWS アカウントに分散している EBS スナップショットのオンデマンドレポートを生成するスクリプトワークフローを示しています。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- Amazon Elastic Block Store (Amazon EBS) は、EC2 インスタンスで使用するためのブロックレベルのストレージボリュームを提供します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Organizationsは、作成して一元管理している複数の AWS アカウントを組織に統合するためのアカウント管理サービスです。

コード

このパターンで使用されているサンプルアプリケーションのコードは、GitHub の aws-ebs-snapshots-awsOrganizations リポジトリにあります。次のセクションの指示に従って、サンプルファイルを使用します。

エピック

スクリプトをダウンロードします。

タスク	説明	必要なスキル
Python スクリプトをダウンロードします。	GitHub リポジトリからスクリプト GetSnapshotDetailsAllAccountsOU.py をダウンロードします。	AWS 全般

AWS アカウントの EBS スナップショットの詳細を取得

タスク	説明	必要なスキル
Python スクリプトを実行する。	コマンドを実行します。	AWS 全般

タスク	説明	必要なスキル

	`python3 getsnapshotinfo.py --file <output-file>.csv --region <region-name>`
	
	ここで、<output-file> は、配置された EBS スナップショットに関する情報が必要な CSV 出力ファイルを指し、<region-name> スナップショットが保存されている AWS リージョンを指します。例：
	
	`python3 getsnapshotinfo.py --file snapshots.csv --region us-east-1`	

組織の EBS スナップショットの詳細を取得

タスク	説明	必要なスキル
Python スクリプトを実行する。	コマンドを実行します。 `python3 getsnapshotinfo.py --file <output-file>.csv --role <IAM-role> --region <region-name>` ここで、<output-file> は EBS スナップショットに関する情報を格納する CSV 出力ファイルを指し、<IAM-role	AWS 全般

タスク	説明	必要なスキル
	> はAWS Organizations にアクセスするための権限を提供するロールで、<region-name> はスナップショットが保存されている AWS リージョンを指します。例： `python3 getsnapshotinfo.py --file snapshots.csv --role <IAM role> --region us-west-2`	

関連リソース

- Amazon EBS ドキュメント
- Amazon EBS アクション
- Amazon ECS API リファレンス
- Amazon EBS パフォーマンスの向上
- Amazon EBS リソース
- EBS スナップショットの料金

追加情報

EBS スナップショットタイプ

Amazon EBS には、所有権とアクセスに基づいて 3 種類のスナップショットが用意されています。

- Owned by you – デフォルトでは、自分が所有するスナップショットからのみボリュームを作成できます。
- パブリックスナップショット – スナップショットは他のすべての AWS アカウントとパブリックに共有できます。スナップショットの権限を変更することで、指定した AWS アカウントとスナップショットを共有できます。許可を受けたユーザーは、共有するスナップショットを使用して自分の EBS ボリュームを作成できますが、元のスナップショットは影響を受けません。必要に応じ

て、暗号化されていないスナップショットをすべての AWS ユーザーに一般公開することもできます。暗号化されたスナップショットを公開することはできません。パブリックスナップショットは、個人データや機密データを公開する可能性があるため、重大なセキュリティリスクをもたらします。EBS スナップショットをすべての AWS アカウントと共有しないことを推薦します。DB スナップショットをコピーする方法については、AWS ドキュメントを参照してください。

- プライベートスナップショット – 指定した個々の AWS アカウントとスナップショットをプライベートに共有できます。スナップショットを特定の AWS アカウントとプライベートに共有するには、AWS ドキュメントの指示 に従い、権限設定でプライベートを選択します。許可を受けたユーザーは、共有するスナップショットを使用して自分の EBS ボリュームを作成できますが、元のスナップショットは影響を受けません。

概要と手順

次のテーブルには、未使用のスナップショットを見つけて削除することで EBS ボリュームコストを削減する方法や、頻繁または高速に取得する必要のない、ほとんどアクセスされないスナップショットをアーカイブする方法など、EBS スナップショットに関する詳細情報へのリンクが記載されています。

参考情報	「」を参照してください
スナップショット、その機能、制限事項	Amazon EBS スナップショットの作成
スナップショットを作成する方法	コンソール：スナップショットの作成 AWS CLI： スナップショット作成コマンド 例： `aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description " volume snapshot"`
スナップショットの削除 (一般情報)	Amazon EBS スナップショットの削除
スナップショットを削除するには	コンソール：スナップショットを削除 AWS CLI：スナップショットの削除コマンド 例：

```
aws ec2 delete-snapshot --snapshot-id
 snap-1234567890abcdef0
```

スナップショットのアーカイブ (一般情報)	Amazon EBS スナップショットのアーカイブ Amazon EBS Snapshots Archive (ブログ投稿)
スナップショットをアーカイブするには	コンソール：スナップショットをアーカイブ AWS CLI：modify-snapshot-tierコマンド
アーカイブされたスナップショットを取得する方法	コンソール：アーカイブされたスナップショットの復元 AWS CLI：restore-snapshot-tier コマンド
スナップショットの料金	Amazon EBS の料金

よくある質問

最小アーカイブ期間は？

最小アーカイブ期間は 90 日です。

アーカイブされたスナップショットを復元するにはどのくらいの時間がかかりますか?

スナップショットのサイズによっては、アーカイブ階層から標準階層にアーカイブスナップショットを復元するのに最大 72 時間かかる場合があります。

アーカイブされるスナップショットは、完全なスナップショットですか。

アーカイブされるスナップショットは、常に完全なスナップショットです。

ユーザーはどのスナップショットをアーカイブできますか?

アーカイブできるのは、アカウント内で所有しているスナップショットだけです。

登録済みの Amazon マシンイメージ (AMI) ルートデバイスボリュームにあるスナップショットは、アーカイブできますか。

登録済みの AMI ルートデバイスボリュームにあるスナップショットは、アーカイブできません。

スナップショットを共有する場合のセキュリティ上の考慮事項は?

スナップショットを共有すると、スナップショットのすべてのデータに他人がアクセスできるようになります。スナップショットの共有は、委託できる人とだけ行ってください。

スナップショットを別の AWS リージョンと共有する方法は?

スナップショットは、スナップショットが作成されたリージョンに制限されます。別のリージョンとスナップショットを共有するには、そのリージョンにスナップショットをコピーして、そのコピーを共有します。

暗号化されたスナップショットの共有方法?

デフォルトの AWS 管理キーで暗号化されたスナップショットを共有することはできません。共有できるのは、カスタマーマネージド型キーを使用して暗号化されたスナップショットだけです。暗号化されたスナップショットを共有する場合は、スナップショットの暗号化に使用するカスタマーマネージド型キーも共有する必要があります。

暗号化されていないスナップショットについてはどうですか?

暗号化されていないスナップショットのみをパブリックに共有できます。

その他のパターン

- で Landing Zone Accelerator を使用してアカウントの作成を自動化する AWS
- Amazon Bedrock を使用して AWS インフラストラクチャオペレーションを自動化する
- AWS リソース評価を自動化する
- 複数のアカウントとリージョンの AWS リソースを自動的にインベントリする
- AWS CDK を使用して AWS Service Catalog ポートフォリオと製品のデプロイを自動化する
- AWS Service Catalog と を使用して、Gitflow 環境にホットフィックスソリューションをデプロイするための動的パイプライン管理を自動化する AWS CodePipeline
- Terraform を使用して Amazon Managed Grafana での Amazon MWAA カスタムメトリクスの取り込みと視覚化を自動化する
- Amazon Inspector と を使用してクロスアカウントワークロードのセキュリティスキャンを自動化する AWS Security Hub
- Cloud Custodian と AWS CDK を使用して、Systems Manager の AWS マネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする
- 既存および新規の Amazon EBS ボリュームを自動的に暗号化する
- Amazon CloudWatch オブザーバビリティアクセスマネージャーを使用してモニタリングを一元化
- 起動時に EC2 インスタンスに必須タグが欠けていないか確認する
- AWS IoT 環境内のセキュリティイベントのログ記録とモニタリングを設定する
- Amazon ECS タスク定義を作成し、Amazon EFS を使用して EC2 インスタンスにファイルシステムをマウントする
- AWS CloudFormation Guard ポリシーを使用して AWS Config カスタムルールを作成する
- AWS CDK アスペクトとエスケープハッチを使用してデフォルトのロール名をカスタマイズする
- AWS Config と を使用して未使用の Amazon EBS ボリュームを削除する AWS Systems Manager
- と CloudFormation を使用して AWS Control Tower コントロールをデプロイ AWS CDK および管理します。
- Terraform を使用した AWS Control Tower コントロールのデプロイと管理
- AWS CodePipeline、AWS CodeCommit、AWS CodeBuild を使用して、複数の AWS リージョンにコードをデプロイする
- AWS CloudFormation テンプレートを使用して Amazon GuardDuty を条件付きで有効にする
- PowerShell を使用して AWS IAM アイデンティティセンターのアイデンティティとその割り当てに関するレポートをエクスポートする

- Troposphere を使用して AWS Config マネージドルールを含む AWS CloudFormation テンプレートを生成します
- SageMaker ノートブックインスタンスに、別の AWS アカウントの CodeCommit リポジトリへの一時的なアクセスを許可する
- Stonebranch ユニバーサルコントローラーと AWS Mainframe Modernizationを統合
- Step Functions と Lambda プロキシ関数を使用して AWS アカウント間で CodeBuild プロジェクトを起動する
- ACM を使用して Windows SSL 証明書をApplication Load Balancer に移行
- IAM ルートユーザーのアクティビティを監視する
- AWS CDK および GitHub Actions ワークフローを使用してマルチアカウントサーバーレスデプロイを最適化する
- AWS CodeCommit イベントからカスタムアクションを実行する
- 非ワークロードサブネット用のマルチアカウント VPC 設計でルーティング可能な IP スペースを節約
- ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする
- Amazon SES を使用して、単一メールアドレスで複数の AWS アカウントを登録する
- AWS Lambda オートメーションを使用して 全体の Amazon EC2 エントリを AWS アカウント から削除 AWS Managed Microsoft AD する
- AWS Lambda オートメーションを使用して AWS アカウント から同じ の Amazon EC2 AWS Managed Microsoft AD エントリを削除する
- コンテナを再起動せずにデータベースの認証情報をローテーションする
- AWS Fargate を使用して、イベント駆動型でスケジュール済みの大規模ワークロードを実行する
- オンプレミスの SMTP サーバーとデータベースメールを使用して、Amazon RDS for SQL Server データベースインスタンスに通知を送信します。
- AWS ParallelCluster 用の Grafana モニタリングダッシュボードを設定する
- AWS Organizations を使用して Transit Gateway アタッチメントに自動的にタグを付ける
- BMC ディスカバリークエリを使用して移行計画のために移行データを抽出
- を使用して PCI DSS 4.0 の運用上のベストプラクティスを検証する AWS Config
- Amazon QuickSight を使用してすべての AWS アカウントの IAM 認証情報レポートを視覚化します

メッセージとコミュニケーション

トピック

- Amazon MQ で RabbitMQ 設定を自動化する
- Amazon Connect コンタクトセンターのエージェントワークステーションの通話品質を向上
- その他のパターン

Amazon MQ で RabbitMQ 設定を自動化する

作成者: Yogesh Bhatia (AWS) と Afroz Khan (AWS)

概要

Amazon MQ は、多くの人気メッセージブローカーとの互換性を提供するマネージドメッセージブローカーサービスです。Amazon MQ を RabbitMQ と併用すると、Amazon Web Services (AWS) クラウドで管理される堅牢な RabbitMQ クラスターが提供され、複数のブローカーと設定オプションを使用できます。Amazon MQ は、可用性、安全性、スケーラビリティの高いインフラストラクチャを提供し、毎秒多数のメッセージを簡単に処理します。複数のアプリケーションが、さまざまな仮想ホスト、キュー、交換でインフラストラクチャを使用できます。ただし、これらの設定オプションの管理またはインフラストラクチャの手動作成には、時間と労力が必要になることがあります。このパターンでは、単一ファイルで、RabbitMQ の構成を 1 つの手順で管理する方法について説明します。このパターンで提供されるコードは、Jenkins または Bamboo などの継続的インテグレーション (CI) ツールに組み込みできます。

このパターンを使用して、任意の RabbitMQ クラスターを設定できます。必要なのはクラスターへの接続のみです。RabbitMQ 設定を管理する方法は他にも多くありますが、このソリューションではアプリケーション全体の設定をワンステップで作成するため、キューやその他の詳細を簡単に管理できます。

前提条件と制限

前提条件

- AWS コマンドラインインターフェイス (AWS CLI) がインストールされ、AWS アカウントを指すように設定されている (手順については、AWS CLI ドキュメントを参照)
- Ansible がインストールされている (プレイブックを実行して構成を作成できる)
- rabbitmqadmin がインストールされている (手順については、RabbitMQ ドキュメントを参照)
- 正常な Amazon CloudWatch メトリクスで作成されたAmazon MQ の RabbitMQ クラスター

その他の要件

- JSON の一部としてではなく、仮想ホストとユーザーの設定を別に作成します。
- 設定 JSON がリポジトリの一部であり、バージョン管理されていることを確認します。
- rabbitmqadmin CLI のバージョンは RabbitMQ サーバーのバージョンと同じである必要があるため、最善のオプションは RabbitMQ コンソールから CLI をダウンロードすることです。

- パイプラインの一部として、各実行前に JSON 構文が検証されていることを確認します。

製品バージョン

- AWS CLI バージョン 2.0
- Ansible バージョン 2.9.13
- rabbitmqadmin バージョン 3.9.13 (RabbitMQ サーバーバージョンと同じである必要があります)

アーキテクチャ

ソーステクノロジースタック

- 既存のオンプレミス仮想マシン (VM) または Kubernetes クラスター (オンプレミスまたはクラウド) で実行中の RabbitMQ クラスター

ターゲットテクノロジースタック

- Amazon MQ for RabbitMQ での RabbitMQ の自動設定

ターゲット アーキテクチャ

RabbitMQ を設定する方法は多くあります。このパターンでは、単一 JSON ファイルにすべての設定が含まれるインポート設定機能を使用します。このファイルにはすべての設定が適用され、Bitbucket または Git などのバージョン管理システムで管理できます。このパターンは Ansible を使用して、rabbitmqadmin CLI で設定を実装します。

ツール

ツール

- rabbitmqadmin は RabbitMQ HTTP ベースの API 用のコマンドラインツールです。RabbitMQ ノードとクラスターの管理と監視に使用されます。
- Ansible は、アプリケーションと IT インフラストラクチャを自動化するオープンソースツールです。
- AWS CLI では、コマンドラインシェルのコマンドを使用して AWS サービスとインタラクトできます。

AWS サービス

- Amazon MQ は、クラウドでメッセージブローカーを簡単にセットアップして操作できるマネージドメッセージブローカーサービスです。
- AWS CloudFormation は、AWS インフラストラクチャをセットアップし、コードとしてインフラストラクチャでクラウドプロビジョニングをスピードアップできます。

コード

このパターンで使用する JSON 設定ファイルと Ansible プレイブックのサンプルが添付ファイルで提供されます。

エピック

AWS インフラストラクチャを作成する

タスク	説明	必要なスキル
AWS に RabbitMQ クラスターを作成します。	RabbitMQ クラスターがない場合は、AWS CloudFormation を使用して AWS 上にスタックを作成できます。または、Ansible の Cloudformation モジュールを使用してスタックを作成できます。後者のアプローチでは、Ansible は、RabbitMQ インフラストラクチャの作成と設定の管理の両方のタスクに使用できます。	AWS CloudFormation、Ansible

Amazon MQ for RabbitMQ 設定を作成する

タスク	説明	必要なスキル
プロパティファイルを作成します。	添付ファイルの JSON 設定ファイル (`rabbitmqc`	JSON

タスク	説明	必要なスキル
	onfig.json) をダウンロード、または RabbitMQ コンソールからエクスポートします。 変更して、キュー、交換、バインドを設定します。 この設定ファイルでは、以下が示されます。 — 2 つのキュー、sample-queue1 および sample-queue2 の作成 — 2 つの交換、sample-exchange1 および sample-exchange2 の作成 — キューと交換間のバインドの実装 これらの設定は、rabbitmqadmin が要求するルート (/) 仮想ホストで実行されます。	

タスク	説明	必要なスキル
Amazon MQ for RabbitMQ インフラストラクチャの詳細を取得します。	AWS 上の RabbitMQ インフラストラクチャの以下の詳細を取得します。 • ブローカー名 • RabbitMQ ホスト • RabbitMQ ユーザー名 (クラスター作成時に作成された管理者ユーザー) • RabbitMQ パスワード AWS マネジメントコンソールまたは AWS CLI を使用して、この情報を取得できます。これらの詳細により、Ansible プレイブックが AWS アカウントに接続され、RabbitMQ クラスタを使用してコマンドを実行できます。 ⚠ Important Ansible プレイブックを実行するコンピュータは AWS アカウントにアクセスできる必要があり、前提条件セクションで説明されているように、AWS CLI がすでに構成されている必要があります。	AWS CLI、Amazon MQ

タスク	説明	必要なスキル

hosts_var ファイルを作成します。	Ansible の hosts_var ファイルを作成し、すべての変数がファイルで定義されていることを確認します。Ansible Vault を使用してパスワードを保存することを検討します。hosts_var ファイルは次のように設定できます (アスタリスクと情報を入れ替えます)。
	`RABBITMQ_HOST: "**********.mq.us-east-2.amazonaws.com"`
	`RABBITMQ_VHOST: "/"`
	`RABBITMQ_USERNAME: "admin"`
	`RABBITMQ_PASSWORD: "*******"`
		Ansible

タスク	説明	必要なスキル
Ansible プレイブックを作成します。	サンプルプレイブックについては、添付の `ansible-rabbit-config.yaml` を参照してください。このファイルをダウンロードして保存します。Ansible プレイブックは、アプリケーションが必要とするキュー、交換、バインドなどのすべての RabbitMQ 設定をインポートして、管理します。 パスワードの保護など、Ansible プレイブックのベストプラクティスに従います。パスワードの暗号化には Ansible Vault を使用し、暗号化されたファイルから RabbitMQ パスワードを取得します。	Ansible

設定をデプロイする

タスク	説明	必要なスキル
プレイブックを実行します。	前のエピックで作成した Ansible プレイブックを実行します。 `ansible-playbook ansible-rabbit-config.yaml`	RabbitMQ、Amazon MQ、Ansible

タスク	説明	必要なスキル
	RabbitMQ コンソールで新しい設定を確認できます。	

関連リソース

- RabbitMQ から Amazon MQ へ移行する (AWS ブログ投稿)
- 管理コマンドラインツール (RabbitMQ ドキュメント)
- AWS CloudFormation スタックを作成または削除する (Ansible ドキュメント)
- RabbitMQ for Amazon MQ へメッセージ駆動型アプリケーションを移行する (AWS ブログ投稿)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon Connect コンタクトセンターのエージェントワークステーションの通話品質を向上

アーネスト・オズドバ (AWS) によって作成された

概要

通話品質の問題は、コンタクトセンターでトラブルシューティングするのが最も難しい問題の1つです。音声品質の問題や複雑なトラブルシューティング手順を回避するには、エージェントの作業環境とワークステーションの設定を最適化する必要があります。このパターンは、Amazon Connect コンタクトセンターのエージェントワークステーションの音声品質最適化手法を説明しています。以下の領域での推奨事項が記載されています。

- 作業環境の調整。エージェントの周囲は、ネットワーク上での音声の伝送方法には影響しませんが、通話品質には影響します。
- エージェントワークステーションの設定 コンタクトセンターワークステーションのハードウェアとネットワークの構成は、通話品質に大きな影響を与えます。
- ブラウザ設定。エージェントはウェブブラウザを使用して Amazon Connect コンタクトコントロールパネル (CCP) ウェブサイトにアクセスし、顧客と通信します。そのため、ブラウザの設定が通話品質に影響する可能性があります。

次のコンポーネントも通話品質に影響する可能性がありますが、これらはワークステーションの範囲外であり、このパターンには含まれていません。

- AWS Direct Connect、フルトンネル VPN、またはスプリットトンネル VPN を介したAmazon Web Services (AWS) クラウドへのトラフィックフロー
- 会社のオフィス内外で作業する場合のネットワーク条件
- 公衆電話交換網 (PSTN) 接続
- お客様のデバイスとテレフォニーキャリア
- 仮想デスクトップインフラストラクチャ (VDI) セットアップ

これらの分野に関する詳細については、Amazon Connect ドキュメントの「コンタクトコントロールパネル (CCP) に関する一般的な問題」と「エンドポイントテストユーティリティの使用」を参照してください。

前提条件と制限

前提条件

- ヘッドセットとワークステーションは、「Amazon Connect 管理者ガイド」で指定されている要件に準拠している必要があります。

制約事項

- このパターンの最適化手法は、ソフトフォンの音声品質にも適用されます。Amazon Connect CCP をデスクフォンモードに設定した場合は適用されません。ただし、ソフトフォンの設定で通話に適した音質が得られない場合は、デスクフォンモードを使用できます。

製品バージョン

- サポートされているブラウザとバージョンについては、「Amazon Connect 管理者ガイド」を参照してください。

アーキテクチャ

このパターンはエージェントワークステーションの設定を対象としているため、アーキテクチャに依存しません。次の図が示すように、エージェントから顧客への音声パスは、エージェントのヘッドセット、ブラウザ、オペレーティングシステム、ワークステーションハードウェア、ネットワークの影響を受けます。

Amazon Connect コンタクトセンターでは、ユーザーの音声接続は WebRTC で確立されます。音声は Opus インタラクティブオーディオコーデックでエンコードされ、転送中はセキュア・リアルタイム・トランスポート・プロトコル (SRTP) で暗号化されます。VPN、プライベート WAN/LAN、ISP ネットワークなど、他のネットワークアーキテクチャも可能です。

ツール

- Amazon Connect エンドポイントテストユーティリティ — このユーティリティは、ネットワーク接続とブラウザの設定をチェックします。
- WebRTC 設定用のブラウザ設定エディター：
  - Firefox の場合：「設定に関する」

  - クロームの場合：「chrome： //フラグ」
- CCP ログパーサー — このツールは、トラブルシューティングを目的として CCP ログを分析するのに役立ちます。

エピック

作業環境を調整します。

タスク	説明	必要なスキル
バックグラウンドノイズを軽減。	騒がしい環境は避けてください。それが不可能な場合は、以下の防音対策のヒントを参考にして環境を最適化してください。 • カーテン、カーペット、柔らかい家具など、音を消散する表面を使用してノイズを吸収してください。 • 机の間に障壁を設けて騒音を遮断します。 • 集中力を高めてプライバシーを確保するために、ホワイトノイズジェネレーターなどのアクティブノイズキャンセリング (ANC) ソリューションを検討するか、ノイズキャンセリングヘッドセットの使用を検討してください。 • 通話中にエコーが発生しないようにしてください。空いている大きなスペースは、エコー効果を生み出したり、ノイズを増幅したりす	エージェント、マネージャー

タスク	説明	必要なスキル
	ることがあります。サウンドが跳ね返るような表面を覆うと、エコーを減らすのに役立ちます。	

エージェントワークステーションの設定を最適化

タスク	説明	必要なスキル
適切なヘッドセットを選択する	• 騒がしい環境では、ステレオヘッドセットを選択してください。両耳に音を向けると、エージェントは集中して顧客の声を聞きやすくなり、エージェントが声を上げる可能性が低くなるため、全体的なノイズが軽減されます。 • 大音量のスピーカーや内蔵コンピューターオーディオの使用は避けてください。最高の品質を得るには、コンタクトセンター専用の有線ヘッドセットを使用してください。ワイヤレスヘッドセットは便利ですが、電波干渉やトランスコーディングが原因で、オーディオの遅延が増えたり、音質が低下したりする可能性があります。	エージェント、マネージャー
ヘッドセットを意図したとおりに使用してください。	• ヘッドセットのアクティブノイズキャンセリング機能	Agent

タスク	説明	必要なスキル
	と音声強調機能が利用できる場合は、有効にしてください。ANC や ANR などの設定を探してください。これらの設定を有効にする手順については、ヘッドセットのユーザーマニュアルを参照してください。 • 直接話せるようにマイクを調整してください。マイクの最適な位置は、あごのすぐ下です。正しく配置すると、サウンドレベルに10デシベル (dB) の差が生じる可能性があります。ほとんどのヘッドセットではマイクアーム (ブーム) を回転させたり曲げたりできるので、話しているときはマイクを正しい位置に置いておくことが重要です。 • ヘッドセットの中には、複数のマイクとボイスビームフォーミングなどの高度な機能を備えているものもあり、スムーズに音声をキャプチャできます。メーカーが意図したとおりにメインマイクを使用していることを確認するには、デバイスのユーザーマニュアルを参照してください。	

タスク	説明	必要なスキル
ワークステーションのリソースを確認してください。	エージェントのコンピューターのパフォーマンスが高いことを確認してください。リソースを消費するサードパーティのアプリケーションを使用している場合、そのコンピュータは CCP を実行するための最小「ハードウェア要件」を満たしていない可能性があります。エージェントが通話品質に問題を抱えている場合は、CCP に十分な処理能力 (CPU)、ディスク容量、ネットワーク帯域幅、メモリがあることを確認してください。担当者は、CCP のパフォーマンスと通話品質を向上させるために、不要なアプリケーションやタブをすべて閉じてください。	管理者

タスク	説明	必要なスキル
オペレーティングシステムのサウンド設定を行います。	通常、マイクレベルとブーストのデフォルト設定は正常に機能します。発信音声が小さい場合や、マイクの音量が大きすぎる場合は、これらの設定を調整すると役立つ場合があります。マイクの設定は、コンピュータのシステムサウンド設定 (「サウンド」、「macOS」では「入力」、「Windows」では「マイクのプロパティ」) にあります。音声品質に影響する可能性のある詳細設定には、システムツールまたはサードパーティアプリケーションからアクセスできます。確認できる設定の一部を以下に紹介します。 • サンプルレート — この値によって、1 秒間にサウンドをプローブする回数が決まります。デフォルト設定は、通常 44 または 48 キロヘルツ (kHz) です。Amazon Connect の最適値は 48 kHz です。ブラウザの設定を使用してデフォルト値を上書きできます。詳細については、「Amazon Connect 管理者ガイド」の「トラブルシューティングセクション」を参照してください。 • ゲイン — この値によって、マイクがサウンドをどれだ	エージェント、管理者

タスク	説明	必要なスキル
	け増幅するかが決まります。ゲインを上げると、マイクが拾うバックグラウンドノイズが増える可能性があります。 • ビット深度 — このデジタル解像度の値は、認識される音の振幅レベルを表します。ビット深度が高いほど音質が良くなります。ただし、従来のテレフォニーネットワークの多くは、8 ビットの解像度しかサポートしないパルスコード変調 (PCM) 規格を使用しています。 • オープンスレッショルド — これはマイクが拾う最小音の振幅です。 音声品質に問題がある場合は、詳しく調べる前にこれらの値をデフォルト設定に戻してみてください。 これらや他の調整可能な設定の詳細については、デバイスのマニュアルを参照してください。	

タスク	説明	必要なスキル
有線ネットワークを使用する。	通常、有線イーサネットは遅延が少ないため、音声データ伝送に必要な一貫した伝送品質を実現しやすくなります。 1 コールあたり最低 100 KB の帯域幅を推奨します。 • エージェントが在宅勤務の場合は、ワイヤレス接続による有線接続をお勧めします。顧客の声を聞くのに 150 ミリ秒以上はかからないはずです。Amazon Connect のレイテンシーテストには、「Amazon Connect エンドポイントテストユーティリティ」からアクセスできます。ただし、このユーティリティはブラウザから Amazon Connect リージョンまでの遅延を測定し、顧客への遅延は測定しません。150 ミリ秒の片道遅延を推奨することで、エージェントと顧客がお互いに話し合うことを防ぎます。値は端から端まで測定され、Amazon Connect リージョンと顧客間の通話の一部を含め、各要素によって遅延が生じます。 • エージェントがオフィスから仕事をしている場合、パラメータが推奨範囲内にあ	ネットワーク管理者、エージェント

タスク	説明	必要なスキル
	り、リアルタイムトランスポートプロトコル (RTP) トラフィックが優先される限り、企業の Wi-Fi は許容されます。	
ハードウェアドライバーを更新します。	USB など、独自のファームウェアを備えたヘッドセットを使用する場合は、常に最新バージョンに更新しておくことをお勧めします。補助ポートを使用するシンプルなヘッドセットは、コンピューターの内蔵オーディオデバイスを使用するため、オペレーティングシステムのハードウェアドライバーが最新であることを確認してください。まれに、オーディオドライバーを更新するとオーディオの問題が発生し、ロールバックが必要になることがあります。ファームウェアとドライバーのバージョン変更の詳細については、デバイスのマニュアルを参照してください。	管理者

タスク	説明	必要なスキル
USB ハブやドングルは避けてください。	ヘッドセットを接続するときは、ドングル、ポートタイプコンバーター、ハブ、延長ケーブルなどの追加のデバイスを避けてください。 これらのデバイスは通話品質に影響する可能性があります。代わりに、デバイスをコンピューターのポートに直接 Connect してください。	Agent

タスク	説明	必要なスキル
CCP ログの確認	CCP ログパーサーを使用すると、アプリケーションログを簡単にチェックできます。 1. 通話後に「CCP ログをダウンロード」します。 2. 「CCP ログパーサー」を開きます。 3. ログファイルをドラッグアンドドロップして、分析用のログをアップロードします。 4. ログが分析されると、デフォルトで「スナップショットとログ」タブが選択されます。ログの横にある「メトリクス」タブを選択すると、分析情報を確認できます。 5. 「WebRTC メトリクス-オーディオ入力」セクションで、次の点を確認してください。 • 「オーディオレベル」グラフで、受信したオーディオレベルが 0 を超えているかどうかを確認します。これは発信者から音声が受信されたことを示します。 • 失われた「パケット」をすべて表示するパケットグラフ。このグラフに大幅な増加が見られる場合	エージェント (上級スキル)

タスク	説明	必要なスキル
	は、IT サポートチームに連絡してください。 6. 「WebRTC メトリクス-オーディオ出力」セクションで、次の点を確認してください。 • 「オーディオレベル」グラフで、デバイスからオーディオが送信されたことを確認します。 • パケットグラフ。パケットロスが急増した場合は、IT サポートチームに報告してください。 • 「ジッターバッファと RTT 」グラフ。ラウンドトリップタイム (RTT) の値が 300 を超えると、通話体験に影響します。これらを IT サポートチームに報告してください。	

ブラウザ設定の最適化

タスク	説明	必要なスキル
デフォルトの WebRTC 設定を復元します。	CCP でソフトフォンコールを行うには、WebRTC を有効にする必要があります。WebRTC 関連機能のデフォルト設定のままにすることをお勧めします。	管理者

タスク	説明	必要なスキル
	• Chrome では、「chrome：//flags」というURLにアクセスしてフラグを設定できます。検索ボックスに「WebRTC」と入力して、CCP に干渉する可能性のある設定を探し、これらを「デフォルト」に設定します。 • Firefox では、アドレスバーに「設定に関する」と入力し、設定ページの検索ボックスに「WebRTC」と入力します。デフォルト以外の設定は太字で表示され、「デフォルト」に変更できます。	
トラブルシューティングの際はブラウザ拡張機能を無効にしてください。	ブラウザ拡張機能の中には、通話品質に影響を与えたり、通話が正しく接続されなかったりするものもあります。ブラウザのシークレットウィンドウまたはプライベートモードを使用し、すべての拡張機能を無効にしてください。これで問題が解決したら、ブラウザの拡張機能を見直して疑わしいアドオンを探すか、個別に無効にしてください。	エージェント、管理者

タスク	説明	必要なスキル
ブラウザーのサンプルレートを確認してください。	マイク入力が最適な 48 kHz サンプルレートに設定されていることを確認します。手順については、Amazon Connect 管理者ガイド を参照してください。	エージェント、管理者

関連リソース

このパターンの手順を実行しても通話品質に問題がある場合は、以下のリソースを参照してトラブルシューティングのヒントを確認してください。

- 問い合わせコントロールパネル (CCP) の一般的な問題を参照してください。
- 「エンドポイントテストユーティリティ」で接続を確認してください。
- その他の問題については、「トラブルシューティングガイド」に従ってください。

トラブルシューティングと調整を行っても通話品質の問題が解決しない場合、根本原因はワークステーションの外部にある可能性があります。詳細なトラブルシューティングについては、IT サポートチームにお問い合わせください。

その他のパターン

- CQRS とイベントソーシングを使用してモノリスをマイクロサービスに分解する
- ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation
- Amazon API Gateway を Amazon SQS と統合して非同期 REST APIs を処理する
- Amazon SES を使用して、単一メールアドレスで複数の AWS アカウントを登録する
- AWS Fargate を使用してメッセージ駆動型の大規模なワークロード実行する
- 自動ワークフローを使用して Amazon Lex ボットの開発とデプロイを合理化する

セキュリティ、アイデンティティ、コンプライアンス

トピック

- Amazon Cognito アイデンティティプールを使用して ASP.NET Core アプリケーションから AWS のサービスにアクセスする
- AWS Directory Service を使用して Amazon EC2 の Microsoft SQL Server を認証する
- インシデント対応とフォレンジックを自動化する
- AWS Security Hub 標準検出結果の修正を自動化する
- Amazon Inspector と を使用してクロスアカウントワークロードのセキュリティスキャンを自動化する AWS Security Hub
- パブリック IP アドレスからのアクセスを許可する AWS セキュリティグループを自動的に監査する
- AWS Config のカスタム修復ルールを使用して AWS CloudTrail を自動的に再有効化する
- 暗号化されていない Amazon RDS DB インスタンスとクラスターを自動的に修正する
- AWS Organizations と AWS Secrets Manager を使用して IAM ユーザーアクセスキーを大規模に自動的にローテーションする
- CodePipeline、IAM アクセスアナライザー、AWS CloudFormation マクロを使用して、AWS アカウントの IAM ポリシーとロールを自動的に検証してデプロイする
- AWS Security Hub と Jira ソフトウェアを双方向に統合する
- EC2 Image Builder と Terraform を使用して、強化されたコンテナイメージ用のパイプラインを構築する
- Terraform を使用して AWS Organizations の IAM アクセスキー管理を一元化する
- Amazon CloudFront ディストリビューションのアクセスロギング、HTTPS、TLS バージョンを確認する
- IPv4 および IPv6 対応セキュリティグループのイングレスルール内の単一ホストネットワークエントリを確認する
- エンタープライズアプリケーションの Amazon Cognito 認証フローを選択する
- AWS CloudFormation Guard ポリシーを使用して AWS Config カスタムルールを作成する
- 複数の から Prowler セキュリティ検出結果の統合レポートを作成する AWS アカウント
- AWS Config と を使用して未使用の Amazon EBS ボリュームを削除する AWS Systems Manager
- と CloudFormation を使用して AWS Control Tower コントロールをデプロイ AWS CDK および管理します。

- Terraform を使用した AWS Control Tower コントロールのデプロイと管理
- 複数のコード成果物のセキュリティ問題を同時に検出するパイプラインをデプロイする
- を使用してパブリックサブネットの検出属性ベースのアクセスコントロールをデプロイする AWS Config
- パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする
- Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする
- CA 証明書の有効期限が切れている Amazon RDS および Aurora データベースインスタンスを検出する
- Step Functions を使用して IAM アクセスアナライザーで IAM ポリシーを動的に生成
- AWS CloudFormation テンプレートを使用して Amazon GuardDuty を条件付きで有効にする
- Amazon RDS for SQL Server で透過的なデータ暗号化を有効にする
- AWS ロードバランサーが安全なリスナープロトコル (HTTPS、SSL/TLS) を使用していることを確認する
- Amazon EMR 保管中のデータの暗号化が起動時に有効になっていることを確保
- IAM プロファイルが EC2 インスタンスと確実に関連付けられているようにします。
- Amazon Redshift クラスターが作成時に暗号化されていることを確保
- PowerShell を使用して AWS IAM アイデンティティセンターのアイデンティティとその割り当てに関するレポートをエクスポートする
- 予定されている AWS KMS キーの削除を監視して修正する
- Security Hub AWS Organizations を使用して でパブリック Amazon S3 バケットを特定する
- Microsoft Sentinel で AWS セキュリティログを取り込んで分析する
- を使用してアクセス AWS IAM Identity Center 許可セットをコードとして管理する AWS CodePipeline
- AWS Secrets Manager を使用して認証情報を管理
- セキュリティグループの ElastiCache クラスターをモニタリング
- Amazon EMR クラスターの起動時に転送中の暗号化をモニタリングする
- Amazon ElastiCache クラスターの保存時の暗号化をモニタリングする
- AWS Config を使用して EC2 インスタンスのキーペアを監視する
- IAM ルートユーザーのアクティビティを監視する
- IAM ユーザーが作成されたときに通知を送信

- サービスコントロールポリシーを使用して、アカウントレベルでのインターネットアクセスを防止する
- を使用して IP アドレスまたは位置情報に基づいてアクセスを制限する AWS WAF
- git-secrets を使用して Git リポジトリに機密情報やセキュリティ上の問題がないかスキャンする
- AWS Network Firewall から Slack チャネルにアラートを送信
- AWS Private CA と AWS RAM を使用してプライベート証明書の管理を簡素化する
- マルチアカウント環境ですべてのセキュリティハブのメンバーアカウントにわたって、セキュリティ標準コントロールをオフにする
- PowerShell を使用して AWS IAM アイデンティティセンターから AWS CLI 認証情報を更新
- AWS Config を使用して Amazon Redshift のセキュリティ設定をモニタリング
- Network Firewall を使用して、アウトバウンドトラフィックのサーバー名表示から DNS ドメイン名をキャプチャする
- Terraform を使用すると、組織の Amazon GuardDuty が自動的に有効になります。
- を使用して PCI DSS 4.0 の運用上のベストプラクティスを検証する AWS Config
- 新しい Amazon Redshift クラスターに必要な SSL エンドポイントがあることを確認する
- 新しい Amazon Redshift クラスターが VPC で起動することを検証
- その他のパターン

Amazon Cognito アイデンティティプールを使用して ASP.NET Core アプリケーションから AWS のサービスにアクセスする

作成者：Bibhuti Sahu (AWS) と Marcelo Barbosa (AWS)

概要

このパターンでは、Amazon Cognito ユーザープールと ID プールを設定し、認証が AWS 成功した後に リソースにアクセスするために ASP.NET://www.comcore アプリを有効にする方法について説明します。

Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可とユーザー管理を提供します。Amazon Cognito の主な 2 つのコンポーネントは、ユーザープールとアイデンティティプールです。

ユーザープールは、Amazon Cognito のユーザーディレクトリです。ユーザープールを使用すると、ユーザーは Amazon Cognito 経由でウェブまたはモバイルアプリにログインできます。また、ユーザーは Google、Facebook、Amazon、Apple などのソーシャル ID プロバイダー、および SAML ベースの ID プロバイダー経由でユーザープールにサインインすることもできます。

Amazon Cognito ID プール (フェデレーティッド ID) は、ユーザーのために一意の ID を作成し、ID プロバイダーでそれらをフェデレートすることを可能にします。ID プールを使用すると、他のにアクセスするための一時的な制限付き特権 AWS 認証情報を取得できます AWS のサービス。新しい Amazon Cognito ID プールの使用を開始する前に、1 つ以上の AWS Identity and Access Management (IAM) ロールを割り当てて、アプリケーションユーザーに AWS リソースに対するアクセスレベルを決定する必要があります。ID プールでは、認証された ID と認証されていない ID という 2 種類の ID が定義されます。それぞれのアイデンティティタイプは、IAMで独自のロールを割り当てることができます。認証された ID は、パブリックログインプロバイダー (Amazon Cognito ユーザープール、Facebook、Amazon、Google、SAML、任意の OpenID Connect プロバイダ) によって認証されたユーザー、または開発者プロバイダー (独自のバックエンド認証プロセス) によって認証されたユーザーに属します。リクエストを受信した Amazon Cognito は、このサービスがそのリクエスト認証済みになっているかを判断し、どのロールがその認証タイプに関連付けられているかを判断し、そのロールに添付されているポリシーを使用してリクエストに応答します。

前提条件と制限

前提条件

- Amazon Cognito および IAM アクセス許可 AWS アカウント を持つ

- 使用する AWS リソースへのアクセス
- ASP.NET Core 2.0.0 以降

アーキテクチャ

テクノロジースタック

- Amazon Cognito
- ASP.NET Core

ターゲット アーキテクチャ

ツール

ツール、SDKs、および AWS のサービス

- Visual Studio または Visual Studio Code
- 「Amazon.AspNetCore.Identity.Cognito (1.0.4)」 – NuGet パッケージ
- AWSSDK.S3 (3.3.110.32)」 – NuGet パッケージ
- Amazon Cognito

コード

添付の.zip ファイルには、以下を説明するサンプルファイルが含まれています。

- ログインしているユーザーのアクセストークンを取得する方法
- アクセストークンを AWS 認証情報と交換する方法
- AWS 認証情報を使用して Amazon Simple Storage Service (Amazon S3) サービスにアクセスする方法

認証された ID の IAM ロール

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "mobileanalytics:PutEvents",
        "cognito-sync:*",
        "cognito-identity:*",
        "s3:ListAllMyBuckets*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

エピック

Amazon Cognito ユーザープールを作成する

タスク	説明	必要なスキル
ユーザープールを作成します。	1. にサインイン AWS Management Console し、Amazon Cognito コンソールを開きます。 2. [Manage User Pools] (ユーザープールの管理) をクリックします。 3. ページの右上にある [ユーザープールを作成する] を選択します。 4. ユーザープールの名前を入力し、デフォルトを確認するを選択して、プールを作成するを選択します。 5. プール ID をメモします。	開発者
アプリクライアントの追加	ユーザーのサインアップとサインインに組み込みウェブ	開発者

タスク	説明	必要なスキル
	ページを使用するアプリを作成できます。 1. ユーザープール ページの左側にあるナビゲーションバーで、[全般設定] の下の [アプリ クライアント] を選択し、[アプリ クライアントの追加] を選択します。 2. アプリに名前を付けて、[アプリ クライアントの作成] を選択します。 3. アプリクライアント ID とクライアントシークレットをメモします (S詳細を表示を選択すると、クライアントシークレットが表示されます)。	

Amazon Cognito アイデンティティプールを作成する

タスク	説明	必要なスキル
ID プールを作成します。	1. Amazon Cognito コンソールで、[ID プールの管理] を選択し、[新しいアイデンティティプールの作成] を選択します。 2. アイデンティティプールの名前を入力します。 3. 認証されていないアイデンティティを有効にしたい場合は、認証されていないアイデンティティセクション	開発者

タスク	説明	必要なスキル
	からそのオプションを選択します。 4. 認証プロバイダーセクションで、ユーザープール ID とアプリクライアント ID を設定して Amazon Cognito ID プールを設定し、プールの作成を選択します。	
アイデンティティプールに IAM ロールを割り当てます。	認証されたユーザーと認証されていないユーザーの IAM ロールを編集するか、デフォルトのままにして許可を選択できます。このパターンでは、認証済みの IAM ロールを編集し、`s3:ListAllMyBuckets` へのアクセスを提供します。サンプルコードについては、前述のツールセクションで説明した IAM ロールを参照してください。	開発者
アイデンティティプール ID をコピーします。	前の手順で許可を選択すると、Amazon Cognito の使用開始方法ページが表示されます。このページでは、AWS認証情報の取得セクションからアイデンティティープール ID をコピーするか、右上のアイデンティティープールの編集を選択して表示される画面からアイデンティティープール ID をコピーできます。	開発者

サンプルアプリの設定

タスク	説明	必要なスキル
サンプルの ASP.NET Core ウェブアプリケーションを複製	1. 「https://github.com/aws/aws-aspnet-cognito-identity-provider.git」からサンプルの .NET Core Web アプリを複製します。 2. samples フォルダに移動し、ソリューションを開きます。このプロジェクトでは、appsettings.json ファイルを設定し、サインインできるとすべての S3 バケットをレンダリングする新しいページを追加します。	開発者
依存関係を追加します。	ASP.NET Core アプリケーションに Amazon.AspNetCore.Identity.Cognito のNuGet 依存関係を追加します。	開発者
設定キーと値を に追加しますappsettings.json 。	添付 appsettings.json ファイルのコードを appsettings.json ファイルに含め、プレースホルダを前の手順の値に置き換えます。	開発者
新しいユーザーを作成してサインインします。	Amazon Cognito ユーザープールに新しいユーザーを作成し、そのユーザーがユーザープールのユーザーとグルー	開発者

タスク	説明	必要なスキル
	プに存在することを確認します。	
という名前の新しい Razor ページを作成します `MyS3Buckets` 。	新しい ASP.NET Core Razor ページをサンプルアプリに追加し、添付されているサンプルのコンテンツと添付サンプルのコンテンツを `MyS3Bucket.cshtml` と `MyS3Bucket.cshtml.cs` に置き換えます。`_Layout.cshtml` ページのナビゲーションの下に新しい MyS3bucket ページを追加します。	開発者

トラブルシューティング

問題	ソリューション
GitHub リポジトリからサンプルアプリケーションを開いた後、NuGet パッケージを Samples プロジェクトに追加しようとすると、エラーが発生します。	`src` フォルダーで、`Amazon.AspNetCore.Identity.Cognito` プロジェクトへの参照が `Samples.sln` ファイルから削除されていることを確認します。その後、NuGet パッケージをサンプルプロジェクトに問題なく追加できます。

関連リソース

- Amazon Cognito
- Amazon Cognito ユーザープール
- 「Amazon Cognito アイデンティティプール」
- 「アクセスポリシーの例」
- 「GitHub - AWS ASP.NET Cognito Identity Provider」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Directory Service を使用して Amazon EC2 の Microsoft SQL Server を認証する

作成者: Jagadish Kantubugata (AWS) および Oludahun Bade Ajidahun (AWS)

概要

このパターンでは、AWS Directory Service ディレクトリを作成し、それを使用して Amazon Elastic Compute Cloud (Amazon EC2) インスタンス上で Microsoft SQL Server を認証する方法を説明します。

AWS Directory Service は、Amazon Cloud Directory と Microsoft Active Directory を他の AWS サービスと併用する複数の方法を提供します。ディレクトリはユーザー、グループ、デバイスに関する情報を保存し、管理者はそれを使用して情報とリソースへのアクセスを管理します。AWS Directory Service は、既存の Microsoft AD または Lightweight Directory Access Protocol (LDAP) 対応のアプリケーションをクラウドで使用するユーザーに複数のディレクトリの選択肢を提供します。また、ユーザー、グループ、デバイス、アクセスを管理するディレクトリーが必要な開発者に、その選択肢を提供します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 最低 2 つのプライベートサブネットと 2 つのパブリックサブネットがある仮想プライベートクラウド (VPC)
- サーバーをドメインに結合する AWS Identity and Access Management (IAM) ロール

アーキテクチャ

ソーステクノロジースタック

- ソースはオンプレミス Active Directory でも可能

ターゲットテクノロジースタック

- AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD)

ターゲット アーキテクチャ

ツール

- SQL Server Management Studio (SSMS) は、SQL サーバーコンポーネントへのアクセス、設定、管理など、Microsoft SQL Server を管理するツールです。

エピック

ディレクトリをセットアップする

タスク	説明	必要なスキル
ディレクトリタイプとして AWS Managed Microsoft AD を選択します。	AWS Directory Service コンソールで、[Directories (ディレクトリ)]、[Set up directory (ディレクトリをセットアップ)]、[AWS Managed Microsoft AD]、[Next (次へ)] を選択します。	DevOps
エディションを選択します。	AWS Managed Microsoft AD の使用可能なエディションから、[Standard Edition (標準エディション)] を選択します。	DevOps
ディレクトリの DNS 名を指定します。	完全修飾ドメイン名を使用します。この名前は、VPC 内でのみ解決されます。パブリックに解決可能である必要はありません。	DevOps
管理者パスワードを設定します。	Admin という名前のデフォルト管理ユーザーのパスワードを設定します。	DevOps

タスク	説明	必要なスキル
VPC とサブネットを選択します。	ディレクトリとドメインコントローラーのサブネットを含む VPC を選択します。サブネットが少なくとも 2 つある VPC がない場合は、1 つ作成する必要があります。	DevOps
ディレクトリを確認して起動します。	ディレクトリのエディションと価格情報を確認してから、[Create directory (ディレクトリの作成)] を選択します。	DevOps

ドメイン内で SQL サーバー用の EC2 インスタンスを起動する

タスク	説明	必要なスキル
SQL サーバーの AMI を選択します。	このエピックの手順は、Windows EC2 インスタンスを AWS Managed Microsoft AD ディレクトリにシームレスに結合します。 Amazon EC2 コンソールで、[Launch instance (インスタンスを起動)] を選択してから、SQL サーバーの適切な Amazon マシンイメージ (AMI) を選択します。	DevOps、DBA
インスタンスの詳細を設定します。	SQL サーバーの要件を満たすように Windows インスタンスを設定します。	DevOps、DBA
キーペア名を選択します。	キーペアを選択してから、インスタンスを起動します。	DevOps、DBA

タスク	説明	必要なスキル
ネットワークを追加します。	ディレクトリが作成された VPC を選択できます。	DevOps、DBA
IAM ロールを選択します。	[Advanced settings (詳細設定)] で、AWS マネージドポリシー `AmazonSSMManagedInstanceCore` があり、`AmazonSSMDirectoryServiceAccess` が添付されている IAM プロファイルを選択します。	DevOps、DBA
サブネットを追加します。	VPC 内のいずれかのパブリックサブネットを選択します。選択するサブネットは、すべての外部トラフィックがインターネットゲートウェイにルーティングされている必要があります。そうでない場合は、インスタンスにリモート接続できません。	DevOps、DBA
ドメインを選択します。	[Domain join (ドメイン結合)] ディレクトリリストから作成したドメインを選択します。	DevOps、DBA
インスタンスを起動します。	[Launch instance (インスタンスの起動)] を選択します。	DBA

ディレクトリサービスを使用して SQL サーバーを認証する

タスク	説明	必要なスキル
Windows 管理者としてログインします。	Windows 管理者の認証情報を使用して、Windows EC2 インスタンスにログインします。	DBA
SQL サーバーにログインします。	SQL Server Management Studio (SSMS) を起動し、Windows 認証方法を使用して SQL サーバーにログインします。	DBA
ディレクトリユーザーのログインを作成します。	SSMS で、[Security (セキュリティ)] を選択してから、[New Login (新規ログイン)] を選択します。	DBA
ログイン名を検索します。	ログインテキストボックスの横にある検索ボタンを選択します。	DBA
場所を選択します。	[Select User or Group (ユーザーまたはグループの選択)] ダイアログボックスで、[Locations (場所)] を選択します。	DBA
ネットワーク認証情報を入力します。	ディレクトリサービスを作成したときに使用した完全修飾ネットワーク認証情報を入力します。たとえば、`test.com\admin` 。	DBA
ディレクトリを選択します。	AWS ディレクトリ名を選択してから、[OK] を選択します。	DBA

タスク	説明	必要なスキル
オブジェクト名を選択します。	ログインを作成するユーザーを選択します。場所を選択し、ディレクトリ全体を選択し、ユーザーを検索してログインを追加します。	DBA
SQL Server インスタンスにログインします。	ドメイン認証情報を使用して SQL Server 用の Windows EC2 インスタンスにログインします。	DBA
SQL Server にドメインユーザーとしてログインします。	SSMS を起動し、Windows 認証方法を使用してデータベースエンジンに接続します。	DBA

関連リソース

- AWS Directory Service ドキュメント (AWS Web サイト)
- AWS Managed Microsoft AD ディレクトリを作成する (AWS Directory Service ドキュメント)
- Windows EC2 インスタンスをシームレスに結合する (AWS Directory Service ドキュメント)
- AWS 上のMicrosoft SQL サーバー (AWS Web サイト)
- SSMS ドキュメント (Microsoft Web サイト)
- SQL Server でログインを作成する (SQL Server ドキュメント)

インシデント対応とフォレンジックを自動化する

作成者: Lucas Kauffman (AWS) および Tomek Jakubowski (AWS)

概要

このパターンは、 AWS Lambda 関数を使用して以下を提供する一連のプロセスをデプロイします。

- 最小限の知識でインシデント対応プロセスを開始する方法
- AWS セキュリティインシデント対応ガイドに沿った、自動化された反復可能なプロセス
- 自動化ステップを運用、アーティファクトを保存、フォレンジック環境を作成するアカウントの分離

自動インシデントレスポンスとフォレンジックフレームワークは、以下のフェーズからなる標準的なデジタルフォレンジックプロセスに従います。

1. 封じ込め
2. 取得
3. 試験
4. 分析

静的データ (取得したメモリまたはディスクイメージなど) や、ライブではあるが別のシステム上にある動的データを調査できます。

詳細については、追加情報セクションを参照してください。

前提条件と制限

前提条件

- 2 つ AWS アカウント：
  - セキュリティアカウントは、既存のアカウントを使用することもできますが、できれば新しいアカウントが望ましい
  - フォレンジックアカウント (できれば新規)
- AWS Organizations セットアップ
- Organizations メンバーアカウント

  - Amazon Elastic Compute Cloud (Amazon EC2) ロールには、Amazon Simple Storage Service (Amazon S3) への Get and List アクセスと、 によるアクセスが必要です AWS Systems Manager。`AmazonSSMManagedInstanceCore` AWS マネージドロールを使用することをお勧めします。インシデント対応が開始されると、このロールは自動的に Amazon EC2 インスタンスにアタッチされることに注意してください。レスポンスが完了すると、 AWS Identity and Access Management (IAM) はインスタンスに対するすべての権限を削除します。
  - AWS メンバーアカウントとインシデント対応および分析 VPCs の Virtual Private Cloud (VPC) エンドポイント。これらのエンドポイントは S3 ゲートウェイ、EC2 メッセージ、SSM、SSM メッセージです。
- AWS Command Line Interface （ AWS CLI) を Amazon EC2 インスタンスにインストールします。Amazon EC2 インスタンス AWS CLI がインストールされていない場合は、ディスクスナップショットとメモリ取得が機能するためにインターネットアクセスが必要です。この場合、スクリプトはインターネットに連絡して AWS CLI インストールファイルをダウンロードし、インスタンスにインストールします。

制限

- このフレームワークは、法廷に提出できる電子的証拠と見なされるアーティファクトを生成することを意図していません。
- 現在、このパターンは x86 アーキテクチャーで実行中の Linux ベースのインスタンスのみをサポートしています。

アーキテクチャ

ターゲット アーキテクチャ

ターゲット環境は、メンバーアカウントに加えて、セキュリティアカウントとフォレンジックアカウントの 2 つのメインアカウントで構成されています。2 つのアカウントは、次の理由で使用されます。

- フォレンジック分析に失敗した場合に爆発半径を狭めるため、他の顧客アカウントと区別する
- 分析対象のアーティファクトの分離と完全性の保護を確実に行う
- 調査の機密性を保つ
- 脅威アクターが、サービスクォータに達する AWS アカウント ことで侵害されたユーザーがすぐに利用できるすべてのリソースを使用し、Amazon EC2 インスタンスをインスタンス化して調査を実行できない状況を回避するため。

また、セキュリティアカウントとフォレンジックアカウントを分離することで、証拠を取得する応答者とそれを分析する調査員という別のロールを作成できます。各ロールは別のアカウントにアクセスします。

次の図は、アカウント間のインタラクションのみを示しています。各アカウントの詳細は後続の図に示され、図全体が添付されています。

次の図はメンバーアカウントを示しています。

1. イベントは Slack Amazon Simple Notification Service (Amazon SNS) トピックに送信されます。

次の図はセキュリティアカウントを示しています。

2. セキュリティアカウントの Amazon SNS トピックはフォレンジックイベントを開始します。

次の図はフォレンジックアカウントを示しています。

セキュリティアカウントでは、メモリとディスクイメージの取得用に 2 つのメイン AWS Step Functions ワークフローが作成されます。ワークフローの実行後、インシデントに関連する Amazon EC2 インスタンスを持つメンバーアカウントにアクセスし、メモリダンプまたはディスクダンプを収集する一連の Lambda 関数を開始します。これらのアーティファクトはフォレンジックアカウントに保存されます。

フォレンジックアカウントは、Step Functions ワークフローによって収集されたアーティファクトを分析アーティファクト Amazon S3 バケットに保持します。フォレンジックアカウントには、フォレンジックインスタンスの Amazon マシンイメージ (AMI) を構築する Amazon EC2 Image Builder パイプラインもあります。現在、イメージは SANS SIFT ワークステーションに基づいています。

ビルドプロセスは、インターネットに接続するメンテナンス VPC を使用します。イメージは、後で分析 VPC で収集されたアーティファクトの分析のために Amazon EC2 インスタンスをスピンアップするために使用できます。

Analysis VPC はインターネットに接続していません。デフォルトで、このパターンは 3 つのプライベート分析サブネットを作成します。VPC 内のサブネット数のクォータである最大 200 個のサブ

ネットを作成できますが、VPC エンドポイントでコマンドの実行を自動化 AWS Systems Manager Session Manager するには、 にそれらのサブネットを追加する必要があります。

ベストプラクティスの観点から、 AWS CloudTrail と AWS Config を使用して以下を実行することをお勧めします。

- フォレンジックアカウントで行われた変更を追跡する
- 保存および分析されたアーティファクトへのアクセスと整合性を監視する

ワークフロー

次の図は、インスタンスが侵害されてから分析されて封じ込められるまでのプロセスと決定ツリーを含むワークフローの主要な手順を示しています。

1. `SecurityIncidentStatus` タグには Analyze という値が設定されていますか? 設定されている場合は、以下を実行します。
   a. AWS Systems Manager および Amazon S3 用の正しい IAM プロファイルをアタッチします。
   b. Amazon SNS メッセージをスラック の Amazon SNS キューに送信します。
   c. Amazon SNS メッセージを `SecurityIncident` キューに送信します。
   d. メモリとディスクの取得ステートマシンを呼び出します。
2. メモリとディスクは取得されていますか? いいえの場合は、エラーです。
3. Amazon EC2 インスタンスに タグを`Contain`付けます。
4. IAM ロールとセキュリティグループをアタッチして、インスタンスを完全に孤立化します。

自動化とスケール

このパターンの目的は、単一の AWS Organizations 組織内の複数のアカウントでインシデント対応とフォレンジックを実行するためのスケーラブルなソリューションを提供することです。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を介して とやり取りするためのオープンソースツールです。
- AWS Identity and Access Management (IAM) は、リソースの使用を認証および認可されたユーザーを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Key Management Service （ AWS KMS ） は、データを保護するための暗号化キーの作成と制御に役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。また、セキュリティ業界標準とベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- AWS Step Functions は、 AWS Lambda 関数と他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できます。

コード

コードおよび具体的な実装および使用ガイダンスについては、GitHub の自動インシデント対応とフォレンジックフレームワークのリポジトリを参照してください。

エピック

CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
CloudFormation テンプレートをデプロイします。	CloudFormation テンプレートは 1 ~ 7 のマークが付い	AWS 管理者

タスク	説明	必要なスキル
	ており、スクリプト名の最初の語は、テンプレートをデプロイする必要があるアカウントを示しています。CloudFormation テンプレートを起動する順序は重要であることに注意してください。 • `1-forensic-AnalysisVPCnS3Buckets.yaml` ：フォレンジックアカウントにデプロイされます。Amazon S3 バケットと分析 VPC を作成し、CloudTrail をアクティブ化します。 • `2-forensic-MaintenanceVPCnEC2ImageBuilderPipeline.yaml` ：SANS SIFT に基づき、メンテナンス VPC とイメージビルダーのパイプラインをデプロイします。 • `3-security_IR-Disk_Mem_automation.yaml` ：ディスクとメモリの取得を可能にする関数をセキュリティアカウントにデプロイします。 • `4-security_LiME_Volatility_Factory.yaml` ：指定した AMI ID に基づき、メモリモジュールの作成を開始するビルド関	

タスク	説明	必要なスキル
	数を開始します。AMI IDs は異なることに注意してください AWS リージョン。新しいメモリモジュールが必要なときは常に、新しい AMI ID でこのスクリプトを再実行できます。これをゴールデンイメージ AMI ビルダーパイプライン (お使いの環境で使用している場合) と統合することを検討してください。 • `5-member-IR-automation.yaml` ：インシデント対応プロセスを開始するメンバーインシデント対応自動化関数を作成します。アカウント間で Amazon Elastic Block Store (Amazon EBS) ボリュームを共有、インシデント対応プロセス中にスラックチャネルに自動投稿、フォレンジックプロセスを開始、プロセス終了後にインスタンスを孤立化できます。 • `6-forensic-artifact-s3-policies.yaml` ：すべてのスクリプトをデプロイ後、このスクリプトはすべてのクロスアカウント間のインタラクションに必要な権限を修正します。	

タスク	説明	必要なスキル
	• `7-security-IR-vpc.yaml` ：インシデント対応ボリューム処理に使用する VPC を設定します。 特定の Amazon EC2 インスタンスのインシデント対応フレームワークを開始するには、 キー`SecurityIncidentStatus` と 値を持つタグを作成します`Analyze`。これで、メンバーの Lambda 関数が開始され、自動的に孤立化とメモリーならびにディスク取得が開始されます。	
フレームワークを操作します。	また、Lambda 関数は、最後に (または失敗時に) アセットに `Contain` タグを再付与します。これで、封じ込めが開始され、INBOUND/OUTBOUND セキュリティグループがなく、すべてのアクセスを拒否する IAM ロールがあるインスタンスが完全に孤立化されます。 GitHub リポジトリの手順に従います。	AWS 管理者

カスタム Security Hub アクションをデプロイする

タスク	説明	必要なスキル
CloudFormation テンプレートを使用してカスタム Security Hub アクションをデプロイします。	Security Hub のドロップダウンリストを使用できるようにカスタムアクションを作成するには、`Modules/SecurityHub Custom Actions/SecurityHubCustomActions.yaml` CloudFormation テンプレートをデプロイします。次に、アクションを実行する Lambda 関数が `IRAutomation` ロールを引き受けられるように、各メンバーアカウントの `IRAutomation` ロールを変更します。詳細については、GitHub リポジトリを参照してください。	AWS 管理者

関連リソース

- AWS セキュリティインシデント対応ガイド

追加情報

この環境を使用することで、セキュリティオペレーションセンター (SOC) チームは次の方法でセキュリティインシデント対応プロセスを改善できます。

- 分離された環境でフォレンジックを実施できるため、本稼働リソースの偶発的な侵害を回避できる。
- 封じ込めと分析するための、標準化され、反復可能な自動プロセスがあります。
- アカウント所有者または管理者に、タグの使用方法に関する最低限の知識でインシデント対応プロセスを開始できる能力を付与する

- インシデント分析とフォレンジックを大規模環境のノイズなしで実行する、標準化され、クリーンな環境がある
- 複数の分析環境を並行して作成できる能力がある
- クラウドフォレンジック環境の保守や文書化ではなく、インシデント対応に SOC リソースを集中させる
- スケーラビリティを実現するために、手動プロセスから自動プロセスに移行する
- 一貫性を保ち、繰り返し発生するタスクを回避するために、CloudFormation テンプレートを使用する

さらに、永続的なインフラストラクチャの使用を避け、必要なときにリソースに支払います。

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Security Hub 標準検出結果の修正を自動化する

作成者：Chandini Penmetsa (AWS) と Aromal Raj Jayarajan (AWS)

概要

を使用すると AWS Security Hub、次のような標準的なベストプラクティスのチェックを有効にできます。

- AWS 基本的なセキュリティのベストプラクティス
- CIS AWS Foundations Benchmark
- Payment Card Industry Data Security Standard (PCI DSS)

これらの標準にはそれぞれ定義済みの制御があります。Security Hub は、特定の のコントロールをチェック AWS アカウント し、結果を報告します。

AWS Security Hub デフォルトでは、 はすべての検出結果を Amazon EventBridge に送信します。このパターンは、EventBridge ルールをデプロイして AWS Foundational Security Best Practices の標準結果を識別するセキュリティコントロールを提供します。このルールは、Auto Scaling、Virtual Private Cloud (VPCs)、Amazon Elastic Block Store (Amazon EBS)、Amazon Relational Database Service (Amazon RDS) に関する以下の検出結果を AWS Foundational Security Best Practices 標準から特定します。

- [AutoScaling.1] ロードバランサーに関連付けられた Auto Scaling グループは、ロードバランサーのヘルスチェックを使用する必要があります
- [EC2.2] VPC のデフォルトのセキュリティグループでは、インバウンドトラフィックとアウトバウンドトラフィックを許可しないようにする必要があります
- [EC2.6] すべての VPC で VPC フローログ記録を有効にすることをお勧めします
- [EC2.7] EBS のデフォルト暗号化を有効にすることをお勧めします
- [RDS.1] RDS スナップショットはプライベートである必要があります
- [RDS.6] RDS DB インスタンスとクラスターの拡張モニタリングを設定する必要があります
- [RDS.7] RDS クラスターでは、削除保護が有効になっている必要があります

EventBridge ルールは、これらの結果を AWS Lambda 関数に転送し、その結果を修復します。Lambda 関数が、修正情報を含む通知を Amazon Simple Notification Service (Amazon SNS) トピックに送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 修復通知を受信する E メールアドレス
- コントロールをデプロイする AWS リージョン 予定の で Security Hub と AWS Config が有効になっている
- AWS Lambda コードをアップロードするコントロールと同じリージョンにある Amazon Simple Storage Service (Amazon S3) バケット

機能制限

- このセキュリティコントロールでは、セキュリティコントロールの導入後に報告された新しい結果を自動的に修正します。既存の検出結果を修正するには、セキュリティハブコンソールで結果を手動で選択します。次に、「アクション」で、デプロイの一部として作成された AFSBPRemedy カスタムアクションを選択します AWS CloudFormation。
- このセキュリティコントロールはリージョン別であり、モニタリング AWS リージョン する にデプロイする必要があります。
- EC2.6 の解決策として、VPC フローログを有効にするために、Amazon CloudWatch Logs ロググループが `/VpcFlowLogs/vpc_id`形式で作成されます。同じ名前のロググループが存在する場合、既存のロググループが使用されます。
- EC2.7 の解決策として、Amazon EBS のデフォルトの暗号化を有効にするには、default AWS Key Management Service （ AWS KMS) キーが使用されます。この変更により、暗号化をサポートしない特定のインスタンスを使用できなくなります。

アーキテクチャ

ターゲットテクノロジースタック

- Lambda function
- Amazon SNS トピック
- EventBridge ルール
- AWS Identity and Access Management Lambda 関数、VPC フローログ、Amazon RDS 拡張モニタリングの (IAM) ロール

ターゲット アーキテクチャ

自動化とスケール

を使用している場合は AWS Organizations、AWS CloudFormation StackSets を使用して、このテンプレートをモニタリングする複数のアカウントにデプロイできます。

ツール

- AWS CloudFormation は、Infrastructure as Code を使用して AWS リソースをモデル化およびセットアップするのに役立つサービスです。
- Amazon EventBridge は、独自のアプリケーション、Software as a Service (SaaS) アプリケーションからリアルタイムデータのストリームを配信し AWS のサービス、そのデータを Lambda 関数などのターゲットにルーティングします。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートします。
- Amazon Simple Storage Service (Amazon S3) は、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できる、高度にスケーラブルなオブジェクトストレージサービスです。
- Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージの配信または送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

ベストプラクティス

- 9 つの AWS Security Hub ベストプラクティス
- AWS 基本的なセキュリティのベストプラクティス標準

エピック

セキュリティコントロールをデプロイ

タスク	説明	必要なスキル
Amazon S3 バケットを定義します。	Amazon S3 コンソールで、先頭のスラッシュを含まない一意の名前の Amazon S3 バケットを選択または作成します。Amazon S3 バケット名はグローバルに一意であり、名前空間はすべての によって共有されます AWS アカウント。Amazon S3 バケットは、評価対象の Security Hub の検出結果と同じリージョンに存在する必要があります。	クラウドアーキテクト
Lambda コードを Amazon S3 バケットにアップロードします。	「Attachments」セクションで提供されている Lambda コード .zip ファイルを、定義された Amazon S3 バケットにアップロードします。	クラウドアーキテクト
AWS CloudFormation テンプレートをデプロイします。	このパターンの添付ファイルとして提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を指定します。	クラウドアーキテクト

AWS CloudFormation テンプレートのパラメータを完了する

タスク	説明	必要なスキル
Amazon S3 バケット名を指定します。	最初のエピックで作成した Amazon S3 バケットの名前を入力します。	クラウドアーキテクト
Amazon S3 プレフィックスを入力します。	先頭にスラッシュ (など) を付けずに、Amazon S3 バケット内の Lambda コード .zip ファイルの場所を指定します`<directory>/<file-name>.zip` 。	クラウドアーキテクト
Amazon SNS トピックの ARN を指定します。	修復通知に既存の Amazon SNS トピックを使用する場合は、Amazon SNS トピックの Amazon リソースネーム (ARN) を指定します。新しい Amazon SNS トピックを使用する場合は、値を `None` (デフォルト値) のままにします。	クラウドアーキテクト
Eメールアドレスを入力します。	修復通知を受信する E メールアドレスを指定します (Amazon SNS トピック AWS CloudFormation を作成する場合にのみ必要) 。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。 `Warning`	クラウドアーキテクト

タスク	説明	必要なスキル
	潜在的に有害な状況を示します。	
VPC フローログの IAM ロールの ARN を指定します。	VPC フローログに使用する IAM ロールの ARN を指定します。と入力するとNone、 は IAM ロール AWS CloudFormation を作成し、それを使用します。	クラウドアーキテクト
Amazon RDS 拡張モニタリングの IAM ロールの ARN を指定します。	Amazon RDS 拡張モニタリングに使用する IAM ロールの ARN を指定します。と入力するとNone、 は IAM ロール AWS CloudFormation を作成し、それを使用します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
Amazon SNS サブスクリプションを確認します。	テンプレートが正常にデプロイされると、新しい Amazon SNS トピックが作成されると、指定した E メールアドレスにサブスクリプションメッセージが表示されます。是正通知を受け取るには、この購読メールメッセージを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS CloudFormation コンソールでのスタックの作成

- AWS Lambda website
- AWS Security Hub ドキュメント

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon Inspector と を使用してクロスアカウントワークロードのセキュリティスキャンを自動化する AWS Security Hub

作成者：Ramya Pulipaka (AWS) と Mikesh Khanal (AWS)

概要

このパターンでは、Amazon Web Services (AWS) クラウド上のクロスアカウントワークロードの脆弱性を自動的にスキャンする方法について説明します。

このパターンは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのタグでグループ化された Amazon Elastic Compute Cloud (Amazon Inspector EC2) インスタンスのホストベーススキャンのスケジュールを作成するのに役立ちます。 AWS CloudFormation スタックは、必要なすべての AWS リソースとサービスを にデプロイします AWS アカウント。

Amazon Inspector の検出結果は にエクスポート AWS Security Hub され、アカウント、仮想プライベートクラウド (VPCs) AWS リージョン、Amazon EC2 インスタンス全体の脆弱性に関するインサイトを提供します。これらの結果を E メールで受け取ることも、HTTP エンドポイントを使用して Amazon Simple Notification Service (Amazon SNS) トピックを作成して、結果をチケットツール、セキュリティ情報およびイベント管理 (SIEM) ソフトウェア、またはその他のサードパーティのセキュリティソリューションに送信することもできます。

前提条件と制限

前提条件

- 中央監査アカウントを含むクロスアカウントワークロードをホスト AWS アカウント するアクティブ。
- Amazon SNS から E メール通知を受信するための既存の E メールアドレス。
- チケットツール、SIEM ソフトウェア、またはその他のサードパーティのセキュリティソリューションで使用される既存の HTTP エンドポイント。
- Security Hub、有効化および設定済み。このパターンは Security Hub がなくても使用できますが、生成される分析情報を考慮して Security Hub の使用を推薦します。詳細については、Security Hub ドキュメントの「Security Hub のセットアップ」を参照してください。
- Amazon Inspector エージェントは、スキャンするそれぞれの EC2 インスタンスにインストールする必要があります。また、AWS Systems Manager Run Command を使用して EC2 インスタンスに Amazon Inspector Classic エージェントをインストールすることもできます。

スキル

- CloudFormation でのスタックセットの `self-managed`および アクセス`service-managed`許可の使用経験。アクセス`self-managed`許可を使用してスタックインスタンスを特定のリージョンの特定のアカウントにデプロイする場合は、必要な AWS Identity and Access Management (IAM) ロールを作成する必要があります。アクセス`service-managed`許可を使用してスタックインスタンスを特定のリージョン AWS Organizations で によって管理されるアカウントにデプロイする場合は、必要な IAM ロールを作成する必要はありません。詳細については、CloudFormation ドキュメントの「スタックセットの作成」を参照してください。

機能制限

- アカウントの Amazon EC2 インスタンスにタグが適用されていない場合、Amazon Inspector はそのアカウントのすべてのインスタンスをスキャンします。
- CloudFormation スタックセットと `onboard-audit-account.yaml` ファイル (添付) は、同じリージョンにデプロイする必要があります。
- デフォルトでは、Amazon Inspector クラシック に集計結果が適用されません。Security Hub は、複数のアカウントまたは の評価を表示するために推奨されるソリューションです AWS リージョン。
- このパターンのアプローチは、米国東部 (バージニア北部) リージョン () の Amazon SNS トピックの 1 秒あたり 30,000 トランザクション (TPS) の発行クォータでスケールできますが`us-east-1`、制限はリージョンによって異なります。より効果的にスケーリングし、データ損失を回避するには、Amazon SNS トピックの前に Amazon Simple Queue Service (Amazon SQS) を使用することをお勧めします。 Amazon SNS

アーキテクチャ

次の図は、Amazon EC2 インスタンスを自動的にスキャンするワークフローを示しています。

ワークフローの主なステップは、以下のとおりです。

1. Amazon EventBridge ルールは cron 式を使用して特定のスケジュールで自己起動し、Amazon Inspector を起動します。
2. Amazon Inspector は、アカウントのタグ付けされた Amazon EC2 インスタンスをスキャンします。

3. Amazon Inspector は結果をSecurity Hub に送信し、セキュリティハブはワークフロー、優先順位付け、修復に関するインサイトを生成します。
4. また、Amazon Inspector は評価のステータスを監査アカウントの Amazon SNS トピックに送信します。`findings reported` イベントが Amazon SNS トピックに発行されると、 AWS Lambda 関数が呼び出されます。
5. Lambda 関数は、検出結果を取得、フォーマットし、監査アカウントの別の Amazon SNS トピックに送信します。
6. 結果は、Amazon SNS トピックにサブスクライブされている E メールアドレスに送信されます。詳細情報と推奨事項は JSON 形式で、購読している HTTP エンドポイントに送信されます。

ツール

- AWS CloudFormation を使用すると、 AWS リソースをモデル化してセットアップできるため、リソースの管理に費やす時間が減り、アプリケーションに集中する時間が増えます。
- AWS CloudFormation StackSets は、単一のオペレーションで複数のアカウントとリージョンにまたがるスタックを作成、更新、または削除できるようにすることで、スタックの機能を拡張します。
- AWS Control Tower は、 AWS のサービス他のいくつかの機能を組み合わせて統合する抽象化レイヤーまたはオーケストレーションレイヤーを作成します AWS Organizations。
- Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードを実行するのに役立つコンピューティングサービスです。
- AWS Security Hub は、AWS のセキュリティ状態を包括的に把握し、セキュリティ業界標準とベストプラクティスに照らして環境をチェックするのに役立ちます。
- Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバーへのメッセージ配信を提供するマネージドサービスです。

エピック

CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
CloudFormation テンプレートを監査アカウントにデプロイします。	`onboard-audit-account.yaml` ファイル (添付済み) をダウンロードして、コンピュータのローカルパスに保存します。 AWS Management Console 監査アカウントの にサインインし、CloudFormation コンソールを開き、スタックの作成を選択します。 前提条件セクションでテンプレートの準備 を選択してから、テンプレートの準備完了を選択します。テンプレートの準備ができています を選択し、テンプレートの指定 セクションで Amazon S3 URLを選択します。`onboard-audit-account.yaml` ファイルをアップロードし、要件に応じて残りのオプションを設定します。 次の入力パラメータを設定してください。 • `DestinationEmailAddress` — 検出結果を受け取るためのメールアドレスを入力します。	開発者、セキュリティエンジニア

タスク	説明	必要なスキル
	• HTTPEndpoint — チケットや SIEM ツールの HTTP エンドポイントを提供します。 **Note** AWS Command Line Interface () を使用して CloudFormation テンプレートをデプロイすることもできます AWS CLI。詳細については、CloudFormation ドキュメントの「スタックの作成」を参照してください。	
Amazon SNS サブスクリプションを確認します。	E メールの受信トレイを確認し、Amazon SNS からの E メールで サブスクリプションの確認を選択します。これにより、ウェブブラウザウィンドウが開き、サブスクリプションの確認が表示されます。	開発者、セキュリティエンジニア

Amazon Inspector スキャンスケジュールを自動化する CloudFormation スタックセットを作成する

タスク	説明	必要なスキル
Audit アカウントでは、スタックセットを作成します。	`vulnerability-management-program.yaml` ファイル (添付済み) をコン	開発者、セキュリティエンジニア

タスク	説明	必要なスキル
	ピューターのローカルパスにダウンロードします。 CloudFormation コンソールで、スタックセットの表示を選択し、StackSetの作成を選択します。テンプレートは準備完了を選択し、テンプレートファイルをアップロードを選択して、`vulnerability-management-program.yaml` ファイルをアップロードします。 アクセス`self-managed` 許可を使用する場合は、CloudFormation ドキュメントの「セルフマネージドアクセス許可を持つスタックセットを作成する」の手順に従います。これにより、個々のアカウントにスタックセットが作成されます。 アクセス`service-managed` 許可を使用する場合は、CloudFormation ドキュメントの「サービスマネージドアクセス許可を持つスタックセットを作成する」の手順に従います。スタックセットは、組織全体または指定した組織単位 (OU) をターゲットにすることができます。	

タスク	説明	必要なスキル
	スタックセットに次の入力パラメータが設定されていることを確認します。 • `AssessmentSchedule` — cron 式を使用する EventBridge のスケジュール。 • `Duration` – Amazon Inspector 評価の実行の時間 (秒) • `CentralSNSTopicArn` – 中央 Amazon SNS トピックの Amazon リソースネーム (ARN)。 • `Tagkey` — リソースグループに関連付けられているタグキー。 • `Tagvalue` — リソースグループに関連付けられているタグ値。 監査アカウントで Amazon EC2 インスタンスをスキャンする場合は、監査アカウントで CloudFormation スタックとして `vulnerability-management-program.yaml` ファイルを実行する必要があります。	

タスク	説明	必要なスキル
ソリューションを更新する	Amazon Inspector に指定したスケジュールで E メールまたは HTTP エンドポイントで検出結果を受け取っていることを確認します。	開発者、セキュリティエンジニア

関連リソース

- Amazon Inspector でセキュリティ脆弱性テストをスケールする (AWS ブログ記事)
- Amazon Inspector のセキュリティ検出結果を自動的に修正する (AWS ブログ記事)
- Amazon EC2、 AWS Systems Managerおよび Amazon Inspector を使用してセキュリティ評価の設定を簡素化する方法 (AWS ブログ記事)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

パブリック IP アドレスからのアクセスを許可する AWS セキュリティグループを自動的に監査する

作成者: Eugene Shifer (AWS) と Stephen DiCato (AWS)

概要

セキュリティのベストプラクティスとして、絶対に必要なもののみに AWS リソースが公開されるのを最小限に抑えることが重要です。例えば、一般ユーザー向けのウェブサーバーでは、インターネットからのインバウンドアクセスを許可する必要がありますが、不要な露出を減らすために、他のワークロードへのアクセスを特定のネットワークに制限する必要があります。Amazon Virtual Private Cloud (Amazon VPC) のセキュリティグループは、リソースアクセスを制限するのに役立つ効果的なコントロールです。ただし、特にマルチアカウントアーキテクチャでは、セキュリティグループの評価は面倒な作業になる可能性があります。 のAWS Config ルールとAWS Security Hub コントロールは、パブリックインターネット (0.0.0.0/0) から Secure Shell (SSH)、HTTP、HTTPS、Windows リモートデスクトッププロトコル (RDP) などの特定のネットワーク通信プロトコルへのアクセスを許可するセキュリティグループを特定するのに役立ちます。ただし、これらのルールとコントロールは、サービスが非標準ポートで実行されている場合や、アクセスが特定のパブリック IP アドレスに制限されている場合には適用されません。例えば、これは、ウェブサービスが標準 TCP ポート 443 ではなく TCP ポート 8443 に関連付けられている場合に発生する可能性があります。これは、開発者がテスト目的など、ホームネットワークからサーバーにアクセスできる場合にも発生する可能性があります。

これに対処するには、このパターンで提供されている infrastructure as code (IaC) ソリューションを使用して、非プライベート (RFC 1918 非準拠) IP アドレスからユーザー AWS アカウント または AWS 組織内の任意のワークロードへのアクセスを許可するセキュリティグループを特定できます。AWS CloudFormation テンプレートは、カスタム AWS Config ルール、 AWS Lambda関数、および必要なアクセス許可をプロビジョニングします。1 つのアカウントにスタックとしてデプロイすることも、組織全体で管理されるスタックセットとしてデプロイすることもできます AWS Organizations。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- GitHub の使用経験

- 単一の にデプロイする場合 AWS アカウント：
  - CloudFormation スタックを作成するアクセス許可
  - ターゲットアカウントでAWS Config をセットアップする
  - （オプション) ターゲットアカウントで Security Hub をセットアップする
- AWS 組織にデプロイする場合：
  - CloudFormation スタックセットを作成するアクセス許可
  - AWS Organizations 統合で Security Hub をセットアップする
  - このソリューションをデプロイするアカウントでAWS Config をセットアップする
  - を AWS Config と Security Hub の委任管理者 AWS アカウント に指定する

制約事項

- Security Hub が有効になっていない個々のアカウントにデプロイする場合は、 AWS Config を使用して検出結果を評価できます。
- AWS Config と Security Hub の委任管理者を持たない組織にデプロイする場合は、個々のメンバーアカウントにログインして結果を表示する必要があります。
- AWS Control Tower を使用して組織内のアカウントを管理および管理する場合は、 AWS Control Tower (CfCT) のカスタマイズを使用して、このパターンで IaC をデプロイします。CloudFormation AWS Control Tower コンソールを使用すると、ガードレールから設定ドリフトが作成され、組織単位 (OUs) またはマネージドアカウントを再登録する必要があります。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、AWS のサービス「リージョン別」を参照してください。特定のエンドポイントについては、「サービスエンドポイントとクォータ」ページを参照し、サービスのリンクを選択します。

アーキテクチャ

個々の へのデプロイ AWS アカウント

次のアーキテクチャ図は、単一の 内の AWS リソースのデプロイを示しています AWS アカウント。CloudFormation コンソールから直接 CloudFormation テンプレートを使用してリソースをプロビジョニングします。Security Hub が有効になっている場合は、 AWS Config または Security Hub で結果を表示できます。Security Hub が有効になっていない場合は、 の結果のみを表示できます AWS Config。

この図表は、次のワークフローを示しています：

1. CloudFormation スタックを作成します。これにより、Lambda 関数と AWS Config ルールがデプロイされます。ルールと 関数の両方に、 AWS Config および ログでリソース評価を発行するために必要な AWS Identity and Access Management (IAM) アクセス許可が設定されます。
2. この AWS Config ルールは検出評価モードで動作し、24 時間ごとに Lambda 関数を呼び出します。
3. Lambda 関数はセキュリティグループを評価し、更新を送信します AWS Config。
4. Security Hub はすべての AWS Config 検出結果を受け取ります。
5. アカウントで設定したサービスに応じて AWS Config、Security Hub または で結果を表示できます。

AWS 組織にデプロイする

次の図は、 AWS Organizations と で管理されている複数のアカウントへのパターンのデプロイを示しています AWS Control Tower。CloudFormation テンプレートは CfCT を使用してデプロイします。評価結果は、委任管理者アカウントの Security Hub に一元化されます。図の AWS CodePipeline ワークフローセクションは、CfCT デプロイ中に発生する背景ステップを示しています。

この図表は、次のワークフローを示しています：

1. 管理アカウントで、IaC テンプレートの圧縮 (ZIP) ファイルを、CfCT によってデプロイされた Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。
2. CfCT パイプラインはファイルを解凍し、cfn-nag (GitHub) チェックを実行し、テンプレートを CloudFormation スタックセットとしてデプロイします。
3. CfCT マニフェストファイルで指定した設定に応じて、CloudFormation StackSets はスタックを個々のアカウントまたは指定された OUs。これにより、Lambda 関数と AWS Config ルールがターゲットアカウントにデプロイされます。ルールと 関数はどちらも、 AWS Config および ログでリソース評価を発行するために必要な IAM アクセス許可で設定されます。
4. この AWS Config ルールは検出評価モードで動作し、24 時間ごとに Lambda 関数を呼び出します。
5. Lambda 関数はセキュリティグループを評価し、更新を送信します AWS Config。

6. AWS Config は、すべての結果を Security Hub に転送します。
7. Security Hub の検出結果は、委任管理者アカウントに集約されます。
8. 集約された検出結果は、委任管理者アカウントの Security Hub で表示できます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体にわたる管理に役立ちます AWS リージョン。
- AWS Config は、 のリソース AWS アカウント の詳細と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。An AWS Config rule は、リソースの理想的な構成設定を定義し、 AWS リソースがルールの条件に準拠しているかどうかを評価 AWS Config できます。
- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境をセットアップして管理するのに役立ちます。AWS Control Tower (CfCT) のカスタマイズは、 AWS Control Tower ランディングゾーンをカスタマイズし、 AWS ベストプラクティスとの整合性を保つのに役立ちます。このソリューションのカスタマイズは、CloudFormation テンプレートと AWS Organizations サービスコントロールポリシー (SCPs。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- AWS Security Hub は、 のセキュリティ状態を包括的に表示します AWS。また、セキュリティ業界標準やベストプラクティスに照らして AWS 環境をチェックするのに役立ちます。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、GitHub Detect の脆弱なセキュリティグループリポジトリで入手できます。

ベストプラクティス

以下のリソースのベストプラクティスに従うことをお勧めします。

- を使用した組織単位のベストプラクティス AWS Organizations (AWS クラウドオペレーションと移行ブログ)
- AWS Control Tower で を使用して初期基盤を確立するためのガイダンス AWS (AWS ソリューションライブラリ)
- AWS Control Tower リソースの作成と変更に関するガイダンス (AWS Control Tower ドキュメント)
- CfCT デプロイに関する考慮事項 (AWS Control Tower ドキュメント)
- 最小特権のアクセス許可を適用する (IAM ドキュメント)

エピック

CloudFormation テンプレートを確認する

タスク	説明	必要なスキル
デプロイ戦略を決定します。	ソリューションとコードを確認して、 AWS 環境のデプロイ戦略を決定します。1 つのアカウントまたは AWS 組織にデプロイするかどうかを決定します。	アプリ所有者、AWS 全般
リポジトリをクローン作成します。	次のコマンドを入力して、脆弱なセキュリティグループを検出リポジトリのクローンを作成します。 `git clone https://github.com/aws-samples/detect-public-security-groups.git`	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
Python バージョンを検証します。	1. クローンされたリポジトリの最上位ディレクトリに移動します。 `cd detect-public-security-groups` 2. Security-Group-Public-Assessment.yaml を開きます。 3. `SgPublicAccessCheckLambdaFunction` リソースで、Python バージョンがターゲットと互換性があることを確認します AWS リージョン。デフォルトでは、この関数は Python 3.12 を使用します。詳細については、「AWS Lambda Python 3.12 のサポートが追加されました」を参照してください。必要に応じて、Python バージョンを更新します。 4. Security-Group-Public-Assessment.yaml を保存して閉じます。	AWS 管理者、アプリ開発者

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	CloudFormation テンプレートを AWS 環境にデプロイし	アプリ開発者、AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	ます。次のいずれかを行います： • 単一の にデプロイする場合は AWS アカウント、「スタックの作成」の手順に従います。 • によって管理されていない組織にデプロイする場合は AWS Control Tower、「スタックセットの作成」の手順に従います。 • が管理する組織にデプロイする場合は AWS Control Tower、「独自のカスタマイズを構築する」の手順を参照してください。	
デプロイメントを確認する	CloudFormation コンソールで、スタックまたはスタックセットが正常にデプロイされていることを確認します。	AWS 管理者、アプリ所有者

検出結果を確認する

タスク	説明	必要なスキル
AWS Config ルールの検出結果を表示します。	Security Hub で、次の操作を実行して個々の検出結果のリストを表示します。 1. Security Hub コンソールを開きます。 2. ナビゲーションペインで 調査結果を選択します。	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル
	3. フィルターの追加 ボックスに、次のフィルターを追加します。 • コンプライアンスステータスは です FAILED • タイトルは です SgPublicAccessCheck 4. [Apply] を選択します。 Security Hub で、以下を実行して、グループ化された検出結果の合計のリストを表示します AWS アカウント。 1. Security Hub コンソールを開きます。 2. ナビゲーションペインで、[Insights] (インサイト) を選択します。 3. [Create insight] (インサイトの作成) を選択します。 4. インサイトのグループ化属性を選択するには: a. 検索ボックスを選択して、フィルターオプションを表示します。 b. [Group by] (グループ化の条件) を選択します。 c. AwsAccountId を選択します。 d. [Apply] を選択します。	

タスク	説明	必要なスキル
	5. フィルターの追加 ボックスに、次のフィルターを追加します。 • タイトルは です `SgPublicAccessCheck` • コンプライアンスステータスは です `FAILED` 6. [Create insight] (インサイトの作成) を選択します。 7. [インサイトの名前] を入力し、[インサイトの作成] を選択します。 で AWS Config検出結果のリストを表示するには、 AWS Config ドキュメントの「コンプライアンス情報と評価結果の表示」の手順に従います。	

トラブルシューティング

問題	ソリューション
CloudFormation スタックセットの作成または削除が失敗します。	AWS Control Tower がデプロイされると、必要なガードレールが適用され、 AWS Config アグリゲータとルールの制御が引き受けられます。これには、CloudFormation による直接的な変更の防止が含まれます。関連するすべてのリソースを含め、この CloudFormation テンプレートを適切にデプロイまたは削除するには、CfCT を使用する必要があります。

問題	ソリューション
CfCT は CloudFormation テンプレートの削除に失敗します。	マニフェストファイルで必要な変更を行い、テンプレートファイルを削除した後も CloudFormation テンプレートが存続する場合は、マニフェストファイルに `enable_stack_set_deletion` パラメータが含まれ、値が に設定されていることを確認しますfalse。詳細については、CfCT ドキュメントの「スタックセットの削除」を参照してください。

関連リソース

- AWS Config カスタムルール (AWS Config ドキュメント)

AWS Config のカスタム修復ルールを使用して AWS CloudTrail を自動的に再有効化する

作成者: Manigandan Shri (AWS)

概要

Amazon Web Services (AWS) アカウントのアクティビティを可視化することは、セキュリティと運用の重要なベストプラクティスです。AWS CloudTrail は、アカウントのガバナンス、コンプライアンス、運用およびリスクの監査に役立ちます。

CloudTrail がアカウントで引き続き有効になっていることを確認するために、AWS Config には `cloudtrail-enabled` マネージドルールが用意されています。CloudTrail がオフになっている場合、`cloudtrail-enabled` ルールは自動修復を使用して自動的に再度有効にします。

ただし、自動修復を使用する場合は、CloudTrail のセキュリティのベストプラクティスに従っていることを確認する必要があります。これらのベストプラクティスには、すべての AWS リージョンでの CloudTrail の有効化、読み取り/書き込みワークロードのログの作成、および AWS Key Management Service (AWS KMS) マネージドキー (SSE-KMS) を使用したサーバー側の暗号化によるログファイルの暗号化が含まれます。

このパターンは、アカウントの CloudTrail を自動的に再び有効にするカスタム修復アクションを提供することで、これらのセキュリティのベストプラクティスに従うのに役立ちます。

> **Important**
>
> CloudTrail の改ざんを防ぐために、サービスコントロールポリシー (SCPs) を使用することをお勧めします。詳細については、AWS セキュリティブログの「How to use AWS Organizations to simplify security at enormous scale」の「Prevent tampering with AWS CloudTrail」セクションを参照してください。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Systems Manager Automation ランブックを作成するための権限
- アカウントの既存の証跡

制限

このパターンでは、以下のアクションはサポートされていません。

- ストレージロケーションの Amazon Simple Storage Service (Amazon S3) プレフィックスキーの設定
- Amazon Simple Notification Service (Amazon SNS)トピックへの公開
- CloudTrail ログをモニタリングするように Amazon CloudWatch Logs を設定する

アーキテクチャ

テクノロジースタック

- AWS Config
- CloudTrail
- Systems Manager
- Systems Manager Automation

ツール

- AWS Config は、アカウントの AWS のリソースの設定を詳細に表示します。
- AWS CloudTrail は、アカウントのガバナンス、コンプライアンス、運用およびリスクの監査を可能にするのに役立ちます。
- AWS Key Management Service (AWS KMS) は、暗号化およびキー管理サービスです。
- AWS Systems Manager は、AWS 上のインフラストラクチャを表示および制御するのに役立ちます。
- AWS Systems Manager Automation は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスおよびその他の AWS リソースの一般的なメンテナンスおよびデプロイメントタスクを簡素化します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。

コード

cloudtrail-remediation-action.yml ファイル (添付) は、セキュリティのベストプラクティスを使用して CloudTrail をセットアップして再度有効にするためのSystems Manager 自動化ランブックを作成するのに役立ちます。

エピック

CloudTrail の設定

タスク	説明	必要なスキル
S3 バケットを作成する。	AWS マネジメントコンソールにサインインし、Amazon S3 コンソールを開き、CloudTrail ログを保存する S3 バケットを作成します。詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	システム管理者
CloudTrail がログファイルを S3 バケットに配信できるようにするバケットポリシーを追加します。	CloudTrail には、ログファイルを S3 バケットに配信するために必要な権限が必要です。Amazon S3 コンソールで、前に作成した S3 バケットを選択してから、[アクセス権限] を選択します。CloudTrail ドキュメントにある CloudTrail 用の Amazon S3 バケットポリシーを使用して、S3 バケットポリシーを作成します。 S3 バケットにポリシーを追加する手順については、Amazon S3 ドキュメントの「Amazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。	システム管理者

タスク	説明	必要なスキル
	Important CloudTrail で証跡を作成したときにプレフィックスを指定した場合は、必ず S3 バケットポリシーに含めてください。プレフィックスは、S3 バケットにフォルダのような組織を作成する S3 オブジェクトキーへのオプションの追加です。詳細については、CloudTrail ドキュメントの「証跡の作成」を参照してください。	
KMS キーを作成します。	CloudTrail 用の AWS KMS キーを作成してオブジェクトを暗号化してから S3 バケットに追加します。この件の詳細は、CloudTrail ドキュメントで「AWS KMS キーによる CloudTrail ログファイルの暗号化 (SSE-KMS)」を参照してください。	システム管理者

タスク	説明	必要なスキル
キーポリシーを KMS キーに追加します。	KMS キーポリシーをアタッチして、CloudTrail が KMS キーを使用できるようにします。この件の詳細は、CloudTrail ドキュメントで「AWS KMS キーによる CloudTrail ログファイルの暗号化 (SSE-KMS)」を参照してください。 **Important** CloudTrail には`Decrypt`アクセス許可は必要ありません。	システム管理者
Systems Manager 向けの AssumeRole ランブックの作成	Systems Manager オートメーション用の `AssumeRole` を作成して、ランブックを実行します。手順と詳細については、Systems Manager ドキュメントの「オートメーションの設定」を参照してください。	システム管理者

Systems Manager Automation ランブックの作成とテスト

タスク	説明	必要なスキル
Systems Manager Automation ランブックの作成	`cloudtrail-remediation-action.yml` ファイル (添付) を使用して Systems Manager Automation ランブックを作成します。この件の詳細は、「SSM ドキュメントコ	システム管理者

タスク	説明	必要なスキル
	ンテンツを作成する」を参照してください。	
ランブックをテストしてください。	Systems Manager コンソールで、前に作成した Systems Manager 自動化ランブックをテストします。この件の詳細は、Systems Manager のドキュメントで「シンプルなオートメーションを実行する」を参照してください。	システム管理者

AWS Config で自動修復ルールを設定する

タスク	説明	必要なスキル
CloudTrail を有効にしたルールを追加します。	AWS Config コンソールで [ルール] を選択し、[ルールを追加] を選択します。[Add rule(ルールの追加)] ページで、[Add custom rule (カスタムルールの追加)] を選択します。[ルールの設定] ページで、名前と説明を入力し、`cloudtrail-enabled` ルールを追加します。詳細については、AWS Config ドキュメントで「AWS Config ルールの追加、更新、削除」を参照してください。	システム管理者
自動修復アクションを追加します。	[アクション] ドロップダウンリストから、[修復の管理] を選択します。[自動修復] を選択し、先ほど作成した	システム管理者

タスク	説明	必要なスキル
	Systems Manager ランブックを選択します。 CloudTrail に必要な入力パラメータは次のとおりです。 • `CloudTrailName` • `CloudTrailS3BucketName` • `CloudTrailKmsKeyId` • `AssumeRole` (オプション) 次の入力パラメータはデフォルトで true に設定されています。 • `IsMultiRegionTrail` • `IsOrganizationTrail` • `IncludeGlobalServiceEvents` • `EnableLogFileValidation` [Rate Limits パラメータ] と [Resource ID パラメータ] はデフォルト値のままにします。[保存] を選択します。 詳細については、AWS Config ドキュメントで「AWS Config ルールによる非準拠リソースの修復」を参照してください。	

タスク	説明	必要なスキル
自動修復ルールをテストします。	自動修復ルールをテストするには、CloudTrail コンソールを開いて [証跡] を選択し、次に証跡を選択します。証跡のログ記録をオフにするには、[ログ記録を停止] を選択します。確認を求められたら、[Stop logging] を選択します。CloudTrail はその証跡のログ記録アクティビティを停止します。 AWS Config ドキュメントの「AWS Config ルールでのリソースの評価」の指示に従って、CloudTrail が自動的に再び有効になったことを確認します。	システム管理者

関連リソース

CloudTrail の設定

- S3 バケットを作成する
- CloudTrail の Amazon S3 バケットポリシー
- Amazon S3 コンソールを使用したバケットポリシーの追加
- 証跡の作成
- オートメーションの設定
- AWS KMS 管理キー (SSE-KMS) を使用した CloudTrail ログファイルの暗号化

Systems Manager Automation ランブックの作成とテスト

- システムマネージャドキュメントの作成

- シンプルなオートメーションワークフローを実行する

AWS Config で自動修復ルールを設定する

- AWS Config ルールの追加、更新、削除
- AWS Config ルールによる非準拠リソースの修復

追加リソース

- AWS CloudTrail でのセキュリティのベストプラクティス
- AWS Systems Manager の使用開始
- AWS Config の使用開始
- AWS CloudTrail の使用開始

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

暗号化されていない Amazon RDS DB インスタンスとクラスターを自動的に修正する

作成者: Ajay Rawat (AWS)、Josh Joy (AWS)

概要

このパターンでは、、 AWS Systems Manager ランブック、および () AWS Key Management Service AWS KMSキーを使用して AWS Config、Amazon Web Services () の暗号化されていない Amazon Relational Database Service (Amazon RDS AWS) DB インスタンスとクラスターを自動的に修復する方法について説明します。

暗号化された RDS DB インスタンスは、基になるストレージへの不正アクセスからデータを保護することによって、データ保護の追加レイヤーを提供します。Amazon RDS 暗号化を使用して、 にデプロイされたアプリケーションのデータ保護を強化し AWS クラウド、保管時の暗号化のコンプライアンス要件を満たすことができます。RDS DB インスタンスの暗号化は作成時に有効にできますが、作成後には有効にできません。ただし、DB インスタンスのスナップショットを作成し、そのスナップショットの暗号化済みコピーを作成して、暗号化されていない RDS DB インスタンスに暗号化を追加できます。その後、暗号化されたスナップショットから DB インスタンスを復元して、元の DB インスタンスの暗号化されたコピーを取得できます。

このパターンでは AWS Config ルール 、 を使用して RDS DB インスタンスとクラスターを評価します。ランブックを使用して修復を適用します。 AWS Systems Manager ランブックは、非準拠の Amazon RDS リソースに対して実行するアクションと、DB スナップショットを暗号化するための AWS KMS キーを定義します。次に、サービスコントロールポリシー (SCP) を適用して、暗号化なしで新しい DB インスタンスやクラスターが作成されないようにします。

このパターンのコードは GitHub で提供されています。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- このパターンの GitHub ソースコードリポジトリからコンピュータにダウンロードされたファイル
- 暗号化されていない RDS DB インスタンスまたはクラスター
- RDS DB インスタンスとクラスターを暗号化するための既存の AWS KMS キー

- KMS キーリソースポリシーを更新するためのアクセス
- AWS Config で有効 AWS アカウント （ AWS ドキュメントの「 の開始方法 AWS Config」を参照 ）

制約事項

- RDS DB インスタンスの暗号化は、作成後ではなく、作成時にのみ有効にできます。
- 暗号化されていない DB インスタンスのリードレプリカを暗号化することや、暗号化されている DB インスタンスのリードレプリカを暗号化しないようにすることはできません。
- 暗号化されていないバックアップやスナップショットを、暗号化された DB インスタンスに復元することはできません。
- Amazon RDS 暗号化は、ほとんどの DB インスタンスクラスで使用できます。例外のリストについては、Amazon RDS ドキュメントの「Amazon RDS リソースの暗号化」を参照してください。
- 暗号化されたスナップショットを AWS リージョン 間でコピーするには、送信先で KMS キーを指定する必要があります AWS リージョン。これは、KMS キーが作成された AWS リージョン に固有のためです。
- ソーススナップショットはコピープロセス全体で暗号化されたままになります。Amazon RDS は、コピー処理中にエンベロープ暗号化を使用してデータを保護します。詳細については、 AWS KMS ドキュメントの「エンベロープ暗号化」を参照してください。
- 暗号化された DB インスタンスの暗号化を解除することはできません。ただし、暗号化された DB インスタンスからデータをエクスポートし、暗号化されていない DB インスタンスにデータをインポートすることはできます。
- KMS キーの削除は、そのキーをもう使用しないことが確実である場合にのみ行ってください。不明な場合は、削除するのではなく、KMS キーを無効化することを検討します。無効化した KMS キーは、後で使用する必要が生じた場合に再度有効化できますが、削除した KMS キーは復元できません。
- 自動バックアップを保持しない場合、DB インスタンス AWS リージョン と同じ にある自動バックアップは削除されます。DB インスタンスを削除した後は、復元できません。
- 自動バックアップは、DB インスタンスの削除時に設定した保持期間だけ保持されます。この設定された保持期間は、最終的な DB スナップショットを作成するかどうかにかかわらず発生します。
- 自動修復が有効になっている場合、このソリューションは同じ KMS キーを持つすべてのデータベースを暗号化します。

アーキテクチャ

次の図は、 AWS CloudFormation 実装のアーキテクチャを示しています。このパターンは、 を使用して実装することもできます AWS Cloud Development Kit (AWS CDK)。

ツール

ツール

- AWS CloudFormation は、 AWS リソースを自動的にセットアップするのに役立ちます。テンプレートファイルを使用して、リソースのコレクションを 1 つのユニット (スタック) として作成および削除できます。
- AWS Cloud Development Kit (AWS CDK) は、コードでクラウドインフラストラクチャを定義し、使い慣れたプログラミング言語を使用してプロビジョニングするためのソフトウェア開発フレームワークです。

AWS サービスと機能

- AWS Config は、 AWS リソースの設定と他のリソースとの関係を追跡します。また、これらの AWS リソースのコンプライアンスを評価することもできます。このサービスは、必要な設定と照らし合わせて AWS リソースを評価するように設定できるルールを使用します。一般的なコンプライアンスシナリオには一連の AWS Config マネージドルールを使用することも、カスタムシナリオには独自のルールを作成することもできます。 AWS リソースが非準拠であることがわかったら、 AWS Systems Manager ランブックを介して修復アクションを指定し、オプションで Amazon Simple Notification Service (Amazon SNS) トピックを介してアラートを送信できます。つまり、修復アクションを に関連付け AWS Config ルール 、手動で介入することなく、非準拠のリソースに対応するために自動的に実行することを選択できます。自動修復後もリソースがまだ準拠していない場合は、自動修復を再試行するようにルールを設定できます。
- Amazon Relational Database Service (Amazon RDS) を使用して、クラウドでリレーショナルデータベースをセットアップ、運用、スケーリングできます。Amazon RDS の基本的な構成要素は、内の分離されたデータベース環境である DB インスタンスです AWS クラウド。Amazon RDS は、さまざまなリレーショナルデータベースのユースケースに合わせて最適化されたインスタンスタイプを選択できます。インスタンスタイプは、CPU、メモリ、ストレージ、およびネットワーク容量のさまざまな組み合わせで構成され、データベースに適したリソースの組み合わせを柔軟に選択できます。各インスタンスタイプには 1 つ以上のインスタンスサイズが含まれているため、ターゲットワークロードの要件に合わせてデータベースをスケーリングできます。

- AWS Key Management Service （ AWS KMS ） は、データを暗号化する作成と制御を容易に AWS KMS keysするマネージドサービスです。KMS キーは、ルートキーの論理表現です。KMS キーには、キー ID、作成日、説明、キーステータスなどのメタデータが含まれます。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理するのに役立ちます。
- サービスコントロールポリシー (SCP) では、組織のすべてのアカウントで使用可能な最大アクセス許可を一元的に制御できます。SCP は、アカウントが組織のアクセスコントロールガイドラインに従っていることを確認するのに役立ちます。SCP は、管理アカウントのユーザーやロールには影響を与えません。SCP は、組織内のメンバーアカウントにのみ影響を与えます。SCP を組織のルートにアタッチする前に、そのポリシーがアカウントに与える影響を徹底的にテストすることを強くお勧めします。代わりに、お客様のアカウントを一度に 1 つずつ、または少なくとも少人数ずつ移動できる組織単位 (OU) を作成し、誤って主要なサービスからユーザーを締め出すことのないようにします。

コード

このパターンのソースコードとテンプレートは、GitHub リポジトリにあります。このパターンには 2 つの実装オプションがあります。 AWS CloudFormation テンプレートをデプロイして、RDS DB インスタンスとクラスターを暗号化する修復ロールを作成するか、 を使用できます AWS CDK。リポジトリには、これら 2 つのオプション用に別々のフォルダがあります。

「エピック」セクションでは、CloudFormation テンプレートをデプロイするためのステップバイステップの手順について説明します。を使用する場合は AWS CDK、GitHub リポジトリの `README.md` ファイルの指示に従います。

ベストプラクティス

- 保管時と転送中のいずれもデータ暗号化を有効にします。
- すべてのアカウント AWS Config で と を有効にします AWS リージョン。
- すべてのリソースタイプの設定変更を記録します。
- IAM 認証情報のローテーションを定期的に行います。
- のタグ付けを活用することで AWS Config、 リソースの管理、検索、フィルタリングが容易になります。

エピック

IAM 修復ロールと Systems Manager ランブックを作成する

タスク	説明	必要なスキル
CloudFormation テンプレートをダウンロードします。	GitHub リポジトリから `unencrypted-to-encrypted-rds.template.json` ファイルをダウンロードします。	DevOps エンジニア
CloudFormation スタックを更新する。	1. にサインイン AWS Management Console し、CloudFormation コンソールを開きます。 2. `unencrypted-to-encrypted-rds.template.json` テンプレートを起動して、新しいスタックを作成します。 テンプレートのデプロイの詳細については、AWS CloudFormation ドキュメントを参照してください。	DevOps エンジニア
CloudFormation パラメータと値を確認してください。	1. スタックの詳細を確認し、環境要件に基づいて値を更新します。 2. [スタックの作成] を選択してテンプレートをデプロイします。	DevOps エンジニア
リソースを確認します。	スタックが作成されると、そのステータスは [CREATE_COMPLETE] に変わります。Cl	DevOps エンジニア

タスク	説明	必要なスキル
	oudFormation コンソールで、作成されたリソース (IAM ロール、Systems Manager ランブック) を確認します。	

AWS KMS キーポリシーを更新する

タスク	説明	必要なスキル
KMS キーポリシーを更新します。	1. キーエイリアスが `alias/RDSEncryptionAtRestKMSAlias` 存在することを確認します。 2. キーポリシーステートメントには IAM 修復ロールを含める必要があります。(前のエピックでデプロイした CloudFormation テンプレートによって作成されたリソースを確認してください。) 3. 次のキーポリシーで、太字の部分を、アカウントと作成した IAM ロールと一致するように更新してください。 `{ "Sid": "Allow access through RDS for all principals in the account that are authorized to use RDS",`	DevOps エンジニア

タスク	説明	必要なスキル

	<pre>"Effect": "Allow",
	"Principal": {
	"AWS": "arn:aws:iam:: <your-AWS-account-ID>”:role/<your-IAM-remediation-role>"
	},
	"Action": [
	"kms:Encrypt",
	"kms:Decrypt",
	"kms:ReEncrypt*",
	"kms:GenerateDataKey*",
	"kms:CreateGrant",
	"kms:ListGrants",
	"kms:DescribeKey"
	],
	"Resource": "*",
	"Condition": {
	"StringEquals": {
	"kms:ViaService": "rds.us-east-1.amazonaws.com",
	"kms:CallerAccount": "<your-AWS-account-ID>"
	}
	}
	}</pre>	

準拠していないリソースを見つけて修正する

タスク	説明	必要なスキル
非準拠リソースを表示します。	1. 非準拠リソースのリストを表示するには、 AWS Config コンソールを開きます。 2. ナビゲーションペインで、[ルール] を選択し、`rds-storage-encrypted` ルールを選択します。 AWS Config コンソールに表示される非準拠のリソースは、クラスターではなくインスタンスになります。修復自動化はインスタンスとクラスターを暗号化し、新しく暗号化されたインスタンスまたは新しく作成されたクラスターを作成します。ただし、同じクラスターに属する複数のインスタンスを同時に修正しないように注意してください。 RDS DB インスタンスまたはボリュームを修正する前に、その RDS DB インスタンスが使用されていないことを確認してください。スナップショットに元のデータが含まれていることを確認するために、スナップショットの作成中に書き込み操作が行われていないことを確認します。修	DevOps エンジニア

タスク	説明	必要なスキル
	正が実行されるメンテナンスの時間帯を設けることを検討してください。	
非準拠のリソースを修復します。	1. 準備が整い、メンテナンスウィンドウが有効になったら、修正するリソースを選択し、[修復] を選択します。 これで [アクションステータス] 列に [アクション実行がキューに入れられました] と表示されるはずです。 2. Systems Manager で修復の進行状況とステータスを表示します。Systems Manager コンソールを開きます。ナビゲーションペインで [オートメーション] を選択し、対応するオートメーションの実行 ID を選択すると、詳細が表示されます。	DevOps エンジニア

タスク	説明	必要なスキル
RDS DB インスタンスが使用可能であることを確認します。	自動化が完了すると、新しく暗号化された RDS DB インスタンスが使用可能になります。暗号化された RDS DB インスタンスには、プレフィックスの後に元の名前が `encrypted` 続きます。例えば、暗号化されていない RDS DB インスタンス名が `database-1` の場合、新しく暗号化された RDS DB インスタンスは `encrypted-database-1` になります。	DevOps エンジニア
暗号化されていないインスタンスを終了します。	修正が完了し、新しく暗号化されたリソースが検証されたら、暗号化されていないインスタンスを終了できます。リソースを終了する前に、新しく暗号化されたリソースが暗号化されていないリソースと一致することを確認してください。	DevOps エンジニア

SCP を強制する

タスク	説明	必要なスキル
SCP を強制します。	SCP を適用して、future DB インスタンスやクラスターが暗号化なしで作成されないようにします。この目的のために GitHub リポジトリで提供されている `rds_encry`	セキュリティエンジニア

タスク	説明	必要なスキル
	pted.json ファイルを使用し、AWS ドキュメントの指示に従います。	

関連リソース

リファレンス

- セットアップ AWS Config
- AWS Config カスタムルール
- AWS KMS の概念
- AWS Systems Manager ドキュメント
- サービスコントロールポリシー

ツール

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)

ガイドとパターン

- でカスタム修復ルール AWS CloudTrail を使用して自動的に再有効化する AWS Config

追加情報

の AWS Config 仕組み

を使用すると AWS Config、まず アカウントに存在するサポートされている AWS リソースを検出し、各リソースの設定項目を生成します。 AWS Config また、 はリソースの設定が変更されると設定項目を生成し、設定レコーダーを起動したときからリソースの設定項目の履歴レコードを維持します。デフォルトでは、 は でサポートされているすべてのリソースの設定項目 AWS Config を作成します AWS リージョン。サポートされているすべてのリソースの設定項目 AWS Config を作成しない場合は、追跡するリソースタイプを指定できます。

AWS Config と の AWS Config ルール 関係 AWS Security Hub

AWS Security Hub は、セキュリティおよびコンプライアンス体制の管理をサービスとして提供するセキュリティおよびコンプライアンスサービスです。 AWS Config と を主要なメカニズム AWS Config ルール として使用して、 AWS リソースの設定を評価します。 は、リソース設定を直接評価するために AWS Config ルール も使用できます。 AWS Control Tower や AWS のサービスなどの他の AWS Firewall Managerも使用します AWS Config ルール。

AWS Organizations と AWS Secrets Manager を使用して IAM ユーザーアクセスキーを大規模に自動的にローテーションする

作成者: Tracy Hickey (AWS)、Gaurav Verma (AWS)、Laura Seletos (AWS)、Michael Davie (AWS)、Arvind Patel (AWS)

概要

> **Important**
>
> ベストプラクティスとして、AWS では、アクセスキーなどの長期的な認証情報を持つ IAM ユーザーの代わりに AWS Identity and Access Management (IAM) ロールを使用することをお勧めします。このパターンで説明されているアプローチは、長期間有効な AWS API 認証情報を必要とするレガシー実装のみを対象としています。このような実装でも、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイルや IAM Roles Anywhere を使用するなど、短期認証情報を使用するオプションを検討することをお勧めします。この記事のアプローチは、短期認証情報の使用にすぐには変更できず、長期認証情報をスケジュールに従ってローテーションする必要がある場合のみを対象としています。このアプローチでは、ローテーションされた API 認証情報を使用するようにレガシーアプリケーションのコードまたは設定を定期的に更新する必要があります。

アクセスキーは、IAM ユーザーまたは の長期的な認証情報です。IAM 認証情報を定期的にローテーションすることで、侵害された IAM アクセスキーのセットが AWS アカウントのコンポーネントにアクセスするのを防ぐことができます。IAM 認証情報のローテーションは、IAM におけるセキュリティのベストプラクティスの重要な部分でもあります。

このパターンは、GitHub IAM キーローテーションリポジトリで提供されている AWS CloudFormation テンプレートを使用して IAM アクセスキーを自動的にローテーションするのに役立ちます。

このパターンは、1 つのアカウントまたは複数のアカウントでのデプロイをサポートします。AWS Organizations を使用している場合、このソリューションは組織内のすべての AWS アカウント ID を識別し、アカウントの削除や新しいアカウントの作成に応じて動的にスケーリングします。一元管理された AWS Lambda 関数は、想定した IAM ロールを使用して、選択した複数のアカウントにわたってローテーション関数をローカルに実行します。

- 新しい IAM アクセスキーは、既存のアクセスキーが 90 日経過すると生成されます。
- 新しいアクセスキーは AWS Secrets Manager にシークレットとして保存されます。リソースベースのポリシーでは、指定した IAM プリンシパルのみにシークレットへのアクセスと取得を許可します。管理アカウントにキーを保存すると、すべてのアカウントのキーが管理アカウントに保存されます。
- 新しいアクセスキーが作成された AWS アカウントの所有者に割り当てられたメールアドレスが通知を受け取ります。
- 以前のアクセスキーは 100 日後に非アクティブ化され、110 日後に削除されます。
- 一元管理された E メール通知が AWS アカウント所有者に送信されます。

Lambda 関数と Amazon CloudWatch は、これらのアクションを自動的に実行します。その後、新しいアクセスキーペアを取得して、コードまたはアプリケーション内で置き換えることができます。ローテーション、削除、非アクティブ化の期間はカスタマイズできます。

前提条件と制限

- 少なくとも 1 つのアクティブな AWS アカウント。
- AWS Organizations、設定、セットアップ (チュートリアルを参照)。
- 管理アカウントから AWS Organizations をクエリする権限。詳細については、AWS Organizations ドキュメントの「AWS Organizations とサービスにリンクされたロール」を参照してください。
- AWS CloudFormation テンプレートと関連リソースを起動する権限を持つ IAM プリンシパル。詳細については、AWS CloudFormation のドキュメントの「セルフマネージド型のアクセス許可を付与する」を参照してください。
- リソースをデプロイするための既存の Amazon Simple Storage Service (Amazon S3) バケット。
- サンドボックス外に移動した Amazon Simple Email Service (Amazon SES)。詳細については、Amazon SES ドキュメントの「Amazon SES サンドボックス外への移動」を参照してください。
- 仮想プライベートクラウド (VPC) で Lambda を実行する場合は、メインの CloudFormation テンプレートを実行する前に以下のリソースを作成する必要があります。
  - VPC。
  - サブネット。
  - Amazon SES、AWS Systems Manager、AWS Security Token Service (AWS STS)、Amazon S3、AWS Secrets Manager のエンドポイント。(GitHub IAM key rotation リポジトリで提供されているエンドポイントテンプレートを実行して、これらのエンドポイントを作成できます。)

- AWS Systems Manager のパラメータ (SSM パラメータ) に保存されている簡易メール転送プロトコル (SMTP) ユーザーおよびパスワード。パラメータは、メインの CloudFormation テンプレートのパラメータと一致する必要があります。

アーキテクチャ

テクノロジースタック

- Amazon CloudWatch
- Amazon EventBridge
- IAM
- AWS Lambda
- AWS Organizations
- Amazon S3

アーキテクチャ

次の図は、このパターンのコンポーネントとワークフローを示しています。このソリューションでは、認証情報をメンバーアカウントと管理アカウントに保存する 2 つのシナリオがサポートされています。

オプション 1: 認証情報をメンバーアカウントに保存する

オプション 2: 認証情報を管理アカウントに保存する

図は次のワークフローを示しています。

1. EventBridge イベントは 24 時間ごとに `account_inventory` Lambda 関数を開始します。
2. この Lambda 関数は、すべての AWS アカウント ID、アカウント名、アカウントメールのリストを AWS Organizations に問い合わせます。
3. `account_inventory` Lambda 関数は AWS アカウント ID ごとに `access_key_auto_rotation` Lambda 関数を開始し、メタデータをそのアカウントに渡して追加の処理を行います。

4. access_key_auto_rotation Lambda 関数は、想定した IAM ロールを使用して AWS アカウント ID にアクセスします。Lambda スクリプトは、アカウント内のすべてのユーザーとそのユーザーの IAM アクセスキーに対して監査を実行します。
5. IAM アクセスキーの有効期間がベストプラクティスのしきい値を超えていない場合、Lambda 関数はそれ以上のアクションを実行しません。
6. IAM アクセスキーの有効期間がベストプラクティスのしきい値を超えた場合、access_key_auto_rotation Lambda 関数は実行するローテーションアクションを決定します。
7. アクションが必要な場合、新しいキーが生成されると、access_key_auto_rotation Lambda 関数は AWS Secrets Manager でシークレットを作成して更新します。また、指定した IAM プリンシパルのみがシークレットにアクセスしてシークレットを取得できるようにするリソースベースのポリシーも作成されます。オプション 1 の場合、認証情報はそれぞれのアカウントの Secrets Manager に保存されます。オプション 2 の場合 (StoreSecretsInCentralAccount フラグが True に設定されている場合)、認証情報は管理アカウントの Secrets Manager に保存されます。
8. notifier Lambda 関数が開始され、アカウントの所有者にローテーションアクティビティが通知されます。この関数は、AWS アカウント ID、アカウント名、アカウント E メール、および実行されたローテーションアクションを受け取ります。
9. notifier Lambda 関数はデプロイメント S3 バケットにメールテンプレートをクエリし、関連するアクティビティメタデータを使用して動的に更新します。その後、E メールはアカウントオーナーの E メールアドレスに送信されます。

注意:

- このソリューションは、複数のアベイラビリティゾーンでの耐障害性をサポートします。ただし、複数の AWS リージョンでの耐障害性はサポートされていません。複数のリージョンでサポートする場合は、2 番目のリージョンにソリューションをデプロイし、キーローテーションの EventBridge ルールを無効のままにしておくことができます。その後、2 つ目のリージョンでソリューションを実行したいときにルールを有効にできます。
- このソリューションは監査モードで実行できます。監査モードでは IAM アクセスキーは変更されませんが、ユーザーに通知するメールが送信されます。ソリューションを監査モードで実行するには、キーローテーションテンプレートを実行するとき、または access_key_auto_rotation Lambda 関数の環境変数で DryRunFlag フラグを True に設定します。

自動化とスケール

このソリューションを自動化する CloudFormation テンプレートは、GitHub IAM key rotation リポジトリで提供され、「Code」セクションにリストされています。AWS Organizations では、ソリューションを各メンバーアカウントに個別にデプロイするのではなく、CloudFormation StackSets を使用して `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` CloudFormation テンプレートを複数のアカウントにデプロイできます。

ツール

AWS サービス

- Amazon CloudWatch は、AWS のリソースや、AWS で実行されるアプリケーションをリアルタイムにモニタリングします。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Organizations」は、複数の AWS アカウントを 1 つの組織に統合し、作成と一元管理するためのアカウント管理サービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- Amazon Simple Email Service (Amazon SES) は、独自の E メールアドレスとドメインを使用して E メールを送受信するのに役立ちます。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。
- Amazon VPC エンドポイントは、AWS PrivateLink を使用するサービス (多くの AWS サービスを含め) に接続するためのインターフェイスを提供します。VVPC から指定したサブネットごとに、サブネット内にエンドポイントネットワークインターフェイスが作成され、サブネットアドレス範囲からプライベート IP アドレスが割り当てられます。

コード

必要な AWS CloudFormation テンプレート、Python スクリプト、およびランブックドキュメントは、GitHub IAM key rotation リポジトリにあります。テンプレートは次のようにデプロイされます。

テンプレート	デプロイ先	Notes (メモ)
`ASA-iam-key-auto-rotation-and-notifier-solution.yaml`	デプロイアカウント	これはソリューションのメインテンプレートです。
`ASA-iam-key-auto-rotation-iam-assumed-roles.yaml`	認証情報をローテーションしたい単一または複数のメンバーアカウント	CloudFormation スタックセットを使用して、このテンプレートを複数のアカウントにデプロイできます。
`ASA-iam-key-auto-rotation-list-accounts-role.yaml`	セントラル/管理アカウント	このテンプレートを使用して、AWS Organizations のアカウントのインベントリを管理します。
`ASA-iam-key-auto-rotation-vpc-endpoints.yaml`	デプロイアカウント	Lambda 関数を VPC で実行する (メインテンプレートで `RunLambdaInVPC` パラメータを True に設定する) 場合にのみ、このテンプレートを使用してエンドポイントの作成を自動化します。

エピック

ソリューションをセットアップする

タスク	説明	必要なスキル
デプロイする S3 バケットを選択します。	アカウントの AWS マネジメントコンソールにサインインし、Amazon S3 コンソールを	クラウドアーキテクト

タスク	説明	必要なスキル
	開いて、デプロイする S3 バケットを選択します。AWS Organizations の複数のアカウントにソリューションを実装する場合は、組織の管理アカウントにサインインします。	
リポジトリをクローン作成します。	GitHub IAM key rotation リポジトリをローカルデスクトップにクローンします。	クラウドアーキテクト
ファイルを S3 バケットにアップロードします。	クローンファイルを S3 バケットにアップロードします。以下のデフォルトフォルダ構造を使用して、クローンされたファイルとディレクトリをすべてコピーして貼り付けます。`asa/asa-iam-rotation` **Note** このフォルダ構造は、CloudFormation テンプレートでカスタマイズできます。	クラウドアーキテクト

タスク	説明	必要なスキル
E メールテンプレートを変更します。	`iam-auto-key-rotation-enforcement.html` E メールテンプレート (`template` フォルダ内) を要件に合わせて変更します。テンプレートの末尾の `[Department Name Here]` を部門名に置き換えます。	クラウドアーキテクト

解決策をデプロイする

タスク	説明	必要なスキル
キーローテーション用の CloudFormation テンプレートを起動します。	1. デプロイアカウントで `ASA-iam-key-auto-rotation-and-notifier-solution.yaml` テンプレートを起動します。詳細については、CloudFormation ドキュメントの「スタックテンプレートの選択」を参照してください。 2. 次のようなパラメータの値を指定します。 • CloudFormation S3 バケット名 (`S3BucketName`) – Lambda コードを含むデプロイメント S3 バケットの名前。 • CloudFormation S3 バケットプレフィックス	クラウドアーキテクト

タスク	説明	必要なスキル
	(S3BucketPrefix)– S3 バケットのプレフィックス。 • 仮想 IAM ロール名 (IAMRoleName)– key-rotation Lambda 関数がキーをローテーションするために想定するロール名。 • IAM 実行ロール名 (ExecutionRoleName) – key-rotation Lambda 関数が使用する IAM 実行ロールの名前。 • インベントリ実行ロール名 (Inventory Execution RoleName)– account_inventory Lambda 関数が使用する IAM 実行ロールの名前。 • ドライランフラグ (監査モード) (**DryRunFlag**) – True に設定すると監査モードがオンになります (デフォルト)。False に設定すると、強制モードがオンになります。 • 組織アカウントを一覧表示するアカウント (OrgListAccount)– 組織内のアカウントを一覧表示するために使用さ	

タスク	説明	必要なスキル
	れる中央/管理アカウントのアカウント ID。 • アカウント一覧ロール名 (`OrgListRole`) – 組織内のアカウントを一覧表示するために使用されるロール名。 • セントラルアカウントの Secrets Store フラグ (`StoreSecretsInCentralAccount`) – True に設定すると、セントラルアカウントにシークレットが保存されます。False に設定すると、シークレットはそれぞれのアカウントに保存されます。 • 認証情報を複製するリージョン (`CredentialReplicationRegions`) – 認証情報を複製する AWS リージョン (Secrets Manager) を、カンマで区切って指定します。例: `us-east-2,us-west-1,us-west-2` スタックの作成しているスタックのリージョンをスキップします。 • VPC で Lambda を実行 (`RunLambdaInVpc`) –	

タスク	説明	必要なスキル
	指定した VPC で Lambda 関数を実行するには True に設定します。VPC エンドポイントを作成し、Lambda 関数を含むサブネットに NAT ゲートウェイをアタッチする必要があります。詳細については、このオプションについて説明している re: POST の記事を参照してください。 • Lambda 関数の VPC ID (VpcId)、セキュリティグループルールの VPC CIDR (VpcCidr)、および Lambda 関数のサブネット ID (SubnetId) – RunLambdaInVpc を True に設定した場合、VPC、CIDR、サブネットに関する情報を提供します。 • 管理者メールアドレス (AdminEmailAddress) – 通知の送信先となる有効なメールアドレス。 • AWS 組織 ID (AWSOrgID) – 組織の固有の ID。この ID は o- で始まり、その後に 10 ~ 32 個の小文字または数字が続きます。	

タスク	説明	必要なスキル
	• E メールテンプレートファイル名 [監査モード] (EmailTemplateAudit) と [強制モード] (EmailTemplateEnforce) – notifier モジュールが監査モードと強制モードで送信する電子メール HTML テンプレートのファイル名。 • SMTP ユーザー SSM パラメータ名 (SMTPUserParamName) と SMTP パスワード SSM パラメータ名 (SMTPPasswordParamName) – 簡易メール転送プロトコル (SMTP) のユーザーとパスワード情報。	

タスク	説明	必要なスキル
引き受けたロール用の CloudFormation テンプレートを起動します。	1. AWS CloudFormation コンソールで、キーをローテーションしたい各アカウントの `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` テンプレートを起動します。複数のアカウントをお持ちの場合は、メインの CloudFormation テンプレートを管理アカウントにスタックとしてデプロイし、CloudFormation スタックセットを含む `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` テンプレートを必要なすべてのアカウントにデプロイできます。詳細については、CloudFormation ドキュメントの「AWS CloudFormation StackSets の操作」を参照してください。 2. 次のパラメータの値を指定します。 • 仮想 IAM ロール名 (`IAMRoleName`) – Lambda `access_key_auto_rotation` 関数によって引き受けられる IAM ロール名。デフォルト値をそのまま使用できます。	クラウドアーキテクト

タスク	説明	必要なスキル
	• IAM 実行ロール名 (`ExecutionRoleName`) – Lambda 関数を実行するサブアカウントロールを引き受ける IAM ロール。 • プライマリ AWS アカウント ID (`PrimaryAccountID`) – メインテンプレートがデプロイされる AWS アカウント ID。 • IAM 免除グループ (`IAMExemptionGroup`) – 自動キーローテーションから除外したい IAM アカウントを円滑に進めるために使用される IAM グループ名。	

タスク	説明	必要なスキル
アカウントインベントリ用の CloudFormation テンプレートを起動します。	1. 管理/中央アカウントで `ASA-iam-key-auto-rotation-list-accounts-role.yaml` テンプレートを起動します 2. 次のパラメータの値を指定します。 • 仮想 IAM ロール名 (`IAMRoleName`) – Lambda `access_key_auto_rotation` 関数が引き受ける IAM ロール名。 • アカウント Lambda の IAM 実行ロール名 (`AccountExecutionRoleName`) – Lambda `notifier` 関数が引き受ける IAM ロールの名前。 • ローテーション用の IAM 実行ロール名 Lambda (`RotationExecutionRoleName`) – Lambda `access_key_auto_rotation` 関数が引き受ける IAM ロールの名前。 • プライマリ AWS アカウント ID (`PrimaryAccountID`) – メインテンプレートがデプロイされる AWS アカウント ID。	クラウドアーキテクト

タスク	説明	必要なスキル
VPC エンドポイント用の CloudFormation テンプレートを起動します。	このタスクはオプションです。 1. デプロイアカウントで `ASA-iam-key-auto-rotation-vpc-endpoints.yaml` テンプレートを起動します。 2. 次のパラメータの値を指定します。 • VPC ID (`pVpcId`)、サブネット ID (`pSubnetId`)、VPC の CIDR 範囲 (`pVPCCidr`) – VPC、CIDR、サブネットに関する情報を提供します。 • 各 VPC エンドポイントのパラメータを True に設定します。エンドポイントが既にある場合は、False を選択できます。	クラウドアーキテクト

関連リソース

- IAM でのセキュリティのベストプラクティス (IAM ドキュメント)
- AWS Organizations とサービスにリンクされたロール (AWS Organizations ドキュメント)
- スタックテンプレートの選択 (CloudFormation ドキュメント)
- AWS CloudFormation StackSets の操作 (CloudFormation ドキュメント)

CodePipeline、IAM アクセスアナライザー、AWS CloudFormation マクロを使用して、AWS アカウントの IAM ポリシーとロールを自動的に検証してデプロイする

作成者: Helton ribeiro (AWS) と Guilherme Simoes (AWS)

概要

注意: AWS CodeCommit は、新規のお客様にはご利用いただけません。AWS CodeCommit の既存のお客様は、通常どおりサービスを引き続き使用できます。詳細はこちら。

このパターンでは、開発チームが Amazon Web Services (AWS)アカウントで AWS Identity and Access Management (IAM) のポリシーとロールを作成できるようにするデプロイパイプラインを作成するための手順を説明し、コードを提供します。このアプローチは、運用チームの負担を減らし、デプロイプロセスをスピードアップするのに役立ちます。また、デベロッパーが既存のガバナンスやセキュリティ統制と両立する IAM ロールやポリシーを作成するのにも役立ちます。

このパターンのアプローチでは、AWS Identity and Access Management アクセスアナライザーを使用して IAM ロールにアタッチしたい IAM ポリシーを検証し、AWS CloudFormation を使用して IAM ロールをデプロイします。ただし、開発チームは AWS CloudFormation テンプレートファイルを直接編集する代わりに、JSON 形式の IAM ポリシーとロールを作成します。AWS CloudFormation マクロは、デプロイを開始する前に、これらの JSON 形式のポリシーファイルを AWS CloudFormation IAM リソースタイプに変換します。

デプロイパイプライン (`RolesPipeline`) には、ソース、検証、デプロイの各段階があります。ソースステージでは、開発チームが IAM ロールとポリシーの定義を含む JSON ファイルを AWS CodeCommit リポジトリにプッシュします。次に、AWS CodeBuild はスクリプトを実行してそれらのファイルを検証し、Amazon Simple Storage Service (Amazon S3) バケットにコピーします。開発チームは別の S3 バケットに保存されている AWS CloudFormation テンプレートファイルに直接アクセスできないため、JSON ファイルの作成と検証のプロセスに従う必要があります。

最後に、デプロイフェーズ中に、AWS CodeDeploy は AWS CloudFormation スタックを使用して、アカウント内の IAM ポリシーとロールを更新または削除します。

> **Important**
>
> このパターンのワークフローは概念実証 (POC) であり、テスト環境でのみ使用することをお勧めします。このパターンのアプローチを本番環境で使用する場合は、IAM ドキュメントの

「IAM でのセキュリティのベストプラクティス」を参照し、IAM ロールと AWS サービスに必要な変更を加えます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- `RolesPipeline` パイプライン用の新規または既存の S3 バケット。使用しているアクセス認証情報に、このバケットにオブジェクトをアップロードする権限があることを確認してください。
- インストールおよび設定済みの AWS コマンドラインインターフェイス (AWS CLI)。詳細については、AWS CLI ドキュメントの「AWS CLI の最新バージョンを使用してインストールまたは更新を行う」を参照してください。
- インストールおよび設定済みの AWS サーバーレスアプリケーションモデル (AWS SAM) CLI。詳細については、AWS SAM ドキュメントの「AWS SAM CLI のインストール」を参照してください。
- ローカルマシンで Python 3 をインストールします。詳細については、Python のドキュメントを参照してください。
- インストールおよび設定済みの Git クライアント。
- ローカルマシンにクローンされた GitHub `IAM roles pipeline` リポジトリ。
- JSON 形式の既存の IAM ポリシーとロール。詳細については、Github `IAM roles pipeline` リポジトリの ReadMe ファイルを参照してください。
- 開発チームには、このソリューションの AWS CodePipeline、CodeBuild、および CodeDeploy リソースを編集する権限があってはなりません。

制限

- このパターンのワークフローは概念実証 (POC) であり、テスト環境でのみ使用することをお勧めします。このパターンのアプローチを本番環境で使用する場合は、IAM ドキュメントの「IAM でのセキュリティのベストプラクティス」を参照し、IAM ロールと AWS サービスに必要な変更を加えます。

アーキテクチャ

次の図は、CodePipeline、IAM アクセスアナライザー、および AWS CloudFormation マクロを使用して IAM ロールとポリシーを自動的に検証してアカウントにデプロイする方法を示しています。

この図表は、次のワークフローを示しています：

1. デベロッパーは IAM ポリシーとロールの定義を含む JSON ファイルを作成します。デベロッパーがコードを CodeCommit リポジトリにプッシュし、CodePipeline が `RolesPipeline` パイプラインを開始します。
2. CodeBuild は IAM Access Analyzer を使用して JSON ファイルを検証します。セキュリティまたはエラー関連の検出結果がある場合、デプロイプロセスは停止されます。
3. セキュリティまたはエラー関連の検出結果がない場合、JSON ファイルは `RolesBucket` S3 バケットに送信されます。
4. 次に、AWS Lambda 関数として実装された AWS CloudFormation マクロが `RolesBucket` バケットから JSON ファイルを読み取り、それらを AWS CloudFormation IAM リソースタイプに変換します。
5. 定義済みの AWS CloudFormation スタックは、アカウント内の IAM ポリシーとロールをインストール、更新、または削除します。

自動化とスケール

このパターンを自動的にデプロイする AWS CloudFormation テンプレートは、GitHub IAM ロールのパイプラインリポジトリで提供されます。

ツール

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- IAM Access Analyzer は、外部エンティティと共有されている組織およびアカウント (S3 バケットや IAM ロールなど) 内のリソースを識別するのに役立ちます。これは、リソースやデータへの意図しないアクセスを特定するのに役立ちます。

- AWS サーバーレスアプリケーションモデル (AWS SAM) は、AWS クラウドでサーバーレスアプリケーションを構築するのに役立つオープンソースフレームワークです。

コード

このパターンのソースコードとテンプレートは、GitHub IAM roles pipeline リポジトリにあります。

エピック

リポジトリをクローンする

タスク	説明	必要なスキル
サンプルリポジトリのクローンを作成します。	GitHub IAM roles pipeline リポジトリをローカルマシンにクローンします。	アプリ開発者、AWS 全般

RolesPipeline パイプラインをデプロイします。

タスク	説明	必要なスキル
パイプラインをデプロイします。	1. クローン作成されたリポジトリが含まれるディレクトリに移動します。 2. **Important** `make deploy bucket=<bucket_name>` コマンドを実行します。: を既存の S3 バケットのバケット名<bucket_name> に置き換える必要があります。	アプリ開発者、AWS 全般

タスク	説明	必要なスキル
	3. `aws codepipeline get-pipeline -name RolesPipeline` コマンドを実行して、デプロイが成功したかどうかを確認します。	
パイプラインのリポジトリをクローンします。	1. `RolesPipeline` AWS CloudFormation スタックは `roles-pipeline-repo` CodeCommit リポジトリを作成します。 2. AWS マネジメントコンソールにサインインし、AWS CodeCommit コンソールを開き、CodeCommit リポジトリの URL をコピーしてローカルマシンに複製します。これに関する詳細については、AWS CodeCommit ドキュメントの「AWS CodeCommit リポジトリに接続する」を参照してください。	アプリ開発者、AWS 全般

Roles/Pipeline パイプラインのテスト

タスク	説明	必要なスキル
有効な IAM ポリシーとロールを使用して RoleSipeline パイプラインをテストします。	1. IAM ポリシーとロールの JSON ファイルを作成します。GitHub `IAM roles pipeline` リポジトリの `role-example` ディレク	アプリ開発者、AWS 全般

タスク	説明	必要なスキル
	トリにあるサンプルを使用できます。 2. **Important** IAM のポリシーとロールを必要な設定で定義します。: GitHub `IAM roles pipeline`リポジトリの `ReadMe` ファイルで説明されている形式に従ってください。 3. 変更を `roles-pipeline-repo` CodeCommit リポジトリにプッシュします。 4. `RolesPipeline` パイプラインの実装を検証します。 5. IAM ポリシーとロールがアカウントに正しくデプロイされていることを確認します。 6. IAM ポリシーまたはロールに関連するアクセス許可の境界があるかどうかを確認します。詳細については、IAM ドキュメントの IAM エンティティのアクセス許可の境界を参照してください。	

タスク	説明	必要なスキル
無効な IAM ポリシーとロールを使用して RoleSpipeline パイプラインをテストします。	1. `roles-pipeline-repo` CodeCommit リポジトリを変更し、無効な IAM ロールまたはポリシーを含めてください。例えば、存在しないアクションや無効な IAM ポリシーバージョンを使用できます。 2. パイプラインの実装を検証してください。IAM Access Analyzer は、無効な IAM ポリシーまたはロールを検出すると、検証段階でパイプラインを停止します。	アプリ開発者、AWS 全般

リソースのクリーンアップ

タスク	説明	必要なスキル
クリーンアップの準備をします。	S3 バケットを空にしてから、`destroy` コマンドを実行します。	アプリ開発者、AWS 全般
RolesStack スタックを削除します。	1. `RolesPipeline` パイプラインは IAM ポリシーとロールをデプロイする `RolesStack` AWS CloudFormation スタックを作成します。`RolesPipeline` パイプラインを削除する前に、このスタックを削除する必要があります。 2. AWS マネジメントコンソールにサインインし、AWS	アプリ開発者、AWS 全般

タスク	説明	必要なスキル
	CloudFormation コンソールを開きます。`RolesStack` スタックを選択して [削除] を選択します。	
RoleSipeline スタックを削除します。	`RolesPipeline` AWS CloudFormation スタックを削除するには、Github `IAM roles pipeline` リポジトリの ReadMe ファイルにある指示に従ってください。	アプリ開発者、AWS 全般

関連リソース

- IAM Access Analyzer - Policy validation (AWS ニュースブログ)
- AWS CloudFormation マクロを使用したテンプレートのカスタム処理の実行 (AWS CloudFormation ドキュメント)
- Python による Lambda 関数の構築 (AWS Lambda ドキュメント)

AWS Security Hub と Jira ソフトウェアを双方向に統合する

ホアキン・マヌエル・リナウド (AWS) により作成

概要

このソリューションは、AWS Security Hub と Jira 間の双方向統合をサポートします。このソリューションを使用すると、Security Hub の検出結果から JIRA チケットを自動的かつ手動で作成および更新できます。セキュリティチームはこのインテグレーションを使用して、対処が必要な重大なセキュリティ検出結果を開発チームに通知できます。

このソリューションは次のことに役立ちます。

- Jira でチケットを自動的に作成または更新する Security Hub コントロールを選択します。
- Security Hub コンソールで、Security Hub カスタムアクションを使用して Jira のチケットを手動でエスカレーションします。
- AWS AWS Organizations で定義されている AWS アカウントタグに基づいて Jira でチケットを自動的に割り当てます。このタグが定義されていない場合、既定の担当者が使用されます。
- Jira で誤検出または許容リスクとしてマークされた Security Hub の検出結果を自動的に非表示にします。
- 関連する検出結果がSecurity Hub にアーカイブされると、Jira チケットを自動的に閉じます。
- Security Hub 検出結果が再発したら、Jira チケットを再度開きます。

Jira ワークフロー

このソリューションでは、開発者がリスクを管理および文書化できるカスタム Jira ワークフローを使用しています。課題がワークフロー内を移動するにつれて、双方向統合により、Jira チケットと Security Hub の検出結果のステータスが両方のサービスのワークフロー間で確実に同期されます。このワークフローは、Dinis Cruz による SecDevOps リスクワークフローから派生したもので、「CC BY 4.0」 の下でライセンスされています。セキュリティチームのメンバーだけがチケットのステータスを変更できるように、Jira ワークフロー条件を追加することをお勧めします。

このソリューションによって自動的に生成される Jira チケットの例については、このパターンの「追加情報」セクションを参照してください。

前提条件と制限

前提条件

- このソリューションをマルチアカウントの AWS 環境にデプロイする場合:
  - マルチアカウント環境はアクティブで、AWS Organizations によって管理されています。
  - Security Hub は AWS アカウントで有効になっています。
  - AWS Organizations では、Security Hub 管理者アカウントが指定されています。
  - AWS Organizations `AWSOrganizationsReadOnlyAccess` 管理アカウントへのアクセス権限を持つクロスアカウント IAM ロールがあります。
  - (オプション) AWS アカウントには `SecurityContactID` のタグが付けられています。このタグは、Jira チケットを定義済みのセキュリティ連絡先に割り当てるために使用されます。
- このソリューションを単一の AWS アカウントにデプロイする場合:
  - アクティブな AWS アカウントがあります。
  - AWS アカウントで Security Hub が有効になっています。
- Jira サーバーインスタンス

> **Important**
>
> このソリューションは、Jira Cloud の使用をサポートしています。ただし、Jira Cloud は XML ワークフローのインポートをサポートしていないため、Jira でワークフローを手動で再作成する必要があります。

- Jira の管理者権限
- 次の Jira トークンのいずれか:
  - Jira エンタープライズの場合は、個人アクセストークン (PAT) です。詳細については、「個人アクセストークンの使用」(アトラシアンサポート) を参照してください。
  - Jira クラウドの場合は Jira API トークンです。詳細については、「API トークンの管理」(Atlassian サポート) を参照してください。

アーキテクチャ

このセクションでは、開発者とセキュリティエンジニアがリスクを受け入れるか、問題を解決するかを決定したときなど、さまざまなシナリオにおけるソリューションのアーキテクチャを示しています。

シナリオ 1: 開発者が問題に対処する

1. Security Hub は、「AWS Foundational Security Best Practices標準」のような特定のセキュリティコントロールに対する結果を生成します。
2. 検出結果とアクションに関連付けられた Amazon CloudWatch Eventsにより、`CreateJIRA` AWS Lambda 関数が開始されます。
3. Lambda 関数は、`GeneratorId` 設定ファイルと結果のフィールドを使用して、検出結果をエスカレートする必要があるかどうかを評価します。
4. Lambda 関数は検出結果をエスカレーションする必要があると判断し、`SecurityContactID` AWS 管理アカウントの AWS Organizations からアカウントタグを取得します。この ID は開発者に関連付けられ、Jira チケットの担当者 ID として使用されます。
5. Lambda 関数は、AWS Secrets Manager に保存されている認証情報を使用して Jira にチケットを作成します。Jira は開発者に通知します。
6. 開発者は基礎となるセキュリティ検出結果に対処し、Jira ではチケットのステータスを `TEST FIX` に変更します。
7. Security Hub は検出結果を `ARCHIVED` として更新し、新しいイベントが生成されます。このイベントにより、Lambda 関数は Jira チケットを自動的に閉じます。

シナリオ 2: 開発者がリスクを受け入れることを決定

1. Security Hub は、「AWS Foundational Security Best Practices標準」のような特定のセキュリティコントロールに対する結果を生成します。
2. 検出結果と `CreateJIRA` アクションに関連付けられた CloudWatch イベントは、Lambda 関数を開始します。
3. Lambda 関数は、`GeneratorId` 設定ファイルと結果のフィールドを使用して、検出結果をエスカレートする必要があるかどうかを評価します。
4. Lambda 関数は検出結果をエスカレーションする必要があると判断し、`SecurityContactID` AWS 管理アカウントの AWS Organizations からアカウントタグを取得します。この ID は開発者に関連付けられ、Jira チケットの担当者 ID として使用されます。
5. Lambda 関数は、Secrets Manager に保存されている認証情報を使用して Jira にチケットを作成します。Jira は開発者に通知します。
6. 開発者はリスクを受け入れることを決定し、Jira ではチケットのステータスを `AWAITING RISK ACCEPTANCE` に変更します。

7. セキュリティエンジニアはリクエストを検討し、ビジネス上の正当性が適切であると判断します。セキュリティエンジニアは Jira チケットのステータスを ACCEPTED RISK に変更します。これにより Jira チケットは終了します。
8. CloudWatch の日次イベントにより Lambda 関数 の更新機能が開始されます。これにより、クローズされた JIRA チケットが識別され、関連するSecurity Hub 検出結果がとして更新されます。SUPPRESSED

ツール

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「Amazon CloudWatch Events」を使用すると、ルールを使用してイベントを照合し、関数またはストリームにルーティングすることで、AWS リソースのシステムイベントを監視できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「AWS Organizations」は、複数の AWS アカウントを 1 つの組織に統合し、作成と一元管理するためのアカウント管理サービスです。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- 「AWS Security Hub」は、AWS のセキュリティ状態の包括的ビューを提供します。また、セキュリティ業界の標準とベストプラクティスに対してお使いの AWS 環境をチェックする上で役立ちます。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-securityhub-jira-ソフトウェア統合」リポジトリで利用できます。このソリューションのサンプルコードと Jira ワークフローが含まれています。

エピック

Jira を設定する

タスク	説明	必要なスキル
ワークフローをインポートします。	Jira の管理者として、`issue-workflow.xml` ファイルを Jira Server インスタンスにインポートします。このファイルは GitHub の「aws-securityhub-jira」ソフトウェアインテグレーションリポジトリにあります。手順については、「XML を使用したワークフローの作成」(Jira ドキュメント) を参照してください。	Jira 管理者
ワークフローを有効化して割り当てます。	ワークフローは、ワークフロースキームに割り当てるまで非アクティブです。次に、そのワークフロースキームをプロジェクトに割り当てます。 1. プロジェクトについて、そのプロジェクト用の課題タイプスキームを特定したことを確認します。新しい課題タイプを作成することも、既存の課題タイプ (Bug など) から選択することもできます。 2. 「ワークフローの有効化 (Jira ドキュメント)」 の指示に従って、インポートしたワークフローをワークフ	Jira 管理者

タスク	説明	必要なスキル
	ロースキームに割り当てます。 3. 「ワークフロースキームをプロジェクトに関連付ける (Jira ドキュメント)」 の指示に従って、ワークフロースキームをプロジェクトに割り当てます。	

ソリューションパラメーターをセットアップします。

タスク	説明	必要なスキル
ソリューションパラメータを設定します。	1. conf フォルダーで、`params_prod.shfile` を開きます。 2. 次のパラメータの値を指定します。 • `ORG_ACCOUNT_ID` — AWS Organizations の管理アカウントのアカウント ID。このソリューションはアカウントタグを読み取り、それらの AWS アカウントタグで定義されている特定のセキュリティ連絡先にチケットを割り当てます。 • `ORG_ROLE` — AWS Organizations 管理アカウントへのアクセスに使用される IAM ロールの	AWS システム管理者

タスク	説明	必要なスキル
	名前。このロールには、OrganizationsReadOnlyAccess のアクセス許可が必要です。 • EXTERNAL_ID — 外部 IDを使用して ORG_ROLE で定義されている IAM ロールを引き継ぐ場合のオプションパラメータ。詳細については、「外部 ID の使用方法」(IAM ドキュメント)を参照してください。 • JIRA_DEFAULT_ASSIGNEE — これはすべてのセキュリティ問題のデフォルト担当者の Jira ID です。このデフォルト割り当ては、アカウントに適切にタグ付けされていない場合や、役割を引き受けられない場合に使用されます。 • JIRA_INSTANCE — Jira サーバーの HTTPS アドレスは次の team-<team-id>.atlassian.net/ 形式です。 • JIRA_PROJECT_KEY — チケットの作成に使用される Jira プロジェクトキーの名前 (SEC や TEST など)。このプロジェクト	

タスク	説明	必要なスキル
	は Jira 内に既に存在している必要があります。 • ISSUE_TYPE — Jira 内のプロジェクトに割り当てられている課題タイプスキームの名前 (Bug や Security Issue など)。 • REGIONS — このソリューションをデプロイする AWS リージョンコードのリスト (eu-west-1 など)。 3. ソリューションパラメータファイルを保存して閉じます。	

タスク	説明	必要なスキル
自動化したい検出結果を特定してください。	1. 「https://console.aws.amazon.com/securityhub/」 のセキュリティハブコンソールを開きます。 2. Security Hub ナビゲーションペインで、[検出結果] を選択します。 3. 結果のタイトルを選択します。 4. 検出結果 ID を選択します。これにより、検出結果の完全な JSON が表示されます。 5. JSON で、`GeneratorId` フィールド内の文字列をコピーします。この値は 「AWS セキュリティ検出結果フォーマット」 (ASFF) です。たとえば、Security Control の検出結果に応じて、S3 `aws-foundational-security-best-practices/v/1.0.0/S3.1` の Block Public Access 設定を有効にする必要があります。 6. `GeneratorID` 自動化したい検出結果の値をすべてコピーするまで、これらの手順を繰り返します。	

タスク	説明	必要なスキル
検出結果を設定ファイルに追加します。	1. src/code で、config.jsonconfig ファイルを開きます。 2. 前の記事で取得した GeneratorID 値を default パラメーターに貼り付け、各 ID をコンマで区切ります。 3. 設定ファイルを保存して閉じます。 次のコード例は、aws-foundational-security-best-practices/v/1.0.0/SNS.1 および aws-foundational-security-best-practices/v/1.0.0/S3.1 の検出結果を自動化する方法を示しています。 `{ "Controls" : { "eu-west-1": [ "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.22" ], "default": [ aws-foundational-security-best-practices/v/1.0.0/SNS.1,`	AWS システム管理者

タスク	説明	必要なスキル

	<pre>aws-foundational-
	security-best-p
	ractices/v/1.0.0/S3.1
	]
	}
	}</pre>
	Note
	AWS リージョンごとに異なる検出結果を自動化することを選択できます。検出結果の重複を防ぐには、IAM 関連の統制の作成を自動化するリージョンを 1 つ選択することをおすすめします。	

インテグレーションをデプロイ

タスク	説明	必要なスキル
統合をデプロイします。	コマンドライン端末で、次のコマンドを入力します。 <pre>./deploy.sh prod</pre>	AWS システム管理者
Jira 認証情報を AWS Secrets Manager にアップロードします。	1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。	AWS システム管理者

タスク	説明	必要なスキル
	2. シークレットで、新しいシークレットの保存を選択します。 3. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。 4. Jira Enterprise を使用している場合は、キーと値のペアに対して次の操作を行います。 • 1 `auth` 行目にキーボックスに入力し、`token_auth` 次に値ボックスに入力します。 • 2 `token` 行目を追加してキーボックスに入力し、バリューボックスに個人アクセストークンを入力します。 Jira Cloud を使用している場合、キーと値のペアでは次の操作を行います。 • 1 `auth` 行目にキーボックスに入力し、`basic_auth` 次に値ボックスに入力します。 • 2 `token` 行目を追加してキーボックスに入力し、値ボックスに API トークンを入力します。	

タスク	説明	必要なスキル
	• 3 email 行目を追加してキーボックスに入力し、値ボックスにメールアドレスを入力します。 5. [Next (次へ)] を選択します。 6. シークレット名に対して、`Jira-Token` を入力して、ページの下部で次へを選択します。 7. シークレットローテーションページで自動ローテーションを無効にするのままにし、ページ下部で次へを選択します。 8. レビューページで、シークレットの詳細を確認し、保存をクリックします。	

タスク	説明	必要なスキル
セキュリティハブカスタムアクションを作成します。	1. AWS リージョンごとに、AWS コマンドラインインターフェイス (AWS CLI) で「create-action-target」コマンドを使用して、`CreateJiraIssue` という名前のSecurity Hub カスタムアクションを作成します。 `aws securityhub create-action-target --name "CreateJiraIssue" \ --description "Create ticket in JIRA" \ --id "CreateJiraIssue" --region $<aws-region>` 2. 「https://console.aws.amazon.com/securityhub/」セキュリティハブコンソールを開きます。 3. Security Hub ナビゲーションペインで、[検出結果] を選択します。 4. 検出結果のリストから、エスカレートしたい検出結果を選択します。 5. アクションメニューで `CreateJiraIssue` を選択します。	AWS システム管理者

関連リソース

- 「AWS Service Management Connector for Jira Service Management」
- 「AWS の基本的なセキュリティのベストプラクティス標準」

追加情報

Jira チケットの例

指定した Security Hub の検出結果が発生すると、このソリューションは自動的に Jira チケットを作成します。チケットには、次の情報が含まれます。

- タイトル — タイトルは次の形式でセキュリティ上の問題を識別します。

```
AWS Security Issue :: <AWS account ID> :: <Security Hub finding title>
```

- 説明 — チケットの説明セクションには、検出結果に関連するセキュリティコントロールについての説明と、Security Hub コンソールの検出結果へのリンク、および Jira ワークフローにおけるセキュリティ問題の処理方法に関する簡単な説明が記載されています。

以下は、自動生成された Jira チケットの例です。

タイトル	AWS セキュリティイシュー :: 012345678912 :: Lambda.1 Lambda 関数ポリシーでは、パブリックアクセスを禁止する必要があります。
説明	問題は何か? お客様が担当している AWS アカウント 012345678912 内でセキュリティ上の検出結果が検出されました。 このコントロールは、Lambda リソースにアタッチされている AWS Lambda 関数ポリシーがパブリックアクセスを禁止するかどうかを検査します。Lambda 関数ポリシーがパブリックアクセスを許可している場合、コントロールは失敗します。 <セキュリティハブの検出結果へのリンク>

チケットはどうする必要がありますか?

- アカウントにアクセスして設定を確認します。チケットを「修正済み」に移動して処理中のチケットを承認します。修正したら、テスト修正に移行し、セキュリティが問題を解決したことを検証します。
- リスクを受け入れるべきだと思う場合は、「リスクの受け入れ待ち」に移動します。これにはセキュリティエンジニアによるレビューが必要です。
- 誤検知と思われる場合は、「誤検知としてマークする」に移行します。これはセキュリティエンジニアによって確認され、それに応じて再開/終了されます。

EC2 Image Builder と Terraform を使用して、強化されたコンテナイメージ用のパイプラインを構築する

作成者: Mike Saintcross (AWS) and Andrew Ranes (AWS)

概要

このパターンは、強化した Amazon Linux 2 ベースコンテナイメージを生成する EC2 Image Builder パイプラインを構築します。Terraform は、強化されたコンテナイメージの作成に使用されるインフラストラクチャを設定してプロビジョニングするための Infrastructure as Code (IaC) ツールとして使用されます。このレシピは、Red Hat Enterprise Linux (RHEL) 7 STIG バージョン 3 リリース 7 — Medium によって強化された Docker ベースの Amazon Linux 2 コンテナイメージのデプロイに役立ちます。(EC2 Image Builder ドキュメントの「Linux STIG コンポーネント」セクションにある「STIG-Build-Linux-Medium バージョン 2022.2.1」を参照してください。) これは、ゴールデンコンテナイメージと呼ばれます。

ビルドには 2 つの Amazon EventBridge ルールが含まれています。1 つ目のルールは、Amazon Inspector の検出結果が「高」または「重要」の場合にコンテナイメージパイプラインを開始し、セキュアでないイメージが置き換えられるようにします。このルールでは、Amazon Inspector と Amazon Elastic Container Registry (Amazon ECR) の両方の拡張スキャンを有効にする必要があります。2 つ目のルールは、Amazon ECR リポジトリへのイメージのプッシュが成功すると、Amazon Simple Queue Service (Amazon SQS) キューに通知を送信します。これにより、最新のコンテナイメージを使用できるようになります。

> **Note**
>
> Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 の FAQs」を参照してください。

前提条件と制限

前提条件

- インフラストラクチャをデプロイできる AWS アカウント。
- ローカルデプロイ用の AWS 認証情報を設定するために AWS コマンドラインインターフェイス (AWS CLI) をインストールします。

- Terraform ドキュメントの「指示」に従って Terraform をダウンロードし、セットアップしました。
- Git (ローカルマシンからプロビジョニングする場合)。
- AWS リソースの作成に使用できる AWS アカウント内の ロール。
- .tfvars ファイルで定義されているすべての変数。 または、Terraform 設定を適用するときにすべての変数を定義できます。

機能制限

- このソリューションは、プライベートサブネットからのインターネット接続用の NAT ゲートウェイ と インターネットゲートウェイ を含む Amazon Virtual Private Cloud (Amazon VPC) インフラストラクチャを作成します。VPC エンドポイント を使用できないのは、AWS タスクオーケストレーターおよびエグゼキューター (AWSTOE) によるブートストラップ プロセスでは、インターネットから AWSCLI バージョン 2 がインストールされるためです。

製品バージョン

- Amazon Linux 2
- AWS CLI バージョン 1.1 以降

アーキテクチャ

ターゲットテクノロジースタック

このパターンでは、以下を含む 43 個のリソースが作成されます。

- 2 つの Amazon Simple Storage Service (Amazon S3) バケット。1 つはパイプラインコンポーネントファイル用、もう 1 つはサーバーアクセスと Amazon VPC フローログ用です
- Amazon ECR リポジトリ
- パブリックサブネット、プライベートサブネット、ルートテーブル、NAT ゲートウェイ、インターネットゲートウェイを含む仮想プライベートクラウド (VPC)
- EC2 Image Builder パイプライン、レシピ、コンポーネント
- コンテナイメージ
- イメージを暗号化するための AWS Key Management Service (AWS KMS) キー
- SQS キュー

- 3 つのロール: 1 つ目は EC2 Image Builder パイプラインを実行し、2 つ目は EC2 Image Builder 用のインスタンスプロファイルを実行し、3 つ目は EventBridge ルールを実行します。
- 2 つの EventBridge ルール

Terraform モジュール構造

ソースコードについては、GitHub リポジトリの「Terraform EC2 Image Builder コンテナ強化パイプライン」を参照してください。

```
### components.tf
### config.tf
### dist-config.tf
### files
#   ###assumption-policy.json
### hardening-pipeline.tfvars
### image.tf
### infr-config.tf
### infra-network-config.tf
### kms-key.tf
### main.tf
### outputs.tf
### pipeline.tf
### recipes.tf
### roles.tf
### sec-groups.tf
### trigger-build.tf
### variables.tf
```

モジュールの詳細

- `components.tf` には、`/files` ディレクトリのコンテンツをアップロードするための Amazon S3 アップロードリソースが含まれています。カスタムコンポーネントの YAML ファイルをモジュールとして追加することもできます。
- `/files` には、`components.tf` で使用されるコンポーネントを定義する `.yml` ファイルが含まれています。
- `image.tf` には、基本的なイメージオペレーティングシステムの定義が含まれています。ここで、別のベースイメージパイプラインの定義を変更できます。
- `infr-config.tf`と `dist-config.tf` には、イメージの起動と配信に必要な最小限の AWS インフラストラクチャのリソースが含まれています。

- `infra-network-config.tf` には、コンテナイメージをデプロイするための最小限の VPC インフラストラクチャが含まれています。
- `hardening-pipeline.tfvars` の適用時に使用される Terraform 変数が含まれています。
- `pipeline.tf` は、Terraform で EC2 Image Builder パイプラインを作成して管理します。
- `recipes.tf` では、さまざまなコンポーネントの組み合わせを指定してコンテナレシピを作成できます。
- `roles.tf` には、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイルとパイプラインデプロイロールの AWS Identity and Access Management (IAM) ポリシー定義が含まれています。
- `trigger-build.tf` には、EventBridge ルールと SQS キューリソースが含まれています。

ターゲット アーキテクチャ

この図は、次のワークフローを示しています。

1. EC2 Image Builder は、定義されたレシピを使用してコンテナイメージを構築します。これにより、オペレーティングシステムの更新がインストールされ、RHEL Medium STIG が Amazon Linux 2 ベースイメージに適用されます。
2. 強化されたイメージはプライベート Amazon ECR レジストリに公開されます。このイメージが正常に公開されると、EventBridge ルールは SQS キューにメッセージを送信します。
3. Amazon Inspector が拡張スキャン用に設定されている場合、Amazon ECR レジストリをスキャンします。
4. Amazon Inspector がイメージの重大度が「非常事態」または「高」という検出結果を生成した場合、EventBridge ルールによって EC2 Image Builder パイプラインが再び実行され、新たに強化されたイメージが公開されます。

自動化とスケール

- このパターンでは、インフラストラクチャをプロビジョニングしてコンピュータにパイプラインを構築する方法を説明しています。　ただし、これは大規模な構築を目的としています。Terraform モジュールをローカルにデプロイする代わりに、Account Factory for Terraform 環境を備えた AWS Control Tower などのマルチアカウント環境で使用できます。その場合は、設定状態をローカルで管理するのではなく、バックエンド状態の S3 バケットを使用して Terraform 状態ファイルを管理する必要があります。

- スケーラブルに使用する場合は、Control Tower またはランディングゾーン アカウントモデルから、共有サービスアカウントや共通サービスアカウントなどのある中央アカウントにソリューションをデプロイし、Amazon ECR リポジトリと AWS KMS キーにアクセスする権限をコンシューマーアカウントに付与します。セットアップの詳細については、re: Post の記事「Amazon ECR イメージリポジトリ内のイメージをセカンダリアカウントにプッシュまたはプルさせるにはどうすればよいですか?」を参照してください。たとえば、アカウント自動販売機または Account Factory for Terraform では、各アカウントベースラインまたはアカウントカスタマイズベースラインにアクセス権限を追加して、その Amazon ECR リポジトリと暗号化キーへのアクセスを提供します。
- コンテナイメージパイプラインがデプロイされたら、コンポーネント などの EC2 Image Builder 機能を使用してパイプラインを変更できます。これにより、より多くのコンポーネントを Docker ビルドにパッケージ化できます。
- コンテナイメージの暗号化に使用される AWS KMS キーは、イメージを使用するアカウント間で共有する必要があります。
- Terraform モジュール全体をコピーし、次の `recipes.tf` 属性を変更することで、他のイメージのサポートを追加できます：
  - 別の画像タイプを `parent_image = "amazonlinux:latest"` に変更します。
  - 既存の Amazon ECR リポジトリを指定するように `repository_name` を変更します。これにより、別の親イメージタイプを既存の Amazon ECR リポジトリにデプロイするパイプラインがもう 1 つ作成されます。

ツール

ツール

- テラフォーム (IaC プロビジョニング)
- Git (ローカルでプロビジョニングする場合)
- AWS CLI バージョン 1 またはバージョン 2 (ローカルにプロビジョニングする場合)

コード

ソースコードについては、GitHub リポジトリの「Terraform EC2 Image Builderコンテナ強化パイプライン」を参照してください。次のセクションの指示に従って、サンプルコードを使用します。

エピック

インフラストラクチャをプロビジョニングする

タスク	説明	必要なスキル

ローカルの認証情報を設定します。	AWS の一時認証情報を設定します。
	
	1. AWS CLI がインストールされているか確認します。
	
	``
	$ aws --version
	aws-cli/1.16.249
	 Python/3.6.8...
	``
	
	• AWS CLI バージョンが 1.1 以降である必要があります。
	• コマンドが見つからない場合は、AWS CLI をインストールします。
	
	2. `aws configure` を実行して、次の値を指定します。
	
	``
	$ aws configure
	AWS Access Key ID
	 [************xxxx
	]: <Your AWS access
	 key ID>
	AWS Secret Access Key
	 [************xxx
	x]: <Your AWS secret
	 access key>
	Default region name:
	 [us-east-1]: <Your
	 desired Region for
	 deployment>
	``
		AWS DevOps

タスク	説明	必要なスキル
	`Default output format [None]: <Your desired output format>`	

リポジトリをクローン作成します。	1. このパターンで提供されるリポジトリをクローンします。HTTPS またはSecure Shell (SSH) を使用できます。
	
	HTTPS:
	
	`git clone https://github.com/aws-samples/terraform-ec2-image-builder-container-hardening-pipeline`
	
	SSH:
	
	`git clone git@github.com:aws-samples/terraform-ec2-image-builder-container-hardening-pipeline.git`
	2. このソリューションが格納されているローカルディレクトリに移動します。
	
	`cd terraform-ec2-image-builder-container-hardening-pipeline`
		AWS DevOps

タスク	説明	必要なスキル
変数を更新します。	環境と必要な構成に合わせて、hardening-pipeline.tfvars ファイル内の変数を更新します。自分で account_id を用意する必要があります。ただし、残りの変数も必要なデプロイに合わせて変更する必要があります。すべての変数が必須です。 account_id = "<DEPLOYMENT-ACCOUNT-ID>" aws_region = "us-east-1" vpc_name = "example-hardening-pipeline-vpc" kms_key_alias = "image-builder-container-key" ec2_iam_role_name = "example-hardening-instance-role" hardening_pipeline_role_name = "example-hardening-pipeline-role" aws_s3_ami_resources_bucket = "example-hardening-ami-resources-bucket-0123" image_name = "example-hardening-al2-container-image" ecr_name = "example-hardening-container-repo"	AWS DevOps

タスク	説明	必要なスキル

	`recipe_version = "1.0.0"`
	`ebs_root_vol_size = 10`
	
	以下に、各変数の説明を示します。
	
	• `account_id` – ソリューションをデプロイする AWS アカウント番号。
	• `aws_region` – ソリューションをデプロイする AWS リージョン。
	• `vpc_name` – VPC インフラストラクチャの名前。
	• `kms_key_alias` – EC2 Image Builder インフラストラクチャ設定で使用される AWS KMS キー名。
	• `ec2_iam_role_name` – EC2 インスタンスプロファイルとして使用されるロールの名前。
	• `hardening_pipeline_role_name` – 強化パイプラインをデプロイするために使用されるロールの名前。
	• `aws_s3_ami_resources_bucket` – パイプラインとコンテナイメージの構築に必要なすべてのファイルをホストする S3 バケットの名前。	

タスク	説明	必要なスキル
	• `image_name` – コンテナイメージの名前。この値は 3 ~ 50 文字で、英数字とハイフンのみを含む必要があります。 • `ecr_name` – コンテナイメージを保存する Amazon ECR レジストリの名前。 • `recipe_version` – イメージ レシピのバージョン。デフォルト値は 1.0.0 です。 • `ebs_root_vol_size` – Amazon Elastic Block Store (Amazon EBS) のルートボリュームのサイズ (ギガバイト単位)。デフォルト値は 10 ギガバイトです。	
Terraform を初期化します。	変数値を更新した後、Terraform 設定ディレクトリを初期化できます。設定ディレクトリを初期化すると、設定で定義されている AWS プロバイダーがダウンロードおよびインストールされます。 `terraform init` Terraform が正常に初期化され、インストールされたプロバイダのバージョンを示すメッセージが表示されます。	AWS DevOps

タスク	説明	必要なスキル
インフラストラクチャをデプロイし、コンテナイメージを作成します。	以下の `.tfvars` コマンドを使用して、ファイルに定義されている変数を使用して Terraform モジュールを初期化、検証し、環境に適用します。 `terraform init && terraform validate && terraform apply -var-file *.tfvars -auto-approve`	AWS DevOps
コンテナをカスタマイズします。	EC2 Image Builder がパイプラインと初期レシピをデプロイした後、コンテナレシピの新しいバージョンを作成できます。 EC2 Image Builder にある 31 種類以上のコンポーネントのいずれかを追加して、コンテナビルドをカスタマイズできます。詳細については、EC2 Image Builder ドキュメントの「コンテナレシピの新しいバージョンを作成する」の「コンポーネント」セクションを参照してください。	AWS 管理者

リソースを検証する

タスク	説明	必要なスキル

AWS インフラストラクチャのプロビジョニングを検証します。	最初の Terraform apply コマンドが正常に完了した後、ローカルで設定を行っている場合は、ローカルコンピュータのターミナルに次のスニペットが表示されます：
	`Apply complete!`
	`Resources: 43 added, 0`
	`changed, 0 destroyed.`
		AWS DevOps

個々の AWS インフラストラクチャリソースを検証します。	ローカルでプロビジョニングする場合、デプロイされた個々のリソースを検証するために、次のコマンドを実行します。 `terraform state list` このコマンドは、43 個のリソースのリストを返します。	AWS DevOps

リソースを削除する

タスク	説明	必要なスキル

インフラストラクチャとコンテナイメージを削除します。	Terraform の設定が完了した後、次のコマンドを実行してリソースを削除できます。
	`terraform init &&`
	`terraform validate &&`
	`terraform destroy -var-`
		AWS DevOps

タスク	説明	必要なスキル
	`file *.tfvars -auto-app rove`	

トラブルシューティング

問題	ソリューション
プロバイダー認証情報の検証中にエラーが発生しました。	ローカルマシンから Terraform `apply` または `destroy` コマンドを実行すると、次のようなエラーが発生する場合があります。 `Error: configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: operation error STS: GetCallerIdentity, https response error StatusCode: 403, RequestID: 123456a9-fbc1-40ed-b8d8-513d0133ba7f, api error InvalidClientTokenId: The security token included in the request is invalid.` このエラーは、ローカルマシンの設定で使用されている認証情報のセキュリティトークンの有効期限が切れていることが原因です。 このエラーを解決するには、AWS CLI ドキュメントの「設定の設定と表示」を参照してください。

関連リソース

- 「Terraform EC2 Image Builder コンテナ強化パイプライン」
- EC2 Image Builder ドキュメント
- 「AWS Control Tower Account Factory for Terraform」 (AWS ブログ記事)

- 「バックエンドステート S3 バケット」 (Terraform ドキュメント)
- 「AWS CLI の最新バージョンをインストールまたは更新する」 (AWS CLI ドキュメント)
- 「Terraform をダウンロード」

Terraform を使用して AWS Organizations の IAM アクセスキー管理を一元化する

作成者: Aarti Rajput (AWS)、Chintamani Aphale (AWS)、T.V.R.L.Phani Kumar Dadi (AWS)、Pradip kumar Pandey (AWS)、Mayuri Shinde (AWS)、Pratap Kumar Nanda (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら

キーとパスワードのセキュリティルールの適用は、すべての組織にとって 不可欠なタスクです。重要なルールの 1 つは、セキュリティを強化するために AWS Identity and Access Management (IAM) キーを定期的にローテーションすることです。通常、AWS アクセスキーは、チームが AWS コマンドラインインターフェイス (AWS CLI) または AWS 外のアプリケーションから AWS にアクセスするたびにローカルで作成および設定されます。組織全体で強力なセキュリティを維持するには、要件が満たされた後、または定期的に、古いセキュリティキーを変更または削除する必要があります。組織内の複数のアカウントにわたるキーローテーションを管理するプロセスは、時間がかかり、面倒です。このパターンは、Account Factory for Terraform (AFT) と AWS のサービスを使用してローテーションプロセスを自動化するのに役立ちます。

このパターンには以下の利点があります。

- アクセスキー IDsとシークレットアクセスキーを、組織内のすべてのアカウントで一元管理します。
- `AWS_ACCESS_KEY_ID` および `AWS_SECRET_ACCESS_KEY`環境変数を自動的にローテーションします。
- ユーザー認証情報が侵害された場合は、更新を強制します。

このパターンでは、Terraform を使用して AWS Lambda 関数、Amazon EventBridge ルール、IAM ロールをデプロイします。EventBridge ルールは定期的に実行され、作成された日時に基づいてすべてのユーザーアクセスキーを一覧表示する Lambda 関数を呼び出します。追加の Lambda 関数は、前のキーが定義したローテーション期間 (45 日間など) より古い場合、新しいアクセスキー ID とシークレットアクセスキーを作成し、Amazon Simple Notification Service (Amazon SNS) と Amazon Simple Email Service (Amazon SES) を使用してセキュリティ管理者に通知します。シークレットはそのユーザーの AWS Secrets Manager で作成され、古いシークレットアクセスキーは Secrets

Manager に保存され、古いキーにアクセスするためのアクセス許可が設定されます。古いアクセスキーが使用されないようにするには、非アクティブ期間 (例: 60 日、この例ではキーがローテーションされてから 15 日) 後に無効になります。非アクティブなバッファ期間 (例: この例ではキーがローテーションされてから 90 日、45 日) 後、古いアクセスキーは AWS Secrets Manager から削除されます。アーキテクチャとワークフローの詳細については、「アーキテクチャ」セクションを参照してください。

前提条件と制限

- AWS Control Tower (バージョン 3.1 以降) を使用して構築された組織のランディングゾーン
- Account Factory for Terraform (AFT) は 3 つのアカウントで設定されています。
  - 組織管理アカウントは、組織全体を一元管理します。
  - AFT 管理アカウントは Terraform パイプラインをホストし、インフラストラクチャをデプロイアカウントにデプロイします。
  - デプロイアカウントは、この完全なソリューションをデプロイし、IAM キーを一元管理します。
- デプロイアカウントにインフラストラクチャをプロビジョニングするための Terraform バージョン 0.15.0 以降。
- Amazon Simple Email Service (Amazon SES) で設定された E メールアドレス。
- (推奨) セキュリティを強化するには、このソリューションを Virtual Private Cloud (VPC) 内のプライベートサブネット (デプロイアカウント) 内にデプロイします。変数をカスタマイズするときに、VPC とサブネットの詳細を指定できます (「エピック」セクションの「コードパイプラインのパラメータをカスタマイズする」を参照してください)。 ???

アーキテクチャ

AFT リポジトリ

このパターンでは、Account Factory for Terraform (AFT) を使用して、必要なすべての AWS リソースを作成し、コードパイプラインを使用してデプロイアカウントにリソースをデプロイします。コードパイプラインは 2 つのリポジトリで実行されます。

- グローバルカスタマイズには、AFT に登録されているすべてのアカウントで実行される Terraform コードが含まれています。
- アカウントのカスタマイズには、デプロイアカウントで実行される Terraform コードが含まれます。

リソースの詳細

AWS CodePipeline ジョブは、デプロイアカウントに次のリソースを作成します。

- AWS EventBridge ルールと設定済みルール
- `account-inventory` Lambda 関数
- `IAM-access-key-rotation` Lambda 関数
- `Notification` Lambda 関数
- E メールテンプレートを含む Amazon Simple Storage Service (Amazon S3) バケット
- 必要な IAM ポリシー

アーキテクチャ

この図表は、以下を示すものです：

1. EventBridge ルールは、24 時間ごとに `account-inventory` Lambda 関数を呼び出します。
2. `account-inventory` Lambda 関数は、AWS Organizations にすべての AWS アカウント IDs、アカウント名、およびアカウント E メールのリストをクエリします。
3. `account-inventory`Lambda 関数は、AWS アカウントごとに `IAM-access-key-auto-rotation` Lambda 関数を開始し、メタデータを渡して追加の処理を行います。
4. `IAM-access-key-auto-rotation` Lambda 関数は、引き受けた IAM ロールを使用して AWS アカウントにアクセスします。Lambda スクリプトは、アカウント内のすべてのユーザーとそのユーザーの IAM アクセスキーに対して監査を実行します。
5. IAM キーローテーションのしきい値 (ローテーション期間) は、`IAM-access-key-auto-rotation`Lambda 関数がデプロイされるときに環境変数として設定されます。ローテーション期間が変更されると、`IAM-access-key-auto-rotation`Lambda 関数は更新された環境変数で再デプロイされます。パラメータを設定して、ローテーション期間、古いキーの非アクティブ期間、および古いキーが削除された後の非アクティブバッファを設定できます (「エピック」セクションの「コードパイプラインのパラメータをカスタマイズする」を参照してください)。 ???
6. `IAM-access-key-auto-rotation` Lambda 関数は、設定に基づいてアクセスキーの経過時間を検証します。IAM アクセスキーの経過時間が定義したローテーション期間を超えていない場合、Lambda 関数はそれ以上アクションを実行しません。
7. IAM アクセスキーの有効期間が定義したローテーション期間を超えた場合、`IAM-access-key-auto-rotation`Lambda 関数は新しいキーを作成し、既存のキーをローテーションします。

8. Lambda 関数は、古いキーを Secrets Manager に保存し、アクセスキーがセキュリティ標準から逸脱したユーザーにアクセス許可を制限します。Lambda 関数は、指定された IAM プリンシパルのみがシークレットにアクセスして取得できるようにするリソースベースのポリシーも作成します。
9. `IAM-access-key-rotation` Lambda 関数は Lambda `Notification` 関数を呼び出します。
10. `Notification` Lambda 関数は S3 バケットに E メールテンプレートをクエリし、関連するアクティビティメタデータを含む E メールメッセージを動的に生成します。
11. `Notification` Lambda 関数は、Amazon SES を呼び出してさらにアクションを実行します。
12. Amazon SES は、アカウント所有者の E メールアドレスに関連情報を含む E メールを送信します。

ツール

AWS サービス

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。このパターンには IAM ロールとアクセス許可が必要です。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Secrets Manager は、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得する上で役立ちます。
- 「Amazon Simple Email Service (Amazon SES)」はユーザー自身のメールアドレスとドメインを使用してメールを送受信する上で役立ちます。

その他のツール

- 「Terraform」は、HashiCorpのinfrastructure as code (IaC) ツールで、クラウドとオンプレミスのリソースの作成と管理を支援します。

コードリポジトリ

このパターンの手順とコードは、GitHub IAM アクセスキーローテーションリポジトリで入手できます。AWS Control Tower の中央デプロイアカウントにコードをデプロイして、中央の場所からのキーローテーションを管理できます。

ベストプラクティス

- IAM については、IAM ドキュメントの「セキュリティのベストプラクティス」を参照してください。
- キーローテーションについては、IAM ドキュメントの「アクセスキーの更新に関するガイドライン」を参照してください。

エピック

ソースファイルのセットアップ

タスク	説明	必要なスキル

リポジトリをクローン作成します。	1. IAM アクセスキーローテーション GitHub リポジトリのクローンを作成します。
	`$ git clone https://github.com/aws-samples/centralized-iam-key-management-aws-organizations-terraform.git`
	2. リポジトリのローカルコピーに 3 つのフォルダが含まれていることを確認します。
	`$ cd Iam-Access-keys-Rotation`
	`$ ls`
	`org-account-customization`
		DevOps エンジニア

タスク	説明	必要なスキル
	`global-account-customization account-customization`	

アカウントの設定

タスク	説明	必要なスキル

ブートストラップアカウントを設定します。	AFT ブートストラッププロセスの一環として、ローカルマシン`aft-bootstrap` にというフォルダが必要です。
	1. すべての Terraform ファイルをローカル GitHub org-account-customization フォルダから `aft-bootstrap` フォルダに手動でコピーします。
	2. Terraform コマンドを実行して、AWS Control Tower 管理アカウントでグローバルクロスアカウントロールを設定します。
	`$ cd aft-bootstrap`
	`$ terraform init`
	`$ terraform apply –auto-approve`
		DevOps エンジニア

グローバルカスタマイズを設定します。	AFT フォルダの設定の一環として、ローカルマシン`aft-global-customi`	DevOps エンジニア

タスク	説明	必要なスキル

	zations に というフォルダが必要です。
	
	1. すべての Terraform ファイルをローカル GitHub global-account-customization フォルダから aft-global-customizations/terraform フォルダに手動でコピーします。
	2. コードを AWS CodeCommit にプッシュします。
	
	``
	$ git add *
	$ git commit -m
	 "message"
	$ git push
	``	

タスク	説明	必要なスキル

アカウントのカスタマイズを設定します。	AFT フォルダの設定の一環として、ローカルマシンaft-account-customizations に というフォルダがあります。
	1. 販売者アカウント番号でフォルダを作成します。
	2. すべての Terraform ファイルをローカル GitHub アカウントカスタマイズフォルダから aft-account-customizations/<vended account>/terraform フォルダに手動でコピーします。
	3. コードを AWS CodeCommit にプッシュします。
	`$ git add *`
	`$ git commit -m "message"`
	`$ git push`
		DevOps エンジニア

コードパイプラインのパラメータをカスタマイズする

タスク	説明	必要なスキル
すべてのアカウントの非 Terraform コードパイプラインパラメータをカスタマイズします。	aft-global-customizations/terraform/ フォルダinput.auto.tfvars に という名前のファイルを作成し、必要な入力データを指定しま	DevOps エンジニア

タスク	説明	必要なスキル
	す。デフォルト値については、GitHub リポジトリの ファイルを参照してください。	

タスク	説明	必要なスキル
デプロイアカウントのコードパイプラインパラメータをカスタマイズします。	`aft-account-customizations/<AccountName>/terraform/` フォルダに `input.auto.tfvars` という名前のファイルを作成し、コードを AWS CodeCommit にプッシュします。AWS CodeCommit にコードをプッシュすると、コードパイプラインが自動的に開始されます。 以下を含む、組織の要件に基づいてパラメータの値を指定します (デフォルト値については、Github リポジトリの ファイルを参照してください)。 • `s3_bucket_name` – E メールテンプレートの一意のバケット名。 • `s3_bucket_prefix` – S3 バケット内のフォルダ名。 • `admin_email_address` – 通知を受け取る管理者の E メールアドレス。 • `org_list_account` – 管理アカウントのアカウント番号。 • `rotation_period` – キーをアクティブから非アクティブにローテーションする日数。	DevOps エンジニア

タスク	説明	必要なスキル
	• inactive_period – ローテーションされたキーを非アクティブ化する日数。この値は、 の値より大きい必要がありますrotation_period 。 • inactive_buffer – キーのローテーションから非アクティブ化までの猶予期間。 • recovery_grace_period – 非アクティブ化からキーの削除までの猶予期間。 • dry_run_flag – キーをローテーションせずにテスト目的で管理者に通知を送信する場合は、true に設定します。 • store_secrets_in_central_account – シークレットをデプロイアカウントに保存する場合は、true に設定します。変数が false (デフォルト) に設定されている場合、シークレットはメンバーアカウントに保存されます。 • credential_replication_region – E メールテンプレートの Lambda 関数と S3 バケットをデプロイする AWS リージョン。	

タスク	説明	必要なスキル
	• `run_lambda_in_vpc` – VPC 内で Lambda 関数を実行するには、true に設定します。 • `vpc_id` – VPC 内で Lambda 関数を実行する場合の、デプロイアカウントの VPC ID。 • `vpc_cidr` – デプロイアカウントの CIDR 範囲。 • `subnet_id` – デプロイアカウントのサブネット IDs。 • `create_smtp_endpoint` – E メールエンドポイントを有効にする場合は true に設定します。	

キーローテーションを検証する

タスク	説明	必要なスキル
ソリューションを更新する	1. AWS マネジメントコンソールから、デプロイアカウントにサインインします。 2. IAM コンソールを開き、ユーザー認証情報 (アクセスキー IDsとシークレットキー) が指定されたとおりにローテーションされているかどうかを確認します。	DevOps エンジニア

タスク	説明	必要なスキル
	3. IAM キーがローテーションされたら、以下を確認します。 • 古い値は AWS Secrets Manager に保存されます。 • シークレット名は の形式ですAccount_<account ID>_User_<username>_AccessKey 。 • admin_email_address パラメータで指定したユーザーは、キーローテーションに関する E メール通知を受け取ります。	

ソリューションを拡張する

タスク	説明	必要なスキル
E メール通知日をカスタマイズします。	アクセスキーを無効にする前に特定の日に E メール通知を送信する場合は、これらの変更で IAM-access-key-auto-rotation Lambda 関数を更新できます。 1. という変数を定義しますnotify-period 。 2. キーを非アクティブ化main.pyする前に、 にif条件を追加します。	DevOps エンジニア

タスク	説明	必要なスキル
	`If (keyage>rotation-period-notify-period){ send_to_notifier(context, aws_account_id, account_name, resource_owner, resource_actions[resource_owner], dryrun, config.emailTemplateAudit) }`	

トラブルシューティング

問題	ソリューション
アカウントの一覧表示`AccessDenied` 中に `account-inventory` Lambda ジョブが で失敗します。	この問題が発生した場合は、アクセス許可を検証する必要があります。 1. 新しく提供されたアカウントにログインし、Amazon CloudWatch コンソールを開き、CloudWatch ロググループ を表示します`/aws/lambda/account-inventory-lambda` 。 2. 最新の CloudWatch ログで、アクセス拒否の問題の原因となっているアカウント番号を特定します。 3. AWS Control Tower 管理アカウントにログインし、ロール`allow-list-account` が作成されたことを確認します。 4. ロールが存在しない場合は、 `terraform apply` コマンドを使用して Terraform コードを再実行します。

問題	ソリューション
	5. Trusted Account タブを選択し、同じアカウントが信頼されていることを確認します。

関連リソース

- Terraform の推奨プラクティス (Terraform ドキュメント)
- IAM でのセキュリティのベストプラクティス (IAM ドキュメント)
- キーローテーションのベストプラクティス (IAM ドキュメント)

Amazon CloudFront ディストリビューションのアクセスロギング、HTTPS、TLS バージョンを確認する

作成者: SaiJeevan Devireddy (AWS) と Bijesh Bal (AWS)

概要

このパターンでは、Amazon CloudFront ディストリビューションが HTTPS を使用していること、Transport Layer Security (TLS) バージョン 1.2 以降を使用していること、およびアクセスロギングが有効になっていることを確認します。CloudFront は、ユーザーへの静的および動的なウェブコンテンツ (.html、.css、.js、イメージファイルなど) の配信を高速化する Amazon Web Services (AWS) です。CloudFront では、エッジロケーションというデータセンターの世界的ネットワークを経由してコンテンツを配信します。CloudFront でサービスを提供しているコンテンツをユーザーがリクエストすると、リクエストはエッジロケーションにルーティングされ、レイテンシー (遅延時間) が最小になります。これにより、コンテンツは可能な限り最高のパフォーマンスで配信されます。

このパターンでは、Amazon CloudWatch Events が CloudFront API コール CreateDistribution、CreateDistributionWithTags、または UpdateDistribution を検出したときに開始される AWS Lambda 関数を示します。Lambda 関数のカスタムロジックは、AWS アカウントで作成または更新されたすべての CloudFront ディストリビューションを評価します。以下の違反を検出すると、Amazon Simple Notification Service (Amazon SNS) を使用して違反通知が送信されます。

- グローバルチェック:
  - カスタム証明書は TLS バージョン 1.2 を使用しません
  - ロギングの配信は無効化されています
- オリジンチェック:
  - オリジンは TLS バージョン 1.2 で構成されていません
  - オリジンとの通信は HTTPS 以外のプロトコルで許可されています
- 動作チェック:
  - デフォルト動作通信は HTTPS 以外のプロトコルで許可されます
  - カスタム動作通信は HTTPS 以外のプロトコルで許可されます

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 違反の通知を受信する E メールアドレス

機能制限

- このセキュリティコントロールでは、ディストリビューションが更新されていない限り、既存の Cloudfront ディストリビューションはチェックされません。
- CloudFront はグローバルサービスと見なされ、特定の AWS リージョンに結び付けられていません。ただし、Amazon CloudWatch Logs および AWS Cloudtrail API ロギングは、米国東部 (バージニア北部) リージョン (us-east-1) で行われます。そのため、この CloudFront のセキュリティコントロールは、us-east-1 でデプロイし、維持する必要があります。この単一のデプロイは、CloudFront のすべてのディストリビューションをモニタリングします。セキュリティコントロールを他の AWS リージョンにデプロイしないでください。(他のリージョンにデプロイすると、CloudWatch イベントと Lambda 関数の開始に失敗し、SNS 通知も送信されません)。
- このソリューションは、CloudFront ウェブコンテンツディストリビューションによる広範なテストを経ています。リアルタイムメッセージングプロトコル (RTMP) ストリーミングディストリビューションは対象外です。

アーキテクチャ

ターゲットテクノロジースタック

- Lambda function
- SNS トピック
- Amazon EventBridge ルール

ターゲットアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、「AWS CloudFormation StackSets」を使用して、モニタリングしたい複数のアカウントに添付のテンプレートをデプロイできます。

ツール

サービス

- AWS CloudFormation — CloudFormation は、インフラストラクチャをコードとして使用することでAWS リソースをモデル化および設定できるサービスです。
- Amazon EventBridge – EventBridge は、お客様独自のアプリケーション、Software as a Service (SaaS) アプリケーション、および AWS サービスからのリアルタイムデータをストリーム配信し、そのデータを Lambda などのターゲットにルーティングします。
- AWS Lambda – AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS — Amazon SNS は、ウェブサーバーや E メールアドレスを含む、パブリッシャーとクライアント間のメッセージの配信または送信を調整および管理するウェブサービスです。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

添付のコードには以下が含まれます。

- Lambda コード (index.py) を含む.zip ファイル
- Lambda コードをデプロイするために実行する CloudFormation テンプレート (.yml ファイル)

エピック

セキュリティコントロールをアップロードする

タスク	説明	必要なスキル
Lambda コード用の S3 バケットを作成します。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを作成します。S3 バケット名はグ	クラウドアーキテクト

タスク	説明	必要なスキル
	ローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケットは、Lambda コードをデプロイする予定のリージョンに存在する必要があります。	
S3 バケットに Lambda コードをアップロードします。	「添付ファイル」セクションで提供されている Lambda コード (cloudfront_ssl_log_lambda.zip ファイル) を、前のステップで作成した S3 バケットにアップロードします。	クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
CloudFormation テンプレートをデプロイします。	AWS CloudFormation コンソールで、S3 バケットと同じ AWS リージョンに、「添付ファイル」セクションで提供されている CloudFormation テンプレート (cloudfront-ssl-logging.yml) をデプロイします。	クラウドアーキテクト
S3 バケット名を指定します。	S3 バケット パラメータには、最初のエピックで作成した S3 バケットの名前を指定します。	クラウドアーキテクト

タスク	説明	必要なスキル
Lambda ファイルの Amazon S3 キー名を指定します。	S3 キーパラメータには、S3 バケット内の Lambda コード .zip ファイルの Amazon S3 の口ケーションを指定します。先頭にスラッシュを含めないでください (たとえば、lambda.zip や controls/lambda.zip と入力してください)。	クラウドアーキテクト
通知用 E メールアドレスを提供します。	通知メールパラメータには、違反通知を受け取るメールアドレスを指定します。	クラウドアーキテクト
口ギングのレベルを定義します。	Lambda 口ギングレベルのパラメータでは、Lambda 関数の口ギングレベルを定義します。次のいずれかの値を選択します。 • INFO を使用すると、アプリケーションの進行状況に関する詳細な情報メッセージを取得できます。 • ERROR を使用すると、アプリケーションの実行を継続できるエラーイベントに関する情報を取得できます。 • WARNING は、潜在的に有害な状況に関する情報を取得するためのものです。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、新しい SNS トピックが作成され、指定したメールアドレスにサブスクリプションメッセージが送信されます。違反通知を受信するには、この E メールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS CloudFormation に関する情報
- AWS CloudFormation コンソールでのスタックの作成 (CloudFormation ドキュメント)
- CloudFront ロギング (CloudFront のドキュメント)
- Amazon S3 に関する情報
- AWS Lambda に関する情報

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

IPv4 および IPv6 対応セキュリティグループのイングレスルール内の単一ホストネットワークエントリを確認する

作成者: SaiJeevan Devireddy (AWS)、Ganesh Kumar (AWS)、および John Reynolds (AWS)

概要

このパターンは、Amazon Web Services (AWS) のリソースが仕様を満たさない場合に通知するセキュリティコントロールを提供します。これにより、Internet Protocol バージョン 4 (IPv4) と IPv6 の両方のセキュリティグループソースアドレスフィールドのシングルホストネットワークエントリを検索する AWS Lambda 関数が提供されます。Lambda 関数は、Amazon CloudWatch Events が Amazon Elastic Compute Cloud (Amazon EC2) の AuthorizeSecurityGroupIngress API コールを検出したときに開始されます。Lambda 関数のカスタムロジックは、セキュリティグループイングレスルールの CIDR ブロックのサブネットマスクを評価します。サブネットマスクが /32 (IPv4) または /128 (IPv6) 以外であると判断された場合、Lambda 関数は Amazon Simple Notification Service (Amazon SNS) を使用して違反通知を送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 違反の通知を受信する E メールアドレス

機能制限

- このセキュリティモニタリングソリューションはリージョン限定であり、モニタリングする各 AWS リージョンごとにデプロイする必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- Lambda function
- SNS トピック
- Amazon EventBridge ルール

ターゲットアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、「AWS CloudFormation StackSets」を使用して、モニタリングしたい複数のアカウントにこのテンプレートをデプロイできます。

ツール

サービス

- AWS CloudFormation — AWS CloudFormation は、インフラストラクチャをコードとして使用することで AWS リソースのモデル化とセットアップを支援するサービスです。
- Amazon EventBridge は、お客様独自のアプリケーション、Software as a Service (SaaS) アプリケーション、AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを Lambda 関数などのターゲットにルーティングします。
- AWS Lambda は、サーバーをプロビジョニングまたは管理せずにコードを実行することをサポートします。
- Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- Amazon SNS は、パブリッシャーとクライアント間のメッセージ (ウェブサーバーや E メールアドレスを含む) の配信または送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

添付のコードには以下が含まれます。

- Lambda セキュリティコントロールコード (`index.py`) を含む .zip ファイル
- Lambda コードをデプロイするために実行する CloudFormation テンプレート (`security-control.yml` ファイル)

エピック

セキュリティコントロールをアップロードする

タスク	説明	必要なスキル
Lambda コード用の S3 バケットを作成します。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケットは、セキュリティグループのイングレスチェックをデプロイする AWS リージョンに存在している必要があります。	クラウドアーキテクト
S3 バケットに Lambda コードをアップロードします。	「添付ファイル」セクションで提供されている Lambda コード (`security-control-lambda.zip` ファイル) を、先の手順で作成した S3 バケットにアップロードします。	クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
Python バージョンを変更する。	「添付ファイル」セクションで提供されている CloudFormation テンプレート (`security-control.yml`) をダウンロードします。ファイルを開き、Lambda が	クラウドアーキテクト

タスク	説明	必要なスキル
	サポートする最新バージョン (現在は Python 3.9) を反映するように Python バージョンを変更します。 たとえば、コード内で `python` を検索し、`Runtime` の値を `python3.6` から `python3.9` に変更できます。 Python ランタイムバージョンのサポートに関する最新情報については、AWS Lambda ドキュメントを参照してください。	
AWS CloudFormation テンプレートをデプロイします。	AWS CloudFormation コンソールで、S3 バケットと同じ AWS リージョンに CloudFormation テンプレート (`security-control.yml`) をデプロイします。	クラウドアーキテクト
S3 バケット名を指定します。	S3 バケット パラメータには、最初のエピックで作成した S3 バケットの名前を指定します。	クラウドアーキテクト

タスク	説明	必要なスキル
Lambda ファイルの Amazon S3 キー名を指定します。	S3 キーパラメータには、S3 バケット内の Lambda コード .zip ファイルの Amazon S3 ロケーションを指定します。先頭にスラッシュを含めないでください (たとえば、`lambda.zip` または `controls/lambda.zip` と入力します)。	クラウドアーキテクト
通知用 E メールアドレスを提供します。	通知メールパラメータには、違反通知を受け取るメールアドレスを指定します。	クラウドアーキテクト
ロギングのレベルを定義します。	Lambda ロギングレベルのパラメータでは、Lambda 関数のロギングレベルを定義します。次のいずれかの値を選択します。 • INFO を使用すると、アプリケーションの進行状況に関する詳細な情報メッセージを取得できます。 • ERROR を使用すると、アプリケーションの実行を継続できるエラーイベントに関する情報を取得できます。 • WARNING は、潜在的に有害な状況に関する情報を取得するためのものです。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、新しい SNS トピックが作成され、指定したメールアドレスにサブスクリプションメッセージが送信されます。違反通知を受信するには、この E メールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS CloudFormationに関する情報
- AWS CloudFormation コンソールでのスタックの作成 (AWS CloudFormation ドキュメント)
- VPC 向けセキュリティグループ (Amazon VPC ドキュメント)
- Amazon S3 に関する情報
- AWS Lambda に関する情報

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

エンタープライズアプリケーションの Amazon Cognito 認証フローを選択する

作成者: Michael Daehnert (AWS)、Fabian Jahnke (AWS)

概要

Amazon Cognito は、ウェブおよびモバイルアプリケーションの認証、認可、ユーザー管理を提供します。フェデレーティッド ID の認証に有益な機能を提供します。これを稼働させるには、テクニカルアーキテクトがそれらの機能をどのように使用するかを決定する必要があります。

Amazon Cognito は、認証リクエストの複数のフローをサポートしています。これらのフローは、ユーザーが ID を検証する方法を定義します。使用する認証フローの決定は、アプリケーションの特定の要件によって異なり、複雑になる可能性があります。このパターンは、エンタープライズアプリケーションに最適な認証フローを決定するのに役立ちます。Amazon Cognito、OpenID Connect (OIDC)、およびフェデレーションに関する基本的な知識が既にあることを前提としており、さまざまなフェデレーション認証フローに関する詳細をガイドします。

このソリューションは、技術的な意思決定者を対象としています。さまざまな認証フローを理解し、アプリケーション要件にマッピングするのに役立ちます。テクニカルリードは、Amazon Cognito 統合を開始するために必要なインサイトを収集する必要があります。エンタープライズ組織は主に SAML フェデレーションに焦点を当てているため、このパターンには SAML フェデレーションを使用する Amazon Cognito ユーザープールの説明が含まれています。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Amazon Cognito へのフルアクセスを持つ AWS Identity and Access Management (IAM) ロールとアクセス許可
- （オプション) Microsoft Entra ID、Active Directory フェデレーションサービス (AD FS)、Okta などの ID プロバイダー (IdP) へのアクセス
- アプリケーションに関する高度な専門知識
- Amazon Cognito、OpenID Connect (OIDC)、フェデレーションに関する基本的な知識

機能制限

- このパターンは、Amazon Cognito ユーザープールと ID プロバイダーに焦点を当てています。Amazon Cognito ID プールの詳細については、「追加情報」セクションを参照してください。

アーキテクチャ

次の表は、認証フローの選択に役立ちます。各フローの詳細については、このセクションを参照してください。

machine-to-machine認証が必要ですか？	アプリケーションは、フロントエンドがサーバーでレンダリングされるウェブベースのアプリケーションですか？	アプリはシングルページアプリケーション (SPA) ですか、モバイルベースのフロントエンドアプリケーションですか？	アプリケーションには「サインインしたままにする」機能の更新トークンが必要ですか？	フロントエンドはブラウザベースのリダイレクトメカニズムを提供していますか？	推奨される Amazon Cognito フロー
はい	いいえ	いいえ	いいえ	いいえ	クライアント認証情報フロー
いいえ	あり	なし	はい	あり	認可コードフロー
いいえ	いいえ	はい	あり	あり	コード交換 (PKCE) の証明キーを使用した認可コードフロー
いいえ	いいえ	いいえ	いいえ	いいえ	リソース所有者パスワードフロー*

* リソース所有者パスワードフローは、絶対に必要な場合にのみ使用してください。詳細については、このパターンの「リソース所有者パスワード」フローセクションを参照してください。

クライアント認証情報フロー

クライアント認証情報フローは、Amazon Cognito フローの中で最も短いフローです。システムまたはサービスがユーザーとのやり取りなしで相互に通信する場合に使用します。リクエスト元のシステムは、クライアント ID とクライアントシークレットを使用してアクセストークンを取得します。どちらのシステムもユーザーとのやり取りなしで動作するため、追加の同意ステップは必要ありません。

この図表は、以下を示すものです：

1. アプリケーション 1 は、クライアント ID とクライアントシークレットを含む認証リクエストを Amazon Cognito エンドポイントに送信し、アクセストークンを取得します。
2. アプリケーション 1 は、アプリケーション 2 への後続の呼び出しごとにこのアクセストークンを使用します。
3. アプリケーション 2 は Amazon Cognito を使用してアクセストークンを検証します。

このフローを使用する必要があります。

- ユーザーとのやり取りのないアプリケーション間の通信の場合

このフローは使用しないでください。

- ユーザーとのやり取りが可能な通信の場合

認可コードフロー

認可コードフローは、従来のウェブベースの認証用です。このフローでは、バックエンドがすべてのトークン交換とストレージを処理します。ブラウザベースのクライアントには実際のトークンは表示されません。このソリューションは、.NET Core、Jakarta Faces、Jakarta Server Pages (JSP) などのフレームワークで記述されたアプリケーションに使用されます。

認可コードフローはリダイレクトベースのフローです。クライアントは、ウェブブラウザまたは同様のクライアントとやり取りできる必要があります。クライアントは認証サーバーにリダイレクトされ、このサーバーに対して認証されます。クライアントが正常に認証されると、サーバーにリダイレクトされます。

この図表は、以下を示すものです：

1. クライアントはウェブサーバーにリクエストを送信します。
2. ウェブサーバーは、HTTP 302 ステータスコードを使用してクライアントを Amazon Cognito にリダイレクトします。クライアントは、設定された IdP ログインへのこのリダイレクトを自動的に実行します。
3. IdP は、IdP 側で既存のブラウザセッションをチェックします。存在しない場合、ユーザーはユーザー名とパスワードを指定して認証を求めるプロンプトを受け取ります。IdP は Amazon Cognito に SAML トークンで応答します。
4. Amazon Cognito は、JSON ウェブトークン (JWT)、特にコードトークンで成功を返します。ウェブサーバーは /oauth2/token を呼び出して、コードトークンをアクセストークンと交換します。ウェブサーバーは、検証のためにクライアント ID とクライアントシークレットを Amazon Cognito に送信します。
5. アクセストークンは、他のアプリケーションへのそれ以降の呼び出しのたびに使用されます。
6. 他のアプリケーションは、Amazon Cognito でアクセストークンを検証します。

このフローを使用する必要があります。

- ユーザーがウェブブラウザまたはクライアントとやり取りできる場合。アプリケーションコードは、ブラウザにシークレットが公開されないように、サーバーで実行およびレンダリングされます。

このフローは使用しないでください。

- 単一ページアプリケーション (SPAs) またはモバイルアプリケーションの場合、これらはクライアントでレンダリングされるため、クライアントシークレットを使用しないでください。

PKCE を使用した認可コードフロー

単一ページのアプリケーションとモバイルアプリケーションには、コード交換の証明キー (PKCE) を使用した認可コードフローを使用する必要があります。これは暗黙的なフローの後継であり、PKCE を使用するため、より安全です。PKCE は、パブリッククライアント向けの OAuth 2.0 認可コード付与の拡張機能です。PKCE は、傍受された認可コードの引き換えから保護します。

この図表は、以下を示すものです：

1. アプリケーションはコード検証機能とコードチャレンジを作成します。これらは明確に定義された一意の値であり、今後の参照のために Amazon Cognito に送信されます。
2. アプリケーションは Amazon Cognito の /oauth2/authorization エンドポイントを呼び出します。これにより、設定された IdP ログインにユーザーが自動的にリダイレクトされます。
3. IdP は既存のセッションをチェックします。存在しない場合、ユーザーはユーザー名とパスワードを指定して認証を求めるプロンプトを受け取ります。IdP は Amazon Cognito に SAML トークンで応答します。
4. Amazon Cognito がコードトークンで成功を返した後、ウェブサーバーは /oauth2/token を呼び出してコードトークンをアクセストークンと交換します。
5. アクセストークンは、他のアプリケーションへのそれ以降の呼び出しのたびに使用されます。
6. 他のアプリケーションは、Amazon Cognito でアクセストークンを検証します。

このフローを使用する必要があります。

- SPAs またはモバイルアプリケーションの場合

このフローは使用しないでください。

- アプリケーションのバックエンドが認証を処理する場合

リソース所有者パスワードフロー

リソース所有者パスワードフローは、リダイレクト機能のないアプリケーションを対象としています。これは、独自のアプリケーションでログインフォームを作成することによって構築されます。ログインは、リダイレクトフローに依存するのではなく、CLI または SDK 呼び出しを介して Amazon Cognito でチェックされます。フェデレーションにはブラウザベースのリダイレクトが必要なため、この認証フローではフェデレーションを実行できません。

この図表は、以下を示すものです：

1. ユーザーは、アプリケーションが提供するログインフォームに認証情報を入力します。
2. AWS コマンドラインインターフェイス (AWS CLI) は、Amazon Cognito にadmin-initiated-auth呼び出しを行います。

> **Note**
>
> または、AWS CLI の代わりに AWS SDKs を使用することもできます。

3. Amazon Cognito はアクセストークンを返します。
4. アクセストークンは、他のアプリケーションへのそれ以降の呼び出しのたびに使用されます。
5. 他のアプリケーションは、Amazon Cognito でアクセストークンを検証します。

このフローを使用する必要があります。

- 保存された認証情報をアクセストークンに変換して、直接認証ロジック (基本アクセス認証やダイジェストアクセス認証など) を使用する既存のクライアントを OAuth に移行する場合

このフローは使用しないでください。

- フェデレーティッド ID を使用する場合
- アプリケーションがリダイレクトをサポートしている場合

ツール

AWS サービス

- Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、ユーザー管理を提供します。

その他のツール

- JSON ウェブトークン (JWT) デバッガーは、ウェブベースの JWT 検証ツールです。

エピック

アプリケーションを評価する

タスク	説明	必要なスキル
認証要件を定義します。	特定の認証要件に従ってアプリケーションを評価します。	アプリ開発者、アプリアーキテクト
要件を認証フローに合わせます。	アーキテクチャセクションで、各フローの決定テーブルと説明を使用して、Amazon Cognito 認証フローを選択します。	アプリ開発者、AWS 全般、アプリアーキテクト

Amazon Cognito ユーザープールを設定する

タスク	説明	必要なスキル
ユーザープールを作成します。	1. AWS マネジメントコンソールにサインインし、Amazon Cognito コンソールを開きます。 2. 新しい Cognito ユーザープールを作成します。手順については、Amazon Cognito ユーザープール」を参照してください。 3. 必要に応じてユーザープールの設定と属性を更新します。たとえば、ユーザープールのパスワードポリシーを設定します。アプリクライアントはまだ作成しないでください。	AWS 全般

タスク	説明	必要なスキル
(オプション) ID プロバイダーを設定します。	1. Amazon Cognito ユーザープールに SAML ID プロバイダーを作成します。手順については、「ユーザープールでの SAML ID プロバイダーの追加と管理」を参照してください。 2. Amazon Cognito ユーザープールのフェデレーションと連携するように、サードパーティーの SAML ID プロバイダーを設定します。詳細については、「サードパーティーの SAML ID プロバイダーの設定」を参照してください。AD FS を使用している場合は、Amazon Cognito ユーザープールを使用したウェブアプリの AD FS フェデレーションの構築」(AWS ブログ記事) を参照してください。	AWS 全般、フェデレーション管理者

タスク	説明	必要なスキル
アプリケーションクライアントを作成します。	1. ユーザープールのアプリケーションクライアントを作成します。手順については、「アプリケーションクライアントの作成」を参照してください。次の点に注意してください: • トークンの有効期限など、必要に応じて設定を変更します。 • 認証フローでクライアントシークレットが必要ない場合は、クライアントシークレットの生成チェックボックスをオフにします。 2. アプリクライアント設定を選択して、統合をユーザープールログイン (ユーザー名とパスワード) または SAML ベースの IdP 経由のフェデレーションログインに変更します。 3. URLsし、必要に応じて OAuth フローまたはスコープを定義して、IdP を有効にします。	AWS 全般

アプリケーションを Amazon Cognito と統合する

タスク	説明	必要なスキル
Amazon Cognito 統合の詳細を交換します。	認証フローに応じて、ユーザープール ID やアプリケーションクライアント ID などの Amazon Cognito 情報をアプリケーションと共有します。	アプリ開発者、AWS 全般
Amazon Cognito 認証を実装します。	これは、選択した認証フロー、プログラミング言語、使用しているフレームワークによって異なります。開始するためのリンクについては、関連リソースセクションを参照してください。	アプリ開発者

関連リソース

AWS ドキュメント

- ユーザープール認証フロー
- JSON ウェブトークンの検証
- Amazon Cognito ID プールを使用して ASP.NET://https://https://https://https://https://
- フレームワークと SDKs:
  - Amazon Amplify 認証
  - Amazon Cognito ID プロバイダーの例 (AWS SDK for Java 2.x ドキュメント)
  - Amazon Cognito によるユーザーの認証 (AWS SDK for .NET ドキュメント)

ブログの投稿

- Cookie を使用した Authorization@Edge: 認証されていないユーザーが Amazon CloudFront コンテンツをダウンロードしないように保護する
- Amazon Cognito ユーザープールを使用したウェブアプリの AD FS フェデレーションの構築

実装パートナー

- 認証ソリューションの AWS パートナー

追加情報

よくある質問

暗黙的なフローが廃止されるのはなぜですか？

OAuth 2.1 フレームワークのリリース以降、暗黙的なフローはセキュリティ上の理由から非推奨としてマークされています。別の方法として、アーキテクチャセクションで説明されている PKCE で認可コードフローを使用してください。

Amazon Cognito で必要な機能が提供されていない場合はどうなりますか？

AWS パートナーは、認証および認可ソリューションにさまざまな統合を提供します。詳細については、「認証ソリューションの AWS パートナー」を参照してください。

Amazon Cognito ID プールフローについてはどうですか？

Amazon Cognito ユーザープールとフェデレーティッド ID は認証用です。Amazon Cognito ID プールは、一時的な AWS 認証情報をリクエストして AWS リソースへのアクセスを承認するために使用されます。ID プールの ID トークンとアクセストークンの交換については、このパターンでは説明しません。詳細については、Amazon Cognito ユーザープールと ID プールの違いは何ですか？」および「一般的な Amazon Cognito シナリオ」を参照してください。 Amazon Cognito

次のステップ

このパターンは、Amazon Cognito 認証フローの概要を提供します。次のステップとして、アプリケーションのプログラミング言語の詳細な実装を選択する必要があります。複数の言語SDKs とフレームワークを提供しており、Amazon Cognito で使用できます。役立つリファレンスについては、「関連リソース」セクションを参照してください。

AWS CloudFormation Guard ポリシーを使用して AWS Config カスタムルールを作成する

作成者: Andrew Lok (AWS)、Kailash Havildar (AWS)、Nicole brown (AWS)、Tanya Howell (AWS)

概要

AWS Config ルールは、 AWS リソースとそのターゲット設定状態を評価するのに役立ちます。ルールには、マネージドルールとカスタム AWS Config ルールの 2 種類があります。は、 AWS Lambda 関数またはpolicy-as-code言語である AWS CloudFormation Guard (GitHub) を使用してカスタムルールを作成できます。

Guard で作成されたルールは、 マネージドルールよりも詳細な制御を提供し、通常、完全にカスタムの Lambda ルールよりも設定が簡単になります。このアプローチにより、エンジニアやアーキテクトは、Lambda を介してカスタムルールをデプロイするために必要な Python、NodeJS、または Java を知ることなくルールを構築できます。

このパターンは、Guard でカスタムルールを採用するのに役立つ実用的なテンプレート、コードサンプル、デプロイアプローチを提供します。このパターンを使用すると、管理者は を使用して AWS Config 、設定項目属性を持つカスタムコンプライアンスルールを構築できます。たとえば、開発者は AWS Config 設定項目に対して Guard ポリシーを使用して、デプロイされたAWS リソース AWS と非リソースの状態を継続的にモニタリングし、ルール違反を検出して、修復を自動的に開始できます。

目的

このパターンを読み終えると、次のことができるようになります。

- Guard ポリシーコードが AWS Config サービスとやり取りする方法を理解します。
- シナリオ 1 をデプロイします。シナリオ 1 は、Guard 構文を使用して暗号化されたボリュームのコンプライアンスを検証する AWS Config カスタムルールです。このルールは、ドライブが使用中であることを確認し、ドライブタイプが gp3 であることを確認します。
- シナリオ 2 をデプロイします。シナリオ 2 は、Guard 構文を使用して Amazon GuardDuty のコンプライアンスを検証する AWS Config カスタムルールです。このルールは、GuardDuty レコーダーで Amazon Simple Storage Service (Amazon S3) Protection と Amazon Elastic Kubernetes Service (Amazon EKS) Protection が有効になっていることを確認します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Config、 でセットアップする AWS アカウント

制約事項

- ガードカスタムルールは、ターゲット設定項目の JSON レコード内のキーと値のペアのみをクエリできます。

アーキテクチャ

Guard 構文をカスタム policy として AWS Config ルールに適用します。 AWS Config は、指定された各リソースの階層 JSON を取得します。 AWS Config 設定項目の JSON には、キーと値のペアが含まれます。これらの属性は、対応する値に割り当てられた変数として Guard 構文で使用されます。

以下は、Guard 構文の説明です。設定項目 JSON の変数が使用され、%文字が付加されます。

```
# declare variable
let <variable name> = <'value'>

# create rule and assign condition and policy
    rule <rule name> when
        <CI json key> == <"CI json value"> {
            <top level CI json key>.<next level CI json key> == %<variable name>
        }
```

シナリオ 1: Amazon EBS ボリューム

シナリオ 1 は、Guard 構文を使用して暗号化されたボリュームのコンプライアンスを検証する AWS Config カスタムルールをデプロイします。このルールは、ドライブが使用中であることを確認し、ドライブタイプが gp3 であることを確認します。

シナリオ 1 AWS Config の設定項目の例を次に示します。この設定項目には、、、volumestatus、および の変数として を使用した 3 つのキーvolumeencryptionstatusと値のペアがありますvolumetype。また、resourceTypeキーは Guard ポリシーのフィルターとして使用されます。

```
{
  "version": "1.3",
  "accountId": "111111111111",
  "configurationItemCaptureTime": "2023-01-15T19:04:45.402Z",
  "configurationItemStatus": "ResourceDiscovered",
  "configurationStateId": "4444444444444",
  "configurationItemMD5Hash": "",
  "arn": "arn:aws:ec2:us-west-2:111111111111:volume/vol-222222222222",
  "resourceType": "AWS::EC2::Volume",
  "resourceId": "vol-222222222222",
  "awsRegion": "us-west-2",
  "availabilityZone": "us-west-2b",
  "resourceCreationTime": "2023-01-15T19:03:22.247Z",
  "tags": {},
  "relatedEvents": [],
  "relationships": [
    {
      "resourceType": "AWS::EC2::Instance",
      "resourceId": "i-33333333333333333",
      "relationshipName": "Is attached to Instance"
    }
  ],
  "configuration": {
    "attachments": [
      {
        "attachTime": "2023-01-15T19:03:22.000Z",
        "device": "/dev/xvda",
        "instanceId": "i-33333333333333333",
        "state": "attached",
        "volumeId": "vol-222222222222",
        "deleteOnTermination": true,
        "associatedResource": null,
        "instanceOwningService": null
      }
    ],
    "availabilityZone": "us-west-2b",
    "createTime": "2023-01-15T19:03:22.247Z",
    "encrypted": false,
    "kmsKeyId": null,
    "outpostArn": null,
    "size": 8,
    "snapshotId": "snap-55555555555555555",
    "state": "in-use",
```

```
    "volumeId": "vol-222222222222",
    "iops": 100,
    "tags": [],
    "volumeType": "gp2",
    "fastRestored": null,
    "multiAttachEnabled": false,
    "throughput": null,
    "sseType": null
  },
  "supplementaryConfiguration": {}
}
```

以下は、シナリオ 1 で Guard 構文を使用して変数とルールを定義する例です。以下の例で、次の操作を行います。

- 最初の 3 行は、 `let` コマンドを使用して変数を定義します。 設定項目の属性から派生した名前と値が割り当てられます。
- `compliancecheck` ルールブロックは、 に一致する`resourceType`キーと値のペアを検索する条件依存関係に を追加します`AWS::EC2::Volume`。一致が見つかった場合、ルールは残りの JSON 属性を続行し`encrypted`、`state`、、 の 3 つの条件で一致を検索します`volumeType`。

```
let volumestatus = 'available'
let volumetype = 'gp3'
let volumeencryptionstatus = true

    rule compliancecheck when
        resourceType == "AWS::EC2::Volume" {
            configuration.state == %volumestatus
            configuration.encrypted == %volumeencryptionstatus
            configuration.volumeType == %volumetype
        }
```

このカスタムルールを実装する完全な Guard カスタムポリシーについては、GitHub コードリポジトリの[awsconfig-guard-cft.yaml]」または[awsconfig-guard-tf-ec2vol.json]」を参照してください。このカスタムポリシーを Guard にデプロイする HashiCorp Terraform コードについては、コードリポジトリの[awsconfig-guard-tf-example.json]」を参照してください。

シナリオ 2: GuardDuty コンプライアンス

シナリオ 2 は、Guard 構文を使用して Amazon GuardDuty のコンプライアンスを検証する AWS Config カスタムルールをデプロイします。このルールは、GuardDuty レコーダーで Amazon S3 Protection と Amazon EKS Protection が有効になっていることを確認します。また、GuardDuty の検出結果が 15 分ごとに発行されることを確認します。このシナリオは、組織全体 AWS アカウントおよび組織内 (内) AWS リージョン にデプロイできます AWS Organizations。

以下は、シナリオ 2 AWS Config の設定項目の例です。この設定項目には、ガードポリシーで変数として を使用した `FindingPublishingFrequency`、、 の 3 つのキー`S3Logs`と値のペアがありますKubernetes。また、`resourceType`キーはポリシーのフィルターとして使用されます。

```
{
  "version": "1.3",
  "accountId": "111111111111",
  "configurationItemCaptureTime": "2023-11-27T13:34:28.888Z",
  "configurationItemStatus": "OK",
  "configurationStateId": "7777777777777",
  "configurationItemMD5Hash": "",
  "arn": "arn:aws:guardduty:us-
west-2:111111111111:detector/66666666666666666666666666666666",
  "resourceType": "AWS::GuardDuty::Detector",
  "resourceId": "66666666666666666666666666666666",
  "resourceName": "66666666666666666666666666666666",
  "awsRegion": "us-west-2",
  "availabilityZone": "Regional",
  "resourceCreationTime": "2020-02-17T02:48:04.511Z",
  "tags": {},
  "relatedEvents": [],
  "relationships": [],
  "configuration": {
    "Enable": true,
    "FindingPublishingFrequency": "FIFTEEN_MINUTES",
    "DataSources": {
      "S3Logs": {
        "Enable": true
      },
      "Kubernetes": {
        "AuditLogs": {
          "Enable": true
        }
      }
    },

    "Id": "66666666666666666666666666666666",
```

```
    "Tags": []
  },
  "supplementaryConfiguration": {
    "CreatedAt": "2020-02-17T02:48:04.511Z"
  }
}
```

以下は、シナリオ 2 で Guard 構文を使用して変数とルールを定義する例です。以下の例で、次の操作を行います。

- 最初の 3 行は、 `let` コマンドを使用して変数を定義します。 設定項目の属性から派生した名前と値が割り当てられます。
- `compliancecheck` ルールブロックは、 に一致する`resourceType`キーと値のペアを検索する条件依存関係に を追加します`AWS::GuardDuty::Detector`。一致が見つかった場合、ルールは残りの JSON 属性を続行し`Kubernetes.AuditLogs.Enable`、`S3Logs.Enable`、、および の 3 つの条件で一致を検索します`FindingPublishingFrequency`。

```
let s3protection = true
let kubernetesprotection = true
let publishfrequency = 'FIFTEEN_MINUTES'

    rule compliancecheck when
        resourceType == "AWS::GuardDuty::Detector" {
            configuration.DataSources.S3Logs.Enable == %s3protection
            configuration.DataSources.Kubernetes.AuditLogs.Enable ==
 %kubernetesprotection
            configuration.FindingPublishingFrequency == %publishfrequency
        }
```

このカスタムルールを実装する完全な Guard カスタムポリシーについては、GitHub コードリポジトリの[awsconfig-guard-cft-gd.yaml]」を参照してください。このカスタムポリシーを Guard にデプロイする HashiCorp Terraform コードについては、コードリポジトリの[awsconfig-guard-tf-gd.json]」を参照してください。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。
- AWS Config は、 内のリソースの詳細ビュー AWS アカウント と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。

コードリポジトリ

このパターンのコードは、GitHub AWS Config with AWS CloudFormation Guard リポジトリで入手できます。このコードリポジトリには、このパターンで説明されている両方のシナリオのサンプルが含まれています。

エピック

AWS Config カスタムルールの作成

タスク	説明	必要なスキル
(オプション) ルールのキーと値のペアを選択します。	カスタム Guard ポリシーを定義する場合は、以下の手順を実行します。シナリオ 1 または 2 のサンプルポリシーのいずれかを使用している場合は、以下の手順をスキップします。 1. にサインイン AWS Management Console し、AWS Config コンソールを開きます。	AWS 管理者、セキュリティエンジニア

タスク	説明	必要なスキル
	2. 左側のナビゲーションで、リソースを選択します。 3. リソースインベントリで、AWS Config カスタムルールを作成するリソースのタイプを選択します。 4. [詳細を表示] を選択します。 5. 設定項目の表示 (JSON) を選択します。このセクションでは、JSON 形式で設定項目を表示するように展開します。 6. AWS Config カスタムルールを構築するキーと値のペアを特定します。	
カスタムルールを作成します。	以前に特定したキーと値のペアを使用するか、提供されているサンプル Guard ポリシーのいずれかを使用して、AWS Config「カスタムポリシールールの作成」の手順に従ってカスタムルールを作成します。	AWS 管理者、セキュリティエンジニア

タスク	説明	必要なスキル
カスタムルールを検証します。	カスタム Guard ルールを検証するには、次のいずれかを実行します。 • AWS Command Line Interface () に次のコマンドを入力しますAWS CLI。 `cfn-guard validate -r guard-s3.guard -d s3bucket-prod-pass.json` • Detective モードのAWS Config「ルールを使用したリソースの評価」の指示に従って、ルールをデプロイします AWS Config。Guard 構文がターゲットアカウントまたはファイルの対応するリソースと正しく一致することを確認します。	AWS 管理者、セキュリティエンジニア

トラブルシューティング

問題	ソリューション
の外部で Guard ポリシーをテストする AWS Config	ユニットテストは、ローカルデバイスまたは IDE などの統合開発環境 (IDE) AWS Cloud9 で実行できます。ユニットテストを実行するには、以下を実行します。 1. AWS CloudFormation Guard CLI とその依存関係をインストールします。

問題	ソリューション

	2. JSON 形式の CI サンプルを .json ファイルとしてワークステーションに保存します。
	3. GuardDuty ポリシーを .guard ファイルとしてワークステーションに保存します。
	4. Guard CLI で、次のコマンドを入力して、Guard ポリシーを使用してサンプル JSON ファイルを検証します。
	`cfn-guard validate \`
	`-r guard-s3.guard \`
	`-d s3bucket-prod-pass.json`

AWS Config カスタムルールのデバッグ	ガードポリシーで、`EnableDebugLogDelivery` 値を に変更します`true`。デフォルト値は `false` です。ログメッセージは Amazon CloudWatch に保存されます。

関連リソース

AWS ドキュメント

- AWS Config カスタムポリシールールの作成 (AWS Config ドキュメント)
- AWS CloudFormation Guard ルールの記述 (ガードドキュメント)

AWS ブログ投稿とワークショップ

- Introducing AWS CloudFormation Guard 2.0 (AWS ブログ記事)

その他のリソース

- AWS CloudFormation Guard (GitHub)
- AWS CloudFormation Guard CLI ドキュメント (GitHub)

複数の から Prowler セキュリティ検出結果の統合レポートを作成する AWS アカウント

作成者: Mike Virgilio (AWS)、Andrea Di Fabio (AWS)、Jay Durga (AWS)

概要

「Prowler」(GitHub) はオープンソースのコマンドラインツールで、Amazon Web Services (AWS) アカウントがセキュリティのベストプラクティスを順守しているかどうかを評価、監査、監視するのに役立ちます。このパターンでは、組織 AWS アカウント 内の一元化された に Prowler をデプロイし、 によって管理され AWS Organizations、次に Prowler を使用して組織内のすべてのアカウントのセキュリティ評価を実行します。

Prowler をデプロイして評価に使用する方法は多くありますが、このソリューションは迅速な導入、組織内のすべてのアカウントまたは定義済みのターゲットアカウントの完全な分析及びセキュリティ検出結果レポートに容易にアクセスできるように設計されています。このソリューションでは、Prowler が組織内のすべてのアカウントのセキュリティ評価を完了すると、結果を統合します。また、Prowler がプロビジョニングされたアカウントの Amazon Simple Storage Service (Amazon S3) バケットをスキャンすることを妨げている制限に関連するエラーなど、予想されるエラーメッセージはすべて除外されます AWS Control Tower。除外された統合結果は、このパターンに含まれている Microsoft Excel テンプレートで報告されます。このレポートを使用して、組織のセキュリティ管理の潜在的な改善点を特定できます。

このソリューションは、以下を念頭に置いて設計されました。

- AWS CloudFormation テンプレートは、このパターンで AWS リソースをデプロイするために必要な労力を削減します。
- デプロイ時に CloudFormation テンプレートと [prowler_scan.sh] スクリプトのパラメーターを見直して、環境に合わせてテンプレートをカスタマイズできます。
- Prowler の評価とレポート速度は、並列処理 AWS アカウント、集計結果、推奨される修復による統合レポート、および自動的に生成された視覚化によって最適化されます。
- ユーザーはスキャンの進行状況をモニタリングする必要がありません。評価が完了すると、ユーザーはレポートを受信できるように Amazon Simple Notification Service (Amazon SNS) トピックを通じて通知を受信します。
- レポートテンプレートを使用すると、組織全体に関連する結果のみを読んで評価できます。

前提条件と制限

前提条件

- で組織のメンバーアカウントとして管理されるセキュリティサービスとツールをホスト AWS アカウント するための AWS Organizations。このパターンでは、このアカウントはセキュリティアカウントと呼ばれます。
- このセキュリティアカウントには、アウトバウンドインターネットアクセスを可能にするプライベート・サブネットが必要です。手順については、「Amazon Virtual Private Cloud (Amazon VPC) ドキュメント」の「プライベートサブネットにサーバーを持つ VPC と NAT」を参照してください。パブリックサブネットにプロビジョニングされた「NAT ゲートウェイ」を使用してインターネットアクセスを確立できます。
- CloudFormation の管理者権限を委任された AWS Organizations 管理アカウントまたはアカウントへのアクセス。手順については、「CloudFormation のドキュメント」の「委任管理者の登録」を参照してください。
- AWS Organizations と CloudFormation 間の信頼されたアクセスを有効にします。手順については、「CloudFormation ドキュメント」の「 信頼できるアクセスを有効にする AWS Organizations」を参照してください。

制約事項

- ターゲットは の組織として管理 AWS アカウント する必要があります AWS Organizations。を使用していない場合は AWS Organizations、環境の IAM-ProwlerExecRole.yaml CloudFormation テンプレートと prowler_scan.sh スクリプトを更新できます。代わりに、スクリプトを実行する AWS アカウント IDs とリージョンのリストを指定します。
- CloudFormation テンプレートは、アウトバウンドインターネットアクセスのあるプライベートサブネットに Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをデプロイするように設計されています。 AWS Systems Manager エージェント (SSM エージェント) は、 AWS Systems Manager サービスエンドポイントに到達するためにアウトバウンドアクセスを必要とし、コードリポジトリのクローンを作成し、依存関係をインストールするためにアウトバウンドアクセスが必要です。パブリックサブネットを使用する場合は、 [prowler-resources.yaml] テンプレートを変更して「Elastic IP アドレス」を EC2 インスタンスに関連付ける必要があります。

製品バージョン

- Prowler バージョン 4.0 以降

アーキテクチャ

図表に示す内容は以下のステップです。

1. の一機能である Session Manager を使用して AWS Systems Manager、ユーザーは EC2 インスタンスを認証し、prowler_scan.sh スクリプトを実行します。このシェルスクリプトはステップ 2 ~ 8 を実行します。
2. EC2 インスタンスは IAM ロールを担当します。`ProwlerEC2Role` IAM ロールは S3 バケットにアクセスし、組織内の他のアカウントの `ProwlerExecRole` IAM ロールを担当する権限を付与します。
3. EC2 インスタンスは、組織の管理アカウントで `ProwlerExecRole` IAM ロールを引き受けるため、組織内のアカウントのリストが生成されます。
4. EC2 インスタンスは、組織のメンバーアカウント (アーキテクチャ図ではワークロードアカウントという) の `ProwlerExecRole` IAM ロールを担当し、各アカウントのセキュリティ評価を実行します。検出結果として、EC2 インスタンスに CSV と HTML ファイルとして保存されます。

   > **Note**
   >
   > HTML ファイルは Prowler 評価の出力です。HTML の性質上、このパターンでは連結、処理または直接使用されることはありません。ただし、これらは個別のアカウントレポートレビューには役立つ場合があります。

5. EC2 インスタンスはすべての CSV ファイルを処理して既知の予想される エラーを削除し、残りの検出結果を一つの CSV ファイルに統合します。
6. EC2 インスタンスは、個々のアカウント結果と集計結果を zip ファイルにパッケージ化します。
7. EC2 インスタンスは zip ファイルを S3 バケットにアップロードします。
8. EventBridge ルールはファイルのアップロードを検出し、Amazon SNS トピックを使用して、評価が完了したことを通知する E メールをユーザーに送信します。
9. ユーザーは S3 バケットから zip ファイルをダウンロードします。ユーザーは結果を Excel テンプレートにインポートし、その結果を確認します。

ツール

AWS のサービス

- Amazon Elastic Compute Cloud (Amazon EC2) は、 AWS クラウドでスケーラブルなコンピューティング容量を提供します。必要な数の仮想サーバーを起動することができ、迅速にスケールアップまたはスケールダウンができます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、他の のイベントバスなどです AWS アカウント。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータの保存、保護、取得に役立つクラウドベースのオブジェクトストレージサービスです。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理を簡素化し、運用上の問題を検出して解決する時間を短縮し、大規模な AWS リソースを安全に管理できるようにします。このパターンは、Systems Manager の機能である Session Manager を使用します。

その他のツール

- Prowler は、 AWS セキュリティのベストプラクティスやその他のセキュリティフレームワークや標準に準拠しているかどうかをアカウントで評価、監査、モニタリングするのに役立つオープンソースのコマンドラインツールです。

コードリポジトリ

このパターンのコードは、GitHub 内の「Prowler によるマルチアカウントのセキュリティ評価」リポジトリで利用できます。コードリポジトリには、以下のファイルが含まれます。

- prowler_scan.sh – この bash スクリプトは AWS アカウント、複数の Prowler セキュリティ評価を並行して開始するために使用されます。Prowler-resources.yaml CloudFormationTemplate で定義されているように、このスクリプトは EC2 インスタンスの `usr/local/prowler` フォルダに自動的にデプロイされます。
- prowler-resources.yaml — この CloudFormation テンプレートを使用して、組織のセキュリティアカウントにスタックを作成します。このテンプレートは、ソリューションをサポート

するためにこのアカウントに必要なリソースをすべてデプロイします。このスタックは IAM-ProwleExecrole.yaml テンプレートの前にデプロイする必要があります。重要な本番環境のワークロードをホストするアカウントにこれらのリソースをデプロイすることはお勧めしません。

> **Note**
>
> このスタックを削除して再デプロイする場合は、IAM ロール間のクロスアカウント依存関係を再構築するために、`ProwlerExecRole`スタックセットを再構築する必要があります。

- IAM-prowlerexecrole.yaml — この CloudFormation テンプレートを使用して、管理アカウントを含む組織内のすべてのアカウントに `ProwlerExecRole` IAM ロールをデプロイするスタックセットを作成します。
- prowler-report-template.xlsm – この Excel テンプレートを使用して Prowler の検出結果を処理します。レポート内のピボットテーブルには、検索機能、グラフと統合した検出結果が表示されます。

エピック

デプロイの準備

タスク	説明	必要なスキル
コードリポジトリを複製します。	1. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 2. 次のコマンドを入力します。 `git clone https://github.com/aws-samples/multi-account-security-assessment-via-prowler.git`	AWS DevOps

タスク	説明	必要なスキル
テンプレートを確認します。	1. クローンされたリポジトリで、Prowler-Resources.yaml ファイルと IAM-ProwlerExecRole.yaml ファイルを開きます。 2. これらのテンプレートによって作成されたリソースを確認し、環境に合わせて必要に応じてテンプレートを調整してください。詳細については、CloudFormation ドキュメントの「テンプレートの使用」を参照してください。 3. Prowler-Resources.yaml ファイルと IAM-ProwlerExecRole.yaml ファイルを保存して閉じます。	AWS DevOps

CloudFormation スタックの作成

タスク	説明	必要なスキル
セキュリティアカウントにリソースをプロビジョニングします。	prowler-resources.yaml テンプレートを使用して、セキュリティアカウントに必要なすべてのリソースをデプロイする CloudFormation スタックを作成します。手順については、「CloudFormation ドキュメント」の「スタックの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。	AWS DevOps

タスク	説明	必要なスキル
	1. テンプレートを指定ページでテンプレートは準備完了を選択して、 [prowler-resources.yaml] ファイルをアップロードします。 2. [スタック詳細の指定] ページで、[スタック名] に `Prowler-Resources` と入力します。 3. [パラメータ] セクションで、次の値を入力します。 • `VPCId` — アカウント内の VPC を選択します。 • `SubnetId` — インターネットにアクセスできるプライベートサブネットを選択します。 注：パブリックサブネットを選択すると、CloudFormation テンプレートはデフォルトで Elastic IP アドレスをプロビジョニングしてアタッチしないため、EC2 インスタンスにはパブリック IP アドレスが割り当てられません。 • `InstanceType` — パラレル評価数に基づいてインスタンスサイズを選択します。	

タスク	説明	必要なスキル
	• 10 の場合は、r6i.large を選択します。 • 12 の場合は、r6i.xlarge を選択します。 • 14 ~ 18 の場合は、r6i.2xlarge を選択してください。 • InstanceImageId — Amazon Linux はデフォルトのままにします。 • KeyPairName — SSH を使用してアクセスする場合は、既存のキーペア名を指定します。 • PermittedSSHInbound — SSH を使用してアクセスする場合は、許可された CIDR ブロックを指定します。SSH を使用していない場合は、127.0.0.1 のデフォルト値のままにします。 • BucketName — デフォルト値は prowler-output-<accountID>-<region> です。これは必要に応じて変更できます。カスタム値を指定すると、アカウント ID と国/地域が指	

タスク	説明	必要なスキル
	定した値に自動的に追加されます。 • `EmailAddress` — Prowler が評価を完了し、.zip ファイルを S3 バケットにアップロードしたとき、Amazon SNS 通知のメールアドレスを指定します。 注：Prowler が評価を完了する前に SNS サブスクリプションの設定を確認する必要がありますが、確認しないと、通知は送信されません。 • `IAMProwlerEC2Role` — 命名規則によりこの IAM ロールに別の名前を付ける必要がある場合以外、デフォルトのままにしてください。 • `IAMProwlerExecRole` — IAM-ProwleExecrole .yaml ファイルのデプロイ時に別の名前を付けない限り、デフォルトのままにします。 • `Parallelism` — 実行するパラレル評価数を指定します。`InstanceType` パラメータ値がこのパラレル評価数をサ	

タスク	説明	必要なスキル
	ポートしていることを確認してください。 • `FindingOutput` — 合格結果を除外したい場合は、`FailOnly` を選択してください。これにより、出力サイズが大幅に削減され、解決が必要なチェックに重点を置きます。合格結果を含めたい場合は、`FailAndPass` を選択してください。 4. 「レビュー」ページで、「次のリソースには機能が必要：[AWS:: IAM:: Role]」を選択して、「Create Stack」を選択します。 5. スタックが正常に作成されたら、CloudFormation コンソールの出力タブで、「`ProwlerEC2Role` Amazonリソースネーム (ARN)」をコピーします。この ARN は、後で IAM-Prowlerexecrole.yaml ファイルをデプロイするときに使用します。	

タスク	説明	必要なスキル
メンバーアカウントで IAM ロールをプロビジョニングします。	AWS Organizations 管理アカウントまたは CloudFormation の委任管理者権限を持つアカウントで、IAM-ProwlerExecRole.yaml テンプレートを使用して CloudFormation スタックセットを作成します。スタックセットは、組織内のすべてのメンバーアカウントに `ProwlerExecRole` IAM ロールをデポロイします。手順については、「CloudFormation のドキュメント」の「サービスマネージド型のアクセス許可を持つスタックセットの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。 1. Prepare template（テンプレートを準備します）で Template is ready (テンプレートの準備ができています)を選択し、IAM-ProwleExecrole.yaml ファイルをアップロードします。 2. [StackSet の詳細を指定] ページで、スタックセットに `IAM-ProwlerExecRole` という名前を付けます。	AWS DevOps

タスク	説明	必要なスキル
	3. [パラメータ] セクションで、次の値を入力します。 • AuthorizedARN — Prowler-Resources スタックの作成時にコピーした ProwlerEC 2Role ARN を入力します。 • ProwlerExecRoleNam e — Prowler-resources. yaml ProwlerEx ecRole ファイルのデプロイ時に別の名前が付けられていない限り、デフォルト値のままにします。 4. [アクセス権限] で、[Service-managed permissions (サービスマネージド型のアクセス許可)] を選択します。 5. [デプロイオプションの設定] ページの [デプロイターゲット] で、[組織へのデプロイ] を選択し、すべてのデ フォルトを受け入れます。 注：スタックをすべてのメンバーアカウントに同時にデプロイする場合は、最大同時アカウント数と耐障害性を、100 などの高い値に設定します。	

タスク	説明	必要なスキル
	6. デプロイリージョン で、Prowler の EC2 インスタンス AWS リージョン がデプロイされている を選択します。IAM リソースはグローバルであり、リージョナルではないため、IAM ロールはすべてのアクティブなリージョンにデプロイされます。 7. レビューページで、 がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを承認し、StackSet の作成を選択します。 8. スタックインスタンスタブ (個々のアカウントのステータス) と操作タブ (全体的なステータス) をモニタリングして、デプロイが完了したかを判断します。	

タスク	説明	必要なスキル
管理アカウントに IAM ロールをプロビジョニングします。	IAM-prowlerexecrole.yaml テンプレートを使用して、組織の管理アカウントに `ProwlerExecRole` IAM ロールをデプロイする CloudFormation スタックを作成します。以前に作成したスタックセットでは、管理アカウントに IAM ロールはデプロイされません。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。このテンプレートを展開する際には、次の点に注意してください。 1. テンプレートを指定ページで Template is ready (テンプレートの準備ができています)を選択し、IAM-ProwleExecrole.yaml ファイルをアップロードします。 2. [スタック詳細の指定] ページで、[スタック名] に `IAM-ProwlerExecRole` と入力します。 3. [パラメータ] セクションで、次の値を入力します。 • `AuthorizedARN` — `Prowler-Resources` スタックの作成時にコピーした `ProwlerEC`	AWS DevOps

タスク	説明	必要なスキル
	2Role ARN を入力します。 • ProwlerExecRoleName — [Prowler-resources.yaml] ファイルのデプロイ時に別の名前が付けられていない限り、デフォルト値のままにします。 4. レビューページで、次のリソースには機能が必要: AWS:: IAM:: Role を選択し、スタックの作成を選択します。	

Prowlerのセキュリティ評価を実行します。

タスク	説明	必要なスキル
スキャンを実行します。	1. 組織のセキュリティアカウントにサインインします。 2. 「Session Manager」を使用して、以前にプロビジョニングした Prowler の EC2 インスタンスに接続します。手順については、「Session Manager を使用して Linux インスタンスに接続」を参照してください。接続できない場合は、このパターンの「トラブルシューティング」セクションを参照してください。	AWS 管理者

タスク	説明	必要なスキル

	3. prowler_scan.sh ファイルに移動して開きます。
	4. お使いの環境に応じて、このスクリプト内の調整可能なパラメータと変数を確認して変更します。カスタマイズオプションの詳細については、スクリプトの冒頭におけるコメントを参照してください。
	
	たとえば、管理アカウントから組織内のすべてのメンバーアカウントのリストを取得する代わりに、スキャン AWS リージョン する AWS アカウント IDs または を指定するようにスクリプトを変更したり、これらのパラメータを含む外部ファイルを参照したりできます。
	5. prowler_scan.sh ファイルを保存して閉じます。
	6. 以下のコマンドを入力します。これにより、prowler_scan.sh スクリプトを実行します。
	
	`sudo -i`
	`screen`
	`cd /usr/local/prowler`
	`./prowler_scan.sh`	

タスク	説明	必要なスキル
	次の点に注意してください: • スクリーンコマンドは、接続がタイムアウトになり、またはコンソールにアクセスできなくなる場合でも、スクリプトの実行を継続できるようにします。 • スキャン開始後、Ctrl+A D を押すことで画面を強制的に切り離すことができます。画面が切り離されると、インスタンス接続を閉じて評価を続行できます。 • デタッチしたセッションを再開するには、インスタンスに接続し、「`sudo -i` Enter `screen -r`」キーを押します。 • 個々のアカウント評価の進捗状況をモニタリングするには、`usr/local/prowler` ディレクトリに移動してコマンドを入力します `tail -f output/stdout-<account-id>` 。 7. Prowler がすべてのアカウントのスキャンを完了するまでお待ちください。スクリプトは、複数のアカウントを同時に評価します	

タスク	説明	必要なスキル
	。すべてのアカウントで評価が完了すると、Prowler-Resources.yaml ファイルをデプロイしたときにメールアドレスを指定した場合は、通知が送信されます。	
Prowler の調査結果を取得します。	1. `prowler-output-<assessDate>.zip` バケットから `prowler-output-<accountID>-<region>` ファイルをダウンロードします。手順については、「Amazon S3 ドキュメント」の「オブジェクトのダウンロード」を参照してください。 2. ダウンロードしたファイルを含め、バケット内のすべてのオブジェクトを削除します。これはコストダウンのベストプラクティスであり、`Prowler-Resources` CloudFormation スタックをいつでも削除できるようにします。手順については、「Amazon S3 ドキュメント」の「オブジェクトの削除」を参照してください。	AWS 全般

タスク	説明	必要なスキル
EC2 インスタンスを停止します。	インスタンスがアイドル状態のときに課金されないようにするには、Prowler を実行している EC2 インスタンスを停止します。手順については、「Amazon EC2 ドキュメント」の「インスタンスの停止と起動」を参照してください。	AWS DevOps

検出結果のレポートを作成します。

タスク	説明	必要なスキル
検出結果をインポートします。	1. Excel で prowler-report-template.xlsx ファイルを開き、Prowler CSV ワークシートを選択します。 2. ヘッダー行を含むすべてのサンプルデータを削除します。削除するデータに関連するクエリを削除するかどうかを確認するメッセージが表示されると、[いいえ] を選択します。 クエリを削除すると、Excel テンプレートのピボットテーブルの機能に影響する可能性があります。 3. S3 バケットからダウンロードした zip ファイルの内容を抽出します。	AWS 全般

タスク	説明	必要なスキル
	4. Excel で prowler-fullorgresults-accessdeniedfiltered.txt を開きます。 AWS Control Tower リソースのスキャンの試行に関連する エラーなど、最も一般的な実行不可能な`Access Denied`エラーが既に削除されているため、このファイルを使用することをお勧めします。検出結果を除外しないでほしい場合は、代わりに prowler-fullorgresults.txt ファイルを開いてください。 5. A 列を選択します。 6. Windows を使用している場合は、Ctrl+C を押します。MacOS を使用している場合はCmd+C を押します。これにより、すべてのデータがクリップボードにコピーされます。 7. Excel レポートテンプレートの Prowler CSV ワークシートで、セル A1 を選択します。 8. Windows を使用している場合は Ctrl+V を押し、MacOS を使用している場合は Cmd+V を押します。これにより、調査結果がレポートにペーストされます。	

タスク	説明	必要なスキル
	9. ペーストされたデータを含むすべてのセルが選択されていることを確認します。選択されていない場合は、列 A を選択します。 10.データタブでテキストから列へを選択します。 11.ウィザードで、以下を実行します。 • ステップ 1 では、区切りを選択します。 • ステップ 2 では、区切り文字でセミコロンを選択します。データプレビューペインで、データが列に分割されていることを確認します。 • ステップ 3 では、完了を選択します。 12.テキストデータが複数の列にまたがって区切られていることを確認します。 13Excel レポートに新しい名前を付けて保存します。 14結果内の `Access Denied` エラーを検索して削除します。これらをプログラムで削除する方法については、「追加情報」セクションのプログラムによるエラーの削除を参照してください。	

タスク	説明	必要なスキル
レポートを完成します。	1. 検出結果ワークシートを選択して、セル A17 を選択します。このセルはピボットテーブルのヘッダーです。 2. リボンのピボットテーブルツールで分析を選択し、更新のすべて更新を選択します。これにより、ピボットテーブルが新しいデータセットで更新されます。 3. デフォルトでは、Excel は AWS アカウント 数字を正しく表示しません。数値フォーマットを修正する方法は次のとおりです。 • 結果ワークシートで、列 A のコンテキスト (右クリック) メニューを開き、セルのフォーマットを選択します。 • [数値] を選択し、小数点以下の桁数で `0` を入力します。 • [OK] を選択します。 注: AWS アカウント 数値が 1 つ以上のゼロで始まる場合、Excel は自動的にゼロを削除します。レポートに 12 桁未満の口座番号が表示される場合、番号の先頭で欠落している数字はゼロです。	AWS 全般

タスク	説明	必要なスキル
	4. (オプション) フィールドを折りたたむと、結果が見やすくなります。以下の操作を実行します。 • 検出結果ワークシートでは、18 行目と 19 行目の間の行 (重要なヘッダーと最初の結果の間のスペース) にカーソルを移動すると、カーソルアイコンが下向きの小さな矢印に変わります。 • すべての検索フィールドをクリックして選択します。 • コンテキスト (右クリック) メニューを開き、展開/折りたたみを選択してから、折りたたみを選択します。 5. 評価の詳細については、検出結果、重要度、合格不合格の各ワークシートを参照してください。	

(オプション) Prowler またはコードリポジトリ内のリソースを更新します。

タスク	説明	必要なスキル
Prowler をアップデートします。	Prowler を最新バージョンに更新するには、次の操作を実行します。	AWS 全般

タスク	説明	必要なスキル

	1. Prowlerのため「Session Manager」を使用して EC2 インスタンスに接続します。手順については、「Session Manager を使用して Linux インスタンスに接続」を参照してください。
	2. 次のコマンドを入力します。
	`sudo -i`
	`pip3 install --upgrade prowler`	

タスク	説明	必要なスキル

[prowler_scan.sh] スクリプトを更新してください。	prowler_scan.sh スクリプトをリポジトリ内で最新バージョンに更新するには、次の操作を実行します。
	1. 「Session Manager」を使用して Prowler のために EC2 インスタンスに接続します。手順については、「Session Manager を使用して Linux インスタンスに接続」を参照してください。
	2. 次のコマンドを入力します。
	`sudo -i`
	3. Prowler スクリプトディレクトリに移動します。
	`cd /usr/local/prowler`
	4. 次のコマンドを入力してローカルスクリプトを非表示にし、カスタム変更を最新バージョンにマージできるようにします。
	`git stash`
	5. スクリプトの最新バージョンを取得するには、以下のコマンドを入力します。
		AWS 全般

タスク	説明	必要なスキル

	`git pull`
	6. 次のコマンドを入力して、カスタムスクリプトを最新バージョンのスクリプトと結合します。
	`git stash pop`
	Note
	検出結果レポートなど、GitHub リポジトリにないローカルに生成されたファイルに関連する警告が表示される場合があります。prowler_scan.sh によってローカルで非表示になっている変更がマージされる場合に限ります。	

(オプション) クリーンアップする

タスク	説明	必要なスキル
デプロイされたリソースをすべて削除します。	リソースをアカウントにデプロイしたままにしておくことができます。未使用のときに EC2 インスタンスをシャットダウンし、S3 バケットを空	AWS DevOps

タスク	説明	必要なスキル
	のままにしておくと、将来のスキャンのためにリソースを維持するコストを削減できます。 すべてのリソースのプロビジョニングを解除するには、次の操作を実行します。 1. 管理アカウントにプロビジョニングされた `IAM-ProwlerExecRole` スタックを削除します。手順については、「CloudFormation のドキュメント」の「スタックの削除」を参照してください。 2. 組織の管理アカウントまたは委任された管理者アカウントにプロビジョニングされた スタックセットを削除します。手順については、「CloudFormation のドキュメント」の「スタックセットの削除」を参照してください。 3. `prowler-output` S3 バケット内のすべてのオブジェクトを削除します。手順については、「Amazon S3 のドキュメント」の「オブジェクトの削除」を参照してください。 4. セキュリティアカウントにプロビジョニングされた	

タスク	説明	必要なスキル
	`Prowler-Resources` スタックを削除します。手順については、「CloudFormation のドキュメント」の「スタックの削除」を参照してください。	

トラブルシューティング

問題	ソリューション
Session Manager を使用した EC2 インスタンスに接続できません。	SSM エージェントは、Systems Manager エンドポイントと通信できる必要があります。以下の操作を実行します。 1. EC2 インスタンスがデプロイされているサブネットがインターネットにアクセスできていることを確認します。 2. EC2 インスタンスを再起動します。
スタックセットをデプロイすると、CloudFormation コンソールに `Enable trusted access with AWS Organizations to use service-managed permissions` を求めるメッセージが表示されます。	これは、 AWS Organizations と CloudFormation の間で信頼されたアクセスが有効になっていないことを示します。サービスマネージド型のスタックセットをデプロイするには、信頼されたアクセスが必要です。このボタンを選択すると、信頼されたアクセスは有効になります。詳細については、「CloudFormation のドキュメント」の「信頼されたアクセスの有効化」を参照してください。

関連リソース

AWS ドキュメント

- に対するセキュリティコントロールの実装 AWS (AWS 規範ガイダンス)

その他のリソース

- 「Prowler」 (GitHub)

追加情報

プログラムによるエラーの削除

Access Denied 結果にエラーを含んでいる場合は、そのエラーを検出結果から削除する必要があります。これらのエラーは通常、Prowler が特定のリソースを評価できないようにする外部からの影響を受けた権限により生じます。たとえば、プロビジョニングされた S3 バケットを確認すると、一部のチェックが失敗します AWS Control Tower。これらの結果をプログラムで抽出し、除外した結果を新しいファイルとして保存できます。

次のコマンドは、1 つのテキスト文字列 (パターン) を含む行を削除し、その結果を新しいファイルに出力します。

- Linux または MacOS (Grep) の場合

```
grep -v -i "Access Denied getting bucket" myoutput.csv > myoutput_modified.csv
```

- Windows (PowerShell) の場合

```
Select-String -Path myoutput.csv -Pattern 'Access Denied getting bucket' -NotMatch >
 myoutput_modified.csv
```

次のコマンドは、複数のテキスト文字列と一致する行を削除し、その結果を新しいファイルに出力します。

- Linux または MacOS の場合 (文字列間にエスケープパイプを使用する)

```
grep -v -i 'Access Denied getting bucket\|Access Denied Trying to Get' myoutput.csv >
 myoutput_modified.csv
```

- Windows の場合 (文字列間にはカンマを使用します)

```
Select-String -Path myoutput.csv -Pattern 'Access Denied getting bucket', 'Access
 Denied Trying to Get' -NotMatch > myoutput_modified.csv
```

レポートの例

次の図は、統合された Prowler の検出結果レポートの検出結果ワークシート例を示しています。

次の図は、統合された Prowler の検出結果レポートに含まれる合格不合格ワークシート例を示しています。(デフォルトでは、合格結果は出力から除外されます)。

次の図は、統合された Prowler の検出結果レポートに含まれる重要度ワークシート例を示しています。

AWS Config と を使用して未使用の Amazon EBS ボリュームを削除する AWS Systems Manager

サンカル・サングボトラ (AWS) によって作成されました

概要

Amazon Elastic Block Store (Amazon EBS) ボリュームのライフサイクルは、通常、アタッチされている Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのライフサイクルとは独立しています。起動時に終了時に削除オプションを選択しない限り、Amazon EC2 インスタンスを終了すると Amazon EBS ボリュームはデタッチされますが、削除されません。特に、Amazon EC2 インスタンスの起動と終了が一般的である開発環境とテスト環境では、多くの未使用の Amazon EBS ボリュームが発生する可能性があります。Amazon EBS ボリュームは、使用されているかどうかにかかわらず AWS アカウント、 に料金が発生します。これらのボリュームを削除すると、 のコストを最適化できます AWS アカウント。さらに、未使用の Amazon EBS ボリュームを削除することは、それらのボリューム内の未使用で機密性の高いデータへのアクセスを防ぐためのセキュリティのベストプラクティスです。

AWS Config は、非準拠のリソースを手動でまたは自動的に修復するのに役立ちます。このパターンでは、アカウント内の未使用の Amazon EBS ボリュームを削除する AWS Config ルールと自動修復アクションを設定する方法について説明します。修復アクションは、 の一機能である Automation の事前定義されたランブックです AWS Systems Manager。削除する前にボリュームのスナップショットを作成するように、ランブックを設定できます。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Identity and Access Management Systems Manager の一機能である Automation の`AWSConfigRemediation-DeleteUnusedEBSVolume`ランブックを実行する (IAM) アクセス許可。詳細については、AWSConfigRemediation-DeleteUnusedEBSVolume の必要な IAM 権限 を参照してください。
- 1 つまたは複数の使用されていない Amazon EBS ボリューム。

制約事項

- 使用されていない Amazon EBS ボリュームは、その `available` 状態である必要があります。

アーキテクチャ

ターゲット アーキテクチャ

1. この AWS Config ルールは Amazon EBS ボリュームを評価します。
2. このルールは、準拠しているリソースと準拠していないリソースのリストを返します。未使用のボリュームである `available`状態にある Amazon EBS ボリュームは、非準拠であると判断されます。
3. AWS Config は自動化ランブックを自動的に開始します。
4. 設定されている場合、Systems Manager は未使用のボリュームのスナップショットを作成してから削除します。
5. Systems Manager は、未使用の Amazon EBS ボリュームを削除します。

自動化とスケール

このソリューションは、組織のすべてのアカウントに適用できます。詳細については、 AWS Config ドキュメントの「組織内のすべてのアカウントのルールの管理」を参照してください。

ツール

- AWS Config は、 内のリソースの詳細ビュー AWS アカウント と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。これにより、アプリケーションとリソースの管理が簡素化され、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できます。
- AWS Systems Manager 自動化は、多くの の一般的なメンテナンス、デプロイ、修復タスクを簡素化します AWS のサービス。

エピック

AWS Config ルールを設定する

タスク	説明	必要なスキル
自動化ランブック用のロールを作成します。	`AssumeRole` というロールを作成します。Systems Manager Automation は、このロールを使用してランブックを実行します。手順については、Systems Manager ドキュメントのオートメーションのサービスロール(ロールを引き受ける)アクセスの設定を参照してください。	AWS システム管理者
AWS Config レコーダーをオンにします。	AWS Config ドキュメントの「コンソール AWS Config でのセットアップ」の手順に従って、 AWS Config が実行されており、Amazon EBS ボリュームを記録するように設定されていることを確認します。	AWS システム管理者
ファイルを実行します。	1. AWS Config ドキュメントの「リソースの評価」の手順に従って`ec2-volume-inuse-check` ルールを実行します。評価が完了するまで待ちます。 2. ルール ページで `ec2-volume-inuse-check` ルールを選択し、範囲内のリソースで非準拠を選択します。	AWS システム管理者

タスク	説明	必要なスキル
	3. 評価結果に未使用の Amazon EBS ボリュームが 1 つ以上あることを確認します。	

未使用の Amazon EBS ボリュームの自動修復を設定します。

タスク	説明	必要なスキル
自動修復アクションを追加します。	1. ルールページで、`ec2-volume-inuse-check` ルールを選択します。 2. AWS Config ドキュメントの「自動修復の設定」の手順に従います。次の点に注意してください: 3. 是正アクションの詳細セクションで、`AWSConfigRemediation-DeleteUnusedEBSVolume` を選択します。 • リソース ID パラメータを選択し、リストから VolumeId を選択します。実行時に、このパラメータは非準拠の Amazon EBS ボリュームの ID に置き換えられます。 • Parameters セクションで、次のパラメーターの値を指定します。	AWS システム管理者

タスク	説明	必要なスキル
	• `CreateSnapshot` – (オプション) に設定すると`true`、オートメーションは削除前に Amazon EBS ボリュームのスナップショットを作成します。 • `AutomationAssumeRole` – `AssumeRole` のために前に作成した ロールの Amazon リソースネーム (ARN)。	
AWS Config ルールの自動修復をテストします。	1. AWS Config コンソールのルールページで、`ec2-volume-inuse-check` ルールを選択します。 2. アクション メニューで 名前を変更 を選択します。 3. ルールが準拠していないリソースを評価できるようにし、未使用の Amazon EBS ボリュームが削除されていることを確認します。	AWS システム管理者

トラブルシューティング

問題	ソリューション
AWS Config はリソースの状態を正確に反映しません。	リソースの状態を更新 AWS Config しない場合があります。レコーダーをオフにしてから、

問題	ソリューション
	AWS Config 設定ページで再度オンにします。レコーダーはリソースの状態をキャプチャします。新しく作成または削除されたリソースの場合、レコーダーが現在の状態を反映するまでに時間がかかることがあります。Amazon EBS ボリュームの状態の詳細については、Amazon EBS ドキュメントの「ボリュームの状態」を参照してください。

関連リソース

- AWSConfigRemediation-DeleteUnusedEBSVolume runbook
- ec2-volume-inuse-check rule
- AWS Config ルールを使用した非準拠 AWS リソースの修復

と CloudFormation を使用して AWS Control Tower コントロールをデプロイ AWS CDK および管理します。

作成者: Iker Reina Fuente (AWS)、Ivan Girardi (AWS)

概要

このパターンでは、 AWS CloudFormation と AWS Cloud Development Kit (AWS CDK) を使用して、Infrastructure as Code (IaC) としての予防的、検出的、プロアクティブ AWS Control Tower コントロールを実装および管理する方法を説明します。コントロール (ガードレールとも呼ばれます) は、 AWS Control Tower 環境全体に継続的なガバナンスを提供する高レベルのルールです。たとえば、 のログ記録を要求するコントロールを使用し AWS アカウント 、特定のセキュリティ関連イベントが発生した場合に自動通知を設定できます。

AWS Control Tower は、 AWS リソースを管理し、複数の にわたるコンプライアンスをモニタリングする予防、検出、プロアクティブコントロールを実装するのに役立ちます AWS アカウント。各コントロールは、1 つのルールを適用します。このパターンでは、提供された IaC テンプレートを使用して、環境にデプロイするコントロールを指定します。

AWS Control Tower コントロールは組織単位 (OU) 全体に適用され、コントロールは OU AWS アカウント 内のすべての に影響します。したがって、ユーザーがランディングゾーン内の任意のアカウントで作業を実行する場合、アクションは OU に適用されるコントロールに従います。

AWS Control Tower コントロールを実装することで、 AWS ランディングゾーンの強力なセキュリティ基盤を確立できます。このパターンを使用して CloudFormation と を介してコントロールを IaC としてデプロイすることで AWS CDK、ランディングゾーンのコントロールを標準化し、より効率的にデプロイおよび管理できます。このソリューションでは、cdk_nag を使用してデプロイ中に AWS CDK アプリケーションをスキャンします。このツールは、アプリケーションが AWS ベストプラクティスに準拠しているかどうかをチェックします。

AWS Control Tower コントロールを IaC としてデプロイするには、代わりに HashiCorp Terraform を使用することもできます AWS CDK。詳細については、「Terraform を使用した AWS Control Tower コントロールのデプロイと管理」を参照してください。

対象者

このパターンは AWS Control Tower、、CloudFormation AWS CDK、および の経験があるユーザーに推奨されます AWS Organizations。

前提条件と制限

前提条件

- AWS Organizations とラン AWS Control Tower ディングゾーンで組織として AWS アカウント 管理されているアクティブ。手順については、 AWS Control Tower ドキュメントの「開始方法」を参照してください。
- AWS Command Line Interface （ AWS CLI) をインストールして設定します。
- ノードパッケージマネージャー (npm)。 用にインストールおよび設定されています AWS CDK。
- の前提条件 AWS CDK。
- デプロイアカウントで既存の AWS Identity and Access Management (IAM) ロールを引き受けるアクセス許可。
- ブートストラップに使用できる組織の管理アカウントで IAM ロールを引き受けるアクセス許可 AWS CDK。ロールには、CloudFormation リソースを変更およびデプロイする権限が必要です。詳細については、 AWS CDK ドキュメントの「ブートストラップ」を参照してください。
- 組織の管理アカウントで IAM ロールとポリシーを作成するアクセス権限。詳細については、IAM ドキュメントの「IAM リソースにアクセスするために必要なアクセス権限」を参照してください。
- CT.CLOUDFORMATION.PR.1 という識別子の付いたサービスコントロールポリシー (SCP) ベースのコントロールを適用します。プロアクティブなコントロールをデプロイするには、この SCP を有効にする必要があります。手順については、AWS CloudFormation 「レジストリ内のリソースタイプ、モジュール、フックの管理の禁止」を参照してください。

制約事項

- このパターンでは、デプロイアカウントから組織の管理アカウントまで AWS アカウント、このソリューションをデプロイする手順を示します。テスト目的で、このソリューションを管理アカウントに直接デプロイすることもできますが、この設定の手順は明示されていません。
- AWS Control Tower コントロールの場合、このパターンでは、次の形式のグローバル識別子を使用する必要があります。

```
arn:<PARTITION>:controlcatalog:::control/<CONTROL_CATALOG_OPAQUE_ID>
```

このパターンの以前のバージョンでは、サポートされなくなったリージョン識別子が使用されていました。リージョン識別子からグローバル識別子に移行することをお勧めします。グローバル識別子は、コントロールを管理し、使用できるコントロールの数を増やすのに役立ちます。

> **Note**
>
> ほとんどの場合、 の値は <PARTITION>ですaws。

製品バージョン

- AWS Control Tower バージョン 3.2 以降
- Python バージョン 3.9 以降
- npm バージョン 2.9.0 以降

アーキテクチャ

このセクションでは、このソリューションの概要と、サンプルコードによって確立されたアーキテクチャについて説明します。次の図は、OU 内のさまざまなアカウントに展開されるコントロールを示しています。

AWS Control Tower コントロールは、その動作とガイダンスに従って分類されます。

コントロールの動作には、主に 3 つのタイプがあります。

1. 予防コントロールは、アクションの発生を防ぐように設計されています。これらは、 のサービスコントロールポリシー (SCPsまたはリソースコントロールポリシー (RCPs) で実装されます AWS Organizations。予防コントロールのステータスは、適用または無効です。予防的コントロールはすべての でサポートされています AWS リージョン。
2. 検出コントロールは、特定のイベントが発生したときに検出し、アクションをログに記録するように設計されています AWS CloudTrail。これらは AWS Config ルールで実装されます。検出コントロールのステータスは、クリア、違反、または無効です。検出コントロールは、 で AWS リージョン サポートされているコントロールにのみ適用されます AWS Control Tower。
3. プロアクティブコントロールは、 によってプロビジョニングされるリソースをスキャンし AWS CloudFormation 、会社のポリシーと目標に準拠しているかどうかをチェックします。準拠してい

ないリソースはプロビジョニングされません。これらはフックでAWS CloudFormation 実装されます。プロアクティブコントロールのステータスは、合格、不合格、または スキップです。

コントロールガイダンスとは、各コントロールを OUs に適用する方法について推奨されるプラクティスを指します。 AWS Control Tower には、必須、強く推奨、選択的の 3 つのカテゴリのガイダンスが用意されています。コントロールのガイダンスは、コントロールの動作とは無関係です。詳細については、「コントロールの動作とガイダンス」を参照してください。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。Toolkit AWS CDKは、 AWS CDK アプリを操作するための主要なツールです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および 全体のライフサイクルを通じてリソースを管理するのに役立ちます AWS リージョン。
- AWS Config は、 のリソース AWS アカウント の詳細と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。
- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境をセットアップして管理するのに役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。

その他のツール

- cdk_nag は、ルールパックを組み合わせて AWS CDK アプリケーションがベストプラクティスに準拠しているかどうかをチェックするオープンソースツールです。
- npm は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイの管理に使用されます。
- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、リポジトリを使用した GitHub Deploy コントロールで使用できます。 AWS Control TowerAWS CDKcdk.json ファイルを使用して AWS CDK アプリを操作し、package.json ファイルを使用して npm パッケージをインストールします。

ベストプラクティス

- 最小特権の原則 (IAM ドキュメント) に従ってください。このパターンで提供されるサンプル IAM ポリシーと信頼ポリシーには、最低限必要なアクセス許可が含まれており、管理アカウントで作成された AWS CDK スタックはこれらのアクセス許可によって制限されます。
- AWS Control Tower 管理者向けのベストプラクティス (AWS Control Tower ドキュメント) に従います。
- (AWS CDK ドキュメント) を使用してクラウドインフラストラクチャを開発およびデプロイするためのベストプラクティスに従います AWS CDK。
- をブートストラップするときは AWS CDK、ブートストラップテンプレートをカスタマイズして、管理アカウントの任意のリソースに対して読み書きできるポリシーと信頼されたアカウントを定義します。詳細については、「起動のカスタマイズ」を参照してください。
- cfn_nag などのコード分析ツールを使用して、生成された CloudFormation テンプレートをスキャンします。cfn-nag ツールは、インフラストラクチャが安全ではないことを示す可能性のあるパターンを CloudFormation テンプレートから探します。cdk-nag を使用して CloudFormation テンプレートを確認することもできます。これには cloudformation-include モジュールを使用します。

エピック

コントロールを有効にする準備をしてください。

タスク	説明	必要なスキル
管理アカウントで IAM ロールを作成します。	1. 「追加情報」セクションの IAM ポリシーで定義されている権限を使用して、管理アカウントに IAM ポリシーを作成します。詳細については、IAM ドキュメントの「IAM ポリシーの作成」を参照してください。ポリシーの Amazon リソー	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル

	スネーム (ARN) をメモします。以下は ARN の例です。
	
	`arn:aws:iam::<MANAGEMENT-ACCOUNT-ID>:policy/<POLICY-NAME>`
	
	2. 管理アカウントに IAM ロールを作成し、前のステップで作成した IAM アクセス権限ポリシーをアタッチし、「追加情報」セクションの「信頼ポリシー」にカスタム信頼ポリシーをアタッチします。詳細については、IAM ユーザーガイドの「カスタム信頼ポリシーを使用したロールの作成 (コンソール)」を参照してください。以下は、新しいロールの ARN の例です。
	
	`arn:aws:iam::<MANAGEMENT-ACCOUNT-ID>:role/<ROLE-NAME>`	

タスク	説明	必要なスキル
ブートストラップ AWS CDK。	1. 管理アカウントで、ブートストラップするアクセス許可を持つロールを引き受けます AWS CDK。 2. 次のコマンドを入力して、以下を置き換えます。 • <MANAGEMENT-ACCOUNT-ID> は組織の管理アカウントの ID です。 • <AWS-CONTROL-TOWER-REGION> は、 がデプロイ AWS リージョン される AWS Control Tower です。リージョンコードの完全なリストについては、AWS 「 全般のリファレンス」の「リージョンエンドポイント」を参照してください。 • <DEPLOYMENT-ACCOUNT-ID> は、デプロイアカウントの ID です。 • <DEPLOYMENT-ROLE-NAME> は、デプロイアカウントの IAM ロールの名前です。 • <POLICY-NAME> は、管理アカウントで作成したポリシーの名前です。 `$ npx cdk bootstrap aws://<MANAGEMENT-ACCOUNT-ID>/<AWS-C`	DevOps エンジニア、AWS 全般、Python

タスク	説明	必要なスキル
	ONTROL-TOWER-REGIO N> \ --trust arn:aws:i am::<DEPLOYMENT-AC COUNT-ID>:role/<DE PLOYMENT-ROLE-NAME> \ --cloudformation- execution-policies arn:aws:iam::<MANA GEMENT-ACCOUNT-ID> :policy/<POLICY-NA ME>	
リポジトリをクローン作成します。	bash シェルで、次のコマンドを入力します。これにより、GitHub のリポジトリを使用して Deploy AWS Control Tower コントロール AWS CDKがクローンされます。 `git clone https://github.com/aws-samples/aws-control-tower-controls-cdk.git`	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
AWS CDK 設定ファイルを編集します。	1. クローンしたリポジトリで constants.py ファイルを開きます。 2. `ACCOUNT_ID` パラメータに、管理アカウントの ID を入力します。 3. `<AWS-CONTROL-TOWER-REGION>` パラメータに、 AWS Control Tower がデプロイ AWS リージョンされている を入力します。 4. `ROLE_ARN` パラメータには、管理アカウントで作成したロールの ARN を入力します。 5. AWS Control Tower ドキュメントですべてのグローバル識別子を開きます。 6. JSON 形式のリストで、実装するコントロールを見つけ、そのグローバル識別子 ({CONTROL_CATALOG_OPAQUE_ID} 値とも呼ばれます) をコピーします。たとえば、AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED コントロールのグローバル識別子は ですk4izcjxhukijhajp6ks5mjxk 。 7. `GUARDRAILS_CONFIGURATION` セクションの `Enable-Control` パラ	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	メータに、コピーしたグローバル識別子を入力します。識別子を二重引用符で囲んで入力し、複数の識別子はカンマで区切ります。 8. `GUARDRAILS_CONFIGURATION` セクションの `OrganizationalUnitIds` パラメータに、コントロール (`ou-1111-11111111` など) を有効にする組織単位の ID を入力します。値を二重引用符で囲んで入力し、複数の ID をコンマで区切ります。OU ID を取得する方法の詳細については、「OU の詳細の表示」を参照してください。 9. constants.py ファイルを保存して閉じます。更新された constants.py ファイルの例については、このパターンの「追加情報」セクションを参照してください。	

管理アカウントのコンソールを有効にする

タスク	説明	必要なスキル
デプロイアカウントで IAM ロールを引き受けます。	デプロイアカウントで、管理アカウントに AWS CDK スタックをデプロイするアクセス許可を持つ IAM ロールを引	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	き受けます。で IAM ロールを引き受ける方法の詳細については AWS CLI、「」の「IAM ロールを使用する AWS CLI」を参照してください。	

環境をアクティブ化します。	Linux または macOS を使用している場合:
	1. 以下のコマンドを入力して仮想環境を作成します。
	`$ python3 -m venv .venv`
	2. 仮想環境の作成後、次のコマンドを入力して仮想環境を有効化します。
	`$ source .venv/bin/activate`
	Windows を使用している場合:
	1. 次のコマンドを入力して仮想環境をアクティブ化します。
	`% .venv\Scripts\activate.bat`
		DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
依存関係をインストールします。	仮想環境がアクティブになったら、次のコマンドを入力して install_deps.sh スクリプトを実行します。このスクリプトは、必要な依存ファイルをインストールします。 `$ ./scripts/install_deps.sh`	DevOps エンジニア、AWS 全般、Python

スタックをデプロイします。	次のコマンドを入力して、CloudFormation スタックを統合してデプロイします。
	`$ npx cdk synth`
	`$ npx cdk deploy`
		DevOps エンジニア、AWS 全般、Python

関連リソース

AWS ドキュメント

- コントロールについて (AWS Control Tower ドキュメント)
- コントロールライブラリ (AWS Control Tower ドキュメント)
- AWS CDK ツールキットコマンド (AWS CDK ドキュメント)
- Terraform を使用した AWS Control Tower コントロールのデプロイと管理 (AWS 規範ガイダンス)

その他のリソース

- Python

追加情報

constants.py ファイルの例

以下は、更新された constants.py ファイルの例です。このサンプルでは、AWS-GR_ENCRYPTED_VOLUMES コントロール (グローバル ID: 503uicglhjkokaajywfpt6ros) と AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED コントロール (グローバル ID:) を有効にします50z1ot237wl8u1lv5ufau6qqo。グローバル IDs「すべてのグローバル識別子」を参照してください。 AWS Control Tower

```
ACCOUNT_ID = 111122223333
AWS_CONTROL_TOWER_REGION = us-east-2
ROLE_ARN = "arn:aws:iam::111122223333:role/CT-Controls-Role"
GUARDRAILS_CONFIGURATION = [
    {
        "Enable-Control": {
            "503uicglhjkokaajywfpt6ros",
            ...
        },
        "OrganizationalUnitIds": ["ou-1111-11111111", "ou-2222-22222222"...],
    },
    {
        "Enable-Control": {
            "50z1ot237wl8u1lv5ufau6qqo",
            ...
        },
        "OrganizationalUnitIds": ["ou-2222-22222222"...],
    },
]
```

IAM ポリシー

次のサンプルポリシーでは、デプロイアカウントから管理アカウントに AWS CDK スタックをデプロイするときに AWS Control Tower コントロールを有効または無効にするために必要な最小限のアクションを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "controltower:EnableControl",
                "controltower:DisableControl",
                "controltower:GetControlOperation",
                "controltower:ListEnabledControls",
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DetachPolicy",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListChildren",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:UpdatePolicy",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

信頼ポリシー

以下のカスタム信頼ポリシーでは、デプロイアカウントの特定の IAM ロールが管理アカウントの IAM ロールを引き継ぐことを許可しています。以下に置き換えます:

- <DEPLOYMENT-ACCOUNT-ID> は、デプロイアカウントの ID です
- <DEPLOYMENT-ROLE-NAME> は、管理アカウントでロールを引き受けることが許可されているデプロイアカウント内のロールの名前です。

```
{
    “Version”: “2012-10-17”,
    “Statement”: [
        {
            “Effect”: “Allow”,
            “Principal”: {
                “AWS”: “arn:aws:iam::<DEPLOYMENT-ACCOUNT-ID>:role/<DEPLOYMENT-ROLE-
NAME>”
```

```
            },
            “Action”: “sts:AssumeRole”,
            “Condition”: {}
        }
    ]
}
```

Terraform を使用した AWS Control Tower コントロールのデプロイと管理

作成者: Iker Reina Fuente (AWS)、Ivan Girardi (AWS)

概要

このパターンでは、 AWS Control Tower コントロール、HashiCorp Terraform、Infrastructure as Code (IaC) を使用して、予防、検出、プロアクティブセキュリティコントロールを実装および管理する方法を説明します。コントロール (ガードレールとも呼ばれます) は、 AWS Control Tower 環境全体に継続的なガバナンスを提供する高レベルのルールです。たとえば、コントロールを使用してのログ記録を要求し AWS アカウント 、特定のセキュリティ関連イベントが発生した場合に自動通知を設定できます。

AWS Control Tower は、 AWS リソースを管理し、複数の にわたってコンプライアンスをモニタリングする予防、検出、プロアクティブコントロールを実装するのに役立ちます AWS アカウント。各コントロールは、1 つのルールを適用します。このパターンでは、提供された IaC テンプレートを使用して、環境にデプロイするコントロールを指定します。

AWS Control Tower コントロールは組織単位 (OU) 全体に適用され、コントロールは OU AWS アカウント 内のすべての に影響します。したがって、ユーザーがランディングゾーン内の任意のアカウントで作業を実行する場合、アクションは OU に適用されるコントロールに従います。

AWS Control Tower コントロールを実装することで、 AWS ランディングゾーンの強力なセキュリティ基盤を確立できます。このパターンを使用してコントロールを Terraform IaC としてデプロイすることで、ランディングゾーンでのコントロールを標準化し、より効率的にデプロイして管理できます。

AWS Control Tower コントロールを IaC としてデプロイするには、Terraform AWS Cloud Development Kit (AWS CDK) の代わりに を使用することもできます。詳細については、「AWS CDK と を使用して AWS Control Tower コントロールをデプロイおよび管理する AWS CloudFormation」を参照してください。

対象者

このパターンは AWS Control Tower、、Terraform、および の経験があるユーザーに推奨されます AWS Organizations。

前提条件と制限

前提条件

- AWS Organizations とラン AWS Control Tower ディングゾーンで組織として AWS アカウント 管理されているアクティブ。手順については、 AWS Control Tower ドキュメントの「開始方法」を参照してください。
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み。
- このパターンをデプロイするアクセス許可を持つ管理アカウントの AWS Identity and Access Management (IAM) ロール。必要な権限とサンプルポリシーの詳細については、このパターンの「追加情報」セクションにある「IAM ロールの最小特権」を参照してください。
- 管理アカウントで IAM ロールを割り当てる権限。
- CT.CLOUDFORMATION.PR.1 という識別子の付いたサービスコントロールポリシー (SCP) ベースのコントロールを適用します。プロアクティブなコントロールをデプロイするには、この SCP を有効にする必要があります。手順については、AWS CloudFormation 「レジストリ内のリソースタイプ、モジュール、フックの管理を禁止する」を参照してください。
- Terraform CLI をインストール済み (Terraform のドキュメント)
- Terraform AWS Provider、設定済み (Terraform ドキュメント)。
- Terraform バックエンドを設定済み (Terraform のドキュメント)

制約事項

- AWS Control Tower コントロールの場合、このパターンでは、次の形式のグローバル識別子を使用する必要があります。

```
arn:<PARTITION>:controlcatalog:::control/<CONTROL_CATALOG_OPAQUE_ID>
```

このパターンの以前のバージョンでは、サポートされなくなったリージョン識別子が使用されていました。リージョン識別子からグローバル識別子に移行することをお勧めします。グローバル識別子は、コントロールを管理し、使用できるコントロールの数を増やすのに役立ちます。

> **Note**
>
> ほとんどの場合、 の値は <PARTITION>ですaws。

製品バージョン

- AWS Control Tower バージョン 3.2 以降
- Terraform バージョン 1.5 以降
- Terraform AWS プロバイダーバージョン 4.67 以降

アーキテクチャ

このセクションでは、このソリューションの概要と、サンプルコードによって確立されたアーキテクチャについて説明します。次の図は、OU 内のさまざまなアカウントに展開されるコントロールを示しています。

AWS Control Tower コントロールは、その動作とガイダンスに従って分類されます。

コントロールの動作には、主に 3 つのタイプがあります。

1. 予防コントロールは、アクションの発生を防ぐように設計されています。これらは、 のサービスコントロールポリシー (SCPsまたはリソースコントロールポリシー (RCPs) で実装されます AWS Organizations。予防コントロールのステータスは、適用または無効です。予防的コントロールはすべての でサポートされています AWS リージョン。
2. 検出コントロールは、特定のイベントが発生したときに検出し、アクションをログに記録するように設計されています AWS CloudTrail。これらは AWS Config ルールで実装されます。検出コントロールのステータスは、クリア、違反、または無効です。検出コントロールは、 で AWS リージョン サポートされているコントロールにのみ適用されます AWS Control Tower。
3. プロアクティブコントロールは、 によってプロビジョニングされるリソースをスキャンし AWS CloudFormation 、会社のポリシーと目標に準拠しているかどうかをチェックします。準拠していないリソースはプロビジョニングされません。これらはフックでAWS CloudFormation 実装されます。プロアクティブコントロールのステータスは、合格、不合格、または スキップです。

コントロールガイダンスは、各コントロールを OUs。 は、必須、強く推奨、選択的の 3 つのカテゴリのガイダンス AWS Control Tower を提供します。コントロールのガイダンスは、コントロールの動作とは無関係です。詳細については、「コントロールの動作とガイダンス」を参照してください。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。
- AWS Config は、 内のリソースの詳細ビュー AWS アカウント と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。
- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境を設定および管理するのに役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。

その他のツール

- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するためのInfrastructure as Code (IaC) ツールです。

コードリポジトリ

このパターンのコードは、GitHub Deploy and manage AWS Control Tower controls by using Terraform repository で入手できます。

ベストプラクティス

- このソリューションのデプロイに使用する IAM ロールは、「最小権限の原則」 (IAM ドキュメント) に従う必要があります。
- AWS Control Tower 管理者向けのベストプラクティス (AWS Control Tower ドキュメント) に従ってください。

エピック

管理アカウントのコンソールを有効にする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	bash シェルで、次のコマンドを入力します。これによ	DevOps エンジニア

タスク	説明	必要なスキル
	り、GitHub の Terraform リポジトリを使用して、デプロイおよび管理 AWS Control Tower コントロールのクローンが作成されます。 `git clone https://github.com/aws-samples/aws-control-tower-controls-terraform.git`	
Terraform バックエンド設定ファイルを編集します。	1. クローンしたリポジトリで backend.tf ファイルを開きます。 2. このファイルを編集して、Terraform バックエンドを設定します。このファイルに定義する設定は、環境によって異なります。詳細については、「バックエンドの設定」(Terraform ドキュメント) を参照してください。 3. backend.tf ファイルを保存して閉じます。	DevOps エンジニア、Terraform

タスク	説明	必要なスキル
Terraform プロバイダー設定ファイルを編集します。	1. クローンしたリポジトリで provider.tf ファイルを開きます。 2. このファイルを編集して、Terraform プロバイダーを設定します。詳細については、「プロバイダーの設定」(Terraform ドキュメント) を参照してください。を AWS Control Tower API が利用可能なリージョン AWS リージョン として設定します。 3. provider.tf ファイルを保存して閉じます。	DevOps エンジニア、Terraform

タスク	説明	必要なスキル
設定ファイルを編集します。	1. クローンしたリポジトリで variables.tfvars ファイルを開きます。 2. AWS Control Tower ドキュメントですべてのグローバル識別子を開きます。 3. JSON 形式のリストで、実装するコントロールを見つけ、そのグローバル識別子 ({CONTROL_CATALOG_OPAQUE_ID} 値とも呼ばれます) をコピーします。たとえば、AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED コントロールのグローバル識別子は ですk4izcjxhukijhajp6ks5mjxk 。 4. controls セクションの control_names パラメータに、コピーしたグローバル識別子を入力します。 5. controls セクションの organizational_unit_ids パラメータに、コントロール (ou-1111-11111111 など) を有効にする組織単位の ID を入力します。値を二重引用符で囲んで入力し、複数の ID をコンマで区切ります。OU ID を取得する方法の詳細については、「OU の詳細を	DevOps エンジニア、AWS 全般、Terraform

タスク	説明	必要なスキル
	表示する」を参照してください。 6. variables.tfvars ファイルを保存して閉じます。最新の variables.tfvars ファイルの例については、このパターンの「追加情報」セクションを参照してください。	
管理アカウントで IAM ロールを割り当てます。	管理アカウントでは、Terraform 設定ファイルをデプロイする権限のある IAM ロールを引き受けます。必要な権限とサンプルポリシーの詳細については、このパターンの「追加情報」セクションにある「IAM ロールの最小特権」を参照してください。で IAM ロールを引き受ける方法の詳細については AWS CLI、『』の「IAM ロールを使用する AWS CLI」を参照してください。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル

設定ファイルをデプロイします。	1. 以下のコマンドを入力して、Terraform を初期化します。
	`$ terraform init -upgrade`
	2. 次のコマンドを入力して、現在の状態と比較した変更内容をプレビューします。
	`$ terraform plan -var-file="variables.tfvars"`
	3. Terraform プランの設定変更を見直し、組織に変更を実装することを確認します。
	4. リソースをデプロイするには、次のコマンドを実行します。
	`$ terraform apply -var-file="variables.tfvars"`
		DevOps エンジニア、AWS 全般、Terraform

(オプション) AWS Control Tower 管理アカウントのコントロールを無効にする

タスク	説明	必要なスキル
`destroy` コマンドを実行します。	以下のコマンドを入力して、このパターンでデプロイされたリソースを削除します。	DevOps エンジニア、AWS 全般、Terraform

タスク	説明	必要なスキル
	`$ terraform destroy -var-file="variables.tfvars"`	

トラブルシューティング

問題	ソリューション
`Error: creating ControlTower Control ValidationException: Guardrail <control ID> is already enabled on organizational unit <OU ID>` エラー	有効化しようとするコントロールは、ターゲット OU で有効になっています。このエラーは、ユーザーが を通じて、 を通じて AWS Management Console、 AWS Control Tower または を通じてコントロールを手動で有効にした場合に発生する可能性があります AWS Organizations。Terraform 設定ファイルをデプロイするには、次のいずれかのオプションを使用できます。 オプション 1: Terraform の現在の状態ファイルを更新する Terraform の現在の状態ファイルにリソースをインポートできます。`apply` コマンドを再実行すると、Terraform はこのリソースをスキップします。次の手順に従い、現在の Terraform 状態にリソースをインポートします。 1. AWS Control Tower 管理アカウントで、次のコマンドを入力して OUs の Amazon リソースネーム (ARNs) のリストを取得します。ここで、 `<root-ID>` は組織のルートです。この ID 取得の詳細については、「ルートの詳細を表示する」を参照してください。

問題	ソリューション

	`aws organizations list-organizational-units-for-parent --parent-id <root-ID>`
	2. 前のステップで返された各 OU について、次のコマンドを入力します。ここで、<OU-ARN> は OU の ARN です。
	`aws controltower list-enabled-controls --target-identifier <OU-ARN>`
	3. ARN をコピーし、必要なモジュールで Terraform インポートを実行して Terraform 状態に含まれるようにします。手順については、「インポート」(Terraform ドキュメント) を参照してください。
	4. 「エピック」セクションにある「設定のデプロイ」の手順を繰り返します。
	
	オプション 2: コントロールを無効にする
	
	実稼働以外の環境で作業している場合は、コンソールでコントロールを無効にできます。エピックセクションの「設定をデプロイする」のステップを繰り返して、再度有効にします。???コントロールが無効になる期間があるため、このアプローチは実稼働環境にはお勧めしません。本番環境でこのオプションを使用する場合は、SCP を一時的に適用するなどの一時的なコントロールを実装できます AWS Organizations。

関連リソース

AWS ドキュメント

- コントロールについて (AWS Control Tower ドキュメント)
- コントロールライブラリ (AWS Control Tower ドキュメント)
- AWS CDK と を使用して AWS Control Tower コントロールをデプロイおよび管理します AWS CloudFormation (AWS 規範ガイダンス)

その他のリソース

- Terraform
- Terraform CLI のドキュメント

追加情報

variables.tfvars ファイルの例

以下に、更新された variables.tfvars ファイルの例を示します。このサンプルでは、AWS-GR_ENCRYPTED_VOLUMES コントロール (グローバル ID: `503uicglhjkokaajywfpt6ros`) と AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED コントロール (グローバル ID:) を有効にします`50z1ot237wl8u1lv5ufau6qqo`。グローバル IDs「すべてのグローバル識別子」を参照してください。 AWS Control Tower

```
controls = [
    {
        control_names = [
            "503uicglhjkokaajywfpt6ros",
            ...
        ],
        organizational_unit_ids = ["ou-1111-11111111", "ou-2222-22222222"...],
    },
    {
        control_names = [
            "50z1ot237wl8u1lv5ufau6qqo",
            ...
        ],
        organizational_unit_ids = ["ou-1111-11111111"...],
    },
```

```
]
```

IAM ロールの最小特権

このパターンでは、管理アカウントで IAM ロールを引き受ける必要があります。一時的な権限を持つロールを割り当て、最小特権の原則に従って権限を制限するのがベストプラクティスです。次のサンプルポリシーでは、 AWS Control Tower コントロールを有効または無効にするために必要な最小限のアクションを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "controltower:EnableControl",
                "controltower:DisableControl",
                "controltower:GetControlOperation",
                "controltower:ListEnabledControls",
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeOrganization",
                "organizations:DetachPolicy",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListChildren",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:UpdatePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

複数のコード成果物のセキュリティ問題を同時に検出するパイプラインをデプロイする

作成者: Benjamin Morris (AWS)、Dina Odum (AWS)、Isaiah Schisler (AWS)、Sapeksh Madan (AWS)、Tim Hahn (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら。

Simple Code Scanning Pipeline (SCSP) では、業界標準のオープンソースセキュリティツールを並行して実行するコード分析パイプラインを 2 クリックで作成できます。これにより、開発者はツールをインストールしたり、実行方法を理解したりすることなく、コードの品質とセキュリティを確認できます。これにより、コード成果物の脆弱性や設定ミスを減らすことができます。また、組織がセキュリティツールのインストール、調査、設定に費やす時間も短縮されます。

SCSP の前に、この特定のツールスイートを使用してコードをスキャンするには、開発者がソフトウェア分析ツールを検索、手動でインストール、設定する必要がありました。自動セキュリティヘルパー (ASH) など、ローカルにインストールされたall-in-oneツールでも、 を実行するには Docker コンテナを設定する必要があります。ただし、SCSP では、業界標準のコード分析ツールのスイートがで自動的に実行されます AWS クラウド。このソリューションでは、Git を使用してコード成果物をプッシュし、セキュリティチェックが失敗したインサイトがat-a-glanceビジュアル出力を受け取ります。

前提条件と制限

- アクティブな AWS アカウント
- セキュリティ上の問題をスキャンしたい 1 つ以上のコード成果物
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み
- Python バージョン 3.0 以降および pip バージョン 9.0.3 以降がインストールされている
- 「インストール済み」Git
- ローカルワークステーションに git-remote-codecommit をインストールする

アーキテクチャ

ターゲットテクノロジースタック

- AWS CodeCommit リポジトリ
- AWS CodeBuild プロジェクト
- AWS CodePipeline パイプライン
- Amazon Simple Storage Service (Amazon S3) バケット
- AWS CloudFormation テンプレート

ターゲット アーキテクチャ

静的コード分析用の SCSP は、配信可能なコードに関するセキュリティフィードバックを提供するように設計された DevOps プロジェクトです。

1. で AWS Management Console、ターゲットにログインします AWS アカウント。パイプラインをデプロイする AWS リージョン にいることを確認します。
2. コードリポジトリの CloudFormation テンプレートを使用して、SCSP スタックをデプロイします。これにより、新しい CodeCommit リポジトリと CodeBuild プロジェクトが作成されます。

> **Note**
>
> 代替のデプロイオプションとして、スタックのデプロイ中にリポジトリの Amazon リソースネーム (ARN) をパラメータとして指定することで、既存の CodeCommit リポジトリを使用できます。

3. リポジトリをローカルワークステーションにクローンし、クローンされたリポジトリ内のそれぞれのフォルダにファイルを追加します。
4. Git を使用して、CodeCommit リポジトリにファイルを追加、コミット、プッシュします。
5. CodeCommit リポジトリにプッシュすると、CodeBuild ジョブが開始されます。CodeBuild プロジェクトは、セキュリティツールを使用してコード成果物をスキャンします。
6. パイプラインの出力を確認します。エラーレベルの問題を検出したセキュリティツールでは、パイプラインでアクションが失敗します。これらのエラーを修正するか、誤検出として抑制します。CodePipeline のアクションの詳細またはパイプラインの S3 バケットのツール出力の詳細を確認します。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。

その他のツール

SCSP がコード成果物のスキャンに使用するツールの完全なリストについては、GitHub の SCSP readme を参照してください。

コードリポジトリ

このパターンのコードは、GitHub の Simple Code Scanning Pipeline (SCSP) リポジトリで入手できます。

エピック

SCSP をデプロイする

タスク	説明	必要なスキル
CloudFormation スタックを更新する。	1. AWS Management Consoleにサインインします。 2. コンソールで、ソリューションをデプロイするターゲットリージョンにいることを確認します。詳細については、「リージョンの選択」を参照してください。	AWS DevOps、AWS 管理者

タスク	説明	必要なスキル
	3. 次のリンクを選択します。これにより、CloudFormation でスタックのクイック作成ウィザードが開きます。 https://console.aws.amazon.com/cloudformation/home?#/stacks/create/review?templateURL=https://proservetools.s3.amazonaws.com/cft/scsp-pipeline-stack.template.json&stackName=SimpleCodeScanPipeline 4. スタックのクイック作成ウィザードで、スタックのパラメータ設定を確認し、ユースケースに応じて変更を加えます。 5. [I acknowledge that AWS CloudFormation might create IAM resources (AWS CloudFormation によって IAM リソースが作成される場合があることを承認します)] を選択し、[スタックの作成] を選択します。 これにより、CodeCommit リポジトリ、CodePipeline パイプライン、複数の CodeBuild ジョブ定義、S3 バケットが作成されます。ビルド実行とスキャン結果がこのバケッ	

タスク	説明	必要なスキル
	トにコピーされます。Cloud Formation スタックが完全にデプロイされると、SCSP は使用できる状態になります。	

パイプラインを使用する

タスク	説明	必要なスキル
スキャンの結果を調べます。	1. Amazon S3 コンソールのバケットで、simplecodescanpipeline-deleteresourcespipelinereso バケットを選択します。 2. scan_results ディレクトリを選択し、最新のスキャン日付スタンプを持つフォルダを選択します。 3. このフォルダのログファイルを確認して、パイプラインで使用されるセキュリティツールによって検出された問題を確認します。エラーレベルの問題を検出したセキュリティツールでは、パイプラインで`failed`アクションが発生します。これらは誤検出の場合は修正または抑制する必要があります。	アプリ開発者、AWS DevOps

タスク	説明	必要なスキル
	Note また、CodePipeline コンソールの「アクションの詳細」セクションでツール出力の詳細 (スキャンの合格と不合格の両方) を表示することもできます。	

トラブルシューティング

問題	ソリューション
HashiCorp Terraform または AWS CloudFormation ファイルはスキャンされません。	Terraform (.tf) および CloudFormation (.yml、.yaml、または .json) ファイルが、クローンされた CodeCommit リポジトリの適切なフォルダに配置されていることを確認します。
`git clone` コマンドは失敗しています。	がインストールされていること、`git-remote-codecommit` および CLI が CodeCommit リポジトリを読み取るアクセス許可を持つ AWS 認証情報にアクセスできることを確認します。
などの同時実行エラー`Project-level concurrent build limit cannot exceed the account-level concurrent build limit of 1 。`	CodePipeline コンソールでリリース変更ボタンを選択して、パイプラインを再実行します。これは、パイプラインが実行される最初の数回で最も一般的と思われる既知の問題です。

関連リソース

SCSP プロジェクトに関するフィードバックを提供します。

追加情報

よくある質問

SCSP プロジェクトは、自動セキュリティヘルパー (ASH) と同じですか？

いいえ。コンテナを使用してコードスキャンツールを実行する CLI ツールが必要な場合は、ASH を使用します。自動セキュリティヘルパー (ASH) は、新しいコード、インフラストラクチャ、または IAM リソース設定でセキュリティ違反が発生する可能性を減らすように設計されたツールです。ASH は、ローカルで実行できるコマンドラインユーティリティです。ローカルで使用するには、コンテナ環境がシステムにインストールされ、動作している必要があります。

ASH よりも簡単なセットアップパイプラインが必要な場合は、SCSP を使用します。SCSP はローカルインストールを必要としません。SCSP は、パイプラインで個別にチェックを実行し、ツールごとに結果を表示するように設計されています。また、SCSP は Docker の設定に伴うオーバーヘッドの多くを回避し、オペレーティングシステム (OS) に依存しません。

SCSP はセキュリティチーム専用ですか？

いいえ。誰でもパイプラインをデプロイして、コードのどの部分がセキュリティチェックに失敗しているかを判断できます。例えば、セキュリティ以外のユーザーは、セキュリティチームに確認する前に SCSP を使用してコードを確認できます。

GitLabitGitHub、SCSP を使用できますか？

2 つの異なるリモートリポジトリを指すようにローカル git リポジトリを設定できます。たとえば、既存の GitLab リポジトリのクローンを作成し、SCSP インスタンス (必要に応じて CloudFormation、Terraform、および AWS Config Rules Development Kit (AWS RDK) フォルダを指定) を作成し、 を使用して SCSP CodeCommit リポジトリのローカルリポジトリを`git remote add upstream <SCSPGitLink>`ポイントできます。これにより、コードの変更を最初に SCSP に送信し、検証してから、検出結果に対応するために追加の更新が行われた後、GitLab、GitHub、または Bitbucket リポジトリにプッシュできます。複数のリモートの詳細については、「追加の Git リポジトリにコミットをプッシュする」(AWS ブログ記事) を参照してください。

> **Note**
>
> ウェブインターフェイスを介して変更を行わないようにするため、ドリフトに注意してください。

独自のアクションの寄稿と追加

SCSP セットアップは、SCSP AWS Cloud Development Kit (AWS CDK) アプリケーションのソースコードを含む GitHub プロジェクトとして維持されます。パイプラインにチェックを追加するには、AWS CDK アプリケーションを更新し、 AWS アカウント パイプラインを実行するターゲットに合成またはデプロイする必要があります。これを行うには、まず SCSP GitHub プロジェクトのクローンを作成し、 `lib`フォルダでスタック定義ファイルを見つけます。

追加する追加のチェックがある場合、 AWS CDK コードの `StandardizedCodeBuildProject` クラスにより、アクションの追加が非常に簡単になります。名前、説明、`install`または `build` コマンドを指定します。 は、適切なデフォルト値を使用して CodeBuild プロジェクト AWS CDK を作成します。ビルドプロジェクトの作成に加えて、ビルドステージの CodePipeline アクションに追加する必要があります。新しいチェックを設計する場合、スキャンツールが問題を検出したか、実行に失敗`FAIL`した場合、アクションは である必要があります。スキャンツール`PASS`が問題を検出しない場合、 アクションは である必要があります。ツールの設定例については、 `Bandit`アクションのコードを確認してください。

予想される入力と出力の詳細については、リポジトリのドキュメントを参照してください。

カスタムアクションを追加する場合は、 `cdk deploy`または を使用して SCSP をデプロイする必要があります`cdk synth + CloudFormation deploy`。これは、クイック作成スタック CloudFormation テンプレートがリポジトリ所有者によって維持されるためです。

を使用してパブリックサブネットの検出属性ベースのアクセスコントロールをデプロイする AWS Config

作成者: Alberto Menendez (AWS)

概要

分散エッジネットワークアーキテクチャは、仮想プライベートクラウド (VPCs) 内のワークロードとともに実行されるネットワークエッジセキュリティに依存しています。これにより、より一般的で一元化されたアプローチと比較して、前例のないスケーラビリティが得られます。パブリックサブネットをワークロードアカウントにデプロイすることには利点がありますが、アタックサーフェスが増えるため、新たなセキュリティリスクも生じます。Application Load Balancer や NAT ゲートウェイなどの Elastic Load Balancing リソースのみを、これらの VPCs のパブリックサブネットにデプロイすることをお勧めします。専用のパブリックサブネットでロードバランサーと NAT ゲートウェイの使用は、インバウンドトラフィックとアウトバウンドトラフィックのきめ細かな制御に役立ちます。

パブリックサブネットにデプロイできるリソースのタイプを制限するために、予防的コントロールと検出的コントロールの両方を実装することをお勧めします。属性ベースのアクセスコントロール (ABAC) を使用してパブリックサブネットの予防的コントロールをデプロイする方法の詳細については、「パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする」を参照してください。ほとんどの状況で有効ですが、これらの予防的コントロールは、考えられるすべてのユースケースに対処できるとは限りません。したがって、このパターンは ABAC アプローチに基づいて構築され、パブリックサブネットにデプロイされている非準拠リソースに関するアラートを設定するのに役立ちます。このソリューションは、Elastic Network Interface がパブリックサブネットで許可されていないリソースに属しているかどうかをチェックします。

これを実現するために、このパターンではAWS Config カスタムルールと ABAC を使用します。カスタムルールは、Elastic Network Interface が作成または変更されるたびに、その設定を処理します。大まかに言うと、このルールは 2 つのアクションを実行して、ネットワークインターフェイスが準拠しているかどうかを判断します。

1. ネットワークインターフェイスがルールの範囲内にあるかどうかを判断するために、ルールは、サブネットにパブリックサブネットであることを示す特定のAWS タグがあるかどうかをチェックします。たとえば、このタグは ですIsPublicFacing=True。
2. ネットワークインターフェイスがパブリックサブネットにデプロイされている場合、ルールはこのリソースを AWS のサービス 作成した をチェックします。リソースが Elastic Load Balancing リソースまたは NAT ゲートウェイでない場合、リソースは非準拠としてマークされます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS Config、ワークロードアカウントでセットアップする
- ワークロードアカウントに必要なリソースをデプロイするアクセス許可
- パブリックサブネットを持つ VPC
- ターゲットパブリックサブネットを識別するために適切に適用されるタグ
- （オプション) の組織 AWS Organizations
- （オプション) AWS Config と の委任管理者である中央セキュリティアカウント AWS Security Hub

アーキテクチャ

ターゲット アーキテクチャ

この図表は、以下を示すものです：

1. Elastic Network Interface リソース (`AWS::EC2::NetworkInterface`) がデプロイまたは変更されると、 はイベントと設定を AWS Config キャプチャします。
2. AWS Config は、このイベントを、設定の評価に使用されるカスタムルールと照合します。
3. このカスタムルールに関連付けられた AWS Lambda 関数が呼び出されます。関数はリソースを評価し、指定されたロジックを適用して、リソース設定が `COMPLIANT`か `NON_COMPLIANT`かを判断します`NOT_APPLICABLE`。
4. リソースが であると判断された場合`NON_COMPLIANT`、 は Amazon Simple Notification Service (Amazon SNS) を介してアラート AWS Config を送信します。

> **Note**
>
> このアカウントが のメンバーアカウントである場合は AWS Organizations、 AWS Config または を使用してコンプライアンスデータを中央セキュリティアカウントに送信できます AWS Security Hub。

Lambda 関数の評価ロジック

次の図は、Elastic Network Interface のコンプライアンスを評価するために Lambda 関数によって適用されるロジックを示しています。

自動化とスケール

このパターンは検出ソリューションです。また、修復ルールで補完して、非準拠のリソースを自動的に解決することもできます。詳細については、「 AWS Config ルールを使用した非準拠リソースの修復」を参照してください。

このソリューションは、次の方法でスケールできます。

- パブリックサブネットを識別するために確立した対応する AWS タグの適用を強制します。詳細については、 AWS Organizations ドキュメントの「タグポリシー」を参照してください。
- 組織内のすべてのワークロードアカウントに AWS Config カスタムルールを適用する中央セキュリティアカウントを設定します。詳細については、「大規模な設定コンプライアンスの自動化 AWS」(AWS ブログ記事) を参照してください。
- AWS Config と統合 AWS Security Hub して、大規模なキャプチャ、一元化、通知を行います。詳細については、 AWS Security Hub ドキュメントの「設定 AWS Config」を参照してください。

ツール

- AWS Config は、 内のリソースの詳細ビュー AWS アカウント と設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。
- 受信したアプリケーションまたはネットワークトラフィックを複数のターゲットに分散するには、Elastic Load Balancing を使用します。例えば、1 つまたは複数のアベイラビリティーゾーンの Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、および IP アドレスにトラフィックを分散できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

ベストプラクティス

カスタム AWS Config ルールを開発するためのその他の例とベストプラクティスについては、GitHub の公式AWS Config ルールリポジトリを参照してください。

エピック

解決策をデプロイする

タスク	説明	必要なスキル
Lambda 関数を作成します。	1. にサインインし AWS Management Console、AWS Lambda コンソールを開きます。 2. [関数] ページで、[関数の作成] を選択します。 3. [一から作成] を選択します。 4. 基本情報ペインの関数名に名前を入力します。 5. [ランタイム] には、[Python 3.12] を選択します。 6. アーキテクチャを x86_64 に設定したままにします。 7. [関数の作成] を選択してください。 8. [コード] タブを選択します。	AWS 全般

タスク	説明	必要なスキル
	9. ファイルエクスプローラーで、lambda_function.py を選択します。 10.このパターンの追加情報セクションにあるサンプルコードを lambda_function.py タブに貼り付けます。サンプルコードをカスタマイズして、`evaluate_change_notification_compliance` 関数内のカスタム評価ロジックを識別します。 11[デプロイ] を選択します。	

タスク	説明	必要なスキル
Lambda 関数の実行ロールにアクセス許可を追加します。	1. ナビゲーションペインで、[関数] を選択します。 2. 作成した関数を選択します。 3. [設定]、[アクセス権限] の順に選択します。 4. （ AWS Identity and Access Management IAM) コンソールでロールを開くロール名を選択します。 5. アクセス許可ポリシーで、アクセス許可の追加を選択し、インラインポリシーの作成を選択します。 6. [JSON] を選択します。 7. 次のポリシーをポリシーエディタに貼り付けます。これにより、Lambda 関数は次のことを実行できます。 • サブネットタグの詳細を取得します。 • コンプライアンスの結果を に送信します AWS Config。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [`	AWS 全般

タスク	説明	必要なスキル

	<pre>"config:PutEvaluations",
	"ec2:DescribeSubnets"
	],
	"Resource": "*",
	"Effect": "Allow"
	}
	]
	}</pre>
	8. [次へ] を選択します。
	9. ポリシーの名前を入力し、[Create policy] (ポリシーの作成) を選択します。	

Lambda 関数の Amazon リソースネーム (ARN) を取得します。	1. Lambdaのコンソールを開きます。 2. ナビゲーションペインで、[関数] を選択します。 3. 作成した関数を選択します。 4. 関数の概要セクションの関数 ARN で、値をコピーします。	AWS 全般

タスク	説明	必要なスキル
AWS Config カスタムルールを作成します。	1. AWS Config コンソールを開きます。 2. [Rules] (ルール) ページで、[Add rule] (ルールの追加) を選択します。 3. ルールタイプの指定ページで、カスタム Lambda ルールの作成を選択し、次へを選択します。 4. [ルールの設定] ページで、次の手順を実行します。 a. 名前と説明を入力します。 b. AWS Lambda 関数 ARN の場合は、以前にコピーした ARN を貼り付けます。 c. [Trigger type] (トリガータイプ)で、[When configuration changes] (設定変更時)を選択します。 d. 変更の範囲 で、リソースを選択します。 e. リソースタイプで、AWS EC2 NetworkInterface を選択します。 f. [次へ] を選択します。 5. 確認と作成ページでルールを確認し、保存を選択します。	AWS 全般

タスク	説明	必要なスキル
通知を設定します。	1. Amazon Amazon SNSトピックを作成するには、「Amazon SNS トピックの作成」の手順に従います。 2. Amazon SNSトピックへのサブスクライブ」の手順に従って、Amazon SNS トピックの通知を受信するエンドポイントを設定します。 3. 「 を使用して非準拠 AWS リソースのカスタム Amazon EventBridge ルールを設定して、 リソースが非準拠である場合に通知する方法 AWS Config」の手順に従います。 EventBridge	AWS 全般

ソリューションをテストする

タスク	説明	必要なスキル
準拠リソースを作成します。	1. パブリックサブネットでサポートされているリソースの 1 つを作成するには、次の手順に従います。 • NAT ゲートウェイを作成する • Network Load Balancer の使用開始 • Application Load Balancer の作成	AWS 全般

タスク	説明	必要なスキル
	2. リソースが作成されると、AWS Config カスタムルールはリソースに関連付けられた Elastic Network Interface を評価します。これらのネットワークインターフェイスを としてマークしますCOMPLIANT 。AWS Config 以下の手順に従って、 でリソースを表示できます。 a. AWS Config コンソールを開きます。 b. ルールページで、ルールを選択します。 c. ルールの詳細ページで、ページの下部に移動します。 d. 対象範囲内のリソースで、コンプライアンスを選択します。作成されたネットワークインターフェイスIDs が表示されていることを確認します。 e. ネットワークインターフェイス設定の詳細については、リソース ID を選択します。	

タスク	説明	必要なスキル
非準拠のリソースを作成します。	1. 次の手順を使用して、パブリックサブネットに非準拠のリソースを作成します。 • Amazon EC2 インスタンスを起動する • Amazon Relational Database Service (Amazon RDS) データベースインスタンスの作成 • VPC エンドポイントを作成する 2. リソースが作成されると、AWS Config カスタムルールはリソースに関連付けられた Elastic Network Interface を評価します。これらのネットワークインターフェイスを としてマークしますNON_COMPLIANT 。 AWS Config 以下の手順に従って、 でリソースを表示できます。 a. AWS Config コンソールを開きます。 b. ルールページで、ルールを選択します。 c. ルールの詳細ページで、ページの下部に移動します。 d. 対象範囲内のリソースで、NonCompliantを選択します。作成され	AWS 全般

タスク	説明	必要なスキル
	たネットワークインターフェイスIDs が表示されていることを確認します。 e. ネットワークインターフェイス設定の詳細については、リソース ID を選択します。 3. Amazon SNS で設定したエンドポイントで通知を受信していることを確認します。	
適用されないリソースを作成します。	1. プライベートサブネットで、Elastic Network Interface を必要とするリソースを作成します。 2. リソースが作成されると、AWS Config カスタムルールはリソースに関連付けられた Elastic Network Interface を評価します。これらのネットワークインターフェイスを としてマークしますNOT_APPLICABLE 。これらのリソースは AWS Config コンソールには表示されません。	AWS 全般

関連リソース

AWS ドキュメント

- セットアップ AWS Config

- AWS Config カスタムルール
- の ABAC AWS
- パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする

その他の AWS リソース

- 大規模な設定コンプライアンスの自動化 AWS
- ゲートウェイロードバランサーを使用した分散型検査アーキテクチャ

追加情報

以下は、デモンストレーション目的で提供される Lambda 関数の例です。

```
import boto3
import json
import os

# Init clients
config_client = boto3.client('config')
ec2_client = boto3.client('ec2')

def lambda_handler(event, context):

    # Init values
    compliance_value = 'NOT_APPLICABLE'
    invoking_event = json.loads(event['invokingEvent'])
    configuration_item = invoking_event['configurationItem']

    status = configuration_item['configurationItemStatus']
    eventLeftScope = event['eventLeftScope']

    # First check if the event configuration applies. Ex. resource event is not delete
    if (status == 'OK' or status == 'ResourceDiscovered') and not eventLeftScope:
        compliance_value = evaluate_change_notification_compliance(configuration_item)


    config_client.put_evaluations(
       Evaluations=[
           {
```

```
                'ComplianceResourceType': invoking_event['configurationItem']
['resourceType'],
                'ComplianceResourceId': invoking_event['configurationItem']
['resourceId'],
                'ComplianceType': compliance_value,
                'OrderingTimestamp': invoking_event['configurationItem']
['configurationItemCaptureTime']
            },
        ],
        ResultToken=event['resultToken'])

# Function with the logs to evaluate the resource
def evaluate_change_notification_compliance(configuration_item):
    is_in_scope = is_in_scope_subnet(configuration_item['configuration']['subnetId'])

    if (configuration_item['resourceType'] != 'AWS::EC2::NetworkInterface') or not
 is_in_scope:
        return 'NOT_APPLICABLE'

    else:
        alb_condition = configuration_item['configuration']['requesterId'] in ['amazon-
elb']
        nlb_condition = configuration_item['configuration']['interfaceType'] in
 ['network_load_balancer']
        nat_gateway_condition = configuration_item['configuration']['interfaceType'] in
 ['nat_gateway']

        if alb_condition or nlb_condition or nat_gateway_condition:
            return 'COMPLIANT'
    return 'NON_COMPLIANT'

# Function to check if elastic network interface is in public subnet
def is_in_scope_subnet(eni_subnet):

    subnet_description = ec2_client.describe_subnets(
        SubnetIds=[eni_subnet]
    )

    for subnet in subnet_description['Subnets']:
        for tag in subnet['Tags']:
            if tag['Key'] == os.environ.get('TAG_KEY') and tag['Value'] ==
 os.environ.get('TAG_VALUE'):
                return True
```

```
    return False
```

パブリックサブネットの予防的属性ベースのアクセスコントロールをデプロイする

作成者: Joel Alfredo Nunez Gonzalez (AWS) と Samuel Ortega Sancho (AWS)

概要

集中型ネットワークアーキテクチャでは、検査とエッジ仮想プライベートクラウド (VPC) が、インターネットとの間のトラフィックなどの、すべてのインバウンドトラフィックとアウトバウンドトラフィックを集中させます。ただし、これによりボトルネックが発生したり、AWS Service Quotasの制限に達したりする可能性があります。ネットワークエッジセキュリティを VPC のワークロードと一緒にデプロイして、一般的な集中型アプローチと比較して、これまでにないスケーラビリティを提供します。これは分散型エッジアーキテクチャと呼ばれます。

パブリックサブネットをワークロードアカウントにデプロイすることには利点がありますが、アタックサーフェスが増えるため、新たなセキュリティリスクも生じます。これらの VPC のパブリックサブネットでは、アプリケーションロードバランサーや NAT ゲートウェイなどのElastic Load Balancing (ELB) リソースのみをデプロイすることを推奨します。専用のパブリックサブネットでロードバランサーと NAT ゲートウェイの使用は、インバウンドトラフィックとアウトバウンドトラフィックのきめ細かな制御に役立ちます。

属性ベースのアクセス制御 (ABAC)は、部署、役職、チーム名など、ユーザーの属性に基づいて、きめ細かなアクセス許可を設定する方法です。詳細については、「AWS のための ABAC」 をご参照ください。ABAC は、ワークロードアカウントのパブリックサブネットにガードレールを提供することができます。これにより、アプリケーションチームは、」インフラストラクチャのセキュリティを損なうことなく機敏に対応できます。

このパターンでは、AWS Organization の「サービスコントロールポリシー (SCP)」 と AWS 識別と管理(IAM) の「ポリシー」 を通じて ABAC を実装することによる、パブリックサブネットを保護する方法を説明します。SCP は、組織のメンバーアカウントまたは組織単位 (OU) のいずれかに適用されます。これらの ABAC ポリシーにより、ユーザーはターゲットサブネットに NAT ゲートウェイをデプロイし、EC2 インスタンスや Elastic Network Interface などの他の Amazon Elastic Compute Cloud (Amazon EC2) リソースをデプロイできなくなります。

前提条件と制限

前提条件

- AWS Organizations 内の組織

- AWS Organizations ルートアカウントに対する管理アクセス
- 組織での、SCP をテストするためのアクティブメンバーアカウントまたは OU

制約事項

- このソリューションの SCP は、サービスにリンクされたロールを使用する AWS サービスが、ターゲットサブネットにリソースをデプロイすることを阻止しません。これらのサービスの例としては、Elastic Load Balancing (ELB)、Amazon Elastic Container Service (Amazon ECS)、および Amazon Relational Database Service (Amazon RDS) があります。詳細については、AWS Organizations のドキュメントの「許可に対する SCP の影響」 を参照してください。これらの例外を検出するためのセキュリティコントロールを実装します。

アーキテクチャ

ターゲットテクノロジースタック

- AWS アカウントまたは AWS Organizations の OU に適用される SCP
- 次の IAM ロールは:
  - `AutomationAdminRole` — SCPの実装後にサブネットタグを変更し、VPC リソースを作成するために使用されます
  - `TestAdminRole` — SCP が、管理者権限を持つプリンシパルを含む他の IAM プリンシパルが、`AutomationAdminRole` 向けのアクションを実行することを妨げているかどうかをテストするために使用されます

ターゲット アーキテクチャ

1. ターゲットアカウントに `AutomationAdminRole` IAMロールを作成します。このロールにはネットワークリソースを管理する権限があります。このロール専用の以下の権限に注意します。
   - このロールは VPC とパブリックサブネットを作成できます。
   - このロールはターゲットサブネットのタグアサインを変更できます。
   - このロールはその自らの権限を管理できます。
2. AWS Organizations では、ターゲットの AWS アカウントまたは OU に SCP を適用します。サンプルポリシーについて、このパターンの「追加情報」 を参照してください。

3. CI/CD パイプライン内のユーザーまたはツールは、AutomationAdminRoleのロールを引き受けて、 SubnetTypeのタグをターゲットサブネットに適用します。
4. 他の IAM ロールを引き受けることで、組織の IAM プリンシパルは、ターゲットサブネットの NAT ゲートウェイ、およびルートテーブルなどの AWS アカウントで許可されているその他のネットワークリソースを管理できます。IAMポリシーを使用して、これらの許可を与します。詳細については、「Amazon VPCの識別とアクセス管理」を参照してください。

自動化とスケール

パブリックサブネットを保護するには、対応する「AWS タグ」 を適用する必要があります。SCP が適用された後、承認済のユーザーが SubnetType:IFA タグを持つサブネットに作成できる Amazon EC2 リソースの種類は NAT ゲートウェイだけです(IFA はインターネット向けのアセットを指します)。SCP は、インスタンスや Elastic Network Interface などの他の Amazon EC2 リソースの作成を防止します。これらのタグがパブリックサブネットに適切に適用されるように、VPC リソースの作成には AutomationAdminRoleロールを引き受ける CI/CD パイプラインを使用することをお勧めします。

ツール

AWS サービス

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Organizations は、作成して一元管理している複数の AWS アカウントを 1 つの組織に統合するためのアカウント管理サービスです。AWS Organizations では、サービスコントロールポリシー (SCP) は、組織のアクセス許可の管理に使用できる組織ポリシーの一種です。
- Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で AWS リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。

エピック

SCP を適用します

タスク	説明	必要なスキル
テスト管理者ロールを作成します。	管理者アカウントで、`TestAdminRole` という名前の IAM ロールを作成します。AdministratorAccess AWS マネージド IAM ポリシーを新しいロールにアタッチします。手順については、IAM ドキュメントの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。	AWS 管理者
自動化管理者ロールを作成します。	1. 管理者アカウントで、`AutomationAdminRole` という名前の IAM ロールを作成します。 2. AdministratorAccess AWS マネージド IAM ポリシーを新しいロールにアタッチします。 以下は、`000000000000` アカウントからのロールをテストするために使用できる、信頼ポリシーの例です。 `{ "Version": "2012-10-17", "Statement": [ {`	AWS 管理者

タスク	説明	必要なスキル
	"Effect": "Allow", "Principal": { "AWS": ["arn:aws:iam::000000000000:root"] }, "Action": "sts:AssumeRole", "Condition": {} }] }	
SCP を作成して添付します。	1. 「追加情報」 セクションにあるサンプルコードを使用して、セキュリティコントロールポリシーを作成します。手順については、AWS Organizations ドキュメントの「SCP の作成」 を参照してください。 2. SCP をターゲット AWS アカウントまたは OU にアタッチします。手順については、AWS Organizations のドキュメントの「サービスコントロールポリシーのアタッチとデタッチ」 を参照してください。	AWS 管理者

SCP をテストします

タスク	説明	必要なスキル
VPC またはサブネットを作成します。	1. ターゲット AWS の TestAdminRole アカウントでロールを引き受けます。 2. VPC を作成するか、既存の VPC に新しいパブリックサブネットを作成してみます。手順については、Amazon VPC ドキュメントの「VPC、サブネット、他の VPC リソースを作成」を参照してください。これらのリソースは作成できないはずです。 3. AutomationAdminRole のロールを引き受けて、前のステップをやり直します。これでネットワークリソースを作成できます。	AWS 管理者
タグを管理します。	1. ターゲット AWS の TestAdminRole アカウントで、ロールを引き受けます。 2. 使用可能なパブリックサブネットに SubnetType:IFA タグを追加します。このタグは追加できるはずです。AWS コマンドラインインターフェイス (AWS CLI)を介してタグを追加する方法の手順	AWS 管理者

タスク	説明	必要なスキル
	は、AWS CLI コマンドリファレンスの「create-tags」 を参照してください。 3. 認証情報を変更することなく、このサブネットにアサインされた SubnetType:IFA のタグを変更します。このタグは変更できないはずです。 4. AutomationAdminRole のロールを引き受けて、前のステップをやり直します。このロールは、このタグを追加したり変更したりできるはずです。	

タスク	説明	必要なスキル
ターゲットサブネットでリソースをデプロイします。	1. `TestAdminRole` のロールを引き受けます。 2. `SubnetType:IFA` のタグが付けられたパブリックサブネットの場合、EC2 インスタンスを作成してみてください。手順については、Amazon EC2 ドキュメントの「インスタンスの起動」を参照してください。このサブネットでは、NAT ゲートウェイ以外の Amazon EC2 リソースを作成、変更、削除ができないはずです。 3. 同じサブネット内に、NAT ゲートウェイを作成します。手順については、Amazon VPC ドキュメントの「NAT ゲートウェイの作成」を参照してください。このサブネットでは NAT ゲートウェイの作成、変更、削除ができるはずです。	AWS 管理者

タスク	説明	必要なスキル
AutomationAdminRole ロールを管理します。	1. TestAdminRole のロールを引き受けます。 2. AutomationAdminRole のロールを変更してみます。手順については、IAM ドキュメントの「ロールの変更」を参照してください。このロールは変更できないはずです。 3. AutomationAdminRole のロールを引き受けて、前のステップをやり直します。これでロールを変更できます。	AWS 管理者

クリーンアップ

タスク	説明	必要なスキル
デプロイされたリソースをクリーンアップします。	1. AWS アカウントまたは OU から SCP をデタッチします。手順については、AWS Organizations ドキュメントの「SCP のデタッチ」を参照してください。 2. SCP を削除します。手順については、「SCP の削除」(AWS Organizations ドキュメント) を参照してください。 3. AutomationAdminRole のロールと TestAdminRole のロールを削除しま	AWS 管理者

タスク	説明	必要なスキル
	す。手順については、IAM ドキュメントの「ロールの削除」 を参照してください。 4. このソリューション用に作成した VPC やサブネットなどのネットワークリソースをすべて削除します。	

関連リソース

AWS ドキュメント

- 「SCPのアタッチとデタッチ」
- 「SCPの作成、更新、削除」
- AWS Config を使用してパブリックサブネットの検出属性ベースのアクセスコントロールをデプロイする
- 「発見的コントロール」
- 「サービス認可リファレンス」
- 「AWS リソースへのタグ付け」
- 「AWS 向けの ABAC とは?」

「追加の AWS リファレンス」

- AWS Organizations のサービスコントロールポリシーを使用した認可に使用されるリソースタグの保護 (AWS ブログ記事)

追加情報

以下のサービスコントロールポリシーは、このアプローチを組織でテストするために使用できる例です。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCActions",
      "Effect": "Deny",
      "Action": [
        "ec2:CreateVPC",
        "ec2:CreateRoute",
        "ec2:CreateSubnet",
        "ec2:CreateInternetGateway",
        "ec2:DeleteVPC",
        "ec2:DeleteRoute",
        "ec2:DeleteSubnet",
        "ec2:DeleteInternetGateway"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:*"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    },
    {
      "Sid": "AllowNATGWOnIFASubnet",
      "Effect": "Deny",
      "NotAction": [
        "ec2:CreateNatGateway",
        "ec2:DeleteNatGateway"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:subnet/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIfExists": {
          "aws:ResourceTag/SubnetType": "IFA"
        },
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    },
    {
```

```
      "Sid": "DenyChangesToAdminRole",
      "Effect": "Deny",
      "NotAction": [
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoleTags"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AutomationAdminRole"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    },
    {
      "Sid": "allowbydefault",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Terraform を使用して AWS WAF ソリューションのセキュリティオートメーションをデプロイする

作成者: Dr. Rahul Sharad Gaikwad (AWS)、Tamilselvan P (AWS)

概要

AWS WAF は、ウェブアクセスコントロールリスト (ACL) で定義してデプロイするカスタマイズ可能なルールを使用して、一般的な悪用からアプリケーションを保護するのに役立つウェブACLs。AWS WAF ルールの設定は、特に専用のセキュリティチームを持たない組織では難しい場合があります。このプロセスを簡素化するために、Amazon Web Services (AWS) は ソリューションのセキュリティオートメーション AWS WAFを提供しています。これにより、ウェブベースの攻撃をフィルタリングする一連の AWS WAF ルールを含む単一のウェブ ACL が自動的にデプロイされます。Terraform のデプロイ時に、どの保護機能を含めるかを指定できます。このソリューションをデプロイすると、 は既存の Amazon CloudFront ディストリビューションまたは Application Load Balancer へのウェブリクエスト AWS WAF を検査し、ルールに一致しないリクエストをブロックします。

AWS WAF ソリューションのセキュリティオートメーションは、「 Security Automations for AWS WAF Implementation Guide」の指示 AWS CloudFormation に従って を使用してデプロイできます。このパターンは、HashiCorp Terraform を推奨 Infrastructure as Code (IaC) ツールとして使用してクラウドインフラストラクチャをプロビジョニングおよび管理する組織にとって、代替のデプロイオプションとなります。このソリューションをデプロイすると、Terraform は自動的にクラウドに変更を適用し、 AWS WAF 設定と保護機能をデプロイして設定します。

前提条件と制限

前提条件

- アクティブ AWS アカウント。
- AWS Command Line Interface （ AWS CLI) バージョン 2.4.25 以降、必要なアクセス許可でインストールおよび設定されている。詳細については、「開始方法 (AWS CLI ドキュメント ）」を参照してください。
- Terraform バージョン 1.1.9 以降がインストールされ、設定されています。詳細については、「Terraform のインストール (Terraform ドキュメント)」を参照してください。

アーキテクチャ

ターゲット アーキテクチャ

このパターンは、 AWS WAF ソリューションのセキュリティオートメーションをデプロイします。ターゲットアーキテクチャの詳細については、「 Security Automations for AWS WAF Implementation Guide」の「アーキテクチャの概要」を参照してください。このデプロイの AWS Lambda オートメーション、アプリケーションログパーサー、 AWS WAF ログパーサー、IP リストパーサー、アクセスハンドラーの詳細については、「 Security Automations for AWS WAF Implementation Guide」の「コンポーネントの詳細」を参照してください。

Terraform デプロイ

`terraform apply` を実行すると、Terraform は次のことを行います。

1. Terraform は、testing.tfvars ファイルからの入力に基づいて AWS Identity and Access Management (IAM) ロールと Lambda 関数を作成します。
2. Terraform は、testing.tfvars ファイルからの入力に基づいて AWS WAF ACL ルールと IP セットを作成します。
3. Terraform は、testing.tfvars ファイルからの入力に基づいて、Amazon Simple Storage Service (Amazon S3) バケット、Amazon EventBridge ルール、 AWS Glue データベーステーブル、Amazon Athena ワークグループを作成します。
4. Terraform は AWS CloudFormation スタックをデプロイしてカスタムリソースをプロビジョニングします。
5. Terraform は、test.tfvars ファイルからの指定された入力に基づいて Amazon API Gateway リソースを作成します。

自動化とスケール

このパターンを使用して、複数の の AWS WAF ルールを作成し AWS アカウント AWS リージョン 、 AWS WAF ソリューションのセキュリティオートメーションを AWS クラウド 環境全体にデプロイできます。

ツール

AWS のサービス

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。

- AWS WAF は、保護されたウェブアプリケーションリソースに転送される HTTP および HTTPS リクエストをモニタリングするのに役立つウェブアプリケーションファイアウォールです。

その他のサービス

- Git はオープンソースの分散型バージョン管理システムです。
- HashiCorp Terraform は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するのに役立つコマンドラインインターフェイスアプリケーションです。

コードリポジトリ

このパターンのコードは、GitHub AWS WAF Automation Using Terraform リポジトリで入手できます。

ベストプラクティス

- 静的ファイルを個別の Amazon S3 バケットに配置します。
- 変数をハードコーディングすることは避けてください。
- カスタムスクリプトの使用を制限してください。
- 命名規則を採用してください。

エピック

ローカルワークステーションをセットアップする

タスク	説明	必要なスキル
Git をインストールする。	「Getting started」(Git Web サイト) の指示に従って、ローカルワークステーションに Git をインストールします。	DevOps エンジニア
リポジトリをクローン作成します。	ローカルワークステーションで、次のコマンドを入力してコードリポジトリのクローンを作成します。	DevOps エンジニア

タスク	説明	必要なスキル
	`git clone https://github.com/aws-samples/aws-waf-automation-terraform-samples.git`	
変数を更新してください。	1. 次のコマンドを入力して、クローンされたディレクトリに移動します。 `cd terraform-aws-waf-automation` 2. 任意のテキストエディタで testing.tfvars ファイルを開きます。 3. testing.tfvars ファイル内の変数の値を更新します。 4. ファイルを保存して閉じます。	DevOps エンジニア

Terraform を使用してターゲットアーキテクチャをプロビジョニングします。

タスク	説明	必要なスキル
Terraform の設定を初期化します。	次のコマンドを入力して、Terraform 設定ファイルを含む作業ディレクトリを初期化します。 `terraform init`	DevOps エンジニア
Terraform プランをプレビューしてください。	次のコマンドを入力します。Terraform は設定ファイルを評価して、宣言された	DevOps エンジニア

タスク	説明	必要なスキル
	リソースのターゲット状態を判断します。次に、ターゲットの状態を現在の状態と比較し、計画を作成します。 `terraform plan -var-file ="testing.tfvars"`	
プランを検証してください。	プランを確認し、ターゲットに必要なアーキテクチャが設定されていることを確認します AWS アカウント。	DevOps エンジニア
ソリューションをデプロイします。	1. プランを適用するには、次のコマンドを入力します。 `terraform apply -var-file="testing .tfvars"` 2. yes を入力して確定します。Terraform は、構成ファイルに宣言されている目標状態を達成するために、インフラストラクチャを作成、更新、または破棄します。シーケンスの詳細については、このパターンの「アーキテクチャ」セクションにある「Terraform デプロイ」を参照してください。	DevOps エンジニア

検証とクリーンアップ

タスク	説明	必要なスキル
変更を確認します。	1. Terraform コンソールで、出力が期待どおりの結果であることを確認します。 2. AWS Management Console にサインインします。 3. Terraform コンソールの出力が に正常にデプロイされたことを確認します AWS アカウント。	DevOps エンジニア
(オプション) インフラストラクチャをクリーンアップします。	このソリューションによって行われたすべてのリソースと設定の変更を削除するには、次の操作を行います。 1. Terraform コンソールで、次のコマンドを入力します。 `terraform destroy -var-file="testing .tfvars”` 2. yes を入力して確定します。	DevOps エンジニア

トラブルシューティング

問題	ソリューション
`WAFV2 IPSet: WAFOptimisticLockException` エラー	`terraform destroy` コマンドの実行時にこのエラーが発生した場合は、IP セットを手動で削除する必要があります。手順について

問題	ソリューション
	は、「IP セットの削除 (AWS WAF ドキュメント)」を参照してください。

関連リソース

AWS リファレンス

- Security Automations for AWS WAF Implementation ガイド
- のセキュリティオートメーション AWS WAF (AWS ソリューションライブラリ)
- AWS WAF よくある質問のセキュリティオートメーション

Terraform のリファレンス

- Terraform バックエンド設定
- Terraform AWS プロバイダー - ドキュメントと使用状況
- Terraform AWS プロバイダー (GitHub リポジトリ)

CA 証明書の有効期限が切れている Amazon RDS および Aurora データベースインスタンスを検出する

作成者: Stephen DiCato (AWS) と Eugene Shifer (AWS)

概要

セキュリティのベストプラクティスとして、アプリケーションサーバーとリレーショナルデータベース間で転送中のデータを暗号化することをお勧めします。SSL または TLS を使用して、データベース (DB) インスタンスまたはクラスターへの接続を暗号化できます。これらのプロトコルは、アプリケーションとデータベース間の機密性、完全性、信頼性を提供するのに役立ちます。データベースは、認証機関 (CA) によって発行され、サーバー ID 検証を実行するために使用されるサーバー証明書を使用します。SSL または TLS は、デジタル署名を検証し、証明書の有効期限が切れていないことを確認することで、証明書の信頼性を検証します。

では AWS Management Console、Amazon Relational Database Service (Amazon RDS) と Amazon Aurora は、証明書の更新を必要とする DB インスタンスに関する通知を提供します。ただし、これらの通知を確認するには、それぞれにログイン AWS アカウント し、それぞれの のサービスコンソールに移動する必要があります AWS リージョン。で組織として管理 AWS アカウント されている多くの で証明書の有効性を評価する必要がある場合、このタスクはより複雑になりますAWS Organizations。

このパターンで提供される Infrastructure as Code (IaC) をプロビジョニングすることで、 AWS アカウント または AWS 組織内のすべての Amazon RDS および Aurora DB インスタンスの期限切れの CA 証明書を検出できます。AWS CloudFormation テンプレートは、 AWS Config ルール、 AWS Lambda 関数、および必要なアクセス許可をプロビジョニングします。スタックとして 1 つのアカウントにデプロイすることも、スタックセットとして AWS 組織全体にデプロイすることもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 単一の にデプロイする場合 AWS アカウント：
  - CloudFormation スタックを作成するアクセス許可があることを確認します。
  - ターゲットアカウントで を有効にします AWS Config 。
  - （ オプション) ターゲットアカウントで を有効にします AWS Security Hub 。

- AWS 組織にデプロイする場合：
  - CloudFormation スタックセットを作成するアクセス許可があることを確認します。
  - AWS Organizations 統合で Security Hub を有効にします。
  - このソリューションをデプロイするアカウントで を有効にします AWS Config 。
  - を AWS Config および Security Hub の委任管理者 AWS アカウント として指定します。

制約事項

- Security Hub が有効になっていない個々のアカウントにデプロイする場合は、 AWS Config を使用して検出結果を評価できます。
- AWS Config と Security Hub の委任管理者を持たない組織にデプロイする場合は、個々のメンバーアカウントにログインして結果を表示する必要があります。
- AWS Control Tower を使用して組織内のアカウントを管理および管理する場合は、 AWS Control Tower (CfCT) のカスタマイズを使用して、このパターンで IaC をデプロイします。CloudFormation AWS Control Tower コンソールを使用すると、ガードレールから設定ドリフトが作成され、組織単位 (OUs) またはマネージドアカウントを再登録する必要があります。
- 一部の AWS のサービス は、すべてで利用できるわけではありません AWS リージョン。リージョンの可用性については、「サービスエンドポイントとクォータ」ページを参照して、サービスのリンクを選択します。

アーキテクチャ

個人へのデプロイ AWS アカウント

次のアーキテクチャ図は、単一の 内の AWS リソースのデプロイを示しています AWS アカウント。これは、CloudFormation コンソールから直接 CloudFormation テンプレートを使用して実装されます。Security Hub が有効になっている場合は、 AWS Config または Security Hub で結果を表示できます。Security Hub が有効になっていない場合は、 AWS Config コンソールでのみ結果を表示できます。

図表に示す内容は以下のステップです。

1. CloudFormation スタックを作成します。これにより、Lambda 関数と AWS Config ルールがデプロイされます。ルールと関数の両方が、 AWS Config および ログでリソース評価を発行するために必要な AWS Identity and Access Management (IAM) アクセス許可で設定されます。

2. この AWS Config ルールは検出評価モードで動作し、24 時間ごとに実行されます。
3. Security Hub はすべての AWS Config 検出結果を受け取ります。
4. アカウントの設定に応じて AWS Config、Security Hub または で結果を表示できます。

AWS 組織にデプロイする

次の図は、 AWS Organizations および を通じて管理される複数のアカウントにわたる証明書の有効期限の評価を示しています AWS Control Tower。CloudFormation テンプレートは CfCT を使用してデプロイします。評価結果は、委任管理者アカウントの Security Hub に一元化されます。図に示す AWS CodePipeline ワークフローは、CfCT デプロイ中に発生するバックグラウンドステップを示しています。

図表に示す内容は以下のステップです。

1. CfCT の設定に応じて、管理アカウントで IaC を リポジトリにプッシュするか、IaC の圧縮 (ZIP) ファイルを Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。 AWS CodeCommit
2. CfCT パイプラインはファイルを解凍し、cfn-nag (GitHub) チェックを実行し、CloudFormation スタックセットとしてデプロイします。
3. CfCT マニフェストファイルで指定された設定に応じて、CloudFormation StackSets はスタックを個々のアカウントまたは指定された OUs。これにより、Lambda 関数と AWS Config ルールがターゲットアカウントにデプロイされます。ルールと関数の両方が、 AWS Config および ログでリソース評価を発行するために必要な IAM アクセス許可で設定されます。
4. この AWS Config ルールは検出評価モードで動作し、24 時間ごとに実行されます。
5. AWS Config は、すべての検出結果を Security Hub に転送します。
6. Security Hub の検出結果は、委任管理者アカウントに集約されます。
7. 結果は、委任管理者アカウントの Security Hub で表示できます。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体のライフサイクルを通じてリソースを管理するのに役立ちます。
- AWS Config には、 のリソース AWS アカウント の詳細と設定方法が表示されます。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。An AWS Config rule は、リソースの理想的な構成設定を定義し、AWS リソースがルールの条件に準拠しているかどうかを評価 AWS Config できます。
- AWS Control Tower は、規範的なベストプラクティスに従って、 AWS マルチアカウント環境のセットアップと管理に役立ちます。AWS Control Tower (CfCT) のカスタマイズは、 AWS Control Tower ランディングゾーンをカスタマイズし、 AWS ベストプラクティスとの整合性を保つのに役立ちます。カスタマイズは、CloudFormation テンプレートとサービスコントロールポリシー (SCPs。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。また、セキュリティ業界標準やベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。

その他のツール

- 「Python」は汎用のコンピュータープログラミング言語です。

コードリポジトリ

このパターンのコードは、有効期限が切れる CA 証明書リポジトリを持つ GitHub Detect Amazon RDS インスタンスで使用できます。 https://github.com/aws-samples/config-rds-ca-expiry

ベストプラクティス

以下のリソースのベストプラクティスに従うことをお勧めします。

- を使用した組織単位のベストプラクティス AWS Organizations (AWS クラウド運用と移行ブログ)

- AWS Control Tower で を使用して初期基盤を確立するためのガイダンス AWS (AWS ソリューションライブラリ)
- AWS Control Tower リソースの作成と変更に関するガイダンス (AWS Control Tower ドキュメント)
- CfCT デプロイに関する考慮事項 (AWS Control Tower ドキュメント)

エピック

ソリューションとコードを確認する

タスク	説明	必要なスキル
デプロイ戦略を決定します。	ソリューションとコードを確認して、 AWS 環境にデプロイする方法を決定します。単一のアカウントまたは AWS 組織にデプロイするかどうかを決定します。	アプリ所有者、AWS 全般
リポジトリをクローン作成します。	次のコマンドを入力して、期限切れの CA 証明書リポジトリを持つ Detect Amazon RDS インスタンスのクローンを作成します。 `git clone https://github.com/aws-samples/config-rds-ca-expiry.git`	アプリ開発者、アプリオーナー
Python バージョンを検証します。	1. クローンされたリポジトリの最上位ディレクトリに移動します。 `cd config-rds-ca-expiry`	アプリ開発者、アプリオーナー

タスク	説明	必要なスキル
	2. config-rds-ca-expiry.yaml を開きます。 3. `CertExpirationCheckLambdaFunction` リソースで、Python バージョンがターゲットと互換性があることを確認します AWS リージョン。デフォルトでは、この関数は Python 3.12 を使用します。詳細については、「 で AWS Lambda Python 3.12 のサポートが追加されました」を参照してください。必要に応じて、Python バージョンを更新します。 4. config-rds-ca-expiry.yaml を保存して閉じます。	

解決策をデプロイする

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	CloudFormation テンプレートを AWS 環境にデプロイします。次のいずれかを行います： • 単一の にデプロイする場合は AWS アカウント、「スタックの作成」の手順に従います。 • によって管理されていない組織にデプロイする場合は	アプリ開発者、AWS 管理者、AWS 全般

タスク	説明	必要なスキル
	AWS Control Tower、「スタックセットを作成する」の手順に従います。 • が管理する組織にデプロイする場合は AWS Control Tower、「独自のカスタマイズを構築する」の手順を参照してください。	
デプロイメントを確認する	CloudFormation コンソールで、スタックまたはスタックセットが正常にデプロイされていることを確認します。	AWS 管理者、アプリ所有者

検出結果を確認する

タスク	説明	必要なスキル
AWS Config ルールの検出結果を表示します。	Security Hub で、以下を実行して個々の検出結果のリストを表示します。 1. Security Hub コンソールを開きます。 2. ナビゲーションペインで 調査結果を選択します。 3. フィルターの追加 ボックスに、次のフィルターを追加します。 • コンプライアンスステータスは です FAILED • タイトル: `rds-has-expiring-ca`	AWS 管理者、クラウド管理者、AWS システム管理者

タスク	説明	必要なスキル
	4. [Apply] (適用) を選択します。 Security Hub で、以下を実行して、グループ化された検出結果の合計のリストを表示します AWS アカウント。 1. Security Hub コンソールを開きます。 2. ナビゲーションペインで、[Insights] (インサイト) を選択します。 3. [Create insight] (インサイトの作成) を選択します。 4. インサイトのグループ化属性を選択するには: a. 検索ボックスを選択して、フィルターオプションを表示します。 b. [Group by] (グループ化の条件) を選択します。 c. AwsAccountId を選択します。 d. [Apply] (適用) を選択します。 5. フィルターの追加 ボックスに、次のフィルターを追加します。 • タイトル: `rds-has-expiring-ca` • コンプライアンスステータスは です `FAILED`	

タスク	説明	必要なスキル
	6. [Create insight] (インサイトの作成) を選択します。 7. [インサイトの名前] を入力し、[インサイトの作成] を選択します。 検出結果のリスト AWS Configを表示するには、 AWS Config ドキュメントの「コンプライアンス情報と評価結果の表示」の手順に従います。	

トラブルシューティング

問題	ソリューション
CloudFormation スタックセットの作成または削除が失敗する	AWS Control Tower がデプロイされると、必要なガードレールが適用され、 AWS Config アグリゲータとルールの制御が引き受けられます。これには、CloudFormation による直接的な変更の防止が含まれます。関連するすべてのリソースを含め、この CloudFormation テンプレートを適切にデプロイまたは削除するには、CfCT を使用する必要があります。
CfCT が CloudFormation テンプレートを削除できない	マニフェストファイルに必要な変更を加え、テンプレートファイルを削除した後も CloudFormation テンプレートが存続する場合は、マニフェストファイルに `enable_stack_set_deletion` パラメータが含まれ、値が に設定されていることを確認しますfalse。詳細については、CfCT ドキュメン

問題	ソリューション
	トの「スタックセットの削除」を参照してください。

関連リソース

- SSL/TLS を使用して DB インスタンスまたはクラスターへの接続を暗号化する (Amazon RDS ドキュメント)
- AWS Config カスタムルール (AWS Config ドキュメント)

Step Functions を使用して IAM アクセスアナライザーで IAM ポリシーを動的に生成

作成者: Thomas Scott (AWS), Adil El Kanabi (AWS), Koen van Blijderveen (AWS), and Rafal Pawlaszek (AWS)

概要

注意: AWS CodeCommit は、新規顧客には利用できなくなりました。の既存のお客様は、通常どおりサービスを AWS CodeCommit 引き続き使用できます。詳細はこちら。

最小特権は、タスクを実行するために最低限必要な権限を付与する際の、セキュリティのベストプラクティスです。既にアクティブな Amazon Web Services (AWS) アカウントに最小特権アクセスを実装することは、アクセス許可を変更することで、ユーザーが意図せずに職務を遂行できないようにするため、難しい場合があります。 AWS Identity and Access Management (IAM) ポリシーの変更を実装する前に、アカウントユーザーが実行しているアクションとリソースを理解する必要があります。

このパターンは、チームの生産性を妨げたり低下させたりすることなく、最小特権アクセスの原則を適用できるように設計されています。IAM Access Analyzer と を使用して、アカウントで現在実行されているアクションに基づいて、ロールのup-to-date IAM ポリシーを動的 AWS Step Functions に生成する方法について説明します。新しいポリシーは、現在のアクティビティを許可する一方で、不必要で昇格された権限は削除するように設計されています。生成されたポリシーは、許可ルールと拒否ルールを定義することでカスタマイズでき、ソリューションはカスタムルールを統合します。

このパターンには、 AWS Cloud Development Kit (AWS CDK) または HashiCorp CDK for Terraform (CDKTF) を使用してソリューションを実装するためのオプションが含まれています。その後、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインを使用して、新しいポリシーをロールに関連付けることができます。マルチアカウントアーキテクチャを使用している場合は、ロールの更新された IAM ポリシーを生成する任意のアカウントにこのソリューションをデプロイして、 AWS クラウド 環境全体のセキュリティを強化できます。

前提条件と制限

前提条件

- AWS CloudTrail 証跡が有効になってい AWS アカウント るアクティブな 。
- 以下に対する IAM 権限:

  - Step Functions ワークフローを作成してデプロイします。詳細については、「 のアクション、リソース、および条件キー AWS Step Functions」(Step Functions ドキュメント) を参照してください。
  - AWS Lambda 関数を作成します。詳細については、「実行ロールおよびユーザーアクセス許可」（ Lambda ドキュメント ）を参照してください。
  - IAM ロールを作成します。詳細については、「ロールを作成して、IAM ユーザーにアクセス許可を委任する」(IAM ドキュメント)を参照してください。
- npm がインストールされています。詳細については、「Node.js と npm のダウンロードとインストール」 (npm ドキュメント) を参照してください。
- このソリューションを AWS CDK （ オプション 1) でデプロイする場合 :
  - AWS CDK Toolkit、インストールおよび設定済み。詳細については、「 のインストール AWS CDK (AWS CDK ドキュメント ）」を参照してください。
- このソリューションを CDKTF でデプロイ (オプション 2):
  - CDKTF がインストールされ、設定されています。詳細については、「Terraform 用 CDK のインストール」(CDKTF ドキュメント)」を参照してください。
  - Terraform がインストールされ、設定されています。詳細については、「使用の開始」（ Terraform ドキュメント ）を参照してください。
- AWS Command Line Interface （ AWS CLI) ローカルにインストールされ、 用に設定されています AWS アカウント。詳細については、「 の最新バージョンのインストールまたは更新 AWS CLI (AWS CLI ドキュメント ）」を参照してください。

制約事項

- このパターンでは、新しい IAM ポリシーはロールには適用されません。このソリューションの最後に、新しい IAM ポリシーが AWS CodeCommit リポジトリに保存されます。CI/CD パイプラインを使用して、アカウント内のロールにポリシーを適用できます。

アーキテクチャ

ターゲットアーキテクチャ

1. 定期的にスケジュールされる Amazon EventBridge イベントルールにより、Step Functions のワークフローが開始されます。この再生スケジュールは、このソリューションの設定の一環として定義します。
2. Step Functions ワークフローでは、Lambda 関数が CloudTrail ログのアカウントアクティビティを分析するときに使用する日付範囲を生成します。
3. 次のワークフローステップでは IAM アクセスアナライザー API を呼び出し、ポリシーの生成を開始します。
4. IAM Access Analyzer は、セットアップ時に指定したロールの Amazon リソースネーム (ARN) を使用して、指定された日付レート内のアクティビティについて CloudTrail ログを分析します。アクティビティに基づいて、IAM Access Analyzer は、指定された日付範囲にロールが使用するアクションとサービスのみを許可する IAM ポリシーを生成します。このステップが完了すると、ジョブ ID が生成されます。
5. 次のワークフローステップでは、30 秒ごとにジョブ ID がチェックされます。ジョブ ID が検出されると、このステップではジョブ ID を使用して IAM Access Analyzer API を呼び出し、新しい IAM ポリシーを取得します。IAM アクセスアナライザーはポリシーを JSON ファイルとして返します。
6. 次のワークフローステップでは、<IAM ロール name>/policy.json ファイルを Amazon Simple Storage Service (Amazon S3) バケットに配置します。この S3 バケットは、このソリューションの設定の一環として定義します。
7. Amazon S3 イベント通知は Lambda 関数を開始します。
8. Lambda 関数は S3 バケットからポリシーを取得し、allow.json ファイルと deny.json ファイルで定義したカスタムルールを統合し、更新されたポリシーを CodeCommit にプッシュします。このソリューションの設定の一部として、CodeCommit リポジトリ、ブランチ、およびフォルダパスを定義します。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS CDK Toolkit は、 AWS Cloud Development Kit (AWS CDK) アプリの操作に役立つコマンドラインクラウド開発キットです。
- AWS CloudTrail は、 のガバナンス、コンプライアンス、運用リスクを監査するのに役立ちます AWS アカウント。

- AWS CodeCommit は、独自のソース管理システムを管理することなく、Git リポジトリをプライベートに保存および管理するためのバージョン管理サービスです。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。このパターンは 「IAM の特徴量である IAM Access Analyzer」 を使用して CloudTrail ログを分析して IAM エンティティ (ユーザーまたはロール) が使用しているアクションとサービスを特定し、そのアクティビティに基づいて IAM ポリシーを生成します。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- Amazon Simple Storage Service (Amazon S3) は、任意の量のデータを保存、保護、取得する上で役立つクラウドベースのオブジェクトストレージサービスです。
- AWS Step Functions は、 AWS Lambda 関数とその他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。このパターンでは、Step Functions で AWS SDK サービス統合を使用して、ワークフローからサービス API アクションを呼び出します。

その他のツール

- 「CDK for Terraform (CDKTF)」は、Python や Typescript などの一般的なプログラミング言語を使用して、Infrastructure as Code (IaC) として定義するのに役立ちます。
- 「Lerna」は、同じリポジトリから複数の JavaScript または TypeScript パッケージを管理および公開するためのビルドシステムです。
- Node.js は、スケーラブルなネットワークアプリケーションを構築するために設計された、イベント駆動型の JavaScript ランタイム環境です。
- 「npm」は Node.js 環境で動作するソフトウェアレジストリで、パッケージの共有や借用、プライベートパッケージのデプロイ管理に使用されます。

コードリポジトリ

このパターンのコードは、GitHub「Automated IAM Access Analyzer ロールポリシージェネレーター」リポジトリにあります。

エピック

デプロイの準備

タスク	説明	必要なスキル
リポジトリを複製します。	次のコマンドは、Automated IAM Access Analyzer Role Policy Generator (GitHub) リポジトリをクローンします。 `git clone https://github.com/aws-samples/automated-iam-access-analyzer.git`	アプリ開発者
Lerna をインストールします。	以下のコマンドで Lerna をインストールします。 `npm i -g lerna`	アプリ開発者

依存関係を設定します。	次のコマンドでリポジトリの依存関係をインストールします。
	`cd automated-iam-access-analyzer/`
	`npm install && npm run bootstrap`
		アプリ開発者

コードをビルドします。	次のコマンドは、Lambda 関数の zip パッケージをテスト、ビルド、準備します。
	`npm run test:code`
	`npm run build:code`
	`npm run pack:code`
		アプリ開発者

タスク	説明	必要なスキル
コンストラクトをビルドします。	次のコマンドは、 AWS CDK と CDKTF の両方のインフラストラクチャ合成アプリケーションを構築します。 `npm run build:infra`	
任意のカスタム権限を設定します。	複製したリポジトリの repo フォルダーで allow.json ファイルと deny.json ファイルを編集して、ロールのカスタム権限を定義します。allow.json ファイルと deny.json ファイルに同じ権限が含まれている場合は、拒否権限が適用されます。	AWS 管理者、アプリ開発者

オプション 1 – を使用してソリューションをデプロイする AWS CDK

タスク	説明	必要なスキル
AWS CDK スタックをデプロイします。	次のコマンドは、 を介してインフラストラクチャをデプロイします AWS CloudFormation。以下のパラメータを指定します。 • `<NAME_OF_ROLE>` — 新しいポリシーを作成している IAM ロールの ARN。 • `<TRAIL_ARN>` — ロールアクティビティが保存されている CloudTrail トレイルの ARN。	アプリ開発者

タスク	説明	必要なスキル

	• <CRON_EXPRESSION_TO_RUN_SOLUTION> — ポリシーの再生スケジュールを定義する Cron 式。Step Functions ワークフローはこのスケジュールで実行されます。
	• <TRAIL_LOOKBACK> — ロールの権限を評価する際に履歴を振り返る期間 (日単位) 。
	
	`cd infra/cdk`
	`cdk deploy –-parameters roleArn=<NAME_OF_ROLE> \`
	`–-parameters trailArn=<TRAIL_ARN> \`
	`--parameters schedule=<CRON_EXPRESSION_TO_RUN_SOLUTION> \`
	`[ --parameters trailLookBack=<TRAIL_LOOKBACK> ]`
	
	Note
	角括弧はオプションのパラメータを示します。	

タスク	説明	必要なスキル
(オプション) 新しいポリシーが適用されるまでお待ちください。	トレイルにそのロールの履歴アクティビティが妥当な量含まれていない場合は、IAM Access Analyzer が正確なポリシーを生成するのに十分なアクティビティがログされていることを確認するまでお待ちください。ロールがアカウント内で一定期間アクティブになっていれば、この待機期間は必要ない可能性があります。	AWS 管理者
生成されたポリシーを手動で確認します。	CodeCommit リポジトリで、生成された.json <ROLE_ARN >ファイルを確認して、許可権限と拒否権限がロールに適していることを確認します。	AWS 管理者

オプション 2 — CDKTF を使用してソリューションをデプロイ

タスク	説明	必要なスキル
Terraform テンプレートを合成します。	以下のコマンドは Terraform テンプレートを合成します。 `lerna exec cdktf synth --scope @aiaa/tfm`	アプリ開発者
Terraform テンプレートをデプロイします。	次のコマンドは、CDKTF で定義されているインフラストラクチャが格納されているディレクトリに移動します。	アプリ開発者

タスク	説明	必要なスキル

	`cd infra/cdktf`
	
	次のコマンドは、インフラストラクチャをターゲットにデプロイします AWS アカウント。以下のパラメータを指定します。
	
	• `<account_ID>` — ターゲットアカウントの ID。
	• `<region>` - ターゲット AWS リージョン。
	• `<selected_role_ARN>` — 新しいポリシーを作成している IAM ロールの ARN。
	• `<trail_ARN>` — ロールアクティビティが保存されている CloudTrail トレイルの ARN。
	• `<schedule_expression>` — ポリシーの再生スケジュールを定義する Cron 式。Step Functions ワークフローはこのスケジュールで実行されます。
	• `<trail_look_back>` — ロールの権限を評価する際に履歴を振り返る期間 (日単位) 。
	
	`TF_VAR_accountId=<account_ID> \`	

タスク	説明	必要なスキル
	TF_VAR_region=<region> \ TF_VAR_roleArns=<selected_role_ARN> \ TF_VAR_trailArn=<trail_ARN> \ TF_VAR_schedule=<schedule_expression> \ [TF_VAR_trailLookBack=<trail_look_back>] \ cdktf deploy **Note** 角括弧はオプションのパラメータを示します。	
(オプション) 新しいポリシーが適用されるまでお待ちください。	トレイルにそのロールの履歴アクティビティが妥当な量含まれていない場合は、IAM Access Analyzer が正確なポリシーを生成するのに十分なアクティビティがログされていることを確認するまでお待ちください。ロールがアカウント内で一定期間アクティブになっていれば、この待機期間は必要ない可能性があります。	AWS 管理者
生成されたポリシーを手動で確認します。	CodeCommit リポジトリで、生成された.json <ROLE_ARN>ファイルを確認して、許可権限と拒否権限がロールに適していることを確認します。	AWS 管理者

関連リソース

AWS リソース

- 「IAM Access Analyzer エンドポイントとクォータ」
- の設定 AWS CLI
- の開始方法 AWS CDK
- 「最小特権のアクセス許可」

その他のリソース

- 「Terraform 用 CDK」 (Terraform ウェブサイト)

AWS CloudFormation テンプレートを使用して Amazon GuardDuty を条件付きで有効にする

作成者: Ram Kandaswamy (AWS)

概要

AWS CloudFormation テンプレートを使用して、Amazon Web Services () アカウントで Amazon GuardDuty を有効にできます。AWSデフォルトでは、CloudFormation を使用して GuardDuty を有効にしようとしたときに GuardDuty が既に有効になっていると、スタックのデプロイは失敗します。ただし、CloudFormation テンプレートの条件を使用して、GuardDuty が既に有効になっているかどうかを確認できます。CloudFormation は、静的値を比較する条件の使用をサポートしていますが、同じテンプレート内の別のリソースプロパティの出力の使用はサポートしていません。詳細については、CloudFormation ドキュメントの「条件」を参照してください。

このパターンでは、 AWS Lambda 関数によってバックアップされた CloudFormation カスタムリソースを使用して、GuardDuty がまだ有効になっていない場合に条件付きで有効にします。GuardDuty が有効になっている場合、スタックはステータスをキャプチャし、スタックの出力セクションに記録します。GuardDuty が有効になっていない場合は、スタックによって有効になります。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- CloudFormation スタックを作成、更新、削除するアクセス許可を持つ AWS Identity and Access Management (IAM) ロール
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み

制約事項

GuardDuty が AWS アカウント または に対して手動で無効になっている場合 AWS リージョン、このパターンではそのターゲットアカウントまたはリージョンの GuardDuty は有効になりません。

アーキテクチャ

ターゲットテクノロジースタック

このパターンでは、Infrastructure as Code (IaC) に CloudFormation を使用します。Lambda 関数に支えられた CloudFormation カスタムリソースを使用して、動的なサービス有効化機能を実現します。

ターゲットアーキテクチャ

以下の大まかなアーキテクチャ図は、CloudFormation テンプレートをデプロイして GuardDuty を有効にするプロセスを示しています。

1. CloudFormation テンプレートをデプロイして CloudFormation スタックを作成します。
2. スタックは IAM ロールと Lambda 関数を作成します。
3. Lambda 関数は IAM ロールを引き受けます。
4. ターゲットで GuardDuty がまだ有効になっていない場合 AWS アカウント、Lambda 関数によって有効になります。

自動化とスケール

AWS CloudFormation StackSet 機能を使用して、このソリューションを複数の AWS アカウントおよび に拡張できます AWS リージョン。詳細については、CloudFormation ドキュメントの「AWS CloudFormation StackSets の使用」を参照してください。

ツール

- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を通じて を操作するのに役立つオープンソースツールです。
- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- Amazon GuardDuty は、ログを分析して処理し、 AWS 環境内の予期しないアクティビティや不正なアクティビティの可能性を特定する継続的なセキュリティモニタリングサービスです。
- AWS Identity and Access Management (IAM) は、誰を認証し、誰に使用する権限を付与するかを制御することで、 AWS リソースへのアクセスを安全に管理できます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

エピック

CloudFormation テンプレートを作成し、スタックをデプロイします。

タスク	説明	必要なスキル
CloudFormation テンプレートを作成します。	1. 「追加情報」セクションの CloudFormation テンプレートのコードをコピーします。 2. テキストエディタでコードを貼り付けます。 3. ファイルをワークステーション上の `sample.yaml` で保存します。	AWS DevOps
CloudFormation スタックを更新する。	1. で AWS CLI、次のコマンドを入力します。これにより、`sample.yaml` ファイルを使用して新しい CloudFormation スタックが作成されます。詳細については、CloudFormation ドキュメントの「スタックの作成」を参照してください。 `aws cloudformation create-stack \ --stack-name guardduty-cf-stack \ --template-body file://sample.yaml` 2. スタックが正常に作成された AWS CLIことを示す次の値が に表示されることを確認します。スタックの作成	AWS DevOps

タスク	説明	必要なスキル
	に必要な時間はさまざまです。 `"StackStatus": "CREATE_COMPLETE",`	
で GuardDuty が有効になっていることを確認します AWS アカウント。	1. にサインイン AWS Management Console し、GuardDuty コンソールを開きます。 2. GuardDuty サービスが有効になっていることを確認します。	クラウド管理者、AWS 管理者
追加のアカウントまたはリージョンを設定します。	ユースケースに応じて、CloudFormation StackSet 機能を使用して、このソリューションを複数の AWS アカウント と に拡張します AWS リージョン。詳細については、CloudFormation ドキュメントのAWS CloudFormation StackSets の使用」を参照してください。	クラウド管理者、AWS 管理者

関連リソース

リファレンス

- AWS CloudFormation ドキュメント
- AWS Lambda リソースタイプのリファレンス
- CloudFormation リソースタイプ： AWS::IAM::Role
- CloudFormation リソースタイプ： AWS:GuardDuty::Detector
- を使用して AWS サービスプロパティを取得する 4 つの方法 AWS CloudFormation (ブログ記事)

チュートリアルと動画

- を使用したインフラストラクチャ管理の簡素化 AWS CloudFormation (チュートリアル)
- Amazon GuardDuty と AWS Security Hub を使用して複数のアカウントを保護する (AWS re:Invent 2020)
- オーサリングのベストプラクティス AWS CloudFormation (AWS re:Invent 2019)
- での脅威検出 AWS: Amazon GuardDuty の概要 (AWS re:Inforce 2019)

追加情報

CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  rLambdaLogGroup:
    Type: 'AWS::Logs::LogGroup'
    DeletionPolicy: Delete
    Properties:
      RetentionInDays: 7
      LogGroupName: /aws/lambda/resource-checker
  rLambdaCheckerLambdaRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Sub 'resource-checker-lambda-role-${AWS::Region}'
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: !Sub 'resource-checker-lambda-policy-${AWS::Region}'
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Sid: CreateLogGroup
                Effect: Allow
                Action:
                  - 'logs:CreateLogGroup'
```

```
                - 'logs:CreateLogStream'
                - 'logs:PutLogEvents'
                - 'iam:CreateServiceLinkedRole'
                - 'cloudformation:CreateStack'
                - 'cloudformation:DeleteStack'
                - 'cloudformation:Desc*'
                - 'guardduty:CreateDetector'
                - 'guardduty:ListDetectors'
                - 'guardduty:DeleteDetector'
              Resource: '*'
  resourceCheckerLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Description: Checks for resource type enabled and possibly name to exist
      FunctionName: resource-checker
      Handler: index.lambda_handler
      Role: !GetAtt
        - rLambdaCheckerLambdaRole
        - Arn
      Runtime: python3.13
      MemorySize: 128
      Timeout: 180
      Code:
        ZipFile: |
          import boto3
          import os
          import json
          from botocore.exceptions import ClientError
          import cfnresponse


          guardduty=boto3.client('guardduty')
          cfn=boto3.client('cloudformation')


          def lambda_handler(event, context):
              print('Event: ', event)
              if 'RequestType' in event:
                if event['RequestType'] in ["Create","Update"]:
                    enabled=False
                    try:
                      response=guardduty.list_detectors()
                      if "DetectorIds" in response and len(response["DetectorIds"])>0:
                        enabled="AlreadyEnabled"
```

```
                      elif "DetectorIds" in response and
 len(response["DetectorIds"])==0:
                        cfn_response=cfn.create_stack(
                          StackName='guardduty-cfn-stack',
                          TemplateBody='{ "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "A sample template",    "Resources": { "IRWorkshopGuardDutyDetector": {
  "Type": "AWS::GuardDuty::Detector",    "Properties": {   "Enable": true  }   } } }'
                          )
                        enabled="True"
                    except Exception as e:
                        print("Exception: ",e)
                    responseData = {}
                    responseData['status'] = enabled
                    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 "CustomResourcePhysicalID" )
                elif event['RequestType'] == "Delete":
                    cfn_response=cfn.delete_stack(
                            StackName='guardduty-cfn-stack')
                    cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
  CheckResourceExist:
    Type: 'Custom::LambdaCustomResource'
    Properties:
      ServiceToken: !GetAtt
        - resourceCheckerLambda
        - Arn
Outputs:
  status:
    Value: !GetAtt
      - CheckResourceExist
      - status
```

Lambda リソースの代替コードオプション

提供されている CloudFormation テンプレートは、参照とガイダンスを容易にするために、インラインコードを使用して Lambda リソースを参照します。または、Lambda コードを Amazon Simple Storage Service (Amazon S3) バケットに格納し、CloudFormation テンプレートで参照できます。インラインコードはパッケージの依存関係やライブラリをサポートしていません。これらをサポートするには、Lambda コードを Amazon S3 バケットに配置し、CloudFormation テンプレートで参照します。

以下のコード行に置き換えます。

```
Code:
```

```
        ZipFile: |
```

次のコード行を使用します。

```
Code:
      S3Bucket: <bucket name>
      S3Key: <python file name>
      S3ObjectVersion: <version>
```

Amazon S3 バケットでバージョニングを使用していない場合は、 S3ObjectVersionプロパティを省略できます。詳細については、Amazon S3 ドキュメントの「Amazon S3 バケットでのバージョニングの使用」を参照してください。 Amazon S3

Amazon RDS for SQL Server で透過的なデータ暗号化を有効にする

作成者: Ranga Cherukuri (AWS)

概要

Amazon Relational Database Service (Amazon RDS) for SQL Server で透過的なデータ暗号化 (TDE) を実装して保管中のデータを暗号化する方法。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- Amazon RDS for SQL Server DB インスタンス

製品バージョン

現在、Amazon RDS は、次の SQL Server バージョンおよびエディションの TDE をサポートしています。

- SQL Server 2016 Enterprise Edition
- SQL Server 2017 Enterprise Edition
- SQL Server 2019: Standard および Enterprise Edition
- SQL Server 2022: Standard および Enterprise Edition

サポート対象のバージョンとエディションに関する最新情報については、Amazon RDS ドキュメントの「SQL Server における透過的なデータ暗号化のサポート」を参照してください。

アーキテクチャ

テクノロジースタック

- Amazon RDS for SQL Server

アーキテクチャ

ツール

- Microsoft SQL Server Management Studio (SSMS) は、SQL Server インフラストラクチャを管理するための統合環境です。SQL Server とやり取りする豊富なスクリプトエディタを備えた、ユーザーインターフェイスとツールグループを備えています。

エピック

Amazon RDS コンソールでオプショングループを作成する

タスク	説明	必要なスキル
Amazon RDS コンソールを開きます。	AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。	開発者、DBA
新しいオプショングループを作成します。	ナビゲーションペインで、[オプショングループ]、[グループの作成] の順に選択します。DB エンジンに sqlserver-ee を選択し、エンジンのバージョンを選択します。	開発者、DBA
TRANSPARENT_DATA_ENCRYPTION オプションを追加します。	作成したオプショングループを編集し、`TRANSPARENT_DATA_ENCRYPTION` というオプションを追加します。	開発者、DBA

オプショングループを DB インスタンスに関連付ける

タスク	説明	必要なスキル
DB インスタンスを選択します。	Amazon RDS コンソールのナビゲーションペインで、デー	開発者、DBA

タスク	説明	必要なスキル
	タベースを選択し、オプショングループに関連付ける DB インスタンスを選択します。	
オプショングループを DB インスタンスに関連付けます。	[変更] を選択し、[オプショングループ] 設定を使用して、以前に作成したオプショングループに SQL Server DB インスタンスを関連付けます。	開発者、DBA
変更を適用します。	変更をすぐに適用するか、次回のメンテナンス期間中に適用するかを指定します。	開発者、DBA

証明書の名前を取得します。	次に示すクエリを使用して、デフォルトの証明書名を取得します。
	`USE [master]`
	`GO`
	`SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'`
	`GO`
		開発者、DBA

データベース暗号化キーを作成します。

タスク	説明	必要なスキル
SSMS を使用して、Amazon RDS for SQL Server DB インスタンスに接続します。	手順については、Microsoft ドキュメントの「SSMS を使用する」を参照してください。	開発者、DBA

タスク	説明	必要なスキル

デフォルトの証明書を使用して、データベース暗号化キーを作成します。	以前に取得したデフォルトの証明書名を使用して、データベース暗号化キーを作成します。次に示す T-SQL クエリを使用して、データベース暗号化キーを作成します。AES_128 の代わりに AES_256 アルゴリズムを指定できます。
	`USE [Databasename]`
	`GO`
	`CREATE DATABASE`
	`ENCRYPTION KEY`
	`WITH ALGORITHM = AES_128`
	`ENCRYPTION BY SERVER`
	`CERTIFICATE [certificatename]`
	`GO`
		開発者、DBA

データベースで暗号化を有効にします。	次に示す T-SQL クエリを使用して、データベース暗号化を有効にします。
	`ALTER DATABASE [Database Name]`
	`SET ENCRYPTION ON`
	`GO`
		開発者、DBA

タスク	説明	必要なスキル
暗号化のステータスをチェックします。	次に示す T-SQL クエリを使用して、暗号化のステータスを確認します。 `SELECT DB_NAME(database_id) AS DatabaseName, encryption_state, percent_complete FROM sys.dm_database_encryption_keys`	開発者、DBA

関連リソース

- 「SQL サーバーの透過的なデータ暗号化のサポート」 (Amazon RDS ドキュメント)
- 「オプショングループの使用」 (Amazon RDS ドキュメント)
- 「Amazon RDS DB インスタンスの変更」 (Amazon RDS ドキュメント)
- 「SQL Server の透過的なデータ暗号化」 (Microsoft ドキュメント)
- 「SSMS を使用する」 (Microsoft ドキュメント)

AWS ロードバランサーが安全なリスナープロトコル (HTTPS、SSL/TLS) を使用していることを確認する

作成者：Chandini Penmetsa (AWS) と Purushotham G K (AWS)

概要

Amazon Web Services (AWS) クラウドでは、エラスティック ロードバランサー が受信アプリケーショントラフィックを Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、コンテナ、IP アドレス、AWS Lambda 関数などの複数のターゲットに自動的に分散します。ロードバランサーはリスナーを使用して、ロードバランサーがユーザーからのトラフィックを受信するために使用するポートとプロトコルを定義します。Application Load Balancer はアプリケーション層でルーティングを決定し、HTTP/HTTPS プロトコルを使用します。Network Load Balancer はトランスポート層でルーティングを決定し、伝送制御プロトコル (TCP)、Transport Layer Security (TLS) 、ユーザーデータグラムプロトコル (UDP)、または TCP_UDP プロトコルを使用します。Classic Load Balancer は、TCP または Secure Sockets Layer (SSL) プロトコルを使用するトランスポート層で、またはアプリケーション層で、HTTP/HTTPS を使用してルーティングを決定します。

組織によっては、ロードバランサーが HTTPS や SSL/TLS などの安全なプロトコル上のユーザーからのトラフィックのみを受け入れるというセキュリティ要件またはコンプライアンス要件を導入している場合があります。

このパターンでは、Amazon EventBridge ルールを使用して、Application Load Balancer と Network Load Balancer の `CreateListener` および `ModifyListener` のAPI コール、そして Classic Load Balancer の `CreateLoadBalancerListeners` および `CreateLoadBalancer` の API コールを監視するセキュリティコントロールを提供します。HTTP、TCP/UDP、または TCP_UDP がロードバランサーのリスナープロトコルに使用されている場合、コントロールは Lambda 関数を呼び出します。Lambda 関数は、Amazon Simple Notification Service (Amazon SNS) トピックにメッセージを発行し、ロードバランサーの詳細を含む通知を送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- 違反の通知を受信する E メールアドレス
- Lambda コード .zip ファイルを保存するAmazon Simple Storage Service (Amazon S3) バケット

機能制限

- このセキュリティコントロールでは、ロードバランサーリスナーが更新されない限り、既存のロードバランサーはチェックされません。
- このセキュリティコントロールは地域ごとに適用されるため、監視対象の AWS リージョンに導入する必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- Lambda function
- Amazon SNS トピック
- EventBridge ルール

ターゲット アーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、監視したい複数のアカウントにこのテンプレートをデプロイできます。

ツール

- CloudFormation — AWS CloudFormation は、インフラストラクチャをコードとして使用して AWS リソースをモデル化およびセットアップするのに役立ちます。
- Amazon EventBridge - Amazon EventBridge は、お客様独自のアプリケーション、Software as a Service (SaaS) アプリケーション、AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを Lambda 関数などのターゲットにルーティングします。
- AWS Lambda – AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。

- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージ配信や送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

ベストプラクティス

使用する SNS トピックが一般公開されていないことを確認してください。詳細については、ドキュメントを参照してください。

エピック

Lambda コードをアップロードする

タスク	説明	必要なスキル
S3 バケットを削除します。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを選択または作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケットは、評価されているロードバランサーと同じリージョンに存在する必要があります。	クラウドアーキテクト
S3 バケットに Lambda コードをアップロードします。	添付ファイルセクションで提供されている Lambda コードの.zip ファイルを、定義した S3 バケットにアップロードします。	クラウドアーキテクト
AWS CloudFormation テンプレートをデプロイします。	S3 バケットと同じ AWS リージョンで AWS CloudFormation コンソールを開き、「添付ファイル」セクションで提	クラウドアーキテクト

タスク	説明	必要なスキル
	供されているテンプレートをデプロイします。次のエピックでは、パラメータの値を提供します。	

CloudFormation パラメータ

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成した S3 バケットの名前を入力します。	クラウドアーキテクト
Amazon S3 プレフィックスを提供します。	S3 バケット内の Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに提供します (例: `<directory>/<file-name>.zip`)。	クラウドアーキテクト
SNS トピックの ARN を入力しする。	違反通知に既存の SNS トピックを使用する場合は、SNS topic Amazon リソースネーム (ARN) を指定します。新しい SNS トピックを作成するには、値を None (デフォルト値) のままにします。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況	クラウドアーキテクト

タスク	説明	必要なスキル
	に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。 `Warning` 潜在的に有害な状況を示します。	

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
テンプレートをダウンロードする。	[添付ファイル] セクションで提供されている CloudFormation テンプレートをダウンロードします。	クラウドアーキテクト
スタックを作成します。	S3 バケットと同じリージョンで、CloudFormation サービスコンソールに移動し、ダウンロードしたテンプレートをデプロイします。パラメータの詳細については、前述のエピックを参照してください。	クラウドアーキテクト
リソースを検証する。	スタックの作成が完了したら、[リソース] タブに移動してリソースを確認します。テンプレートによって以下のリソースが作成されます。 • EventBridge ルール • Lambda function • Lambda 実行ロール	クラウドアーキテクト

タスク	説明	必要なスキル
	• Lambda 呼び出し許可	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、新しい SNS トピックが作成された場合、パラメータで指定されたメールアドレスにサブスクリプションメールメッセージが送信されます。違反通知を受信するには、この E メールサブスクリプションを確認する必要があります。	クラウドアーキテクト

トラブルシューティング

問題	ソリューション
スタックの作成に失敗した。GetObject 中にエラーが発生した。S3 エラーコード: Permanent Redirect。S3 エラーメッセージ: バケットはこのリージョン にあります: xx-xxxx-1. このリージョンを使用してリクエストを再試行してください	S3 バケットリージョンとスタックがデプロイされているリージョンが同じであることを確認してください。
スタックの作成に失敗した。AWS Lambda 関数を作成または更新する場合に、python3.6 のランタイムパラメータはサポートされなくなりました。	ダウンロードしたテンプレートの 186 行目を Python バージョン 3.6 から 3.9 に更新してください。

関連リソース

- AWS CloudFormation コンソールでのスタックの作成
- Lambda
- 「Classic Load Balancer とは？」
- Application Load Balancer とは?
- Network Load Balancer とは?
- AWS Lambda 関数を使用するためのベストプラクティス
- AWS CloudFormation ベストプラクティス

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon EMR 保管中のデータの暗号化が起動時に有効になっていることを確保

作成者: Priyanka Chaudhary (AWS)

概要

このパターンは、Amazon Web Services (AWS) の Amazon EMR クラスターの暗号化を監視するセキュリティコントロールを提供します。

データの暗号化は、承認されていないユーザーがクラスターおよび関連するデータストレージシステムのデータを読み取れないようにするのに役立ちます。これには、転送中のデータとして知られるネットワークを移動中に奪われる可能性のあるデータ、そして保管中のデータとして知られ、永続的なメディアに保存されているデータが含まれます。Amazon Simple Storage Service (Amazon S3) に保存されたデータは、2 つの方法で暗号化できます。

- Amazon S3 マネージドキーを使用したサーバー側の暗号化 (SSE-S3)
- Amazon EMR に適したポリシーを適用してセットアップされた AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側暗号化。

このセキュリティコントロールは API 呼び出しをモニタリングし、「RunJobFlow」 で Amazon CloudWatch Events イベントを初期化します。トリガーは Python スクリプトを実行する AWS Lambda を呼び出します。この関数は、イベント JSON 入力から EMR クラスター ID を取得し、次のチェックを実行してセキュリティ違反があるかを判断します。

1. EMR クラスターが Amazon EMR 固有のセキュリティ設定に関連付けられているかどうかを確認します。
2. Amazon EMR 固有のセキュリティ設定が EMR クラスターに関連付けられている場合、保存時の暗号化が有効になっているかを確認します。
3. 保存時の暗号化が有効になっていない場合、Amazon Simple Notification Service (Amazon SNS) 通知を送信します。これには、EMR クラスター名、違反の詳細、AWS リージョン、AWS アカウント、およびこの通知の送信元である Lambda Amazon リソースネーム (ARN) を含みます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Lambda コードの .zip ファイル用の S3 バケット
- 違反通知を受信する Eメールアドレス
- すべての API ログを取得できるように Amazon EMR ロギングがオフになっています

制約事項

- この検出統制はリージョンごとに適用されるため、監視対象の AWS リージョンに導入する必要があります。

製品バージョン

- Amazon EMR リリース 4.8.0 以上

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EMR
- Amazon CloudWatch Events イベント
- Lambda function
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、監視したい複数のアカウントにこのテンプレートをデプロイできます。

ツール

ツール

- AWS CloudFormation は、Infrastructure as Code を使用して AWS リソースをモデル化およびセットアップするのに役立つサービスです。
- Amazon CloudWatch Events は、AWS リソースの変更を示すシステムイベントをほぼリアルタイムのストリームとして提供します。
- Amazon EMR は、ビッグデータフレームワークの実行を簡素化するマネージドクラスタープラットフォームです。
- AWS Lambda は、サーバーをプロビジョニングまたは管理せずにコードを実行することをサポートします。
- Amazon S3 は、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できる、高度にスケーラブルなオブジェクトストレージサービスです。
- Amazon SNS は、パブリッシャーとクライアント間のメッセージ (ウェブサーバーや E メールアドレスを含む) の配信または送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

- このプロジェクトの EMREncryptionAtRest.zip ファイルと EMREncryptionATRest.yml ファイルは、添付ファイルとして入手できます。

エピック

S3 バケットを定義

タスク	説明	必要なスキル
S3 バケットを削除します。	Amazon S3 コンソールで、先頭にスラッシュを含まない一意の名前で S3 バケットを選択または作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されます。S3 バケットは、評	クラウドアーキテクト

タスク	説明	必要なスキル
	価対象の Amazon EMR クラスターと同じリージョンにあることが必要です。	

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
S3 バケットに Lambda コードをアップロードします。	添付ファイルセクションで提供されている Lambda コードの.zip ファイルを、定義した S3 バケットにアップロードします。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをデプロイします。	AWS CloudFormation コンソールで、S3 バケットと同じリージョンに、このパターンへのアタッチメントとして提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を指定します。AWS CloudFormation テンプレートのデプロイの詳細については、「関連リソース」セクションを参照してください。	クラウドアーキテクト

AWS CloudFormation テンプレートのパラメータを入力します

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成した S3 バケットの名前を入力します。	クラウドアーキテクト
Amazon S3 キーを指定します。	S3 バケット内の Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例、<directory>/<file-name>.zip)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ロギングのレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。「Info」はアプリケーションの進行状況に関する詳細な情報メッセージを表します。「Error」は、それでもアプリケーションの実行を継続できるエラーイベントを示します。「警告」は潜在的に有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定したメールアドレスに購読メールメッセージが送信されます。違反	クラウドアーキテクト

タスク	説明	必要なスキル
	通知を受信するには、このメールサブスクリプションを確認する必要があります。	

関連リソース

- AWS CloudFormation コンソールでのスタックの作成
- Lambda
- 「Amazon EMR 暗号化オプション」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

IAM プロファイルが EC2 インスタンスと確実に関連付けられているようにします。

作成者: Mansi Suratwala (AWS)

概要

このパターンは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで AWS 識別とアクセス管理(IAM) プロファイル違反が発生した場合、自動通知を設定する AWS CloudFormation セキュリティコントロールテンプレートを提供します。

インスタンスプロファイルは IAM ロールのコンテナであり、インスタンスの起動時に EC2 インスタンスにロール情報を渡すために使用できます。

Amazon CloudWatch Events は、 `RunInstances` 、`AssociateIamInstanceProfile` および `ReplaceIamInstanceProfileAssociation` のアクションに基づいて AWS CloudTrail が Amazon EC2 APIコールを記録する時、このチェックを初期化します。トリガーは AWS Lambda 関数を呼び出し、それはAmazon CloudWatch Events を使用して IAM プロファイルを確認します。

IAM プロファイルが存在しない場合、Lambda 関数は、Amazon Web Services (AWS) アカウント ID と AWS リージョンを含む、Amazon Simple Notification Service (Amazon SNS) の E メール通知を開始します。

IAM プロファイルが存在する場合、Lambda 関数はポリシードキュメント内のワイルドカードエントリをチェックします。ワイルドカードエントリが存在する場合、Amazon SNS 違反通知が開始されます。これは、セキュリティ強化に役立ちます。通知には、IAM プロファイルの名前、イベント、EC2 インスタンス ID、管理ポリシーの名前、違反、アカウント ID、およびリージョンが含まれます。

前提条件と制限

前提条件

- アクティブ アカウント
- Lambda コードの .zip ファイルの Amazon Simple Storage Service (Amazon S3) バケット

機能制限

- AWS CloudFormation テンプレートは `RunInstances`、`AssociateIamInstanceProfile`、`ReplaceIamInstanceProfileAssociation` のアクションのみをデプロイする必要があります。
- セキュリティコントロールは IAM プロファイルのデタッチを監視しません。
- セキュリティコントロールでは、EC2 インスタンスの IAM プロファイルにアタッチされている IAM ポリシーの変更は確認されません。
- セキュリティコントロールは、`"Resource":*` の使用が必要な「適用されないリソースレベルの許可」について考慮しません。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EC2
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon S3
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

AWS CloudFormation テンプレートを使用して、さまざまな AWS リージョンとアカウントに複数回使用できます。テンプレートは、アカウントまたはリージョンごとに 1 回だけ起動する必要があります。

ツール

ツール

- Amazon EC2 — Amazon EC2は、AWS クラウドでスケーラブルなコンピューティングキャパシティー（仮想サーバ=）を提供します。

- 「AWSCloudTrail」 — AWS CloudTrailは、AWS アカウントのガバナンス、コンプライアンス、および運用とリスク監査を行えるように支援する AWS のサービスです。ユーザー、ロール、または AWS のサービスによって起こされたアクションは、イベントとして CloudTrail に記録されます。
- 「Amazon CloudWatch Events」 — CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda 」— AWS Lambdaは、サーバーをプロビジョニングや管理をしなくてもコードを実行するために使用できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。
- 「Amazon S3」 — Amazon S3 は、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できる拡張性の高いオブジェクトストレージを提供します。
- Amazon SNS — アプリケーション、エンドユーザー、およびデバイスがクラウドとの間ですぐに通知を送受信できるようにします。

コード

- プロジェクトの .zip ファイルは添付ファイルとして入手できます。

エピック

S3 バケットを定義

タスク	説明	必要なスキル
S3 バケットを定義します。	Lambda コードの .zip ファイルをホストするには、先頭にスラッシュを含まない一意の名前で S3 バケットを選択または作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されます。S3 バケットは、評価されている EC2 インスタンスと	クラウドアーキテクト

タスク	説明	必要なスキル
	同じリージョンにあることが必要です。	

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
S3 バケットに Lambda コードをアップロードします。	添付ファイルセクションで提供されている Lambda コードを S3 バケットにアップロードします。S3 バケットは、評価されている EC2 インスタンスと同じリージョンにあることが必要です。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをデプロイします。	このパターンの添付ファイルとして提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を指定します。	クラウドアーキテクト

AWS CloudFormation テンプレートのパラメータを入力します

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成した S3 バケットの名前を入力します。	クラウドアーキテクト

タスク	説明	必要なスキル
S3 キーを指定します。	S3 バケットの Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例: `<directory>/<file-name>.zip`)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。 `Warning` 潜在的に有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定したメールアドレスに購読メールメッセージが送信されます。違反通知を受信するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「S3 バケットの作成」
- ファイルを S3 バケットにアップロード
- インスタンスプロファイルの使用
- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon Redshift クラスターが作成時に暗号化されていることを確保

作成者: Mansi Suratwala (AWS)

概要

このパターンでは、新しい Amazon Redshift クラスターが暗号化なしで作成されたときに自動的に通知をする AWS CloudFormation テンプレートを提供します。

AWS CloudFormation テンプレートは、Amazon CloudWatch Events イベントと AWS Lambda 関数を作成します。このイベントでは、AWS CloudTrail を介してスナップショットから作成、または復元されている Amazon Redshift クラスターを監視します。AWS アカウントで AWS AWS Key Management Service (AWS KMS)(AWS KMS) またはクラウドハードウェアセキュリティモデル (HSM) 暗号化なしで、クラスターを作成した場合、CloudWatch は Lambda 関数を開始して違反を通知する Amazon Simple Notification Service (Amazon SNS) 通知を送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- クラスターサブネットグループと関連するセキュリティグループを持つ仮想プライベートクラウド (VPC)

機能制限

- AWS CloudFormation テンプレートは、`CreateCluster` と `RestoreFromClusterSnapshot` のアクションにのみデプロイできます。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon Redshift
- AWS CloudTrail
- Amazon CloudWatch

- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

ターゲット アーキテクチャ

自動化とスケール

AWS CloudFormation テンプレートを使用して、さまざまな AWS リージョンとアカウントに複数回使用できます。各リージョンまたはアカウントで 必要な実行は1 回のみです。

ツール

ツール

- 「Amazon Redshift 」 — Amazon Redshiftは、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。Amazon Redshift はデータレイクと統合されているため、データを使用して、ビジネスと顧客に関する新しいインサイトを得ることができます。
- 「AWS CloudTrail 」 — AWS CoudTrailは、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査を行えるように支援する AWS のサービスです。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- 「Amazon CloudWatch Events」 — CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 — AWS Lambda は、サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。AWS Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケーリングします。
- 「Amazon S3」 — Amazon S3 は拡張性の高いオブジェクトストレージサービスです。ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。
- 「Amazon SNS」 — Amazon SNS は、ウェブサーバーや E メールアドレスを含む、パブリッシャーとクライアント間のメッセージの配信または送信を、調整および管理するウェブサービスです。

コード

- プロジェクトの .zip ファイルは添付ファイルとして入手できます。

エピック

S3 バケットを定義

タスク	説明	必要なスキル
S3 バケットを定義します。	Amazon S3 コンソールで S3 バケットを選択するか作成します。この S3 バケットは Lambda コードの.zip ファイルをホストします。S3 バケットは、評価する Amazon Redshift クラスターと同じリージョンに存在する必要があります。S3 バケットの名前の先頭にスラッシュを含めないでください。	クラウドアーキテクト

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
S3 バケットに Lambda コードをアップロードします。	添付セクションで提供された Lambda コードを S3 バケットにアップロードします。S3 バケットは、Amazon Redshift クラスターと同じ リージョンに存在する必要があります。	クラウドアーキテクト

AWS CloudFormation テンプレートをデプロイする

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートをデプロイします。	このパターンの添付ファイルとして提供されている AWS CloudFormation テンプレートをデプロイします。次のエピックでは、パラメータの値を指定します。	クラウドアーキテクト

AWS CloudFormation テンプレートのパラメータを入力します

タスク	説明	必要なスキル
S3 バケットに名前を付けます。	最初のエピックで作成した S3 バケットの名前を入力します。	クラウドアーキテクト
S3 キーを指定します。	S3 バケットの Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例: `<directory>/<file-name>.zip`)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定します。 `Error` それでもアプリケーションの実行を継続できるエラーイベン	クラウドアーキテクト

タスク	説明	必要なスキル
	トを指定します。 `Warning` 潜在的に有害な状況を示します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定したEメールアドレスに購読メールメッセージが送信されます。違反通知を受信するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「S3 バケットの作成」
- 「ファイルを S3 バケットにアップロードする」
- 「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールを作成する」
- 「Amazon Redshift クラスターを作成する」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

PowerShell を使用して AWS IAM アイデンティティセンターのアイデンティティとその割り当てに関するレポートをエクスポートする

作成者: Jorge Pava (AWS)、Chad Miles (AWS)、Frank Allotta (AWS)、および Manideep Reddy Gillela (AWS)

概要

AWS IAM アイデンティティセンター (AWS Single Sign-On の後継) を使用して、すべての Amazon Web Services (AWS) アカウントとクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理する場合、AWS マネジメントコンソールによる割り当てのレポーティングと監査は面倒で時間がかかることがあります。数十または数百の AWS アカウントにまたがるユーザーまたはグループのアクセス権限についてレポートする場合は特にそうです。

多くの人にとって、この情報を表示するための理想的なツールは、Microsoft Excel などのスプレッドシートアプリケーションです。これにより、AWS Organizations が管理する組織全体のデータをフィルタリング、検索、視覚化できます。

このパターンでは、AWS Tools for PowerShell を使用して IAM アイデンティティセンターで SSO アイデンティティ構成のレポートを生成する方法を説明します。レポートは CSV 形式で生成され、ID 名 (プリンシパル)、ID タイプ (ユーザーまたはグループ)、ID がアクセスできるアカウント、および権限セットが記載されます。レポートの生成後、これを任意のアプリケーションで開き、必要に応じてデータを検索、フィルタリング、および監査することができます。以下の画像は、スプレッドシートアプリケーションを使ったサンプルデータです。

> **Important**
>
> このレポートには機密情報が含まれているため、安全に保存し、need-to-know場合にのみ共有することを強くお勧めします。

前提条件と制限

前提条件

- 構成済みで有効になっている IAM アイデンティティセンターと AWS Organizations。
- PowerShell がインストールされ、構成済みであること。詳細については、「PowerShellをインストール」 (Microsoft のドキュメント)を参照してください。
- AWS Tools for PowerShell がインストールされ、設定されました。パフォーマンスの理由で、モジュール化された AWS Tools for PowerShell バージョンをインストールすることを強く推奨します。 `AWS.Tools` と呼びます。AWS の各サービスが、個別の小さなモジュールによってサポートされます。PowerShell シェルで、以下のコマンドを入力して、このパターンに必要な次のモジュールをインストールします: `AWS.Tools.Installer`、`Organizations`、`SSOAdmin`、および `IdentityStore`

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule -Name Organizations, SSOAdmin, IdentityStore
```

  詳細については、「Windows 用 .Tools のインストール」または「Linux または macOS 用 .Tools のインストール」(AWS Tools for PowerShell ドキュメント) を参照してください。モジュールのインストール時にエラーが発生した場合は、このパターンの「トラブルシューティング」セクションを参照してください。
- AWS コマンドラインインターフェイス (AWS CLI) または AWS SDK は、次のいずれかの方法で、あらかじめ有効な認証情報を使用して構成しておく必要があります。
  - AWS CLI `aws configure` を使用する。詳細については、「クイック構成」 (AWS CLI ドキュメント)を参照してください。
  - AWS Identity and Access Management (IAM) ロールを使用して一時的にアクセスできるように、AWS CLI または AWS Cloud Development Kit (AWS CDK) を構成する。詳細については、「CLI アクセス用の IAM ロール認証情報の取得」(IAM アイデンティティセンター ドキュメント) を参照してください。
- 以下のような IAM プリンシパルの認証情報を保存した AWS CLI の名前付きプロファイル
  - AWS Organizations の管理アカウントまたは IAM アイデンティティセンター 用の委任管理者アカウントにアクセスできる
  - `AWSSSOReadOnly` と `AWSSSODirectoryReadOnly` AWS マネージドポリシーが適用されている

  詳細については、「名前付きプロファイルの使用」 (AWS CLI ドキュメント) および「AWS マネージドポリシー」 (IAM ドキュメント) を参照してください。

制約事項

- ターゲット AWS アカウントは、AWS Organizations 内の 1 つの組織として管理する必要があります。

製品バージョン

- すべてのオペレーティングシステムでは、「PowerShell バージョン 7.0」 以降を使用することを推奨します。

アーキテクチャ

ターゲットアーキテクチャ

1. ユーザーは PowerShell コマンドラインでスクリプトを実行します。
2. このスクリプトは、AWS CLI の名前付きプロファイルを前提としています。これにより、IAM アイデンティティセンター へのアクセスが許可されます。
3. このスクリプトは IAM アイデンティティセンターから SSO ID 構成を取得します。
4. このスクリプトは、スクリプトが保存されているローカルワークステーションの同じディレクトリに CSV ファイルを生成します。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS IAM アイデンティティセンター」 により、すべての AWS アカウントとクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理できます。
- 「AWS Tools for PowerShell」 は PowerShell のコマンドラインから AWS リソースの操作をスクリプト処理することを支援する PowerShell モジュールのセットです。

その他のツール

- PowerShell は Windows、Linux、および macOS で動作する Microsoft の自動化および構成管理プログラムです。

エピック

レポートを生成する

タスク	説明	必要なスキル
スクリプトを作成する。	1. このパターンの「追加情報」セクションにある PowerShell スクリプトをコピーします。 2. `Param` セクションでは、お使いの AWS 環境に合わせて、以下の変数の値を定義します。 • `OutputFile` – レポートの名前。 • `ProfileName` – レポートの生成に使用する AWS CLI の名前付きプロファイル。 • `Region` – IAM アイデンティティセンターがデプロイされている AWS リージョン。リージョンのリストについては、「リージョンのエンドポイント」を参照してください。 3. スクリプトを `SSO-Report.ps1` という名前で保存します。	クラウド管理者
スクリプトを実行します。	PowerShell シェルで次のコマンドを使用してカスタムスクリプトを実行することをお勧めします。	クラウド管理者

タスク	説明	必要なスキル

	`.\SSO-Report.ps1`
	
	または、次のコマンドを入力して、スクリプトを別のシェルから実行します。
	
	`pwsh .\SSO-Report.ps1`
	
	このスクリプトは、スクリプトファイルと同じディレクトリに CSV ファイルを生成します。	

レポートデータを分析します。	出力 CSV ファイルには、「アカウント名」、「権限セット」、「プリンシパル」、「タイプ」の各ヘッダーがあります。このファイルを任意のスプレッドシートアプリケーションで開きます。データテーブルを作成して、出力をフィルターしたりソートしたりできます。	クラウド管理者

トラブルシューティング

問題	ソリューション
`The term 'Get-<parameter>' is not recognized as the name of a cmdlet, function, script file, or operable program.` エラー	AWS Tools for PowerShell またはそのモジュールがインストールされていません。PowerShell シェルで、以下のコマンドを入力して、AWS Tools for PowerShell と、このパターンに必要なモジュールをインストールします: `AWS.Tools.Installer` 、Organizat

問題	ソリューション

	ions 、SSOAdmin、および IdentityStore
	
	`Install-Module AWS.Tools.Installer`
	`Install-AWSToolsModule -Name Organizations, SSOAdmin, IdentityStore`

No credentials specified or obtained from persisted/shell defaults エラー	「エピック」セクションの「スクリプトの作成」で、変数 ProfileName および Region を正しく入力したことを確認します。指定したプロファイルの設定と認証情報に IAM アイデンティティセンター を管理するための十分な権限があることを確認してください。
AWS.Tools モジュールをインストールすると Authenticode Issuer … エラーが発生する	-SkipPublisherCheck パラメータを Install-AWSToolsModule コマンドに追加します。

問題	ソリューション

`Get-ORGAccountList : Assembly AWSSDK.SSO could not be found or loaded.` エラー	このエラーは、名前付きの AWS CLI プロファイルが指定され、IAM アイデンティティセンターでユーザーを認証するように AWS CLI が構成され、更新された認証トークンを自動的に取得するように AWS CLI が構成されている場合に発生する可能性があります。この問題を解決するには、次のいずれかの操作を実行します。
	
	1. 次のコマンドを入力して、SSO および SSOOIDC モジュールがインストールされていることを確認します。
	
	`Install-AWSToolsModule SSO, SSOOIDC`
	
	2. 以下の行を `param()` ブロックの下のスクリプトに挿入します。
	
	`Import-Module AWS.Tools.SSO`
	
	`Import-Module AWS.Tools.SSOOIDC`

関連リソース

- 構成設定はどこに保存されていますか? (AWS CLI ドキュメント)
- 「AWS IAM IM アイデンティティセンターを使用するための AWS CLI の設定」(AWS CLI ドキュメント)
- 「名前付きプロファイルを使用」(AWS CLI ドキュメント)

追加情報

次のスクリプトで、以下のパラメータの値を更新する必要があるかどうかを判断します。

- AWS CLI の名前付きプロファイルを使用して IAM アイデンティティセンター が構成されているアカウントにアクセスする場合は、$ProfileName 値を更新してください。
- IAM アイデンティティセンターが AWS CLI または AWS SDK 構成のデフォルトリージョンとは異なる AWS リージョンにデプロイされている場合は、IAM アイデンティティ センターがデプロイされているリージョンを使用するように $Region 値を更新します。
- どちらの状況にも当てはまらない場合は、スクリプトを更新する必要はありません。

```
param (
    # The name of the output CSV file
    [String] $OutputFile  = "SSO-Assignments.csv",
    # The AWS CLI named profile
    [String] $ProfileName = "",
    # The AWS Region in which IAM Identity Center is configured
    [String] $Region      = ""
)
$Start = Get-Date; $OrgParams = @{}
If ($Region){ $OrgParams.Region = $Region}
if ($ProfileName){$OrgParams.ProfileName = $ProfileName}
$SSOParams   = $OrgParams.Clone(); $IdsParams = $OrgParams.Clone()
$AccountList = Get-ORGAccountList @OrgParams | Select-Object Id, Name
$SSOinstance = Get-SSOADMNInstanceList @OrgParams
$SSOParams['InstanceArn']       = $SSOinstance.InstanceArn
$IdsParams['IdentityStoreId']   = $SSOinstance.IdentityStoreId
$PSsets       = @{}; $Principals   = @{}
$Assignments  = @(); $AccountCount = 1; Write-Host ""
foreach ($Account in $AccountList) {
    $Duration = New-Timespan -Start $Start -End (Get-Date) | ForEach-Object
 {[Timespan]::New($_.Days, $_.Hours, $_.Minutes, $_.Seconds)}
    Write-Host "`r$Duration - Account $AccountCount of $($AccountList.Count)
 (Assignments:$($Assignments.Count))        " -NoNewline
    $AccountCount++
    foreach ($PS in Get-SSOADMNPermissionSetsProvisionedToAccountList -AccountId
 $Account.Id @SSOParams) {
        if (-not $PSsets[$PS]) {$PSsets[$PS] = (Get-SSOADMNPermissionSet @SSOParams -
PermissionSetArn $PS).Name;$APICalls++}
        $AssignmentsResponse = Get-SSOADMNAccountAssignmentList @SSOParams -
PermissionSetArn $PS -AccountId $Account.Id
        if ($AssignmentsResponse.NextToken) {$AccountAssignments =
 $AssignmentsResponse.AccountAssignments}
        else {$AccountAssignments = $AssignmentsResponse}
        While ($AssignmentsResponse.NextToken) {
```

```
            $AssignmentsResponse = Get-SSOADMNAccountAssignmentList @SSOParams -
PermissionSetArn $PS -AccountId $Account.Id -NextToken $AssignmentsResponse.NextToken
            $AccountAssignments += $AssignmentsResponse.AccountAssignments}
        foreach ($Assignment in $AccountAssignments) {
            if (-not $Principals[$Assignment.PrincipalId]) {
                $AssignmentType = $Assignment.PrincipalType.Value
                $Expression     = "Get-IDS"+$AssignmentType+" @IdsParams -"+
$AssignmentType+"Id "+$Assignment.PrincipalId
                $Principal      = Invoke-Expression $Expression
                if ($Assignment.PrincipalType.Value -eq "GROUP")
 { $Principals[$Assignment.PrincipalId] = $Principal.DisplayName }
                else { $Principals[$Assignment.PrincipalId] = $Principal.UserName }
            }
            $Assignments += [PSCustomObject]@{
                AccountName     = $Account.Name
                PermissionSet   = $PSsets[$PS]
                Principal       = $Principals[$Assignment.PrincipalId]
                Type            = $Assignment.PrincipalType.Value}
        }
    }
}
$Duration = New-Timespan -Start $Start -End (Get-Date) | ForEach-Object
 {[Timespan]::New($_.Days, $_.Hours, $_.Minutes, $_.Seconds)}
Write-Host "`r$($AccountList.Count) accounts done in $Duration. Outputting result to
 $OutputFile"
$Assignments | Sort-Object Account | Export-CSV -Path $OutputFile -Force
```

予定されている AWS KMS キーの削除を監視して修正する

ミケシュ・カナル (AWS) とラムヤ・プリパカ (AWS) によって作成された

概要

Amazon Web Services (AWS) クラウドでは、AWS キー管理サービス (AWS KMS) キーを削除するとデータが失われる可能性があります。削除することで、キーマテリアルとAWS KMS キーに関連付けられているすべてのメタデータを削除し、元に戻すことはできません。AWS KMS キーを削除すると、その AWS KMS キーで暗号化されたデータを復号できなくなります。これは、そのデータを回復することがきなくなります。

このパターンは、アプリケーションまたはユーザーが AWS KMS キーの削除をスケジュールしたときに通知するモニタリングを設定します。この通知を受け取った場合は、AWS KMS キーの削除をキャンセルして、削除する決定を検討し直す必要があります。このパターンでは、AWS Systems Manager Automationランブック「AWSConfigRemediation-CancelKeyDeletion」 を使用して、AWS KMS キーの削除を簡単にキャンセルできます。

> **Note**
>
> パターンの CloudFormation テンプレートは、AWS KMS キーの削除をモニタリングするすべての AWS リージョンにデプロイする必要があります。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- 以下の AWS サービスの理解
  - Amazon EventBridge
  - AWS KMS
  - Amazon Simple Notification Service (Amazon SNS)
  - AWS Systems Manager

機能制限

- ソリューションをカスタマイズするには、AWS CloudFormation テンプレートと、このパターンで使用される AWS サービスに関する知識が必要です。
- 現在、このソリューションはデフォルトのイベントバスを使用しており、要件に応じてカスタマイズできます。カスタムイベントバスの詳細については、「AWS ドキュメント」を参照してください。

アーキテクチャ

ターゲットテクノロジースタック

- Amazon EventBridge
- AWS KMS
- Amazon SNS
- AWS Systems Manager
- 以下を使用した自動化
  - AWS コマンドラインインターフェイス (AWS CLI)または AWS SDK
  - AWS CloudFormation スタック

ターゲットアーキテクチャ

1. AWS KMS キーの削除が予定されています。
2. 予定削除イベントは、EventBridge ルールによって評価されます。
3. EventBridge ルールは Amazon SNS トピックに関係します。
4. EventBridge ルールは、Systems Manager の自動化とランブックを開始します。
5. Runbook は削除がキャンセルされます。

自動化とスケール

CloudFormation スタックは、このソリューションが機能するために必要なすべてのリソースとサービスをデプロイします。このパターンは、単一のアカウントで個別に実行することも、複数の独立したアカウントまたは組織に対して AWS CloudFormation StackSets を使用して実行することもできます。

```
aws cloudformation create-stack --stack-name  <stack-name>\
    --template-body file://<Full-Path-of-file> \
    --parameters ParameterKey=,ParameterValue= \
    --capabilities CAPABILITY_NAMED_IAM
```

ツール

ツール

- AWS CloudFormation – AWS CloudFormation は、Amazon Web Servicesリソースのモデル化とセットアップを支援するサービスであるため、これらのリソースの管理に費やす時間を減らし、AWS で実行されるアプリケーションに集中する時間を増やすことができます。CloudFormation テンプレートを使用して、AWS リージョンの AWS アカウントにスタックを作成できます。必要なすべての AWS リソースを説明するテンプレートが、CloudFormation がお客様に代わってこれらのリソースのプロビジョニングや設定を処理します。
- AWS CLI – AWS コマンドラインインターフェイス (AWS CLI)はオープンソースのツールであり、コマンドラインシェルのコマンドを使ってAWSサービスと対話することができます。
- Amazon EventBridge— Amazon EventBridgeは、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。EventBridge は、お客様独自のアプリケーション、AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを AWS Lambda などのターゲットにルーティングします。EventBridge は、イベント駆動型アーキテクチャを構築するプロセスを簡素化します。
- AWS KMS — AWS Key Management Service（AWS KMS）は、AWS KMS キー（データの暗号化に使用される暗号化キー）の作成と管理を容易にするマネージドサービスです。
- AWS SDK — AWS ツールには SDK が含まれているため、選択したプログラミング言語で AWS 上のアプリケーションを開発および管理できます。
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバー (または生産者から消費者) へのメッセージ配信を提供するマネージドサービスです。パブリッシャーは、論理アクセスポイントおよび通信チャネルであるトピックにメッセージを送信することで、受信者と非同期的に通信します。
- AWS Systems Manager – AWS Systems Manager は、 AWS でインフラストラクチャの表示と制御に使用できる AWS サービスです。Systems Manager コンソールを使用すると、AWS リソース全体の運用タスクを自動化できます。Systems Manager は、マネージドインスタンスをスキャンし、検出されたポリシー違反を報告（または是正措置を講じる）して、セキュリティとコンプライアンスを維持することができます。

コード

- プロジェクトの`alerting_ct_logs.yaml` CloudFormation テンプレートが添付されています。

エピック

AWS アカウントを準備する

タスク	説明	必要なスキル
AWS CLI をインストールして設定します。	AWS CLI バージョン 2 をインストールします。次に、アイデンティティ、デフォルトの出力形式、AWS CLI が AWS とのやり取りに使用されるデフォルトの AWS リージョンのセキュリティ認証情報を設定します。 アイデンティティーには、タスクを実行するために必要なアクセス許可があることが求められます。	開発者、セキュリティエンジニア

AWS CloudFormation テンプレートをデプロイします

タスク	説明	必要なスキル
CloudFormation テンプレートファイルをダウンロードします。	添付ファイルをコンピューターのローカルパスにダウンロードし、`alerting_ct_logs.yaml` テンプレートファイルを抽出します。	開発者、セキュリティエンジニア
テンプレートをデプロイします。	AWS アカウントプロファイルが設定されたターミナルウィ	開発者、セキュリティエンジニア

タスク	説明	必要なスキル
	ンドウで、以下のコマンドを実行します。 `aws cloudformation create-stack --stack-name <stack_name> \ --capabilities <Value> \ --template-body file://<Full_Path> \ --parameters ParameterKey=DestinationEmailAddress,ParameterValue=<Value> \ ParameterKey=SNSTopicName,ParameterValue=<Value> \ ParameterKey=EnableRemediation ,ParameterValue=<Value> \ ParameterKey=AutomationAssumeRole, ParameterValue=<Value>` 次のステップでは、テンプレートパラメータの値を入力します。	

タスク	説明	必要なスキル
テンプレートパラメータを入力します。	パラメータの必須値を入力します。 • `DestinationEmailAddress` — AWS KMS キーの削除がスケジュール済みの場合、アラートを受け取る E メールアドレス。 • `SNSTopicName` — Amazon SNS トピックの名前。 • `EnableRemediation` — Systems Manager ランブックを使用して、予定されていたキーの削除をキャンセルします。指定できる値は `true` と `false` です。 • 説明： (オプション) Automation がユーザーに代わってアクションを実行できるようにするロールの Amazon リソースネーム (ARN)。詳細については、「AWSConfigRemediation-CancelKeyDeletion」ドキュメントの「必要な IAM 権限」セクションを参照してください。 • `Capabilities` — AWS CloudFormation が「スタックを作成する」には、スタックテンプレートに特定の機能が含まれていること	開発者、セキュリティエンジニア

タスク	説明	必要なスキル
	を明示的に確認する必要があります。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	E メールの受信トレイを確認し、Amazon SNS から受信する E メールメッセージで [サブスクリプションの確認] を選択します。ウェブブラウザが開き、サブスクリプション ID とともにサブスクリプションの確認を表示します。	開発者、セキュリティエンジニア

関連リソース

リファレンス

- AWS サービス用のロールを作成する
- Amazon CloudWatch アラームを作成して、削除保留中の AWS KMS キー の使用を検出する

チュートリアルと動画

- Amazon EventBridge の開始方法
- Amazon EventBridgeのディープ・ダイブ（AWS オンラインテックトーク）

AWS ワークショップ

- EventBridge ルールの使用

追加情報

以下のコードは、あらゆる AWS サービスの変更を監視して通知するようにソリューションを拡張する例を示しています。例には、定義済みのパターンとカスタムパターンが含まれます。詳細については、「EventBridge でのイベントとイベントパターン」を参照してください。

```
EventPattern:
        source:
        - aws.kms
        detail-type:
        - AWS API Call via CloudTrail
        detail:
          eventSource:
          - kms.amazonaws.com
          eventName:
          - ScheduleKeyDeletion
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Security Hub AWS Organizations を使用して でパブリック Amazon S3 バケットを特定する

ムラッド・シェルファウイ (AWS)、アルン・チャンダピライ (AWS)、パラグ・ナグウェカー (AWS) により作成

概要

このパターンは、 AWS Organizations アカウント内のパブリック Amazon Simple Storage Service (Amazon S3) バケットを識別するメカニズムを構築する方法を示しています。このメカニズムは、AWS の Foundational Security Best Practices (FSBP) 標準のコントロールを使用して Amazon S3 バケットをモニタリング AWS Security Hub することで機能します。Amazon EventBridge を使用してSecurity Hub 「発見」を処理し、その検出結果を Amazon Simple Notification Service (Amazon SNS) トピックに投稿できます。組織内の利害関係者はトピックを購読して、検出結果に関する電子メール通知をすぐに受け取ることができます。

新しい Amazon S3 バケットとそのオブジェクトは、デフォルトではパブリックアクセスを許可しません。このパターンは、組織の要件に基づいてデフォルトの Amazon S3 設定を変更する必要があるシナリオで使用できます。たとえば、インターネット上のすべてのユーザーが Amazon S3 バケットから読み取ることができるように、公開ウェブサイトまたはファイルをホストする Amazon S3 バケットがあるシナリオが考えられます。

Security Hub は、多くの場合、セキュリティ標準やコンプライアンス要件に関連するものを含め、すべてのセキュリティ検出結果を統合するための中心的なサービスとして導入されます。パブリック Amazon S3 バケットを検出するために AWS のサービス 使用できる他のものもありますが、このパターンでは、最小限の設定で既存の Security Hub デプロイを使用します。

前提条件と制限

前提条件

- 専用の Security Hub 管理者アカウントを使用した AWS マルチアカウント設定 https://docs.aws.amazon.com/securityhub/latest/userguide/designate-orgs-admin-account.html
- モニタリングする で AWS Config有効 AWS リージョン になっている Security Hub と

> **Note**
>
> 1 つの集約リージョンから複数のリージョンをモニタリングする場合は、Security Hub でクロスリージョン集約を有効にする必要があります。

- Security Hub 管理者アカウントにアクセスして更新するためのユーザーアクセス許可、組織内のすべての Amazon S3 バケットへの読み取りアクセス許可、およびパブリックアクセスを無効にするためのアクセス許可 (必要な場合)

アーキテクチャ

次の図は、Security Hub を使用してパブリック Amazon S3 バケットを識別するためのアーキテクチャを示しています。

この図は、次のワークフローを示しています。

1. Security Hub は、FSBP セキュリティ標準の S3.2 および S3.3 コントロールを使用して、すべてのアカウント (管理者アカウントを含む) の Amazon S3 バケットの設定をモニタリングし、バケットがパブリックとして設定されている場合は検出結果を検出します。 AWS Organizations S3
2. Security Hub 管理者アカウントは、すべてのメンバーアカウントの検出結果 (S3.2 と S3.3 のものを含む) にアクセスします。
3. Security Hub は、すべての新しい検出結果と既存の検出結果のすべての更新を EventBridge にSecurity Hub Findings - Importedイベントとして自動的に送信します。これには、管理者アカウントとメンバーアカウントの両方からの検出検出結果のイベントが含まれます。
4. EventBridge ルールは、S3.2 と S3.3 の調査検出結果のうち、`ComplianceStatus`が`FAILED`で、ワークフローステータスが`NEW`で、`RecordState`が`ACTIVE`であるものをフィルタリングする。
5. ルールはイベントパターンを使用してイベントを識別し、一致したら Amazon SNS トピックに送信します。
6. Amazon SNS トピックは、イベントをサブスクライバーに送信します (E メールなど)。
7. E メール通知を受信するように指定されたセキュリティアナリストは、問題の Amazon S3 バケットを確認します。
8. バケットのパブリックアクセスが承認されると、セキュリティアナリストは Security Hub 内の対応する検出結果のワークフローステータスを`SUPPRESSED`に設定します。それ以外の場合、アナ

リストはステータスをNOTIFIEDに設定します。これにより、Amazon S3 バケットの今後の通知がなくなり、通知ノイズが軽減されます。

9. ワークフローのステータスがNOTIFIEDに設定されている場合、セキュリティアナリストは検出結果をバケット所有者と確認して、パブリックアクセスが正当であり、プライバシーとデータ保護の要件を満たしているかどうかを判断します。調査の結果、バケットへのパブリックアクセスを削除するか、パブリックアクセスを承認するかのどちらかになります。後者の場合、セキュリティアナリストはワークフローのステータスをSUPPRESSEDに設定します。

> **Note**
>
> このアーキテクチャ図は、単一リージョンとクロスリージョンの両方の集約デプロイに適用されます。図のアカウント A、B、C では、Security Hub は管理者アカウントと同じリージョンに属していても、クロスリージョン集約が有効になっている場合は別のリージョンに属していてもかまいません。

ツール

- 「Amazon EventBridge」 は、アプリケーションをさまざまなソースのデータに接続するために支援するサーバーレスイベントバスサービスです。EventBridge は、独自のアプリケーション、Software as a Service (SaaS) アプリケーション、および からリアルタイムデータのストリームを提供します AWS のサービス。EventBridge は、データがユーザー定義ルールと一致する場合に、そのデータを Amazon SNS トピックや AWS Lambda 関数などのターゲットにルーティングします。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。Security Hub は、セキュリティ業界標準とベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。Security Hub は AWS アカウント、、 のサービス、およびサポートされているサードパーティーパートナー製品からセキュリティデータを収集し、セキュリティの傾向を分析し、最も優先度の高いセキュリティ問題を特定するのに役立ちます。

エピック

Security Hub アカウントの設定

タスク	説明	必要なスキル
AWS Organizations アカウントで Security Hub を有効にします。	Amazon S3 バケットをモニタリングする組織アカウントで Security Hub を有効にするには、Security Hub ドキュメントの「Security Hub 管理者アカウント (コンソール) の指定」と「組織に属するメンバーアカウントの管理」のガイドラインを参照してください。	AWS 管理者
(オプション) クロスリージョン集約を有効にします。	1 つのリージョンから複数のリージョンの Amazon S3 バケットをモニタリングする場合は、クロスリージョン集約を設定します。	AWS 管理者
FSBP セキュリティ標準の S3.2 と S3.3 のコントロールを有効にします。	FSBP セキュリティ標準の S3.2 と S3.3 のコントロールを有効にする必要があります。 1. S3.2 コントロールを有効にするには、Security Hub ドキュメントの [S3.2] S3 バケットでパブリック読み取りアクセスを禁止する必要があります。 2. S3.3 コントロールを有効にするには、Security Hub ドキュメントの [3] S3 バケットでパブリック書き込みア	AWS 管理者

タスク	説明	必要なスキル
	クセスを禁止する必要があります。	

環境をセットアップする

タスク	説明	必要なスキル
Amazon SNS トピックと E メールサブスクリプションを設定します。	1. にサインイン AWS Management Console し、Amazon SNS コンソールを開きます。 2. ナビゲーションペインで、[Topics (トピック)]、[Create topic (トピックの作成)] の順に選択してください。 3. [Type (タイプ)] で、[Standard (標準)] を選択してください。 4. 「名前」には、トピックの名前 (たとえば、public-s3-buckets) を入力します。 5. [トピックの作成] を選択してください。 6. [サブスクリプション] タブで [サブスクリプションの作成] を選択します。 7. [Protocol] で [Email] を選択します。 8. [Endpoint] (エンドポイント) で、通知を受信するメールアドレスを入力します。 AWS 管理者、IT プロ	AWS 管理者

タスク	説明	必要なスキル
	フェッショナル、または Infosec プロフェッショナルの E メールアドレスを使用できます。 9. [Create subscription] を選択してください。E メールサブスクリプションを追加で作成するには、必要に応じてステップ 6 ~ 8 を繰り返します。	

タスク	説明	必要なスキル
EventBridge ルールを設定します。	1. [EventBridge コンソール] を開きます。 2. 「はじめに」セクションで、「EventBridge ルール」を選択し、「ルールを作成」を選択します。 3. 「ルール詳細の定義」ページの「名前」に、ルールの名前 (「public-s3-buckets」など) を入力します。[次へ] を選択します。 4. [Define pattern] (パターンの定義) セクションで [Event pattern] (イベントパターン) を選択します。 5. 次のコードをコピーして、「イベントパターン」コードエディターに貼り付け、「次へ」を選択します。 `{ "source": ["aws.securityhub"], "detail-type": ["Security Hub Findings - Imported"], "detail": { "findings": { "Compliance": { "Status": ["FAILED"] }, "RecordState": ["ACTIVE"], "Workflow": {`	AWS 管理者

タスク	説明	必要なスキル
	<pre>"Status": ["NEW"] }, "ProductFields": { "ControlId": ["S3.2", "S3.3"] } } } }</pre> 6. 「ターゲットの選択」ページの「ターゲットの選択」で、ターゲットとして「SNS トピック」を選択し、先ほど作成したトピックを選択します。 7. 「次へ」を選択し、もう一度「次へ」を選択し、「ルールを作成」を選択します。	

トラブルシューティング

問題	ソリューション
パブリックアクセスが有効になっている Amazon S3 バケットがありますが、E メール通知が届きません。	これは、バケットが別のリージョンで作成され、Security Hub 管理者アカウントでクロスリージョン集約が有効になっていないことが原因である可能性があります。この問題を解決するには、クロスリージョン集約を有効にするか、Amazon S3 バケットが現在存在するリージョンにこのパターンのソリューションを実装します。

関連リソース

- とは AWS Security Hub (Security Hub ドキュメント)
- AWS Foundational Security Best Practices (FSBP) 標準 (Security Hub ドキュメント)
- AWS Security Hub マルチアカウント有効化スクリプト (AWS Labs)
- Amazon S3 のセキュリティのベストプラクティス (Amazon S3 ドキュメント)

追加情報

パブリック Amazon S3 バケットをモニタリングするためのワークフロー

次のワークフローは、組織内のパブリック Amazon S3 バケットをモニタリングする方法を示しています。ワークフローは、このパターンのAmazon SNSトピックと E メールサブスクリプションの設定」のステップを完了したことを前提としています。

1. Amazon S3 バケットにパブリックアクセスが設定されると、E メール通知が送信されます。
   - バケットのパブリックアクセスが承認されたら、対応する検出結果のワークフローステータスを Security Hub 管理者アカウントでSUPPRESSEDに設定します。これにより、Security Hub がこのバケットについてさらに通知を発行することを防ぎ、重複するアラートを排除できます。
   - バケットのパブリックアクセスが承認されていない場合は、Security Hub 管理者アカウントの対応する検出結果のワークフローステータスをNOTIFIEDに設定します。これにより、Security Hub からこのバケットに関する通知が今後発行されるのを防ぎ、ノイズを排除できます。
2. バケットに機密データが含まれている可能性がある場合は、レビューが完了するまですぐにパブリックアクセスをオフにしてください。パブリックアクセスをオフにすると、Security Hub はワークフローのステータスをRESOLVEDに変更します。その後、バケットのメール通知は停止します。
3. バケットをパブリックとして (たとえば、 を使用して AWS CloudTrail) 設定したユーザーを検索し、レビューを開始します。レビューの結果、バケットへのパブリックアクセスが削除されるか、パブリックアクセスが承認されます。パブリックアクセスが承認されたら、該当する検出結果のワークフローステータスをSUPPRESSEDに設定します。

Microsoft Sentinel で AWS セキュリティログを取り込んで分析する

作成者: Ivan Girardi (AWS) と Sebastian Wenzel (AWS)

概要

このパターンでは、ログ、Amazon CloudWatch Logs データ AWS CloudTrail 、Amazon VPC フローログデータ、Amazon GuardDuty の検出結果などの AWS セキュリティログの Microsoft Sentinel への取り込みを自動化する方法について説明します。組織が Microsoft Sentinel をセキュリティ情報イベント管理 (SIEM) システムとして使用している場合、セキュリティ関連のイベントを検出するためにログを一元的にモニタリングおよび分析するのに役立ちます。ログが利用可能になるとすぐに、5 分以内に Amazon Simple Storage Service (Amazon S3) バケットに自動的に配信されます。これにより、 AWS 環境内のセキュリティイベントをすばやく検出できます。

Microsoft Sentinel は、イベントが記録された元のタイムスタンプを含む表形式の形式で CloudTrail ログを取り込みます。取り込まれたログの構造により、Microsoft Sentinel で Kusto クエリ言語を使用してクエリ機能が有効になります。

このパターンは、1 分以内に取り込みの失敗を検出するモニタリングおよびアラートソリューションをデプロイします。また、外部 SIEM がモニタリングできる通知システムも含まれています。を使用して AWS CloudFormation 、必要なリソースをログ記録アカウントにデプロイします。

ターゲットオーディエンス

このパターンは AWS Control Tower、、CloudFormation AWS Organizations、 AWS Identity and Access Management (IAM)、および AWS Key Management Service () の経験があるユーザーに推奨されますAWS KMS。

前提条件と制限

前提条件

このソリューションをデプロイするための前提条件は次のとおりです。

- で組織として AWS アカウント 管理 AWS Organizations され、 AWS Control Tower ランディングゾーンの一部であるアクティブ。組織には、ログ記録専用のアカウントを含める必要があります。手順については、 AWS Organizations ドキュメントの「組織の作成と設定」を参照してください。

- 組織全体のイベントをログに記録し、ログ記録アカウントの Amazon S3 バケットにログを保存する CloudTrail 証跡。手順については、「組織の証跡の作成」を参照してください。
- ログ記録アカウントで、次のアクセス許可を持つ既存の IAM ロールを引き受けるアクセス許可。
  - 提供された CloudFormation テンプレートで定義されたリソースをデプロイします。
  - 提供された CloudFormation テンプレートをデプロイします。
  - ログがカスタマーマネージド AWS KMS キーで暗号化されている場合は、キーポリシーを変更します。
- AWS Command Line Interface （ AWS CLI ）、インストールおよび設定済み。
- Microsoft Sentinel を使用するサブスクリプションを持つ Microsoft Azure アカウント。
- Microsoft Sentinel を有効にしてセットアップします。手順については、Microsoft Sentinel ドキュメントの「Microsoft Sentinel を有効にする」および「初期機能とコンテンツ」を参照してください。
- Microsoft Sentinel S3 コネクタを設定するための前提条件を満たします。

制約事項

- このソリューションは、ログ記録アカウントの Amazon S3 バケットから Microsoft Sentinel にセキュリティログを転送します。Amazon S3 にログを送信する方法については、明示的には説明されていません。
- このパターンでは、 AWS Control Tower ランディングゾーンにデプロイする手順を示します。ただし、 AWS Control Tower の使用は必須ではありません。
- このソリューションは、ログアーカイブで作成された Amazon S3 バケットのバケットポリシーへの変更の禁止など、Amazon S3 ログバケットがサービスコントロールポリシー (SCPs) で制限されている環境と互換性があります。 AWS Control Tower Amazon S3
- このパターンは CloudTrail ログを転送する手順を提供しますが、CloudWatch Logs、Amazon VPC フローログ、GuardDuty からのログなど、Microsoft Sentinel がサポートする他のログを送信するようにこのソリューションを調整することができます。
- 手順では、 AWS CLI を使用して CloudFormation テンプレートをデプロイしますが、 を使用することもできます AWS Management Console。手順については、「 AWS CloudFormation コンソールの使用」を参照してください。コンソールを使用してスタックをデプロイする場合は、ログ記録バケット AWS リージョン と同じ にスタックをデプロイします。
- このソリューションは、Amazon Simple Queue Service (Amazon SQS) キューをデプロイして Amazon S3 通知を配信します。キューには、実際のデータではなく、Amazon S3 バケットにアップロードされたオブジェクトのパスを含むメッセージが含まれます。キューは SSE-SQS 暗号化を

使用して、メッセージの内容を保護します。SQS キューを SSE-KMS で暗号化する場合は、カスタマーマネージド KMS キーを使用できます。詳細については、Amazon SQS での保管時の暗号化」を参照してください。

アーキテクチャ

このセクションでは、サンプルコードが確立するアーキテクチャの概要を説明します。次の図は、既存の Amazon S3 バケットから Microsoft Sentinel にログを取り込むために、ログ記録アカウントにデプロイされたリソースを示しています。

アーキテクチャ図は、次のリソースインタラクションを示しています。

1. ログ記録アカウントでは、Microsoft Sentinel は OpenID Connect (OIDC) を介して IAM ロールを引き受け、特定の Amazon S3 バケットと Amazon SQS キューのログにアクセスします。
2. Amazon Simple Notification Service (Amazon SNS) と Amazon S3 は暗号化 AWS KMS に使用します。
3. Amazon S3 は、新しいログを受信するたびに Amazon SQS キューに通知メッセージを送信します。
4. Microsoft Sentinel は Amazon SQS で新しいメッセージをチェックします。Amazon SQS キューは SSE-SQS 暗号化を使用します。メッセージ保持期間は 14 日間に設定されています。
5. Microsoft Sentinel は Amazon SQS キューからメッセージを取得します。メッセージには、アップロードされた Amazon S3 オブジェクトのパスが含まれます。Microsoft Sentinel は、これらのオブジェクトを Amazon S3 バケットから Microsoft Azure アカウントに取り込みます。
6. CloudWatch アラームは Amazon SQS キューをモニタリングします。メッセージが 5 分以内に Amazon SQS キューから受信および削除されない場合、E メールを送信する Amazon SNS 通知が開始されます。

AWS Control Tower は、基本的な組織単位 (OU) 構造をセットアップし、CloudTrail ログを口グ記録アカウントに一元化するのに役立ちます。また、口ギングバケットを保護するために必須SCPs を実装します。

ラン AWS Control Tower ディングゾーンにターゲットアーキテクチャを用意しましたが、これは厳密に必須ではありません。この図では、管理アカウントのリソースには、組織全体のイベントを口グに記録する AWS Control Tower デプロイと CloudTrail 証跡が反映されています。

このパターンは、ログ記録アカウントにリソースをデプロイすることに焦点を当てています。 AWS Control Tower ランディングゾーンの Amazon S3 に保存されているログがカスタマーマネージド KMS キーで暗号化されている場合は、Microsoft Sentinel がログを復号できるようにキーポリシーを更新する必要があります。 AWS Control Tower ランディングゾーンでは、キーが作成された管理アカウントからキーポリシーを管理します。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースをセットアップし、迅速かつ一貫してプロビジョニングし、 AWS アカウント および リージョン全体でライフサイクル全体を通じてリソースを管理するのに役立ちます。
- Amazon CloudWatch は、 AWS リソースと で実行するアプリケーションのメトリクスを AWS リアルタイムでモニタリングするのに役立ちます。
- AWS Control Tower は、ベストプラクティスに従って、 AWS マルチアカウント環境をセットアップして管理するのに役立ちます。
- AWS Key Management Service （ AWS KMS ） は、データの保護に役立つ暗号化キーの作成と制御に役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- 「Amazon Simple Queue Service (Amazon SQS) 」は、安全で耐久性があり、配信ソフトウェアシステムとコンポーネントを統合および分離できる利用可能なホスト型キューを提供します。
- 「Amazon Simple Storage Service (Amazon S3) 」は、どのようなデータの量であっても、保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

その他のツール

- Microsoft Sentinel は、セキュリティオーケストレーション、自動化、およびレスポンス (SOAR) を提供するクラウドネイティブの SIEM システムです。

コードリポジトリ

このパターンのコードは、GitHub Ingest and analyze AWS security logs in Microsoft Sentinel repository で入手できます。

ベストプラクティス

- 最小特権の原則 (IAM ドキュメント) に従ってください。
- AWS Control Tower 管理者向けのベストプラクティス (AWS Control Tower ドキュメント) に従います。
- AWS CloudFormation ベストプラクティス (CloudFormation ドキュメント) に従います。
- cfn_nag などのコード分析ツールを使用して、生成された CloudFormation テンプレートをスキャンします。cfn_nag ツールは、パターンを検索して CloudFormation テンプレートの潜在的なセキュリティ問題を特定します。

エピック

Microsoft Sentinel を Amazon S3 に接続する

タスク	説明	必要なスキル
Microsoft Sentinel S3 コネクタを準備します。	1. Microsoft Sentinel で、データコネクタを選択します。 2. データコネクタギャラリーから、Amazon Web Services S3 を選択します。 **Note** コネクタが表示されない場合は、Microsoft Sentinel の Content Hub から Amazon Web Services ソリューションをインストールします。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	3. コネクタの詳細ペインで、コネクタを開くページを選択します。 4. 設定セクションで、外部 ID をコピーします。この ID は後で必要になります。	

CloudFormation スタックをデプロイする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	bash シェルで、次のコマンドを入力します。これにより、Microsoft Sentinel リポジトリで AWS セキュリティログの取り込みと分析のクローンが作成されます。 `git clone https://github.com/aws-samples/ingest-and-analyze-aws-security-logs-in-microsoft-sentinel.git`	DevOps エンジニア、AWS 全般
ログ記録アカウントの IAM ロールを引き受けます。	ログ記録アカウントで、CloudFormation スタックをデプロイするアクセス許可を持つ IAM ロールを引き受けます。で IAM ロールを引き受ける方法の詳細については AWS CLI、『』の「IAM ロールを使用する AWS CLI」を参照してください。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
スタックをデプロイします。	CloudFormation スタックをデプロイするには、次のコマンドを入力します。 • `<Bucket name>` は、ログ記録 Amazon S3 バケットの名前です。 • `<Sentinel external ID>` は、Microsoft Sentinel の Amazon S3 コネクタの外部 ID です。 • `<Email address>` は、通知を受け取る有効な E メールアドレスです。 • `<Customer managed key ARN>` は、カスタマーマネージド KMS キーの Amazon リソースネーム (ARN) です。ログがカスタマーマネージド KMS キーで暗号化されている場合にのみ、このパラメータを指定します。 • `<Suffix>` は、リソース名の競合を回避するためのオプションのパラメータです。 • `<ARN for the OIDC provider>` は、OIDC プロバイダーがすでに存在する場合の ARN です。このパラメータを指定しない場合、CloudFormation は OIDC	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	プロバイダーを作成します。 **Important** AWS 組織が Microsoft Code Defender でモニタリングされている場合、Microsoft の OIDC プロバイダーは既にデプロイされています。このパラメータと既存のプロバイダーの ARN を指定する必要があります。 `aws cloudformation deploy --stack-name cloudtrail-sentinel-integration \ --no-fail-on-empty-changeset \ --template-file template.yml \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \ --parameter-overrides \ ControlTowerS3BucketName="<Bucket name>" \`	

タスク	説明	必要なスキル

	<pre> AzureWorkspaceID="<Sentinel external ID>" \
	    EmailAddress="<Email address>" \
	    KMSKeyArn="<Customer managed key ARN>" \
	    Suffix="<Suffix to avoid name conflicts>" \
	    OIDCProviderArn="<ARN for the OIDC provider>"</pre>	

出力をコピーします。	CloudFormation スタックの出力から、 SentinelRoleArn と の値をコピーしますSentinelSQS 。これらの値は、後で Microsoft Sentinel で設定を完了するために使用します。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
キーポリシーを変更します。	カスタマーマネージド KMS キーを使用して Amazon S3 バケットのログを暗号化していない場合は、このステップをスキップできます。 ログがカスタマーマネージド KMS キーで暗号化されている場合は、キーポリシーを変更して、ログを復号するアクセス許可を Microsoft Sentinel に付与します。以下は、キーポリシーの例です。このポリシー例では、KMS キーが別のにある場合、クロスアカウントアクセスを許可します AWS アカウント。 `{ "Version": "2012-10-17", "Id": "key-policy", "Statement": [ ... { "Sid": "Grant access to decrypt", "Effect": "Allow", "Principal": { "AWS": "<SentinelRoleArn>" }, "Action": "kms:Decrypt",`	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	"Resource": "<KeyArn>" }] }	

Microsoft Sentinel でコネクタを設定する

タスク	説明	必要なスキル
Microsoft Sentinel で設定を完了します。	1. Microsoft Sentinel で、データコネクタを選択します。 2. データコネクタギャラリーから、Amazon Web Services S3 を選択します。 3. コネクタの詳細ペインで、コネクタを開くページを選択します。 4. 設定セクションで、次の操作を行います。 a. 追加するロールに、コピーした`SentinelRoleArn` 値を入力します。 b. SQS URL で、コピーした`SentinelSQS` 値を入力します。 c. 送信先テーブルリストで、 を選択します`AWSCloudTrail` 。 5. [接続を追加] を選択します。	DevOps エンジニア

タスク	説明	必要なスキル
Amazon S3 イベント通知を Amazon SQS に送信します。	Amazon S3コンソールを使用したイベント通知の有効化と設定」の手順に従って、Amazon SQS キューにイベント通知を送信するように Amazon S3 ログ記録バケットを設定します。 Amazon SQS CloudTrail が組織全体に設定されている場合、このバケットのログにはプレフィックスがあり`<OrgID>/AWSLogs/<OrgID>/` 、 `<OrgID>`は組織 ID です。詳細については、「組織の詳細の表示」を参照してください。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
ログが取り込まれていることを確認します。	1. ログが Microsoft Sentinel に取り込まれるまで待ちます。これには数分間かかる場合があります。 2. Microsoft Sentinel で Amazon S3 データコネクタページを開き、次の操作を行います。 • Amazon S3 Data Connector のステータスが であることを確認しますConnected 。 • データ受信グラフのデータ量を確認します。 データコネクタアクティビティの検査の詳細については、Microsoft ドキュメントの「データコネクタ」を参照してください。	DevOps エンジニア

ソリューションを検証する

タスク	説明	必要なスキル
CloudWatch ログと Sentinel ログを比較します。	のデフォルト設定では AWS Control Tower、CloudTrail ログは Amazon CloudWatch に送信され、 AWS Control Tower 管理アカウントに保存されます。詳細については、「 での ログ記録とモニタリング AWS Control Tower」を参照してください。次の	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	手順を使用して、ログが Microsoft Sentinel に自動的に取り込まれることを確認します。 1. CloudWatch コンソールを開きます。 2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。 3. Select log group (s) で、 など、CloudTrail ログが保存されているロググループを選択しますaws-controltower/CloudTrail Logs 。 4. クエリエディタボックスに、 と入力しますfields eventID。 5. [Run query] (クエリの実行) を選択します。 6. 結果のエクスポートを選択し、テーブルをクリップボードにコピー (CSV) を選択します。 7. 結果をテキストエディタに貼り付けます。 8. Microsoft Sentinel クエリで使用できるように出力の形式を変更します。以下は、Kusto クエリ言語を使用する例です。	

タスク	説明	必要なスキル

	<pre>AWSCloudTrail
	| where AwsEventId in
	(
	'aa08b5fe-3bfb-391a-
	a14e-5fcebe14dab2',
	'9decd805-269c-451c-
	b75b-762f5dce59f9'
	)</pre>
	9. Microsoft Sentinel で、Amazon S3 データコネクタページを開きます。データ受信グラフの横にある「Go to log analytics」を選択します。
	10. クエリエディタボックスにクエリを入力し、実行を選択します。
	11. Microsoft Sentinel と CloudWatch で、エントリの数が同じであることを確認します。必要に応じて時間範囲を調整します。	

関連リソース

AWS ドキュメントとリソース

- AWS CLI コマンドリファレンス (AWS CLI ドキュメント)
- オプションで (ドキュメント) を設定する AWS KMS keysAWS Control Tower
- Amazon SQS の保管時の暗号化 (Amazon SQS ドキュメント)
- メーリングリストの受信者が Amazon SNS トピックの E メールからリストのすべてのユーザーのサブスクリプションを解除しないようにするにはどうすればよいですか？ (AWS ナレッジセンター)

Microsoft ドキュメント

- Microsoft Sentinel を Amazon Web Services に接続して AWS サービスログデータを取り込む
- Microsoft Sentinel の Kusto クエリ言語

を使用してアクセス AWS IAM Identity Center 許可セットをコードとして管理する AWS CodePipeline

作成者：Andre Cavalcante (AWS) と Claison Amorim (AWS)

概要

AWS IAM Identity Center は、すべての AWS アカウント およびアプリケーションへのシングルサインオン (SSO) アクセスを一元管理するのに役立ちます。IAM アイデンティティセンターでユーザー ID の作成および管理を行い、またはMicrosoft Active Directory ドメインや外部 ID プロバイダー (IdP) など既存の ID ソースに接続することもできます。IAM Identity Center は、アクセス許可セットを使用して AWS 環境へのきめ細かなアクセスを定義、カスタマイズ、割り当てるための統一された管理エクスペリエンスを提供します。アクセス許可セットは、IAM Identity Center ID ストアまたは外部 IdP のフェデレーティッドユーザーとグループに適用されます。

このパターンは、組織として管理されているマルチアカウント環境で、IAM Identity Center アクセス許可セットをコードとして管理するのに役立ちます AWS Organizations。このパターンでは、以下のことを実現できます。

- アクセス権限セットの作成、削除と更新
- ターゲット、組織単位 (OUs)、または組織ルートへのアクセス許可セットの割り当てを作成 AWS アカウント、更新、または削除します。

IAM Identity Center のアクセス許可と割り当てをコードとして管理するために、このソリューションは AWS CodeBuild と を使用する継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインをデプロイします AWS CodePipeline。リモートリポジトリに保存する JSON テンプレートでアクセス許可セットと割り当てを管理します。Amazon EventBridge ルールは、リポジトリへの変更を検出したり、ターゲット OU のアカウントへの変更を検出したりすると、 AWS Lambda 関数を開始します。Lambda 関数は、IAM アイデンティティセンターの権限セットと割り当てを更新する CI/CD パイプラインを開始します。

前提条件と制限

前提条件

- 組織として管理されるマルチアカウント環境 AWS Organizations。詳細については、「組織の作成」を参照してください。

- IAM ID センターであり、ID ソースで有効化と設定が行われます。詳細については、IAM アイデンティティセンタードキュメントの「開始方法」を参照してください。
- 以下の委任管理者として登録されているメンバーアカウント AWS のサービス。
  - IAM Identity Center – 手順については、IAM Identity Center ドキュメントの「メンバーアカウントを登録する」を参照してください。
  - AWS Organizations – 手順については、「 の委任管理者 AWS Organizations」を参照してください。このアカウントには、アカウントと OUs一覧表示および記述するアクセス許可が必要です。

> **Note**
>
> 両方のサービスの委任管理者と同じアカウントを使用する必要があります。

- IAM Identity Center の委任管理者アカウントと組織の管理アカウントに AWS CloudFormation スタックをデプロイするアクセス許可。詳細については、CloudFormation ドキュメントの「アクセスの制御」を参照してください。
- IAM アイデンティティセンターの委任管理者アカウントの Amazon Simple Storage Service (Amazon S3) バケット。このバケットにアーティファクトコードをアップロードします。手順については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。
- 組織の管理アカウントのアカウント ID。手順については、AWS アカウント 「ID の検索」を参照してください。
- GitHub などのソースコードホスト内のリポジトリ。

制約事項

- このパターンは、単一アカウント環境または組織として管理されていないアカウントのアクセス許可セットを管理または割り当てるために使用することはできません AWS Organizations。
- アクセス権限セット名、割り当て ID、IAM アイデンティティセンタープリンシパルのタイプと ID は、デプロイ後に変更することはできません。
- このパターンは、「カスタムアクセス権限」の作成と管理に役立ちます。このパターンを使用して「定義済みの権限」を管理しまたは割り当てることはできません。
- このパターンは、組織の管理アカウントのアクセス権限セットの管理には使用できません。

アーキテクチャ

ターゲット アーキテクチャ

この図表は、次のワークフローを示しています：

1. ユーザーは、次の変更の全部または一部を実行します。
   - GitHub などのリモートリポジトリに 1 つ以上の変更をコミットします
   - の OU のアカウントを変更します AWS Organizations
2. ユーザーがリモートリポジトリへの変更をメインブランチにコミットすると、パイプラインが開始されます。

   ユーザーが OU のアカウントを変更した場合、`MoveAccount`EventBridge ルールは変更を検出し、組織の管理アカウントで Lambda 関数を開始します。
3. 開始された Lambda 関数は CodePipeline で CI/CD パイプラインを開始します。
4. CodePipeline は `TemplateValidation` CodeBuild プロジェクトを開始します。`TemplateValidation` CodeBuild プロジェクトは、リモートリポジトリの Python スクリプトを使用して、アクセス許可セットテンプレートを検証します。CodeBuild は以下の検証を行います。
   - アクセス権限セット名は一意です。
   - 割り当てステートメント ID (`Sid`) は一意です。
   - `CustomPolicy` パラメータ内のポリシー定義は有効です。(この検証では が使用されます AWS Identity and Access Management Access Analyzer。)
   - 管理ポリシーの Amazon リソースネーム (ARN) は有効です。
5. `Deploy` CodeBuild プロジェクトの`PermissionSet`アクショングループは、 AWS SDK for Python (Boto3) を使用して IAM Identity Center のアクセス許可セットを削除、作成、または更新します。この `SSOPipeline:true` タグが付いたアクセス権限セットのみが影響を受けます。このパイプラインを通じて管理されるすべてのアクセス権限セットにはこのタグが付きます。
6. `Deploy` CodeBuild プロジェクトの`Assignments`アクショングループは、Terraform を使用して IAM Identity Center での割り当てを削除、作成、または更新します。Terraform バックエンド状態ファイルは、同じアカウントの Amazon S3 バケットに保存されます。
7. CodeBuild は IAM アイデンティティセンターのアクセス権限セットと割り当てを更新します。

自動化とスケール

マルチアカウント環境のすべての新しいアカウントは の特定の組織単位に移動されるため AWS Organizations、このソリューションは自動的に実行され、割り当てテンプレートで指定したすべてのアカウントに必要なアクセス許可セットが付与されます。追加の自動化やスケーリングアクションは必要ありません。

大規模な環境では、IAM Identity Center に対する API リクエスト数が多いと、このソリューションの実行が遅くなることがあります。Terraform と Boto3 は、パフォーマンスの低下を最小限に抑えるために自動的にスロットリングを管理します。

ツール

AWS のサービス

- AWS CloudFormation は、 AWS リソースの設定、迅速かつ一貫したプロビジョニング、および AWS アカウント 全体のライフサイクル全体にわたる管理に役立ちます AWS リージョン。
- AWS CodeBuild は、ソースコードのコンパイル、ユニットテストの実行、デプロイ可能なアーティファクトの生成に役立つフルマネージド型のビルドサービスです。
- AWS CodePipeline は、ソフトウェアリリースのさまざまなステージを迅速にモデル化して設定し、ソフトウェアの変更を継続的にリリースするために必要なステップを自動化するのに役立ちます。
- Amazon EventBridge は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS IAM Identity Center は、すべての AWS アカウント およびクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理するのに役立ちます。
- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを と統合するのに役立つソフトウェア開発キットです AWS のサービス。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。

コードリポジトリ

このパターンのコードは、「aws-iam-identity-center-pipeline」リポジトリで利用できます。リポジトリの templates フォルダには、アクセス権限セットと割り当ての両方のサンプルテンプレートが含まれています。また、CI/CD パイプラインと AWS リソースをターゲットアカウントにデプロイするための AWS CloudFormation テンプレートも含まれています。

ベストプラクティス

- アクセス権限セットと割り当てテンプレートの変更を開始する前に、組織のアクセス権限セットを計画することをお勧めします。アクセス権限はどうあるべきか、そのアクセス権限セットをどのアカウントまたは OU に適用すべきか、そしてどの IAM Identity Center プリンシパル (ユーザーまたはグループ) がアクセス権限セットの影響を受けるべきかを検討してください。アクセス許可セット名、割り当て ID、IAM アイデンティティセンタープリンシパルのタイプと ID は、デプロイ後に変更することはできません。
- 最小限権限原則に従い、タスクの実行に必要最小限の権限を付与します。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「最小特権の付与」と「セキュリティのベストプラクティス」を参照してください。

エピック

プラン、アクセス権限セット、割り当て

タスク	説明	必要なスキル
リポジトリをクローン作成します。	コマンドシェルで、次のコマンドを入力します。これにより、GitHub から「 aws-iam-identity-center-pipeline 」リポジトリがクローンされます。 `git clone https://github.com/aws-samples/aws-iam-identity-center-pipeline.git`	DevOps エンジニア
アクセス権限セットを定義します。	1. 複製されたリポジトリで `templates/permissionsets` フォルダに移動	DevOps エンジニア

タスク	説明	必要なスキル
	し、使用可能なテンプレートのいずれかを開きます。 2. Name パラメータに、アクセス権限セットの名前を入力します。この値は一意でなければならず、デプロイ後に変更することはできません。 3. Description パラメータには、アクセス権限セットをユースケースなど簡単に説明します。 4. SessionDuration パラメータで、ユーザーが にサインインできる時間の長さを指定します AWS アカウント。PT4H の 4 時間など「ISO-8601 継続時間形式」(ウィキペディア) を使用します。値が定義されていない場合、IAM アイデンティティセンターはデフォルトで 1 時間です。 5. RelayState パラメータで、ユーザーのロールに最も適したコンソールへのクイックアクセスを提供する URL を指定します。 6. アクセス権限セットのポリシーをカスタマイズします。以下のパラメータはすべてオプションで、導入後に変更できます。アクセス権限セット内のポリシーを	

タスク	説明	必要なスキル

	定義するには、少なくとも 1 つのパラメータを使用する必要があります。
	• `ManagedPolicies` パラメータに、割り当てたいAWS 管理ポリシーの ARNs を入力します。
	• `CustomerManagedPolicies` パラメータに、割り当てる任意の「カスタマー管理ポリシー」名を入力します。ARN は使用しません。
	• `PermissionBoundary` パラメータで、次の操作を実行してアクセス許可の境界を割り当てます。
	  • AWS 管理ポリシーをアクセス許可の境界として使用している場合は、 に `PolicyType` と入力しAWS、 にポリシーの ARN `Policy`を入力します。
	  • カスタマー管理ポリシーをアクセス許可の境界として使用している場合は、 に `PolicyType` と入力し`Customer`、 にポリシーの名前`Policy`を	

タスク	説明	必要なスキル
	入力します。ARN は使用しません。 • `CustomPolicy` パラメータでは、割り当てる JSON 形式のカスタムポリシーを定義します。JSON ポリシー構造の詳細については、「JSON ポリシーの概要」を参照してください。 7. アクセス権限セットテンプレートを保存して閉じます。アクセス権限セット名と同じ名前でファイルを保存することをお勧めします。 8. このプロセスを繰り返して組織に必要な数のアクセス権限セットを作成し、不要なサンプルテンプレートをすべて削除します。	

タスク	説明	必要なスキル
割り当てを定義します。	1. 複製されたリポジトリで `templates/assignments` フォルダに移動し、`iam-identitycenter-assigments.json` を開きます。このファイルでは、アクセス許可セットを AWS アカウント または OUs に割り当てる方法について説明します。 2. `SID` パラメータには、割り当ての ID を入力します。この値は一意でなければならず、デプロイ後に変更することはできません。 3. `Target` パラメータでは、アクセス権限セットを適用するアカウントまたは組織を定義します。有効な値は、アカウント IDs、OUs、または ですroot。 は、管理アカウントを除く組織内のすべてのメンバーアカウントに権限セットをroot割り当てます。値を二重引用符で囲んで入力し、複数の値をコンマで区切ります。アカウント IDs と OUs、`{{account_name}}:{{account_id}}` または は のパターンに従う必要があります`{{ou_name}}:{{ou_id}}` 。ネストさ	DevOps エンジニア

タスク	説明	必要なスキル
	れた OUs に再帰的にアクセス許可を割り当てる場合は、最後にワイルドカードが付いた OU パターンを使用します。例: `{{ou_name}}:{{ou_id}}:*` 4. `PrincipalType` パラメータには、アクセス権限セットの影響を受ける IAM Identity Center プリンシパルのタイプを入力します。有効な値は USER または GROUP です。この値は、デプロイ後に変更することはできません。 5. `PrincipalID` パラメータには、アクセス権限セットの影響を受ける IAM アイデンティティセンターのアイデンティティストアにおけるユーザーまたはグループの名前を入力します。この値は、デプロイ後に変更することはできません。 6. `PermissionSetName` パラメータに、割り当てるアクセス権限セットの名前を入力します。 7. ステップ 2 ~ 6 を繰り返して、このファイルに必要な数のアサインメントを作成します。通常、アクセス権限セットごとに 1 つの割り当てがあります。必要のな	

タスク	説明	必要なスキル
	いサンプル課題はすべて削除してください。 8. `iam-identitycenter-assigments.json` ファイルを保存して閉じます。	

アクセス権限セットと割り当てをデプロイ

タスク	説明	必要なスキル
IAM アイデンティティセンターの委任管理者アカウントにリソースをデプロイします。	1. IAM Identity Center の委任管理者アカウントで、AWS CloudFormation コンソールを開きます。 2. `iam-identitycenter-pipeline.yaml` テンプレートをデプロイします。スタックにはわかりやすい名前を付け、指示に従い、パラメータを更新します。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。	DevOps エンジニア
AWS Organizations 管理アカウントにリソースをデプロイします。	1. 組織の管理アカウントにサインインします。 2. AWS CloudFormation コンソールを開きます。 3. ナビゲーションバーで、現在表示されている の名前を選択します AWS リージョン。次に、us-	DevOps エンジニア

タスク	説明	必要なスキル
	east-1 リージョンを選択します。このリージョンは、MoveAccount EventBridge ルールが組織の変更に関連する AWS CloudTrail イベントを検出できるようにするために必要です。 4. iam-identitycenter-organization テンプレートをデプロイします。スタックにはわかりやすい名前を付け、指示に従い、パラメータを更新します。手順については、「CloudFormation のドキュメント」の「スタックの作成」を参照してください。	
リモートリポジトリのセットアップを完了します。	AWS CodeConnections 接続のステータスを から PENDINGに変更しますAVAILABLE 。この接続は、CloudFormation スタックをデプロイしたときに作成されました。手順については、CodeConnections ドキュメントの「保留中の接続の更新」を参照してください。	DevOps エンジニア

タスク	説明	必要なスキル
リモートリポジトリにファイルをアップロードします。	`aws-samples` リポジトリからダウンロードし、前のステップで編集したすべてのファイルをリモートリポジトリにアップロードします。`main` ブランチへの変更はパイプラインを開始し、アクセス許可セットと割り当てを作成または更新します。	DevOps エンジニア

アクセス権限セットと割り当てを更新

タスク	説明	必要なスキル
アクセス権限セットと割り当てを更新します。	`MoveAccount` Amazon EventBridge ルールが組織内のアカウントへの変更を検出すると、CI/CD パイプラインが自動的に起動し、アクセス権限セットを更新します。たとえば、割り当ての JSON ファイルで指定された OU にアカウントを追加すると、CI/CD パイプラインはそのアクセス権限セットを新しいアカウントに適用します。 デプロイされたアクセス許可セットと割り当てを変更する場合は、JSON ファイルを更新してからリモートリポジトリにコミットします。 CI/CD パイプラインで以前にデプロイされたアクセス権限	DevOps エンジニア

タスク	説明	必要なスキル
	セットとアソシエーションを管理する場合は、次の点に注意してください。 • アクセス権限セット名を変更すると、CI/CD パイプラインは元のアクセス権限セットを削除し、新しいアクセス権限セットを作成します。 • このパイプラインは、`SSOPipeline:true` タグが付いたアクセス権限セットのみを管理します。 • リポジトリ内の同じフォルダに複数のアクセス権限セットと割り当てテンプレートを保存することができます。 • テンプレートを削除すると、パイプラインはアサインまたはアクセス権限セットを削除します。 • 割り当て JSON ブロック全体を削除すると、パイプラインは IAM アイデンティティセンターからその割り当てを削除します。 • に割り当てられたアクセス許可セットを削除することはできません AWS アカウント。まず、アクセス権限セットの割り当てを解除する必要があります。	

トラブルシューティング

問題	ソリューション
アクセス拒否エラー	CloudFormation テンプレートおよびテンプレート内で定義されたリソースをデプロイするために必要なアクセス権限があることを確認してください。詳細については、CloudFormation ドキュメントの「アクセスの制御」を参照してください。
検証段階におけるパイプラインエラー	このエラーは、アクセス権限セットまたは割り当てテンプレートにエラーがある場合に表示されます。 1. CodeBuild での「ビルドの詳細」を表示します。 2. ビルドログで、ビルドが失敗した原因に関する詳細情報が記載されている検証エラーを見つけてください。 3. アクセス権限セットまたは割り当てテンプレートを更新し、リポジトリにコミットします。 4. CI/CD パイプラインは CodeBuild プロジェクトを再起動します。ステータスをモニタリングして、検証エラーが解決されたことを確認します。

関連リソース

- 「アクセス権限セット」(IAM アイデンティティセンターのドキュメント)

AWS Secrets Manager を使用して認証情報を管理

作成者: Durga Prasad Cheepuri (AWS)

概要

AWS Secrets Manager を使用して Java Spring アプリケーションのデータベース認証情報を動的に取得します。

以前は、データベースから情報を取得するカスタムアプリケーションを作成する場合は、アプリケーションにデータベースに直接アクセスするための認証情報 (シークレット) を埋め込む必要がありました。認証情報をローテーションする必要がある場合、新しい認証情報を使用するためにアプリケーションを更新し、デプロイするために時間を費やす必要があります。認証情報を共有している複数のアプリケーションがあり、そのうちの 1 つを更新しなかった場合、アプリケーションは中断されます。このリスクのために、多くのお客様は定期的に認証情報を更新せずに、実際のところは代わりに別のリスクを選択していました。

Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を、Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。シークレットはそこに存在しないため、これは、コードを調べている誰かがシークレットを漏らさないようにするのに役立ちます。また、指定したスケジュールに従って自動的にシークレットを更新するように Secrets Manager を設定することができます。これにより、長期のシークレットを短期のシークレットに置き換えることが可能となり、侵害されるリスクが大幅に減少します。詳細については、「AWS Secrets Manager のドキュメント」を参照してください。

前提条件と制限

前提条件

- Secrets Manager に対するアクセス権のある AWS アカウント
- Java Spring アプリケーション

アーキテクチャ

ソーステクノロジースタック

- データベースにアクセスするコードを含む Java Spring アプリケーションであり、アプリケーションプロパティファイルから管理されるDB 認証情報も含んでいます。

ターゲットテクノロジースタック

- データベースにアクセスするコードを含む Java Spring アプリケーションであり、Secrets Manager で管理されるDB 認証情報も含んでいます。アプリケーションプロパティファイルには、Secrets Manager のシークレットが格納されています。

Secrets Manager とアプリケーションの統合

ツール

- Secrets Manager — 「AWS Secrets Manager」は、シークレットの管理をより簡単にする AWS サービスです。シークレットとは、データベース認証情報、パスワード、サードパーティーの API キーなどの任意のテキストです。Secrets Manager コンソール、Secrets Manager コマンドラインインターフェイス (CLI)、Secrets Manager API および SDK を使用すると、これらのシークレットへのアクセスを一元的に保存および制御できます。

エピック

シークレットを Secrets Manager に保存します。

タスク	説明	必要なスキル
Secrets Manager で DB 認証情報をシークレットとして保存します。	Secrets Manager のドキュメントの「シークレットの作成」の手順に従い、Amazon Relational Database Service (Amazon RDS) またはその他の DB 認証情報をSecrets Manager のシークレットとして保存します。	Sys Admin
Spring アプリケーションが Secrets Manager にアクセスするための権限を設定します。	Java Spring アプリケーションが Secrets Manager をどのように使用するかに応じて、適切な権限を設定します。シー	Sys Admin

タスク	説明	必要なスキル
	クレットへのアクセスを制御するには、Secrets Manager のマニュアルに記載されている情報に基づいて、Secrets Managerドキュメントの「Secrets Manager でアイデンティティベースのポリシー (IAM ポリシー) を使用する」と「Secrets Manager のリソースベースのポリシーを使用する」セクションに記載されている情報に基づいてポリシーを作成します。「Secrets Manager のドキュメント」の「シークレット値の取得」セクションの手順に従ってください。	

Spring アプリケーションの更新

タスク	説明	必要なスキル
Secrets Manager を使用するには JAR 依存関係を追加します。	詳細については、追加情報セクションを参照してください。	Java 開発者
シークレットの詳細を Spring アプリケーションに追加します。	アプリケーションプロパティファイルをシークレット名、エンドポイント、AWS リージョンで更新します。例として、「追加情報」セクションを参照してください。	Java 開発者
Java で DB 認証情報取得コードを更新します。	アプリケーションで、DB 認証情報を取得する Java コード	Java 開発者

タスク	説明	必要なスキル
	を更新して Secrets Manager からそれらの詳細を取得します。コードの例として、追加情報セクションを参照してください。	

関連リソース

- 「AWS Secrets Manager のドキュメント」
- 「Secrets Manager に対する ID ベースのポリシー (IAM ポリシー) と ABAC の使用」
- 「Secrets Manager のリソースベースのポリシーを使用する」
- 「サンプルコード」

追加情報

Secrets Manager を使用するための JAR 依存関係の追加

Maven:

```
<groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-secretsmanager</artifactId>
    <version>1.11. 355 </version>
```

Gradle:

```
compile group: 'com.amazonaws', name: 'aws-java-sdk-secretsmanager', version:
 '1.11.355'
```

アプリケーションプロパティファイルをシークレット名の詳細で更新

```
spring.aws.secretsmanager.secretName=postgres-local
spring.aws.secretsmanager.endpoint=secretsmanager.us-east-1.amazonaws.com
spring.aws.secretsmanager.region=us-east-1
```

Java で DB 認証情報取得コードの更新

```
String  secretName  =  env.getProperty("spring.aws.secretsmanager.secretName");
String  endpoints  =  env.getProperty("spring.aws.secretsmanager.endpoint");
String  AWS Region  =  env.getProperty("spring.aws.secretsmanager.region");
AwsClientBuilder.EndpointConfiguration  config  =  new
  AwsClientBuilder.EndpointConfiguration(endpoints, AWS Region);
AWSSecretsManagerClientBuilder  clientBuilder  =
  AWSSecretsManagerClientBuilder.standard();
clientBuilder.setEndpointConfiguration(config);
AWSSecretsManager  client  =  clientBuilder.build();

ObjectMapper  objectMapper  =  new  ObjectMapper();

JsonNode  secretsJson  =  null;

ByteBuffer  binarySecretData;

GetSecretValueRequest  getSecretValueRequest  =  new
  GetSecretValueRequest().withSecretId(secretName);

GetSecretValueResult  getSecretValueResponse  =  null;

try  {
     getSecretValueResponse  =  client.getSecretValue(getSecretValueRequest);
    }

catch  (ResourceNotFoundException  e)  {
     log.error("The requested secret "  +  secretName  +  " was not found");
    }

catch  (InvalidRequestException  e)  {
     log.error("The request was invalid due to: "  +  e.getMessage());
     }

catch  (InvalidParameterException  e)  {
     log.error("The request had invalid params: "  +  e.getMessage());
     }
if  (getSecretValueResponse  ==  null)  {
     return  null;
     }  // Decrypted secret using the associated KMS key // Depending on whether the
 secret was a string or binary, one of these fields will be populated
```

```
String secret = getSecretValueResponse.getSecretString();

if (secret != null) {
     try {
                secretsJson  =  objectMapper.readTree(secret);
           }

     catch  (IOException  e)  {
                log.error("Exception while retrieving secret values: "  +
  e.getMessage());
           }
}

else  {
     log.error("The Secret String returned is null");

     return null;

     }
     String  host  =  secretsJson.get("host").textValue();
     String  port  =  secretsJson.get("port").textValue();
     String  dbname  =  secretsJson.get("dbname").textValue();
     String  username  =  secretsJson.get("username").textValue();
     String  password  =  secretsJson.get("password").textValue();
}
```

セキュリティグループの ElastiCache クラスターをモニタリング

作成者：Susanne Kangnoh (AWS) と Archit Mathur (AWS)

概要

Amazon ElastiCache は、クラウド内のインメモリデータストアまたはキャッシュ環境を分散するための、高性能、スケーラブル、費用対効果の高いキャッシュソリューションを提供する Amazon Web Services (AWS) サービスです。スループットが高く、レイテンシーが低いメモリ内データストアからデータを取得します。この機能により、キャッシュ、セッションストア、ゲーム、地理空間サービス、リアルタイム分析、キューイングなどのリアルタイムユースケースによく利用されています。ElastiCache には Redis と Memcached データストアがあり、どちらもミリ秒未満の応答時間を実現します。

セキュリティグループは、インバウンドとアウトバウンドのトラフィックを制御することで、ElastiCache インスタンスの仮想ファイアウォールとして機能します。セキュリティグループは、サブネットレベルでなくインスタンスレベルで動作します。セキュリティグループごとに、インスタンスへのインバウンドトラフィックをコントロールするルールと、アウトバウンドトラフィックをコントロールする一連のルールを個別に追加します。許可ルールを指定できますが、拒否ルールは指定できません。

このパターンは、API コールをモニタリングし、CreateReplicationGroup、CreateCacheCluster、ModifyCacheCluster、および ModifyReplicationGroup オペレーションの Amazon CloudWatch Events でイベントを生成するセキュリティコントロールを提供します。このイベントは、Python スクリプトを実行する AWS Lambda 関数を呼び出します。この関数は、イベント JSON 入力からレプリケーショングループ ID を取得し、次のチェックを実行してセキュリティ違反がないかを判断します。

- クラスターのセキュリティグループが Lambda 関数で設定されているセキュリティグループと一致するかを確認します。
- クラスターのセキュリティグループが一致しない場合、関数は Amazon Simple Notification Service (Amazon SNS) 通知により、指定したメールアドレスに違反メッセージを送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- 提供される Lambda コードをアップロードする Amazon Simple Storage Service (Amazon S3) バケット。
- 違反の通知を受信するメールアドレス
- すべての API ログにアクセスするための ElastiCache ロギングが有効になっています。

制約事項

- この検出コントロールはリージョン別であり、モニタリング AWS リージョン する各 にデプロイする必要があります。
- コントロールは、仮想プライベートクラウド (VPC) で実行されているリプリケーショングループをサポートします。

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- を使用している場合は AWS Organizations、AWS CloudFormation StackSets を使用して、このテンプレートをモニタリングする複数のアカウントにデプロイできます。

ツール

AWS サービス

- Amazon ElastiCache を使用すると、 で分散インメモリキャッシュ環境を簡単にセットアップ、管理、スケーリングできます AWS クラウド。このサービスは、高パフォーマンスでコスト効率に優れ、さらにサイズ変更も可能なインメモリキャッシュを提供しながら、分散キャッシュ環境のデプロイと管理に伴う複雑性を排除します。ElastiCache は Redis エンジンと Memcached エンジンの両方で動作します。
- AWS CloudFormation は、 AWS リソースのモデル化とセットアップ、迅速かつ一貫したプロビジョニング、ライフサイクル全体の管理に役立ちます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。スタックは、複数の と にまたがって管理 AWS アカウント およびプロビジョニングできます AWS リージョン。

- Amazon CloudWatch Events は、 AWS リソースの変更を記述するシステムイベントのほぼリアルタイムのストリームを提供します。CloudWatch Events は、運用上の変更が生じると同時にそれらを認識して対応し、環境に応答するためのメッセージを送信する、機能をアクティブ化する、変更を行う、および状態情報を収集することによって、必要に即した是正措置を講じます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードの実行をサポートするコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスを含む、パブリッシャーとクライアント間のメッセージの送信を調整し管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、次の 2 つのファイルを含む添付ファイルが含まれます。

- `ElastiCacheAllowedSecurityGroup.zip` は、セキュリティコントロール (Lambda コード) を含む圧縮ファイルです。
- `ElastiCacheAllowedSecurityGroup.yml` は、セキュリティコントロールをデプロイする CloudFormation テンプレートです。

これらのファイルの使用方法については、「エピック」セクションを参照してください。

エピック

セキュリティコントロールをデプロイ

タスク	説明	必要なスキル
S3 バケットにコードをアップロードします。	新しい S3 バケットを作成するか、既存の S3 バケットを使用して、添付 `ElastiCacheAllowedSecurityGroup.zip` ファイル (Lambda コード) をアッ	クラウドアーキテクト

タスク	説明	必要なスキル
	プロードします。このバケットは、評価するリソース AWS リージョン と同じ にある必要があります。	
CloudFormation テンプレートをデプロイする。	S3 バケット AWS リージョン と同じ で CloudFormation コンソールを開き、添付ファイルに提供されている`ElastiCacheAllowedSecurityControl.yml` ファイルをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を入力します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。この S3 バケットには Lambda コードの .zip ファイルが含まれており、CloudFormation テンプレートおよび評価されるリソース AWS リージョン と同じ にある必要があります。	クラウドアーキテクト
S3 キーを入力します。	S3 バケット内の Lambda コードの.zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例：`ElasticCacheAllowedSecurity`	クラウドアーキテクト

タスク	説明	必要なスキル
	Group.zip または controls/ElasticCacheAllowedSecurityGroup.zip)。	
メールアドレスを入力します。	違反の通知を受信するメールアドレスを入力します。	クラウドアーキテクト
ログレベルを指定します。	ログ記録レベルと詳細レベルを定義します。Info アプリケーションの進行状況に関する詳細な情報メッセージを指定し、デバッグのみに使用されます。Error それでもアプリケーションの実行を継続できるエラーイベントを指定します。Warning 潜在的に危険有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
メールサブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定した E メールアドレスにサブスクリプション E メールメッセージが送信されます。通知を受信するには、この E メールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- AWS CloudFormation コンソールでのスタックの作成 (AWS CloudFormation ドキュメント)
- Amazon VPCsと ElastiCache セキュリティ (Amazon ElastiCache ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon EMR クラスターの起動時に転送中の暗号化をモニタリングする

作成者: Susanne Kangnoh (AWS)

概要

このパターンは、起動時に Amazon EMR クラスターを監視し、転送中の暗号化が有効になっていない場合はアラートを送信するセキュリティコントロールを提供します。

Amazon EMR は、Apache Hadoop などのビッグデータフレームワークを簡単に実行してデータを処理および分析できるようにするウェブサービスです。Amazon EMR では、マッピングとステップ削減を並行実行することで、膨大な量のデータを費用対効果の高い方法で処理できます。

データ暗号化は、権限のないユーザーが保管中のデータや転送中のデータにアクセスしたり読み取ったりすることを防ぎます。[保管中のデータ] とは、各ノードのローカルファイルシステム、Hadoop 分散ファイルシステム (HDFS)、または Amazon Simple Storage Service (Amazon S3) を通じて EMR ファイルシステム (EMRFS) などのメディアに保存されているデータを指します。[転送中のデータ] とは、ネットワーク上を移動し、ジョブ間で転送中のデータを指します。転送中の暗号化は、Apache Spark、Apache TEZ、Apache Hadoop、Apache HBase、Presto のオープンソース暗号化機能をサポートしています。暗号化を有効にするには、AWS コマンドラインインターフェイス (AWS CLI)、コンソール、AWS SDK からセキュリティ設定を作成し、データ暗号化設定を指定します。転送中の暗号化用の暗号化アーティファクトは、次の 2 つの方法で提供できます。

- 証明書の圧縮済みファイルを Amazon S3 にアップロードします。
- 暗号化アーティファクトを提供するカスタム Java クラスを参照する。

このパターンに含まれるセキュリティコントロールは API コールを監視し、[RunJobFlow] アクションで Amazon CloudWatch Events イベントを生成します。このイベントは、Python スクリプトを実行する AWS Lambda 関数を呼び出します。この関数は、イベント JSON 入力から EMR クラスター ID を取得し、次のチェックを実行してセキュリティ違反があるかどうかを判断します。

- EMR クラスターに Amazon EMR 固有のセキュリティ設定があるかどうかを確認します。
- クラスターにセキュリティ設定がある場合は、転送中の暗号化が有効になっているかどうかを確認します。
- クラスターにセキュリティ設定がない場合は、Amazon Simple Notification Service (Amazon SNS) を使用して、指定したメールアドレスにアラートを送信します。通知には、EMR クラスター

名、違反の詳細、AWS リージョンとアカウント情報、および通知の送信元の AWS Lambda ARN (Amazon リソースネーム) を指定します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- このパターンで提供される Lambda コードをアップロードする S3 バケットです。
- 違反の通知を受け取るメールアドレス。
- すべての API ログにアクセスするための Amazon EMR ロギングが有効になっています。

制約事項

- この検出統制は地域ごとに異なり、監視する各 AWS リージョンに導入する必要があります。

製品バージョン

- Amazon EMR リリース 4.8.0 またはそれ以降。

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、「AWS Cloudformation StackSets」を使用して、監視したい複数のアカウントにテンプレートをデプロイできます。

ツール

AWS サービス

- 「Amazon EMR」— Amazon EMR は、AWS でビッグデータフレームワーク (「Apache Hadoop」や「Apache Spark」など) の実行を簡素化して、ビッグデータを処理および分析するマ

ネージドクラスタープラットフォームです。これらのフレームワークと、関連するオープンソースプロジェクトを使用することで、分析用のデータやビジネスインテリジェンスワークロードを処理できます。さらに、Amazon EMR を使用して、Amazon S3 や Amazon DynamoDB などの他の AWS データストアやデータベースへ、大量のデータを変換し、移動することができます。

- 「AWS Cloudformation」— AWS CloudFormation は、AWS リソースのモデル化とセットアップ、迅速かつ一貫したプロビジョニング、ライフサイクル全体にわたる管理を支援します。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理およびプロビジョニングできます。
- 「Amazon CloudWatch Events」— Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。CloudWatch Events は、運用上の変更が生じると同時にそれらを認識して対応し、環境に応答するためのメッセージを送信する、機能をアクティブ化する、変更を行う、および状態情報を収集することによって、必要に即した是正措置を講じます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「AWS SNS」— Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスを含む、パブリッシャーとクライアント間のメッセージ送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、次の 2 つのファイルを含む添付ファイルが含まれます。

- `EMRInTransitEncryption.zip` は、セキュリティコントロール (Lambda コード) を含む圧縮ファイルです。
- `EMRInTransitEncryption.yml` は、セキュリティコントロールをデプロイする CloudFormation テンプレートです。

これらのファイルの使用方法については、「エピック」セクションを参照してください。

エピック

セキュリティコントロールをデプロイ

タスク	説明	必要なスキル
S3 バケットにコードをアップロードします。	新しい S3 バケットを作成するか、既存の S3 バケットを使用して添付 `EMRInTransitEncryption.zip` ファイル (Lambda コード) をアップロードします。このバケットは、CloudFormation テンプレートと同じ AWS リージョンにある必要があります。また、評価するリソースが含まれている必要があります。	クラウドアーキテクト
CloudFormation テンプレートをデプロイする。	S3 バケットと同じ AWS リージョンで Cloudformation コンソールを開き、添付ファイルで提供されている `EMRInTransitEncryption.yml` ファイルをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト、

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を入力します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。この S3 バケットには Lambda コードの.zip ファイルが含まれてお	クラウドアーキテクト

タスク	説明	必要なスキル
	り、CloudFormation のテンプレートと評価するリソースと同じ AWS ゾーンにある必要があります。	
S3 キーを提供します。	S3 バケット内の Lambda コードの.zip ファイルの場所を、先頭にスラッシュを付けずに指定します (たとえば、`EMRInTransitEncryption.zip` 、`controls/EMRInTransitEncryption.zip`)。	クラウドアーキテクト
メールアドレスを提供します。	違反の通知を受け取る、有効なメールアドレスを指定します。	クラウドアーキテクト
ログ記録レベルを指定します。	Lambda ログのロギングレベルと詳細度を指定します。`Info` はアプリケーションの進行状況に関する詳細な情報メッセージを指定するため、デバッグにのみ使用します。`Error` は、アプリケーションの実行を継続できるエラーイベントを指定します。`Warning` は、潜在的に有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
メールサブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定した E メールアドレスにサブスクリプション E メールメッセージが送信されます。通知を受信するには、この E メールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「AWS CloudFormation コンソールでのスタックの作成」(AWS CloudFormation ドキュメント)
- 「暗号化オプション」(Amazon EMR ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

Amazon ElastiCache クラスターの保存時の暗号化をモニタリングする

作成者: Susanne Kangnoh (AWS)

概要

Amazon ElastiCache は、高パフォーマンス、拡張性、コストパフォーマンスの高いキャッシュソリューションを提供するAmazon Web Services (AWS) サービスで、インメモリデータストアやキャッシュ環境をクラウドに分散します。スループットが高く、レイテンシーが低いメモリ内データストアからデータを取得します。この機能により、キャッシュ、セッションストア、ゲーム、地理空間サービス、リアルタイム分析、キューイングなどのリアルタイムユースケースによく利用されています。ElastiCache には Redis データストアと Memcached データストアがあり、どちらもミリ秒未満の応答時間を実現します。

データ暗号化は、権限のないユーザーがRedisクラスターや関連するキャッシュストレージシステムで利用可能な機密データを読み取ることを防ぐのに役立ちます。このデータには、保管中のデータと呼ばれる、永続的なメディアに保存されているデータや、転送中のデータと呼ばれる、ネットワークを介した転送の間に傍受される可能性のあるデータが含まれます。

AtRestEncryptionEnabled パラメータを [true] に設定することで、レプリケーション・グループの作成時に ElastiCache for Redis のアットレスト暗号化を有効にすることができます。このパラメータを有効にすると、同期、バックアップ、スワップ操作中にディスクが暗号化され、Amazon Simple Storage Service (Amazon S3) に保存されたバックアップが暗号化されます。既存のレプリケーショングループ上で保管時の暗号化を有効にすることはできません。レプリケーショングループを作成するときに、以下の 2 つの方法で保管中の暗号化を有効にすることができます

- [デフォルト] オプションを選択すると、保存時にはサービス管理型の暗号化が使用されます。
- カスタマー管理のキーを使用し、AWS Key Management Service (AWS KMS) からキー ID または Amazon リソースネーム(ARN) を指定します。

このパターンは、API 呼び出しを監視し、[CreateReplicationGroup] オペレーションで Amazon CloudWatch Events イベントを生成するセキュリティコントロールを提供します。このイベントは、Python スクリプトを実行する AWS Lambda 関数を呼び出します。この関数は、イベント JSON 入力からレプリケーショングループ ID を取得し、次のチェックを実行してセキュリティ違反がないかどうかを判断します。

- [ATRestEncryptionEnabled] キーが存在するかどうかを確認します。

- [AtRestEncryptionEnabled] が存在する場合は、値が正しいかどうかを確認します。
- [AtRestEncryptionEnabled] の値が [false] に設定されている場合、Amazon Simple Notification Service (Amazon SNS) 通知を使用して、違反を追跡し、指定したメールアドレスに違反メッセージを送信する変数を設定します。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 提供されたLambda コードをアップロードする S3 バケットに
- 違反の通知を受信するメールアドレス
- すべての API ログにアクセスするための ElastiCache ロギングが有効になっています。

制約事項

- この検出統制は地域的なものであるため、モニタリング対象の各 AWS リージョンにデプロイする必要があります。
- コントロールは、仮想プライベートクラウド (VPC) で実行中のレプリケーショングループをサポートします。
- このコントロールは、以下のノードタイプを実行するレプリケーショングループをサポートします。
  - R5、R4、R3
  - M5、M4、M3
  - T3、T2

製品バージョン

- ElastiCache for Redis バージョン 3.2.6 またはそれ以降

アーキテクチャ

ワークフローアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、監視したい複数のアカウントにこのテンプレートをデプロイできます。

ツール

AWS サービス

- 「Amazon ElastiCache」 — Amazon ElastiCache を使用すると、AWS クラウドでのインメモリ分散キャッシュ環境のセットアップ、管理、およびスケーリングが簡単になります。高パフォーマンスでサイズ変更可能、コスト効果的なインメモリキャッシュを提供すると同時に、分散キャッシュ環境のデプロイや管理に関連する複雑さを解消します。ElastiCache は Redis エンジンと Memcached エンジンの両方で動作します。
- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。複数の AWS アカウントと AWS リージョンにまたがるスタックを管理およびプロビジョニングできます。
- 「Amazon CloudWatch Events」— Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。CloudWatch Events は、運用上の変更が生じると同時にそれらを認識して対応し、環境に応答するためのメッセージを送信する、機能をアクティブ化する、変更を行う、および状態情報を収集することによって、必要に即した是正措置を講じます。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。課金は実際に消費したコンピューティング時間に対してのみ発生します。コードが実行されていない場合、料金は発生しません。
- 「Amazon SNS」 — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーやメールアドレスを含むパブリッシャーとクライアント間のメッセージ送信を調整および管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、次の 2 つのファイルを含む添付ファイルが含まれます。

- `ElasticCache-EncryptionAtRest.zip` は、セキュリティコントロール (Lambda コード) を含む圧縮ファイルです。
- `elasticache_encryption_at_rest.yml` は、セキュリティコントロールをデプロイする CloudFormation テンプレートです。

これらのファイルの使用方法については、「エピック」セクションを参照してください。

エピック

セキュリティコントロールをデプロイ

タスク	説明	必要なスキル
S3 バケットにコードをアップロードします。	新しい S3 バケットを作成するか、既存の S3 バケットを使用して添付 `ElastiCache-EncryptionAtRest.zip` ファイル (Lambda コード) をアップロードします。このバケットは、評価するリソースと同じ AWS リージョンにあることが必要です。	クラウドアーキテクト
CloudFormation テンプレートをデプロイする。	S3 バケットと同じ AWS リージョンで Cloudformation コンソールを開き、添付ファイルで提供されている `elasticache_encryption_at_rest.yml` ファイルをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を入力します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。この S3 バケットには Lambda コードの.zip ファイルが含まれており、CloudFormation のテンプレートと評価するリソースと同じ AWS ゾーンにある必要があります。	クラウドアーキテクト
S3 キーを入力します。	S3 バケット内の Lambda コードの.zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例：`ElasticCache-EncryptionAtRest.zip` または `controls/ElasticCache-EncryptionAtRest.zip`)。	クラウドアーキテクト
メールアドレスを入力します。	違反の通知を受信するメールアドレスを入力します。	クラウドアーキテクト
ログレベルを指定します。	ログ記録レベルと詳細レベルを定義します。`Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定し、デバッグのみに使用されます。`Error` それでもアプリケーションの実行を継続できるエラーイベントを指定します。`Warning` 潜在的に危険有害な状況を示します。	クラウドアーキテクト

サブスクリプションを確認

タスク	説明	必要なスキル
メールサブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定した E メールアドレスにサブスクリプション E メールメッセージが送信されます。通知を受信するには、この E メールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「AWS CloudFormation コンソールでのスタックの作成」(AWS CloudFormation ドキュメント)
- 「ElastiCache for Redis での保管時の暗号化 (Amazon ElastiCache」ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

AWS Config を使用して EC2 インスタンスのキーペアを監視する

作成者: Wassim Benhallam (AWS)、Sergio Bilbao Lopez (AWS)、Vikrant Telkar (AWS)

概要

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをAmazon Web Services (AWS) クラウドで起動する場合のベストプラクティスは、インスタンスに接続する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成または使用することです。キーペアは、インスタンスに保存されているパブリックキーとユーザーに提供されるプライベートキーで構成され、Secure Shell (SSH) を介してインスタンスに安全にアクセスでき、パスワードの使用を回避できます。ただし、ユーザーがkey pair をアタッチせずに誤ってインスタンスを起動することがあります。キーペアはインスタンスの起動時にのみ割り当てることができるため、キーペアなしで起動したインスタンスをすばやく特定し、非対応としてフラグを立てることが重要です。これは、インスタンスへのアクセスにキーペアの使用が義務付けられているアカウントや環境で作業する場合に特に便利です。

このパターンでは、AWS Config でカスタムルールを作成して EC2 インスタンスのキーペアを監視する方法を説明します。インスタンスが非準拠として識別されると、Amazon EventBridge イベントを通じて開始される Amazon Simple Notification Service（Amazon SNS）通知を用いて送信されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- モニタリングしたい AWS リージョンで AWS Config が有効になっており、すべての AWS リソースを記録するように設定されている

機能制限

- このソリューションはリージョン固有です。これらのすべてのリソースを、同じ AWS リージョン内に作成する必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Config
- Amazon EventBridge
- AWS Lambda
- Amazon SNS

ターゲットアーキテクチャ

1. AWS Config がルールを開始します。
2. このルールは Lambda 関数を呼び出して EC2 インスタンスのコンプライアンスを評価します。
3. Lambda 関数は、更新されたコンプライアンス状態を AWS Config に送信します。
4. AWS Config はイベントブリッジにイベントを送信します。
5. EventBridge は SNS トピックにコンプライアンス変更通知を公開します。
6. Amazon SNS はアラートを電子メールで送信します。

自動化とスケール

このソリューションでは、リージョン内の任意の数の EC2 インスタンスを監視できます。

ツール

ツール

- AWS Config – AWS Config を使用すると、AWS リソースの設定を評価、監査、審査できます。AWS Config では、AWS リソースの設定の評価、監査、評価を行うことができます。は、AWS リソース設定を継続的に監視および記録し、必要な設定に対して記録された設定の評価を自動化することができます。
- Amazon EventBridge— Amazon EventBridgeは、アプリケーションをさまざまなソースからのデータに接続するために使用できるサーバーレスのイベントバスサービスです。
- AWS Lambda は、サーバーのプロビジョニングや管理、ワークロードに対応したクラスタースケーリングロジックの作成、イベント統合の維持、あるいはランタイムの管理などを行うことなくコードの実行をサポートする、サーバーレスコンピューティングサービスです。
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS)は、アプリケーション間 (A2A) およびアプリケーションと人間の間 (A2P) の通信の両方に対応するフルマネージド型のメッセージングサービスです。

コード

Lambda 関数のコードが添付されています。

エピック

Amazon EC2 コンプライアンスを評価する Lambda 関数を作成する

タスク	説明	必要なスキル
Lamda の AWS Identity and Access Management (IAM) ロールを作成します。	AWS マネジメントコンソールで「IAM」を選択し、ロールを作成します。このとき、Lambda を信頼できるエンティティとして使用し、`AmazonEventBridgeFullAccess` と `AWSConfigRulesExecutionRole` のアクセス権限を追加します。詳細については、AWS ドキュメントを参照してください。	DevOps
Lambda 関数を作成してデプロイします。	1. Lambda コンソールで、「Python 3.6」をランタイムとして、以前に作成した IAM ロールを使用して、「ゼロから作る」を使用して関数を作成します。Amazon リソースネーム (ARN) を記録しておきます。 2. 「コード」タブで、`lambda_function.py` を選択し、このパターンに添付されているコードを貼り付けます。	DevOps

タスク	説明	必要なスキル
	3. 変更を保存するには [デプロイ] を選択します。	

カスタム AWS Config ルールを作成する

タスク	説明	必要なスキル
カスタム AWS Config ルールを追加します。	AWS Config コンソールで、次の設定を使用してカスタムルールを追加します。 • ARN — 以前に作成されたLambda 関数の ARN です。 • トリガータイプ： 設定変更 • 変更範囲 — リソース • リソースタイプ — Amazon EC2 インスタンス 詳細については、AWS ドキュメントを参照してください。	DevOps

コンプライアンス変更イベントが検出されたときの E メール通知を設定します。

タスク	説明	必要なスキル
SNS トピックとサブスクリプションを作成します。	Amazon SNS コンソールで、タイプとして「スタンダード」を使用してトピックを作成し、次にプロトコルとして「E メール」を使用してサブスクリプションを作成します。	DevOps

タスク	説明	必要なスキル
	確認 E メールを受信したら [サブスクリプションを確認] リンクを選択します。 詳細については、AWS ドキュメントを参照してください。	
Amazon SNS 通知を開始する EventBridge ルールを作成します。	EventBridge コンソールで、以下の設定を使用してルールを作成します。 • サービス名 — AWS Config • イベントタイプ — Config ルールコンプライアンス変更 • メッセージタイプ — 特定のメッセージタイプ、コンプライアンス/変更通知 • 特定のルール名 — 以前に作成した AWS Config ルールの名前 • ターゲット — SNS トピック、以前に作成したトピック 詳細については、AWS ドキュメントを参照してください。	DevOps

ルールと通知を確認する

タスク	説明	必要なスキル
EC2 インスタンスを作成します。	任意のタイプの 2 つの EC2 インスタンスを作成してキーペ	DevOps

タスク	説明	必要なスキル
	アをアタッチし、キーペアなしで 1 つの EC2 インスタンスを作成します。	
ルールを確認する。	1. AWS Config コンソールの「ルール」ページで、ルールを選択します。 2. 準拠している EC2 インスタンスと準拠していない EC2 インスタンスを表示するには、「スコープ内のリソース」を「すべて」に変更します。2 つのインスタンスが準拠していると表示され、1 つのインスタンスが非準拠としてリストされていることを確認します。 3. EC2 インスタンスのコンプライアンス状態に関する Amazon SNS E メール通知を受信するまでお待ちください。	DevOps

関連リソース

- AWS のサービスにアクセス許可を委任するロールの作成
- AWS Config でのカスタムルールの作成
- Amazon SNS トピックを作成する
- Amazon SNS トピックへサブスクライブする
- Amazon EventBridge でルールを作成する

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

IAM ルートユーザーのアクティビティを監視する

作成者：モステファ・ブルギ (AWS)

概要

各Amazon Web Services (AWS) アカウントにはルートユーザーが付いています。AWS Identity and Access Management (IAM) の「セキュリティベストプラクティス」として、ルートユーザーのみが実行可能なタスクに使用することをお勧めします。これらのタスクの完全なリストについては、「全般のリファレンス」の「ルートユーザー認証情報が必要なタスク」を参照してください。ルートユーザーはすべての AWS リソースと請求情報に完全にアクセスできるので、このアカウントを使用せず、ルートユーザーの認証情報が侵害されている可能性があるアクティビティをモニタリングすることをお勧めします。

このパターンを使用して、IAM ルートユーザーをモニタリングする「イベント駆動型アーキテクチャ」を設定します。このパターンは、複数の AWS アカウント (スポークアカウント) をモニタリングし、管理とレポートを 1 つのアカウント (ハブアカウント) に集中させるハブアンドスポークソリューションを確立します。

IAM ルートユーザー認証情報が使用されると、Amazon CloudWatch と AWS CloudTrail は、それぞれログと証跡にアクティビティを記録します。スポークアカウントでは、Amazon EventBridge ルールによってハブアカウントの中央「イベントバス」にイベントが送信されます。ハブアカウントでは、EventBridge ルールがイベントを AWS Lambda 関数に送信します。この関数は、ルートユーザーアクティビティを通知する Amazon Simple Notification Service (Amazon SNS) トピックを使用します。

このパターンでは、AWS CloudFormation テンプレートにより、スポークアカウントにモニタリングサービスとイベント処理サービスをデプロイします。HashiCorp Terraform テンプレートにより、ハブアカウントにイベント管理サービスと通知サービスをデプロイします。

前提条件と制限

前提条件

1. AWS 環境に AWS リソースをデプロイする権限。
2. CloudFormation スタックセットをデプロイするための権限。詳細については、「スタックセットオペレーションの前提条件」(CloudFormation ドキュメント) を参照してください。
3. Terraform がインストール済みで、すぐに使用できます。詳細については、「ドキュメント」の「使用の開始」を参照してください。

4. 各スポークアカウントの既存の証跡。詳細については、「AWS CloudTrail で開始する」(CloudTrail ドキュメント) を参照してください。
5. 証跡は CloudWatch Logs にイベントを送信するように設定されています。詳細については、「CloudWatch Logs へのイベント送信」(CloudTrail ドキュメント) を参照してください。
6. ハブアンドスポークアカウントは AWS Organizations によって管理される必要があります。

アーキテクチャ

次の図は、実装の構成要素を示しています。

1. IAM ルートユーザー認証情報が使用されると、CloudWatch と CloudTrail は、それぞれログと証跡にアクティビティを記録します。
2. スポークアカウントでは、EventBridge ルールがハブアカウントの中央「イベントバス」にイベントを送信します。
3. ハブアカウントでは、EventBridge ルールがイベントを Lambda 関数に送信します。
4. Lambda 関数は、ルートユーザーのアクティビティを通知する Amazon SNS トピックを使用します。

ツール

AWS サービス

- AWS CloudFormation を使用すると、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクル全体にわたってリソースを管理できます。
- 「AWS CloudTrail」は、AWS アカウントのガバナンス、コンプライアンス、運用のリスクの監査をサポートします。
- 「Amazon CloudWatch Logs」は、すべてのシステム、アプリケーション、AWS からのログを一元管理できるため、ログを監視して安全にアーカイブできます。
- 「Amazon EventBridge」は、アプリケーションをさまざまなソースのリアルタイムデータに接続できるようにするサーバーレスイベントバスサービスです。たとえば、AWS Lambda 関数、API 宛先を使用する HTTP 呼び出しエンドポイント、または他の AWS アカウントのイベントバスなどです。

- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- 「Amazon Simple Notification Service (Amazon SNS)」は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージの交換を調整および管理するのに役立ちます。

ツールやサービス

- 「Terraform」は、設定ファイル形式のコードを使用してクラウドインフラストラクチャとリソースをプロビジョニングし管理するための CLI アプリケーションです。

コードリポジトリ

このパターンのソースコードとテンプレートは、「GitHub リポジトリ」で利用できます。このパターンには次の 2 つのテンプレートがあります。

- ハブアカウントにデプロイするリソースを含む Terraform テンプレート
- スポークアカウントにスタックセットインスタンスとしてデプロイする CloudFormation テンプレート

リポジトリの全体構造は以下のようになっています。

```
.
|__README.md
|__spoke-stackset.yaml
|__hub.tf
|__root-activity-monitor-module
    |__main.tf  # contains Terraform code to deploy resources in the Hub account
    |__iam      # contains IAM policies JSON files
        |__ lambda-assume-policy.json          # contains trust policy of the IAM role
used by the Lambda function
        |__ lambda-policy.json                 # contains the IAM policy attached to
the IAM role used by the Lambda function
    |__outputs  # contains Lambda function zip code
```

エピック セクションでは、テンプレートをデプロイする手順の詳細について説明します。

エピック

ハブアカウントにリソースをデプロイ

タスク	説明	必要なスキル
サンプルコードリポジトリを複製します。	1. 「AWS IAM Root User Activity Monitor」リポジトリを開きます。 2. ファイルリストの上にあるコードタブで Code を選択し、HTTPS URL をコピーします。 3. コマンドラインインターフェースで、作業ディレクトリをサンプルファイルを保存する場所に変更します。 4. 次のコマンドを入力します。 `git clone <repoURL>`	AWS 全般
Terraform テンプレートを更新します。	1. 組織 ID を取得します。手順については、「管理アカウントからの組織の詳細の表示」 (AWS Organizations ドキュメント) を参照してください。 2. クローンしたリポジトリで、hub.tf を開きます。 3. お使いの環境に対して以下のように適切な値で更新します。 • `OrganizationId` — 組織 ID を追加します。	AWS 全般

タスク	説明	必要なスキル
	• SNSTopicName — Amazon SNS トピックの名前を追加します。 • SNSSubscriptions — Amazon SNS 通知の送信先となる E メールを追加します。 • Region — リソースをデプロイする AWS リージョンコードを追加します。例えば、eu-west-1 と指定します。 • Tags — タグを追加します。タグの詳細については、「Tagging resources」(全般のリファレンス) を参照してください。 4. hub.tf ファイルを保存して閉じます。	

タスク	説明	必要なスキル

リソースを AWS ハブアカウントにデプロイします。	1. Terraform のコマンドラインインターフェイスで、クローンされたリポジトリのルートフォルダに移動し、次のコマンドを入力します。
	`terraform init && terraform plan`
	2. 出力を確認し、説明されたリソースを作成することを確認します。
	3. 次のコマンドを入力します。
	`terraform apply`
	4. プロンプトが表示されたら、yes を入力して展開を確認します。
		AWS 全般

スポークアカウントにリソースをデプロイ

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	1. AWS マネジメントコンソールにサインインし、「CloudFormation コンソール」を開きます。 2. ナビゲーションペインから [StackSets] を選択します。	AWS 全般

タスク	説明	必要なスキル
	3. [StackSets] ページの上部で [Create StackSet (StackSet の作成)] を選択します。 4. 権限で、サービス管理権限を選択します。 CloudFormation は、AWS Organizations が管理するターゲットアカウントへのデプロイに必要なアクセス許可を自動的に設定します。 5. [前提条件 – テンプレートの準備] で、[テンプレートの準備完了] を選択します。 6. [テンプレートの指定] で、[テンプレートファイルのアップロード] を選択します。 7. ファイルを選択を選択し、複製されたリポジトリで `spoke-stackset.yaml` を選択します。 8. [Next (次へ)] を選択します。 9. スタックの詳細の指定ページで、スタックの名前を入力します。 10パラメータで、ハブアカウントのアカウント ID を入力し、次へを選択します。 11.StackSet オプションの設定ページのタグで、タグを追加します。	

タスク	説明	必要なスキル
	12.実行設定で非アクティブを選択して、次へを選択します。 13.デプロイオプションの設定ページで、スタックセットをデプロイする組織単位とリージョンを指定し、次へを選択します。 14.レビューページで、[AWS CloudFormationがIAMリソースを作成する可能性があることを認めます] を選択し、[送信] を選択します。CloudFormation がスタックセットのデプロイを開始します。 詳細については、「スタックセットの作成」(CloudFormation ドキュメント) を参照してください。	

(オプション) 通知をテスト

タスク	説明	必要なスキル
ルートユーザーの認証情報を使用します。	1. ルートユーザー認証情報を使用して、スポークアカウントまたはハブアカウントにサインインします。 2. 指定したメールアカウントが Amazon SNS 通知を受信することを確認します。	AWS 全般

関連リソース

- 「セキュリティのベストプラクティス」(IAM ドキュメント)
- 「StackSets での作業」(CloudFormation ドキュメント)
- 「はじめに」(Terraform ドキュメント)

追加情報

「Amazon GuardDuty」は、継続的なセキュリティ監視サービスで、ログを分析処理し、AWS 環境における予期せぬ不正な可能性のあるアクティビティを特定します。このソリューションに代わる方法として、GuardDuty を有効にすると、root ユーザーの認証情報が使用された場合に警告が表示されます。GuardDuty の検出結果は `Policy:IAMUser/RootCredentialUsage` であり、重要度はデフォルトで低です。詳細については、「Amazon GuardDutyの検出結果の管理」を参照してください。

IAM ユーザーが作成されたときに通知を送信

マンシ・スラトワラ (AWS) とセルゲイ・シェフチェンコ (AWS) によって作成されました

概要

Amazon Web Services (AWS) では、このパターンを使用して AWS CloudFormation テンプレートをデプロイし、AWS Identity and Access Management (IAM) ユーザーが作成されたときに自動的に通知を受け取ることができます。

IAM を使用すると、AWS サービスとリソースへのアクセスを安全に管理できます。AWS のユーザーとグループを作成および管理し、アクセス権を使用して ユーザーとグループの AWS リソースへのアクセスを許可および拒否できます。

CloudFormation テンプレートは Amazon CloudWatch Events と AWS Lambda 関数を作成します。このイベントでは、AWS CloudTrail を使用して、AWS アカウントで作成されているすべての IAM ユーザーを監視します。ユーザーが作成されると、CloudWatch Events イベントによって Lambda 関数が開始され、新しいユーザー作成イベントを通知する Amazon Simple Notification Service (Amazon SNS) 通知が送信されます。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS CloudTrail トレイルが作成され、デプロイされました

機能制限

- AWS CloudFormation テンプレートは、`CreateUser` 専用にデプロイする必要があります。

アーキテクチャ

ターゲットテクノロジースタック

- IAM
- AWS CloudTrail
- Amazon CloudWatch Events

- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

ターゲットアーキテクチャ

自動化とスケール

AWS CloudFormation テンプレートは、さまざまな AWS リージョンとアカウントに複数回使用できます。各リージョンまたはアカウントで 1 回実行するだけで済みます。複数のアカウントへのデプロイを自動化するには、「AWS CloudFormation StackSets」 を使用してください。CloudFormation テンプレートは、各アカウントに必要なすべてのリソースをデプロイできます。

ツール

ツール

- 「IAM」 – AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスのセキュアな制御に役立つ Web サービスです。IAM を使用して、誰を認証 (サインイン) し、誰にリソースの使用を認可する (アクセス許可を付与する) かを制御します。
- 「AWS CloudFormation」 – AWS CloudFormation は、Amazon ウェブサービスリソースのモデル化とセットアップを支援して、これらのリソースの管理に費やす時間を減らし、AWS で実行されるアプリケーションに集中する時間を増やすことができます。必要なすべての AWS リソースを説明するテンプレートを作成して、CloudFormation がお客さまに代わって、これらのリソースのプロビジョニングや設定を処理します。
- 「AWS CloudTrail」 – AWS CloudTrail は、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査を管理します。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。イベントには、AWS マネジメントコンソール、AWS コマンドラインインターフェイス、AWS SDK や API で実行されるアクションが含まれます。
- 「Amazon CloudWatch Events」 – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケールします。

- 「Amazon S3」— Amazon Simple Storage Service (Amazon S3)は、インターネット用のストレージです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。
- 「Amazon SNS」 — Amazon Simple Notification Service (Amazon SNS) は、Lambda、HTTP、E メール、モバイルプッシュ通知 (SMS) を使用してメッセージを配信するマネージドサービスです。

コード

プロジェクトの .zip ファイルは添付ファイルとして入手できます。

エピック

Lambda スクリプト用の S3 バケットを作成する

タスク	説明	必要なスキル
S3 バケットを定義します。	Amazon S3 コンソールを開き、S3 バケットを選択、または作成します。この S3 バケットは Lambda コードの.zip ファイルをホストします。S3 バケット名の先頭にスラッシュを含めることはできません。	クラウドアーキテクト

S3 バケットに Lambda コードをアップロードします

タスク	説明	必要なスキル
Lambda コードをアップロードします。	「添付ファイル」セクションにある Lambda コードの.zip ファイルを S3 バケットにアップロードします。	クラウドアーキテクト

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
CloudFormation のテンプレートをデプロイします。	CloudFormation コンソールで、このパターンへの添付ファイルとして提供されている CloudFormation `createIAMuser.yaml` テンプレートをデプロイします。次のエピックでは、テンプレートパラメータの値を指定します。	クラウドアーキテクト

CloudFormation テンプレートのパラメーターを入力します。

タスク	説明	必要なスキル
S3 バケット名を指定します。	最初のエピックで作成または選択した S3 バケットの名前を入力します。	クラウドアーキテクト
S3 キーを指定します。	S3 バケットの Lambda コードの .zip ファイルの場所を、先頭にスラッシュを付けずに指定します (例: `<directory>/<file-name>.zip`)。	クラウドアーキテクト
Eメールアドレスを入力します。	Amazon SNS 通知を受信するための有効な Eメールアドレスを指定します。	クラウドアーキテクト
ログ記録のレベルを定義します。	Lambda 関数のロギングレベルと頻度を定義します。 `Info` アプリケーションの進行状況に関する詳細な情報メッセージを指定します。 `Error` そ	クラウドアーキテクト

タスク	説明	必要なスキル
	れでもアプリケーションの実行を継続できるエラーイベントを指定します。 `Warning` 潜在的に有害な状況を示します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	テンプレートが正常にデプロイされると、指定されたメールアドレスに購読メールメッセージが送信されます。通知を受信するには、このメールのサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- 「証跡の作成」
- 「S3 バケットの作成」
- 「ファイルを S3 バケットにアップロード」
- 「CloudFormation テンプレートをデプロイする」
- 「IAM ユーザーの作成」
- 「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成」

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

サービスコントロールポリシーを使用して、アカウントレベルでのインターネットアクセスを防止する

作成者: Sergiy Shevchenko (AWS)、Sean O'Sullivan (AWS)、Victor Mazeo Whitaker (AWS)

概要

組織は、多くの場合、プライベートなままにしておくべきアカウントリソースのインターネットアクセスを制限したいと考えています。これらのアカウントでは、Virtual Private Cloud (VPCs) のリソースは、いかなる方法でもインターネットにアクセスしないでください。多くの組織は、一元化された検査アーキテクチャを選択します。一元的な検査アーキテクチャの東西 (VPC-to-VPCトラフィック の場合、スポークアカウントとそのリソースがインターネットにアクセスできないことを確認する必要があります。南北 (インターネット出力とオンプレミス) トラフィックの場合、検査 VPC を介してのみインターネットアクセスを許可する必要があります。

このパターンでは、サービスコントロールポリシー (SCP) を使用してインターネットアクセスを防止します。この SCP は、アカウントまたは組織単位 (OU) レベルで適用できます。SCP は、以下を防ぐことでインターネット接続を制限します。

- VPC への直接インターネットアクセスを許可する IPv4 または IPv6 インターネットゲートウェイの作成またはアタッチ
- 別の VPC を介した間接的なインターネットアクセスを許可する可能性のある VPC ピアリング接続の作成または承諾
- VPC リソースへの直接インターネットアクセスを許可する可能性のあるAWS Global Accelerator設定の作成または更新

前提条件と制限

前提条件

- 組織として AWS アカウント 管理される 1 つ以上の AWS Organizations。
- すべての機能は で有効になっています AWS Organizations。
- SCPsは組織内で有効になっています。
- 以下のアクセス許可：
  - 組織の管理アカウントにアクセスします。
  - SCPs。最小アクセス許可の詳細については、「SCP の作成」を参照してください。

- SCP をターゲットアカウントまたは組織単位 (OUs。最小アクセス許可の詳細については、「サービスコントロールポリシーのアタッチとデタッチ」を参照してください。

制約事項

- SCP は、管理アカウントのユーザーやロールには影響を与えません。SCP は、組織内のメンバーアカウントにのみ影響を与えます。
- SCPs組織の一部であるアカウントによって管理される AWS Identity and Access Management (IAM) ユーザーとロールにのみ影響します。詳細については、「許可に対する SCP の影響」を参照してください。

ツール

AWS サービス

- AWS Organizations は、作成して一元管理する AWS アカウント 組織に複数の を統合するのに役立つアカウント管理サービスです。このパターンでは、 でサービスコントロールポリシー (SCPsを使用します AWS Organizations。
- Amazon Virtual Private Cloud (Amazon VPC) は、定義した仮想ネットワークに AWS リソースを起動するのに役立ちます。この仮想ネットワークは、ユーザー自身のデータセンターで運用されていた従来のネットワークと似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるという利点があります。

ベストプラクティス

組織でこの SCP を確立したら、インターネットアクセスに影響を与える可能性のある新機能 AWS のサービス に対処するために、この SCP を頻繁に更新してください。

エピック

SCP を作成してアタッチする

タスク	説明	必要なスキル
SCP を作成します。	1. AWS Organizations コンソール にサインインします	AWS 管理者

タスク	説明	必要なスキル
	。組織の管理アカウントにサインインする必要があります。 2. 左側のペインで、ポリシーを選択します。 3. ポリシーページで、サービスコントロールポリシーを選択します。 4. サービスコントロールポリシーページで、[Create policy] (ポリシーの作成) を選択します。 5. 新しいサービスコントロールポリシーの作成ページで、ポリシー名とオプションのポリシーの説明を入力します。 6. (オプション) ポリシーにAWS タグを追加します。 7. JSON エディタで、プレースホルダーポリシーを削除します。 8. 以下の ポリシーを JSON エディタに貼り付けます。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "ec2:AttachInternetGateway",`	

タスク	説明	必要なスキル

	<pre>"ec2:Crea
	teInternetGateway",
	
	"ec2:Crea
	teVpcPeeringConnec
	tion",
	"ec2:Acce
	ptVpcPeeringConnec
	tion",
	"ec2:Crea
	teEgressOnlyIntern
	etGateway"
	],
	"Resource":
	"*",
	"Effect": "Deny"
	},
	{
	"Action": [
	"globalac
	celerator:Create*",
	"globalac
	celerator:Update*"
	],
	"Resource":
	"*",
	"Effect": "Deny"
	}
	]
	}</pre> 9. [Create policy] を選択します。	

タスク	説明	必要なスキル
SCP をアタッチします。	1. サービスコントロールポリシーページで、作成したポリシーを選択します。 2. [Targets] (ターゲット) タブで [Attach] (アタッチ) を選択します。 3. ポリシーをアタッチする OU またはアカウントを選択します。必要な OUs またはアカウントを見つけるには、OU を展開する必要がある場合があります。 4. Attach policy] (ポリシーのアタッチ) を選択してください。	AWS 管理者

関連リソース

- AWS Organizations ドキュメント
- サービスコントロールポリシー (SCP)
- AWS Gateway Load Balancer と を使用した一元的な検査アーキテクチャ AWS Transit Gateway (AWS ブログ記事)

を使用して IP アドレスまたは位置情報に基づいてアクセスを制限する AWS WAF

作成者: Louis Hourcade (AWS)

概要

AWS WAF は、可用性に影響を与えたり、セキュリティを侵害したり、過剰なリソースを消費したりする可能性のある一般的なウェブエクスプロイトやボットからウェブアプリケーションや APIs を保護するのに役立つウェブアプリケーションファイアウォールです。のウェブアクセスコントロールリスト (ウェブ ACLs) AWS WAF を使用すると、トラフィックがアプリケーションに到達する方法を制御できます。ウェブ ACL では、正当なトラフィックを許可し、ボットトラフィックを制御し、一般的な攻撃パターンをブロックするように設計されたルールまたはルールグループを追加します。詳細については、「 AWS WAF の仕組み」を参照してください。

次のタイプのルールを AWS WAF ウェブ ACLs に関連付けることができます。

- マネージドルールグループ – AWS マネージドルールチームと AWS Marketplace 販売者は、事前設定されたルールセットを提供します。一部のマネージドルールグループは、特定のタイプのウェブアプリケーションを保護するように設計されています。また、既知の脅威や一般的な脆弱性に対する広範な保護を提供するものもあります。
- カスタムルールとカスタムルールグループ – ウェブアプリケーションと APIs へのアクセスをカスタマイズするルールとルールグループを作成することもできます。例えば、特定の IP アドレスのリストまたは国のリストに基づいてトラフィックを制限できます。

このパターンと関連するコードリポジトリを使用することで、 を使用してカスタムルールで AWS WAF ウェブ ACLs AWS Cloud Development Kit (AWS CDK)をデプロイできます。これらのルールは、エンドユーザーの IP アドレスまたは位置情報に基づいて、ウェブアプリケーションリソースへのアクセスを制限します。オプションで、複数のマネージドルールグループをアタッチすることもできます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- AWS WAF リソースをデプロイするためのアクセス許可

- AWS CDK、 アカウントにインストールおよび設定されている
- 「インストール済み」Git

制約事項

- このパターンは、 AWS リージョン AWS WAF が利用可能な でのみ使用できます。リージョンの可用性については、AWS のサービス 「リージョン別」を参照してください。

ツール

AWS のサービス

- AWS Cloud Development Kit (AWS CDK) は、コードで AWS クラウド インフラストラクチャを定義およびプロビジョニングするのに役立つソフトウェア開発フレームワークです。
- AWS WAF は、保護されたウェブアプリケーションリソースに転送される HTTP および HTTPS リクエストをモニタリングするのに役立つウェブアプリケーションファイアウォールです。

コードリポジトリ

このパターンのコードは、GitHub の IP およびリポジトリを使用した位置情報制限 AWS WAFで使用できます。このコードは 2 つの AWS WAF ウェブ ACLs。1 つ目は、Amazon API Gateway リソースを対象としたリージョンウェブ ACL です。2 つ目は、Amazon CloudFront リソースのグローバルウェブ ACL です。どちらのウェブ ACLs、次のカスタムルールが含まれています。

- `IPMatch` は、許可されていない IP アドレスからのリクエストをブロックします。
- `GeoMatch` は、許可されていない国からのリクエストをブロックします。

デプロイ中に、オプションで以下のすべてのマネージドルールグループをウェブ ACLs にアタッチできます。

- コアルールセット (CRS) – このルールグループには、ウェブアプリケーションに一般的に適用されるルールが含まれています。これは、OWASP Top 10 など、OWASP の出版物に記載されている高リスクの脆弱性や一般的に発生する脆弱性など、幅広い脆弱性の悪用を防ぐのに役立ちます。
- 管理者保護 – このルールグループには、公開されている管理ページへの外部アクセスをブロックするのに役立つルールが含まれています。

- 既知の不正な入力 – このルールグループは、無効であることがわかっており、脆弱性の悪用または検出に関連付けられているリクエストパターンをブロックするのに役立ちます。
- Amazon IP の評価リスト – このルールグループには、Amazon 内部脅威インテリジェンスに基づくルールが含まれています。ボットやその他の脅威に通常関連付けられている IP アドレスをブロックするのに役立ちます。
- Linux オペレーティングシステムマネージドルールグループ – このルールグループは、Linux 固有のローカルファイルインクルージョン (LFI) 攻撃など、Linux の脆弱性の悪用に関連するリクエストパターンをブロックするのに役立ちます。
- SQL データベースマネージドルールグループ – このルールグループは、SQL インジェクション攻撃などの SQL データベースの悪用に関連するリクエストパターンをブロックするのに役立ちます。

エピック

AWS WAF ウェブ ACLs を設定する

タスク	説明	必要なスキル
リポジトリをクローン作成します。	次のコマンドを入力して、ローカルワークステーションに IP と位置情報の制限をリポジトリで AWS WAFクローンします。 `git clone https://github.com/aws-samples/ip-and-geolocation-restriction-with-waf-cdk.git`	Git
ルールを設定します。	1. クローンされたリポジトリで、 app.py://https://https://https://https://https://https://https://https://https://https	一般的な AWS、Python

タスク	説明	必要なスキル

	2. 次の変数の値を変更して、ルールをカスタマイズします。
	
	`aws_acccount = "AWS_ACCOUNT"`
	`region = "AWS_REGION"`
	`ip_list = ["CIDR_RANGE_1", "CIDR_RANGE_2"]`
	`geo_list = ["COUNTRY_CODE_1", "COUNTRY_CODE_2"]`
	`aws_managed_rules = True`
	
	コードの説明は以下のとおりです。
	• `aws_account` はターゲットの ID です AWS アカウント。
	• `region` は、API Gateway リソース AWS リージョン のウェブ ACL のターゲットです。
	
	Note
	CloudFront リソースのウェブ ACL はグローバルであり、 `us-east-1` リージョンにデプロイされます。	

タスク	説明	必要なスキル
	• ip_list は、アクセスが許可されている CIDR 範囲のリストです。 • geo_list は、アクセスが許可されている国のリストです。有効な値については、AWS WAF ドキュメントを参照してください。 • aws_managed_rules は、マネージドルールグループがウェブ ACL に追加されるかどうかを制御します。この値が の場合True、追加されます。この値が の場合False、それらは除外されます。 3. app.py ファイルを保存して閉じます。	

コードをブートストラップしてデプロイする

タスク	説明	必要なスキル
AWS 環境をブートストラップします。	まだ完了していない場合は、AWS CDK アプリケーションをデプロイする前に AWS 環境をブートストラップする必要があります。 1. AWS CDK CLI で、次のコマンドを入力してus-east-1 リージョンをブートストラップします。	AWS 全般

タスク	説明	必要なスキル

	`cdk bootstrap aws://<account-id>/us-east-1`
	2. API Gateway のウェブ ACL を 以外のリージョンにデプロイする場合はus-east-1 、次のコマンドを入力してターゲットリージョンをブートストラップします。
	`cdk bootstrap aws://<account-id>/<region>`	

AWS CDK アプリケーションをデプロイします。	1. 次のコマンドを入力して AWS CDK アプリケーションをデプロイします。 `cdk deploy --all` 2. AWS CloudFormation スタックのデプロイが完了するまで待ちます。	AWS 全般

デプロイを検証する

タスク	説明	必要なスキル
ウェブ ACLsデプロイされたことを確認します。	1. にサインインし AWS Management Console、AWS WAF コンソールを開きます。	AWS 全般

タスク	説明	必要なスキル
	2. ナビゲーションペインで [Web ACLs] (ウェブ ACL) を選択します。 3. のリストで AWS リージョン、グローバル (CloudFront) を選択します。 4. 新しい CloudFront ウェブ ACL がデプロイされたことを確認し、定義した IP アドレスと位置情報ルールがあることを確認します。このウェブ ACL のデフォルト名は ですWebACLCloudfront-<ID> 。 5. のリストで AWS リージョン、スタックをデプロイしたリージョンを選択します。 6. API Gateway リソースの新しいウェブ ACL がデプロイされたことを確認します。定義した IP アドレスと位置情報ルールがあることを確認します。このウェブ ACL のデフォルト名は ですWebACLApiGW-<ID> 。	

タスク	説明	必要なスキル
(オプション) ウェブ ACLs をリソースに関連付けます。	Application Load Balancer、API Gateway、CloudFront ディストリビューションなどの AWS リソースに AWS WAF ウェブ ACLs を関連付けます。手順については、「ウェブ ACL と リソースの関連付けまたは関連付け解除AWS」を参照してください。https://docs.aws.amazon.com/waf/latest/developerguide/web-acl-associating-aws-resource.html例については、 AWS CDK ドキュメントの「クラス CfnWebACLAssociation (コンストラクト) 」を参照してください。	AWS 全般

リソースをクリーンアップする

タスク	説明	必要なスキル
スタックを削除します。	1. ウェブ ACL とリソースの関連付けを解除します AWS 。手順については、AWS WAF ドキュメントを参照してください。 2. AWS CDK CLI で、次のコマンドを入力して AWS CDK アプリケーションを削除します。	AWS 全般

タスク	説明	必要なスキル
	`cdk destroy --all`	

関連リソース

- API リファレンス (AWS CDK ドキュメント)
- aws-cdk-lib.aws_wafv2 モジュール (AWS CDK ドキュメント)
- ウェブ ACLs の使用 (AWS WAF ドキュメント)
- 独自のルールグループの管理 (AWS WAF ドキュメント)
- ルール (AWS WAF ドキュメント)

git-secrets を使用して Git リポジトリに機密情報やセキュリティ上の問題がないかスキャンする

作成者: Saurabh Singh (AWS)

概要

このパターンでは、 AWS Labs のオープンソースの git-secrets ツールを使用して Git ソースリポジトリをスキャンし、ユーザーパスワードや AWS アクセスキーなどの機密情報が含まれる可能性のあるコードや、その他のセキュリティ上の問題があるコードを検索する方法について説明します。

`git-secrets` は、コミット、コミットメッセージ、マージをスキャンして、シークレットなどの機密情報が Git リポジトリに追加されないようにします。例えば、コミット、コミットメッセージ、またはマージ履歴のコミットが、設定済みで禁止されている正規表現パターンのいずれかに一致した場合、そのコミットは拒否されます。

前提条件と制限

前提条件

- アクティブな AWS アカウント
- セキュリティスキャンが必要な Git リポジトリ
- Git クライアント (バージョン 2.37.1 以降) がインストールされていること

アーキテクチャ

ターゲットアーキテクチャ

- Git
- `git-secrets`

ツール

- git-secrets は、機密情報を Git リポジトリにコミットすることを防ぐツールです。
- Git はオープンソースの分散型バージョン管理システムです。

ベストプラクティス

- 必ず、すべてのリビジョンを含めて Git リポジトリをスキャンしてください。

```
git secrets --scan-history
```

エピック

EC2 インスタンスに接続する

タスク	説明	必要なスキル
SSH を使用して EC2 インスタンスに接続する。	SSH とキーペアファイルを使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに接続します。 ローカルマシンでリポジトリをスキャンする場合は、このステップをスキップできます。	AWS 全般

Git をインストールする

タスク	説明	必要なスキル
Git をインストールする。	次のコマンドを使用して Git をインストールします。 `yum install git -y` ローカルマシンを使用している場合は、特定の OS バージョンの Git クライアントをインストールできます。詳	AWS 全般

タスク	説明	必要なスキル
	細については、Git Web サイトを参照してください。	

ソースリポジトリをクローンして git-secrets をインストールします

タスク	説明	必要なスキル
Git ソースリポジトリをクローンする。	スキャンする Git リポジトリを複製するには、ホームディレクトリから Git clone コマンドを選択します。	AWS 全般
git-secrets をクローンする。	`git-secrets` git リポジトリをクローンします。 `git clone https://github.com/awslabs/git-secrets.git` `git-secrets` の実行時に Git がそれを取得できるように、`PATH` のどこかに `git-secrets` を配置します。	AWS 全般
git-secrets をインストールする。	Unix およびバリアント (Linux/MacOS) の場合: `Makefile` (`git-secrets` リポジトリに用意されている) `install` のターゲットを使用してツールをインストールできます。`PREFIX` および `MANPREFIX` 変数を使用してインストールパスをカスタマイズできます。	AWS 全般

タスク	説明	必要なスキル

	`make install`
	
	Windows の場合:
	
	`git-secrets` リポジトリで提供されている PowerShell `install.ps1` スクリプトを実行します。このスクリプトは、インストールファイルをインストールディレクトリ (`%USERPROFILE%/.git-secrets` デフォルト) にコピーし、そのディレクトリを現在のユーザー PATH に追加します。
	
	`PS > ./install.ps1`
	
	Homebrew (macOS ユーザー) の場合:
	
	以下を実行します:
	
	`brew install git-secrets`	

git コードリポジトリをスキャンする

タスク	説明	必要なスキル
ソースリポジトリに移動する。	スキャンする Git リポジトリのディレクトリに切り替えます。	AWS 全般

タスク	説明	必要なスキル
	`cd my-git-repository`	
AWS ルールセット (Git フック) を登録する。	各コミットで Git リポジトリをスキャンするように `git-secrets` を設定するには、以下のコマンドを実行します。 `git secrets --register-aws`	AWS 全般
リポジトリをスキャンする。	次のコマンドを実行して、リポジトリのスキャンを開始します。 `git secrets --scan`	AWS 全般

タスク	説明	必要なスキル

出力ファイルを確認する。	Git リポジトリに脆弱性が見つかると、ツールは出力ファイルを生成します。以下に例を示します。
	
	`example.sh:4:AWS_SECRET_ACCESS_KEY = *********`
	
	`[ERROR] Matched one or more prohibited patterns`
	
	`Possible mitigations:`
	`- Mark false positives as allowed using: git config --add secrets.allowed ...`
	`- Mark false positives as allowed by adding regular expressions to .gitallowed at repository's root directory`
	`- List your configured patterns: git config --get-all secrets.patterns`
	`- List your configured allowed patterns: git config --get-all secrets.allowed`
	`- List your configured allowed patterns in .gitallowed at repository's root directory`
		AWS 全般

タスク	説明	必要なスキル
	```- Use --no-verify if this is a one-time false positive```	

関連リソース

- git-secrets ツール

AWS Network Firewall から Slack チャネルにアラートを送信

ベンキ・スリヴァツァフ (AWS) とアロマル・ラージ・ジャヤラジャン (AWS) によって作成されました

概要

このパターンでは、Amazon Web Services (AWS) ネットワークファイアウォールを分散デプロイモデルで使用してファイアウォールをデプロイする方法と、AWS Network Firewall によって生成されたアラートを設定可能な Slack チャネルに伝達する方法について説明します。

Payment Card Industry Data Security Standard (PCI DSS) などのコンプライアンス基準では、顧客データを保護するためにファイアウォールをインストールして維持する必要があります。AWS クラウドでは、これらのコンプライアンス要件の観点から、仮想プライベートクラウド (VPC) は物理ネットワークと同じと見なされます。Network Firewall を使用して VPC 間のネットワークトラフィックを監視し、コンプライアンス標準に準拠する VPC で実行されるワークロードを保護できます。Network Firewall は、同じアカウントの他の VPC からの不正アクセスを検出すると、アクセスをブロックするか、アラートを生成します。ただし、Network Firewall がサポートするアラートの配信先の数は限られています。これらの送信先には、Amazon Simple Storage Service (Amazon S3) バケット、Amazon CloudWatch ロググループ、Amazon Data Firehose 配信ストリームが含まれます。これらの通知に対してさらにアクションを行うには、Amazon Athena または Amazon Kinesis のいずれかを使用したオフライン分析が必要です。

このパターンは、Network Firewall によって生成されたアラートを設定可能な Slack チャネルに伝播させ、ほぼリアルタイムで次のアクションを実行できるようにします。PagerDuty、Jira、メールなどの他のアラートメカニズムにも機能を拡張できます。(これらのカスタマイズは、このパターンの範囲外です。)

前提条件と制限

前提条件

- Slack チャネル (Slack ヘルプセンターの 「はじめに」 を参照)
- チャンネルにメッセージを送信するのに必要な権限
- Slack エンドポイント URL と API トークン (「アプリを選択し」、受信ウェブフックを選択すると URL が表示されます。詳細については、Slack API ドキュメントの 「受信ウェブフックの作成」 を参照してください)
- ワークロードサブネット内の Amazon Elastic Compute Cloud (Amazon EC2) テストインスタンス

- Network Firewall テストルール
- テストルールをトリガーする実際のトラフィックまたはシミュレートされたトラフィック
- デプロイするソースファイルを保持する S3 バケット

機能制限

- 現在、このソリューションは、送信元 IP と送信先 IPs のフィルターとして 1 つのクラスレスドメイン間ルーティング (CIDR) 範囲のみをサポートしています。

アーキテクチャ

ターゲットテクノロジースタック

- 単一の VPC
- 4 つのサブネット (ファイアウォール用に 2 つ、ワークロード用に 2 つ)
- インターネットゲートウェイ
- 4 つのルートテーブルとルール
- Lambda 関数を実行するためのバケットポリシーとイベント設定で設定された、アラート送信先として使用される S3 バケット
- Slack 通知を送信するための実行ロールを持つ Lambda 関数
- Slack URL を保存するための AWS Secrets Manager のシークレット
- アラート設定付きのネットワークファイアウォール
- Slack チャンネル

Slack チャネルを除くすべてのコンポーネントは、このパターンで提供される CloudFormation テンプレートと Lambda 関数によってプロビジョニングされます (「コード」 セクションを参照)。

ターゲットアーキテクチャ

このパターンは、Slack 統合による分散型ネットワークファイアウォールを設定します。このアーキテクチャは 2 つのアベイラビリティーゾーンを持つ VPC で構成されています。VPC には、保護された 2 つのサブネットと、ネットワークファイアウォールのエンドポイントを持つ 2 つのファイアウォールサブネットがあります。保護対象サブネットに出入りするすべてのトラフィックは、「ファイアウォールポリシー」とルールを作成することで監視できます。ネットワークファイアウォールは、すべてのアラートを S3 バケットに格納するように設定されています。この S3 バケッ

トは、put イベントを受信したときに Lambda 関数を呼び出すように設定されています。Lambda 関数は、Secrets Manager から設定された Slack URL を取得し、通知メッセージを Slack ワークスペースに送信します。

このアーキテクチャの詳細については、AWS ブログ記事 「AWS Network Firewall デプロイモデル」 を参照してください。

ツール

AWS サービス

- AWS Network Firewall は、ステートフルでマネージド型のネットワークファイアウォールならびに侵入検知および防止サービスです。Network Firewall を使用して、VPC の境界でトラフィックをフィルターして、 AWS でワークロードを保護します。
- AWS Secrets Manager は、認証情報の保存と取得のためのサービスです。Secrets Manager を使用して、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、シークレットをプログラムで取得できます。このパターンでは、Secrets Manager を使用して Slack URL を保存します。
- Amazon Simple Storage Service (Amazon S3)は、オブジェクトストレージを提供します。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。このパターンでは、Amazon S3 を使用して CloudFormation テンプレートと Lambda 関数用の Python スクリプトを保存します。また、ネットワークファイアウォールのアラートの送信先として S3 バケットを使用します。
- 「AWS CloudFormation」は、AWS リソースのモデル化と設定、迅速かつ一貫したプロビジョニング、ライフサイクル全体にわたる管理を支援します。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。このパターンでは、AWS CloudFormation を使用してFirewall Manager の分散アーキテクチャを自動的にデプロイします。

コード

このパターンのコードは、GitHub 内の「Network Firewall Slack 統合」リポジトリで利用できます。`src` リポジトリのフォルダには以下があります。

- YAML 形式のCloudFormation ファイルのセット。これらのテンプレートを使用して、このパターンのコンポーネントをプロビジョニングします。

- Lambda 関数を作成するための Python ソースファイル (`slack-lambda.py`)。
- Lambda 関数コードをアップロードするための.zip アーカイブデプロイパッケージ (`slack-lambda.py.zip`)。

これらのファイルを使用するには、次のセクションの指示に従います。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを作成する。	1. AWS マネジメントコンソールにサインインして Amazon S3 コンソール「https://console.aws.amazon.com/s3/」を開きます。 2. コードをホストする S3 バケットを選択または作成します。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケット名の先頭にスラッシュを含めることはできません。「プレフィックス」を使用してこのパターンのコードを整理することをお勧めします。 詳細については、Amazon S3 ドキュメントの「バケットの作成」を参照してください。	アプリ開発者、アプリ所有者、クラウド管理者

タスク	説明	必要なスキル
CloudFormation テンプレートと Lambda コードをアップロードします。	1. このパターンの次のファイルを 「GitHub リポジトリ」 からダウンロードします。 • `base.yml` • `igw-ingress-route.yml` • `slack-lambda.py` • `slackLambda.yml` • `decentralized-deployment.yml` • `protected-subnet-route.yml` • `slack-lambda.py.zip` 2. 作成した S3 バケットにファイルをアップロードします。	アプリ開発者、アプリ所有者、クラウド管理者

CloudFormation テンプレートをデプロイ

タスク	説明	必要なスキル
CloudFormation のテンプレートを起動します。	同じAWS リージョンで S3 バケットとして「AWS CloudFormation コンソール」 を開き、テンプレート `base.yml` をデプロイします。このテンプレートは、アラートを Slack チャネルに送信するために必要な AWS リ	アプリ開発者、アプリ所有者、クラウド管理者

タスク	説明	必要なスキル
	ソースと Lambda 関数を作成します。 CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントの「AWS CloudFormation コンソールでのスタックの作成」を参照してください。	
テンプレートのパラメータを入力します。	スタック名を指定し、パラメータ値を設定します。パラメーターの一覧、説明、デフォルト値については、「追加情報」セクションのCloudFormation パラメーターを参照してください。	アプリ開発者、アプリ所有者、クラウド管理者
スタックを作成します。	1. スタックの詳細を確認し、環境要件に基づいて値を更新します。 2. スタックの作成を選択して、テンプレートをデプロイします。	アプリ開発者、アプリ所有者、クラウド管理者

ソリューションを検証する

タスク	説明	必要なスキル
デプロイをテストします。	AWS CloudFormation コンソールまたは AWS コマンドラインインターフェイス (AWS CLI) を使用して、「ターゲットテクノロジース	アプリ開発者、アプリ所有者、クラウド管理者

タスク	説明	必要なスキル
	タック」 セクションにリストされているリソースが作成されていることを確認します。 CloudFormation テンプレートが正常にデプロイされない場合は、pAvailabilityZone1 および pAvailabilityZone2 パラメータに指定した値を確認してください。これらは、ソリューションをデプロイする AWS リージョンに適したものでなければなりません。各リージョンのアベイラビリティーゾーンのリストについては、Amazon EC2 ドキュメントの 「リージョンとゾーン」 を参照してください。	

タスク	説明	必要なスキル

機能をテストします。	1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
	
	2. 保護されたサブネットの 1 つに EC2 インスタンスを作成します。HTTPS サーバーとして使用する Amazon Linux 2 AMI (HVM) を選択します。手順については、Amazon EC2 ドキュメントの「インスタンスの起動」を参照してください。
	
	Note
	Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。
	
	3. EC2 インスタンスにウェブサーバーをインストールするには、次のユーザーデータを使用します。
	
	`#!/bin/bash`
	`yum install httpd -y`
	`systemctl start httpd`
	`systemctl stop`
	`firewalld`
	`cd /var/www/html`
		アプリ開発者、アプリ所有者、クラウド管理者

タスク	説明	必要なスキル

	`echo "Hello!! this is a NFW alert test page, 200 OK" > index.html`
	
	4. 以下のネットワークファイアウォールルールを作成します。
	
	ステートレスルール:
	
	`Source: 0.0.0.0/0`
	`Destination 10.0.3.65 /32 (private IP of the EC2 instance)`
	`Action: Forward`
	
	ステートフルルール:
	
	`Protocol: HTTP`
	`Source ip/port: Any / Any`
	`Destination ip/port: Any /Any`
	
	5. ステップ 3 で作成した Web サーバーのパブリック IP を取得します。
	
	6. ブラウザでパブリック IP にアクセスします。ブラウザで次のメッセージが表示されます。
	
	`Hello!! this is a NFW alert test page, 200 OK`	

タスク	説明	必要なスキル
	Slack チャンネルにも通知が届きます。メッセージのサイズによっては、通知が遅れる場合があります。テスト目的で、狭すぎない CIDR フィルターを提供することを検討します (たとえば、/32 の CIDR 値は狭すぎ、/8 は広すぎると見なされます)。詳細については、「追加情報」 のフィルターの動作セクションを参照してください。	

関連リソース

- 「AWS Network Firewall デプロイモデル」 (AWS ブログ記事)
- 「AWS Network Firewall ポリシー」 (AWS ドキュメント)
- 「Network Firewall Slack インテグレーション」 (GitHub リポジトリ)
- 「Slack ワークスペースの作成」 (Slack ヘルプセンター)

追加情報

CloudFormation パラメーター

パラメータ	説明	デフォルト値またはサンプル値
`pVpcName`	作成する VPC の名前。	インスペクション
`pVpcCidr`	VPC が作成する CIDR 範囲。	10.0.0.0/16
`pVpcInstanceTenancy`	EC2 インスタンスが物理ハードウェアにわたって分散される方法。オプションは	デフォルト

	default (共有テナンシー) または dedicated (シングルテナンシー) です。	
pAvailabilityZone1	インフラストラクチャーの最初のアベイラビリティーゾーン。	us-east-2a
pAvailabilityZone2	インフラストラクチャの 2 つ目のアベイラビリティーゾーン。	us-east-2b
pNetworkFirewallSubnet1Cidr	1 つ目のファイアウォールサブネットの CIDR 範囲 (最低 /28)。	10.0.1.0/24
pNetworkFirewallSubnet2Cidr	2 番目のファイアウォールサブネットの CIDR 範囲 (最低 /28)。	10.0.2.0/24
pProtectedSubnet1Cidr	最初に保護された (ワークロード) サブネットの CIDR 範囲。	10.0.3.0/24
pProtectedSubnet2Cidr	2 番目の保護 (ワークロード) サブネットの CIDR 範囲。	10.0.4.0/24
pS3BucketName	Lambda ソースコードをアップロードした既存の S3 バケットの名前。	us - w2 - あなたの名前 - Lambda 関数
pS3KeyPrefix	Lambda ソースコードをアップロードした S3 バケットのプリフィックス。	AOD テスト
pAWSSecretName4Slack	Slack URL を保持するシークレットの名前。	スラッケンポイント-CFN
pSlackChannelName	作成した Slack チャンネルの名前。	一部の名前通知

pSlackUserName	Slack ユーザー名。	Slack ユーザー
pSecretKey	どのキーでもかまいません。デフォルトを使用することをお勧めします。	ウェブフック URL
pWebHookUrl	スラック URL の値。	https://hooks.slack.com/services/T???9T??/A031885JRM7/9D4Y?????
pAlertS3Bucket	ネットワークファイアウォールのアラート送信先として使用する S3 バケットの名前。このバケットが作成されます。	us - w2 - あなたの名前 - セキュリティ - aod - アラート
pSecretTagName	シークレットのタグ名。	AppName
pSecretTagValue	指定されたタグ名のタグ値。	ラムダとスラックインテグレーション
pdestCidr	デスティネーション CIDR 範囲のフィルター。詳細については、次のセクションのフィルター行為をご覧ください。	10.0.0.0/16
pdestCondition	デスティネーションマッチを除外するか含めるかを示すフラグ。詳細については、次のセクションをご覧ください。有効な値は、include および exclude です。	include
psrcCidr	警告するソース CIDR 範囲のフィルター。詳細については、次のセクションをご覧ください。	118.2.0.0/16

psrcCondition	ソースマッチを除外または含めるフラグ。詳細については、次のセクションをご覧ください。	include

フィルター動作

AWS Lambda でフィルターを設定していない場合、生成されたすべてのアラートは Slack チャンネルに送信されます。生成されたアラートの送信元 IP と送信先 IP は、CloudFormation テンプレートをデプロイしたときに設定した CIDR 範囲と照合されます。一致が検出された場合、条件が適用されます。送信元または送信先のどちらかが設定された CIDR 範囲内にあり、そのうち 1 つでも include 条件が設定されている場合、アラートが生成されます。以下の表は、CIDR の値、条件、結果の例を示しています。

	CIDR	アラート IP	Configured	アラート
ソース	10.0.0.0/16	10.0.0.25	include	はい
送信先	100.0.0.0/16	202.0.0.13	include	

	CIDR	アラート IP	Configured	アラート
ソース	10.0.0.0/16	10.0.0.25	exclude	いいえ
送信先	100.0.0.0/16	202.0.0.13	include	

	CIDR	アラート IP	Configured	アラート
ソース	10.0.0.0/16	10.0.0.25	include	はい
送信先	100.0.0.0/16	100.0.0.13	include	

	CIDR	アラート IP	Configured	アラート

ソース	10.0.0.0/16	90.0.0.25	include	はい
送信先	Null	202.0.0.13	include	

	CIDR	アラート IP	Configured	アラート
ソース	10.0.0.0/16	90.0.0.25	include	いいえ
送信先	100.0.0.0/16	202.0.0.13	include	

AWS Private CA と AWS RAM を使用してプライベート証明書の管理を簡素化する

作成者: Everett Hinckley (AWS) と Vivek Goyal (AWS)

概要

AWS Private Certificate Authority (AWS Private CA) を使用して、内部リソースを認証し、コンピュータコードに署名するためのプライベート証明書を発行できます。このパターンは、複数レベルの CA 階層を迅速にデプロイし、一貫したプロビジョニングを実現するための AWS CloudFormation テンプレートを提供します。オプションで、AWS Resource Access Manager (AWS RAM) を使用して、組織内または AWS Organizations の組織単位 (OU) 内で CA を安全に共有し、CA を一元化しながら AWS RAM を使用して権限を管理できます。すべてのアカウントに private CA は必要ないため、この方法によりコストを削減できます。さらに、Amazon Simple Storage Service (Amazon S3) を使用して、証明書失効リスト (CRL) およびアクセスログを保存 できます。

この実装には次の特徴と利点があります。

- AWS Private CA を使用して、AWS Private CA 階層の管理を一元化および簡素化できる。
- AWS とオンプレミスの顧客管理デバイスに証明書とキーをエクスポートできる。
- AWS CloudFormation テンプレートを使用して、迅速なデプロイと一貫したプロビジョニングを実現。
- 1、2、3、または 4 つの下位 CA 階層とともにプライベートルート CA を作成できる。
- オプションで、AWS RAM を使用してエンドエンティティの下位 CA を組織または OU レベルで他のアカウントと共有できる。
- AWS RAM を使用することですべてのアカウントに private CA を設定する必要がなくなるため、コストを削減できる。
- CRL 用にオプションの S3 バケットを作成できる
- CRL アクセスログ用にオプションの S3 バケットを作成できる。

前提条件と制限

前提条件

AWS Organizations 構造内で CA を共有する場合は、以下を特定または設定します。

- CA 階層と共有を作成するためのセキュリティアカウント。

- テスト用の別の OU またはアカウント。
- AWS Organizations 管理アカウント内で共有が有効になっていること。詳細については、AWS RAM ドキュメントの「AWS Organizations 内でリソース共有を有効にする」を参照してください。

機能制限

- CA はリージョンのリソースです。CA はすべて、1 つの AWS アカウントと 1 つの AWS リージョンに存在します。
- ユーザー生成の証明書とキーはサポートされていません。このユースケースでは、外部のルート CA を使用するようにこのソリューションをカスタマイズすることをお勧めします。
- パブリック CRL バケットはサポートされていません。CRL はプライベートに維持することをお勧めします。CRL へのインターネットアクセスが必要な場合は、AWS Private CA ドキュメントの「S3 ブロックパブリックアクセス (BPA) 機能の有効化」の「Amazon CloudFront を使用して CRL を提供する」セクションを参照してください。
- このパターンでは、単一リージョンのアプローチを実装しています。マルチリージョン認証局が必要な場合は、2 つ目の AWS リージョンまたはオンプレミスに下位 CA を実装できます。その場合の実装は特定のユースケース、ワークロードボリューム、依存関係、要件によって異なるため、そうした複雑性はこのパターンの範囲外になります。

アーキテクチャ

ターゲットテクノロジースタック

- AWS Private CA
- AWS RAM
- Amazon S3
- AWS Organizations
- AWS CloudFormation

ターゲットアーキテクチャ

このパターンには、AWS Organizations と共有するためのオプションが 2 つあります。

オプション 1 – 組織レベルで共有を作成する。次の図に示すように、組織内のすべてのアカウントが共有 CA を使用してプライベート証明書を発行できます。

オプション 2 – 組織単位 (OU) レベルで共有を作成する。指定した OU 内のアカウントのみが、共有 CA を使用してプライベート証明書を発行できます。たとえば、以下の図では、共有がサンドボックス OU レベルで作成されている場合、デベロッパー 1 とデベロッパー 2 の両方が共有 CA を使用してプライベート証明書を発行できます。

ツール

サービス

- Private CA – AWS Private Certificate Authority (AWS Private CA) は、プライベートデジタル証明書の発行と取り消しに使用するホスト型 CA サービスです。オンプレミス CA の運用にかかる投資コストや保守コストなしに、ルート CA や下位 CA を含む CA 階層を作成できます。
- RAM — AWS Resource Access Manager (AWS RAM) を使用すると、AWS アカウント間、組織内、または AWS Organizations 内の OU 内でリソースを安全に共有できます。マルチアカウント環境における運用オーバーヘッドを減らすには、リソースを作成し、AWS RAM を使用してそのリソースをアカウント間で共有できます。
- Organizations – AWS Organizations は、作成して一元管理している複数の AWS アカウントを組織に統合するためのアカウント管理サービスです。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、オブジェクトストレージサービスです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。このパターンでは、証明書失効リスト (CRL) とアクセスログを Amazon S3 を使用して保存します。
- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して構成できます。このパターンでは、AWS CloudFormation を使用してマルチレベルの CA 階層を自動的にデプロイします。

コード

このパターンのソースコードは、GitHub の Private CA 階層リポジトリにあります。リポジトリには以下が含まれます。

- AWS CloudFormation テンプレート `ACMPCA-RootCASubCA.yaml`。このテンプレートは、この実装の CA 階層をデプロイするために使用できます。
- 証明書のリクエスト、エクスポート、記述、削除などのユースケース用のテストファイル。

これらのファイルを使用するには、「エピック」セクションの指示に従ってください。

エピック

CA 階層の設計

タスク	説明	必要なスキル
証明書の対象情報を収集します。	証明書の所有者に関する証明書サブジェクト情報 (組織名、組織単位、国、州、リージョン、通称) を収集します。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア
AWS Organizations に関するオプション情報を収集します。	CA を AWS Organizations 構造の一部とし、その構造内で CA 階層を共有したい場合は、管理アカウント番号、組織 ID、およびオプションで OU ID を収集します (CA 階層を特定の組織とのみ共有する場合)。また、CA を共有したい AWS Organizations アカウントまたは OU (ある場合) も決定してください。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア
CA 階層を設計します。	ルート CA と下位 CA をどのアカウントに格納するかを決めます。ルート証明書とエンドエンティティ証明書の間に階層が必要とする下位レベルの数を決定します。詳細については、AWS Private CA ド	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア

タスク	説明	必要なスキル
	キュメントの「CA 階層の設計」を参照してください。	
CA 階層の命名規則とタグ付け規則を決定します。	AWS リソースの名前 (ルート CA と各下位 CA) を決定します。各 CA にどのタグを割り当てるかを決めてください。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア
必要な暗号化アルゴリズムと署名アルゴリズムを決定する。	以下を決定します。 • CA が証明書を発行する際に使用するパブリックキーに関する組織の暗号化アルゴリズム要件。デフォルトは RSA_2048 です。 • CA が証明書署名に使用するキーアルゴリズム。デフォルトは SHA256WITHRSA です。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア
CA 階層の証明書失効要件を決定します。	証明書失効機能が必要な場合は、証明書失効リスト (CRL) を含む S3 バケットの命名規則を設定します。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア
CA 階層の口ギング要件を決定します。	アクセス口ギング機能が必要な場合は、アクセスログを含む S3 バケットの命名規則を設定します。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア

タスク	説明	必要なスキル
証明書の有効期限を決定します。	ルート証明書 (デフォルトは 10 年)、エンドエンティティ証明書 (デフォルトは 13 か月)、および下位 CA 証明書 (デフォルトは 3 年) の有効期限を決定します。下位 CA 証明書は、階層内の上位レベルの CA 証明書よりも早く有効期限が切れる必要があります。詳細については、AWS Private CA ドキュメントの「Private CA ライフサイクルの管理」を参照してください。	クラウドアーキテクト、セキュリティアーキテクト、PKI エンジニア

CA 階層のデプロイ

タスク	説明	必要なスキル
前提条件を満たす。	このパターンの「前提条件」セクションの手順を完了してください。	クラウド管理者、セキュリティエンジニア、PKI エンジニア
さまざまなペルソナの CA ロールを作成します。	1. RootCAAdmin、SubordinateCAAdmin、CertificateConsumer など、CA 階層のさまざまなレベルを管理するために必要な、AWS IAM アイデンティティセンター (AWS シングルサインオンの後継) のAWS Identity and Access Management (IAM) ロールまたはユー	クラウド管理者、セキュリティエンジニア、PKI エンジニア

タスク	説明	必要なスキル
	ザーのタイプを決定します。 2. 職務を分担するのに必要なポリシーの細分性を決定します。 3. CA 階層が存在するアカウントの IAM Identity Center に必要な IAM ロールまたはユーザーを作成します。	
CloudFormation スタックをデプロイします。	1. このパターンの GitHub リポジトリから AWSPCA-RootCASubCA.yaml テンプレートをダウンロードします。 2. AWS CloudFormation コンソールまたは AWS コマンドラインインターフェイス (AWS CLI) からテンプレートをデプロイします。詳細については、CloudFormation ドキュメントの「スタックの作業」を参照してください。 3. 組織名、OU 名、キーアルゴリズム、署名アルゴリズム、その他のオプションなど、テンプレートのパラメータを入力します。	クラウド管理者、セキュリティエンジニア、PKI エンジニア

タスク	説明	必要なスキル
ユーザー管理リソースが使用する証明書を更新するためのソリューションを設計します。	Elastic Load Balancing などの統合 AWS サービスのリソースは、有効期限が切れる前に証明書を自動的に更新します。ただし、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されるウェブサーバーなどのユーザー管理のリソースには、別のメカニズムが必要です。 1. Private CA からのエンドエンティティ証明書を必要とするユーザー管理のリソースを決定します。 2. ユーザーが管理するリソースと証明書の有効期限を通知するプロセスを計画してください。 の例については、次を参照してください。 • AWS Config マネージドルールを使用する • Amazon EventBridge および Amazon CloudWatch を使用する 3. ユーザー管理のリソースの証明書を更新するカスタムスクリプトを記述し、それらを AWS サービスと統合して更新を自動化します。統合された AWS サービスの詳細については、ACM ドキュメントの	クラウド管理者、セキュリティエンジニア、PKI エンジニア

タスク	説明	必要なスキル
	「AWS Certificate Manager と統合されたサービス」を参照してください。	

CA 階層の検証と文書化

タスク	説明	必要なスキル
オプションの AWS RAM 共有を検証します。	CA 階層が AWS Organizations の他のアカウントと共有されている場合は、AWS マネジメントコンソールからいずれかのアカウントにログインし、Private CA コンソールに移動して、新しく作成した CA がこのアカウントと共有されていることを確認します。階層の最下位レベルの CA のみが表示されます。これは、エンドエンティティ証明書を生成する CA だからです。CA を共有しているアカウントのサンプルについても同じ手順を繰り返します。	クラウド管理者、セキュリティエンジニア、PKI エンジニア
証明書ライフサイクルテストで CA 階層を検証します。	このパターンの GitHub リポジトリで、ライフサイクルテストを見つけます。AWS CLI からテストを実行して、証明書のリクエスト、証明書のエクスポート、証明書の説明、証明書の削除を行います。	クラウド管理者、セキュリティエンジニア、PKI エンジニア

タスク	説明	必要なスキル
証明書チェーンをトラストストアにインポートします。	ブラウザやその他のアプリケーションが証明書を信頼するには、証明書の発行者がブラウザーのトラストストア (信頼できる CA のリスト) に含まれている必要があります。新しい CA 階層の証明書チェーンをブラウザとアプリケーションのトラストストアに追加します。エンドエンティティの証明書が信頼されていることを確認してください。	クラウド管理者、セキュリティエンジニア、PKI エンジニア
CA 階層を文書化するランブックを作成します。	CA 階層のアーキテクチャ、エンドエンティティ証明書を要求できるアカウント構造、ビルドプロセス、およびエンドエンティティ証明書の発行 (子アカウントによるセルフサービスを許可する場合を除く)、使用方法、追跡などの基本的な管理タスクを説明するランブックドキュメントを作成します。	クラウド管理者、セキュリティエンジニア、PKI エンジニア

関連リソース

- CA 階層の設計 (AWS Private CA ドキュメント)
- Private CA の作成 (AWS Private CA ドキュメント)
- RAM を使用して Private CA のクロスアカウントを共有する方法 (AWS ブログ記事)
- Private CA のベストプラクティス (AWS ブログ記事)
- AWS Organizations 内でのリソース共有を有効にする (AWS RAM ドキュメント)
- Private CA ライフサイクルの管理 (AWS Private CA ドキュメント)

- Config の ACM 証明書の有効期限チェック (AWS Config ドキュメント
- Certificate Manager が Amazon CloudWatch を通じて証明書の有効期限のモニタリングを提供するようになりました (AWS 発表)
- AWS Certificate Manager (ACM) と統合されたサービス (ACM ドキュメント)

追加情報

証明書をエクスポートするときは、暗号的に強力で、組織のデータ損失防止戦略に沿ったパスフレーズを使用してください。

マルチアカウント環境ですべてのセキュリティハブのメンバーアカウントにわたって、セキュリティ標準コントロールをオフにする

作成者: Michael Fuellbier (AWS)とAhmed Bakry (AWS)

概要

> **Important**
>
> AWS Security Hub は、アカウント間でセキュリティ標準とコントロールの一元的な設定をサポートするようになりました。この新機能は、この AWS 規範ガイダンスパターンのソリューションで力バーされる多くのシナリオに対処します。このパターンでソリューションをデプロイする前に、Security Hub の「Central configuration」を参照してください。

Amazon Web Services (AWS) クラウドでは、CIS AWS Foundations Benchmark や AWS Foundational Security Best Practices などの AWS Security Hub 標準コントロールは、単一の 内からのみ手動でオフ (無効化) にすることができます AWS アカウント。マルチアカウント環境では、複数のセキュリティハブメンバーアカウントのコントロールを「ワンクリック」(つまり 1 回の API 呼び出し) でオフにすることはできません。このパターンでは、セキュリティハブの管理者アカウントによって管理されているすべてのセキュリティハブメンバーアカウントにわたって、ワンクリックでセキュリティハブ標準コントロールをオフにする方法を示します。

前提条件と制限

前提条件

- 複数のメンバーアカウントを管理するセキュリティハブ管理者アカウントで構成されるマルチアカウント環境
- AWS Command Line Interface （ AWS CLI) バージョン 2、インストール済み
- AWS Serverless Application Model コマンドラインインターフェイス (AWS SAM CLI)、インストール済み

機能制限

- このパターンは、単一のセキュリティハブ管理者アカウントが複数のメンバーアカウントを管理する、マルチアカウント環境でのみ機能します。

- 非常に短い時間枠で多くのコントロールを変更すると、イベント開始により複数の並列呼び出しが発生します。これにより、 API スロットリングが発生し、呼び出しが失敗する可能性があります。たとえば、このシナリオでは、「セキュリティハブコントロールCLI」 を使用して、多くのコントロールをプログラムで変更した場合に発生することがあります。

アーキテクチャ

次の図は、複数の Security Hub メンバーアカウント (Security Hub 管理者アカウントから表示) で Security Hub 標準コントロールをオフにする AWS Step Functions ワークフローの例を示しています。

この図表は、次のワークフローを示しています：

1. Amazon EventBridge ルールは毎日のスケジュールで開始され、ステートマシンを呼び出します。AWS CloudFormation テンプレートの `Schedule`パラメータを更新することで、ルールのタイミングを変更できます。
2. EventBridge ルールは、セキュリティハブ管理者アカウントでコントロールがオンまたはオフになるたびに開始されます。
3. ステップ関数ステートマシンは、セキュリティ標準コントロール (つまり、オンまたはオフになるコントロール) のステータスを、セキュリティハブ管理者アカウントからメンバーアカウントに伝達します。
4. クロスアカウント AWS Identity and Access Management (IAM) ロールは、各メンバーアカウントにデプロイされ、ステートマシンによって引き受けられます。ステートマシンは、各メンバーアカウントのコントロールをオンまたはオフにします。
5. Amazon DynamoDB テーブルには、特定のアカウントでどのコントロールを有効または無効にするかに関する例外と情報が含まれています。この情報は、指定されたメンバーアカウントの、セキュリティハブ管理者アカウントから取得した設定を上書きします。

> **Note**
>
> スケジュールされた EventBridge ルールの目的は、新しく追加された Security Hub メンバーアカウントが既存のアカウントと同じコントロールステータスになるようにすることです。

ツール

AWS のサービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを提供します。
- 「Amazon EventBridge」 は、アプリケーションをさまざまなソースのデータに接続するために支援するサーバーレスイベントバスサービスです。例えば、 AWS Lambda 関数、API 送信先を使用する HTTP 呼び出しエンドポイント、その他のイベントバスなどです AWS アカウント。
- AWS Command Line Interface （ AWS CLI ） は、コマンドラインシェルのコマンド AWS のサービス を使用して を操作するのに役立つオープンソースツールです。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。
- AWS Serverless Application Model （ AWS SAM ） は、 でサーバーレスアプリケーションを構築するのに役立つオープンソースフレームワークです AWS クラウド。
- AWS Security Hub は、 のセキュリティ状態の包括的なビューを提供します AWS。また、セキュリティ業界標準やベストプラクティスに照らして AWS 環境を確認するのにも役立ちます。
- AWS Step Functions は、 AWS Lambda 関数と他の を組み合わせてビジネスクリティカルなアプリケーション AWS のサービス を構築するのに役立つサーバーレスオーケストレーションサービスです。

コードリポジトリ

このパターンのコードは、GitHub AWS Security Hub Cross-Account Controls Disabler リポジトリで入手できます。コードリポジトリには以下のファイルとフォルダが含まれています。

- UpdateMembers/template.yaml – このファイルには、Step Functions ステートマシンや EventBridge ルールなど、Security Hub 管理者アカウントにデプロイされたコンポーネントが含まれています。
- member-iam-role/template.yaml – このファイルには、クロスアカウント IAM ロールをメンバーアカウントにデプロイするためのコードが含まれています。
- stateMachine.json – このファイルはステートマシンのワークフローを定義します。
- GetMembers/index.py – このファイルには GetMembers ステートマシンのコードが含まれています。スクリプトは、既存のすべてのセキュリティハブメンバーアカウントのセキュリティ標準コントロールのステータスを取得します。

- UpdateMember/index.py – このファイルには、各メンバーアカウントのコントロールステータスを更新するスクリプトが含まれています。
- CheckResult/index.py – このファイルには、ワークフロー呼び出しのステータス (承認または失敗) を確認するスクリプトが含まれています。

エピック

セキュリティハブメンバーアカウントにクロスアカウント IAM ロールをデプロイ

タスク	説明	必要なスキル
セキュリティハブ管理者アカウントのアカウント ID を識別します。	「セキュリティハブ管理者アカウント」 を設定し、管理者アカウントのアカウント ID を書き留めます。	クラウドアーキテクト
クロスアカウント IAM ロールを含む CloudFormation テンプレートをメンバーアカウントにデプロイします。	Security Hub 管理者アカウントによって管理されるすべてのメンバーアカウントに member-iam-role/template.yaml テンプレートをデプロイするには、次のコマンドを実行します。 `aws cloudformation deploy \ --template-file member-iam-role/template.yaml \ --capabilities CAPABILITY_NAMED_IAM \ --stack-name <your-stack-name> \ --parameter-overrides SecurityHubAdminAccountId=<account-ID>`	AWS DevOps

タスク	説明	必要なスキル
	SecurityHubAdminAccountId のパラメータは、先にメモしておいたセキュリティハブ管理者アカウント ID と一致する必要があります。	

セキュリティハブ管理者アカウントにステートマシンをデプロイ

タスク	説明	必要なスキル
ステートマシンを含む CloudFormation テンプレートをパッケージ化します AWS SAM。	Security Hub 管理者アカウントに UpdateMembers/template.yaml テンプレートをパッケージ化するには、次のコマンドを実行します。 `sam package \ --template-file UpdateMembers/template.yaml \ --output-template-file UpdateMembers/template-out.yaml \ --s3-bucket <amzn-s3-demo-bucket>` **Note** Amazon Simple Storage Service (Amazon S3) バケットは、CloudFormation テンプレートをデプロイする AWS リージョ	AWS DevOps

タスク	説明	必要なスキル
	ン のと同じ にある必要があります。	

タスク	説明	必要なスキル
セキュリティハブ管理者アカウントにパッケージ化済みの CloudFormation テンプレートをデプロイします。	セキュリティハブ管理者アカウントに CloudFormation テンプレートをデプロイするには、以下のコマンドを実行します： `aws cloudformation deploy \ --template-file UpdateMembers/template-out.yaml \ --capabilities CAPABILITY_IAM \ --stack-name <stack-name>` member-iam-role/template.yaml テンプレートでは、`MemberIAMRolePath` パラメータは `IAMRolePath` パラメータと一致し、 と一致する`MemberIAMRoleName` 必要がありますIAMRoleName 。 **Note** Security Hub はリージョンレベルのサービスであるため、テンプレートはそれぞれに個別にデプロイする必要があります AWS リージョン。まず、ソリューションを各リージョンの Amazon S3	AWS DevOps

タスク	説明	必要なスキル
	バケットにパッケージ化してください。	

関連リソース

- Security Hub 管理者アカウントの指定 (Security Hub ドキュメント)
- エラーの処理、再試行、 AWS Step Functions ステートマシンの実行へのアラートの追加 (AWS ブログ記事)

PowerShell を使用して AWS IAM アイデンティティセンターから AWS CLI 認証情報を更新

作成者: Chad Miles (AWS) と Andy Bowen (AWS)

概要

AWS IAM アイデンティティセンター (AWS シングルサインオンの後継) 認証情報を、AWS コマンドラインインターフェイス (AWS CLI)、AWS SDK、または AWS Cloud Development Kit (AWS CDK) と共に使用する場合、通常、IAM アイデンティティセンターコンソールからコマンドラインインターフェイスに認証情報をコピーして貼り付ける必要があります。このプロセスにはかなりの時間がかかることがあり、アクセスが必要なアカウントごとに繰り返す必要があります。

１つの一般的な解決策は、AWS CLI `aws sso configure` コマンドを使用することです。このコマンドにより、IAM アイデンティティセンターで有効にされたプロファイルを AWS CLI または AWS SDK に追加します。ただし、このソリューションの欠点は、このように設定した AWS CLI プロファイルまたはアカウントごとにコマンド `aws sso login` を実行する必要があることです。

代替ソリューションとして、このパターンでは、AWS CLI「名前付きプロファイル」と AWS Tools for PowerShell を使用して、単一の IAM アイデンティティセンターのインスタンスから複数のアカウントの認証情報を同時に保存および更新する方法を説明します。また、このスクリプトは IAM アイデンティティセンターセッションデータをメモリに保存するので、IAM アイデンティティセンターに再度ログインしなくても認証情報を更新します。

前提条件と制限

前提条件

- PowerShell がインストールされ、設定されています。詳細については、「PowerShellをインストール」(Microsoft のドキュメント)を参照してください。
- AWS Tools for PowerShell がインストールされ、設定されました。パフォーマンスの理由で、モジュール化された AWS Tools for PowerShell バージョンをインストールすることを強く推奨します。 `AWS.Tools` と呼びます。各AWSサービスが、それ自身の個別の小さなモジュールによって適用されます。PowerShell プロンプトで、このパターンに必要なモジュールをインストールするためのコマンド `AWS.Tools.Installer` 、 `SSO` 、および `SSOIDC` を入力します。

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule SSO, SSOOIDC
```

詳細については、「Windows で AWS.Tools をインストール」 または 「Linux または macOS で AWS.Tools をインストール」 を参照してください。

- AWS CLI または AWS SDK は、以下のいずれかを実行して、あらかじめ有効な認証情報を使用して設定する必要があります：
  - AWS CLIの`aws configure`コマンドを使用します。詳細については、「クィック設定」（AWS CLI 文書）を参照してください。
  - IAM ロールを通じて一時的なアクセスを取得するように 、AWS CLI または AWS CDK を設定します。詳細については、「CLI アクセスの IAM ロール認証情報を取得」(IAM アイデンティティセンタードキュメント) を参照してください。

制約事項

- このスクリプトは、パイプラインまたは完全自動化ソリューションに使用できません。このスクリプトをデプロイする場合、IAM アイデンティティセンターからのアクセスを手動で承認する必要があります。その後、スクリプトは自動的に続行されます。

製品バージョン

- すべてのオペレーティングシステムでは、「PowerShell バージョン 7.0」 以降を使用することを推奨します。

アーキテクチャ

このパターンでスクリプトを使用して、複数の IAM センターの認証情報を同時に更新できます。また、AWS CLI、AWS SDK、または AWS CDK で使用する認証情報ファイルを作成できます。

ツール

AWS サービス

- 「AWS コマンドラインインターフェイス (AWS CLI)」は、オープンソースのツールであり、コマンドラインシェルのコマンドを使用して AWS サービスとやり取りすることができます。
- 「AWS IAM アイデンティティセンター」 により、すべての AWS アカウントとクラウドアプリケーションへのシングルサインオン (SSO) アクセスを一元管理できます。

- 「AWS Tools for PowerShell」 は PowerShell のコマンドラインから AWS リソースの操作をスクリプト処理することを支援する PowerShell モジュールのセットです。

その他のツール

- 「PowerShell」 は Windows、Linux、および macOS で動作するMicrosoft の自動化および構成管理プログラムです。

ベストプラクティス

このスクリプトは、各 IAM アイデンティティセンターのインスタンスに対して 1 つずつ保留します。1 つのスクリプトを複数のインスタンスに使用することは適用されません。

エピック

SSO スクリプトを実行する

タスク	説明	必要なスキル
SSO スクリプトをカスタマイズします。	1. 「追加情報」 セクションの SSO スクリプトをコピーします。 2. Param セクションでは、使用中の AWS 環境に合わせて、以下の変数の値を定義します。 • DefaultRoleName — デフォルトで使用されるように設定された IAM ロールまたは権限です。 • Region — IAM アイデンティティセンターがデプロイされている AWS リージョン。リージョンの全リストについては、「リージョンのエンドポ	クラウド管理者

タスク	説明	必要なスキル
	イント」を参照してください。 • StartUrl — IAM アイデンティティセンターのログインページへのアクセスに使用される URL。スクリプト内のサンプル値と同じ形式を使用します。 • EnvironmentName — 同じセッションで複数のスクリプトコピーを実行する場合に使用する、このスクリプトのコピーを参照する略称。 3. 10 行目 の下で # Add your Account Information と呼ばれ、環境に合わせてハッシュテーブル内の以下の値を編集します： • Profile — 一時的な認証情報を保存する AWS CLI プロファイル名。 • AccountId — 認証情報を取得する AWS アカウントの ID。 • RoleName — 使用する IAM アイデンティティセンターのロールまたは権限セットの名前。Param セクションで定義したのと同じロールを使用	

タスク	説明	必要なスキル
	する場合、 `$DefaultRoleName` のままにしても良いです。 ハッシュテーブルの各行は、最後の行を除いてカンマで終わる必要があります。	

実行するスクリプト。	PowerShell シェルで次のコマンドを使用してカスタムスクリプトを実行することを推奨します。
	
	`./Set-AwsCliSsoCredentials.ps1`
	
	代りに、次のコマンドを入力して、別のシェルからスクリプトを実行できます。
	
	`pwsh Set-AwsCliSsoCredentials.ps1`
		クラウド管理者

トラブルシューティング

問題	ソリューション
`No Access` エラー	使用している IAM ロールには、`RoleName`のパラメータで定義したロールまたは権限セットにアクセスする権限がありません。使用しているロールの権限を更新するか、スクリプトで別のロールまたは権限セットを定義します。

関連リソース

- 構成設定はどこに保存されていますか？（ AWS CLI ドキュメント ）
- 「AWS IAM IM アイデンティティセンターを使用するための AWS CLI の設定」(AWS CLI ドキュメント)
- 「名前付きプロファイルを使用」(AWS CLI ドキュメント)

追加情報

SSO スクリプト

次のスクリプトでは、山括弧 (<>) 内のプレースホルダーを独自の情報に置き換えたら、山括弧を削除します。

```
Set-AwsCliSsoCredentials.ps1
Param(
    $DefaultRoleName = '<AWSAdministratorAccess>',
    $Region          = '<us-west-2>',
    $StartUrl        = "<https://d-12345abcde.awsapps.com/start/>",
    $EnvironmentName = "<CompanyName>"
)
Try {$SsoAwsAccounts = (Get-Variable -name "$($EnvironmentName)SsoAwsAccounts" -Scope
 Global -ErrorAction 'SilentlyContinue').Value.Clone()}
Catch {$SsoAwsAccounts = $False}
if (-not $SsoAwsAccounts) { $SsoAwsAccounts = @(
# Add your account information in the list of hash tables below, expand as necessary,
 and do not forget the commas
    @{Profile = "<Account1>"      ; AccountId = "<012345678901 >"; RoleName =
 $DefaultRoleName },
    @{Profile = "<Account2>"      ; AccountId = "<123456789012>"; RoleName =
 "<AWSReadOnlyAccess>" }
)}
$ErrorActionPreference = "Stop"
if (-not (Test-Path ~\.aws))      { New-Item ~\.aws -type Directory }
if (-not (Test-Path ~\.aws\credentials)) { New-Item ~\.aws\credentials -type File }
$CredentialFile = Resolve-Path ~\.aws\credentials
$PsuedoCreds    = @{AccessKey =
 'AKAEXAMPLE123ACCESS';SecretKey='PsuedoS3cret4cceSSKey123PsuedoS3cretKey'} # Pseudo
 Creds, do not edit.
```

```
Try {$SSOTokenExpire = (Get-Variable -Scope Global -Name
 "$($EnvironmentName)SSOTokenExpire" -ErrorAction 'SilentlyContinue').Value} Catch
 {$SSOTokenExpire = $False}
Try {$SSOToken       = (Get-Variable -Scope Global -Name "$($EnvironmentName)SSOToken"
 -ErrorAction 'SilentlyContinue').Value }      Catch {$SSOToken       = $False}
if ( $SSOTokenExpire -lt (Get-Date) ) {
    $SSOToken = $Null
    $Client   = Register-SSOOIDCClient -ClientName cli-sso-client -ClientType public -
Region $Region @PsuedoCreds
    $Device   = $Client | Start-SSOOIDCDeviceAuthorization -StartUrl $StartUrl -Region
 $Region @PsuedoCreds
    Write-Host "A Browser window should open. Please login there and click ALLOW." -
NoNewline
    Start-Process $Device.VerificationUriComplete
    While (-Not $SSOToken){
        Try {$SSOToken = $Client | New-SSOOIDCToken -DeviceCode $Device.DeviceCode -
GrantType "urn:ietf:params:oauth:grant-type:device_code" -Region $Region @PsuedoCreds}
        Catch {If ($_.Exception.Message -notlike "*AuthorizationPendingException*")
{Write-Error $_.Exception} ; Start-Sleep 1}
    }
    $SSOTokenExpire = (Get-Date).AddSeconds($SSOToken.ExpiresIn)
    Set-Variable -Name "$($EnvironmentName)SSOToken" -Value $SSOToken -Scope Global
    Set-Variable -Name "$($EnvironmentName)SSOTokenExpire" -Value $SSOTokenExpire -
Scope Global
}
$CredsTime     = $SSOTokenExpire - (Get-Date)
$CredsTimeText = ('{0:D2}:{1:D2}:{2:D2} left on SSO Token' -f $CredsTime.Hours,
 $CredsTime.Minutes, $CredsTime.Seconds).TrimStart("0 :")
for ($i = 0; $i -lt $SsoAwsAccounts.Count; $i++) {
    if (([DateTimeOffset]::FromUnixTimeSeconds($SsoAwsAccounts[$i].CredsExpiration /
 1000)).DateTime -lt (Get-Date).ToUniversalTime()) {
        Write-host "`r
    `rRegistering Profile $($SsoAwsAccounts[$i].Profile)" -NoNewline
        $TempCreds = $SSOToken | Get-SSORoleCredential -AccountId
 $SsoAwsAccounts[$i].AccountId -RoleName $SsoAwsAccounts[$i].RoleName -Region $Region
 @PsuedoCreds
        [PSCustomObject]@{AccessKey = $TempCreds.AccessKeyId; SecretKey =
 $TempCreds.SecretAccessKey; SessionToken = $TempCreds.SessionToken
        } | Set-AWSCredential -StoreAs $SsoAwsAccounts[$i].Profile -ProfileLocation
 $CredentialFile
        $SsoAwsAccounts[$i].CredsExpiration = $TempCreds.Expiration
    }
}
```

```
Set-Variable -name "$($EnvironmentName)SsoAwsAccounts" -Value $SsoAwsAccounts.Clone() -
Scope Global
Write-Host "`r$($SsoAwsAccounts.Profile) Profiles registered, $CredsTimeText"
```

AWS Config を使用して Amazon Redshift のセキュリティ設定をモニタリング

作成者: Lucas Kauffman (AWS) と abhishek sengar (AWS)

概要

AWS Config をして、AWS リソースのセキュリティ構成を評価できます。AWS Config はリソースを監視できます。構成の設定が定義したルールに違反している場合は、AWS Config はそのリソースに非準拠のフラグを付けます。

AWS Config を使用して、Amazon Redshift クラスターとデータベースを評価し監視できます。セキュリティの推奨事項と特徴量の詳細について、「Amazon Redshift のセキュリティ」 を参照してください。このパターンでは、AWS Config のカスタム AWS Lambda ルールが含まれています。これらのルールをアカウントにデプロイして、Amazon Redshift クラスターとデータベースのセキュリティ設定をモニタリングできます。このパターンのルールでは、AWS Config を使用して、次のことを確認できます:

- Amazon Redshift クラスターのデータベースで監査ログが有効になっている
- Amazon Redshift クラスターに接続するには SSL が必要である
- 連邦情報処理規格 (FIPS) 暗号が使用中である
- Amazon Redshift クラスター内のデータベースが暗号化されている
- ユーザーアクティビティモニタリングが有効になっている

前提条件と制限

前提条件

- アクティブなAWS アカウント
- AWS Config は、AWS アカウントで有効にする必要があります。詳細については、「コンソールでAWS Configをセットアップする」または「AWS CLIでAWS Configをセットアップする」を参照してください。
- AWS Lambda ハンドラーには、Python バージョン 3.9 以降を使用する必要があります。詳細については、「Python での作業」 (AWS Lambda ドキュメント) を参照してください。

製品バージョン

- Python バージョン 3.9 以降

アーキテクチャ

ターゲットテクノロジースタック

- AWS Config

ターゲットアーキテクチャ

1. AWS Config は定期的にカスタムルールを実行します。
2. カスタムルールは、Lambda 関数を呼び出します。
3. Lambda 関数は、Amazon Redshift クラスターに非準拠の設定がないかどうかをチェックします。
4. Lambda 関数は、各 Amazon Redshift クラスターのコンプライアンス状態を AWS Config に報告します。

自動化とスケール

AWS Config カスタムルールは、アカウント内のすべての Amazon Redshift クラスターを評価するようにスケールされます。このソリューションをスケールするために追加のアクションは必要ありません。

ツール

AWS サービス

- AWS Config は、AWS アカウントの AWS リソースの設定とその設定方法ついて詳細に表示します。リソースがどのように相互に関連しているかと、それらの構成が時間の経過とともにどのように変化したかを特定することを支援します。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Lambda は、サーバーのプロビジョニングや管理を行うことなくコードを実行できるコンピューティングサービスです。必要に応じてコードを実行し、自動的にスケーリングするため、課金は実際に使用したコンピューティング時間に対してのみ発生します。

- Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。

コードリポジトリ

このパターンのコードは、GitHub 内の「aws-config-rules」リポジトリで利用できます。このリポジトリのカスタムルールは Python プログラミング言語の Lambda ルールです。このリポジトリには、AWS Config の多くのカスタムルールが含まれています。このパターンでは以下のルールのみが使用されます：

- REDSHIFT_AUDIT_ENABLED — Amazon Redshift クラスターで監査ログが有効になっていることを確認します。ユーザーアクティビティモニタリングが有効になっていることも確認する場合、代わりに REDSHIFT_USER_ACTIVITY_MONITORING_ENABLEDのルールをデプロイします。
- REDSHIFT_SSL_REQUIRED — Amazon Redshift クラスターに接続するには SSL が必要であることを確認します。連邦情報処理標準 (FIPS) の暗号が使用されていることも確認する場合、代わりに REDSHIFT_FIPS_REQUIRED ルールをデプロイします。
- REDSHIFT_FIPS_REQUIRED — SSL が必要で、FIPS 暗号が使用されていることを確認します。
- REDSHIFT_DB_ENCRYPTED — Amazon Redshift クラスター内のデータベースが暗号化されていることを確認します。
- REDSHIFT_USER_ACTIVITY_MONITORING_ENABLED — 監査ログ記録とユーザーアクティビティモニタリングが有効になっていることを確認します。

エピック

ルールを展開する準備をする

タスク	説明	必要なスキル
IAM ポリシーを設定	1. Lambda 実行ロールが Amazon Redshift クラスター設定を読み取れるようにする、カスタム IAM アイデンティティベースのポリシーを作成します。詳細については、「リソースへのアクセスの管理」 (Amazon	AWS 管理者

タスク	説明	必要なスキル
	Redshift ドキュメント) および「IAM ポリシーの作成」(IAM ドキュメント) を参照してください。 `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["redshift:DescribeClusterParameterGroups", "redshift:DescribeClusterParameters", "redshift:DescribeClusters", "redshift:DescribeClusterSecurityGroups", "redshift:DescribeClusterSnapshots", "redshift:DescribeClusterSubnetGroups", "redshift:DescribeEventSubscriptions", "redshift:DescribeLoggingStatus"], "Resource": "*" }]`	

タスク	説明	必要なスキル
	<pre>}</pre> 2. 「AWSLambdaExecute」と「awsConfigRulesExecutionRole」管理ポリシーを、「Lambda 実行ロール」のアクセス権限ポリシーとしてアサインします。手順については、「IAM アイデンティティアクセス権限の追加」(IAM ドキュメント)を参照してください。	
リポジトリをクローン作成します。	Bash シェルで、次のコマンドを実行します。これにより、GitHub から「aws-config-rules」リポジトリがクローンされます。 <pre>git clone https://github.com/awslabs/aws-config-rules.git</pre>	AWS 全般

AWS Config にルールをデプロイ

タスク	説明	必要なスキル
ルールを AWS Config にデプロイします。	「カスタム Lambda ルールの作成」(AWS Config ドキュメント)の手順に従って、アカウントに次の 1 つ以上のルールをデプロイします:	AWS 管理者

タスク	説明	必要なスキル
	• REDSHIFT_AUDIT_ENABLED • REDSHIFT_SSL_REQUIRED • REDSHIFT_FIPS_REQUIRED • REDSHIFT_DB_ENCRYPTED • REDSHIFT_USER_ACTIVITY_MONITORING_ENABLED	
ルールが機能していることを確認します。	ルールをデプロイした後、「リソースの評価」(AWS Config ドキュメント)の手順んに従って、AWS Config が Amazon Redshift リソースを正しく評価していることを確認します。	AWS 全般

関連リソース

AWS サービスのドキュメント

- 「Amazon Redshift のセキュリティ」(Amazon Redshift ドキュメント)
- 「データベースセキュリティの管理」(Amazon Redshift ドキュメント)
- 「AWS Config カスタムルール」(AWS Config ドキュメント)

AWS 規範ガイダンス

- 「新しい Amazon Redshift クラスターに必要な SSL エンドポイントを持つことを確認」
- 「Amazon Redshift クラスターが作成時に暗号化されていることを確保」

追加情報

AWS Config で次の AWS マネージドルールを使用して、Amazon Redshift の以下のセキュリティ構成を確認できます：

- 「redshift-cluster-configuration-check」 — このルールを使用して、Amazon Redshift クラスターのデータベースで監査ログが有効になっていることを確認し、データベースが暗号化されていることを確認します。
- 「redshift-require-tls-ssl」 — このルールを使用して、Amazon Redshift クラスターに接続するには SSL が必要であることを確認します。

Network Firewall を使用して、アウトバウンドトラフィックのサーバー名表示から DNS ドメイン名をキャプチャする

作成者: Kirankumar Chandrashekar (AWS)

概要

このパターンは、AWS Network Firewall を使用して、アウトバウンドネットワークトラフィックの HTTPS ヘッダーでサーバー名表示 (SNI) によって提供される DNS ドメイン名を収集する方法を示しています。Network Firewall は、Amazon Virtual Private Cloud (Amazon VPC) の重要なネットワーク保護を簡単にデプロイできるようにするマネージド型サービスです。これには、特定のセキュリティ要件を満たさないパケットをブロックするファイアウォールでアウトバウンドトラフィックを保護する機能が含まれます。特定の DNS ドメイン名へのアウトバウンドトラフィックを保護することをエグレスフィルタリングと呼びます。これは、あるネットワークから別のネットワークへのアウトバウンド情報の流れを監視し、場合によっては制限する手法です。

Network Firewall を通過する SNI データをキャプチャしたら、Amazon CloudWatch Logs と AWS Lambda を使用して、E メール通知を生成する Amazon Simple Notification Service (Amazon SNS) トピックにデータを公開できます。E メール通知には、サーバー名とその他の関連する SNI 情報が含まれます。さらに、このパターンの出力を使用して、ファイアウォールルールを使用して SNI 内のドメイン名によるアウトバウンドトラフィックを許可または制限できます。詳細については、Network Firewall ドキュメントの「AWS Network Firewall のステートフルルールグループの操作」をご参照ください。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- Linux、macOS、または Windows にインストールおよび設定された AWS コマンドラインインターフェイス (AWS CLI) バージョン 2。
- Network Firewall は、Amazon VPC で設定および設定され、アウトバウンドトラフィックの検査に使用されます。Network Firewall は、次のいずれかの VPC 設定を使用するように設定できます。
  - 「インターネットゲートウェイを備えたシンプルなシングルゾーンアーキテクチャ」
  - 「インターネットゲートウェイを備えたマルチのゾーンアーキテクチャ」
  - 「インターネットゲートウェイと NAT ゲートウェイを使用するアーキテクチャ」

アーキテクチャ

次の図は、ネットワークファイアウォールを使用してアウトバウンドネットワークトラフィックから SNI データを収集し、CloudWatch Logs と Lambda を使用してそのデータを SNS トピックに公開する方法を示しています。

この図表は、次のワークフローを示しています：

1. Network Firewall は、アウトバウンドネットワークトラフィックの HTTPS ヘッダーにある SNI データからドメイン名を収集します。
2. CloudWatch Logs は SNI データを監視し、アウトバウンドネットワークトラフィックが Network Firewall を通過するたびに Lambda 関数を呼び出します。
3. Lambda 関数は CloudWatch Logs によってキャプチャされた SNI データを読み取り、そのデータを SNS トピックに公開します。
4. SNS トピックから SNI データを含むメール通知が送信されます。

自動化とスケール

- 「AWS CloudFormation」を使用して、「インフラストラクチャをコードとして」使用することでこのパターンを作成できます。

テクノロジースタック

- Amazon CloudWatch Logs
- Amazon SNS
- Amazon VPC
- AWS Lambda
- AWS Network Firewall

ツール

AWS サービス

- 「Amazon CloudWatch Logs」— Amazon CloudWatch Logs を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、AWS CloudTrail、Amazon Route 53 およびその他のソースからログファイルを監視、保存、およびアクセスすることができます。
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバー (または生産者から消費者) へのメッセージ配信を提供するマネージドサービスです。
- 「Amazon VPC」— Amazon Virtual Private Cloud (Amazon VPC) では、AWS クラウドの論理的に隔離されたセクションをプロビジョニングすることで、ユーザーが定義した仮想ネットワーク内で AWS リソースを起動できます。仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークによく似ていますが、AWS のスケーラブルなインフラストラクチャを使用できるというメリットがあります。
- 「AWS Lambda」— AWS Lambda はサーバーをプロビジョニングまたは管理しなくてもコードを実行できるコンピュートサービスです。
- 「AWS Network Firewall」— AWS Network Firewall は、すべてのAmazon VPC に不可欠なネットワーク保護を簡単に導入できる管理サービスです。

エピック

Network Firewall の CloudWatch Logs ロググループの作成

タスク	説明	必要なスキル
CloudWatch Logs ロググループを作成します。	1. AWS マネジメントコンソールにサインインし、「CloudWatch コンソール」を開きます。 2. ナビゲーションペインで、[ロググループ] を選択します。 3. [Actions (アクション)] を選択し、[Create log group (ロググループの作成)] を選択します。 4. ロググループの名前を入力し、[Create log group (ログ	クラウド管理者

タスク	説明	必要なスキル
	グループの作成)] を選択します。 詳細については、CloudWatch Logs ドキュメントの「ロググループとログストリームの操作」を参照してください。	

SNS トピックおよびサブスクリプションの作成

タスク	説明	必要なスキル
SNS トピックを作成します。	SNS トピックを作成するには、「Amazon SNS」ドキュメントの指示に従ってください。	クラウド管理者
SNS トピックにエンドポイントを登録します。	作成した SNS トピックのエンドポイントとしてメールアドレスを登録するには、「Amazon SNS ドキュメント」の指示に従います。[プロトコル] には「Email/Email-JSON」を選択します。 **Note** また、要件に基づいて別のエンドポイントを選択することもできます。	クラウド管理者

Network Firewall へのロギングをセットアップする

タスク	説明	必要なスキル
ファイアウォールロギングを有効にする。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで、[ネットワークファイアウォール] の下にある [ファイアウォール] を選択します。 3. 「ファイアウォール」セクションで、送信トラフィック用に SNI からサーバー名を取得するファイアウォールを選択します。 4. 「ファイアウォールの詳細」タブを選択し、「ログ」セクションで「編集」を選択します。 5. [ログタイプ] には [アラート] を選択します。[アラートのログ先] で、[CloudWatch ロググループ] を選択します。 6. CloudWatch ロググループでは、前に作成したロググループを検索して選択し、[保存] を選択します。 CloudWatch Logs をネットワークファイアウォールのログ送信先として使	クラウド管理者

タスク	説明	必要なスキル
	用する方法の詳細については、Network Firewall ドキュメントの「Amazon CloudWatch Logs」を参照してください。	

Network Firewall でステートフルルールを設定する

タスク	説明	必要なスキル
ステートフルルールグループの作成	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで、[ネットワークファイアウォール] で、[ネットワークファイアウォールルールグループ] を選択します。 3. [Network Firewall ルールグループの作成] を選択します。 4. [Network Firewall ルールグループの作成] ページで、[ルールグループタイプ] に [ステートフルルールグループ] を選択します。注: 詳細については、「AWS Network Firewall のステートフルルールグループの操作」を参照してください。 5. 「ステートフルルールグループ」セクションで、	クラウド管理者

タスク	説明	必要なスキル
	ルールグループの名前と説明を入力します。 6. [容量] には、ステートフルルールグループに許可する最大容量 (最大 30,000) を設定します。注:ルールグループを作成した後は、この設定を変更できません。容量の計算方法については、「AWS Network Firewall でのルールグループの容量の設定」を参照してください。最大設定については、「AWS Network Firewall クォータ」を参照してください。 7. [ステートフルルールグループのオプション] では、[5-tuple] を選択します。 8. 「ステートフルルール順序」セクションで、[デフォルト] を選択します。 9. 「ルール変数」セクションでは、デフォルト値をそのまま使用します。 10.「ルールを追加」セクションで、[プロトコル] に [TLS] を選択します。[ソース] には [任意] を選択します。[送信元ポート] には [任意のポート] を選択します。[送信先] には [任意] を選択します。[送信先ポート] で [任意のポート] を選択しま	

タスク	説明	必要なスキル
	す。[トラフィック方向] には [転送] を選択します。[アクション] で、[アラート] を選択します。[Add rule] を選択してください。 11[ステートフルルールグループの作成] を選択します。	
ステートフルルールをNetwork Firewall に関連付けます。	1. AWS マネジメントコンソールにサインインし、「Amazon VPC コンソール」 を開きます。 2. ナビゲーションペインで、[ネットワークファイアウォール] の下にある [ファイアウォール] を選択します。 3. アウトバウンドトラフィック用に SNI からサーバー名を取得するファイアウォールを選択します。 4. 「ステートフルルールグループ」セクションで [アクション] を選択し、[アンマネージドステートフルルールグループの追加] を選択します。 5. [アンマネージドステートフルルールグループの追加] ページで、先ほど作成したステートフルルールグループを選択し、[ステートフルルールグループの追加] を選択します。	クラウド管理者

ログを読み取る Lambda 関数を作成します。

タスク	説明	必要なスキル

Lambda 関数のコードを作成します。	アウトバウンドトラフィックのNetwork Firewall から CloudWatch Logs イベントを読み取ることができる統合開発環境 (IDE) で、次の Python 3 コードを貼り付け、<SNS-topic-ARN> を指定した値に置き換えます。
	
	`import json`
	`import gzip`
	`import base64`
	`import boto3`
	`sns_client = boto3.client('sns')`
	`def lambda_handler(event, context):`
	`  decoded_event = json.loads(gzip.decompress(base64.b64decode(event['awslogs']['data'])))`
	`  body = '''`
	`  {filtermatch}`
	`  '''.format(`
	`  loggroup=decoded_event['logGroup'],`
	`  logstream=decoded_event['logStream'],`
	`  filtermatch=decoded_event['logEvents'][0]['message'],`
	`  )`
	`  print(body)`
		アプリ開発者

タスク	説明	必要なスキル

	filterMatch = json.loads(body)
	data = []
	if 'http' in filterMatch['event']:
	data.append(filterMatch['event']['http']['hostname'])
	elif 'tls' in filterMatch['event']:
	data.append(filterMatch['event']['tls']['sni'])
	result = 'Domain accessed ' + 1*' ' + (data[0]) + 1*' ' 'via AWS Network Firewall ' + 1*' ' + (filterMatch['firewall_name'])
	print(result)
	message = {'ServerName': result}
	send_to_sns = sns_client.publish(
	TargetArn=<SNS-topic-ARN>,
	#Replace with the SNS topic ARN
	Message=json.dumps({'default': json.dumps(message),
	
	'sms': json.dumps(message),
	
	'email': json.dumps(message)}),
	Subject='Server Name passed through the Network Firewall',	

タスク	説明	必要なスキル

	<pre>MessageSt
	ructure='json'
	)</pre>
	このコードサンプルは、CloudWatch Logs のコンテンツを解析し、SNI によって提供されたサーバー名を HTTPS ヘッダーにキャプチャします。	

Lambda 関数を作成します。	Lambda 関数を作成するには、「Lambda ドキュメント」の指示に従い、[ランタイム] に [Python 3.9] を選択します。	クラウド管理者
Lambda 関数にコードを追加します。	前に作成した Lambda 関数に Python コードを追加するには、「Lambda」ドキュメントの指示に従ってください。	クラウド管理者

タスク	説明	必要なスキル
CloudWatch Logs をトリガーとして Lambda 関数に追加します。	1. AWS マネジメントコンソールにサインインし、「Lambda コンソール」を開きます。 2. ナビゲーションペインで [関数] を選択し、先ほど作成した関数を選択する。 3. [関数の概要] セクションで、[トリガーを追加] を選択します。 4. 「トリガーを追加」ページの「トリガー設定」セクションで、[Cloudwatch ログ] を選択し、[追加] を選択します。 5. [ロググループ] では、前に作成した CloudWatch ログロググループを選択します。 6. [Filter name] (フィルター名) で、フィルターの名前を入力します。 7. [追加] を選択します。 8. 関数ページの [設定] タブの「トリガー」セクションで、追加したトリガーを選択し、[有効化] を選択します。 詳細については、Lambda ドキュメントの「CloudWatch ログで Lambda の使用」を参照してください。	クラウド管理者

タスク	説明	必要なスキル

SNS 公開権限を追加します。	Lambda 実行ロールに [sns:publish] アクセス許可を追加して、Lambda が API コールを行って SNS にメッセージを発行できるようにします。
	
	1. 先ほど作成した Lambda 関数の「実行ロールを検索」します。
	2. AWS Identity and Access Management (IAM) ロールに「以下のポリシーを追加」します：
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Sid": "AllowSNSPublish",
	"Effect": "Allow",
	"Action": [
	"sns:GetTopicAttributes",
	"sns:Subscribe",
	"sns:Unsubscribe",
	"sns:Publish"
	],
	"Resource": "*"
	}
	]
		クラウド管理者

タスク	説明	必要なスキル
	<pre>}</pre>	

SNS 通知の機能をテストしてください。

タスク	説明	必要なスキル

Network Firewall 経由でトラフィックを送信します。	1. HTTPS トラフィックを送信するか、Network Firewall を通過するまで待ちます。
	2. トラフィックが Network Firewall を通過したときに AWS から受信する SNS 通知メールを確認してください。E メールには、アウトバウンドトラフィックの SNI の詳細が含まれています。たとえば、アクセスされたドメイン名が [https://aws.amazon.com] で、サブスクリプションプロトコルが [EMAIL-JSON] の場合、上記の Lambda コードから生成される E メールには次の内容が含まれます。
	<pre>{
	"Type": "Notification",
	"MessageId": "<messageID>",
	"TopicArn": "arn:aws:sns:us-west-2:123456789:testSNSTopic",</pre>
		テストエンジニア

タスク	説明	必要なスキル
	"Subject": "Server Name passed through the Network Firewall", "Message": "{\"ServerName\": \"Domain 'aws.amazon.com' accessed via AWS Network Firewall 'AWS-Network-Firewall-Multi-AZ-firewall\"}", "Timestamp": "2022-03-22T04:10:04.217Z", "SignatureVersion": "1", "Signature": "<Signature>", "SigningCertURL": "<SigningCertUrl>", "UnsubscribeURL": "<UnsubscribeURL>" } 次に、「Amazon CloudWatch ドキュメント」の指示に従って、Amazon CloudWatch の Network Firewall アラートログを確認します。アラートログには次のように出力されます。 { "firewall_name": "AWS-Network-Firewall-Multi-AZ-firewall", "availability_zone": "us-east-2b",	

タスク	説明	必要なスキル
	"event_timestamp": "<event timestamp>", "event": { "timestamp": "2021-03-22T04:10:04.214222+0000", "flow_id": <flow ID>, "event_type": "alert", "src_ip": "10.1.3.76", "src_port": 22761, "dest_ip": "99.86.59.73", "dest_port": 443, "proto": "TCP", "alert": { "action": "allowed", "signature_id": 2, "rev": 0, "signature": "", "category": "", "severity": 3 }, "tls": { "subject": "CN=aws.amazon.com", "issuerdn": "C=US, O=Amazon, OU=Server CA 1B, CN=Amazon", "serial": "<serial number>",	

タスク	説明	必要なスキル
	"fingerpr int": "<fingerprint ID>", "sni": "aws.amazon.com", "version": "TLS 1.2", "notbefor e": "2020-09-30T00:00: 00", "notafter ": "2021-09-23T12:00: 00", "ja3": {}, "ja3s": {} }, "app_proto": "tls" } }	

Terraform を使用すると、組織の Amazon GuardDuty が自動的に有効になります。

作成者：アーティ・ カンナン (AWS)

概要

Amazon GuardDuty は、お客様のAmazon Web Services (AWS) アカウントを継続的に監視し、脅威インテリジェンスを使用して、お客様の AWS 環境内での予期しないアクティビティや潜在的に悪意のあるアクティビティを特定します。複数のアカウントや組織、複数の AWS リージョン、または AWS マネジメントコンソールを使用して GuardDuty を手動で有効にするのは面倒な場合があります。Terraform などのinfrastructure as code (IaC ツールを使用すると、このプロセスを自動化できます。このツールでは、マルチアカウント、マルチリージョンのサービスとリソースをクラウドにプロビジョニングして管理できます。

AWS では、AWS Organizations を使用して GuardDuty で複数のアカウントを設定および管理することを推奨しています。このパターンは、その推奨事項に準拠しています。このアプローチの利点の 1 つは、新しいアカウントが作成されたり、組織に追加されたりしたときに、手動で操作しなくても、サポートされているすべての地域でこれらのアカウントで GuardDuty が自動的に有効になることです。

このパターンは、HashiCorp Terraform を使用して、組織内の 3 つ以上のAmazon Web Services (AWS) アカウントで Amazon GuardDuty を有効にする方法を示しています。このパターンで提供されるサンプルコードは以下の処理を行います。

- AWS Organizationsのターゲット組織の現在のメンバーであるすべての AWS アカウントで GuardDuty を有効にします。
- GuardDuty の「自動有効化機能」を有効にします。これにより、対象組織に今後追加されるすべてのアカウントで GuardDuty が自動的に有効になります。
- GuardDuty を有効にするリージョンを選択できます。
- 組織のセキュリティアカウントを GuardDuty 委任管理者として使用します
- ログ記録アカウントに Amazon Simple Storage Service (Amazon S3) バケットを作成し、このバケット内のすべてのアカウントから集約された検出結果を公開するように GuardDuty を設定します。
- デフォルトで 365 日後に S3 バケットから Amazon S3 Glacier Flexible Retrievalストレージに検出結果を移行するライフサイクルポリシーを割り当てます

このサンプルコードは、継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインに統合することができます。

ターゲットオーディエンス

このパターンは、Terraform、Python、GuardDuty、AWS Organizations を使った経験があるユーザーにおすすめです。

前提条件と制限

前提条件

- アクティブなAWS アカウント
- 組織は AWS Organizations で設定され、少なくとも次の 3 つのアカウントが含まれています。
  - 管理アカウント — スタンドアロンまたは CI/CD パイプラインの一部として Terraform コードをデプロイするアカウントです。Terraform の状態もこのアカウントに保存されます。
  - セキュリティアカウント — このアカウントは GuardDuty の委任された管理者として使用されます。詳細については、「GuardDuty の委任された管理者の重要な考慮」(GuardDuty ドキュメント) を参照してください。
  - ロギングアカウント — このアカウントには、GuardDuty がすべてのメンバーアカウントからの集計検出結果を公開する S3 バケットが含まれます。

  必要な設定で組織を設定する方法の詳細については、「アカウント構造を作成する」 (AWS Well-Architected Labs)を参照してください。
- Terraform の状態を管理アカウントに保存するためのリモートバックエンドとして機能する Amazon S3 バケットと Amazon DynamoDB テーブル。Terraform 状態にリモートバックエンドを使用する方法の詳細については、「S3 バックエンド」 (Terraform ドキュメント)を参照してください。S3 バックエンドでリモートステート管理をセットアップするコードサンプルについては、「remote-state-s3-backend」 (Terraform レジストリ) を参照してください。次の要件に注意してください。
  - キーおよび S3 バケットは同じリージョンにある必要があります。
  - DynamoDB テーブルを作成する場合、パーティションキーは `LockID` (大文字と小文字を区別)、パーティションキータイプは「文字列」でなければなりません。他の設定はすべてデフォルト値のままにしておきます。詳細については、「プライマリキーについて」と「テーブルを作成する」 (DynamoDB ドキュメント) を参照してください。
- GuardDuty が検出結果を公開する S3 バケットのアクセスログを保存するために使用される S3 バケット。詳細については、「Amazon S3 サーバーアクセスログを有効にします (Amazon S3 ド

キュメント)」を参照してください。AWS Control Tower のランディングゾーン にデプロイする場合は、「ログアーカイブ」アカウントの S3 バケットをこの目的で再利用できます。

- Terraform バージョン 0.14.6 以降がインストールされ、設定されている。詳細については、「AWSを開始する」（ Terraform ドキュメント ）を参照してください。
- Python、バージョン 3.9.6 以降がインストールされ、設定されています。詳細については、「ソースリリース (Python ウェブサイト)」を参照してください。
- AWS SDK for Python (Boto3) をインストールするには 詳細については、Boto3 ドキュメントを参照してください。
- jq がインストールされ、設定されている。詳細については、「jq のダウンロード (jq ドキュメント)」を参照してください。

制約事項

- このパターンは macOS と Amazon Linux 2 オペレーティングシステムをサポートします。このパターンは Windows オペレーティングシステムでの使用についてはテストされていません。

> **Note**
>
> Amazon Linux 2 のサポートは間もなく終了します。詳細については、「Amazon Linux 2 のFAQs」を参照してください。

- GuardDuty は、どのアカウントでも、どのターゲットリージョンでもまだ有効になっていない必要があります。
- このパターンの IaC ソリューションでは、前提条件は適用されません。
- このパターンは、以下のベストプラクティスに準拠する AWS Landing Zone 向けに設計されています。
  - ランディングゾーン は AWS Control Tower を使用して作成されました。
  - セキュリティと口ギングには別の AWS アカウントが使用されます。

製品バージョン

- Terraform バージョン 0.14.6 以降。サンプルコードはバージョン 1.2.8 でテストされています。
- Python バージョン 3.9.6 以降。

アーキテクチャ

このセクションでは、このソリューションの概要と、サンプルコードによって確立されたアーキテクチャについて説明します。次の図は、単一の AWS リージョン内の組織内のさまざまなアカウントにデプロイされたリソースを示しています。

1. Terraform は、セキュリティアカウントとロギングアカウントに「GuardDutyTerraformorGrole」AWS Identity およびAccess Management (IAM) ロールを作成します。
2. Terraform は、ロギングアカウントのデフォルトの AWS リージョンに S3 バケットを作成します。このバケットは、組織内のすべてのリージョンおよびすべてのアカウントからの GuardDuty の検出結果をすべて集約するための公開先として使用されます。また、Terraform は S3 バケット内の検出結果を暗号化するために使用するセキュリティアカウントに AWS Key Management Service (AWS KMS) キーを作成し、S3 バケットの検出結果を S3 Glacier フレキシブル検索ストレージに自動的にアーカイブするように設定します。
3. Terraform は、管理アカウントから、セキュリティアカウントを GuardDuty の委任された管理者として指定します。つまり、セキュリティアカウントは、管理アカウントを含むすべてのメンバーアカウントの GuardDuty サービスを管理するようになります。個々のメンバーアカウントは、自分で GuardDuty を一時停止または無効にすることはできません。
4. Terraform は、GuardDuty 委任管理者のために、セキュリティアカウントに GuardDuty ディテクターを作成します。
5. まだ有効になっていない場合、Terraform は GuardDuty で S3 保護を有効にします。詳細については、「Amazon GuardDuty ドキュメント (GuardDuty ドキュメント) の Amazon S3 Protection」を参照してください。
6. Terraform は、組織内の現在のアクティブなメンバーアカウントすべてを GuardDuty メンバーとして登録します。
7. Terraform は、すべてのメンバーアカウントから集約された検出結果をロギングアカウントの S3 バケットに公開するように GuardDuty 委任管理者を設定します。
8. Terraform は、選択した AWS リージョンごとにステップ 3 から 7 を繰り返します。

自動化とスケール

提供されるサンプルコードはモジュール化されているため、CI/CD パイプラインに統合してデプロイを自動化できます。

ツール

AWS サービス

- Amazon DynamoDB は、フルマネージド NoSQL データベースサービスです。高速かつ予測可能でスケーラブルなパフォーマンスを発揮します。
- Amazon GuardDuty は、AWS 環境内での不正または悪質である可能性がある予期しないアクティビティを特定するために役立つセキュリティ監視サービスです。
- 「AWS Identity and Access Management (IAM)」は、AWS リソースへのアクセスを安全に管理し、誰が認証され、使用する権限があるかを制御するのに役立ちます。
- AWS Key Management Service (AWS KMS) は、ユーザーのデータを保護するために使用される、暗号化キーの作成と制御を容易にするマネージドサービスです。
- 「AWS Organizations」は、複数の AWS アカウントを 1 つの組織に統合し、作成と一元管理するためのアカウント管理サービスです。
- Amazon Simple Storage Service (Amazon S3) は、どのようなデータ量であっても、データを保存、保護、取得することを支援するクラウドベースのオブジェクトストレージサービスです。
- AWS SDK for Python (Boto3) は、Python アプリケーション、ライブラリ、またはスクリプトを AWS のサービスと統合するのに役立つソフトウェア開発キットです。

その他のツールやサービス

- 「HashiCorp Terraform」は、コードを使用してクラウドインフラストラクチャとリソースをプロビジョニングおよび管理するのに役立つコマンドラインインターフェイスアプリケーションです。
- 「Python」は汎用プログラミング言語です。
- 「jq」は JSON ファイルの操作を支援するコマンドラインプロセッサです。

コードリポジトリ

このパターンのコードは GitHub 内の「amazon-guardduty-for-aws-organizations-with-terraform」リポジトリで利用できます。

エピック

組織内で GuardDuty を有効にする

タスク	説明	必要なスキル
リポジトリをクローン作成します。	Bash シェルで、次のコマンドを実行します。「追加情報」セクションの「リポジトリのクローン」では、GitHub リポジトリの URL を含むコマンド全体をコピーできます。これにより、GitHub の「amazon-guardduty-for-aws-Organizations-with-terraform」リポジトリのクローンが作成されます。 `git clone <github-repository-url>`	DevOps エンジニア
Terraform 設定ファイルを編集します。	1. 複製したリポジトリの`root`フォルダーに、以下のコマンドを実行して「configuration.json.sample」ファイルを複製します。 `cp configuration.json.sample configuration.json` 2. 新しい「configuration.json」ファイルを編集し、以下の各変数の値を定義します。 • `management_acc_id` — 管理アカウントのアカウント ID。	DevOpsエンジニア、AWS 全般、Terraform、Python

タスク	説明	必要なスキル
	• `delegated_admin_acc_id` — セキュリティアカウントのアカウント ID。 • `logging_acc_id` — ログアカウントのアカウント ID。 • `target_regions` — GuardDuty を有効化したい AWS リージョンのカンマ区切りリスト。 • `organization_id` — GuardDuty を有効にしている組織の AWS Organizations ID。 • `default_region` — Terraform の状態が管理アカウントに保存されているリージョン。これは、Terraform バックエンドに S3 バケットと DynamoDB テーブルをデプロイしたのと同じリージョンです。 • `role_to_assume_for_role_creation` — セキュリティアカウントとロギングアカウントの新しい IAM ロールに割り当てる名前。この新しいロールは次の記事で作成します。Terraform はこのロールを引き受け、	

タスク	説明	必要なスキル
	セキュリティアカウントとロギングアカウントに GuardDutyTerraformOrgRole IAM ロールを作成します。 • finding_publishing_frequency — GuardDuty が検出結果を S3 バケットに公開する頻度。 • guardduty_findings_bucket_region — 公開された検出結果に対して S3 バケットを作成する優先リージョン。 • logging_acc_s3_bucket_name — 公開された調査検出結果で使用する S3 バケットの推奨名。 • security_acc_kms_key_alias — GuardDuty の検出結果を暗号化するために使用されるキーの AWS KMS エイリアス。 • s3_access_log_bucket_name — GuardDuty の検出結果に使用される S3 バケットのアクセスログを収集する、既存の S3 バケットの名前。このバケットは、GuardDuty 検出結果バケットと同じ	

タスク	説明	必要なスキル
	AWS リージョンに存在する必要があります。 • `tfm_state_backend_s3_bucket` — Terraform リモートバックエンドの状態を保存する既存の S3 バケットの名前。 • `tfm_state_backend_dynamodb_table` — テラフォームの状態をロックするための既存の DynamoDB テーブルの名前。 3. 設定ファイルを保存して閉じます。	

タスク	説明	必要なスキル
新しい IAM ロール用の CloudFormation テンプレートを生成します。	このパターンには、2 つの CloudFormation テンプレートを作成する IaC ソリューションが含まれています。これらのテンプレートは、Terraform がセットアッププロセスで使用する 2 つの IAM ロールを作成します。これらのテンプレートは、「最小権限」というセキュリティのベストプラクティスに準拠しています。 1. Bash シェルのリポジトリ `root`フォルダで、`cfn-templates/` に移動します。このフォルダには、スタブ付きのCloudFormation テンプレートファイルが含まれています。 2. 以下のコマンドを実行してください。これにより、スタブは「configuration.json」ファイルに指定した値に置き換えられます。 `bash scripts/replace_config_stubs.sh` 3. 次の CloudFormation テンプレートが`cfn-templates/` フォルダーに作成されたことを確認します。 • management-account-role.yaml — このファイ	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
	ルには、管理アカウント内の IAM ロールのロール定義と関連するアクセス権限が含まれています。これらのロールには、このパターンを完了するために最低限必要な権限しかありません。 • role-to-assume-for-role-creation.yaml — このファイルには、セキュリティアカウントとロギングアカウントの IAM ロールのロール定義と関連する権限が含まれています。Terraform は、これらのアカウントに「GuardDuty TerraformorGrole」ロールを作成するためにこのロールを引き受けます。	

タスク	説明	必要なスキル
IAM ロールを作成します。	「スタックの作成」(CloudFormation ドキュメント) の手順に従って、次の操作を行います。 1. セキュリティアカウントとロギングアカウントの両方に、「role-to-assume-for-role-creation.yaml」スタックをデプロイします。 2. 「management-account-role.yaml」スタックを管理アカウントにデプロイします。スタックが正常に作成され、CREATE_COMPLETE スタックのステータスが出力に表示されたら、この新しいロールの Amazon リソースネーム (ARN) を書き留めておきます。	DevOps エンジニア、AWS 全般
管理アカウントで IAM ロールを割り当てます。	セキュリティのベストプラクティスとして、先に進む前に新しい「management-account-role」IAMロールを引き受けることを推奨します。AWS コマンドラインインターフェイス (AWS CLI) の「追加情報」セクションの「管理アカウントの IAM ロールを引き受ける」にコマンドを入力します。	DevOps エンジニア、AWS 全般

タスク	説明	必要なスキル
セットアップスクリプトを実行します。	リポジトリrootフォルダで、次のコマンドを実行してセットアップスクリプトを起動します。 `bash scripts/full-setup.sh` full-setup.shスクリプトは以下のアクションを実行します。 • すべての設定値を環境変数としてエクスポートします。 • 各 Terraform モジュールの「バックエンド.tf」コードファイルと「terraform.tfvars」コードファイルを生成します。 • AWS CLI を通じて、組織内の GuardDuty への信頼できるアクセスを可能にします。 • 組織の状態を Terraform の状態にインポートします。 • ログアカウントで検出結果を公開するための S3 バケットを作成します。 • セキュリティアカウントの検出結果を暗号化するための AWS KMS キーを作成します • 「アーキテクチャ」セクションで説明されているよ	DevOps エンジニア、Python

タスク	説明	必要なスキル
	うに、選択したすべてのリージョンで、組織全体で GuardDuty を有効にします。	

(オプション) 組織内の GuardDuty を無効にする

タスク	説明	必要なスキル
クリーンアップのスクリプトを実行します。	このパターンを使用して組織の GuardDuty を有効にし、GuardDuty を無効にする場合は、リポジトリ`root`フォルダーで次のコマンドを実行して 「cleanup-gd.sh」スクリプトを起動します。 `bash scripts/cleanup-gd.sh` このスクリプトは、ターゲット組織の GuardDuty を無効にし、デプロイされたリソースをすべて削除し、Terraform を使用して GuardDuty を有効にする前の状態に組織を復元します。 **Note** このスクリプトは、ローカルバックエンドとリモートバックエンドから Terraform 状態ファイルやロックファ	DevOpsエンジニア、AWS 全般、Terraform、Python

タスク	説明	必要なスキル
	イルを削除しません。必要な場合は、これらのアクションを手動で実行する必要があります。また、このスクリプトでは、インポートされた組織やその組織によって管理されているアカウントは削除されません。GuardDuty の信頼できるアクセスは、クリーンアップスクリプトの一部として無効になっていません。	
IAM ロールを削除します。	「role-to-assume-for-role-creation.yaml」と「management-account-role.yaml 」CloudFormation テンプレートで作成されたスタックを削除します。詳細については、「スタックの削除 (CloudFormation ドキュメント)」を参照してください。	DevOps エンジニア、AWS 全般

関連リソース

AWS ドキュメント

- 複数のアカウントの管理 (GuardDuty ドキュメント)
- 最小特権の付与 (IAM ドキュメント)

AWS マーケティング

- Amazon GuardDuty
- AWS Organizations

その他のリソース

- Terraform
- テラフォーム CLI ドキュメンテーション

追加情報

リポジトリをクローン

次のコマンドを実行して、Github リポジトリのクローンを作成します。

```
git clone https://github.com/aws-samples/amazon-guardduty-for-aws-organizations-with-terraform
```

管理アカウントの IAM ロールを引き受けます。

管理アカウントで IAM ロールを割り当てます。<IAM role ARN> は IAM ロールの名前に置き換えます。

```
export ROLE_CREDENTIALS=$(aws sts assume-role --role-arn <IAM role ARN> --role-session-name AWSCLI-Session --output json)
export AWS_ACCESS_KEY_ID=$(echo $ROLE_CREDENTIALS | jq .Credentials.AccessKeyId | sed 's/"//g')
export AWS_SECRET_ACCESS_KEY=$(echo $ROLE_CREDENTIALS | jq .Credentials.SecretAccessKey | sed 's/"//g')
export AWS_SESSION_TOKEN=$(echo $ROLE_CREDENTIALS | jq .Credentials.SessionToken | sed 's/"//g')
```

を使用して PCI DSS 4.0 の運用上のベストプラクティスを検証する AWS Config

作成者: Tala Qraitem (AWS) と Alex Goff (AWS)

概要

Payment Card Industry Data Security Standard (PCI DSS) は、支払いデータを保護するために不可欠な技術的および運用的なプロトコルを概説しています。PCI DSS は、支払いカードアカウントのデータセキュリティを促進および強化するために開発されました。また、一貫したセキュリティ対策をグローバルに採用することが容易になります。支払いカードアカウントデータがある環境向けに特別に設計されていますが、PCI DSS を使用して脅威から保護し、支払いエコシステム内の他の要素を保護することができます。

PCI DSS バージョン 4.0 は、進化する要件に対処し、明確化や追加のガイダンスを提供し、標準の構造と形式を改善するためにリリースされました。変更の詳細については、「PCI DSS バージョン 3.2.1 から 4.0 への変更の概要」を参照してください。

AWS Config コンフォーマンスパックは、セキュリティ、運用、またはコスト最適化のガバナンスチェックを作成するのに役立つ AWS Config ルールと修復アクションのコレクションです。コンフォーマンスパックは、 AWS アカウント および に単一のエンティティとしてデプロイすることも AWS リージョン、 の組織全体にデプロイすることもできます AWS Organizations。

PCI DSS バージョン 4.0 のコンフォーマンスパックは、バージョン 3.2.1 のコンフォーマンスパックを補強して構築します。コンフォーマンスパックのルールは、標準のルールにマッピングされます。詳細については、添付ファイルセクションで提供されているマッピングを参照してください。このコンフォーマンスパックの 2 つのバージョンから選択できます。1 つはグローバルリソースタイプを含み、もう 1 つはそれらを除外します。

> **Important**
>
> コンフォーマンスパックは、特定のガバナンスまたはコンプライアンス標準への準拠を完全に保証するように設計されていません。ユーザーは、使用が適用可能な法的および規制上の要件を満たしているかどうかについて、独自の評価を行う責任があります。

前提条件と制限

前提条件

- をアクティブにします AWS アカウント。
- セットアップします AWS Config。
- コンフォーマンスパックの前提条件を満たします。
- PCI DSS バージョン 3.2.1 コンフォーマンスパックをデプロイします。
- コンフォーマンスパックにアクセスして AWS Config 管理するアクセス許可を持っている。ポリシーの例については、このパターンの???「追加情報」セクションを参照してください。

制約事項

- にはデフォルトのクォータ AWS アカウント があり、以前は制限と呼ばれていました AWS のサービス。特に明記していない限り、クォータはリージョン固有です。一部のクォータの引き上げをリクエストできますが、すべてのクォータを増やすことはできません。単一アカウントコンフォーマンスパックや組織コンフォーマンスパックの制限など、AWS Config サービスの制限に精通していることを確認してください。
- グローバルリソースタイプを含むこのコンフォーマンスパックのバージョンは、 `us-east-1`リージョンでのみデプロイすることを目的としています。
- グローバルリソースタイプを除外するこのコンフォーマンスパックのバージョンは、次のリージョンでのみデプロイすることを目的としています。
  - `ap-east-1`
  - `ap-south-1`
  - `ap-northeast-2`
  - `ap-southeast-1`
  - `ap-southeast-2`
  - `ap-northeast-1`
  - `ca-central-1`
  - `eu-central-1`
  - `eu-west-1`
  - `eu-west-2`
  - `eu-west-3`
  - `eu-north-1`
  - `sa-east-1`
  - `us-east-2`

- us-west-1
- us-west-2

ツール

AWS のサービス

- AWS Config は、 のリソースの詳細ビュー AWS アカウント と、その設定方法を提供します。リソースがどのように相互に関連しているか、またそれらの構成が時間の経過とともにどのように変化したかを特定するのに役立ちます。
- 「AWS Systems Manager」は、 AWS クラウドで実行されるアプリケーションとインフラストラクチャの管理に役立ちます。アプリケーションとリソースの管理を簡素化し、運用上の問題を検出して解決する時間を短縮し、 AWS リソースを大規模に安全に管理できるようにします。

コードリポジトリ

コンフォーマンスパックは、コンAWS Config フォーマンスパック GitHub リポジトリにあります。このリポジトリには、PCI DSS バージョン 4.0 に関連する次のテンプレートが含まれています。

- Operational-Best-Practices-for-PCI-DSS-v4 including-global-resourcetypes
- Operational-Best-Practices-for-PCI-DSS-v4 excluding-global-resourcetypes

エピック

コンフォーマンスパックのデプロイと管理

タスク	説明	必要なスキル
コンフォーマンスパックをダウンロードします。	コンフォーマンスパックを us-east-1 リージョンにデプロイする場合は、Operational-Best-Practices-for-PCI-DSS-v4.0-including-global-resourcetypes.yaml テンプレートをダウンロードします。	DevOps エンジニア

タスク	説明	必要なスキル
	コンフォーマンスパックを別のリージョンにデプロイする場合は、Operational-Best-Practices-for-PCI-DSS-v4.0-excluding-global-resourcetypes.yaml テンプレートをダウンロードします。	
(オプション) コンフォーマンスパックを変更します。	組織の固有のニーズに合わせてコンフォーマンスパックテンプレートを変更できます。たとえば、カスタム修復アクションを作成できます。テンプレートを作成および変更する方法の詳細については、 AWS Config ドキュメントの「カスタムコンフォーマンスパックのテンプレートの作成」を参照してください。	AWS 全般

タスク	説明	必要なスキル
適合パックをデプロイします。	ターゲット AWS アカウント または にデプロイする場合は AWS リージョン、AWS Config ドキュメントの「適合パックのデプロイ」の手順に従います。 AWS Management Console または AWS Command Line Interface () を使用できますAWS CLI。 で組織全体にコンフォーマンスパックをデプロイする場合は AWS Organizations、 AWS Systems Manager ドキュメントの「高速セットアップを使用してコン AWS Config フォーマンスパックをデプロイする」の手順に従います。	AWS 全般
(オプション) コンフォーマンスパックを編集します。	コンフォーマンスパックを編集する場合は、 AWS Config ドキュメントの「コンフォーマンスパックの編集」の手順に従います。 AWS Management Console または を使用できます AWS CLI。	AWS 全般
(オプション) コンフォーマンスパックを削除します。	コンフォーマンスパックを削除する場合は、 AWS Config ドキュメントの「コンフォーマンスパックの削除」の手順に従います。 AWS Management Console または を使用できます AWS CLI。	AWS 全般

関連リソース

AWS リソース

- (ドキュメント) のコンフォーマンスパック AWS ConfigAWS Config
- 高速セットアップを使用してコン AWS Config フォーマンスパックをデプロイする (Systems Manager ドキュメント)
- での PCI DSS コンプライアンス AWS (AWS ウェブサイト)
- での PCI DSS バージョン 4.0 AWS (コンプライアンスガイド)

PCI DSS リソース

- PCI DSS バージョン 4.0 Resource Hub
- PCI Security Standards Council ドキュメントライブラリ
- PCI DSS バージョン 3.2.1 から 4.0 への変更の概要

追加情報

以下は、コンフォーマンスパックへのアクセス AWS Config と管理をユーザーに許可するサンプル AWS Identity and Access Management (IAM) ポリシーです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "config:PutConfigRule",
                "config:PutConformancePack",
                "config:DeleteConfigRule",
                "config:DeleteRemediationConfiguration",
                "config:DeleteConformancePack",
                "config:PutRemediationConfigurations",
                "config:BatchGetAggregateResourceConfig",
                "config:BatchGetResourceConfig",
                "config:Get*",
                "config:Describe*",
                "config:Deliver*",
                "config:List*",
```

```
                "config:Select*"
            ],
            "Resource": "*"
        }
    ]
}
```

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

新しい Amazon Redshift クラスターに必要な SSL エンドポイントがあることを確認する

作成者: Priyanka Chaudhary (AWS)

概要

このパターンは、セキュアソケットレイヤー (SSL) エンドポイントなしで新しい Amazon Redshift クラスターが起動されたときに自動的に通知するAmazon Web Services (AWS) CloudFormation テンプレートを提供します。

Amazon Redshift は、 フルマネージド型、ペタバイト規模のクラウドベース型データウェアハウスサービスです。大規模なデータセットの保存と分析を目的として設計されています。また、大規模なデータベース移行にも使用されます。セキュリティ上の理由から、Amazon Redshift はユーザーの SQL Server クライアントアプリケーションと Amazon Redshift クラスター間の接続を暗号化する SSL をサポートしています。クラスターが SSL 接続を要求するように設定するには、リリースの間クラスターに関連付けられているパラメータグループで `require_SSL` パラメータを `true` に設定します。

このパターンで提供されるセキュリティコントロールは、AWS CloudTrail ログ内の Amazon Redshift API コールを監視し、「クラスターの作成」、「クラスターの変更」、「クラスタースナップショットからリストアする」、「クラスターパラメータグループの作成」、および 「クラスターパラメータグループの変更」API の Amazon CloudWatch Eventsを開始します。イベントがこれらの API のいずれかを検出すると、Python スクリプトを実行する AWS Lambda を呼び出します。Python 関数は、リストされている CloudTrail イベントの CloudWatch イベントを分析します。Amazon Redshift クラスターが既存のスナップショットから作成、変更、または復元されるとき、クラスターの新しいパラメータグループが作成されるとき、または既存のパラメータグループが変更されると、関数はクラスターの`require_SSL`パラメータをチェックします。パラメータ値が`false`の場合、関数は Amazon Redshift クラスター名、AWS リージョン、AWS アカウント、およびこの通知の送信元である Lambda の Amazon リソースネーム (ARN) などの関連情報を含む Amazon Simple Notification Service (Amazon SNS) 通知をユーザーに送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。

- クラスターサブネットグループと関連するセキュリティグループを持つ仮想プライベートクラウド (VPC)

機能制限

- このセキュリティコントロールは地域ごとに行われます。監視する AWS リージョンごとにデプロイする必要があります。

アーキテクチャ

ターゲットアーキテクチャ

自動化とスケール

- AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、モニタリングする複数のアカウントにこのテンプレートをデプロイできます。

ツール

AWS サービス

- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。
- Amazon Redshift - Amazon Redshift は、クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、オブジェクトストレージサービスです。Simple Storage Service (Amazon S3) を使用すると、いつでもウェブ上の任意の場所から任意の量のデータを保存および取得できます。

- Amazon SNS — Amazon Simple Notification Service (Amazon SNS) は、ウェブサーバーやメールアドレスなど、パブリッシャーとクライアント間のメッセージ配信または送信を調整して管理します。サブスクライバーは、サブスクライブしているトピックに対して発行されたすべてのメッセージを受信します。また、同じトピックのサブスクライバーはすべて同じメッセージを受信します。

コード

このパターンには、以下の添付ファイルが含まれます。

- `RedshiftSSLEndpointsRequired.zip` — セキュリティコントロール用の Lambda コード。
- `RedshiftSSLEndpointsRequired.yml` — イベントとLambda 関数をセットアップする CloudFormation テンプレート。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを定義します。	「Amazon S3 コンソールで」、Lambda コードの.zip ファイルをホストする S3 バケットを選択または作成します。この S3 バケットが、監視したい Amazon Redshift クラスターと同じ AWS リージョンに存在する必要があります。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケット名には、先頭にスラッシュを含めることはできません。	クラウドアーキテクト
Lambda コードをアップロードします。	「添付ファイル」セクションにある Lambda コードの.zip	クラウドアーキテクト

タスク	説明	必要なスキル
	ファイルを S3 バケットにアップロードします。	

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
AWS CloudFormation のテンプレートを起動します。	S3 バケットと同じ「AWS リージョンで AWS CloudFormation コンソール」を開き、添付のテンプレート `RedshiftSSLEndpointsRequired.yml` をデプロイします。AWS CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントの「AWS CloudFormation コンソールでのスタックの作成」を参照してください。	クラウドアーキテクト
テンプレートのパラメータを入力します。	テンプレートを起動すると、次の情報の入力を求められます。 • S3 バケット: 最初のエピックで作成または選択したバケットを指定します。ここで添付の Lambda コード (.zip ファイル) をアップロードしました。 • S3 キー: S3 バケット内の Lambda .zip ファイルの場所を指定します (たとえば、<ファイル名>.zip また	クラウドアーキテクト

タスク	説明	必要なスキル
	は <コントロール/ファイル名>.zip)。先頭にスラッシュを含めないでください。 • 通知 E メール：Amazon SNS 通知を受け取る E メールアドレスを入力します。 • Lamba ロギングレベル：Lambda 関数のロギングレベルと頻度を指定します。Info を使用して、を使用して、進行状況に関する詳細な情報メッセージをログに記録し、引き続きデプロイを続行できるエラーイベントにはエラー 、潜在的に有害な状況の場合は警告を使用します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定したメールアドレスにサブスクリプションメールが送信されます。違反通知の受信を開始するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- S3 バケットの作成 (Amazon S3 ドキュメント)
- ファイルを S3 バケットにアップロードする (Amazon S3 ドキュメント)
- AWS CloudFormation コンソールでのスタックの作成 (AWS CloudFormation ドキュメント)
- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成 (AWS CloudTrailドキュメント)
- Amazon Redshift クラスターの作成 (Amazon Redshift ドキュメント)
- 接続用のセキュリティオプションの設定 (Amazon Redshift ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

新しい Amazon Redshift クラスターが VPC で起動することを検証

作成者: Priyanka Chaudhary (AWS)

概要

このパターンでは、Amazon Redshift クラスターが仮想プライベートクラウド (VPC) の外部で起動されると自動的に通知するAmazon Web Services (AWS) CloudFormation テンプレートを提供します。

Amazon Redshift は、フルマネージド型、ペタバイト規模のクラウドベースのデータウェアハウス製品です。大規模なデータセットの保存と分析を目的として設計されています。また、大規模なデータベース移行にも使用されます。Amazon Virtual Private Cloud (Amazon VPC) では、AWS クラウドの論理的に隔離されたセクションをプロビジョニングすることで、ユーザーが定義した仮想ネットワーク内でAmazon Redshiftなど AWS リソースを起動できます。

このパターンで提供されるセキュリティコントロールは、AWS CloudTrail ログの Amazon Redshift API 呼び出しを監視し、CreateCluster API と RestoreFromClusterSnapshot API の Amazon CloudWatch Events イベントを開始します。イベントがこれらの API のいずれかを検出すると、Python スクリプトを実行する AWS Lambda を呼び出します。Python 関数はCloudWatch イベントを分析します。Amazon Redshift クラスターがスナップショットから作成または復元され、Amazon VPC ネットワークの外部に表示された場合、この関数は、関連情報 (Amazon Redshift クラスター名、AWS リージョン、AWS アカウント、およびこの通知の送信元である Lambda の Amazon リソースネーム (ARN)) を含む Amazon Simple Notification Service (Amazon SNS) 通知をユーザーに送信します。

前提条件と制限

前提条件

- アクティブな AWS アカウント。
- クラスターサブネットグループと関連するセキュリティグループを含む VPC。

機能制限

- AWS CloudFormation テンプレートに、CreateCluster と RestoreFromClusterSnapshot アクション (新しいクラスター) のみが適用されます。VPC の外部で作成された既存の Amazon Redshift クラスターが検出されません。

- このセキュリティコントロールは地域ごとに行われます。監視する AWS リージョンごとにデプロイする必要があります。

アーキテクチャ

ターゲット アーキテクチャ

自動化とスケール

AWS Organizations を使用している場合は、AWS Cloudformation StackSets を使用して、モニタリングする複数のアカウントにこのテンプレートをデプロイできます。

ツール

AWS サービス

- 「AWS CloudFormation」 を使用することで、AWS リソースをセットアップし、迅速かつ一貫したプロビジョニングを行い、AWS アカウントとリージョン全体でライフサイクルの最初から最後までリソースを管理できます。リソースを個別に管理する代わりに、テンプレートを使用してリソースとその依存関係を記述し、それらをスタックとしてまとめて起動して設定できます。
- AWS CloudTrail – AWS CloudTrailは、AWS アカウントのガバナンス、コンプライアンス、および運用とリスクの監査を行えるように支援する AWS のサービスです。ユーザー、ロール、または AWS のサービスによって実行されたアクションは、CloudTrail にイベントとして記録されます。
- Amazon CloudWatch Events – Amazon CloudWatch Events は、AWS リソースでの変更を説明するシステムイベントのほぼリアルタイムのストリームを提供します。
- 「AWS Lambda」 – AWS Lambda はサーバーのプロビジョニングや管理を行わずにコードの実行を支援できるコンピューティングサービスです。AWS Lambda は必要に応じてコードを実行し、1 日あたり数個のリクエストから 1 秒あたり数千のリクエストまで自動的にスケーリングします。
- Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。ビジネスと顧客のために新しい洞察を得る目的でデータを使用できるために、Amazon Redshift がデータレイクと統合しています。
- Amazon S3 — Amazon Simple Storage Service (Amazon S3) は、拡張性の高いオブジェクトストレージサービスで、ウェブサイト、モバイルアプリケーション、バックアップ、データレイクなど、幅広いストレージソリューションに使用できます。

- Amazon SNS — Amazon SSimple Notification Service (Amazon SNS) は、ウェブサーバーや E メールアドレスなど、パブリッシャーとクライアント間のメッセージ配信や送信を調整および管理します。

コード

このパターンには、以下の添付ファイルが含まれます。

- `RedshiftMustBeInVPC.zip` — セキュリティコントロール用の Lambda コード。
- `RedshiftMustBeInVPC.yml` — イベントとLambda 関数をセットアップする CloudFormation テンプレート。

これらのファイルを使用するには、次のセクションの指示に従います。

エピック

S3 バケットのセットアップ

タスク	説明	必要なスキル
S3 バケットを定義します。	Amazon S3 コンソール上で、Lambda コードの.zip ファイルをホストする S3 バケットを選択または作成します。このS3 バケットが、Amazon Redshift クラスターと同じ AWS リージョンに存在する必要があります。S3 バケット名はグローバルに一意であり、名前空間はすべての AWS アカウントによって共有されています。S3 バケット名には、先頭にスラッシュを含めることはできません。	クラウドアーキテクト
Lambda コードをアップロードする	添付ファイルセクションで提供された Lambda コード	クラウドアーキテクト

タスク	説明	必要なスキル
	(RedshiftMustBeInVPC.zip ファイル) を S3 バケットにアップロードします。	

CloudFormation のテンプレートを入手するには

タスク	説明	必要なスキル
CloudFormation のテンプレートを起動します。	S3 バケットと同じAWS リージョンで AWS CloudFormation コンソール を開き、添付されたテンプレート (RedshiftMustBeInVPC.yml) をデプロイします。AWS CloudFormation テンプレートのデプロイに関する詳細については、CloudFormation ドキュメントのAWS CloudFormation コンソールでのスタックの作成 を参照してください。	クラウドアーキテクト
テンプレートのパラメータを入力します。	テンプレートを起動すると、次の情報の入力を求められます。 • S3 バケット: 最初のエピックで作成または選択したバケットを指定します。ここで添付の Lambda コード (.zip ファイル) をアップロードしました。 • S3 キー: S3 バケット内の Lambda .zip ファイルの	クラウドアーキテクト

タスク	説明	必要なスキル
	場所を指定します (たとえば、<ファイル名>.zip または <コントロール/ファイル名>.zip)。先頭にスラッシュを含めないでください。 • 通知 E メール：Amazon SNS 通知を受け取る E メールアドレスを入力します。 • Lamba ロギングレベル：Lambda 関数のロギングレベルと頻度を指定します。Info を使用して、を使用して、進行状況に関する詳細な情報メッセージをログに記録し、引き続きデプロイを続行できるエラーイベントにはエラー 、潜在的に有害な状況の場合は警告を使用します。	

サブスクリプションを確認

タスク	説明	必要なスキル
サブスクリプションを確認します。	CloudFormation テンプレートが正常にデプロイされると、指定したメールアドレスにサブスクリプションメールが送信されます。違反通知の受信を開始するには、このメールサブスクリプションを確認する必要があります。	クラウドアーキテクト

関連リソース

- S3 バケットの作成 (Amazon S3 ドキュメント)
- ファイルを S3 バケットにアップロードする (Amazon S3 ドキュメント)
- AWS CloudFormation コンソールでのスタックの作成 (AWS CloudFormation ドキュメント)
- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成 (AWS CloudTrailドキュメント)
- Amazon Redshift クラスターの作成 (Amazon Redshift ドキュメント)

添付ファイル

このドキュメントに関連する追加コンテンツにアクセスするには、次のファイルを解凍してください。「attachment.zip」

その他のパターン

- Session Manager と Amazon EC2 Instance Connect により踏み台ホストにアクセス
- AWS Fargate、AWS PrivateLink、Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします。
- AWS PrivateLink と Network Load Balancer を使用して、Amazon ECS 上のコンテナ アプリケーションにプライベートにアクセスします
- AWS PrivateLink と Network Load Balancer を使用して、Amazon EKS 上のコンテナアプリケーションにプライベートアクセスします。
- EC2 インスタンスを AMS アカウントの S3 バケットへの書き込みアクセスを許可
- ある の AWS CodeCommit リポジトリ AWS アカウント を別のアカウントの Amazon SageMaker AI Studio Classic に関連付ける
- Amazon Cognito と AWS Amplify UI を使用して既存の React アプリケーションユーザーを認証する
- AWS Systems Manager を使用して Windows レジストリエントリの追加または更新を自動化する
- AWS CloudFormation のテンプレートを使用して、AWS Glue での暗号化の適用を自動化
- Cloud Custodian と AWS CDK を使用して、Systems Manager の AWS マネージドポリシーを EC2 インスタンスプロファイルに自動的にアタッチする
- 既存および新規の Amazon EBS ボリュームを自動的に暗号化する
- クラウドカストディアンを使用して Amazon RDS へのパブリックアクセスをブロック
- AWS Managed Microsoft AD およびオンプレミスの Microsoft Active Directory を使用して DNS 解決を一元化する
- AWS インフラストラクチャをデプロイする前に、一元化されたカスタム Checkov スキャンを実装してポリシーを適用する
- AWS CDK アプリケーションまたは CloudFormation テンプレートのベストプラクティスを cdk-nag ルールパックを使用して確認する
- 起動時に EC2 インスタンスに必須タグが欠けていないか確認する
- Amazon DynamoDB へのクロスアカウントアクセスを設定する
- Application Load Balancer を使用して Oracle WebLogic 上の Oracle JD Edwards EnterpriseOne の HTTPS 暗号化を設定します
- AWS IoT 環境内のセキュリティイベントのログ記録とモニタリングを設定する
- Amazon EKS で実行されているアプリケーションの相互 TLS 認証を設定する

- pgAdmin の SSH トンネルを使用して接続
- AWS Amplify を使用して React アプリケーションを作成し、Amazon Cognito による認証を追加する
- 複数の でのインバウンドインターネットアクセスに関する Network Access Analyzer の検出結果のレポートを作成する AWS アカウント
- の Amazon CloudWatch アラートをカスタマイズする AWS Network Firewall
- AWS Network Firewall と AWS Transit Gateway を使用してファイアウォールをデプロイする
- ChatOps ソリューションをデプロイして、チャットアプリケーションのカスタムアクションと で Amazon Q Developer を使用して SAST スキャン結果を管理する AWS CloudFormation
- AWS ランディングゾーンの設計を文書化する
- Amazon RDS の PostgreSQL DB インスタンスに対して暗号化された接続を有効にする
- 既存の Amazon RDS for PostgreSQL DB インスタンスを暗号化する
- 起動時に Amazon RDS データベースの自動タグ付けを強制する
- 起動時に Amazon EMR クラスターのタグ付けを強制する
- Amazon S3 へのAmazon EMR ロギングが有効になっていることを確認する
- Troposphere を使用して AWS Config マネージドルールを含む AWS CloudFormation テンプレートを生成します
- AWS KMS キーのキーの状態が変更されたときに Amazon SNS 通知を受け取る
- DynamoDB でのタグ付けの強制を支援
- Amazon Data Firehose リソースが AWS KMS キーで暗号化されていない場合の識別とアラート
- SQL Server から PostgreSQL に移行するときに PII データの SHA1 ハッシュを実装する
- AWS CDK を使用して複数の AWS リージョン、アカウント、OU にわたって Amazon DevOps Guru を有効にすることで、運用パフォーマンスを向上させます。
- EC2 Windows インスタンスを AWS Managed Services アカウントに取り込み、移行
- AWS DMS を使用して Amazon RDS for Oracle を Amazon RDS for PostgreSQL に移行します
- ELK スタックを AWS 上のElasticクラウドに移行する
- F5 BIG-IPワークロードをAWS クラウド上のF5 BIG-IP VEに移行する
- 暗号化されていない Amazon Aurora インスタンスをモニタリングする
- ロール自動販売機ソリューションをデプロイして最小特権の IAM ロールをプロビジョニングする
- コンテナを再起動せずにデータベースの認証情報をローテーションする

- トラステッドコンテキストを使用して、AWS の Db2 フェデレーションデータベースのユーザーアクセスを保護し、合理化する
- AWS Firewall Manager と Amazon Data Firehose を使用して Splunk に AWS WAF ログを送信する
- Amazon CloudFront を使用して Amazon S3 バケットの静的なコンテンツを VPC 経由で配信する
- cert-managerと Let's Encrypt を使用して Amazon EKS のアプリケーションのエンドツーエンド暗号化を設定
- アクセスコントロールと自動化に IAM ポリシーでユーザー IDs
- ELB ロードバランサーに TLS 終了が必要であることを確認
- Splunk を使用して AWS Network Firewall のログとメトリクスを表示する
- Amazon QuickSight を使用してすべての AWS アカウントの IAM 認証情報レポートを視覚化します

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。