aws

デベロッパーガイド

Amazon OpenSearch Service



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon OpenSearch Service: デベロッパーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

Amazon OpenSearch Service とは？ 1
Amazon OpenSearch Service の機能 2
どのようなときに使うか 3
サポートされているバージョンの Elasticsearch および OpenSearch 4
標準サポートおよび延長サポート 4
標準サポートおよび延長サポート対象バージョン 7
延長サポート料金の計算 7
料金 8
請求および使用状況レポートの理解 9
関連サービス 10
設定 12
許可を付与する 12
にサインアップする AWS アカウント 12
のセットアップ AWS CLI 12
コンソールを開きます。 13
開始方法 14
ドメインの作成 14
インデックス作成のためにデータをアップロードする 16
オプション 1: 単一のドキュメントをアップロードする 17
オプション 2: 複数のドキュメントをアップロードする 17
ドキュメントの検索 18
コマンドラインからドキュメントを検索する 18
OpenSearch Dashboards を使用してドキュメントを検索する 19
ドメインの削除 20
Amazon OpenSearch Ingestion 22
主要なコンセプト 22
利点 24
制限事項 24
サポートされている Data Prepper のバージョン 25
パイプラインのスケーリング 26
料金 28
サポートされている AWS リージョン 28
ロールとユーザーの設定 28
パイプラインロール 29

取り込みロール ... 33
パイプラインにドメインへのアクセスを付与する ... 35
パイプラインにコレクションへのアクセス権を付与する ... 38
OpenSearch Ingestion の開始方法 ... 43
チュートリアル: ドメインにデータを取り込む ... 44
チュートリアル: コレクションにデータを取り込む ... 53
パイプライン機能 ... 59
永続的バッファリング ... 60
分割 ... 62
チェーン ... 63
デッドレターキュー ... 64
インデックス管理 ... 66
エンドツーエンドの確認応答 ... 69
ソースバックプレッシャー ... 70
パイプラインの作成 ... 71
前提条件と必要な IAM ロール ... 71
必要な IAM 許可 ... 72
パイプラインのバージョンの指定 ... 74
取り込みパスの指定 ... 75
パイプラインの作成 ... 75
パイプラインの作成ステータスの追跡 ... 80
ブループリントの使用 ... 82
パイプラインの表示 ... 83
パイプラインの更新 ... 86
考慮事項 ... 86
必要なアクセス許可 ... 87
パイプラインの更新 ... 88
パイプライン更新用のブルー/グリーンデプロイ ... 89
パイプラインのコストの管理 ... 89
パイプラインの停止 ... 90
パイプラインの開始 ... 91
パイプラインの削除 ... 92
サポートされているプラグインとオプション ... 93
サポートされているプラグイン ... 94
ステートレスプロセッサとステートフルプロセッサ ... 96
設定の要件と制限 ... 97

パイプラインの統合 102
取り込みエンドポイントの構築 102
取り込みロールの作成 103
Atlassian のサービス 105
Amazon Aurora 123
Amazon DynamoDB 157
Amazon DocumentDB 173
Confluent Cloud Kafka 191
Amazon MSK 201
Amazon RDS 209
Amazon S3 238
Amazon Security Lake 248
Fluent Bit 253
Fluentd 254
ML オフラインバッチ推論 256
OpenTelemetry Collector 273
Amazon Managed Service for Prometheus 279
セルフマネージド型の Kafka 287
セルフマネージド OpenSearch クラスター 295
Amazon Kinesis Data Streams 303
次の手順 315
AWS Lambda 316
ドメインとコレクション間のデータの移行 319
制限事項 320
送信元としての OpenSearch Service 320
複数の OpenSearch Service ドメインシンクの指定 323
OpenSearch Serverless VPC コレクションへのデータの移行 323
AWS SDKs を使用したパイプラインの管理 324
Python 324
OpenSearch Ingestion のセキュリティ 328
パイプラインの VPC アクセスを設定する 329
クロスアカウント取り込み用のパイプラインの設定 334
Identity and Access Management 341
CloudTrail によるモニタリング 349
VPC エンドポイント (AWS PrivateLink) 353
パイプラインのタグ付け 356

必要なアクセス許可 ... 357
タグの操作 (コンソール) ... 357
タグの操作 (AWS CLI) ... 358
ログ記録とモニタリング ... 358
パイプラインのログのモニタリング ... 359
パイプラインメトリクスのモニタリング ... 361
ベストプラクティス ... 392
一般的なベストプラクティス ... 392
推奨される CloudWatch アラーム ... 392
Amazon OpenSearch Serverless ... 399
利点 ... 399
Amazon OpenSearch Serverless とは ... 400
OpenSearch Serverless のユースケース ... 401
仕組み ... 401
コレクションタイプを選択する ... 402
サポートされている AWS リージョン ... 403
制限事項 ... 404
OpenSearch Service と OpenSearch Serverless を比較する ... 404
チュートリアル: OpenSearch Serverless の開始方法 ... 408
ステップ 1: アクセス許可を設定する ... 409
ステップ 2: コレクションを作成する ... 410
ステップ 3: データをアップロードして検索する ... 411
エラー処理 ... 412
ステップ 4: コレクションを削除する ... 412
次の手順 ... 413
コレクション ... 413
コレクションの管理 ... 414
コレクションエンドポイント ... 446
ベクトル検索コレクションの使用 ... 449
データライフサイクルポリシーを使用する ... 465
AWS SDKs を使用したコレクションの管理 ... 474
CloudFormation を使用したコレクションの作成 ... 486
スナップショットを使用したコレクションのバックアップ ... 487
Amazon OpenSearch Serverless での Zstandard Codec サポート ... 495
派生ソースを使用してストレージを保存する ... 497
コレクショングループ ... 499

主要なコンセプト 500
コレクショングループの容量制限 500
コレクショングループの暗号化キーと KMS キー 503
コレクショングループを作成する 505
コレクショングループを管理する 508
ゼロにスケール 511
動作をゼロにスケールする 511
ゼロへのスケールの有効化 512
ゼロへのスケールアウト 512
キャパシティ制限の管理 513
キャパシティの設定 515
最大キャパシティの制限 516
キャパシティ使用量のモニタリング 516
コレクションへのデータの取り込み 516
必要な最小限のアクセス許可 517
OpenSearch Ingestion 518
Fluent Bit 518
Amazon Data Firehose 519
Go 519
Java 522
JavaScript 524
Logstash 527
Python 529
Ruby 531
その他のクライアント 532
機械学習を設定する 533
機械学習 533
Connector 533
モデル 534
機械学習のアクセス許可を設定する 535
サポートされていない API と機能 536
ニューラル検索とハイブリッド検索の設定 537
Neural Search 537
ハイブリッド検索 537
ニューラルクエリとハイブリッドクエリ 538
アクセス許可の設定 539

サポートされている AWS リージョン 542
ワークフローの設定 542
ワークフロー 542
アクセス許可の設定 543
OpenSearch Serverless のセキュリティ 545
暗号化ポリシー 547
ネットワークポリシー 547
データアクセスポリシー 548
IAM および SAML 認証 549
インフラストラクチャセキュリティ 550
セキュリティの開始方法 550
Identity and Access Management 565
暗号化 583
ネットワークアクセス 595
FIPS 準拠 607
データアクセスコントロール 613
データプレーン VPC エンドポイント 624
コントロールプレーン VPC エンドポイント 638
SAML 認証 641
コンプライアンス検証 650
タグコレクション 651
必要なアクセス許可 651
コレクションのタグ付け (コンソール) 652
コレクションのタグ付け (AWS CLI) 652
サポートされているオペレーションとプラグイン 653
サポートされる OpenSearch API オペレーションとアクセス許可 653
サポートされている OpenSearch プラグイン 664
OpenSearch Serverless のモニタリング 665
CloudWatch によるモニタリング 666
CloudTrail によるモニタリング 672
EventBridge によるモニタリング 677
ドメインの作成と管理 681
OpenSearch Service ドメインの作成 681
OpenSearch Service ドメインの作成 (コンソール) 681
OpenSearch Service ドメインの作成 (AWS CLI) 688
OpenSearch Service ドメイン (AWS SDKsの作成 690

OpenSearch Service ドメインの作成 (AWS CloudFormation) ... 690
アクセスポリシーの設定 ... 691
Migration Assistant を使用した OpenSearch Service への移行 ... 691
概要 ... 691
移行シナリオ ... 692
Migration Assistant の開始方法 ... 693
ソリューションアーキテクチャ ... 694
高度なクラスター設定 ... 694
自動セマンティックエンリッチメント ... 695
序章 ... 695
モデルの詳細とパフォーマンスベンチマーク ... 696
サポートされている言語 ... 697
ドメインの自動セマンティックエンリッチメントインデックスを設定する ... 697
既存のインデックスを更新する ... 699
データの取り込みと検索 ... 701
自動セマンティックエンリッチメントのアクセス許可の設定 ... 702
クエリの書き換え ... 705
自動セマンティックエンリッチメントの制限 ... 705
料金 ... 706
サポートされる AWS リージョン ... 706
設定変更 ... 707
通常は blue/green デプロイの原因となる変更 ... 707
通常は blue/green デプロイが発生しない変更 ... 708
ブルー/グリーンデプロイオプション ... 710
変更によってブルー/グリーンデプロイが実行されるかどうかを判断する ... 712
設定変更の追跡 ... 716
設定変更のステージ ... 718
ブルー/グリーンデプロイのパフォーマンスへの影響 ... 721
設定変更に関連する料金 ... 721
検証エラーのトラブルシューティング ... 722
サービスソフトウェア更新 ... 727
オプションの更新と必須の更新 ... 728
パッチ更新 ... 729
考慮事項 ... 729
更新を開始する ... 730
オフピークウィンドウ ... 733

更新のモニタリング 735
ドメインが更新の対象でない場合 735
更新のロールバック 736
オフピークウィンドウ 741
オフピーク時のサービスソフトウェアの更新 742
オフピーク時の Auto-Tune の最適化 742
オフピークウィンドウの有効化 743
カスタムのオフピークウィンドウの設定 744
スケジュール済みのアクションを表示する 745
アクションのスケジュール変更 746
Auto-Tune のメンテナンスウィンドウからの移行 748
通知 749
通知の使用開始 749
通知重要度 750
サンプル EventBridge イベント 751
マルチ AZ ドメインの設定 752
Multi-AZ with Standby 752
Multi-AZ without Standby 754
アベイラビリティーゾーンの中断 757
VPC サポート 759
VPC 対パブリックドメイン 759
制限事項 760
アーキテクチャ 761
送信トラフィックオプション 768
VPC を介したドメイン出力トラフィックのルーティング 768
インデックススナップショットの作成 773
前提条件 774
手動スナップショットレポジトリの登録 778
手動スナップショットの作成 782
スナップショットからのデータの復元 784
手動スナップショットの削除 787
Snapshot Management を用いたスナップショットの自動化 788
インデックスステート管理を用いたスナップショットの自動化 792
スナップショットの Curator の使用 792
ドメインのアップグレード 793
サポートされているアップグレードパス 793

ドメインのアップグレード (コンソール) ... 798
ドメインのアップグレード (CLI) ... 798
ドメインのアップグレード (SDK) ... 799
検証障害のトラブルシューティング ... 801
アップグレードのトラブルシューティング ... 801
スナップショットを使用してデータを移行する ... 804
カスタムエンドポイントの作成 ... 812
新しいドメインのカスタムエンドポイント ... 812
既存のドメインのカスタムエンドポイント ... 813
CNAME マッピング ... 813
Auto-Tune ... 814
変更のタイプ ... 815
Auto-Tune を有効または無効にする ... 816
Auto-Tune の機能強化のスケジューリング ... 817
オートチューンの変更の監視 ... 818
ドメインのタグ付け ... 818
タグ付けの例 ... 819
ドメインのタグ付け (コンソール) ... 820
ドメインのタグ付け (AWS CLI) ... 820
ドメインのタグ付け (AWS SDKs) ... 822
管理アクションの実行 ... 823
データノード上で OpenSearch プロセスを再起動するには ... 823
データノードの再起動 ... 824
Dashboards プロセスの再起動 ... 824
制限事項 ... 825
ダイレクトクエリの使用 ... 827
S3 でのダイレクトクエリ ... 827
S3 データソースの作成 ... 828
S3 データソースの設定 ... 836
料金 ... 838
制限事項 ... 839
推奨事項 ... 840
クォータ ... 840
サポートされている AWS リージョン ... 841
CloudWatch Logs でのダイレクトクエリ ... 842
CloudWatch Logs データソースの作成 ... 842

CloudWatch Logs データソースの設定 ... 849
料金 ... 851
制限事項 ... 851
推奨事項 ... 852
クォータ ... 853
サポートされている AWS リージョン ... 854
Security Lake でのダイレクトクエリ ... 855
Security Lake データソースの作成 ... 856
Security Lake データソースの設定 ... 862
料金 ... 865
制限事項 ... 865
推奨事項 ... 866
クォータ ... 867
サポートされている AWS リージョン ... 868
Managed Prometheus での直接クエリ ... 868
Managed Prometheus データソースの作成 ... 869
Managed Prometheus データソースの設定とクエリ ... 874
料金 ... 875
制限事項 ... 875
推奨事項 ... 875
クォータ ... 876
サポートされる AWS リージョン ... 876
データソースの管理 ... 877
CloudWatch メトリクスデータソースによるモニタリング ... 878
データソースの有効化と無効化 ... 881
AWS Budget によるモニタリング ... 881
データソースの削除 ... 881
サポートされている SQL コマンドと PPL コマンド ... 882
サポートされている SQL コマンド ... 884
サポートされている PPL コマンド ... 1085
ドメインのモニタリング ... 1265
クラスターメトリクスのモニタリング ... 1266
CloudWatch でのメトリクスの表示 ... 1267
OpenSearch Service でのヘルスチャートの解釈 ... 1267
クラスターメトリクス ... 1268
専用マスターノードメトリクス ... 1276

専用コーディネーターノードメトリクス 1278
EBS ボリュームメトリクス 1278
インスタンスメトリクス 1281
ウォームメトリクス 1294
コールドストレージのメトリクス 1301
OpenSearch Optimized インスタンス (OR1) メトリクス 1302
アラートメトリクス 1302
異常検出のメトリクス 1304
非同期検索メトリクス 1306
Auto-Tune メトリクス 1308
Multi-AZ with Standby メトリクス 1308
ポイントインタイムメトリクス 1311
SQL メトリクス 1311
k-NN メトリクス 1312
クラスター間検索のメトリクス 1316
クラスター間レプリケーションメトリクス 1317
Learning to Rank のメトリクス 1319
Piped Processing Language のメトリクス 1319
ログをモニタリングする 1320
ログ発行を有効にする (コンソール) 1322
ログ発行の有効化 (AWS CLI) 1324
ログ発行 (AWS SDKs) の有効化 1326
ログの発行を有効にする (CloudFormation) 1326
検索リクエストスのスローログのしきい値設定 1328
シャードスローログのしきい値設定 1329
スローログのテスト 1329
ログの表示 1330
監査ログのモニタリング 1331
制限事項 1331
監査ログ記録の有効化 1332
を使用して監査ログ記録を有効にする AWS CLI 1334
設定 API を使用して監査ログを有効にする 1334
監査ログのレイヤーとカテゴリ 1334
監査ログの設定 1337
監査ログの例 1340
REST API を使用した監査ログの設定 1343

イベントのモニタリング 1344
サービスソフトウェア更新イベント 1345
Auto-Tune イベント 1352
クラスターヘルスイベント 1357
VPC エンドポイントイベント 1370
ノードの廃止イベント 1373
デグレードされたノードの廃止イベント 1375
ドメインエラーイベント 1377
チュートリアル: OpenSearch Service イベントをリッスンする 1379
チュートリアル: 利用可能な更新に関する SNS アラートの送信 1381
CloudTrail によるモニタリング 1383
CloudTrail での Amazon OpenSearch Service 情報 673
Amazon OpenSearch Service のログファイルエントリを理解する 676
クラスターインサイト 1388
利点 1388
コンソールからクラスターインサイトにアクセスする 1389
OpenSearch UI を使用して Cluster Insights と詳細なメトリクスにアクセスする 1390
クラスターインサイトを表示するようにアプリケーションを作成して設定する 1390
クラスターインサイトについて 1392
概要ダッシュボード 1393
現在のクラスターステータス 1393
インサイトの傾向 1393
現在のオープンインサイト 1394
OpenSearch Service クラスター 1394
重要度別の上位のインサイト 1394
インサイトの詳細 1394
クラスターの詳細 1394
概要 1394
設定のベストプラクティス 1395
Insights 1395
メトリクスセクション 1396
ノード、インデックス、シャードビュー 1396
クエリビュー 1397
Amazon EventBridge イベントを通じて Insights にアクセスする 1399
Insights Catalog 1399
オブザーバビリティ 1403

Amazon OpenSearch Service が提供するもの 1403
オブザーバビリティに対する Amazon OpenSearch Service のアプローチ 1403
はじめに 1404
クイックスタート 1404
CDK デプロイ 1405
テレメトリの送信 1405
詳細情報 1406
MCP アプリによるエージェントのオブザーバビリティ 1406
MCP アプリの例 1407
主要なコンセプト 1412
前提条件 1413
MCP サーバーのセットアップ 1413
エージェントオブザーバビリティに利用可能な MCP アプリ 1415
mcpb ファイルの検証 1415
アプリケーションのテレメトリの取り込み 1416
OpenTelemetry Collector の設定 1416
OpenSearch Ingestion パイプラインの設定 1417
取り込みの検証 1421
次の手順 1421
データセット 1422
データセットタイプ 1422
ログデータセットを作成するには 1422
トレースデータセットを作成するには 1424
データセットを表示するには 1425
Discover でのデータセットの分析 1426
ログの検出 1426
ログページにアクセスするには 1427
ログデータの探索 1427
PPL を使用したログのクエリ 1428
ログからの視覚化の作成 1429
ダッシュボードへの視覚化の追加 1432
トレースの検出 1432
トレースページにアクセスするには 1432
トレースデータセットの設定 1433
トレースデータの探索 1434
特定のスパンの表示 1435

トレースの詳細ページ 1435
トレースとログの関連付け 1436
PPL を使用したトレースのクエリ 1438
メトリクスの検出 1438
Prometheus データソースの設定 1439
クエリパネル 1439
時間フィルター 1440
結果の表示 1440
ビジュアライゼーションの設定 1440
相関関係 1441
相関関係の要件 1441
ログtrace-to-logs相関を作成するには 1442
ログデータセットでの相関関係の表示 1444
トレースページでの相関関係の使用 1444
相関関係の管理 1445
ダッシュボード 1445
ダッシュボード構造 1447
Discover からのダッシュボードの構築 1447
ダッシュボードフィルター 1448
ダッシュボードの構築 1449
ダッシュボードの共有 1452
ダッシュボードのトラブルシューティング 1452
アプリケーションのモニタリング 1455
前提条件 1455
仕組み 1455
サービス 1456
アプリケーションマップ 1462
AI オブザーバビリティ 1466
開始方法 1467
エージェントトレース UI 1468
セキュリティ 1475
データ保護 1476
保管中の暗号化 1477
ノード間の暗号化 1481
Identity and Access Management 1482
ポリシーのタイプ 1482

OpenSearch Service リクエストの作成と署名 ... 1491
複数のポリシーが衝突する場合 ... 1493
ポリシーエレメントのリファレンス ... 1494
詳細オプションと API に関する考慮事項 ... 1500
アクセスポリシーの設定 ... 1503
追加のサンプルポリシー ... 1504
API アクセス許可のリファレンス ... 1504
AWS マネージドポリシー ... 1504
サービス間の混乱した代理の防止 ... 1515
きめ細かなアクセスコントロール ... 1517
全体像: きめ細かなアクセスコントロールと OpenSearch Service セキュリティ ... 1518
主要なコンセプト ... 1521
マスターユーザーについて ... 1522
きめ細かなアクセスコントロールの有効化 ... 1523
マスターユーザーとして OpenSearch Dashboards にアクセスする ... 1527
許可の管理 ... 1528
推奨される設定 ... 1533
制限事項 ... 1536
マスターユーザーの変更 ... 1538
追加のマスターユーザー ... 1538
手動スナップショット ... 1540
統合 ... 1540
REST API の相違点 ... 1541
チュートリアル: Cognito 認証によるきめ細かなアクセスコントロール ... 1543
チュートリアル: 内部ユーザーデータベースと基本認証 ... 1548
コンプライアンス検証 ... 1551
耐障害性 ... 1552
JSON Web トークン ... 1552
考慮事項 ... 1552
ドメインアクセスポリシーの変更 ... 1553
JWT 認証と認可の設定 ... 1553
JWKS エンドポイントを使用して JWT を検証する ... 1554
JWT を使用してテストリクエストを送信する ... 1556
インフラストラクチャセキュリティ ... 1558
OpenSearch Service で管理される VPC エンドポイントを使用する ... 1559
OpenSearch Dashboards の SAML 認証 ... 1563

SAML 設定の概要 1564
考慮事項 1564
VPC ドメインの SAML 認証 1565
ドメインアクセスポリシーの変更 1565
SP 開始認証または IdP 開始認証の設定 1567
SP 開始認証と IdP 開始認証両方の設定 1573
SAML 認証の設定 (AWS CLI) 1574
SAML 認証の設定 (設定 API) 1574
SAML のトラブルシューティング 1575
SAML 認証の無効化 1578
IAM アイデンティティセンターによる OpenSearch のサポート 1579
考慮事項 1580
ドメインアクセスポリシーの変更 1580
IAM アイデンティティセンターの認証と認可の設定 (コンソール) 1581
きめ細かなアクセスコントロールの設定 1581
IAM アイデンティティセンターの認証と認可の設定 (CLI) 1582
ドメインでの IAM アイデンティティセンター認証の無効化 1582
OpenSearch Dashboards の Amazon Cognito 認証 1582
前提条件 1583
Amazon Cognito 認証を使用するためのドメインの設定 1587
認証されたロールの許可 1591
ID プロバイダの設定 1592
(オプション) きめ細かなアクセスの設定 1592
(オプション) サインインページのカスタマイズ 1594
(オプション) アドバンストセキュリティを設定する 1594
テスト 1594
クォータ 1594
一般的な設定の問題 1595
OpenSearch Dashboards の Amazon Cognito 認証を無効にする 1600
OpenSearch Dashboards の Amazon Cognito 認証を使用するドメインを削除する 1600
サービスにリンクされたロールの使用 1600
VPC ドメインとデータソース作成ロール 1601
コレクション作成ロール 1605
パイプライン作成ロール 1608
「サンプルコード」 1611
Elasticech クライアントの互換性 1611

HTTP リクエストの圧縮 1612
gzip 圧縮を有効にする 1612
必要なヘッダー 1613
サンプルコード (Python 3) 1613
AWS SDKsの使用 1615
Java 1615
Python 1626
ノード 1629
データのインデックス作成 1632
インデックスの命名制限 1632
レスポンスサイズの削減 1633
インデックスコーデック 1635
OpenSearch Service へのストリーミングデータをロードする 1635
OpenSearch Ingestion からのストリーミングデータのロード 1636
Amazon S3 からストリーミングデータをロードする 1636
Amazon Kinesis Data Streams からストリーミングデータをロードする 1642
Amazon DynamoDB テーブルからストリーミングデータをロードする 1646
Amazon Data Firehose からストリーミングデータをロードする 1650
Amazon CloudWatch からストリーミングデータをロードする 1651
からのストリーミングデータのロード AWS IoT 1651
Logstash を用いてデータをロードする 1651
設定 1651
データの検索 1655
URI 検索 1656
リクエストボディ検索 1657
ブーストフィールド 1659
検索結果のハイライト 1659
Count API 1661
検索結果のページ分割 1662
ポイントインタイム 1662
`from` と `size` のパラメーター 1662
Dashboards Query Language 1663
パッケージ 1665
必要なアクセス許可 1666
Amazon S3 へのパッケージのアップロード 1666
パッケージのインポートと関連付け 1667

OpenSearch でのパッケージの使用 1668
パッケージの更新 1673
新しい辞書でインデックスを手動で更新する 1676
パッケージの関連付け解除と削除 1678
カスタムプラグイン 1679
サードパーティープラグイン 1696
SQL のサポート 1700
サンプル呼び出し 1702
注意と相違点 1703
SQL Workbench 1704
SQL CLI 1557
JDBC ドライバー 1704
ODBC ドライバー 1704
クラスター間検索 1704
制限事項 1705
クラスター間検索の前提条件 1706
クラスター間検索の料金 1706
接続のセットアップ 1706
接続の削除 1708
セキュリティのセットアップとサンプルチュートリアル 1708
OpenSearch Dashboards 1714
Learning to Rank 1715
Learning to Rank を開始する 1715
Learning to Rank API 1737
非同期検索 1744
サンプルの検索コール 1744
非同期検索アクセス許可 1746
非同期検索設定 1746
クラスター間検索 1747
UltraWarm 1749
ポイントインタイム 1749
考慮事項 1749
PIT の作成 1750
ポイントインタイムアクセス許可 1752
PIT の設定 1753
クラスター間検索 1753

UltraWarm 1753
エージェント検索 1753
セマンティック検索 1754
同時セグメント検索 1754
自然言語クエリの生成 1755
前提条件 1755
開始方法 1756
アクセス許可の設定 1756
設定の自動化 1757
ベクトル検索 1758
Amazon S3 ベクトルから OpenSearch Serverless へのインポート 1759
統合アーキテクチャ 1760
必要な IAM 許可 1762
Amazon S3 Vectors と OpenSearch の統合を設定する 1765
Amazon S3 ベクトルエンジンを使用した高度な検索機能 1767
S3 ベクトルの有効化 1767
S3 ベクトルエンジンを使用したインデックスの作成 1768
機能的制限 1770
ドキュメントのインデックス作成 1771
ドキュメントの検索 1772
サポートされているマッピングパラメータ 1773
使用量測定と請求 1775
s3vector エンジンの無効化 1775
k-NN 検索 1775
k-NN を用いた開始方法 1775
k-NN の違い、チューニング、および制限 1778
ベクトル取り込み 1778
仕組み 1779
OpenSearch Vector 取り込み料金 1779
前提条件 1779
ベクトルデータベースを作成する 1780
関連機能 1781
Amazon S3 ベクトルインデックスを OpenSearch Service ベクトルエンジンにエクスポートする 1781
Amazon S3 ベクトル名前空間を OpenSearch Service ベクトルエンジンにインポートする 1782

ベクトル取り込みジョブとインポート履歴を表示する 1784
自動最適化 1785
利点 1785
仕組み 1786
前提条件 1786
自動最適化のユースケース 1788
制限事項 1788
請求とコスト 1790
サポートされているデータ形式 1790
コンソールでの自動最適化の使用 1793
関連機能 1799
ベクトルインデックス作成の GPU アクセラレーション 1799
前提条件 1799
仕組み 1799
GPU アクセラレーションの料金 1800
GPU アクセラレーションと書き込みオペレーション 1800
サポートされているインデックス設定 1801
サポートされている AWS リージョン 1801
GPU アクセラレーションを有効にする 1802
GPU アクセラレーションベクトルインデックスを作成する 1805
インデックスとカのマージ 1806
ベストプラクティス 1807
OpenSearch Dashboards 1808
Dashboards へのアクセスの制御 1809
プロキシを使用して Dashboards から OpenSearch Service にアクセスする 1809
WMS マップサーバーを使用するための Dashboards の設定 1813
ローカルの Dashboards サーバーを OpenSearch Service に接続する 1814
Dashboards でのインデックスの管理 1815
その他の機能 1816
OpenSearch UI 1817
リリース履歴 1818
開始方法 1823
Amazon OpenSearch Service アプリケーションを作成するために必要なアクセス許可 1823
Creating an application 1827
アプリケーション管理者の管理 1835
エージェント AI 1838

主な機能 1839
料金と使用制限 1840
セキュリティ 1840
エージェント AI 機能の有効化と無効化 1840
リージョンの可用性 1841
エージェントチャット 1842
調査エージェント 1846
エージェントメモリ 1848
暗号化とカスタマーマネージドキー 1849
カスタマーマネージドキーを使用するための前提条件 1850
コンソールを使用したカスタマーマネージドキー暗号化を使用したアプリケーションの作成 1851
を使用したカスタマーマネージドキー暗号化によるアプリケーションの作成 AWS CLI 1852
カスタマーマネージドキーの使用状況のモニタリング 1853
暗号化設定の更新 1853
IAM による SAML フェデレーションの有効化 1854
ステップ 1: ID プロバイダーアプリケーションを設定する (Okta) 1855
ステップ 2: Okta AWS の設定をセットアップする 1858
ステップ 3: IAM で Amazon OpenSearch Service アクセスポリシーを作成する 1860
ステップ 4: SAML で ID プロバイダーが開始したシングルサインオンエクスペリエンスを検証する 1863
ステップ 5: SAML 属性ベースの詳細なアクセス制御を設定する 1867
データソースの関連付けと VPC アクセス許可の管理 1871
データソースと OpenSearch UI アプリケーションの関連付け 1871
VPC 内のドメインへのアクセスの管理 1872
VPC 内の OpenSearch Serverless コレクションへのアクセスの設定 1873
Amazon OpenSearch Service ワークスペースの使用 1877
OpenSearch UI アプリケーションワークスペースの作成 1877
ワークスペースのプライバシーと共同作業者 1878
ワークスペースタイプ 1878
クロスリージョンおよびクロスアカウントデータアクセス 1879
OpenSearch ドメインへのクロスアカウントデータアクセス 1880
OpenSearch ドメインへのクロスリージョンデータアクセス 1904
クラスター間検索 1926
VPC エンドポイントからの OpenSearch UI へのアクセスの管理 1936
VPC と OpenSearch UI 間のプライベート接続の作成 1936

OpenSearch UI アプリケーションへのアクセスを許可するように VPC エンドポイントポリシーを更新する ... 1938
VPC エンドポイントポリシーでの OpenSearch UI へのアクセスの取り消し ... 1939
OpenSearch Dashboard からの移行 ... 1939
移行の概要 ... 1940
前提条件 ... 1941
ステップ 1: OpenSearch Dashboards から保存されたオブジェクトをエクスポートする ... 1941
ステップ 2: 保存したオブジェクトを OpenSearch UI にインポートする ... 1944
エンドポイントとクォータ ... 1946
OpenSearch UI エンドポイント ... 1946
OpenSearch UI サービスクォータ ... 1949
インデックス管理 ... 1951
OpenSearch 最適化インスタンス ... 1951
制限事項 ... 1952
取り込みスループットを向上させるための調整 ... 1953
OpenSearch 最適化インスタンスと他のインスタンスの違い ... 1953
OpenSearch 最適化インスタンスと UltraWarm インスタンスの違い ... 1953
OpenSearch 最適化インスタンスを使用したドメインのプロビジョニング ... 1954
多層ストレージ ... 1956
多層ストレージアーキテクチャ ... 1957
UltraWarm ストレージ ... 1963
前提条件 ... 1964
UltraWarm ストレージの要件とパフォーマンスに関する考慮事項 ... 1966
UltraWarm 料金設定 ... 1967
UltraWarm を有効にする ... 1967
UltraWarm ストレージへのインデックスの移行 ... 1969
移行を自動化する ... 1973
移行の調整 ... 1973
移行をキャンセルする ... 1973
ホットインデックスとウォームインデックスを一覧表示する ... 1974
ウォームインデックスをホットストレージに戻す ... 1974
スナップショットからのウォームインデックスの復元 ... 1974
ウォームインデックスの手動スナップショット ... 1976
ウォームインデックスをコールドストレージへ移行する ... 1977
KNN インデックスのベストプラクティス ... 1977
UltraWarm を無効にする ... 1978

コールドストレージ 1978
前提条件 1979
コールドストレージの要件とパフォーマンスに関する考慮事項 1981
コールドストレージの料金 1981
コールドストレージを有効にする 1981
OpenSearch Dashboards でのコールドインデックスの管理 1983
コールドストレージへのインデックスの移行 1984
コールドストレージへの移行の自動化 1985
コールドストレージへの移行のキャンセル 1986
コールドインデックスの一覧表示 1986
ウォームストレージへのインデックスの移行 1990
スナップショットからのコールドインデックスの復元 1991
コールドストレージからウォームストレージへの移行のキャンセル 1992
コールドインデックスメタデータの更新 1992
コールドインデックスの削除 1993
コールドストレージを無効にする 1993
インデックスステート管理 1993
ISM ポリシーを作成する 1994
サンプルポリシー 1995
ISM テンプレート 1999
相違点 2000
チュートリアル: ISM プロセスの自動化 2002
インデックスロールアップ 2006
インデックスロールアップジョブの作成 2007
インデックス変換 2008
インデックス変換ジョブの作成 2009
クラスター間レプリケーション 2010
制限事項 2011
前提条件 2012
アクセス許可の要件 2012
クラスター間接続のセットアップ 2014
レプリケーションの開始 2015
レプリケーションの確認 2016
レプリケーションの一時停止と再開 2017
レプリケーションの開始 2018
自動フォロー 2018

接続されたドメインのアップグレード 2019
リモート再インデックス 2020
前提条件 2021
OpenSearch Service インターネットドメイン間でデータを再インデックスする 2021
リモートドメインが VPC にあるときのデータの再インデックス 2023
OpenSearch Service 以外のドメイン間でデータの再インデックスを行う 2027
大規模なデータセットの再インデックスを行う 2028
リモート再インデックス設定 2030
データストリーム 2030
Data Streams の使用開始 2031
データのモニタリング 2035
[アラート] 2035
アラートの許可 2036
アラートの開始方法 2036
通知 2037
相違点 2038
異常検出 2039
前提条件 2040
異常検出の開始方法 2040
チュートリアル: 異常検出で高い CPU 使用率を検出する 2043
機械学習 2047
のコネクタ AWS のサービス 2047
前提条件 2047
OpenSearch Service コネクタを作成する 2051
外部プラットフォーム用コネクタ 2053
前提条件 2054
OpenSearch Service コネクタを作成する 2057
CloudFormation テンプレートの統合 2059
使用可能な CloudFormation テンプレート 2060
前提条件 2061
Amazon Bedrock テンプレート 2063
Bedrock Claude を使用したエージェント検索の設定 2064
MCP サーバー統合テンプレート 2065
Amazon SageMaker テンプレート 2068
セマンティックハイライトテンプレートに対するリモート推論 2069
サポートされていない ML Commons 設定 2070

フローフレームワークプラグイン 2071
OpenSearch Service での ML コネクタの作成 2071
アクセス許可を設定する 2079
AI エージェント統合 2081
エージェントスキル 2081
ユースケース 2082
使用可能なスキル 2083
前提条件 2084
スキルのインストールと使用 2084
セキュリティに関する考慮事項 2085
その他のリソース 2086
MCP サーバー 2086
前提条件 2086
インストール 2087
認証 2087
IDE で を設定する 2088
エージェントフレームワーク 2091
セキュリティに関する考慮事項 2094
トラブルシューティング 2094
その他のリソース 2095
セキュリティ分析 2096
セキュリティ分析のコンポーネントと概念 2096
ログタイプ 2097
ディテクター 2097
Rules 2097
検出結果 2097
アラート 2097
セキュリティ分析を理解する 2098
アクセス許可の設定 2100
トラブルシューティング 2102
そのようなインデックスエラーはありません 2102
ベストプラクティス 2103
モニタリングとアラート 2103
CloudWatch アラームを設定する 2103
ログの発行を有効にする 2104
シャード戦略 2104

シャードとデータノード数を決定する 2105
ストレージスキューを回避する 2106
安定性 2106
OpenSearch で最新の状態に保つ 2106
スナップショットパフォーマンスの向上 2107
専用マスターノードを有効にする 2107
複数のアベイラビリティゾーンにデプロイする 2108
取り込みフローとバッファリングを制御する 2108
検索ワークロードのマッピングを作成する 2109
インデックステンプレートを使用する 2109
Index State Management でインデックスを管理する 2110
未使用インデックスの削除 2111
高可用性を実現するために複数のドメインを使用する 2111
パフォーマンス 2111
一括リクエストのサイズと圧縮を最適化する 2112
一括リクエストのレスポンスのサイズを小さくする 2112
更新間隔を調整する 2112
Auto-Tune を有効にする 2113
セキュリティ 2113
きめ細かなアクセスコントロールを有効にする 2113
VPC 内にドメインをデプロイする 2113
制限的なアクセスポリシーを適用する 2114
保管中の暗号化を有効にする 2113
ノード間の暗号化を有効にする 2114
によるモニタリング AWS Security Hub CSPM 2115
コスト最適化 2115
最新世代のインスタンスタイプを使用する 2115
最新の Amazon EBS gp3 ボリュームを使用する 2115
時系列のログデータのために UltraWarm とコールドストレージを使用する 2116
リザーブドインスタンスの推奨事項を確認する 2116
推奨される CloudWatch アラーム 2116
検討した方が良いその他のアラーム 2121
ドメインのサイジング 2124
ストレージ要件の計算 2124
シャード数の選択 2126
インスタンスタイプとテストの選択 2128

ペタバイトスケール 2130
専用コーディネーターノード 2131
専用コーディネーターノードを使用すべきシチュエーション 2132
アーキテクチャと動作 2132
要件と制限 2132
専用コーディネーターノードのプロビジョニング 2133
ベストプラクティス 2134
専用マスターノード 2135
専用マスターノードの数の選択 2137
専用マスターノードのインスタンスタイプの選択 2138
全般的なリファレンス 2140
コスト最適化 2140
1-Yearまたは複数年のコミットメントで大きな節約 2140
Amazon OpenSearch Service マネージドクラスターのコスト最適化 2141
Amazon OpenSearch Service Serverless のコスト最適化 2146
サポートされるインスタンスタイプ 2151
現行世代のインスタンスタイプ 2151
旧世代のインスタンスタイプ 2169
エンジンバージョン別の機能 2180
エンジンバージョンに応じたプラグイン 2186
オプションプラグイン 2189
サポートされているオペレーション 2190
API の重要な相違点 2190
クォータ 2246
ウォームノードストレージクォータ 2246
1 AZ あたりのデータノード数 2138
インスタンスファミリー別の合計ノード制限 2248
EBS ボリュームサイズのクォータ 2249
ネットワークのクォータ 2261
シャードサイズのクォータ 2272
シャード数のクォータ 2272
Java プロセスのクォータ 2273
ドメインポリシーのクォータ 2273
リザーブドインスタンス 2273
リザーブドインスタンスの購入 (コンソール) 2274
リザーブドインスタンスの購入 (AWS CLI) 2275

リザーブドインスタンス (AWS SDKsの購入 ... 2277
コストを確認する ... 2279
サポートされている他のリソース ... 2279
チュートリアル ... 2281
ドキュメントの作成および検索 ... 2281
前提条件 ... 2281
ドキュメントのインデックスへの追加 ... 2282
自動的に生成される ID の作成 ... 2283
POST コマンドを使用したドキュメントの更新 ... 2284
一括アクションの実行 ... 2285
ドキュメントの検索 ... 2286
関連リソース ... 2288
OpenSearch Service への移行 ... 2288
スナップショットの作成とアップロード ... 2289
ドメインの作成 ... 2290
S3 バケットへの許可を提供します。 ... 2291
スナップショットを復元する ... 2293
検索アプリケーションを作成する ... 2296
前提条件 ... 2297
ステップ 1: サンプルデータのインデックス作成 ... 2297
ステップ 2: Lambda 関数を作成してデプロイする ... 2298
ステップ 3: API Gateway で API を作成する ... 2301
ステップ 4 (オプション): ドメインアクセスポリシーを変更する ... 2304
Lambda ロールをマッピングする (きめ細かなアクセスコントロールを使用している場合) 2307
ステップ 5: ウェブアプリケーションをテストする ... 2307
次の手順 ... 2309
サポートコールを可視化する ... 2310
ステップ 1: 前提条件を設定する ... 2311
ステップ 2: サンプルコードをコピーする ... 2312
(オプション) ステップ 3: サンプルデータのインデックスを作成する ... 2316
ステップ 4: データを分析し、可視化する ... 2318
ステップ 5: リソースのクリーンアップと次のステップ ... 2322
Amazon OpenSearch Service 名称変更 ... 2324
API の新バージョン ... 2324
名称変更されたインスタンスタイプ ... 2324
アクセスポリシーの変更 ... 2325

IAM ポリシー 2325
SCP ポリシー 2325
新しいリソースタイプ 2326
Kibana は OpenSearch Dashboards に名称変更されました 2327
名称変更された CloudWatch メトリクス 2327
請求およびコストマネジメントコンソールの変更 2329
新しいイベント形式 2329
同じままのものは何ですか？ 2330
使用開始: ドメインを OpenSearch 1.x にアップグレードします 2330
トラブルシューティング 2332
OpenSearch Dashboards にアクセスできない 2332
VPC ドメインにアクセスできない 2332
VPC ドメインからのエグレストラフィック 2332
読み取り専用状態のクラスター 2333
赤のクラスター状態 2334
赤いクラスターの自動修復 2336
処理の継続的な高負荷からの復旧 2336
黄色のクラスター状態 2338
ClusterBlockException 2339
使用可能なストレージ領域の不足 2339
JVM メモリ負荷が高い 2339
Multi-AZ with Standby への移行中にエラーが発生した 2340
スタンバイのないドメインからスタンバイのあるドメインへの移行している最中に、インデックス、インデックステンプレート、ISM ポリシーのいずれかを作成する 2102
データコピーの数が間違っている 2340
JVM OutOfMemoryError 2341
障害が発生したクラスターノード 2342
シャードの最大制限を超えました 2342
ドメインが処理状態でスタックしている 2343
低 EBS バーストバランス 2343
ボリュームサイズ変更中の EBS メトリクスの増加 2343
監査ログを有効にできない 2344
インデックスが閉じない 2344
クライアントライセンスのチェック 2345
リクエストのスロットリング 2345
ノードに SSH 接続できない 2345

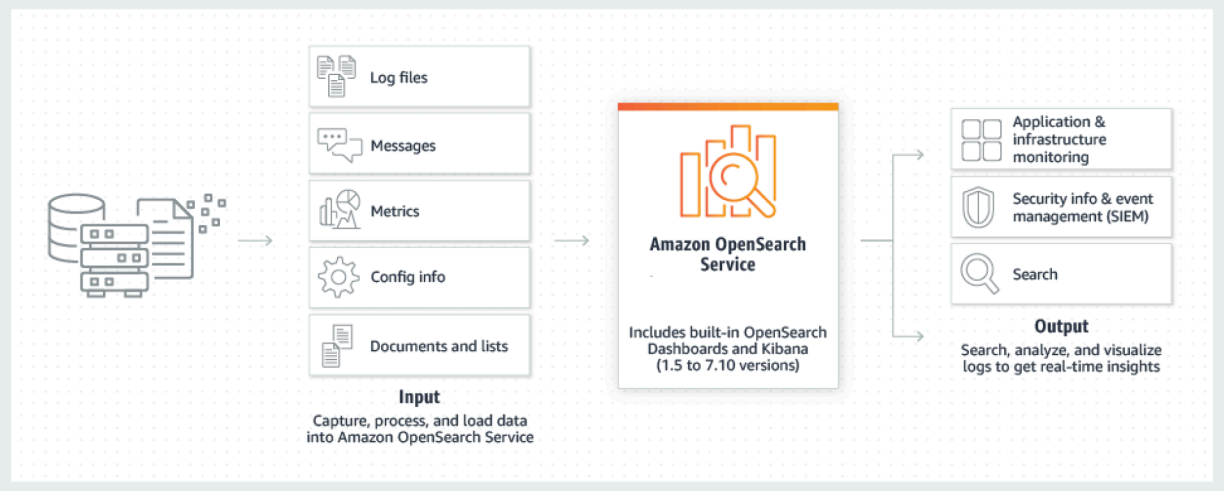
「オブジェクトのストレージクラスで有効ではない」スナップショットエラー 2345
無効なホストヘッダー 2346
無効な M3 インスタンスタイプ 2346
ホットクエリが UltraWarm を有効にした後に動作を停止する 2346
アップグレード後にダウングレードできない 2347
すべての のドメインの概要が必要 AWS リージョン 2347
OpenSearch Dashboards を使用する場合のブラウザエラー 2347
ノードシャードとストレージスキュー 2348
インデックスシャードとストレージスキュー 2349
VPC アクセス選択後の許可されていないオペレーション 2349
VPC ドメイン作成後、読み込みでスタックする 2350
OpenSearch API への拒否されたリクエスト 2350
Alpine Linux から接続できない 2351
Search Backpressure のリクエストが多すぎる 2351
SDK を使用する場合の証明書のエラー 2352
バージョンの互換性の問題によりカスタムプラグインのインストールが失敗する 2353
ドキュメント履歴 2355
以前の更新 2424
AWS 用語集 2427
..... mmcdxxviii

Amazon OpenSearch Service とは？

Amazon OpenSearch Service は、 AWS クラウドでの OpenSearch クラスターのデプロイ、運用、スケーリングを容易にするマネージドサービスです。OpenSearch Service ドメインは、OpenSearch クラスターと同義です。ドメインは、指定した設定、インスタンスタイプ、インスタンスカウント、およびストレージリソースを持つクラスターです。Amazon OpenSearch Service は、OpenSearch および従来の Elasticsearch OSS (ソフトウェアのファイナルオープンソースバージョンである 7.10 まで) をサポートしています。ドメインを作成するときに、使用する検索エンジンを選択できます。

OpenSearch はログ分析、リアルタイムのアプリケーションモニタリング、クリックストリーム分析などのユースケース向けの、完全なオープンソースの検索および分析エンジンです。詳細については、「OpenSearch ドキュメント」を参照してください。

Amazon OpenSearch Service は、OpenSearch クラスターのすべてのリソースをプロビジョニングして、OpenSearch クラスターを起動します。また、障害が発生した OpenSearch Service ノードを自動的に検出して置き換え、セルフマネージドインフラストラクチャに関連するオーバーヘッドを減らします。また、単一の API コールを使用するか、コンソールで数回クリックするだけで、クラスターを簡単にスケーリングできます。



OpenSearch Service の使用を開始するには、OpenSearch クラスターに相当する OpenSearch Service ドメインを作成します。クラスター内の各 EC2 インスタンスは、1 つの OpenSearch Service ノードとして機能します。

OpenSearch Service コンソールを使用すると、ドメインを数分でセットアップして設定できます。プログラムによるアクセスを希望する場合は、AWS CLI、AWS SDK、または Terraform を使用できます。

Amazon OpenSearch Service の機能

OpenSearch Service には以下の機能があります。

[Scale] (スケール)

- 費用効率の高い Graviton インスタンスを含む、インスタンスタイプと呼ばれる、CPU、メモリ、ストレージ容量の多数の設定
- 最大 1002 個のデータノードをサポート
- 最大 25 PB のアタッチ済みストレージ
- 読み取り専用データ用のコストパフォーマンスに優れた UltraWarm およびコールドストレージ

セキュリティ

- AWS Identity and Access Management (IAM) アクセスコントロール
- Amazon VPC および VPC セキュリティグループと簡単に統合
- 保管時のデータの暗号化およびノード間の暗号化
- OpenSearch Dashboards の Amazon Cognito 認証、HTTP 基本認証、または SAML 認証
- インデックスレベル、ドキュメントレベル、フィールドレベルのセキュリティ
- 監査ログ
- Dashboards マルチテナンシー

安定性

- リージョンおよびアベイラビリティーゾーンと呼ばれる、リソース用の複数の地理的場所
- マルチ AZ と呼ばれる、同じ AWS リージョン内の 2 つまたは 3 つのアベイラビリティーゾーンにまたがるノード割り当て
- クラスター管理タスクをオフロードする専用マスターノード
- OpenSearch Service ドメインをバックアップおよび復元する自動スナップショット

柔軟性

- ビジネスインテリジェンス (BI) アプリケーションとの統合のための SQL サポート
- 検索結果を改善するためのカスタムパッケージ

人気のあるサービスとの統合

- OpenSearch Dashboards を使用したデータの可視化
- OpenSearch Service ドメインメトリクスのモニタリングとアラームの設定のための Amazon CloudWatch との統合
- OpenSearch Service ドメイン AWS CloudTrail への設定 API コールを監査するための との統合
- OpenSearch サービスへのストリーミングデータのロードのための Amazon S3、Amazon Kinesis、および Amazon DynamoDB との統合
- データが特定のしきい値を超えたときの Amazon SNS からのアラート

OpenSearch と Amazon OpenSearch Service の使い分け

次の表を使用して、プロビジョニングされた Amazon OpenSearch Service とセルフマネージド OpenSearch のどちらが正しい選択であるかを判断します。

OpenSearch	Amazon OpenSearch Service
• 組織には、自己プロビジョニングされたクラスターを手動でモニタリングして維持する意思があり、それを行う適切なスキルを持つ人材がいる。 • コードの完全なコンパイルレベルの制御が必要である。 • 組織は、オープンソースソフトウェアを優先するか、独自に使用する。 • マルチクラウド戦略があり、ベンダー固有ではないテクノロジーが必要である。 • チームは、重要な本番稼働問題に対処できる。 • 製品を自由に使用、変更、拡張できる必要がある。 • 新機能がリリースされたらすぐにアクセスする必要がある。	• インフラストラクチャを手動で管理、モニタリング、保守することを望まない。 • Amazon S3 の耐久性と低コストを活用して、ストレージ階層間でデータを階層化することで、増大する分析コストを管理する簡単な方法が必要。 • DynamoDB、Amazon DocumentDB (MongoDB 互換)、IAM、CloudWatch 、CloudFormation などの他の AWS のサービス との統合を活用したい。 • 予防メンテナンスや本番稼働時の問題発生時に サポート 、 からのサポートに簡単にアクセスできます。 • 自己修復、予防的なメンテナンス、回復力、バックアップなどの機能を活用したい。

サポートされているバージョンの Elasticsearch および OpenSearch

OpenSearch Service は OpenSearch の以下のバージョンをサポートしています。

- 3.5、3.3、3.1、2.19、2.17、2.15、2.13、2.11、2.9、2.7、2.5、2.3、1.3、1.2、1.1、1.0

OpenSearch Service は、レガシー Elasticsearch の以下のバージョンをサポートしています。

- 7.10、7.9、7.8、7.7、7.4、7.1、6.8、6.7、6.5、6.4、6.3、6.2、6.0、5.6、5.5、5.3、5.1、2.3、1.5

価格性能、機能の豊富さ、セキュリティの向上の観点から、OpenSearch Service を最大限に活用するために、利用可能な最新の OpenSearch バージョンへのアップグレードを推奨します。

標準サポートおよび延長サポート

AWS は、標準サポートのバージョンのバグ修正とセキュリティ更新を提供します。延長サポートのバージョンの場合、 は標準サポート終了後少なくとも 12 か月間、正規化されたインスタンス時間 (NIH) あたりの定額料金で重要なセキュリティ修正 AWS を提供します。NIH はインスタンスのサイズと使用時間に基づきます。

延長サポート料金は、ドメインが標準サポートの対象ではなくなったバージョンを実行すると自動的に適用されます。これらの料金を回避するには、サポートされているバージョンにアップグレードしてください。

以下の表に、OpenSearch およびレガシー Elasticsearch バージョンのサポート終了スケジュールを示します。

OpenSearch Service は、複数の OpenSearch バージョンと、レガシーオープンソース Elasticsearch バージョンをサポートしています。一部のバージョンでは、標準サポートの終了と延長サポートの日程が既に公開されています。価格性能、機能の豊富さ、セキュリティの向上の観点から、OpenSearch Service を最大限に活用するために、利用可能な最新の OpenSearch バージョンへのアップグレードを推奨します。以下の表に、Elasticsearch および OpenSearch のバージョンと、そのサポートスケジュールのリストを示します。

Elasticsearch バージョンのサポート終了スケジュールは以下のとおりです。

ソフトウェアのバージョン	標準サポート終了日	延長サポート終了日
Elasticsearch バージョン 1.5 および 2.3	2025 年 11 月 7 日	2026 年 11 月 7 日
Elasticsearch バージョン 5.1 から 5.5	2025 年 11 月 7 日	2026 年 11 月 7 日
Elasticsearch バージョン 5.6	2025 年 11 月 7 日	2028 年 11 月 7 日
Elasticsearch バージョン 6.0 から 6.7	2025 年 11 月 7 日	2026 年 11 月 7 日
Elasticsearch バージョン 6.8	未発表	未発表
Elasticsearch バージョン 7.1 から 7.8	2025 年 11 月 7 日	2026 年 11 月 7 日
Elasticsearch バージョン 7.9	未発表	未発表

ソフトウェアのバージョン	標準サポート終了日	延長サポート終了日
Elasticsearch バージョン 7.10	未発表	未発表

OpenSearch バージョンのサポート終了スケジュールは次のとおりです。

ソフトウェアのバージョン	標準サポート終了日	延長サポート終了日
OpenSearch バージョン 1.0 から 1.2	2025 年 11 月 7 日	2026 年 11 月 7 日
Opensearch バージョン 1.3	未発表	未発表
OpenSearch バージョン 2.3 から 2.9	2025 年 11 月 7 日	2026 年 11 月 7 日
OpenSearch バージョン 2.11 以降のバージョン	未発表	未発表

OpenSearch と Elasticsearch の標準サポートと延長サポート

AWS は、標準サポートの対象となるバージョンの定期的なバグ修正とセキュリティ更新を提供します。延長サポート対象のバージョンについては、 AWS は標準サポート終了から少なくとも 12 か月間、正規化されたインスタンス時間 (NIH) ごとに追加の定額料金で重要なセキュリティ修正を提供します。NIH は、インスタンスサイズ (中、大など) とインスタンス時間数の係数として計算されます (例については、以下の延長サポート料金の計算セクションを参照してください)。延長サポート料金は、ドメインが標準サポートが終了したバージョンを実行している場合に自動的に適用されます。標準サポートの対象となる最新バージョンにアップグレードすることで、延長サポート料金の発生を回避できます。延長サポート料金の詳細については、料金表ページを参照してください。延長サポートの一般情報については、「Extended Support FAQ」を参照してください。

延長サポート料金の計算

延長サポート対象のバージョンを実行しているドメインには、定額追加料金/正規化されたインスタンス時間 (NIH) が課金されます。例えば、米国東部 (バージニア北部) リージョンでは 0.0065 USD です。NIH は、インスタンスサイズ (中、大など) とインスタンス時間数の係数として計算されます。例えば、米国東部 (バージニア北部) リージョンで m7g.medium.search インスタンスを 24 時間実行している場合、料金は 0.068 USD/インスタンス時間 (オンデマンド) で、通常は 1.632 USD (0.068 USD x 24) になります。延長サポート対象のバージョンを実行している場合は、0.0065 USD/NIH を追加で支払います。この料金は、0.0065 USD x 24 (インスタンス時間数) x 2 (サイズ正規化係数、中規模インスタンスの場合は 2) として計算され、24 時間延長サポートの場合は 0.312 USD になります。24 時間の合計支払い額は、標準インスタンスの使用コストと延長サポートコストの合計 1.944 USD (1.632 USD+0.312 USD) になります。以下の表に、OpenSearch Service のさまざまなインスタンスサイズの正規化係数を示します。

インスタンスサイズ	正規化係数	
nano	0.25	
micro	0.5	
small	1	
medium	2	
large	4	

インスタンスサイズ	正規化係数	
xlarge	8	
2xlarge	16	
4xlarge	32	
8xlarge	64	
9xlarge	72	
10xlarge	80	
12xlarge	96	
16xlarge	128	
18xlarge	144	
24xlarge	192	
32xlarge	256	

Amazon OpenSearch Service の料金

OpenSearch Service の場合、1 時間ごとの EC2 インスタンスの使用およびインスタンスにアタッチされたすべての EBS ストレージボリュームの累積サイズに対して料金が発生します。標準 AWS データ転送料金も適用されます。

ただし、いくつかの注意すべきデータ転送の例外があります。ドメインが複数のアベイラビリティーゾーンを使用する場合、OpenSearch Service では 2 つのアベイラビリティゾーン間のトラフィックに対して課金されません。シャードの割り当てと再分散中にドメイン内で大幅なデータ転送が発生した場合、OpenSearch Service はこのトラフィックに対して測定も課金も行いません。同様に、OpenSearch Service では UltraWarm/コールドノードと Amazon S3 の間のデータ転送に対して料金は発生しません。

料金の詳細については、「Amazon OpenSearch Service の料金表」を参照してください。設定変更中に発生する料金については、「the section called “設定変更に関連する料金”」を参照してください。

請求および使用状況レポートの理解

Amazon OpenSearch Service の使用状況は、特定の使用タイプコードを含む AWS 請求レポートに表示されます。これらのコードを理解することで、デプロイタイプ全体のコストを分析できます。

使用タイプコード

- `ESInstance` – マネージド OpenSearch Service ドメインのインスタンス時間。
- `EBS` – ドメインインスタンスにアタッチされた EBS ストレージボリューム。
- `ServerlessOCU` – OpenSearch Serverless コレクションによって消費される OpenSearch コンピューティングユニット。
- `IngestionOCU` – OpenSearch Ingestion パイプラインによって消費される OpenSearch コンピューティングユニット。

使用タイプコードのリージョンの略語

使用タイプコードには、リージョンの略語がプレフィックスとして付けられます。次の表は、一般的な例を示しています。

略語	リージョン
USE1	米国東部 (バージニア北部)
USE2	米国東部 (オハイオ)
USW1	米国西部 (北カリフォルニア)
USW2	米国西部 (オレゴン)
EUW1	欧州 (アイルランド)
EUC1	欧州 (フランクフルト)
APS1	アジアパシフィック (シンガポール)

略語	リージョン
APS2	アジアパシフィック (シドニー)
APN1	アジアパシフィック (東京)

たとえば、 の使用タイプは、米国東部 (バージニア北部) の`r6g.large.search`インスタンス時間`USE1-ESInstance:r6g.large.search`を表します。請求の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

関連サービス

OpenSearch Service は、次のサービスでよく使用されます。

Amazon CloudWatch

ドメインヘルスやパフォーマンスをモニタリングできるように、メトリクスは、OpenSearch Service ドメインから CloudWatch に自動的に送信されます。詳細については、「Amazon CloudWatch を用いた OpenSearch クラスターメトリクスのモニタリング」を参照してください。

CloudWatch Logs は、別の方向に送信することもできます。分析のために OpenSearch Service にデータをストリーミングするように CloudWatch Logs を設定できます。詳細については"the section called “Amazon CloudWatch からストリーミングデータをロードする”"を参照してください。

AWS CloudTrail

を使用して AWS CloudTrail 、アカウントの OpenSearch Service 設定 API コールおよび関連イベントの履歴を取得します。詳細については、「を使用した Amazon OpenSearch Service API コールのモニタリング AWS CloudTrail」を参照してください。

Amazon Kinesis

Kinesis は、大規模なストリーミングデータをリアルタイムで処理するマネージドサービスです。詳細については、「"the section called “Amazon Kinesis Data Streams からストリーミングデータをロードする”"」および「"the section called “Amazon Data Firehose からストリーミングデータをロードする”"」を参照してください。

Amazon S3

Amazon Simple Storage Service (Amazon S3) は、インターネット用のストレージを提供します。このガイドでは、Amazon S3 と統合するための Lambda サンプルコードが提供されています。詳細については、「the section called “Amazon S3 からストリーミングデータをロードする”」を参照してください。

AWS IAM

AWS Identity and Access Management (IAM) は、OpenSearch Service ドメインへのアクセスを管理するために使用できるウェブサービスです。詳細については、「the section called “Identity and Access Management”」を参照してください。

AWS Lambda

AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードを実行できるようにするコンピューティングサービスです。このガイドでは、DynamoDB、Amazon S3、および Kinesis からデータをストリーミングするための Lambda サンプルコードが提供されています。詳細については、「the section called “OpenSearch Service へのストリーミングデータをロードする”」を参照してください。

Amazon DynamoDB

Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。OpenSearch Service へのデータのストリーミングの詳細については、「the section called “Amazon DynamoDB テーブルからストリーミングデータをロードする”」を参照してください。

Amazon Quick

クイックダッシュボードを使用して、OpenSearch Service からデータを視覚化できます。詳細については、「 クイックユーザーガイド」の「Amazon OpenSearch Service とクイックの使用」を参照してください。

> **Note**
>
> OpenSearch には、Elasticsearch B.V. からの特定の Apache ライセンスの Elasticsearch コードやその他のソースコードが含まれています。Elasticsearch B.V. は、他のソースコードのソースではありません。ELASTICSEARCH は Elasticsearch B.V. の登録商標です。

Amazon OpenSearch Service の設定

許可を付与する

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、「 AWS アカウント管理 リファレンスガイド」の「 の開始方法 AWS アカウント」を参照してください。

をインストールして設定する AWS CLI

OpenSearch Service API を使用する場合は、 AWS Command Line Interface (AWS CLI) の最新バージョンをインストールする必要があります。コンソールから OpenSearch Service AWS CLI を使用するには は必要ありません。 の手順に従って CLI を使用せずに開始できますAmazon OpenSearch Service の開始方法。

をセットアップするには AWS CLI

1. macOS、Linux、または Windows AWS CLI 用の の最新バージョンをインストールするには、「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。
2. OpenSearch Service を含む へのアクセスの AWS CLI と安全なセットアップを設定するには AWS のサービス、「 を使用したクイック設定aws configure」を参照してください。
3. 設定を確認するには、コマンドプロンプトで次の DataBrew コマンドを入力します。

   ```
   aws opensearch help
   ```

   AWS CLI コマンドは、パラメータまたはプロファイルで設定しない限り、設定 AWS リージョン のデフォルトを使用します。パラメータ AWS リージョン を使用して を設定するには、各コマンドに `--region`パラメータを追加します。

   プロファイル AWS リージョン で を設定するには、まず ファイルまたは `~/.aws/config` ファイル `%UserProfile%/.aws/config` (Microsoft Windows の場合) に名前付きプロファイルを追加します。「AWS CLI の名前付きプロファイル」のステップに従います。次に、次の例のようなコマンドを使用して、 AWS リージョン およびその他の設定を行います。

```
[profile opensearch]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

コンソールを開きます。

このセクションのコンソール指向のトピックのほとんどは、OpenSearch Service コンソールを起点としています。にまだサインインしていない場合は AWS アカウント、サインインして OpenSearch Service コンソールを開き、次のセクションに進み、OpenSearch Service の使用を開始します。

Amazon OpenSearch Service の開始方法

開始するには、AWS アカウント をまだお持ちでない場合はサインアップします。アカウントをセットアップしたら、Amazon OpenSearch Service の開始方法チュートリアルを使用します。サービスについて学習中に詳しい情報が必要になった場合は、以下の概要トピックを参照してください。

- ドメインを作成する
- ワークロードのドメインを適切にサイズ設定します。
- ドメインアクセスポリシーまたはきめ細かなアクセスコントロールを使用したドメインへのアクセスを制御します。
- 手動で、または他の AWS のサービスからデータをインデックス化します。
- OpenSearch Dashboards を使用してデータを検索し可視化を作成します。
- ドメインを作成するためのアドバンストオプションについて説明します。詳細については、「ドメインの作成と管理」を参照してください。
- ドメイン内のインデックスを管理する方法を見つけてください。詳細については、「インデックス管理」を参照してください。
- Amazon OpenSearch Service で作業するためのチュートリアルの 1 つを試してください。詳細については、「チュートリアル」を参照してください。

セルフマネージド OpenSearch クラスターから OpenSearch Service への移行の詳細については、「the section called “OpenSearch Service への移行”」を参照してください。

詳細については、このガイドの「ドメインの作成と管理」とその他のトピックを参照してください。セルフマネージド OpenSearch クラスターから OpenSearch Service への移行の詳細については、「the section called “OpenSearch Service への移行”」を参照してください。

OpenSearch Service コンソール、、または AWS SDK を使用して AWS CLI、次の手順を実行できます。のインストールとセットアップの詳細については AWS CLI、AWS Command Line Interface「 ユーザーガイド」を参照してください。

Amazon OpenSearch Service ドメインを作成する

> **Important**
>
> これは、テスト Amazon OpenSearch Service ドメインを設定するための簡潔なチュートリアルです。本番稼働用ドメインを作成するには、このプロセスを使用しないでください。同

じプロセスの包括的なバージョンについては、「ドメインの作成と管理」を参照してください。

OpenSearch Service ドメインは、OpenSearch クラスターと同義です。ドメインは、指定した設定、インスタンスタイプ、インスタンスカウント、およびストレージリソースを持つクラスターです。コンソール、、 AWS CLIまたは AWS SDKs を使用して OpenSearch Service ドメインを作成できます。

コンソールを使用して OpenSearch Service ドメインを作成するには

1. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. [ドメインの作成] を選択します。
4. ドメインの名前を入力します。このチュートリアルの例では、movies という名前を使用します。
5. ドメインの作成方法は、[標準作成] を選択します。

> **Note**
>
> ベストプラクティスを使用して本番稼働用ドメインをすばやく設定するには、[簡易作成] を選択します。このチュートリアルの開発とテストには、[標準作成] を使用します。

6. テンプレートは、[開発/テスト] を選択します。
7. デプロイオプションは、[スタンバイが有効のドメイン] を選択します。
8. [Elasticsearch バージョン] で、最新バージョンを選択します。
9. 現時点では、[データノード]、[ウォームデータストレージとコールドデータストレージ]、[専用マスターノード]、[スナップショットの設定]、[カスタムエンドポイント] のセクションは無視してください。デフォルトのデータノードインスタンスタイプ (`r7g.large.search`) には料金が発生することに注意してください。このチュートリアルのコストを最小限に抑える場合は、データノード`t3.small.search`の など、より小さなインスタンスタイプを選択することを検討してください。
10. このチュートリアルを簡単にするために、パブリックアクセスドメインを使用します。[ネットワーク] で [パブリックアクセス] を選択します。

11. きめ細かなアクセスコントロール設定で、[きめ細かなアクセスコントロールを有効化] チェックボックスをオンのままにします。[マスターユーザーの作成] をクリックして、ユーザー名とパスワードを入力します。
12. 今のところ、SAML 認証および Amazon Cognito 認証セクションは無視します。
13. [アクセスポリシー]で [きめ細かなアクセスコントロールのみを使用] を選択します。このチュートリアルでは、ドメインアクセスポリシーではなく、きめ細かなアクセスコントロールが認証を処理します。
14. 残りの設定は無視して、[作成] を選択します。通常、新しいドメインは初期化に 15～30 分かかりますが、設定によってはさらに時間がかかります。ドメインが初期化されたら、ドメインを選択して設定ペインを開きます。次の手順で使用する [一般情報] の下のドメインエンドポイント (例えば、`https://search-my-domain.us-east-1.es.amazonaws.com`) に注意してください。

次: インデックス作成のためにデータを OpenSearch Service ドメインにアップロードする

インデックス作成のために Amazon OpenSearch Service にデータをアップロードする

> **Important**
>
> これは Amazon OpenSearch Service に少量のテストデータをアップロードするための簡潔なチュートリアルです。本番ドメインでのデータのアップロードの詳細については、「データのインデックス作成]を参照してください。

コマンドラインまたはほとんどのプログラミング言語を使用して、OpenSearch Service ドメインにデータをアップロードできます。

次の例のリクエストでは、簡潔にするため、および便宜上、一般的な HTTP クライアントである curl を使用しています。curl などのクライアントは、アクセスポリシーが IAM ユーザーあるいはロールを指定している場合に、必要なリクエスト署名を実行できません。このプロセスを正常に完了するには、ステップ 1 で設定したように、プライマリユーザー名とパスワードを使用してきめ細かなアクセスコントロールを使用する必要があります。

Windows に curl をインストールしてコマンドプロンプトから使用できますが、Cygwin または Windows Subsystem for Linux などのツールをお勧めします。macOS およびほとんどの Linux ディストリビューションでは、curl がプレインストールされています。

オプション 1: 単一のドキュメントをアップロードする

単一のドキュメントを movies ドメインに追加するには、次のコマンドを実行します。

```
curl -XPUT -u 'master-user:master-user-password' 'domain-endpoint/movies/_doc/1' -d
 '{"director": "Burton, Tim", "genre": ["Comedy","Sci-Fi"], "year": 1996, "actor":
 ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"], "title": "Mars Attacks!"}'
 -H 'Content-Type: application/json'
```

コマンドで、ステップ 1 で作成したユーザー名とパスワードを入力します。

このコマンドの詳細および OpenSearch Service への署名付きリクエストを作成する方法の詳細については、「データのインデックス作成」を参照してください。

オプション 2: 複数のドキュメントをアップロードする

複数のドキュメントを含む JSON ファイルを OpenSearch Service ドメインにアップロードするには

1. bulk_movies.json という名前のローカルファイルを作成します。以下のコンテンツをファイルに貼り付けます。末尾の改行も追加します。

   ```
   { "index" : { "_index": "movies", "_id" : "2" } }
   {"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
    "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
    Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
    "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
    Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
    Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
    Manchurian Candidate"}
   { "index" : { "_index": "movies", "_id" : "3" } }
   {"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
    1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
    "Pantoliano, Joe", "Jacob, Ir\u00e8ne", "Nelligan, Kate", "Roebuck, Daniel",
    "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
    "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
    "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
    Charlie"], "title": "U.S. Marshals"}
   ```

```
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch,
 Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson",
 "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon,
 David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
```

2. ファイルが保存されているローカルディレクトリで、次のコマンドを実行して、movies ドメインにそれをアップロードします。

```
curl -XPOST -u 'master-user:master-user-password' 'domain-endpoint/movies/_bulk' --
data-binary @bulk_movies.json -H 'Content-Type: application/x-ndjson'
```

bulk ファイルの形式については、「データのインデックス作成」を参照してください。

次: ドキュメントの検索

Amazon OpenSearch Service でドキュメントを検索する

Amazon OpenSearch Service ドメインでドキュメントを検索するには、OpenSearch 検索 API を使用します。または、OpenSearch Dashboards を使用してドメインのドキュメントを検索することもできます。

コマンドラインからドキュメントを検索する

movies ドメインで mars という単語を検索するには、次のコマンドを実行します。

```
curl -XGET -u 'master-user:master-user-password' 'domain-endpoint/movies/_search?
q=mars&pretty=true'
```

前のページでバルクデータを使用している場合は、代わりに rebel を検索してください。

次のようなレスポンスが表示されます。

```
{
  "took" : 5,
  "timed_out" : false,
  "_shards" : {
```

```
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 0.2876821,
    "hits" : [
      {
        "_index" : "movies",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 0.2876821,
        "_source" : {
          "director" : "Burton, Tim",
          "genre" : [
            "Comedy",
            "Sci-Fi"
          ],
          "year" : 1996,
          "actor" : [
            "Jack Nicholson",
            "Pierce Brosnan",
            "Sarah Jessica Parker"
          ],
          "title" : "Mars Attacks!"
        }
      }
    ]
  }
}
```

OpenSearch Dashboards を使用してドキュメントを検索する

OpenSearch Dashboards は OpenSearch で稼働するように設計された、ポピュラーなオープンソースの可視化ツールです。これは、インデックスを検索してモニタリングするための便利なユーザーインターフェイスを提供します。

Dashboards を使用して OpenSearch Service ドメインからドキュメントを検索するには

1. 使用しているドメインの OpenSearch Dashboards URL に移動します。この URL は、OpenSearch Service コンソールのドメインダッシュボードに表示されています。URL はこの形式に従います。

   ```
   domain-endpoint/_dashboards/
   ```

2. プライマリユーザー名とパスワードを使ってログインします。
3. Dashboards を使用するには、少なくとも 1 つのインデックスパターンを作成する必要があります。Dashboards は、そのパターンを使用して、どのインデックスを分析するかを特定します。左側のナビゲーションパネルを開き、ダッシュボード管理 (以前のバージョンではスタック管理) を選択し、インデックスパターンを選択し、インデックスパターンの作成を選択します。このチュートリアルでは、movies と入力します。
4. [次のステップ] を選択してから、[インデックスパターンの作成] を選択します。パターンが作成されたら、`actor` および `director` などのさまざまなドキュメントフィールドを表示できます。
5. [インデックスパターン] ページに戻り、`movies` がデフォルトとして設定されていることを確認します。そうでない場合は、パターンを選択し、星アイコンを選択してデフォルトにします。
6. データの検索を開始するには、左のナビゲーションパネルをもう一度開き、[検出] を選択します。
7. 検索バーに、単一のドキュメントをアップロードした場合は mars と入力するか、複数のドキュメントをアップロードした場合は rebel と入力してから [Enter] を押します。俳優名や監督名など、他の用語を検索してみることができます。

次: ドメインの削除

Amazon OpenSearch Service ドメインを削除する

このチュートリアルの movies ドメインはテスト用のため、チュートリアルを終了したら、料金が発生しないようにこれを削除してください。

コンソールから OpenSearch Service ドメインを削除するには

1. [Amazon OpenSearch Service] コンソールにサインインします。
2. [ドメイン] で、[movies]ドメインを選択します。

3. [削除] を選択して、削除を確認します。

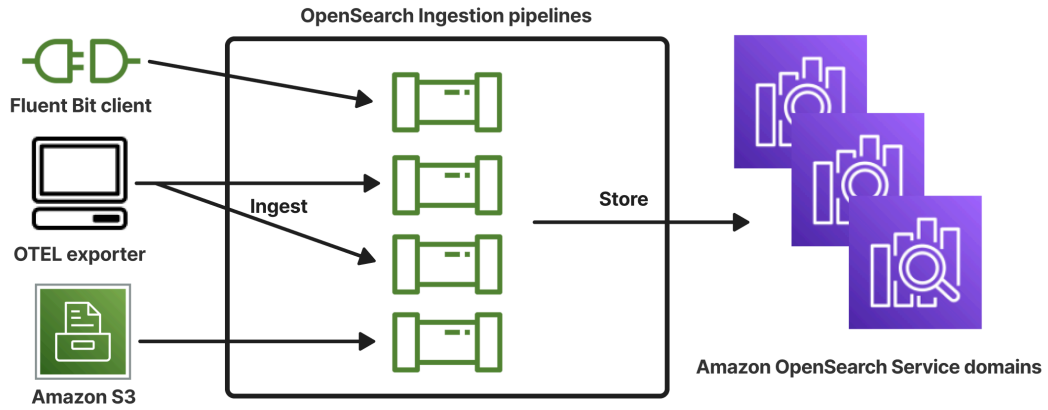
Amazon OpenSearch Ingestion の概要

Amazon OpenSearch Ingestion は、フルマネージド型のサーバーレスデータコレクターです。リアルタイムログ、メトリクス、トレースデータを Amazon OpenSearch Service ドメインと OpenSearch Serverless コレクションにストリーミングします。

OpenSearch Ingestion を使用すると、Logstash や Jaeger などのサードパーティーツールがデータを取り込む必要がなくなります。OpenSearch Ingestion にデータを送信するようにデータプロデューサーを設定すると、指定したドメインまたはコレクションにデータが自動的に配信されます。配信前にデータを変換することもできます。

OpenSearch Ingestion はサーバーレスであるため、インフラストラクチャの管理、ソフトウェアのパッチ適用、クラスターの手動スケーリングを行う必要はありません。取り込みパイプラインは AWS マネジメントコンソールで直接プロビジョニングでき、OpenSearch Ingestion が残りを処理します。

Amazon OpenSearch Service のコンポーネントである OpenSearch Ingestion は、Data Prepper により動作します。Data Prepper は、ダウンストリームの分析と視覚化のためにデータをフィルタリング、強化、変換、正規化、集約するオープンソースのデータコレクターです。



Amazon OpenSearch Ingestion の主な概念

OpenSearch Ingestion の使用を開始する前に、これらの主要な概念を理解しておくと役に立ちます。

パイプライン

OpenSearch Ingestion の観点から見ると、パイプラインとは、OpenSearch Service 内で作成される単一のプロビジョニング済みデータコレクターを指します。これは、1 つ以上のサブパイプ

ラインを含む YAML 設定ファイル全体と考えることができます。取り込みパイプラインを作成する手順については、「the section called “パイプラインの作成”」を参照してください。

サブパイプライン

サブパイプラインは、YAML 設定ファイル内で定義します。各サブパイプラインは、ソース、バッファ、0 個以上のプロセッサ、1 個以上のシンクの組み合わせです。1 つの YAML ファイルで複数のサブパイプラインを定義することができます。各サブパイプラインには固有のソース、プロセッサ、シンクがあります。CloudWatch やその他のサービスによるモニタリングを容易にするために、パイプライン名は、すべてのサブパイプラインと異なる名前を指定することをお勧めします。

1 つ目のサブパイプラインのソースが 2 つ目のサブパイプラインで、シンクが 3 つ目のサブパイプラインとなるように、複数のサブパイプラインを 1 つの YAML ファイル内に紐づけることができます。例については、the section called “OpenTelemetry Collector”を参照してください。

ソース

サブパイプラインの入力コンポーネントです。パイプラインがレコードを使用するメカニズムを定義します。ソースは、HTTPS 経由でイベントを受信するか、Amazon S3 などの外部エンドポイントから読み取ることでイベントを使用できます。ソースには、プッシュ型とプル型の 2 種類があります。HTTP や OTel logs などのプッシュ型のソースは、レコードを取り込みエンドポイントにストリーミングします。OTel trace や S3 などのプル型のソースは、ソースからデータを取得します。

Processors

レコードをシンクに発行する前に、レコードを目的の形式にフィルタリング、変換、エンリッチ化できる中間処理ユニットです。プロセッサはパイプラインのオプションコンポーネントです。プロセッサを定義しない場合、レコードはソースで定義されている形式で発行されます。複数のプロセッサを使用することができます。パイプラインは、定義した順序でプロセッサを実行します。

シンク

サブパイプラインの出力コンポーネントです。サブパイプラインがレコードを発行する 1 つ以上の宛先を定義します。OpenSearch Ingestion は、OpenSearch Service ドメインをシンクとしてサポートします。また、サブパイプラインをシンクとしてサポートします。これは、単一の OpenSearch Ingestion パイプライン (YAML ファイル) 内で複数のサブパイプラインをまとめて連結できることを意味します。セルフマネージド型の OpenSearch クラスターは、シンクとしてサポートされていません。

バッファ

プロセッサの一部で、ソースとシンクの間のレイヤーとして機能します。パイプライン内のバッファを手動で設定することはできません。OpenSearch Ingestion はデフォルトのバッファ設定を使用します。

ルート

プロセッサの一部で、パイプラインの作成者が、特定の条件に一致するイベントのみを異なるシンクに送信できるようにします。

有効なサブパイプラインの定義には、ソースとシンクが含まれている必要があります。こうしたパイプラインの各要素の詳細については、「設定リファレンス」を参照してください。

Amazon OpenSearch Ingestion の利点

OpenSearch Ingestion には次の利点があります。

- セルフプロビジョニングされたパイプラインを手動で管理する必要がなくなります。
- 定義したキャパシティ制限に基づいてパイプラインを自動的にスケーリングします。
- セキュリティパッチとバグに対するパッチで、パイプラインを最新の状態に維持します。
- オプションで、パイプラインを仮想プライベートクラウド (VPC) に接続し、セキュリティレイヤーを追加できます。
- コストを抑えるために、パイプラインの停止および開始が可能です。
- 一般的なユースケース用のパイプライン設定のブループリントを提供しているため、すみやかに起動して実行できます。
- さまざまな AWS SDKs と OpenSearch Ingestion API を使用して、パイプラインとプログラムでやり取りできます。
- Amazon CloudWatch でのパフォーマンスモニタリングと CloudWatch Logs のエラーロギングをサポートしています。

Amazon OpenSearch Ingestion の制限事項

OpenSearch Ingestion には次の制約事項があります。

- OpenSearch 1.0 以降、または Elasticsearch 6.8 以降を実行しているドメインにのみデータを取り込むことができます。OTel trace ソースを使用している場合は、OpenSearch Dashboards プラグインを使用できるように、Elasticsearch 7.9 以降を使用することをお勧めします。
- パイプラインが VPC 内の OpenSearch Service ドメインに書き込む場合、パイプラインはドメイン AWS リージョン と同じ に作成する必要があります。
- パイプラインの定義内で設定できるデータソースは 1 つのみです。
- セルフマネージド型の OpenSearch クラスターをシンクとして指定することはできません。
- カスタムエンドポイントをシンクとして指定することはできません。カスタムエンドポイントが有効になっているドメインへの書き込みは可能ですが、標準エンドポイントを指定する必要があります。
- オプトインリージョン内のリソースをソースまたはシンクとして指定することはできません。
- パイプライン設定に含めることができるパラメータにはいくつかの制約があります。詳細については、「the section called “設定の要件と制限”」を参照してください。

サポートされている Data Prepper のバージョン

OpenSearch Ingestion は現在、次の Data Prepper のメジャーバージョンをサポートしています。

- 2.x

Code Editor を使用してパイプラインを作成するときは、必要な `version` オプションを用いて、使用する Data Prepper のメジャーバージョンを指定します。例えば、`version: "2"`。OpenSearch Ingestion は、メジャーバージョンのサポートされている最新マイナーバージョンを取得し、そのバージョンでパイプラインをプロビジョニングします。

Code Editor を使用してパイプラインを作成しない場合、OpenSearch Ingestion はサポートされている最新バージョンでパイプラインを自動的にプロビジョニングします。

現在、OpenSearch Ingestion は Data Prepper のバージョン 2.7 でパイプラインをプロビジョニングします。詳細については、「2.7 release notes」を参照してください。OpenSearch Ingestion では、特定のメジャーバーションのすべてのマイナーバージョンをサポートしているわけではありません。

パイプラインの設定を更新すると、新しいマイナーバージョンの Data Prepper がサポートされている場合、OpenSearch Ingestion は、パイプライン設定で指定されたメジャーバージョンの、サポートされている最新のマイナーバージョンに、パイプラインを自動でアップグレードします。例えば、パイプラインの設定に `version: "2"` があり、OpenSearch Ingestion がまず、バージョン 2.6.0 の

パイプラインをプロビジョニングしたとします。バージョン 2.7.0 のサポートが追加され、パイプライン設定に変更を加えると、OpenSearch Ingestion はパイプラインをバージョン 2.7.0 にアップグレードします。このプロセスにより、パイプラインは、最新のバグ修正とパフォーマンスの改善が施されて最新状態に保たれます。OpenSearch Ingestion は、パイプライン設定内の `version` オプションを手動で変更しない限り、パイプラインのメジャーバージョンを更新することはできません。詳細については、「the section called “パイプラインの更新”」を参照してください。

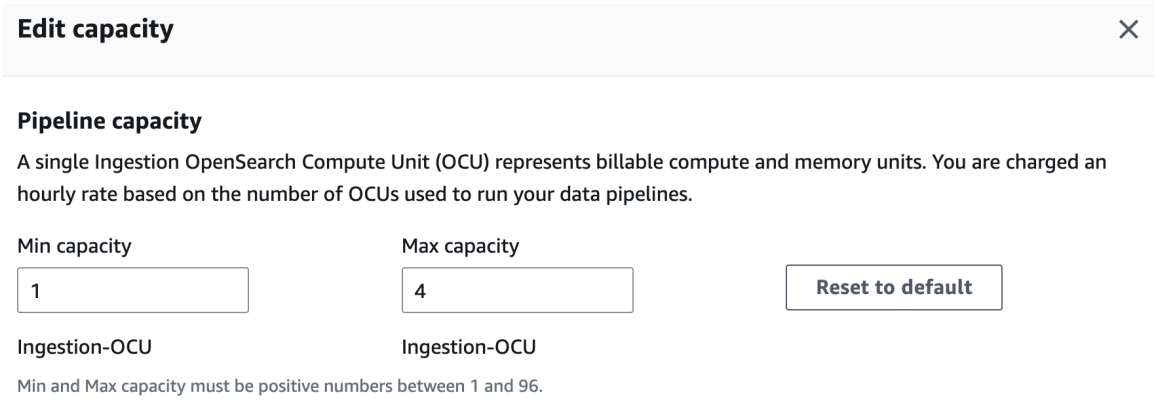
Amazon OpenSearch Ingestion でのパイプラインのスケーリング

OpenSearch Ingestion は、パイプラインのキャパシティを自動的にスケーリングします。これは、指定した Ingestion OpenSearch Compute Unit (Ingestion OCU) の最小値と最大値に基づいて行われます。これにより、手動のプロビジョニングと管理が不要になります。

各取り込み OCU は、約 15 GiB のメモリと 2 つの vCPUs。パイプラインの OCU の最小値と最大値を指定でき、これらの制限に基づいて、OpenSearch Ingestion はパイプラインのキャパシティを自動的にスケーリングします。

パイプラインを作成するときに、次の値を指定します。

- 最小キャパシティ - パイプラインは、この数値の Ingestion OCU までキャパシティを減らすことができます。指定された最小キャパシティは、パイプラインの開始キャパシティでもあります。
- 最大キャパシティ - パイプラインは、この数値の Ingestion OCU までキャパシティを増やすことができます。



パイプラインの最大キャパシティがワークロードの急増に対処できる程度に十分な大きさであること、およびパイプラインがビジー状態でないときに、最小キャパシティがコストを最小限に抑えるのに十分な程度に低い値であるようにします。OpenSearch Serverless は、設定に基づいてパイプライ

ンの Ingestion OCU 数を自動的にスケーリングし、取り込みワークロードを処理します。いつの時点においても、課金されるのはパイプラインで実際に使用している Ingestion OCU 分のみです。

OpenSearch Ingestion パイプラインに割り当てられるキャパシティは、パイプラインの処理要件と、クライアントアプリケーションによって生成された負荷に応じてスケールアップおよびスケールダウンされます。キャパシティが限られている場合、OpenSearch Ingestion はより多くのコンピューティングユニット (GiB のメモリ) を割り当てることでスケールアップします。パイプラインが処理するワークロードが小さい場合や、データをまったく処理しない場合は、設定された最小限の Ingestion OCU までスケールダウンできます。

> **Note**
>
> OpenSearch Ingestion パイプラインは、取り込み OCU あたり最大 150 個のシャードを処理します。パイプラインソースに設定された OCU 数で許可されているシャードよりも多くのシャードが含まれている場合は、最大 OCU 容量を増やすか、ソース内のシャード数を減らすことを検討してください。

指定できる Ingestion OCU は最小 1 個で、最大ではステートレスパイプラインで 96 個、ステートフルパイプラインで 48 個です。プッシュ型のソースの場合、最小の Ingestion OCU は少なくとも 2 個をお勧めします。永続的バッファリングが有効な場合、最小 2 個および最大 384 個の取り込み OCU を指定できます。

単一のソース、シンプルな Grok パターン、シンクを持つ標準のログパイプラインを前提とする場合、各コンピューティングユニットは 1 秒あたり最大 2 MiB をサポートできます。複数のプロセッサを持つ複雑なログパイプラインの場合、各コンピューティングユニットがサポートする取り込み負荷は少なくなります。パイプラインのキャパシティとリソースの使用率に基づいて、OpenSearch Ingestion のスケーリングプロセスが開始されます。

高可用性を確保するために、Ingestion OCU はアベイラビリティーゾーン (AZ) に分散されます。AZ の数は、指定した最小キャパシティに依存します。

例えば、最低 2 つのコンピューティングユニットを指定した場合、任意の時点で使用される Ingestion OCU は 2 つの AZ に均等に分散されます。コンピューティングユニットを 3 つ以上指定すると、Ingestion OCU は 3 つの AZ に均等に分散されます。取り込みパイプラインの可用性が 99.9% になるように、少なくとも 2 つの Ingestion OCU をプロビジョニングすることをお勧めします。

パイプラインのステータスが `Create failed`、`Creating`、`Deleting`、`Stopped` の場合は、Ingestion OCU の料金は請求されません。

パイプラインのキャパシティ設定を行って取得する手順については、「the section called “パイプラインの作成”」を参照してください。

OpenSearch Ingestion の料金

パイプラインを流れるデータがあるかどうかにかかわらず、特定の時点で、パイプラインに割り当てられた Ingestion OCU の数に基づく料金のみをお支払いいただきます。OpenSearch Ingestion は、使用状況に応じてパイプラインのキャパシティをスケールアップまたはスケールダウンすることで、ワークロードに即座に対応します。

料金の詳細については、「Amazon OpenSearch Service の料金表」を参照してください。

サポートされている AWS リージョン

OpenSearch Ingestion は、OpenSearch Service AWS リージョン が利用可能な のサブセットで使用できます。サポートされているリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon OpenSearch Service エンドポイントとクォータ」を参照してください。

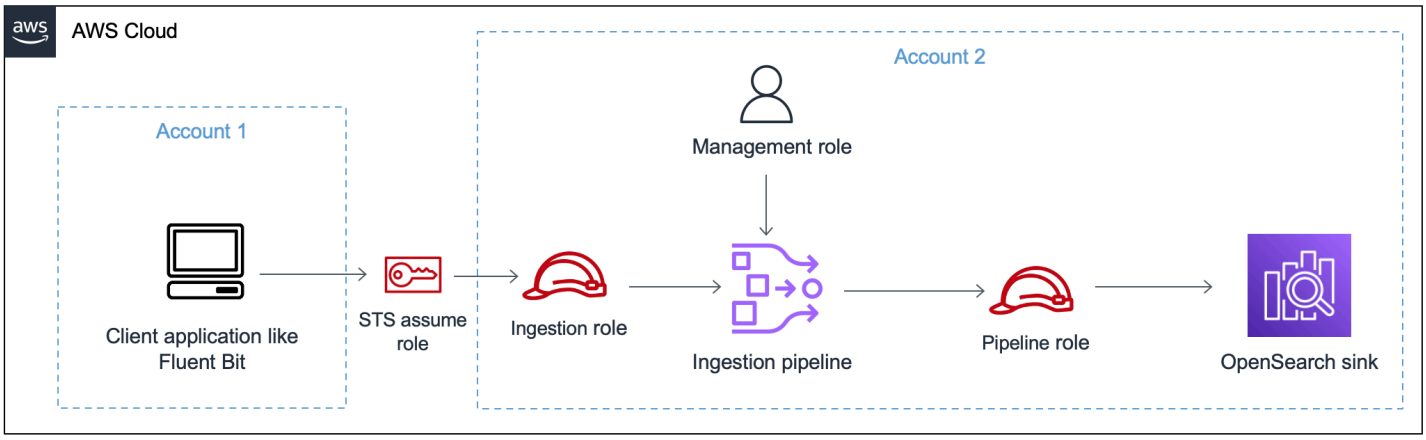
Amazon OpenSearch Ingestion のロールとユーザーの設定

Amazon OpenSearch Ingestion では、ソースアプリケーションからパイプラインへの書き込みと、パイプラインからシンクへの書き込みを行えるように、さまざまなアクセス許可モデルと IAM ロールを使用しています。データの取り込みを開始する前に、特定のアクセス許可を持つ IAM ロールをユースケースに基づいて 1 つ以上作成する必要があります。

パイプラインを正常に設定するには、少なくとも次のロールが必要です。

名前	説明
パイプラインロール	パイプラインロールは、パイプラインがソースから読み取り、ドメインまたはコレクションシンクに書き込むために必要なアクセス許可を付与します。パイプラインロールは手動で作成することも、OpenSearch Ingestion で作成することもできます。
取り込みロール	取り込みロールには、パイプラインリソースの `osis:Ingest` 許可が含まれています。これにより、プッシュ型のソースはパイプラインにデータを取り込むことができます。

次の図は、Amazon S3 や Fluent Bit などのデータソースから別のアカウントのパイプラインに書き込むときの一般的なパイプライン設定を示しています。この場合、パイプラインにアクセスするには、クライアントが取り込みロールを引き受けている必要があります。詳細については、「the section called “クロスアカウント取り込み”」を参照してください。



簡単な設定ガイドについては、「the section called “チュートリアル: ドメインにデータを取り込む”」を参照してください。

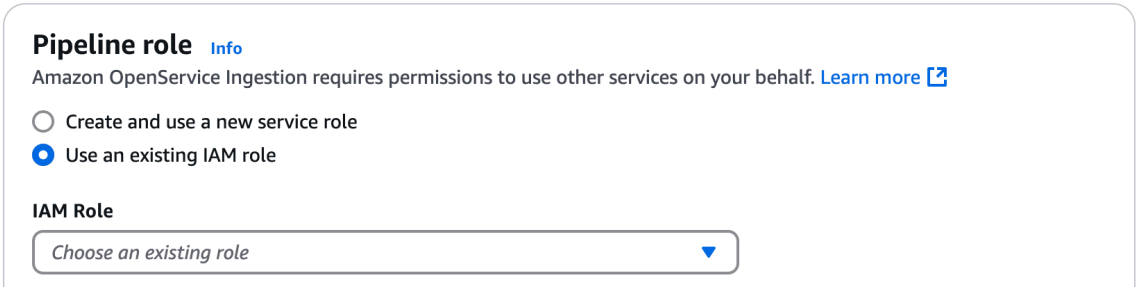
トピック

- the section called “パイプラインロール”
- the section called “取り込みロール”
- the section called “クロスアカウント取り込み”

パイプラインロール

パイプラインは、ソースから読み込み、シンクに書き込むために特定のアクセス許可が必要です。これらのアクセス許可は、クライアントアプリケーションまたはパイプラインに書き込む AWS のサービス 、およびシンクが OpenSearch Service ドメイン、OpenSearch Serverless コレクション、または Amazon S3 OpenSearch のいずれであるかによって異なります。さらに、パイプラインには、ソースアプリケーションから物理的にデータをプルするための権限 (ソースがプルベースのプラグインの場合)、および S3 デッドレターキュー (有効化されている場合) に書き込むための権限が必要になる場合があります。

パイプラインを作成するときは、手動で作成した既存の IAM ロールを指定するか、選択したソースとシンクに基づいて OpenSearch Ingestion にパイプラインロールを自動的に作成させることができます。次の図は、 AWS マネジメントコンソールでパイプラインロールを指定する方法を示しています。



トピック

- パイプラインロールの作成の自動化
- パイプラインロールの手動作成

パイプラインロールの作成の自動化

OpenSearch Ingestion でパイプラインロールを作成するように選択できます。設定されたソースとシンクに基づいて、ロールに必要なアクセス許可が自動的に識別されます。`OpenSearchIngestion-` というプレフィックスと、入力したサフィックスを持つ IAM ロールを作成します。たとえば、サフィックスとして `PipelineRole` と入力すると、OpenSearch Ingestion は `OpenSearchIngestion-PipelineRole` という名前のロールを作成します。

パイプラインロールを自動で作成すると、セットアッププロセスが簡素化され、設定エラーの発生可能性が軽減されます。ロールの作成を自動化することで、アクセス許可を手動で割り当てる必要がなくなり、セキュリティの誤設定のリスクなしに正しいポリシーが適用されます。これにより時間を節約でき、ベストプラクティスを適用することでセキュリティコンプライアンスが強化され、複数のパイプラインデプロイ間で一貫性が確保されます。

OpenSearch Ingestion のみが AWS マネジメントコンソールでパイプラインロールを自動的に作成できます。 AWS CLI、OpenSearch Ingestion API、またはいずれかの SDKs を使用している場合は、手動で作成したパイプラインロールを指定する必要があります。

OpenSearch でロールを作成するには、[新しいサービスロールを作成して使用] を選択します。

> **Important**
>
> パイプラインロールへのアクセスを許可するには、ドメインまたはコレクションアクセスポリシーを手動で変更する必要があります。詳細なアクセスコントロールを使用するドメインの場合は、パイプラインロールをバックエンドロールにマッピングする必要があります。これらのステップは、パイプラインの作成前または作成後に実行できます。

手順については、次のトピックを参照してください。

- Configure data access for the domain
- Configure data and network access for the collection

パイプラインロールの手動作成

特定のセキュリティまたはコンプライアンス要件を満たすためにアクセス許可をより細かく制御する必要がある場合は、パイプラインロールを手動で作成することをお勧めします。手動作成では、既存のインフラストラクチャまたはアクセス管理戦略に合わせてロールを調整できます。また、手動セットアップを選択して、ロールを他の と統合 AWS のサービス したり、独自の運用ニーズに合わせて調整したりできます。

手動で作成したパイプラインロールを選択するには、[既存の IAM ロールを使用] を選択し、既存のロールを選択します。ロールには、選択したソースからデータを受信し、選択したシンクに書き込むために必要なすべてのアクセス許可が必要です。以下のセクションでは、パイプラインロールを手動で作成する方法について説明します。

トピック

- ソースから読み取るためのアクセス許可
- ドメインシンクに書き込むためのアクセス許可
- コレクションシンクに書き込むためのアクセス許可
- Amazon S3 またはデッドレターキューに書き込むためのアクセス許可

ソースから読み取るためのアクセス許可

OpenSearch Ingestion パイプラインには、指定されたソースからデータを読み取って受信するためのアクセス許可が必要です。例えば、Amazon DynamoDB ソースの場合、`dynamodb:DescribeTable` や `dynamodb:DescribeStream` などのアクセス許可が必要です。Amazon S3、Fluent Bit、OpenTelemetry Collector などの一般的なソースのパイプラインロールアクセスポリシーの例については、「the section called “パイプラインの統合”」を参照してください。

ドメインシンクに書き込むためのアクセス許可

OpenSearch Ingestion パイプラインには、シンクとして設定されている OpenSearch Service ドメインに書き込むためのアクセス許可が必要です。これらのアクセス許可には、ドメインを記述して、そ

こに HTTP リクエストを送信できることが含まれます。これらのアクセス許可は、パブリックパイプラインと VPC ドメインのどちらも同じです。パイプラインロールを作成し、ドメインアクセスポリシーで指定する手順については、「Allowing pipelines to access domains」を参照してください。

コレクションシンクに書き込むためのアクセス許可

シンクとして設定されている OpenSearch Serverless コレクションに書き込みを行うときは、OpenSearch Ingestion パイプラインにアクセス権限が必要になります。これらのアクセス権限には、コレクションを記述しそこに HTTP リクエストを送信できることが含まれます。

まず、パイプラインロールのアクセスポリシーが必要なアクセス許可を付与していることを確認します。次に、このロールをデータアクセスポリシーに追加し、インデックスの作成、インデックスの更新、インデックスの記述、コレクション内でのドキュメントの書き込みを行うためのアクセス権限をそれに付与します。これらの各ステップを完了する手順については、「パイプラインにコレクションへのアクセスを許可する」を参照してください。

Amazon S3 またはデッドレターキューに書き込むためのアクセス許可

パイプラインのシンク送信先として Amazon S3 を指定する場合、またはデッドレターキュー (DLQ) を有効にする場合、パイプラインロールは送信先として指定した S3 バケットへのアクセスを許可する必要があります。

DLQ アクセスを提供するパイプラインロールに別の許可ポリシーをアタッチします。少なくとも、ロールにはバケットリソースに対する `S3:PutObject` アクションが付与されている必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "WriteToS3DLQ",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::my-dlq-bucket/*"
    }
  ]
}
```

取り込みロール

取り込みロールは、外部サービスが OpenSearch Ingestion パイプラインと安全にやり取りしてデータを送信できるようにする IAM ロールです。Amazon Security Lake などのプッシュベースのソースの場合、このロールは、`osis:Ingest` を含むパイプラインにデータをプッシュするためのアクセス許可を付与する必要があります。Amazon S3 などのプルベースのソースの場合、ロールは OpenSearch Ingestion がロールを引き受け、必要なアクセス許可でデータにアクセスできるようにする必要があります。

トピック

- プッシュベースのソース用の取り込みロール
- プルベースのソース用の取り込みロール
- クロスアカウント取り込み

プッシュベースのソース用の取り込みロール

プッシュベースのソースの場合、データは Amazon Security Lake や Amazon DynamoDB などの別のサービスから取り込みパイプラインに送信またはプッシュされます。このシナリオでは、取り込みロールには、少なくともパイプラインを操作するための `osis:Ingest` アクセス許可が必要です。

次の IAM アクセスポリシーは、このアクセス許可を取り込みロールに付与する方法を示しています。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "osis:Ingest"
      ],
      "Resource": "arn:aws:osis:us-east-1:111122223333:pipeline/pipeline-name/*"
    }
  ]
}
```

プルベースのソース用の取り込みロール

プルベースのソースの場合、OpenSearch Ingestion パイプラインは Amazon S3 などの外部ソースからデータをアクティブにプルまたは取得します。この場合、パイプラインはデータソースにアクセスするために必要なアクセス許可を付与する IAM パイプラインロールを引き受ける必要があります。これらのシナリオでは、取り込みロールはパイプラインロールと同義です。

ロールには、OpenSearch Ingestion がロールを引き受けられるようにする信頼関係と、データソースに固有のアクセス許可を含める必要があります。詳細については、「the section called “ソースから読み取るためのアクセス許可”」を参照してください。

クロスアカウント取り込み

アプリケーションアカウントなど AWS アカウント、別のパイプラインにデータを取り込む必要がある場合があります。クロスアカウント取り込みを設定するには、パイプラインと同じアカウント内で取り込みロールを定義し、その取り込みロールとアプリケーションアカウント間に信頼関係を確立します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::444455556666:root"
      },
     "Action": "sts:AssumeRole"
  }]
}
```

次に、取り込みロールを引き受けるようにアプリケーションを設定します。アプリケーションアカウントは、パイプラインアカウントの取り込みロールに対する AssumeRole 許可を、アプリケーションロールに付与する必要があります。

詳細な手順と IAM ポリシーの例については、「the section called “クロスアカウント取り込みアクセスの提供”」を参照してください。

Amazon OpenSearch Ingestion パイプラインにドメインへのアクセスを付与する

Amazon OpenSearch Ingestion パイプラインが、シンクとして設定されている OpenSearch Service ドメインに書き込みを行うには、アクセス許可が必要です。アクセスを提供するには、パイプラインがデータを送信しているドメインへのアクセスを制限する制限付きアクセス許可ポリシーで AWS Identity and Access Management (IAM) ロールを設定します。例えば、このユースケースをサポートするために必要なドメインとインデックスのみに、取り込みパイプラインを制限するとします。

> **Important**
>
> パイプラインロールを手動で作成することも、パイプラインの作成時に OpenSearch Ingestion に作成させることもできます。自動ロール作成を選択した場合、OpenSearch Ingestion は、選択したソースとシンクに基づいて、パイプラインロールのアクセスポリシーに必要なすべてのアクセス許可を追加します。プレフィックス `OpenSearchIngestion-` と入力したサフィックスを使用して、IAM にパイプラインロールを作成します。詳細については、「the section called “パイプラインロール”」を参照してください。
>
> OpenSearch Ingestion でパイプラインロールを作成している場合は、そのロールをドメインアクセスポリシーに含め、パイプラインの作成前または作成後にバックエンドロール (ドメインが詳細なアクセスコントロールを使用している場合) にマッピングする必要があります。手順については、ステップ 2 を参照してください。

トピック

- ステップ 1: パイプラインロールを作成する
- ステップ 2: ドメインのデータアクセスを設定する

ステップ 1: パイプラインロールを作成する

パイプラインロールには、データをドメインシンクへ送信できるようにするアクセス許可ポリシーをアタッチしてある必要があります。また、OpenSearch Ingestion にこのロールの引き受けを許可する信頼関係も必要になります。ポリシーをロールにアタッチする方法については、「IAM ユーザーガイド」の「IAM ID アクセス許可の追加」を参照してください。

次のポリシー例では、単一ドメインへの書き込みを可能にするパイプラインで提供可能な最小特権が付与されています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "es:DescribeDomain",
            "Resource": "arn:aws:es:*:111122223333:domain/*"
        },
        {
            "Effect": "Allow",
            "Action": "es:ESHttp*",
            "Resource": "arn:aws:es:*:111122223333:domain/domain-name/*"
        }
    ]
}
```

ロールを再利用して複数のドメインに書き込む場合は、ドメイン名をワイルドカード文字 (*) に置き換えることでポリシーの範囲を広げることができます。

このロールには次の信頼関係が必要です。この信頼関係により、OpenSearch Ingestion はパイプラインのロールを引き受けることができます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"osis-pipelines.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ステップ 2: ドメインのデータアクセスを設定する

パイプラインがドメインにデータを書き込むには、sts_role_arn パイプラインロールにドメインへのアクセスを許可するドメインレベルのアクセスポリシーが、このドメインに必要になります。

次のドメインアクセスポリシーの例では、pipeline-role と言う名前のパイプラインロールに、ingestion-domain と言う名前のドメインへのデータの書き込みが許可されています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/pipeline-role"
            },
            "Action": [
                "es:DescribeDomain",
                "es:ESHttp*"
            ],
            "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
        }
    ]
}
```

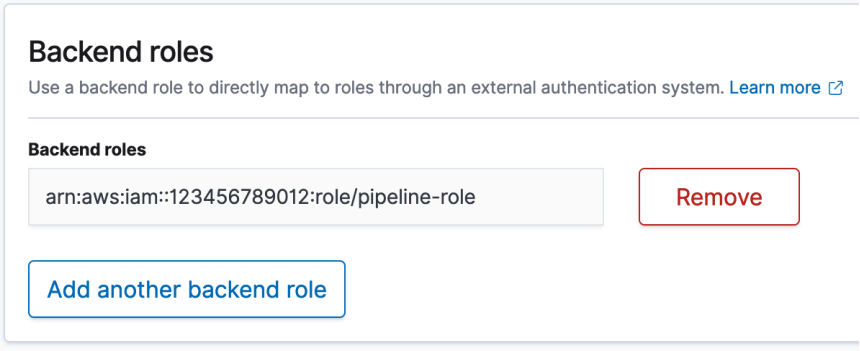
パイプラインロールをマッピングする (詳細なアクセスコントロールを使用するドメインについてのみ)

ドメインが、認証用に詳細なアクセス制御を使用している場合は、パイプラインにドメインへのアクセスを許可するには、追加の手順が必要になります。この手順は、ドメインの設定によって異なります。

- シナリオ 1: マスターロールとパイプラインロールが異なる – IAM Amazon リソースネーム (ARN) をマスターユーザーとして使用し、これが、パイプラインロールとは異なる場合、パイプラインロールを OpenSearch all_access バックエンドロールにマッピングする必要があります。これにより、パイプラインロールが追加のマスターユーザーとして追加されます。詳細については、「追加のマスターユーザー」を参照してください。

- シナリオ 2: 内部ユーザーデータベースのマスターユーザー — ドメインで内部ユーザーデータベースのマスターユーザーと、OpenSearch Dashboards の HTTP 基本認証を使用している場合、マスターユーザー名とパスワードを、パイプライン設定に直接渡すことはできません。代わりに、パイプラインロールを OpenSearch `all_access` バックエンドロールにマッピングします。これにより、パイプラインロールが追加のマスターユーザーとして追加されます。詳細については、「追加のマスターユーザー」を参照してください。
- シナリオ 3: マスターロールとパイプラインロールが同じ (まれなシナリオ) – IAM ARN をマスターユーザーとして使用し、パイプラインロールとして使用している ARN がこれと同じである場合、追加の操作は必要ありません。このパイプラインは、ドメインへの書き込みに必要なアクセス許可がすでにあります。ほとんどの環境では、管理者ロールまたは他のロールをマスターロールとして使用するため、これはまれなシナリオです。

次の図は、パイプラインのロールをバックエンドロールにマッピングする方法を示したものです。



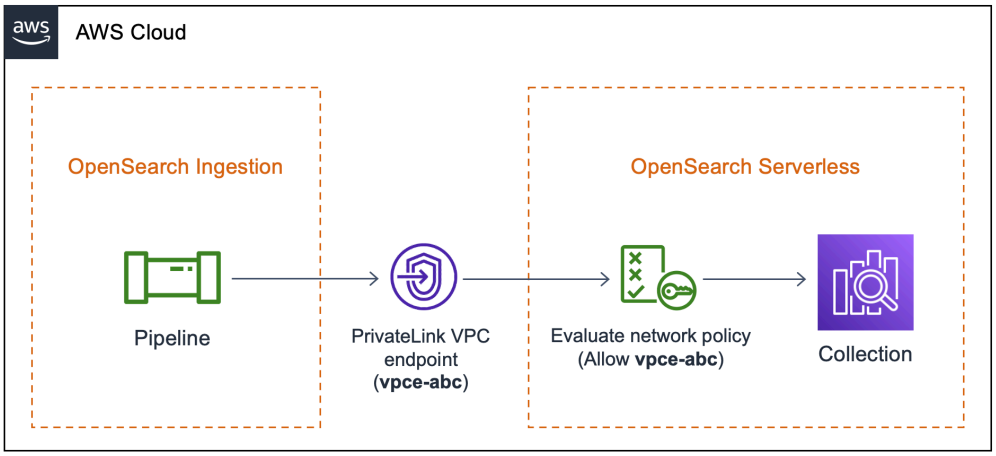
Amazon OpenSearch Ingestion パイプラインにコレクションへのアクセス権を付与する

Amazon OpenSearch Ingestion パイプラインは、OpenSearch Serverless パブリックコレクションまたは VPC コレクションに書き込むことができます。コレクションへのアクセスを提供するには、コレクションへのアクセスを許可するアクセス許可ポリシーを持つ AWS Identity and Access Management (IAM) パイプラインロールを設定します。パイプラインは、OpenSearch Serverless コレクションシンクへのリクエストに署名するために、このロールを引き受けます。

> **Important**
>
> パイプラインロールを手動で作成することも、パイプラインの作成時に OpenSearch Ingestion に作成させることもできます。自動ロール作成を選択した場合、OpenSearch Ingestion は、選択したソースとシンクに基づいて、パイプラインロールのアクセスポリシーに必要なすべてのアクセス許可を追加します。プレフィックス `OpenSearchIngestion-`

と入力したサフィックスを使用して、IAM にパイプラインロールを作成します。詳細については、「the section called “パイプラインロール”」を参照してください。
OpenSearch Ingestion でパイプラインロールを作成している場合は、パイプラインの作成前または作成後に、コレクションのデータアクセスポリシーにロールを含める必要があります。手順については、ステップ 2 を参照してください。

パイプラインの作成中に、OpenSearch Ingestion はパイプラインと OpenSearch Serverless コレクション間の AWS PrivateLink 接続を作成します。パイプラインからのすべてのトラフィックはこの VPC エンドポイントを通過し、コレクションにルーティングされます。コレクションに到達するには、エンドポイントにネットワークアクセスポリシーを介してコレクションへのアクセスを付与する必要があります。



トピック

- ステップ 1: パイプラインロールを作成する
- ステップ 2: コレクションのデータとネットワークアクセスを設定する

ステップ 1: パイプラインロールを作成する

パイプラインロールには、コレクションシンクへのデータ送信を許可するアクセス許可ポリシーがアタッチされている必要があります。また、OpenSearch Ingestion にこのロールの引き受けを許可する信頼関係も必要になります。ポリシーをロールにアタッチする方法については、「IAM ユーザーガイド」の「IAM ID アクセス許可の追加」を参照してください。

次のポリシー例では、コレクションへの書き込みを可能にするパイプラインロールアクセスポリシーで提供できる最小特権が付与されています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll",
                "aoss:BatchGetCollection",
                "aoss:CreateSecurityPolicy",
                "aoss:GetSecurityPolicy",
                "aoss:UpdateSecurityPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

このロールには次の信頼関係が必要です。この信頼関係により、OpenSearch Ingestion はこのロールを引き受けることができます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "osis-pipelines.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

ステップ 2: コレクションのデータとネットワークアクセスを設定する

OpenSearch Serverless コレクションを以下の設定で作成します。コレクションの作成手順については、「the section called “コレクションの作成”」を参照してください。

データアクセスポリシー

パイプラインロールに必要なアクセス許可を付与するコレクション用のデータアクセスポリシーを作成します。例えば、次のようになります。

```
[
  {
    "Rules": [
      {
        "Resource": [
          "index/collection-name/*"
        ],
        "Permission": [
          "aoss:CreateIndex",
          "aoss:UpdateIndex",
          "aoss:DescribeIndex",
          "aoss:WriteDocument"
        ],
        "ResourceType": "index"
      }
    ],
    "Principal": [
      "arn:aws:iam::account-id:role/pipeline-role"
    ],
    "Description": "Pipeline role access"
  }
]
```

> **Note**
>
> Principal 要素で、パイプラインのロールの Amazon リソースネーム (ARN) を指定します。

ネットワークアクセスポリシー

OpenSearch Serverless で作成する各コレクションには、少なくとも 1 つのネットワークアクセスポリシーが関連付けられています。ネットワークアクセスポリシーでは、コレクションがインターネット経由でパブリックネットワークからアクセス可能か、プライベートでアクセスする必要があるかを決定します。ネットワークポリシーの詳細については、「the section called “ネットワークアクセス”」を参照してください。

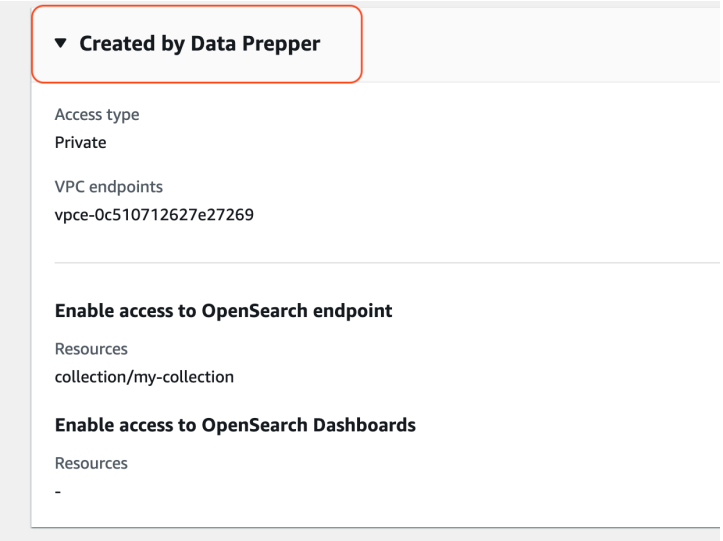
ネットワークアクセスポリシー内では、OpenSearch Serverless マネージド VPC エンドポイントのみを指定できます。詳細については、「the section called “データプレーン VPC エンドポイント”」を参照してください。ただし、パイプラインがコレクションに書き込むには、OpenSearch Ingestion がパイプラインとコレクションの間に自動的に作成する VPC エンドポイントへのアクセスもポリシーで付与する必要があります。したがって、パイプラインの送信先シンクとして OpenSearch Serverless コレクションを選択する場合は、関連するネットワークポリシーの名前を [ネットワークポリシー名] フィールドに入力する必要があります。

パイプラインの作成中、OpenSearch Ingestion は指定されたネットワークポリシーの存在をチェックします。存在しない場合は、OpenSearch Ingestion が作成します。存在する場合、OpenSearch Ingestion は新しいルールを追加して更新します。このルールは、パイプラインとコレクションを接続する VPC エンドポイントへのアクセスを付与します。

例えば、次のようになります。

```
{
   "Rules":[
      {
         "Resource":[
            "collection/my-collection"
         ],
         "ResourceType":"collection"
      }
   ],
   "SourceVPCEs":[
      "vpce-0c510712627e27269" # The ID of the VPC endpoint that OpenSearch Ingestion
 creates between the pipeline and collection
   ],
   "Description":"Created by Data Prepper"
}
```

コンソールでは、OpenSearch Ingestion がネットワークポリシーに追加するルールの名前は、[Created by Data Prepper] です。



> **Note**
>
> 一般に、コレクションのパブリックアクセスを指定するルールは、プライベートアクセスを指定するルールよりも優先されます。したがって、ポリシーにパブリックアクセスがすでに設定されている場合、OpenSearch Ingestion が追加するこの新しいルールは、実際にポリシーの動作を変更しません。詳細については、「the section called “ポリシーの優先順位”」を参照してください。

パイプラインを停止または削除すると、OpenSearch Ingestion はパイプラインとコレクション間の VPC エンドポイントを削除します。また、ネットワークポリシーを変更して、許可されたエンドポイントのリストから VPC エンドポイントを削除します。パイプラインを再起動すると、VPC エンドポイントが再作成され、ネットワークポリシーがエンドポイント ID で再更新されます。

Amazon OpenSearch Ingestion の開始方法

Amazon OpenSearch Ingestion は、マネージド型の OpenSearch Service ドメインと OpenSearch Serverless コレクションへのデータの取り込みに対応しています。次のチュートリアルでは、パイプラインを起動して実行するための基本的な手順を説明します。

最初のチュートリアルでは、Amazon OpenSearch Ingestion を使用してシンプルなパイプラインを設定し、Amazon OpenSearch Service ドメインにデータを取り込む方法を示します。

2 番目のチュートリアルでは、Amazon OpenSearch Ingestion を使用してシンプルなパイプラインを設定し、Amazon OpenSearch Serverless コレクションにデータを取り込む方法を示します。

> **Note**
>
> 適切なアクセス許可を設定しなければ、パイプラインの作成は失敗します。パイプラインを作成する前に、必要なロールへの理解を深めるために「the section called “ロールとユーザーの設定”」を参照してください。

トピック

- チュートリアル: Amazon OpenSearch Ingestion を使用してドメインにデータを取り込む
- チュートリアル: Amazon OpenSearch Ingestion を使用してデータをコレクションに取り込む

チュートリアル: Amazon OpenSearch Ingestion を使用してドメインにデータを取り込む

このチュートリアルでは、Amazon OpenSearch Ingestion を使用してシンプルなパイプラインを設定し、Amazon OpenSearch Service ドメインにデータを取り込む方法を示します。パイプラインは、OpenSearch Ingestion がプロビジョニングおよび管理するリソースです。パイプラインを使用すると、OpenSearch Service におけるダウンストリームの分析とビジュアライゼーションのために、データのフィルタリング、強化、変換、正規化、集計を行うことができます。

このチュートリアルでは、パイプラインをすばやく起動するための基本的な手順を説明します。より包括的な手順については、「the section called “パイプラインの作成”」を参照してください。

このチュートリアルでは、次の手順を実行します。

1. ドメインを作成する
2. パイプラインを作成する
3. サンプルデータを取り込む

このチュートリアルでは、次のリソースを作成します。

- パイプラインの書き込み先となる、`ingestion-domain` という名前のドメイン
- `ingestion-pipeline` という名前のパイプライン

必要なアクセス許可

このチュートリアルを完了するには、ユーザーまたはロールに、次の最低限の許可を含む ID ベースのポリシーが、アタッチされている必要があります。これらの許可により、パイプラインロールの作成とポリシーのアタッチ (`iam:Create*` および `iam:Attach*`)、ドメインの作成または変更 (`es:*`)、パイプラインの使用 (`osis:*`) が可能になります。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Resource":"*",
         "Action":[
            "osis:*",
            "iam:Create*",
            "iam:Attach*",
            "es:*"
         ]
      },
      {
         "Resource":[
            "arn:aws:iam::111122223333:role/OpenSearchIngestion-PipelineRole"
         ],
         "Effect":"Allow",
         "Action":[
            "iam:CreateRole",
            "iam:AttachRolePolicy",
            "iam:PassRole"
         ]
      }
   ]
}
```

ステップ 1: パイプラインロールを作成する

まず、OpenSearch Service ドメインのシンクにアクセスするために、パイプラインが引き受けるロールを作成します。これは、このチュートリアルの後半でパイプライン設定に含めます。

パイプラインロールを作成するには

1. https://console.aws.amazon.com/iamv2/ で AWS Identity and Access Management コンソールを開きます。
2. [ポリシー] を選択してから、[ポリシーを作成] を選択します。
3. このチュートリアルでは、次のステップで作成する `ingestion-domain` というドメインにデータを取り込みます。[JSON] を選択し、次のポリシーをエディタに貼り付けます。*`your-account-id`* を自分のアカウント ID に置き換え、必要に応じてリージョンを変更します。

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "es:DescribeDomain",
               "Resource": "arn:aws:es:us-east-1:111122223333:domain/ingestion-
   domain"
           },
           {
               "Effect": "Allow",
               "Action": "es:ESHttp*",
               "Resource": "arn:aws:es:us-east-1:111122223333:domain/ingestion-
   domain/*"
           }
       ]
   }
   ```

   既存のドメインにデータを書き込む場合は、`ingestion-domain` を自分のドメイン名に置き換えます。

   > **Note**
   >
   > このチュートリアルでは、わかりやすいように広範なアクセスポリシーを使用します。しかし、本番稼働用環境では、パイプラインロールに制限の厳しいアクセスポリシーを適用することをお勧めします。必要最低限のアクセス許可を付与するポリシーの例につ

いては、「the section called “パイプラインにドメインへのアクセスを付与する”」を参照してください。

4. [次へ]、[次へ] の順にクリックし、ポリシーに pipeline-policy という名前を付けます。
5. [Create policy] (ポリシーの作成) を選択します。
6. 次に、ポリシーを作成してロールにアタッチします。[ロール]、[ロールの作成] の順に選択します。
7. [カスタム信頼ポリシー] をクリックし、次のポリシーをエディタに貼り付けます。

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"osis-pipelines.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

8. [次へ] を選択します。その後、pipeline-policy (先ほど作成したもの) を検索して選択します。
9. [次へ] をクリックし、ロールに [PipelineRole] という名前を付けます。
10. [ロールの作成] を選択してください。

ロールの Amazon リソースネーム (ARN) (例: `arn:aws:iam::your-account-id:role/PipelineRole`) を覚えておいてください。これは、パイプラインを作成するときに必要です。

ステップ 2: ドメインを作成する

まず、データを取り込む、`ingestion-domain` という名前のドメインを作成します。

Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/home) に移動し、次の要件を満たすドメインを作成します。

- OpenSearch 1.0 以降、または Elasticsearch 7.4 以降を実行している

- パブリックアクセスを使用する
- きめ細かいアクセスコントロールを使用しない

> **Note**
>
> これらの要件は、このチュートリアルをわかりやすくするためのものです。本番環境では、VPC アクセスを使用してドメインを設定できるほか、きめ細かなアクセスコントロールを使用することも可能です。きめ細かなアクセスコントロールを使用するには、「v」を参照してください。

ドメインには、OpenSearch Service が次のステップで作成する `OpenSearchIngestion-PipelineRole` IAM ロールにアクセス許可を付与するアクセスポリシーが必要です。パイプラインは、ドメインシンクにデータを送信するためにこの役割を引き受けます。

ドメインが次のドメインレベルのアクセスポリシーを持っていることを確認します。このポリシーは、パイプラインロールにドメインへのアクセスを付与します。リージョンとアカウント ID を自分のものに置き換えます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/OpenSearchIngestion-PipelineRole"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/ingestion-domain/*"
    }
  ]
}
```

ドメインレベルでのアクセスポリシーの作成方法については、「the section called “リソースベースのポリシー”」を参照してください。

ドメインを既に作成済みの場合、その既存のアクセスポリシーを変更し、OpenSearchIngestion-PipelineRole に上記のアクセス許可を付与します。

ステップ 3: パイプラインを作成する

これでドメインが作成され、パイプラインを作成できます。

パイプラインを作成するには

1. Amazon OpenSearch Service コンソールで、左側のナビゲーションペインから [パイプライン] をクリックします。
2. パイプラインの作成 を選択します。
3. [空] のパイプラインを選択し、[ブループリントを選択] を選択します。
4. このチュートリアルでは、HTTP source プラグインを使用するシンプルなパイプラインを作成します。プラグインは、JSON 配列形式のログデータを受け入れます。1 つの OpenSearch Service ドメインをシンクとして指定し、すべてのデータを application_logs インデックスに取り込みます。

   [ソース] メニューで、[HTTP] を選択します。[パス] には、[/logs] と入力します。
5. このチュートリアルでは、わかりやすいようにパイプラインにパブリックアクセスを設定します。[ソースネットワークオプション] で、[パブリックアクセス] を選択します。VPC アクセスの設定の詳細については、「the section called “パイプラインの VPC アクセスを設定する”」を参照してください。
6. [次へ] を選択します。
7. [プロセッサ] に [日付] と入力し、[追加] を選択します。
8. [受信からの時間] を有効にします。その他すべての設定はデフォルトのままにします。
9. [次へ] を選択します。
10. シンクの詳細を設定します。[OpenSearch リソースタイプ] で、[マネージドクラスター] を選択します。次に、前のセクションで作成した [OpenSearch Service ドメイン] を選択します。

    [インデックス名] に [application_logs] と入力します。OpenSearch Ingestion は、このインデックスが存在しない場合、ドメインに自動的にインデックスを作成します。
11. [次へ] を選択します。
12. パイプラインに ingestion-pipeline という名前を付けます。容量設定はデフォルトのままにします。

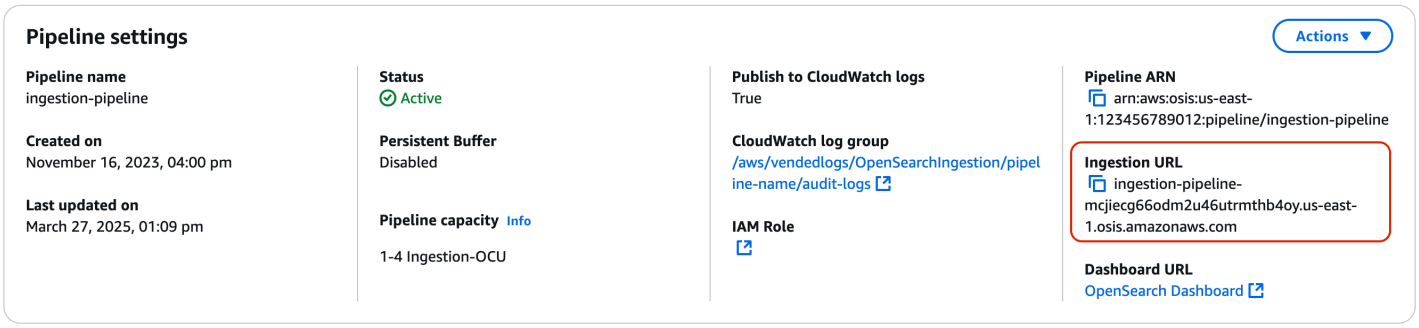
13. [パイプラインロール] で、[新しいサービスロールを作成して使用する] を選択します。パイプラインロールは、パイプラインからドメインシンクに書き込んだり、プル型のソースから読み込んだりするために必要なアクセス許可を付与します。このオプションを選択すると、OpenSearch Ingestion が IAM で手動でロールを作成するのではなく、自動的にロールを作成できるようになります。詳細については、「the section called “ロールとユーザーの設定”」を参照してください。
14. [サービスロール名のサフィックス] に [PipelineRole] と入力します。IAM では、ロールの形式は `arn:aws:iam::your-account-id:role/OpenSearchIngestion-PipelineRole` になります。
15. [次へ] を選択します。パイプライン設定を確認し、[パイプラインの作成] をクリックします。パイプラインがアクティブになるまでに 5 ~ 10 分かかります。

ステップ 4: サンプルデータを取り込む

パイプラインのステータスが `Active` になると、パイプラインへのデータの取り込みを開始できます。Signature Version 4 を使用して、パイプラインへのすべての HTTP リクエストに署名する必要があります。Postman や awscurl などの HTTP ツールを使用して、パイプラインにデータを送信します。データのインデックス作成をドメインに直接行う場合と同様に、パイプラインにデータを取り込むには、常に IAM ロールまたは IAM アクセスキーとシークレットキーのいずれかが必要です。

> **Note**
>
> リクエストに署名するプリンシパルは、`osis:Ingest` という IAM アクセス許可を持っている必要があります。

まず、[パイプライン設定] ページから取り込み URL を取得します。



次に、いくつかのサンプルデータを取り込みます。次のリクエストでは、awscurl を使用して 1 つのログファイルをパイプラインに送信します。

```
awscurl --service osis --region us-east-1 \
    -X POST \
    -H "Content-Type: application/json" \
    -d
 '[{"time":"2014-08-11T11:40:13+00:00","remote_addr":"122.226.223.69","status":"404","request":
 http://www.k2proxy.com//hello.html HTTP/1.1","http_user_agent":"Mozilla/4.0
 (compatible; WOW64; SLCC2;)"}]' \
    https://pipeline-endpoint.us-east-1.osis.amazonaws.com/logs
```

200 OK というレスポンスが確認できるはずです。認証エラーが発生した場合は、パイプラインとは別のアカウントからデータを取り込んでいることが原因かもしれません。「the section called “アクセス許可に関する問題に対応する”」を参照してください。

次に、application_logs インデックスにクエリを実行して、ログエントリが正常に取り込まれたことを確認します。

```
awscurl --service es --region us-east-1 \
     -X GET \
     https://search-ingestion-domain.us-east-1.es.amazonaws.com/application_logs/
_search | json_pp
```

レスポンス例:

```
{
   "took":984,
   "timed_out":false,
   "_shards":{
      "total":1,
      "successful":5,
      "skipped":0,
      "failed":0
   },
   "hits":{
      "total":{
         "value":1,
         "relation":"eq"
      },
      "max_score":1.0,
```

```
      "hits":[
         {
            "_index":"application_logs",
            "_type":"_doc",
            "_id":"z6VY_IMBRpceX-DU6V4O",
            "_score":1.0,
            "_source":{
               "time":"2014-08-11T11:40:13+00:00",
               "remote_addr":"122.226.223.69",
               "status":"404",
               "request":"GET http://www.k2proxy.com//hello.html HTTP/1.1",
               "http_user_agent":"Mozilla/4.0 (compatible; WOW64; SLCC2;)",
               "@timestamp":"2022-10-21T21:00:25.502Z"
            }
         }
      ]
   }
}
```


アクセス許可に関する問題に対応する

チュートリアルのステップに従っても、データを取り込もうとしたときに認証エラーが表示される場合は、パイプラインに書き込むロールがパイプライン自体 AWS アカウント とは異なる にある可能性があります。この場合は、明確にデータの取り込みを可能にするロールを作成して引き受ける必要があります。手順については、「the section called “クロスアカウント取り込みアクセスの提供”」を参照してください。

関連リソース

このチュートリアルでは、HTTP 経由で 1 つのドキュメントを取り込むというシンプルなユースケースを紹介しました。本番シナリオでは、1 つ以上のパイプラインにデータを送信するように、クライアントアプリケーション (Fluent Bit、Kubernetes、OpenTelemetry Collector など) を設定します。パイプラインは、このチュートリアルのシンプルな例よりも複雑になる可能性があります。

クライアントの設定とデータの取り込みを開始するには、次のリソースを参照してください。

- パイプラインの作成と管理
- OpenSearch Ingestion にデータを送信するようにクライアントを設定する
- Data Prepper のドキュメント

チュートリアル: Amazon OpenSearch Ingestion を使用してデータをコレクションに取り込む

このチュートリアルでは、Amazon OpenSearch Ingestion を使用してシンプルなパイプラインを設定し、Amazon OpenSearch Serverless コレクションにデータを取り込む方法を示します。パイプラインは、OpenSearch Ingestion がプロビジョニングおよび管理するリソースです。パイプラインを使用すると、OpenSearch Service におけるダウンストリームの分析とビジュアライゼーションのために、データのフィルタリング、強化、変換、正規化、集計を行うことができます。

プロビジョニングされた OpenSearch Service ドメインにデータを取り込む方法を示すチュートリアルについては、「the section called “チュートリアル: ドメインにデータを取り込む”」を参照してください。

このチュートリアルでは、次の手順を実行します。

1. コレクションを作成します。
2. パイプラインを作成する
3. サンプルデータを取り込む

このチュートリアルでは、次のリソースを作成します。

- パイプラインの書き込み先となる、`ingestion-collection` という名前のコレクション
- `ingestion-pipeline-serverless` という名前のパイプライン

必要なアクセス許可

このチュートリアルを完了するには、ユーザーまたはロールに、次の最低限の許可を含む ID ベースのポリシーが、アタッチされている必要があります。これらの許可により、パイプラインロールの作成とポリシーのアタッチ (`iam:Create*` と `iam:Attach*`)、コレクションの作成または変更 (`aoss:*`)、パイプラインの使用 (`osis:*`) が可能になります。

さらに、パイプラインロールを自動的に作成し、OpenSearch Ingestion に渡してコレクションにデータを書き込むには、複数の IAM アクセス許可が必要です。

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Resource":"*",
         "Action":[
            "osis:*",
            "iam:Create*",
            "iam:Attach*",
            "aoss:*"
         ]
      },
      {
         "Resource":[
            "arn:aws:iam::111122223333:role/OpenSearchIngestion-PipelineRole"
         ],
         "Effect":"Allow",
         "Action":[
            "iam:CreateRole",
            "iam:AttachRolePolicy",
            "iam:PassRole"
         ]
      }
   ]
}
```

ステップ 1: コレクションを作成する

まず、データを取り込むコレクションを作成します。コレクションに `ingestion-collection` という名前を付けます。

1. https://console.aws.amazon.com/aos/home の Amazon OpenSearch Service コンソールに移動します。
2. 左側のナビゲーションペインで [コレクション] をクリックし、次に [コレクションを作成] をクリックします。
3. Serverless generation フィールドで、Classic に切り替えを選択します。
4. コレクションに [ingestion-collection] という名前を付けます。
5. [セキュリティ]で、[標準作成] を選択します。
6. [ネットワークアクセスの設定] で、アクセスタイプを [パブリック] に変更します。

7. 他のすべての設定をデフォルトのままにして、[Next] (次へ) を選択します。
8. 次に、コレクション用のデータアクセスポリシーを設定します。[アクセスポリシー設定を自動的に一致させる] の選択を解除します。
9. [定義方法] で [JSON] をクリックし、エディタに以下のポリシーを貼り付けます。このポリシーは次の 2 つを実行します。

   - パイプラインロールがコレクションに書き込むことを許可します。
   - コレクションから読み取ることを許可します。そして、いくつかのサンプルデータをパイプラインに取り込んだ後、コレクションにクエリを実行し、データが正常に取り込まれてインデックスに書き込まれたことを確認します。

```
[
  {
    "Rules": [
      {
        "Resource": [
          "index/ingestion-collection/*"
        ],
        "Permission": [
          "aoss:CreateIndex",
          "aoss:UpdateIndex",
          "aoss:DescribeIndex",
          "aoss:ReadDocument",
          "aoss:WriteDocument"
        ],
        "ResourceType": "index"
      }
    ],
    "Principal": [
      "arn:aws:iam::your-account-id:role/OpenSearchIngestion-PipelineRole",
      "arn:aws:iam::your-account-id:role/Admin"
    ],
    "Description": "Rule 1"
  }
]
```

10. AWS アカウント ID を含めるようにPrincipal要素を変更します。2 つ目のプリンシパルでは、後でコレクションをクエリするために使用できるユーザーまたはロールを指定します。
11. [次へ] を選択します。アクセスポリシーに pipeline-collection-access という名前を付け、再度 [次へ] を選択します。

12. コレクション設定を確認してから、[Submit] (送信) を選択します。

ステップ 2: パイプラインを作成する

これでコレクションが作成され、パイプラインを作成できます。

パイプラインを作成するには

1. Amazon OpenSearch Service コンソールで、左側のナビゲーションペインから [パイプライン] をクリックします。
2. パイプラインの作成 を選択します。
3. [空] のパイプラインを選択し、[ブループリントを選択] を選択します。
4. このチュートリアルでは、HTTP source プラグインを使用するシンプルなパイプラインを作成します。プラグインは、JSON 配列形式のログデータを受け入れます。1 つの OpenSearch Serverless コレクションをシンクとして指定し、すべてのデータを `my_logs` インデックスに取り込みます。

   [ソース] メニューで、[HTTP] を選択します。[パス] には、[/logs] と入力します。
5. このチュートリアルでは、わかりやすいようにパイプラインにパブリックアクセスを設定します。[ソースネットワークオプション] で、[パブリックアクセス] を選択します。VPC アクセスの設定の詳細については、「the section called “パイプラインの VPC アクセスを設定する”」を参照してください。
6. [次へ] を選択します。
7. [プロセッサ] に [日付] と入力し、[追加] を選択します。
8. [受信からの時間] を有効にします。その他すべての設定はデフォルトのままにします。
9. [次へ] を選択します。
10. シンクの詳細を設定します。[OpenSearch リソースタイプ] で、[コレクション (サーバーレス)] を選択します。次に、前のセクションで作成した [OpenSearch Service コレクション] を選択します。

    ネットワークポリシー名はデフォルトのままにします。[インデックス名] に [my_logs] と入力します。OpenSearch Ingestion は、このインデックスが存在しない場合、コレクションに自動的にインデックスを作成します。
11. [次へ] を選択します。
12. パイプラインに ingestion-pipeline-serverless という名前を付けます。容量設定はデフォルトのままにします。

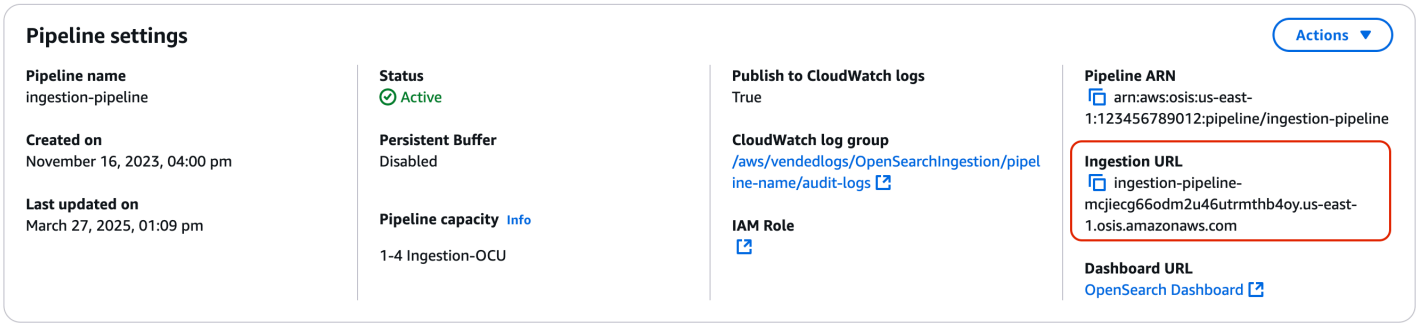
13. [パイプラインロール] で、[新しいサービスロールを作成して使用する] を選択します。パイプラインロールは、コレクションシンクに書き込んだり、プル型のソースから読み込んだりするために必要なアクセス許可をパイプラインに付与します。このオプションを選択すると、OpenSearch Ingestion が IAM で手動でロールを作成するのではなく、自動的にロールを作成できるようになります。詳細については、「the section called “ロールとユーザーの設定”」を参照してください。
14. [サービスロール名のサフィックス] に [PipelineRole] と入力します。IAM では、ロールの形式は `arn:aws:iam::`*`your-account-id`*`:role/OpenSearchIngestion-`**`PipelineRole`** になります。
15. [次へ] を選択します。パイプライン設定を確認し、[パイプラインの作成] をクリックします。パイプラインがアクティブになるまでに 5～10 分かかります。

ステップ 3: サンプルデータを取り込む

パイプラインのステータスが `Active` になると、パイプラインへのデータの取り込みを開始できます。Signature Version 4 を使用して、パイプラインへのすべての HTTP リクエストに署名する必要があります。Postman や awscurl などの HTTP ツールを使用して、パイプラインにデータを送信します。データのインデックス作成をコレクションに直接行う場合と同様に、パイプラインにデータを取り込むには、常に IAM ロールまたは IAM アクセスキーとシークレットキーのいずれかが必要です。

> **Note**
>
> リクエストに署名するプリンシパルは、`osis:Ingest` という IAM アクセス許可を持っている必要があります。

まず、[パイプライン設定] ページから取り込み URL を取得します。



次に、いくつかのサンプルデータを取り込みパスに送信します。次のサンプルリクエストでは、awscurl を使用して 1 つのログファイルをパイプラインに送信します。

```
awscurl --service osis --region us-east-1 \
    -X POST \
    -H "Content-Type: application/json" \
    -d
 '[{"time":"2014-08-11T11:40:13+00:00","remote_addr":"122.226.223.69","status":"404","request":
 http://www.k2proxy.com//hello.html HTTP/1.1","http_user_agent":"Mozilla/4.0
 (compatible; WOW64; SLCC2;)"}]' \
    https://pipeline-endpoint.us-east-1.osis.amazonaws.com/logs
```

200 OK というレスポンスが確認できるはずです。

次に、my_logs インデックスにクエリを実行して、ログエントリが正常に取り込まれたことを確認します。

```
awscurl --service aoss --region us-east-1 \
     -X GET \
     https://collection-id.us-east-1.aoss.amazonaws.com/my_logs/_search | json_pp
```

レスポンス例:

```
{
   "took":348,
   "timed_out":false,
   "_shards":{
      "total":0,
      "successful":0,
      "skipped":0,
      "failed":0
   },
   "hits":{
      "total":{
         "value":1,
         "relation":"eq"
      },
      "max_score":1.0,
      "hits":[
         {
            "_index":"my_logs",
            "_id":"1%3A0%3ARJgDvIcBTy5m12xrKE-y",
```

```
            "_score":1.0,
            "_source":{
               "time":"2014-08-11T11:40:13+00:00",
               "remote_addr":"122.226.223.69",
               "status":"404",
               "request":"GET http://www.k2proxy.com//hello.html HTTP/1.1",
               "http_user_agent":"Mozilla/4.0 (compatible; WOW64; SLCC2;)",
               "@timestamp":"2023-04-26T05:22:16.204Z"
            }
         }
      ]
   }
}
```

関連リソース

このチュートリアルでは、HTTP 経由で 1 つのドキュメントを取り込むというシンプルなユースケースを紹介しました。本番シナリオでは、1 つ以上のパイプラインにデータを送信するように、クライアントアプリケーション (Fluent Bit、Kubernetes、OpenTelemetry Collector など) を設定します。パイプラインは、このチュートリアルのシンプルな例よりも複雑になる可能性があります。

クライアントの設定とデータの取り込みを開始するには、次のリソースを参照してください。

- パイプラインの作成と管理
- OpenSearch Ingestion にデータを送信するようにクライアントを設定する
- Data Prepper のドキュメント

Amazon OpenSearch Ingestion のパイプライン機能の概要

Amazon OpenSearch Ingestion は、ソース、バッファ、0 個以上のプロセッサ、1 個以上のシンクで構成されるパイプラインをプロビジョニングします。取り込みパイプラインは、データエンジンとして Data Prepper を利用しています。パイプラインのさまざまなコンポーネントの概要については、「the section called “主要なコンセプト”」を参照してください。

次の各セクションでは、Amazon OpenSearch Ingestion で最も一般的に使用されるいくつかの機能の概要を説明します。

> **Note**
>
> これは、パイプラインで利用可能な機能をすべて網羅したリストではありません。使用可能なすべてのパイプライン機能に関する包括的なドキュメントについては、「Data Prepper のドキュメント」を参照してください。OpenSearch Ingestion では、使用できるプラグインやオプションにいくつかの制約があることに注意してください。詳細については、「the section called “サポートされているプラグインとオプション”」を参照してください。

トピック

- 永続的バッファリング
- 分割
- チェーン
- デッドレターキュー
- インデックス管理
- エンドツーエンドの確認応答
- ソースバックプレッシャー

永続的バッファリング

永続バッファは、データを複数のアベイラビリティーゾーンにわたるディスクベースのバッファに保存して、データの耐久性を高めます。永続的バッファリングを使用して、スタンドアロンバッファを設定することなく、サポートされているすべてのプッシュベースのソースからデータを取り込むことができます。これらのソースには、ログ、トレース、メトリクスの HTTP と OpenTelemetry が含まれます。永続的バッファリングを有効にするには、パイプラインを作成または更新するときに [永続的バッファリングを有効にする] を選択します。詳細については、「the section called “パイプラインの作成”」を参照してください。

OpenSearch Ingestion は、データソース、ストリーミング変換、およびシンクの送信先を考慮して、永続的なバッファリングに使用する OCU の数を動的に決定します。一部の OCU がバッファリングに割り当てられるため、同じ取り込みスループットを維持するには、最小 OCU と最大 OCUs を増やす必要がある場合があります。パイプラインは、データをバッファに最大 72 時間保持します。

パイプラインの永続的バッファリングを有効にする場合、デフォルトの最大リクエストペイロードサイズは次のとおりです。

- HTTP ソース – 10 MB
- OpenTelemetry ソース – 4 MB

HTTP ソースの場合、ペイロードの最大サイズを 20 MB に増やすことができます。リクエストペイロードサイズには、通常複数のイベントを含む HTTP リクエスト全体が含まれます。各イベントは 3.5 MB を超えることはできません。

永続的バッファが有効になっているパイプラインは、プロセッサタイプに関係なく、1:1 buffer-to-computeの比率を使用して、設定された OCUs (パイプラインユニット) をコンピューティングとバッファの間で均等に分割します。

デフォルトでは、パイプラインは を使用してバッファデータを暗号化 AWS 所有のキー します。これらのパイプラインでは、パイプラインロールに追加の許可は必要ありません。

あるいは、カスタマーマネージドキーを指定して、次の IAM 許可をパイプラインロールに追加することもできます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "KeyAccess",
            "Effect": "Allow",
            "Action": [
              "kms:Decrypt",
              "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/ASIAIOSFODNN7EXAMPLE"
        }
    ]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

> **Note**
>
> 永続的バッファリングを無効にすると、パイプラインは完全にメモリ内バッファリングで動作を開始します。

分割

OpenSearch Ingestion パイプラインを設定し、受信イベントをサブパイプラインに分割すると、同じ受信イベントに対してさまざまなタイプの処理を実行できます。

次のパイプラインの例では、受信イベントを 2 つのサブパイプラインに分割します。各サブパイプラインは、独自のプロセッサを使用してデータをエンリッチ化および操作し、データを異なる OpenSearch インデックスに送信します。

```
version: "2"
log-pipeline:
  source:
    http:
    ...
  sink:
    - pipeline:
        name: "logs_enriched_one_pipeline"
    - pipeline:
        name: "logs_enriched_two_pipeline"

logs_enriched_one_pipeline:
  source:
    pipeline:
      name: "log-pipeline"
  processor:
   ...
  sink:
    - opensearch:
        # Provide a domain or collection endpoint
        # Enable the 'serverless' flag if the sink is an OpenSearch Serverless
 collection
        aws:
          ...
        index: "enriched_one_logs"

logs_enriched_two_pipeline:
```

```
  source:
    pipeline:
      name: "log-pipeline"
  processor:
   ...
  sink:
    - opensearch:
        # Provide a domain or collection endpoint
        # Enable the 'serverless' flag if the sink is an OpenSearch Serverless
 collection
        aws:
          ...
          index: "enriched_two_logs"
```

チェーン

複数のサブパイプラインを連結し、データ処理とエンリッチメントをチャンク単位で実行できます。つまり、1 つのサブパイプライン内で特定の処理機能を用いて受信イベントをエンリッチ化し、それを別のサブパイプラインに送信して別のプロセッサでさらにエンリッチ化して、最後に OpenSearch シンクに送信できます。

次の例では、log_pipeline サブパイプラインは受信ログイベントをプロセッサのセットでエンリッチ化し、そのイベントを enriched_logs という名前の OpenSearch インデックスに送信します。パイプラインは同じイベントを log_advanced_pipeline サブパイプラインに送信し、サブパイプラインはそれを処理して enriched_advanced_logs という名前の別の OpenSearch インデックスに送信します。

```
version: "2"
log-pipeline:
  source:
    http:
    ...
  processor:
    ...
  sink:
    - opensearch:
        # Provide a domain or collection endpoint
        # Enable the 'serverless' flag if the sink is an OpenSearch Serverless
 collection
        aws:
          ...
          index: "enriched_logs"
```

```
    - pipeline:
        name: "log_advanced_pipeline"

log_advanced_pipeline:
  source:
    pipeline:
      name: "log-pipeline"
  processor:
   ...
  sink:
    - opensearch:
        # Provide a domain or collection endpoint
        # Enable the 'serverless' flag if the sink is an OpenSearch Serverless
 collection
        aws:
          ...
          index: "enriched_advanced_logs"
```

デッドレターキュー

デッドレターキュー (DLQ) とは、パイプラインがシンクへの書き込みに失敗したイベントの送信先です。OpenSearch Ingestion では、DLQ として使用するために、適切な書き込み許可を持つ Amazon S3 バケットを指定する必要があります。パイプライン内のすべてのシンクに DLQ 設定を追加できます。パイプラインで書き込みエラーが発生すると、設定された S3 バケットに DLQ オブジェクトが作成されます。DLQ オブジェクトは、失敗したイベントの配列として JSON ファイル内に存在します。

次のいずれかの条件が満たされたとき、パイプラインは DLQ にイベントを書き込みます。

- OpenSearch シンクの[最大再試行] 回数が使い果たされました。OpenSearch Ingestion では、この設定での必要最低数は 16 個です。
- エラー状態のため、イベントがシンクによって拒否されています。

設定

サブパイプラインのデッドレターキューを設定するには、シンクの送信先を設定するときに [S3 DLQ を有効にする] を選択します。次に、キューに必要な設定を指定します。詳細については、Data Prepper DLQ ドキュメントの「Configuration」を参照してください。

この S3 DLQ に書き込まれたファイルには、次の命名パターンが付けられます。

```
dlq-v${version}-${pipelineName}-${pluginId}-${timestampIso8601}-${uniqueId}
```

DLQ が書き込む S3 バケットへのアクセスを許可するようにパイプラインロールを手動で設定する手順については、「the section called “Amazon S3 またはデッドレターキューに書き込むためのアクセス許可”」を参照してください。

例

次の DLQ ファイルの例を考えてみます。

```
dlq-v2-apache-log-pipeline-opensearch-2023-04-05T15:26:19.152938Z-e7eb675a-
f558-4048-8566-dac15a4f8343
```

次は、シンクへの書き込みに失敗したデータの例です。これは、さらなる分析のために DLQ S3 バケットに送信されます。

```
Record_0
pluginId            "opensearch"
pluginName          "opensearch"
pipelineName        "apache-log-pipeline"
failedData
index    "logs"
indexId   null
status    0
message  "Number of retries reached the limit of max retries (configured value 15)"
document
log      "sample log"
timestamp     "2023-04-14T10:36:01.070Z"

Record_1
pluginId            "opensearch"
pluginName          "opensearch"
pipelineName        "apache-log-pipeline"
failedData
index               "logs"
indexId   null
status    0
message  "Number of retries reached the limit of max retries (configured value 15)"
document
log                 "another sample log"
timestamp           "2023-04-14T10:36:01.071Z"
```

インデックス管理

Amazon OpenSearch Ingestion には、次の機能など多くのインデックス管理機能があります。

インデックスの作成

パイプラインのシンクでインデックス名を指定すると、OpenSearch Ingestion でパイプラインをプロビジョニングするときに、そのインデックスが作成されます。インデックスが既に存在する場合、パイプラインはそれを使用して受信イベントのインデックスを作成します。インデックスがまだ存在しない場合、パイプラインを停止して再起動する、または YAML 設定を更新すると、パイプラインは新しいインデックスの作成を試みます。パイプラインではインデックスを一切削除できません。

次のシンクのサンプルでは、パイプラインがプロビジョニングされるときに 2 つのインデックスを作成します。

```
sink:
  - opensearch:
      index: apache_logs
  - opensearch:
      index: nginx_logs
```

インデックス名とパターンの生成

受信イベントのフィールドにある変数を使用すると、動的なインデックス名を生成できます。シンク設定では、形式 `string${}` を使用して文字列補間を示し、JSON ポインタを使用してイベントからフィールドを抽出します。`index_type` のオプションは `custom` または `management_disabled` です。`index_type` は OpenSearch ドメインでは `custom` に、OpenSearch Serverless コレクションでは `management_disabled` にデフォルトで設定されるため、未設定のままにすることができます。

例えば、次のパイプラインは、受信イベントから `metadataType` フィールドを選択してインデックス名を生成します。

```
pipeline:
  ...
  sink:
    opensearch:
      index: "metadata-${metadataType}"
```

次の設定では、1 日または 1 時間ごとに新しいインデックスを生成し続けます。

```
pipeline:
  ...
  sink:
    opensearch:
      index: "metadata-${metadataType}-%{yyyy.MM.dd}"

pipeline:
  ...
  sink:
    opensearch:
      index: "metadata-${metadataType}-%{yyyy.MM.dd.HH}"
```

インデックス名は、my-index-%{yyyy.MM.dd} のように、サフィックスとして日付/時刻パターンを持つプレーン文字列にすることもできます。シンクでは、データを OpenSearch に送信するときに日時パターンを UTC 時間に置き換え、my-index-2022.01.25 のような新しいインデックスを 1 日ごとに作成します。詳細については、「DateTimeFormatter」クラスを参照してください。

このインデックス名は、my-${index}-name のように形式化された文字列にすることもできます (日付/時刻パターンのサフィックスの有無にかかわらず)。シンクでは、データを OpenSearch に送信するときに、"${index}" の部分を処理中のイベント内の値に置き換えます。形式が "${index1/index2/index3}" の場合、フィールド index1/index2/index3 をイベント内の値に置き換えます。

ドキュメント ID の生成

OpenSearch にドキュメントのインデックスを作成するとき、パイプラインはドキュメント ID を生成できます。また、それらのドキュメント ID を受信イベント内のフィールドから推測することも可能です。

次の例では、受信イベントの uuid フィールドを使用してドキュメント ID を生成します。

```
pipeline:
  ...
  sink:
    opensearch:
      index_type: custom
      index: "metadata-${metadataType}-%{yyyy.MM.dd}"
      "document_id": "uuid"
```

次の例では、Add entries プロセッサを使用し、受信イベントから uuid フィールドと other_field フィールドをマージしてドキュメント ID を生成します。

create アクションは、同じ ID のドキュメントが上書きされないようにします。パイプラインは再試行や DLQ イベントを必要とせずに、重複したドキュメントを削除します。ここでの目的は、既存ドキュメントの更新を避けることなので、このアクションを使用するパイプライン作成者にとっては当然想定されるものです。

```
pipeline:
  ...
  processor:
   - add_entries:
      entries:
        - key: "my_doc_id_field"
          format: "${uuid}-${other_field}"
  sink:
    - opensearch:
       ...
       action: "create"
       document_id: "my_doc_id"
```

イベントのドキュメント ID をサブオブジェクトのフィールドに設定したい場合があります。次の例では、OpenSearch シンクプラグインで info/id というサブオブジェクトを使用して、ドキュメント ID を生成します。

```
sink:
  - opensearch:
       ...
       document_id: info/id
```

次のイベントが発生すると、パイプラインは _id フィールドに json001 を設定したドキュメントを生成します。

```
{
   "fieldA":"arbitrary value",
   "info":{
      "id":"json001",
      "fieldA":"xyz",
      "fieldB":"def"
   }
}
```

ルーティング ID の生成

OpenSearch シンクプラグイン内の routing_field オプションを使用すると、ドキュメントルーティングプロパティ (_routing) の値を受信イベントの値に設定できます。

ルーティングは JSON ポインタ構文をサポートしているため、最上位のフィールドだけでなく、ネストされたフィールドも使用できます。

```
sink:
  - opensearch:
        ...
        routing_field: metadata/id
        document_id: id
```

次のイベントが発生すると、プラグインは _routing フィールドに abcd を設定したドキュメントを生成します。

```
{
   "id":"123",
   "metadata":{
      "id":"abcd",
      "fieldA":"valueA"
   },
   "fieldB":"valueB"
}
```

インデックスを作成するときに、パイプラインで使用できるインデックステンプレートを作成する手順については、「インデックステンプレート」を参照してください。

エンドツーエンドの確認応答

OpenSearch Ingestion では、エンドツーエンドの確認応答を使用して、ステートレスパイプライン内のソースからシンクまでのデータの配信を追跡します。これにより、データの耐久性と信頼性を保証します。

> **Note**
>
> 現在、エンドツーエンドの確認応答をサポートしているのは S3 ソースプラグインのみです。

エンドツーエンドの確認応答では、パイプラインのソースプラグインが確認応答セットを作成し、イベントのバッチを監視します。イベントがシンクに正常に送信された場合は肯定応答を受け取り、いずれかのイベントがシンクに送信できなかった場合は否定応答を受け取ります。

パイプラインコンポーネントに障害またはクラッシュが発生した場合、またはソースが確認応答を受け取れなかった場合、ソースはタイムアウトし、再試行や障害のログ記録などの必要なアクションを実行します。パイプラインに複数のシンクまたは複数のサブパイプラインが設定されている場合、イベントレベルの確認応答は、イベントが全サブパイプラインの全シンクに送信された後にのみ送信されます。シンクに DLQ が設定されている場合、エンドツーエンドの確認応答は DLQ に書き込まれたイベントも追跡します。

エンドツーエンドの確認を有効にするには、Amazon S3 ソース設定で [追加オプション] を展開し [エンドツーエンドのメッセージの確認を有効にする] を選択します。

ソースバックプレッシャー

パイプラインでは、データ処理で負荷がかかっているときや、シンクが一時的にダウンしていたり、データの取り込みに時間がかかったりすると、バックプレッシャーが生じることがあります。OpenSearch Ingestion では、パイプラインで使用されているソースプラグインによってバックプレッシャーの処理方法が異なります。

HTTP ソース

HTTP ソースプラグインを使用するパイプラインでは、混雑しているパイプラインコンポーネントによってバックプレッシャーの処理方法が異なります。

- バッファ - バッファがいっぱいになると、パイプラインはエラーコード 408 の HTTP ステータス `REQUEST_TIMEOUT` をソースエンドポイントに返し始めます。バッファが解放されると、パイプラインは HTTP イベントの処理を再開します。
- ソーススレッド - すべての HTTP ソーススレッドがリクエストを実行中で負荷がかかっており、未処理のリクエストキューサイズがリクエストの最大許容数を超えた場合、パイプラインはエラーコード 429 の HTTP ステータス `TOO_MANY_REQUESTS` をソースエンドポイントに返し始めます。リクエストキューが最大許容キューサイズを下回ると、パイプラインはリクエストの処理を再開します。

OTel ソース

OpenTelemetry ソース (OTel logs、OTel metrics、および OTel trace) を使用するパイプラインのバッファがいっぱいになると、パイプラインはエラーコード 408 の HTTP ステータス

REQUEST_TIMEOUT をソースエンドポイントに返し始めます。バッファが解放されると、パイプラインはイベントの処理を再開します。

S3 ソース

S3 ソースを使用するパイプラインのバッファがいっぱいになると、パイプラインは SQS 通知の処理を停止します。バッファが解放されると、パイプラインは通知の処理を再開します。

シンクがダウンしている、またはデータを取り込むことができず、ソースのエンドツーエンドの確認応答が有効になっている場合、パイプラインはすべてのシンクから正常な確認を受け取るまで SQS 通知の処理を停止します。

Amazon OpenSearch Ingestion パイプラインの作成

パイプラインとは、Amazon OpenSearch Ingestion がデータをソース (データの取得元) からシンク (データの送信先) へ移動するために使用するメカニズムです。OpenSearch Ingestion では、シンクは常に単一の Amazon OpenSearch Service ドメインになりますが、データのソースは Amazon S3、Fluent Bit、OpenTelemetry Collector などのクライアントである可能性があります。

詳細については、OpenSearch ドキュメントの「パイプライン」を参照してください。

トピック

- 前提条件と必要な IAM ロール
- 必要な IAM 許可
- パイプラインのバージョンの指定
- 取り込みパスの指定
- パイプラインの作成
- パイプラインの作成ステータスの追跡
- ブループリントの使用

前提条件と必要な IAM ロール

OpenSearch Ingestion パイプラインを作成するには、次のリソースが必要です。

- OpenSearch Ingestion がシンクに書き込むために引き受けるパイプラインロールと呼ばれる IAM ロール。このロールは事前に作成することも、パイプラインの作成中に OpenSearch Ingestion で自動的に作成することもできます。

- シンクとして機能する OpenSearch Service ドメインまたは OpenSearch Serverless コレクション。ドメインに書き込む場合、OpenSearch 1.0 以降、または Elasticsearch 7.4 以降を実行している必要があります。シンクには、IAM パイプラインロールに適切な許可を付与するアクセスポリシーが必要です。

これらのリソースの作成手順については、次の各トピックを参照してください。

- the section called “パイプラインにドメインへのアクセスを付与する”
- the section called “パイプラインにコレクションへのアクセス権を付与する”

> **Note**
>
> きめ細かなアクセスコントロールを使用するドメインに書き込む場合は、追加の手順を実行する必要があります。「the section called “パイプラインロールをマッピングする (詳細なアクセスコントロールを使用するドメインについてのみ)”」を参照してください。

必要な IAM 許可

OpenSearch Ingestion は、パイプライン作成時に次の IAM アクセス許可を使用します。

- `osis:CreatePipeline` — パイプラインを作成します。
- `osis:ValidatePipeline` — パイプライン設定が有効かどうかを確認します。
- `iam:CreateRole` および `iam:AttachPolicy` – OpenSearch Ingestion でパイプラインロールを自動的に作成します。
- `iam:PassRole` — パイプラインロールを OpenSearch Ingestion に渡し、ドメインにデータを書き込めるようにします。このアクセス許可は、パイプラインロールリソースにある必要があります。また、パイプラインごとに異なるロールを使用する場合は、単に * を使用します。

例えば、次のポリシーは、パイプラインを作成するためのアクセス許可を付与します。

JSON

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Resource":"*",
         "Action":[
            "osis:CreatePipeline",
            "osis:ListPipelineBlueprints",
            "osis:ValidatePipeline"
         ]
      },
      {
         "Resource":[
            "arn:aws:iam::111122223333:role/pipeline-role"
         ],
         "Effect":"Allow",
         "Action":[
            "iam:CreateRole",
            "iam:AttachRolePolicy",
            "iam:PassRole"
         ]
      }
   ]
}
```

OpenSearch Ingestion には、`osis:Ingest` というパーミッションも含まれています。これは、署名バージョン 4 を使用して署名付きリクエストをパイプラインに送信するために必要です。詳細については、「the section called “取り込みロールの作成”」を参照してください。

> **Note**
>
> また、アカウントで最初にパイプラインを作成するユーザーは、`iam:CreateServiceLinkedRole` アクションの許可を有している必要があります。詳細については、「パイプラインロールリソース」を参照してください。

各アクセス許可の詳細については、「サービス認証リファレンス」の「Actions, resources, and condition keys for OpenSearch Ingestion」を参照してください。

パイプラインのバージョンの指定

設定エディターを使用してパイプラインを設定するときは、パイプラインを実行する Data Prepper のメジャーバージョンを指定する必要があります。バージョンを指定するには、パイプライン設定に `version` オプションを含めます。

```
version: "2"
log-pipeline:
  source:
    ...
```

[作成] を選択すると、OpenSearch Ingestion は指定したメジャーバージョンの利用可能な最新マイナーバージョンを確認し、そのバージョンでパイプラインをプロビジョニングします。例えば、`version: "2"` を指定し、Data Prepper の最新サポートバージョンが 2.1.1 の場合、OpenSearch Ingestion はパイプラインをバージョン 2.1.1 でプロビジョニングします。パイプラインで実行されているマイナーバージョンは公開されません。

Data Prepper の新しいメジャーバージョンが利用可能になったときに、パイプラインをアップグレードするには、パイプライン設定を編集して新しいバージョンを指定します。パイプラインを以前のバージョンにダウングレードすることはできません。

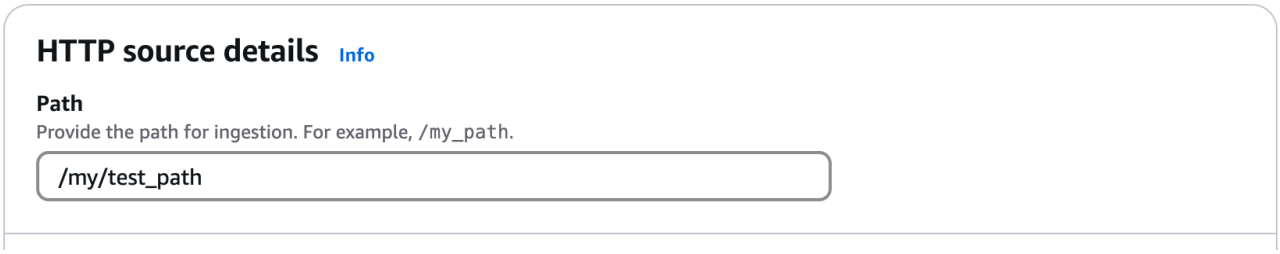
> **Note**
>
> OpenSearch Ingestion は、新しいバージョンの Data Prepper がリリースされてもすぐにはサポートしません。新しいバージョンが公開されてから OpenSearch Ingestion でサポートされるまでには、多少の遅れが生じます。さらに、OpenSearch Ingestion は特定のメジャーバージョンまたはマイナーバージョンを明示的にはサポートしていない場合があります。詳細な一覧については、「the section called “サポートされている Data Prepper のバージョン”」を参照してください。

ブルー/グリーンデプロイメントを開始する変更をパイプラインに加えるたびに、OpenSearch Ingestion は、現在パイプラインに設定されているメジャーバージョンの最新マイナーバージョンにアップグレードできます。詳細については、「the section called “パイプライン更新用のブルー/グリーンデプロイ”」を参照してください。OpenSearch Ingestion では、パイプライン設定内の `version` オプションを明示的に更新しない限り、パイプラインのメジャーバージョンを変更することはできません。

取り込みパスの指定

OTel trace や OTel metrics などのプルベースのソースの場合、OpenSearch Ingestion ではソース設定に追加の `path` オプションが必要です。パスは `/log/ingest` のような文字列で、取り込み用の URI パスを示しています。このパスが、データをパイプラインに送信する際に使用する URI を定義します。

たとえば、HTTP ソースを持つパイプラインに次のパスを指定するとします。



パイプラインにデータを取り込むときは、クライアント設定で次のエンドポイントを指定する必要があります: `https://pipeline-name-abc123.us-west-2.osis.amazonaws.com/my/test_path`

パスはスラッシュ (/) から開始し、「-」、「_」、「.、「/」などの特殊文字や `${pipelineName}` プレースホルダーを含めることができます。`${pipelineName}` を (`/${pipelineName}/test_path` のように) 使用すると、OpenSearch Ingestion は変数を関連するサブパイプラインの名前に置き換えます。

パイプラインの作成

このセクションでは、OpenSearch Service コンソールと AWS CLIを使用して、OpenSearch Ingestion パイプラインを作成する方法を説明します。

コンソール

パイプラインを作成するには、https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインし、パイプラインの作成を選択します。

空のパイプラインを選択するか、設定ブループリントを選択します。ブループリントには、さまざまな一般的なユースケース用に事前設定済みのパイプラインが含まれています。詳細については、「the section called “ブループリントの使用”」を参照してください。

[ブループリントを選択] を選択します。

ソースを設定する

1. 空のパイプラインから開始する場合は、ドロップダウンメニューからソースを選択します。使用可能なソースには、他のソース AWS のサービス、OpenTelemetry、または HTTP などがあります。詳細については、「the section called “パイプラインの統合”」を参照してください。
2. 選択したソースに応じて、ソースの追加設定を行います。例えば、Amazon S3 をソースとして使用するには、パイプラインがメッセージを受信する Amazon SQS キューの URL を指定する必要があります。サポートされているソースプラグインのリストとドキュメントへのリンクについては、「the section called “サポートされているプラグインとオプション”」を参照してください。
3. 一部のソースでは、[ソースネットワークオプション] を指定する必要があります。[VPC アクセス] または [パブリックアクセス] を選択します。[パブリックアクセス] を選択した場合は、次のステップに進みます。[VPC アクセス] を選択した場合は、次の設定を行います。

設定	説明
[エンドポイントの管理]	仮想プライベートクラウド (VPC) エンドポイントを自分で作成するか、OpenSearch Ingestion で作成するかを選択します。エンドポイント管理のデフォルトは、OpenSearch Ingestion によって管理されるエンドポイントです。
VPC	使用する VPC の ID を選択します。VPC とパイプラインは同じ AWS リージョンの中になければなりません。
サブネット	1 つ以上のサブネットを選択します。OpenSearch Service は VPC エンドポイントと Elastic Network Interface (ENI) をサブネットに配置します。
セキュリティグループ	必要なアプリケーションが、パイプラインによって公開されているポート (80 または 443) とプロトコル (HTTP または HTTP) で、OpenSearch Ingestion パイプラインに到達できるよう、1 つ以上の VPC セキュリティグループを選択します。
[VPC アタッチメントオプション]	ソースがセルフマネージドエンドポイントの場合は、パイプラインを VPC にアタッチします。提供されているデフォルトの CIDR オプションのいずれかを選択するか、カスタム CIDR を使用します。

詳細については、「the section called “パイプラインの VPC アクセスを設定する”」を参照してください。

4. [Next] (次へ) を選択します。

プロセッサを設定する

パイプラインに 1 つ以上のプロセッサを追加します。プロセッサは、ドメインまたはコレクションシンクにレコードを発行する前にイベントをフィルタリング、変換、強化できるサブパイプライン内のコンポーネントです。サポートされているプロセッサのリストとドキュメントへのリンクについては、「the section called “サポートされているプラグインとオプション”」を参照してください。

[アクション] を選択し、以下を追加できます。

- [条件付きルーティング] – 特定の条件に基づいてイベントを異なるシンクにルーティングします。詳細については、「Conditional routing」を参照してください。
- [サブパイプライン] – 各サブパイプラインは、1 個のソース、0 個以上のプロセッサ、1 個のシンクの組み合わせです。外部ソースを持つことができるサブパイプラインは 1 つだけです。他のすべては、パイプライン設定全体で他のサブパイプラインであるソースが必要です。1 個のパイプライン設定には、1 ~ 10 個のサブパイプラインを含めることができます。

[次へ] を選択します。

シンクを設定する

パイプラインがレコードを発行する送信先を選択します。すべてのサブパイプラインには、少なくとも 1 つのシンクが含まれている必要があります。パイプラインには最大 10 個のシンクを追加できます。

OpenSearch については、以下のフィールドを設定します。

設定	説明
[ネットワークポリシー名] (サーバーレスシンクのみ)	OpenSearch Serverless コレクションを選択した場合は、[ネットワークポリシー名] を入力します。OpenSearch Ingestion は、ポリシーが存在しない場合は作成するか、存在する場合はパイプラインとコレクションを接続する VPC エンドポイントへのアクセスを許可するルールでポリシーを更新しま

設定	説明
	す。詳細については、「the section called “パイプラインにコレクションへのアクセス権を付与する”」を参照してください。
インデックス名	パイプラインのデータ送信先となるインデックスの名前。OpenSearch Ingestion は、このインデックスがまだ存在しない場合に作成します。
[インデックスマッピングオプション]	パイプラインがドキュメントとそのフィールドを保存して OpenSearch シンクにインデックスを作成する方法を選択します。[動的マッピング] を選択すると、OpenSearch はドキュメントのインデックス作成時にフィールドを自動的に追加します。[マッピングをカスタマイズ] を選択した場合は、インデックスマッピングテンプレートを入力します。詳細については、「Index templates」を参照してください。
[DLQ を有効にする]	パイプラインの Amazon S3 デッドレターキュー (DLQ) を設定します。詳細については、「the section called “デッドレターキュー”」を参照してください。
追加設定	OpenSearch シンクの詳細オプションを設定します。詳細については、Data Prepper ドキュメントの「Configuration options」を参照してください。

Amazon S3 シンクを追加するには、[シンクを追加] と [Amazon S3] を選択します。詳細については、「the section called “送信先としての Amazon S3”」を参照してください。

[Next] (次へ) を選択します。

パイプラインを設定する

次の追加パイプライン設定を設定します。

設定	説明
パイプライン名	パイプラインに独自の名前を入力します。
[永続バッファ]	永続的バッファリングは、複数の Availability Zones にまたがってディスクベースのバッファにデータを保存します。詳細については、「the section called “永続的バッファリング”」を参照してください。

設定	説明
	永続的バッファリングを有効にする場合は、バッファデータを暗号化するための AWS Key Management Service キーを選択します。
[パイプライン容量]	Ingestion OpenSearch Compute Unit (OCU) でパイプラインの最小キャパシティと最大キャパシティ。詳細については、「the section called “パイプラインのスケーリング”」を参照してください。
パイプラインロール	パイプラインがシンクに書き込み、プルベースのソースから読み取るために必要なアクセス許可を提供する IAM ロール。ロールは自分で作成することも、選択したユースケースに基づいて OpenSearch Ingestion で作成することもできます。 詳細については、「the section called “ロールとユーザーの設定”」を参照してください。
タグ	パイプラインに 1 つ以上のタグを追加します。詳細については、「the section called “パイプラインのタグ付け”」を参照してください。
[ログ発行オプション]	Amazon CloudWatch Logs へのパイプラインログの発行を有効にします。パイプラインの問題をより簡単にトラブルシューティングできるように、ログの発行を有効にすることをお勧めします。詳細については、「the section called “パイプラインのログのモニタリング”」を参照してください。

[次へ] を選択し、パイプライン設定を確認し、[パイプラインを作成] をクリックします。

OpenSearch Ingestion は、非同期プロセスを実行してパイプラインを構築します。パイプラインのステータスが `Active` になると、データの取り込みを開始できます。

AWS CLI

create-pipeline コマンドは、パイプライン設定を文字列として、または .yaml もしくは、.json ファイル内で受け入れます。設定を文字列として入力する場合、改行するたびに `\n` でエスケープする必要があります。例: `"log-pipeline:\n source:\n http:\n processor:\n - grok:\n ..."`

次のサンプルコマンドは、以下にある設定でパイプラインを作成します。

- 最低 4 つの取り込み OCU、最大 10 個の取り込み OCU

- 仮想プライベートクラウド (VPC) 内でプロビジョニング
- ログ発行の有効化

```
aws osis create-pipeline \
  --pipeline-name my-pipeline \
  --min-units 4 \
  --max-units 10 \
  --log-publishing-options
 IsLoggingEnabled=true,CloudWatchLogDestination={LogGroup="MyLogGroup"} \
  --vpc-options
 SecurityGroupIds={sg-12345678,sg-9012345},SubnetIds=subnet-1212234567834asdf \
  --pipeline-configuration-body "file://pipeline-config.yaml" \
  --pipeline-role-arn  arn:aws:iam::1234456789012:role/pipeline-role
```

OpenSearch Ingestion は、非同期プロセスを実行してパイプラインを構築します。パイプラインのステータスが `Active` になると、データの取り込みを開始できます。パイプラインのステータスを確認するには、GetPipeline コマンドを使用します。

OpenSearch Ingestion API

OpenSearch Ingestion API を使用して OpenSearch Ingestion パイプラインを作成するには、CreatePipeline オペレーションを呼び出します。

パイプラインが正常に作成されたら、クライアントを設定し、OpenSearch Service ドメインへのデータの取り込みを開始できます。詳細については、「the section called “パイプラインの統合”」を参照してください。

パイプラインの作成ステータスの追跡

OpenSearch Ingestion がパイプラインをプロビジョニングしてデータを取り込む準備をしているときに、パイプラインのステータスを追跡できます。

コンソール

パイプラインを最初に作成した後、OpenSearch Ingestion がデータを取り込むための準備をするときに、パイプラインは複数の段階を経ます。パイプライン作成のさまざまな段階を確認するには、パイプライン名をクリックして[パイプラインの設定] ページを表示します。[ステータス] で、[詳細を表示] をクリックします。

パイプラインは、データの取り込みが可能になるまでに次の段階を経ます。

- 検証 — パイプライン設定を検証する。この段階が完了すると、すべての検証が成功したことになります。
- 環境の作成 — リソースの準備とプロビジョニングを行います。この段階が完了すると、新しいパイプライン環境が作成されます。
- パイプラインのデプロイ — パイプラインをデプロイします。この段階が完了すると、パイプラインは正常にデプロイされます。
- パイプラインのヘルスチェック — パイプラインのヘルス状態をチェックします。この段階が完了すると、すべてのヘルスチェックを通過したことになります。
- トラフィックの有効化 — パイプラインがデータを取り込むことができるようにします。この段階が完了すると、パイプラインへのデータの取り込みを開始できます。

CLI

get-pipeline-change-progress コマンドを使用して、パイプラインのステータスを確認します。次の AWS CLI リクエストは、 という名前のパイプラインのステータスをチェックしますmy-pipeline。

```
aws osis get-pipeline-change-progress \
    --pipeline-name my-pipeline
```

レスポンス:

```
{
   "ChangeProgressStatuses": {
      "ChangeProgressStages": [
         {
            "Description": "Validating pipeline configuration",
            "LastUpdated": 1.671055851E9,
            "Name": "VALIDATION",
            "Status": "PENDING"
         }
      ],
      "StartTime": 1.671055851E9,
      "Status": "PROCESSING",
      "TotalNumberOfStages": 5
   }
}
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用してパイプライン作成のステータスを追跡するには、GetPipelineChangeProgress オペレーションを呼び出します。

ブループリントの使用

パイプラインの定義をゼロから作成するのではなく、設定のブループリントを使用できます。設定のブループリントは、トレース分析や Apache ログなどの一般的な取り込みシナリオ用に事前設定されたテンプレートです。設定のブループリントを使用すると、設定をゼロから作成しなくても、パイプラインを簡単にプロビジョニングできます。

コンソール

パイプラインのブループリントを使用するには

1. https://console.aws.amazon.com/aos/osis/home で OpenSearch Ingestion コンソールにサインインします。Pipelines ページが表示されます。
2. パイプラインの作成 を選択します。
3. ユースケースのリストからブループリントを選択し、[ブループリントを選択] を選択します。パイプライン設定には、選択したユースケースのサブパイプラインが入力されます。

   パイプラインのブループリントは、そのままでは有効になりません。選択したソースに応じて、追加の設定を指定する必要があります。

CLI

を使用して使用可能なすべてのブループリントのリストを取得するには AWS CLI、list-pipeline-blueprints リクエストを送信します。

```
aws osis list-pipeline-blueprints
```

このリクエストは、利用可能なすべてのブループリントのリストを返します。

特定のブループリントに関する詳細な情報を取得するには、get-pipeline-blueprint コマンドを使用します。

```
aws osis get-pipeline-blueprint --blueprint-name AWS-ApacheLogPipeline
```

このリクエストは、Apache ログパイプラインブループリントの内容を返します。

```
{
   "Blueprint":{
      "PipelineConfigurationBody":"###\n  # Limitations: https://docs.aws.amazon.com/
opensearch-service/latest/ingestion/ingestion.html#ingestion-limitations\n###\n###\n
 # apache-log-pipeline:\n    # This pipeline receives logs via http (e.g. FluentBit),
 extracts important values from the logs by matching\n    # the value in the 'log' key
 against the grok common Apache log pattern. The grokked logs are then sent\n    # to
 OpenSearch to an index named 'logs'\n###\n\nversion: \"2\"\napache-log-pipeline:\n
 source:\n    http:\n      # Provide the path for ingestion. ${pipelineName} will be
 replaced with pipeline name configured for this pipeline.\n      # In this case it
 would be \"/apache-log-pipeline/logs\". This will be the FluentBit output URI value.
\n      path: \"/${pipelineName}/logs\"\n  processor:\n    - grok:\n        match:\n
         log: [ \"%{COMMONAPACHELOG_DATATYPED}\" ]\n  sink:\n    - opensearch:\n
   # Provide an AWS OpenSearch Service domain endpoint\n        # hosts: [ \"https://
search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-east-1.es.amazonaws.com\" ]\n        aws:
\n          # Provide the region of the domain.\n          # region: \"us-east-1\"\n
        # Enable the 'serverless' flag if the sink is an Amazon OpenSearch Serverless
 collection\n          # serverless: true\n        index: \"logs\"\n        # Enable
 the S3 DLQ to capture any failed requests in an S3 bucket\n        # dlq:\n
  # s3:\n            # Provide an S3 bucket\n            # bucket: \"your-dlq-bucket-
name\"\n            # Provide a key path prefix for the failed requests\n
  # key_path_prefix: \"${pipelineName}/logs/dlq\"\n            # Provide the region
 of the bucket.\n            # region: \"us-east-1\"\n            # Provide a Role
 ARN with access to the bucket. This role should have a trust relationship with osis-
pipelines.amazonaws.com\n"
      "BlueprintName":"AWS-ApacheLogPipeline"
   }
}
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用してパイプラインの設計図に関する情報を取得するには、ListPipelineBlueprints および GetPipelineBlueprint オペレーションを使用します。

Amazon OpenSearch Ingestion パイプラインの表示

Amazon OpenSearch Ingestion パイプラインの詳細は、 AWS マネジメントコンソール、 AWS CLI、または OpenSearch Ingestion API を使用して表示できます。

コンソール

パイプラインを表示するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. (オプション) 特定のステータスのパイプラインを表示するには、[任意のステータス] をクリックし、フィルタリングするステータスを選択します。

   パイプラインには、次のステータスがあります。

   - `Active` - パイプラインはアクティブで、データを取り込む準備ができています。
   - `Creating` - パイプラインを作成中です。
   - `Updating` - パイプラインを更新中です。
   - `Deleting` - パイプラインを削除中です。
   - `Create failed` - パイプラインを作成できませんでした。
   - `Update failed` - パイプラインを更新できませんでした。
   - `Stop failed` – パイプラインを停止できませんでした。
   - `Start failed` - パイプラインを開始できませんでした。
   - `Stopping` - パイプラインを停止中です。
   - `Stopped` - パイプラインは停止中で、いつでも再開できます。
   - `Starting` - パイプラインを開始中です。

パイプラインのステータスが `Create failed`、`Creating`、`Deleting`、`Stopped` の場合は、Ingestion OCU の料金は請求されません。

CLI

を使用してパイプラインを表示するには AWS CLI、list-pipelines リクエストを送信します。

```
aws osis list-pipelines
```

リクエストは、既存の全パイプラインのリストを返します。

```
{
    "NextToken": null,
```

```
    "Pipelines": [
        {
            "CreatedAt": 1.671055851E9,
            "LastUpdatedAt": 1.671055851E9,
            "MaxUnits": 4,
            "MinUnits": 2,
            "PipelineArn": "arn:aws:osis:us-west-2:123456789012:pipeline/log-pipeline",
            "PipelineName": "log-pipeline",
            "Status": "ACTIVE",
            "StatusReason": {
                "Description": "The pipeline is ready to ingest data."
            }
        },
            "CreatedAt": 1.671055851E9,
            "LastUpdatedAt": 1.671055851E9,
            "MaxUnits": 2,
            "MinUnits": 8,
            "PipelineArn": "arn:aws:osis:us-west-2:123456789012:pipeline/another-
pipeline",
            "PipelineName": "another-pipeline",
            "Status": "CREATING",
            "StatusReason": {
                "Description": "The pipeline is being created. It is not able to ingest
 data."
            }
        }
    ]
}
```

1 つのパイプラインに関する情報を取得するには、get-pipeline コマンドを使用します。

```
aws osis get-pipeline --pipeline-name "my-pipeline"
```

リクエストは、指定されたパイプラインの設定情報を返します。

```
{
    "Pipeline": {
        "PipelineName": "my-pipeline",
        "PipelineArn": "arn:aws:osis:us-east-1:123456789012:pipeline/my-pipeline",
        "MinUnits": 9,
        "MaxUnits": 10,
        "Status": "ACTIVE",
        "StatusReason": {
```

```
            "Description": "The pipeline is ready to ingest data."
        },
        "PipelineConfigurationBody": "log-pipeline:\n source:\n http:\n processor:\n
 - grok:\n match:\nlog: [ '%{COMMONAPACHELOG}' ]\n - date:\n from_time_received: true
\n destination: \"@timestamp\"\n  sink:\n - opensearch:\n hosts: [ \"https://search-
mdp-performance-test-duxkb4qnycd63rpy6svmvyvfpi.us-east-1.es.amazonaws.com\" ]\n index:
 \"apache_logs\"\n aws_sts_role_arn: \"arn:aws:iam::123456789012:role/my-domain-role
\"\n  aws_region: \"us-east-1\"\n  aws_sigv4: true",,
        "CreatedAt": "2022-10-01T15:28:05+00:00",
        "LastUpdatedAt": "2022-10-21T21:41:08+00:00",
        "IngestEndpointUrls": [
            "my-pipeline-123456789012.us-east-1.osis.amazonaws.com"
        ]
    }
}
```


OpenSearch Ingestion API

OpenSearch Ingestion API を使用して OpenSearch Ingestion パイプラインを表示するには、ListPipelines オペレーションと GetPipeline オペレーションを呼び出します。

Amazon OpenSearch Ingestion パイプラインの更新

Amazon OpenSearch Ingestion パイプラインは AWS マネジメントコンソール、、 AWS CLI、または OpenSearch Ingestion API を使用して更新できます。パイプライン設定を更新すると、OpenSearch Ingestion がブルー/グリーンデプロイを開始します。詳細については、「the section called “パイプライン更新用のブルー/グリーンデプロイ”」を参照してください。

トピック

- 考慮事項
- 必要なアクセス許可
- パイプラインの更新
- パイプライン更新用のブルー/グリーンデプロイ

考慮事項

パイプラインを更新するときは、次の点を考慮します。

- パイプラインの名前またはネットワーク設定を更新することはできません。

- パイプラインが VPC ドメインシンクに書き込みを行う場合、パイプラインを作成した後に、前に戻ってシンクを別の VPC ドメインに変更することはできません。そのパイプラインを削除し、新しいシンクを使って改めて作成する必要があります。シンクを VPC ドメインからパブリックドメイン、パブリックドメインから VPC ドメイン、パブリックドメインから別のパブリックドメインに、それぞれ切り替えることはできます。
- パイプラインシンクは、パブリック OpenSearch Service ドメインと OpenSearch Serverless コレクションの間で、随時切り替えることができます。
- パイプラインのソース、プロセッサー、またはシンクの設定を更新すると、OpenSearch Ingestion がブルー/グリーンデプロイを開始します。詳細については、「the section called “パイプライン更新用のブルー/グリーンデプロイ”」を参照してください。
- パイプラインのソース、プロセッサー、またはシンクの設定を更新すると、OpenSearch Ingestion は、パイプラインが実行している Data Prepper のメジャーバージョンでサポートされている最新のマイナーバージョンに、パイプラインを自動でアップグレードします。このプロセスにより、パイプラインは、最新のバグ修正とパフォーマンスの改善が施されて最新状態に保たれます。
- パイプラインは、停止した場合でも更新することができます。

必要なアクセス許可

OpenSearch Ingestion は、パイプラインの更新に次の IAM アクセス許可を使用します。

- `osis:UpdatePipeline` - パイプラインを更新します。
- `osis:ValidatePipeline` — パイプライン設定が有効かどうかを確認します。
- `iam:PassRole` — パイプラインロールを OpenSearch Ingestion に渡し、ドメインにデータを書き込めるようにします。このアクセス許可は、パイプラインの設定を更新するときのみ必要で、ログの公開や容量制限など、他の設定を変更する場合は必要ありません。

例えば、次のポリシーではパイプラインを更新するためのアクセス許可を付与しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
```

```
            "Action": [
                "osis:UpdatePipeline",
                "osis:ValidatePipeline"
            ]
        },
        {
            "Resource": [
                "arn:aws:iam::111122223333:role/pipeline-role"
            ],
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ]
        }
    ]
}
```

パイプラインの更新

Amazon OpenSearch Ingestion パイプラインは AWS マネジメントコンソール、、 AWS CLI、または OpenSearch Ingestion API を使用して更新できます。

コンソール

パイプラインを更新するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. パイプラインを選択し、設定を開きます。次に、[編集] オプションのいずれかを選択します。
3. 変更が完了したら、[保存] を選択します。

CLI

を使用してパイプラインを更新するには AWS CLI、update-pipeline リクエストを送信します。次のリクエスト例では、新しい設定ファイルをアップロードし、最小容量と最大容量値を更新しています。

```
aws osis update-pipeline \
  --pipeline-name "my-pipeline" \
  --pipline-configuration-body "file://new-pipeline-config.yaml" \
```

```
  --min-units 11 \
  --max-units 18
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用して OpenSearch Ingestion パイプラインを更新するには、UpdatePipeline オペレーションを呼び出します。

パイプライン更新用のブルー/グリーンデプロイ

パイプライン設定を更新すると、OpenSearch Ingestion がブルー/グリーンデプロイプロセスを開始します。

ブルー/グリーンとは、パイプライン更新用の新しい環境を作成し、これらの更新が完了した後に新しい環境にトラフィックをルーティングする方法のことを指します。この方法では、新しい環境へのデプロイに失敗しても、ダウンタイムを最小限に抑えることができ、元の環境を維持することができます。ブルー/グリーンデプロイ自体はパフォーマンスに影響しませんが、パイプラインの設定が、パフォーマンスを変えるような形で変更されると、パフォーマンスが変わる可能性があります。

OpenSearch Ingestion は、ブルー/グリーンデプロイの間は自動スケーリングをブロックします。新しいパイプラインにリダイレクトされるまで、古いパイプラインへのトラフィックに対してのみ、引き続き料金が請求されます。トラフィックがリダイレクトされると、新しいパイプラインに対してのみ料金が請求されます。2 つのパイプラインの料金が同時に請求されることはありません。

パイプラインのソース、プロセッサー、またはシンクの設定を更新すると、OpenSearch Ingestion により、パイプラインが実行している Data Prepper のメジャーバージョンでサポートされている最新のマイナーバージョンに、パイプラインを自動でアップグレードできます。例えば、パイプラインの設定に `version: "2"` があり、OpenSearch Ingestion がまず、バージョン 2.1.0 のパイプラインをプロビジョニングしたとします。バージョン 2.1.1 のサポートが追加され、パイプライン設定に変更を加えると、OpenSearch Ingestion はパイプラインをバージョン 2.1.1 にアップグレードします。

このプロセスにより、パイプラインは、最新のバグ修正とパフォーマンスの改善が施されて最新状態に保たれます。OpenSearch Ingestion は、パイプライン設定内の `version` オプションを手動で変更しない限り、パイプラインのメジャーバージョンを更新することはできません。

Amazon OpenSearch Ingestion パイプラインのコストの管理

Amazon OpenSearch Ingestion で取り込みパイプラインを開始および停止して、ニーズに基づいてデータフローを制御できます。パイプラインを停止すると、設定を保持しながらデータ処理が停止す

るため、再設定せずに再起動できます。これにより、コストの最適化、リソースの使用状況の管理、問題のトラブルシューティングを行うことができます。パイプラインを停止すると、OpenSearch Ingestion は受信データを処理しませんが、以前に取り込まれたデータは OpenSearch で引き続き使用できます。

パイプラインを開始および停止することで、開発やテスト、継続的な可用性を必要としない類似のアクティビティに使用するパイプラインの設定と終了の処理を簡素化できます。パイプラインが停止している間は、Ingestion OCU の時間に対して料金は請求されません。停止したパイプラインは、引き続き更新することが可能で、マイナーバージョンの更新とセキュリティパッチは自動的に受け取られます。

パイプラインを停止して開始すると、プルベースのパイプライン (DDB、S3、DocDB など) のすべてのデータが最初から再処理されます。パイプラインを停止すると、パイプラインによって作成されたサービスマネージド VPC エンドポイントはすべて削除されます。セルフマネージド VPC エンドポイントを持つパイプラインの場合、パイプラインを再起動するときに、アカウントに VPC エンドポイントを再作成する必要があります。詳細については、「the section called “セルフマネージド VPC エンドポイント”」を参照してください。

> **Note**
>
> パイプラインの容量が過剰であっても運用を継続する必要がある場合は、パイプラインを停止して再起動するのではなく、最大容量制限を調整することを検討してください。これにより、パイプラインがデータを効率的に処理し続けながら、コストを管理することができます。詳細については、the section called “パイプラインのスケーリング”を参照してください。

以下のトピックでは、 AWS マネジメントコンソール、、および OpenSearch Ingestion API を使用してパイプラインを開始 AWS CLIおよび停止する方法について説明します。

トピック

- Amazon OpenSearch Ingestion パイプラインの停止
- Amazon OpenSearch Ingestion パイプラインの開始

Amazon OpenSearch Ingestion パイプラインの停止

OpenSearch Ingestion パイプラインを使用したり、管理を実行したりするときは、必ずアクティブなパイプラインから開始します。次に、パイプラインを停止して、その後に、再びパイプラインを

開始します。パイプラインが停止している間は、Ingestion OCU の時間に対して料金は請求されません。

コンソール

パイプラインを停止するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. パイプラインを選択します。このページで停止オペレーションを実行するか、停止させるパイプラインの詳細ページに移動します。
3. [アクション] で [パイプラインを停止] を選択します。

   パイプラインを停止および開始できない場合は、パイプラインを停止のアクションは使用できません。

AWS CLI

を使用してパイプラインを停止するには AWS CLI、以下のパラメータを指定して stop-pipeline コマンドを呼び出します。

- `--pipeline-name` – パイプラインの名前。

Example

```
aws osis stop-pipeline --pipeline-name my-pipeline
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用してパイプラインを停止するには、次のパラメータで StopPipeline オペレーションを呼び出します。

- `PipelineName` – パイプラインの名前。

Amazon OpenSearch Ingestion パイプラインの開始

OpenSearch Ingeestion パイプラインを開始するときは、必ず、常に停止状態になっているパイプラインから始めます。このパイプラインは、容量制限、ネットワーク設定、ログ公開オプションなどの構成設定が維持されています。

パイプラインを再度開始するときは、数分かかります。

コンソール

パイプラインを開始するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. パイプラインを選択します。このページから開始オペレーションを実行するか、開始するパイプラインの詳細ページに移動します。
3. [アクション] で [パイプラインを開始] を選択します。

AWS CLI

を使用してパイプラインを開始するには AWS CLI、以下のパラメータを指定して start-pipeline コマンドを呼び出します。

- `--pipeline-name` – パイプラインの名前。

Example

```
aws osis start-pipeline --pipeline-name my-pipeline
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用して OpenSearch Ingestion パイプラインを開始するには、次のパラメータで StartPipeline オペレーションを呼び出します。

- `PipelineName` – パイプラインの名前。

Amazon OpenSearch Ingestion パイプラインの削除

Amazon OpenSearch Ingestion パイプラインは AWS マネジメントコンソール、、 AWS CLI、または OpenSearch Ingestion API を使用して削除できます。ステータスが `Creating` または `Updating` の場合、パイプラインは削除できません。

コンソール

パイプラインを削除するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. 削除するパイプラインを選択し、[アクション]、[削除] の順に選択します。
3. 削除を確認し、[Delete] (削除) を選択します。

CLI

を使用してパイプラインを削除するには AWS CLI、delete-pipeline リクエストを送信します。

```
aws osis delete-pipeline --pipeline-name "my-pipeline"
```

OpenSearch Ingestion API

OpenSearch Ingestion API を使用して OpenSearch Ingestion パイプラインを削除するには、次のパラメータを指定して DeletePipeline オペレーションを呼び出します。

- `PipelineName` – パイプラインの名前。

Amazon OpenSearch Ingestion パイプラインでサポートされているプラグインとオプション

Amazon OpenSearch Ingestion では、OpenSearch Data Prepper と異なり、ソース、プロセッサ、シンクのサブセットがサポートされています。また、OpenSearch Ingestion では、サポートされている各プラグインで使用できるオプションにいくつかの制限があります。以下のセクションでは、OpenSearch Ingestion でサポートされているプラグインと関連するオプションについて、説明します。

> **Note**
>
> OpenSearch Ingestion では、デフォルトのバッファーが自動的に設定されるため、バッファープラグインはサポートされていません。パイプラインの設定にバッファーを含めると、検証エラーが発生します。

トピック

- サポートされているプラグイン
- ステートレスプロセッサとステートフルプロセッサ
- 設定の要件と制限

サポートされているプラグイン

OpenSearch Ingestion では、以下のData Prepper プラグインがサポートされています。

ソース:

- DocumentDB
- DynamoDB

- HTTP
- Kafka
- Kinesis
- OpenSearch
- OTel logs
- OTel metrics
- OTel trace
- S3

プロセッサ:

- Add entries
- Aggregate
- Anomaly detector
- AWS Lambda
- Convert entry type
- Copy values
- CSV

- 日付
- 遅延
- 解凍
- Delete entries
- 解読
- Drop events
- フラット化
- Geo IP
- Grok
- Key value
- List to map
- Lowercase string
- リストにマップする
- ミューテーションイベント (プロセッサのシリーズ)
- ミューテーション文字列 (プロセッサのシリーズ)
- Obfuscate
- OTel metrics
- OTel trace group
- OTel trace
- Ion を解析
- Parse JSON
- XML を解析
- Rename keys
- エントリを選択
- サービスマップ
- Split even
- Split string
- String converter

- Substitute string
- Trace peer forwarder
- [Translate] (変換)
- Trim string
- 切り詰め
- Uppercase string
- User agent
- Write JSON

シンク:

- OpenSearch (OpenSearch Service、OpenSearch Serverless、Elasticsearch 6.8 以降をサポート)
- S3

シンクコーデック:

- Avro
- NDJSON
- JSON
- Parquet

ステートレスプロセッサとステートフルプロセッサ

ステートレスプロセッサは変換やフィルタリングなどの操作を行い、ステートフルプロセッサは前回の実行結果を記憶する集約などの操作を行います。OpenSearch Ingestion では、ステートフルプロセッサの Aggregate と Service-map がサポートされています。その他のサポートされているプロセッサは、すべてステートレスです。

ステートレスプロセッサのみを含むパイプラインの最大容量制限は、96 Ingestion OCU です。パイプラインにステートレスプロセッサが含まれる場合の最大容量制限は、48 Ingestion OCU です。ただし、パイプラインで永続的バッファリングが有効になっている場合、ステートレスプロセッサのみを使用する場合は 最大 384 Ingestion OCU、ステートフルプロセッサが含まれている場合は 192 Ingestion OCU を使用できます。詳細については、「the section called “パイプラインのスケーリング”」を参照してください。

エンドツーエンドの受信確認は、ステートレスプロセッサでのみサポートされています。詳細については、「the section called “エンドツーエンドの確認応答”」を参照してください。

設定の要件と制限

以下で特に明記されていない限り、上記のサポートされているプラグインに関して Data Prepper の設定リファレンスに記載されているオプションは、すべて OpenSearch Ingestion パイプラインで使用できます。次のセクションでは、OpenSearch Ingestion が特定のプラグインオプションに設定している制限について説明します。

> **Note**
>
> OpenSearch Ingestion では、デフォルトのバッファーが自動的に設定されるため、バッファープラグインはサポートされていません。パイプラインの設定にバッファーを含めると、検証エラーが発生します。

`authentication` や `acm_certificate_arn` など、OpenSearch Ingestion によって多くのオプションが内部で設定され管理されています。`thread_count` や `request_timeout` など、その他のオプションは、手動で変更するとパフォーマンスに影響します。したがって、これらの値は、パイプラインの最適なパフォーマンスを確保するために内部で設定されます。

最後に、`ism_policy_file` や `sink_template` など、オプションの中には OpenSearch Ingestion に渡せないものがあります。これらは、オープンソースの Data Prepper で実行するときローカルファイルになるためです。こうした値はサポートされていません。

トピック

- 一般的なパイプラインのオプション
- Grok プロセッサ
- HTTP ソース
- OpenSearch シンク
- OTel メトリクスソース、OTel トレースソース、OTel ログソース
- OTel トレースグループプロセッサ
- OTel トレースプロセッサ
- サービスマッププロセッサ
- S3 ソース

一般的なパイプラインのオプション

以下の一般的なパイプラインのオプションは、OpenSearch Ingestion により設定され、パイプライン設定ではサポートされていません。

- `workers`
- `delay`

Grok プロセッサ

以下の Grok プロセッサオプションはサポートされていません。

- `patterns_directories`
- `patterns_files_glob`

HTTP ソース

HTTP ソースプラグインには、次の要件および制限があります。

- `path` のオプションは必須です。このパスは `/log/ingest` のような文字列で、ログを取り込むための URI パスを表しています。このパスが、データをパイプラインに送信する際に使用する URI を定義します。例えば、`https://log-pipeline.us-west-2.osis.amazonaws.com/log/ingest`。パスは、スラッシュ (/) から開始し、「-」、「_」、「.」、「/」などの特殊文字や `${pipelineName}` プレースホルダーを含めることができます。
- 以下の HTTP ソースオプションは、OpenSearch Ingestion により設定され、パイプライン設定ではサポートされていません。
  - `port`
  - `ssl`
  - `ssl_key_file`
  - `ssl_certificate_file`
  - `aws_region`
  - `authentication`
  - `unauthenticated_health_check`
  - `use_acm_certificate_for_ssl`
  - `thread_count`

- request_timeout
- max_connection_count
- max_pending_requests
- health_check_service
- acm_private_key_password
- acm_certificate_timeout_millis
- acm_certificate_arn

OpenSearch シンク

OpenSearch シンクのプラグインには、以下の要件および制限があります。

- aws のオプションは必須であり、次のオプションを含んでいる必要があります。
  - sts_role_arn
  - region
  - hosts
  - serverless (シンクが OpenSearch Serverless コレクションの場合)
- sts_role_arn のオプションは、YAML 定義ファイル内で、各シンクで同じロールを指している必要があります。
- hosts のオプションは、OpenSearch Service のドメインエンドポイントまたは OpenSearch Serverless コレクションエンドポイントを指定する必要があります。ドメインのカスタムエンドポイントは指定できません。こちらは、標準エンドポイントでなければなりません。
- hosts のオプションがサーバーレスコレクションのエンドポイントである場合は、serverless のオプションを true に設定する必要があります。さらに、YAML 定義ファイルに index_type のオプションが含まれている場合は、これを management_disabled に設定する必要があります。そうしないと検証に失敗します。
- 次のオプションはサポートされていません。
  - username
  - password
  - cert
  - proxy
  - dlq_file - 失敗したイベントをデッドレターキュー (DLQ) にオフロードする場合は、dlq オプションを使用して S3 バケットを指定します。

- ism_policy_file
- socket_timeout
- template_file
- insecure

OTel メトリクスソース、OTel トレースソース、OTel ログソース

OTel メトリクスソース、OTel トレースソース、OTel ログソースプラグインには、次の要件および制限があります。

- path のオプションは必須です。このパスは /log/ingest のような文字列で、ログを取り込むための URI パスを表しています。このパスが、データをパイプラインに送信する際に使用する URI を定義します。例えば、https://log-pipeline.us-west-2.osis.amazonaws.com/*log/ingest*。パスは、スラッシュ (/) から開始し、「-」、「_」、「.」、「/」などの特殊文字や ${pipelineName} プレースホルダーを含めることができます。
- 次のオプションは、OpenSearch Ingestion で設定されており、パイプライン設定ではサポートされていません。
  - port
  - ssl
  - sslKeyFile
  - sslKeyCertChainFile
  - authentication
  - unauthenticated_health_check
  - useAcmCertForSSL
  - unframed_requests
  - proto_reflection_service
  - thread_count
  - request_timeout
  - max_connection_count
  - acmPrivateKeyPassword
  - acmCertIssueTimeOutMillis
  - health_check_service

- acmCertificateArn
- awsRegion

OTel トレースグループプロセッサ

OTel トレースグループプロセッサには、次の要件および制限があります。

- aws のオプションは必須であり、次のオプションを含んでいる必要があります。
  - sts_role_arn
  - region
  - hosts
- sts_role_arn のオプションは、OpenSearch シンク設定で指定されるパイプラインロールと、同じロールを指定します。
- username、password、cert、insecure のオプションはサポートされていません。
- aws_sigv4 のオプションは必須であり、true に設定する必要があります。
- OpenSearch シンクプラグイン内の serverless のオプションはサポートされていません。Otel トレースグループプロセッサは、現在 OpenSearch Serverless コレクションでは動作しません。
- パイプライン設定本体内の otel_trace_group プロセッサの数は、8 を超えることはできません。

OTel トレースプロセッサ

OTel トレースプロセッサには、次の要件および制限があります。

- trace_flush_interval オプションの値は、300 秒を超えることはできません。

サービスマッププロセッサ

サービスマッププロセッサには、次の要件および制限があります。

- window_duration オプションの値は、300 秒を超えることはできません。

S3 ソース

S3 ソースのプラグインには、次の要件および制限があります。

- aws のオプションは必須であり、region と sts_role_arn のオプションを含んでいる必要があります。
- records_to_accumulate オプションの値は、200 を超えることはできません。
- maximum_messages オプションの値は、10 を超えることはできません。
- 指定する場合は、disable_bucket_ownership_validation オプションを false に設定する必要があります。
- 指定する場合は、input_serialization オプションを parquet に設定する必要があります。

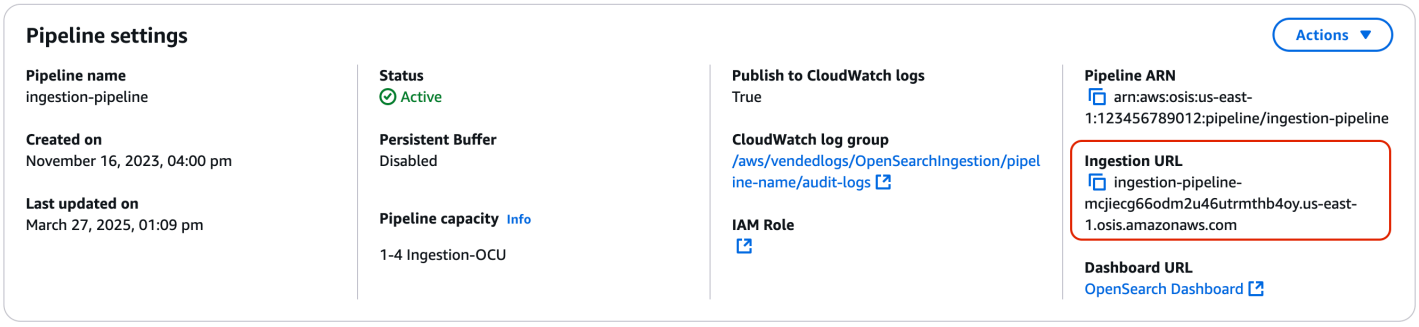
Amazon OpenSearch Ingestion パイプラインを他のサービスやアプリケーションと統合する

Amazon OpenSearch Ingestion パイプラインにデータを正常に取り込むには、パイプラインエンドポイントにデータを送信するように、クライアントアプリケーション (ソース) を設定する必要があります。ソースは Fluent Bit ログ、OpenTelemetry Collector、または単純な S3 バケットなどのクライアントである可能性があります。正確な設定は、クライアントごとに異なります。

ソース設定における重要な違い (OpenSearch Service ドメインまたは OpenSearch Serverless コレクションへのデータの直接送信と比較して) は、 AWS サービス名 (osis) とホストエンドポイントです。これはパイプラインエンドポイントである必要があります。

取り込みエンドポイントの構築

データをパイプラインに取り込むには、取り込みエンドポイントにデータを送信します。取り込み URL を見つけるには、[パイプラインの設定] ページに移動して [取り込み URL] をコピーします。



OTel trace や OTel metrics などのプルベースのソースの完全な取り込みエンドポイントを構築するには、パイプライン設定からの取り込みパスを取り込み URL に追加します。

例えば、パイプライン設定に次のような取り込みパスがあるとします。



クライアント設定で指定した完全な取り込みエンドポイントは、次の形式になります: `https://ingestion-pipeline-abcdefg.us-east-1.osis.amazonaws.com/my/test_path`。

取り込みロールの作成

OpenSearch Ingestion へのリクエストは、すべて署名バージョン 4 で署名する必要があります。少なくとも、リクエストに署名するロールには `osis:Ingest` アクションの許可を与える必要があります。これにより、OpenSearch Ingestion パイプラインにデータを送信できるようになります。

たとえば、次の AWS Identity and Access Management (IAM) ポリシーでは、対応するロールが単一のパイプラインにデータを送信することを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "osis:Ingest",
      "Resource": "arn:aws:osis:us-east-1:111122223333:pipeline/pipeline-name"
    }
  ]
}
```

> **Note**
>
> すべてのパイプラインにロールを使用するには、`Resource` 要素内の ARN をワイルドカード (*) に置き換えます。

クロスアカウント取り込みアクセスの提供

> **Note**
>
> クロスアカウント取り込みアクセスを提供できるのは、パブリックパイプラインに対してのみであり、VPC パイプラインには提供できません。

ソースアプリケーションを格納するアカウントなど AWS アカウント、別の からパイプラインにデータを取り込む必要がある場合があります。パイプラインに書き込むプリンシパルが、パイプライン自体とは別のアカウントにある場合は、パイプラインにデータを取り込むために別の IAM ロールを信頼するようにプリンシパルを設定する必要があります。

クロスアカウント取り込みアクセス許可の設定するには

1. パイプライン AWS アカウント と同じ 内に アクセス`osis:Ingest`許可 (前のセクションで説明) を持つ取り込みロールを作成します。手順については、「IAM ロールの作成」を参照してください。
2. 取り込みロールに信頼ポリシーをアタッチして、別のアカウントのプリンシパルがそのポリシーを引き受けることができるようにします。

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [{
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:root"
         },
        "Action": "sts:AssumeRole"
     }]
   }
   ```

3. もう一方のアカウントで、取り込みロールを引き受けるようにクライアントアプリケーション (例: Fluent Bit) を設定します。これを機能させるには、アプリケーションアカウントは、アプリケーションユーザーまたはロールに取り込みロールを引き受ける許可を付与する必要があります。

次のアイデンティティベースポリシーの例では、アタッチされたプリンシパルがパイプラインアカウントから ingestion-role を引き受けることを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/ingestion-role"
    }
  ]
}
```

その後、クライアントアプリケーションは AssumeRole オペレーションを使用して ingestion-role を引き受け、関連するパイプラインにデータを取り込むことができます。

Atlassian サービスでの OpenSearch Ingestion パイプラインの使用

Atlassian Jira および Confluence ソースプラグインを使用して、Atlassian サービスからのデータを OpenSearch Ingestion パイプラインにデータを取り込むことができます。これらの統合により、継続的なモニタリングと更新の自動同期を通じてリアルタイムの関連性を維持しながら、完全な Jira プロジェクトと Confluence スペースを同期することで、統合された検索可能なナレッジベースを作成できます。

Integrating with Jira

Jira コンテンツを OpenSearch に統合することで、強力なコンテキスト検索機能を使用して Jira エクスペリエンスを変換します。Data Prepper Atlassian Jira ソースプラグインを使用すると、継続的なモニタリングと更新の自動同期を通じてリアルタイムの関連性を維持しながら、完全な Jira プロジェクトを同期することで、統合された検索可能なナレッジベースを作成できます。この統合により、特定のプロジェクト、問題タイプ、ステータスの柔軟なフィルタリングオプションによるデータ同期が可能になり、必要な情報のみがインポートされます。

安全で信頼性の高い接続を確保するために、プラグインは基本的な API キー認証や OAuth2 認証など、複数の認証方法をサポートし、 AWS Secrets Managerに保存されているシークレットを使

用して認証情報を管理するセキュリティを強化します。また、中断のないアクセスのためのトークンの自動更新機能も備えているため、継続的なオペレーションが保証されます。Atlassian の API バージョン 2 上に構築されたこの統合により、チームは OpenSearch の高度な検索機能を通じて Jira データから貴重なインサイトを引き出すことができます。

Integrating with Confluence

Data Prepper の Confluence ソースプラグインを使用して Atlassian Confluence コンテンツを OpenSearch に統合することで、チームのナレッジ管理とコラボレーション機能を強化します。この統合により、一元化された検索可能な集合的知識のリポジトリを作成し、情報検出とチームの生産性を向上させることができます。Confluence コンテンツを同期し、更新を継続的にモニタリングすることで、プラグインは OpenSearch インデックスが常に最新かつ包括的な状態に保たれるようにします。

統合には柔軟なフィルタリングオプションが用意されているため、特定のスペースやページタイプからコンテンツを選択的にインポートし、同期されたコンテンツを組織のニーズに合わせて調整できます。プラグインは、基本的な API キーと OAuth2 認証方法の両方をサポートし、AWS Secrets Managerを介して認証情報を安全に管理することもできます。プラグインの自動トークン更新機能により、中断のないアクセスとシームレスな操作が保証されます。Atlassian の Confluence API 上に構築されたこの統合により、チームは Confluence コンテンツ全体で OpenSearch の高度な検索機能を活用して、組織内の情報のアクセシビリティと使用率を高めることができます。

トピック

- 前提条件
- パイプラインロールの設定
- Jira コネクタパイプライン設定
- Confluence コネクタパイプライン設定
- データ整合性
- 制限事項
- アトラシアンコネクタの CloudWatch のメトリクス
- OAuth 2.0 を使用して Amazon OpenSearch Ingestion パイプラインを Atlassian Jira または Confluence に接続する

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を完了します。

1. 次のいずれかのオプションを選択して、Jira サイトの認証情報を準備します。OpenSearch Ingestion では、コンテンツに対する `ReadOnly` 認可のみが必要です。

   a. オプション 1: API キー – Atlassian アカウントにログインし、次のトピックの情報を使用して API キーを生成します。

   - Atlassian アカウントの API トークンを管理する

   b. オプション 2: OAuth2 – Atlassian アカウントにログインし、the section called “OAuth 2.0 を使用して Amazon OpenSearch Ingestion パイプラインを Atlassian Jira または Confluence に接続する” の情報を使用します。

2. AWS Secrets Managerにシークレットを作成し、前の手順で作成した認証情報の詳細を保存します。手順に従って、次の選択を行います。

   - [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
   - [キー/値のペア] の場合、選択した認可タイプに応じて次のペアを作成します。

   API key

   ```
   {
      "username": user-name-usualy-email-id,
      "password": api-key
   }
   ```

   OAuth 2.0

   ```
   {
      "clientId": client-id
      "clientSecret": client-secret
      "accessKey": access-key
      "refreshKey": refresh-key
   }
   ```

   シークレットを作成したら、シークレットの Amazon リソースネーム (ARN) をコピーします。パイプラインロールのアクセス許可ポリシーに含めます。

パイプラインロールの設定

パイプラインで渡されるロールには、前提条件セクションで作成されたシークレットを読み書きするための次のポリシーがアタッチされている必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretReadWrite",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret-name-random-6-characters"
        }
    ]
}
```

ロールには、選択したシンクにアクセスして書き込むためのポリシーもアタッチする必要があります。例えば、シンクとして OpenSearch を選択した場合、ポリシーは次のようになります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "OpenSearchWritePolicy",
            "Effect": "Allow",
            "Action": "aoss:*",
            "Resource": "arn:aws:aoss:us-
east-1:111122223333:collection/collection-id"
```

```
        }
    ]
}
```

Jira コネクタパイプライン設定

事前設定された Atlassian Jira ブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

*#########*を、ユーザー自身の情報に置き換えます。

```
version: "2"
extension:
  aws:
    secrets:
      jira-account-credentials:
        secret_id: "secret-arn"
        region: "secret-region"
        sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
atlassian-jira-pipeline:
  source:
    jira:
      # We only support one host url for now
      hosts: ["jira-host-url"]
      acknowledgments: true
      authentication:
        # Provide one of the authentication method to use. Supported methods are
 'basic' and 'oauth2'.
        # For basic authentication, password is the API key that you generate using
 your jira account
        basic:
          username: ${{aws_secrets:jira-account-credentials:username}}
          password: ${{aws_secrets:jira-account-credentials:password}}
        # For OAuth2 based authentication, we require the following 4 key values stored
 in the secret
        # Follow atlassian instructions at the below link to generate these keys.
        # https://developer.atlassian.com/cloud/confluence/oauth-2-3lo-apps/
        # If you are using OAuth2 authentication, we also require, write permission to
 your AWS secret to
        # be able to write the renewed tokens back into the secret.
        # oauth2:
          # client_id: ${{aws_secrets:jira-account-credentials:clientId}}
```

```
          # client_secret: ${{aws_secrets:jira-account-credentials:clientSecret}}
          # access_token: ${{aws_secrets:jira-account-credentials:accessToken}}
          # refresh_token: ${{aws_secrets:jira-account-credentials:refreshToken}}
      filter:
        project:
          key:
            include:
              # This is not project name.
              # It is an alphanumeric project key that you can find under project
 details in Jira.
              - "project-key"
              - "project-key"
            # exclude:
              # - "project-key"
              # - "project-key"
        issue_type:
          include:
            - "issue-type"
            # - "Story"
            # - "Bug"
            # - "Task"
         # exclude:
             # - "Epic"
        status:
          include:
            - "ticket-status"
            # - "To Do"
            # - "In Progress"
            # - "Done"
         # exclude:
           # - "Backlog"

  sink:
    - opensearch:
        # Provide an Amazon OpenSearch Service domain endpoint
        hosts: [ "https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-
east-1.es.amazonaws.com" ]
        index: "index_${getMetadata(\"project\")}"
        # Ensure adding unique document id which is the unique ticket id in this case
        document_id: '${/id}'
        aws:
          # Provide a Role ARN with access to the domain. This role should have a trust
 relationship with osis-pipelines.amazonaws.com
          sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
```

```
          # Provide the region of the domain.
          region: "us-east-1"
          # Enable the 'serverless' flag if the sink is an Amazon OpenSearch Serverless
collection
          serverless: false
          # serverless_options:
            # Specify a name here to create or update network policy for the serverless
collection
            # network_policy_name: "network-policy-name"
        # Enable the 'distribution_version' setting if the Amazon OpenSearch Service
domain is of version Elasticsearch 6.x
        # distribution_version: "es6"
        # Enable and switch the 'enable_request_compression' flag if the default
compression setting is changed in the domain.
        # See the section called "HTTP ########"
        # enable_request_compression: true/false
        # Optional: Enable the S3 DLQ to capture any failed requests in an S3 bucket.
Delete this entire block if you don't want a DLQ.
        dlq:
          s3:
            # Provide an S3 bucket
            bucket: "your-dlq-bucket-name"
            # Provide a key path prefix for the failed requests
            # key_path_prefix: "kinesis-pipeline/logs/dlq"
            # Provide the region of the bucket.
            region: "us-east-1"
            # Provide a Role ARN with access to the bucket. This role should have a
trust relationship with osis-pipelines.amazonaws.com
            sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
```

Jira ソースの属性へのキー:

1. hosts: Jira クラウドまたはオンプレミス URL。一般的に、`https://your-domain-name.atlassian.net/` のようになります。
2. acknowledgments: シンクへのデータの配信を保証します。
3. authentication: パイプラインが Jira インスタンスにアクセスする方法について説明します。`Basic` または `OAuth2`を選択し、シー AWS クレット内のキーを参照する対応するキー属性を指定します。
4. filter: このセクションでは、Jira データのどの部分を抽出して同期するかを選択するのに役立ちます。

a. project: 同期するプロジェクトキーを include セクションに一覧表示します。それ以外の場合は、exclude セクションで除外するプロジェクトを一覧表示します。任意の時点で包含オプションまたは除外オプションを 1 つだけ指定します。

b. issue_type: 同期する特定の問題タイプ。ニーズに合った同様のパターン include または exclude パターンに従います。添付ファイルは元の添付ファイルへのアンカーリンクとして表示されますが、添付ファイルの内容は抽出されないことに注意してください。

c. status: データ抽出クエリに適用する特定のステータスフィルター。include を指定した場合、これらのステータスのチケットのみが同期されます。exclude を指定した場合、除外ステータスがリストされているチケットを除くすべてのチケットが同期されます。

Confluence コネクタパイプライン設定

事前設定された Atlassian Confluence ブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

```
version: "2"
extension:
  aws:
    secrets:
      confluence-account-credentials:
        secret_id: "secret-arn"
        region: "secret-region"
        sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
atlassian-confluence-pipeline:
  source:
    confluence:
      # We currently support only one host URL.
      hosts: ["confluence-host-url"]
      acknowledgments: true
      authentication:
        # Provide one of the authentication method to use. Supported methods are
 'basic' and 'oauth2'.
        # For basic authentication, password is the API key that you generate using
 your Confluence account
        basic:
          username: ${{aws_secrets:confluence-account-credentials:confluenceId}}
          password: ${{aws_secrets:confluence-account-
credentials:confluenceCredential}}
        # For OAuth2 based authentication, we require the following 4 key values stored
 in the secret
```

```
        # Follow atlassian instructions at the following link to generate these keys:
        # https://developer.atlassian.com/cloud/confluence/oauth-2-3lo-apps/
        # If you are using OAuth2 authentication, we also require write permission to
 your AWS secret to
        # be able to write the renewed tokens back into the secret.
        # oauth2:
          # client_id: ${{aws_secrets:confluence-account-credentials:clientId}}
          # client_secret: ${{aws_secrets:confluence-account-credentials:clientSecret}}
          # access_token: ${{aws_secrets:confluence-account-credentials:accessToken}}
          # refresh_token: ${{aws_secrets:confluence-account-credentials:refreshToken}}
      filter:
        space:
          key:
            include:
              # This is not space name.
              # It is a space key that you can find under space details in Confluence.
              - "space key"
              - "space key"
           # exclude:
             #  - "space key"
             #  - "space key"
        page_type:
          include:
            - "content type"
            # - "page"
            # - "blogpost"
            # - "comment"
         # exclude:
            # - "attachment"

  sink:
    - opensearch:
        # Provide an Amazon OpenSearch Service domain endpoint
        hosts: [ "https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-
east-1.es.amazonaws.com" ]
         index: "index_${getMetadata(\"space\")}"
        # Ensure adding unique document id which is the unique ticket ID in this case.
        document_id: '${/id}'
        aws:
          # Provide the Amazon Resource Name (ARN) for a role with access to the
 domain. This role should have a trust relationship with osis-pipelines.amazonaws.com.
          sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
          # Provide the Region of the domain.
          region: "us-east-1"
```

```
          # Enable the 'serverless' flag if the sink is an Amazon OpenSearch Serverless
collection
          serverless: false
          # serverless_options:
            # Specify a name here to create or update network policy for the serverless
collection.
            # network_policy_name: "network-policy-name"
        # Enable the 'distribution_version' setting if the Amazon OpenSearch Service
domain is of version Elasticsearch 6.x
        # distribution_version: "es6"
        # Enable and switch the 'enable_request_compression' flag if the default
compression setting is changed in the domain.
        # For more information, see the section called “HTTP ########”.
        # enable_request_compression: true/false
        # Optional: Enable the S3 DLQ to capture any failed requests in an S3 bucket.
Delete this entire block if you don't want a DLQ.
        dlq:
          s3:
            # Provide an S3 bucket
            bucket: "your-dlq-bucket-name"
            # Provide a key path prefix for the failed requests
            # key_path_prefix: "kinesis-pipeline/logs/dlq"
            # Provide the Rregion of the bucket.
            region: "us-east-1"
            # Provide the Amazon Resource Name (ARN) for a role with access to the
bucket. This role should have a trust relationship with osis-pipelines.amazonaws.com
            sts_role_arn: "arn:aws:iam::123456789012:role/Example-Role"
```

Confluence ソースのキー属性:

1. hosts: Confluence クラウドまたはオンプレミス URL。一般的に、`https://your-domain-name.atlassian.net/` のようになります
2. acknowledgments: シンクへのデータの配信を保証します。
3. authentication: パイプラインが Confluence インスタンスにアクセスする方法について説明します。`Basic` または `OAuth2`を選択し、シー AWS クレット内のキーを参照する対応するキー属性を指定します。
4. filter: このセクションでは、Confluence データのどの部分を抽出して同期するかを選択するのに役立ちます。

a. space: 同期するスペースキーを `include` セクションに一覧表示します。それ以外の場合は、`exclude` セクションで除外するスペースを一覧表示します。任意の時点で包含オプションまたは除外オプションを 1 つだけ指定します。

b. page_type: 同期する特定のページタイプ (ページ、ブログ投稿、添付ファイルなど)。ニーズに合った同様のパターン `include` または `exclude` パターンに従います。添付ファイルは元の添付ファイルへのアンカーリンクとして表示されますが、添付ファイルの内容は抽出されないことに注意してください。

データ整合性

パイプライン YAML で指定されたフィルターに基づいて、選択したプロジェクト (またはスペース) が 1 回抽出され、ターゲットシンクに完全に同期されます。その後、継続的な変更モニタリングは、変更が発生したときに変更をキャプチャし、シンク内のデータを更新します。1 つの例外は、変更モニタリングが `delete` アクションではなく、アクション `create` と `update` アクションのみを同期することです。

制限事項

- ユーザー削除アクションは同期されません。シンクに記録されたデータは、シンクに残ります。ID マッピングがシンク設定で指定されている場合、更新によって既存のコンテンツが新しい変更で上書きされます。
- 次の API をサポートしていない古いバージョンの Atlassian ソフトウェアを使用するオンプレミスインスタンスは、このソースと互換性がありません。
  - Jira Search API バージョン 3
    - `rest/api/3/search`
    - `rest/api/3/issue`
  - Confluence
    - `wiki/rest/api/content/search`
    - `wiki/rest/api/content`
    - `wiki/rest/api/settings/systemInfo`

アトラシアンコネクタの CloudWatch のメトリクス

タイプ: Jira コネクタメトリクス

ソース	メトリクス	メトリクスタイプ
acknowledgementSetSuccesses.count	Counter	確認が有効になっている場合、このメトリクスは正常に同期されたチケットの数を提供します。
acknowledgementSetFailures.count	Counter	acknowledgments が有効になっている場合、このメトリクスは同期に失敗したチケットの数を提供します。
crawlingTime.avg	Timer	すべての新しい変更をクロールするのにかかった時間。
ticketFetchLatency.avg	Timer	チケットフェッチ API レイテンシーの平均。
ticketFetchLatency.max	Timer	チケットフェッチ API の最大レイテンシー。
ticketsRequested.count	Counter	チケットフェッチリクエストの数。
ticketRequestedFailed.count	Counter	チケットフェッチリクエストの数が失敗しました。
ticketRequestedSuccess.count	Counter	成功したチケットフェッチリクエストの数。

ソース	メトリクス	メトリクスタイプ
searchCallLatency.avg	Timer	API コールレイテンシーの平均を検索します。
searchCallLatency.max	Timer	API コールの最大レイテンシーを検索します。
searchResultsFound.count	Counter	特定の検索呼び出しで見つかった項目の数。
searchRequestFailed.count	Counter	検索 API コールの失敗数。
authFailures.count	Counter	認証の失敗数。

タイプ: Confluence コネクタメトリクス

ソース	メトリクス	メトリクスタイプ
acknowledgementSetSuccesses.count	Counter	acknowledgments が有効になっている場合、このメトリクスは正常に同期されたページ数を提供します。
acknowledgementSetFailures.count	Counter	acknowledgments が有効になっている場合、このメトリクスは同期に失敗したページ数を提供します。
crawlingTime.avg	Timer	すべての新しい変更をクロールするのにかかった時間。

ソース	メトリクス	メトリクスタイプ
pageFetchLatency.avg	Timer	API レイテンシー (平均) を取得するコンテンツ。
pageFetchLatency.max	Timer	API レイテンシー (最大) を取得するコンテンツ。
pagesRequested.count	Counter	コンテンツフェッチ API の呼び出しの数。
pageRequestFailed.count	Counter	コンテンツフェッチ API の失敗したリクエストの数。
pageRequestedSuccess.count	Counter	コンテンツフェッチ API の成功したリクエストの数。
searchCallLatency.avg	Timer	API コールレイテンシーの平均を検索します。
searchCallLatency.max	Timer	API コールの最大レイテンシーを検索します。
searchResultsFound.count	Counter	特定の検索呼び出しで見つかった項目の数。
searchRequestsFailed.count	Counter	検索 API コールの失敗数。

ソース	メトリクス	メトリクスタイプ
authFailures.count	Counter	認証の失敗数。

OAuth 2.0 を使用して Amazon OpenSearch Ingestion パイプラインを Atlassian Jira または Confluence に接続する

このトピックの情報は、OAuth 2.0 認証を使用して Amazon OpenSearch Ingestion パイプラインを設定して Jira または Confluence アカウントに接続するのに役立ちます。が Atlassian Services で OpenSearch Ingestion パイプラインを使用するための the section called “前提条件” を完了し、API キー認証情報を使用しない場合は、このタスクを実行します。

トピック

- OAuth 2.0 統合アプリケーションを作成する
- Atlassian Developer アクセストークンの生成と更新

OAuth 2.0 統合アプリケーションを作成する

次の手順を使用して、Atlassian Developer ウェブサイトで OAuth 2.0 統合アプリケーションを作成します。

OAuth 2.0 統合アプリケーションを作成するには

1. https://developer.atlassian.com/console/myapps/ で Atlassian Developer アカウントにログインします。
2. [作成]、[OAuth 2.0 統合] を選択します。
3. [名前] には、アプリケーションの目的を識別する名前を入力します。
4. 次に、[Atlassian の開発者規約に同意する] チェックボックスをオンにして [作成] を選択します。
5. 左側のナビゲーションで、[認可] を選択し、[追加] を選択します。
6. [コールバック URL] には、**`https://www.amazon.com`** や **`https://www.example.com`** などの任意の URL を入力し、[変更を保存] を選択します。

7. 左側のナビゲーションで、[アクセス許可] ページを選択し、Jira API の行で [追加] を選択し、[構成] を選択します。すべての Classic Scopes Read のアクセス許可 (以下のリストを参照) をすべて選択し、[保存] を選択します。
8. [きめ細かいスコープ] タブを選択し、[スコープを編集] を選択して [Jira API を編集] ダイアログボックスを開きます。
9. 使用しているソースプラグインのアクセス許可を選択します。

   Jira

   ```
   read:audit-log:jira
   read:issue:jira
   read:issue-meta:jira
   read:attachment:jira
   read:comment:jira
   read:comment.property:jira
   read:field:jira
   read:field.default-value:jira
   read:field.option:jira
   read:field-configuration-scheme:jira
   read:field-configuration:jira
   read:issue-link:jira
   read:issue-link-type:jira
   read:issue-link-type:jira
   read:issue.remote-link:jira
   read:issue.property:jira
   read:resolution:jira
   read:issue-details:jira
   read:issue-type:jira
   read:issue-worklog:jira
   read:issue-field-values:jira
   read:issue.changelog:jira
   read:issue.transition:jira
   read:issue.vote:jira
   read:jira-expressions:jira
   ```

   Confluence

   ```
   read:content:confluence
   read:content-details:confluence
   read:space-details:confluence
   read:audit-log:confluence
   ```

```
read:page:confluence
read:blogpost:confluence
read:custom-content:confluence
read:comment:confluence
read:space:confluence
read:space.property:confluence
read:space.setting:confluence
read:content.property:confluence
read:content.metadata:confluence
read:task:confluence
read:whiteboard:confluence
read:app-data:confluence
manage:confluence-configuration
```

10. [保存] を選択します。

関連情報については、Atlassian Developer ウェブサイトの「Implementing OAuth 2.0 (3LO)」と「Determining the scopes required for an operation」を参照してください。

Atlassian Developer アクセストークンの生成と更新

Atlassian Developer ウェブサイトで Atlassian Developer アクセストークンを生成および更新するには、次の手順を使用します。

Jira アクセストークンを生成して更新するには

1. https://developer.atlassian.com/console/myapps/ で Atlassian Developer アカウントにログインします。
2. 「the section called “OAuth 2.0 統合アプリケーションを作成する”」で作成したアプリを選択します。
3. 左側のナビゲーションメニューで [認可] を選択します。
4. ページの下部から Atlassian API 認可 URL の値をコピーし、任意のテキストエディタに貼り付けます。

   URL の形式は次のとおりです。

```
https://auth.atlassian.com/authorize?
audience=api.atlassian.com
&client_id=YOUR_CLIENT_ID
&scope=REQUESTED_SCOPE%20REQUESTED_SCOPE_TWO
&redirect_uri=https://YOUR_APP_CALLBACK_URL
```

```
&state=YOUR_USER_BOUND_VALUE
&response_type=code
&prompt=consent
```

5. state=YOUR_USER_BOUND_VALUE については、パラメータ値を state=「**sample_text**」など、任意の値に変更します。

   詳細については、Atlassian Developer ウェブサイトの「What is the state parameter used for?」を参照してください。

6. scope セクションには、前のタスクで選択した詳細なスコープが一覧表示されていることに注意してください。例: scope=read%3Ajira-work%20read%3Ajira-user%20offline_access

   offline_access は、refresh_token を生成することを示します。

7. ウェブブラウザウィンドウを開き、コピーした認可 URL をブラウザウィンドウのアドレスバーに入力します。

8. ターゲットページが開いたら、情報が正しいことを確認し、[承認] を選択すると、Jira または Confluence のホームページにリダイレクトされます。

9. ホームページがロードされたら、このページの URL をコピーします。URL にはお使いのアプリケーション用の認証コードが含まれています。このコードを使用してアクセストークンを生成します。code= の後のセクション全体が認証コードです。

10. アクセストークンを生成するには、次の cURL コマンドを使用します。*#########*を、ユーザー自身の情報に置き換えます。

> **Tip**
>
> Postman などのサードパーティーサービスを使用することもできます。

```
curl --request POST --url 'https://auth.atlassian.com/oauth/token' \
--header 'Content-Type: application/json' \
--data '{"grant_type": "authorization_code",
"client_id": "YOUR_CLIENT_ID",
"client_secret": "YOUR_CLIENT_SECRET",
"code": "AUTHORIZATION_CODE",
"redirect_uri": "YOUR_CALLBACK_URL"}'
```

このコマンドへのレスポンスには、access_code と refresh_token の値が含まれます。

Amazon Aurora で OpenSearch Ingestion パイプラインの使用

Amazon Aurora または OpenSearch Ingestion パイプラインを使用すると、既存のデータをエクスポートしたり、変更 (作成、更新、削除など) を Amazon OpenSearch Service のドメインやコレクションにストリーミングしたりできます。OpenSearch Ingestion パイプラインには変更データキャプチャ (CDC) インフラストラクチャが組み込まれており、Amazon Aurora からデータを継続的にストリーミングするための大規模で低レイテンシーの方法を提供します。Aurora MySQL と Aurora PostgreSQL がサポートされています。

データを処理するソースとして Amazon Aurora を使用するには、完全な初期スナップショットを使用する方法と使用しない方法の 2 つがあります。完全な初期スナップショットは、指定されたテーブルのスナップショットであり、このスナップショットは Amazon S3 にエクスポートされます。そこから、OpenSearch Ingestion パイプラインによってドメイン内の 1 つのインデックスに送信されるか、またはドメイン内の複数のインデックスにパーティショニングされます。Amazon Aurora と OpenSearch のデータの整合性を保つために、パイプラインは、Amazon Aurora クラスター内のすべての作成、更新、削除イベントを、(1 つまたは複数の) OpenSearch インデックスに保存されているドキュメントと同期します。

完全な初期スナップショットを使用する場合、OpenSearch Ingestion パイプラインは、まずスナップショットを取り込んでから、Amazon Aurora 変更ストリームからのデータの読み取りを開始します。最終的には追い付いて、Amazon Aurora と OpenSearch の間でほぼリアルタイムのデータ整合性を維持します。

OpenSearch Ingestion と Amazon Aurora の統合を使用して、変更データのキャプチャを追跡し、Aurora のすべての更新を OpenSearch に取り込むこともできます。他のメカニズムの完全なスナップショットが既にある場合、または Amazon Aurora クラスターのデータに対するすべての変更をキャプチャする場合は、このオプションを選択します。

このオプションを選択する場合は、Aurora MySQL のバイナリログ記録を設定するか、クラスターで Aurora PostgreSQL の論理レプリケーションを設定する必要があります。

トピック

- Aurora MySQL
- Aurora PostgreSQL
- 複数のテーブルからのデータを結合して 1 つのドキュメントに取り込む

Aurora MySQL

Amazon Aurora for Aurora MySQL で OpenSearch Ingestion パイプラインを設定するには、次の手順を完了します。

トピック

- Aurora MySQL の前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- データ整合性
- データ型のマッピング
- 制限事項
- 推奨される CloudWatch アラーム

Aurora MySQL の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. バイナリログを構成するために、Amazon Aurora でカスタムの Aurora DB クラスター パラメータグループを作成します。

   ```
   aurora_enhanced_binlog=1
   binlog_backup=0
   binlog_format=ROW
   binlog_replication_globaldb=0
   binlog_row_image=full
   binlog_row_metadata=full
   ```

   さらに、binlog_transaction_compression パラメータが ON に設定されていないこと、および binlog_row_value_options パラメータが PARTIAL_JSON に設定されていないことを確認してください。

2. Aurora MySQL DB クラスターを選択または作成し、前のステップで作成したパラメータグループを DB クラスターに関連付けます。
3. バイナリログの保持期間を 24 時間以上に設定します。

4. Amazon Aurora クラスターで Aurora と AWS Secrets Managerのパスワード管理を使用して、ユーザー名とパスワード認証を設定します。Secrets Manager シークレットを作成するで、ユーザー名とパスワードの組み合わせを作成することもできます。
5. 完全な初期スナップショット機能を使用する場合は、 AWS KMS key と Amazon Aurora から Amazon S3 にデータをエクスポートするための IAM ロールを作成します。

   IAM ロールには、次の許可ポリシーが必要です。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::s3-bucket-used-in-pipeline",
                "arn:aws:s3:::s3-bucket-used-in-pipeline/*"
            ]
        }
    ]
}
```

   ロールにはまた、次の信頼関係が必要です:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "export.rds.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

6. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを選択または作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
7. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は Amazon Aurora DB クラスターからドメインまたはコレクションにデータを書き込むことができます。

ステップ 1: パイプラインロールを設定する

Amazon Aurora パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定します。また、Amazon Aurora ソースの次のアクセス許可をロールに追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
    "Sid": "allowReadingFromS3Buckets",
    "Effect": "Allow",
    "Action": [
    "s3:GetObject",
    "s3:DeleteObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:PutObject"
    ],
    "Resource": [
    "arn:aws:s3:::s3_bucket",
    "arn:aws:s3:::s3_bucket/*"
    ]
    },
    {
    "Sid": "allowNetworkInterfacesActions",
    "Effect": "Allow",
```

```
    "Action": [
    "ec2:AttachNetworkInterface",
    "ec2:CreateNetworkInterface",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:DetachNetworkInterface",
    "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": [
    "arn:aws:ec2:*:111122223333:network-interface/*",
    "arn:aws:ec2:*:111122223333:subnet/*",
    "arn:aws:ec2:*:111122223333:security-group/*"
    ]
    },
    {
    "Sid": "allowDescribeEC2",
    "Effect": "Allow",
    "Action": [
    "ec2:Describe*"
    ],
    "Resource": "*"
    },
    {
    "Sid": "allowTagCreation",
    "Effect": "Allow",
    "Action": [
    "ec2:CreateTags"
    ],
    "Resource": "arn:aws:ec2:*:111122223333:network-interface/*",
    "Condition": {
    "StringEquals": {
    "aws:RequestTag/OSISManaged": "true"
    }
    }
    },
    {
    "Sid": "AllowDescribeInstances",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBInstances"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:db:*"
```

```
    ]
    },
    {
    "Sid": "AllowDescribeClusters",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBClusters"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id"
    ]
    },
    {
    "Sid": "AllowSnapshots",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBClusterSnapshots",
    "rds:CreateDBClusterSnapshot",
    "rds:AddTagsToResource"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id",
    "arn:aws:rds:us-east-2:111122223333:cluster-snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowExport",
    "Effect": "Allow",
    "Action": [
    "rds:StartExportTask"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id",
    "arn:aws:rds:us-east-2:111122223333:cluster-snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowDescribeExports",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeExportTasks"
    ],
    "Resource": "*",
    "Condition": {
```

```
    "StringEquals": {
    "aws:RequestedRegion": "us-east-2",
    "aws:ResourceAccount": "111122223333"
    }
    }
    },
    {
    "Sid": "AllowAccessToKmsForExport",
    "Effect": "Allow",
    "Action": [
    "kms:Decrypt",
    "kms:Encrypt",
    "kms:DescribeKey",
    "kms:RetireGrant",
    "kms:CreateGrant",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
    ],
    "Resource": [
    "arn:aws:kms:us-east-2:111122223333:key/export-key-id"
    ]
    },
    {
    "Sid": "AllowPassingExportRole",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": [
    "arn:aws:iam::111122223333:role/export-role"
    ]
    },
    {
    "Sid": "SecretsManagerReadAccess",
    "Effect": "Allow",
    "Action": [
    "secretsmanager:GetSecretValue"
    ],
    "Resource": [
    "arn:aws:secretsmanager:*:111122223333:secret:*"
    ]
    }
    ]
    }
```

ステップ 2: パイプラインを作成する

次のような OpenSearch Ingestion パイプラインを設定します。サンプルパイプラインは、Amazon Aurora クラスターをソースとして指定します。

```
version: "2"
aurora-mysql-pipeline:
  source:
    rds:
      db_identifier: "cluster-id"
      engine: aurora-mysql
      database: "database-name"
      tables:
        include:
          - "table1"
          - "table2"
      s3_bucket: "bucket-name"
      s3_region: "bucket-region"
      s3_prefix: "prefix-name"
      export:
        kms_key_id: "kms-key-id"
        iam_role_arn: "export-role-arn"
      stream: true
      aws:
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        region: "us-east-1"
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
        index: "${getMetadata(\"table_name\")}"
        index_type: custom
        document_id: "${getMetadata(\"primary_key\")}"
        action: "${getMetadata(\"opensearch_action\")}"
        document_version: "${getMetadata(\"document_version\")}"
        document_version_type: "external"
        aws:
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
          region: "us-east-1"
extension:
  aws:
    secrets:
```

```
      secret:
        secret_id: "rds-secret-id"
        region: "us-east-1"
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        refresh_interval: PT1H
```

事前設定された Amazon Aurora ブループリントを使用して、このパイプラインを作成できます。詳細については、「ブループリントの使用」を参照してください。

Amazon Aurora をソースとして使用するには、パイプラインの VPC アクセスを設定する必要があります。選択する VPC は、Amazon Aurora ソースが使用するのと同じ VPC である必要があります。次に、1 つ以上のサブネットと 1 つ以上の VPC セキュリティグループを選択します。パイプラインには Aurora MySQL データベースへのネットワークアクセスが必要であるため、パイプラインの VPC セキュリティグループからデータベースポートへのインバウンドトラフィックを許可する VPC セキュリティグループが Aurora クラスターに設定されていることも確認する必要があります。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

を使用してパイプライン AWS マネジメントコンソール を作成する場合は、ソースとして Amazon Aurora を使用するには、パイプラインを VPC にアタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] チェックボックスをオンにして、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自のものを選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。

カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion と Amazon Aurora 間の IP アドレスの衝突を回避するには、Amazon Aurora VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認します。

詳細については、「Configuring VPC access for a pipeline」を参照してください。

データ整合性

パイプラインは、Amazon Aurora クラスターから変更を継続的にポーリングまたは受信し、OpenSearch インデックス内の対応するドキュメントを更新することで、データ整合性を確保します。

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了とし

てマークします。OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込む場合は、パイプラインがドキュメント ID を生成しないため、パイプラインシンク設定`document_id: "${getMetadata(\"primary_key\")}"`で を省略する必要があることに注意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、Amazon Aurora 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更と整合されます。

データ型のマッピング

OpenSearch Ingestion パイプラインは、MySQL データ型を、消費する OpenSearch Service ドメインまたはコレクションに適した表現にマッピングします。OpenSearch でマッピングテンプレートが定義されていない場合、OpenSearch は最初に送信されたドキュメントに基づいて動的マッピングを持つフィールドタイプを自動的に決定します。マッピングテンプレートを使用して、OpenSearch で最適なフィールドタイプを明示的に定義することもできます。

次の表に、MySQL データ型と対応する OpenSearch フィールドタイプを示します。明示的なマッピングが定義されていない場合、デフォルトの OpenSearch フィールドタイプ列には OpenSearch の対応するフィールドタイプが表示されます。この場合、OpenSearch は動的マッピングを使用してフィールドタイプを自動的に決定します。推奨 OpenSearch フィールドタイプ列は、マッピングテンプレートで明示的に指定することが推奨される対応するフィールドタイプです。これらのフィールドタイプは MySQL のデータ型とより密接に連携しており、通常、OpenSearch で使用できる検索機能を改善できます。

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
BIGINT	long	long
BIGINT UNSIGNED	long	符号なしロング
BIT	long	ビット数に応じてバイト、ショート、整数、ロング
DECIMAL	テキスト	double またはキーワード

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
DOUBLE	float	double
FLOAT	float	float
INT	long	整数
INT UNSIGNED	long	long
MEDIUMIN	long	整数
MEDIUMIN UNSIGNED	long	整数
NUMERIC	テキスト	double またはキーワード
SMALLINT	long	short
SMALLINT UNSIGNED	long	整数
TINYINT	long	バイト
TINYINT UNSIGNED	long	short
BINARY	テキスト	バイナリ
BLOB	テキスト	バイナリ
CHAR	テキスト	テキスト
ENUM	テキスト	キーワード
LONGBLOE	テキスト	バイナリ
LONGTEXT	テキスト	テキスト

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
MEDIUMBL B	テキスト	バイナリ
MEDIUMTE T	テキスト	テキスト
SET	テキスト	キーワード
TEXT	テキスト	テキスト
TINYBLOB	テキスト	バイナリ
TINYTEXT	テキスト	テキスト
VARBINAR	テキスト	バイナリ
VARCHAR	テキスト	テキスト
DATE	long (エポックミリ秒単位)	date
DATETIME	long (エポックミリ秒単位)	date
TIME	long (エポックミリ秒単位)	date
TIMESTAM	long (エポックミリ秒単位)	date
YEAR	long (エポックミリ秒単位)	date
GEOMETR	text (WKT 形式)	geo_shape
GEOMETR OLLECTIO	text (WKT 形式)	geo_shape
LINESTRIN G	text (WKT 形式)	geo_shape

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
MULTILINESTRING	text (WKT 形式)	geo_shape
MULTIPOINT	text (WKT 形式)	geo_shape
MULTIPOLYGON	text (WKT 形式)	geo_shape
POINT	text (WKT 形式)	geo_point または geo_shape
POLYGON	text (WKT 形式)	geo_shape
JSON	テキスト	オブジェクト

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

制限事項

Aurora MySQL 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- 統合は、パイプラインごとに 1 つの MySQL データベースのみをサポートします。
- 統合は現在、クロスリージョンデータインジェストをサポートしていません。Amazon Aurora クラスターと OpenSearch ドメインは同じ AWS リージョンに存在する必要があります。
- 統合は現在、クロスアカウントデータインジェストをサポートしていません。Amazon Aurora クラスターと OpenSearch Ingestion パイプラインは同じ AWS アカウントに存在する必要があります。

- Amazon Aurora クラスターで、Secrets Manager を使用した認証が有効になっているのを確認します。これが唯一サポートされている認証メカニズムとなります。
- 既存のパイプライン設定を更新して、別のデータベースや別のテーブルからデータを取り込むことはできません。パイプラインのデータベースやテーブル名を更新するには、パイプラインを停止して、更新された設定で再起動するか、新しいパイプラインを作成する必要があります。
- データ定義言語 (DDL) ステートメントは一般的にサポートされていません。以下の場合、データ整合性は維持されません。
  - プライマリキーが変更されます (追加/削除/名前変更)。
  - テーブルは削除/切り捨てられます。
  - 列名またはデータ型が変更されました。
- 同期する MySQL テーブルにプライマリキーが定義されていない場合、データ整合性は保証されません。更新/削除を OpenSearch に同期できるようにするには、OpenSearch シンク設定でカスタム `document_id` オプションを適切に定義する必要があります。
- カスケード削除アクションを使用した外部キー参照はサポートされていないため、Aurora MySQL と OpenSearch の間でデータの不整合が発生する可能性があります。
- サポートされているバージョン: Aurora MySQL バージョン 3.05.2 以降。

推奨される CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント数、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメントの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

メトリクス	説明
pipeline-name .rds.credentialsChanged	このメトリクスは、 AWS シークレットがローテーションされる頻度を示します。
pipeline-name .rds.executorRefreshErrors	このメトリクスは、 AWS シークレットの更新に失敗したことを示します。

メトリクス	説明
pipeline-name .rds.exportRecordsTotal	このメトリクスは、Amazon Aurora からエクスポートされたレコードの数を示します。
pipeline-name .rds.exportRecordsProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたレコードの数を示します。
pipeline-name .rds.exportRecordProcessingErrors	このメトリクスは、Amazon Aurora クラスターからデータを読み取る際の OpenSearch Ingestion パイプラインの処理エラーの数を示します。
pipeline-name .rds.exportRecordsSuccessTotal	このメトリクスは、正常に処理されたエクスポートレコードの合計数を示します。
pipeline-name .rds.exportRecordsFailedTotal	このメトリクスは、処理に失敗したエクスポートレコードの合計数を示します。
pipeline-name .rds.bytesReceived	このメトリクスは、OpenSearch Ingestion パイプラインによって受信された合計バイト数を示します。
pipeline-name .rds.bytesProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたバイトの合計数を示します。
pipeline-name .rds.streamRecordsSuccessTotal	このメトリクスは、ストリームから正常に処理されたレコードの数を示します。

メトリクス	説明
pipeline-name .rds.streamRecordsFailedTotal	このメトリクスは、ストリームから処理に失敗したレコードの合計数を示します。

Aurora PostgreSQL

Amazon Aurora for Aurora PostgreSQL で OpenSearch Ingestion パイプラインを設定するには、次の手順を実行します。

トピック

- Aurora PostgreSQL の前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- データ整合性
- データ型のマッピング
- 制限事項
- 推奨される CloudWatch アラーム

Aurora PostgreSQL の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon Aurora でカスタム DB クラスターパラメータグループを作成して、論理レプリケーションを設定します。

```
rds.logical_replication=1
    aurora.enhanced_logical_replication=1
    aurora.logical_replication_backup=0
    aurora.logical_replication_globaldb=0
```

2. Aurora PostgreSQL DB クラスターを選択または作成し、ステップ 1 で作成したパラメータグループを DB クラスターに関連付けます。

3. Amazon Aurora クラスターで Aurora と AWS Secrets Managerのパスワード管理を使用して、ユーザー名とパスワード認証を設定します。Secrets Manager シークレットを作成するで、ユーザー名とパスワードの組み合わせを作成することもできます。
4. 完全な初期スナップショット機能を使用する場合は、 AWS KMS key と Amazon Aurora から Amazon S3 にデータをエクスポートするための IAM ロールを作成します。

   IAM ロールには、次の許可ポリシーが必要です。

   JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "ExportPolicy",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject*",
                    "s3:ListBucket",
                    "s3:GetObject*",
                    "s3:DeleteObject*",
                    "s3:GetBucketLocation"
                ],
                "Resource": [
                    "arn:aws:s3:::s3-bucket-used-in-pipeline",
                    "arn:aws:s3:::s3-bucket-used-in-pipeline/*"
                ]
            }
        ]
    }
```

   ロールにはまた、次の信頼関係が必要です:

   JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "export.rds.amazonaws.com"
```

```
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }
```

5. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを選択または作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
6. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は Amazon Aurora DB クラスターからドメインまたはコレクションにデータを書き込むことができます。

ステップ 1: パイプラインロールを設定する

Amazon Aurora パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定します。また、Amazon Aurora ソースの次のアクセス許可をロールに追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
    "Sid": "allowReadingFromS3Buckets",
    "Effect": "Allow",
    "Action": [
    "s3:GetObject",
    "s3:DeleteObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:PutObject"
    ],
    "Resource": [
    "arn:aws:s3:::s3_bucket",
    "arn:aws:s3:::s3_bucket/*"
    ]
    },
    {
    "Sid": "allowNetworkInterfacesActions",
    "Effect": "Allow",
```

```
    "Action": [
    "ec2:AttachNetworkInterface",
    "ec2:CreateNetworkInterface",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:DetachNetworkInterface",
    "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": [
    "arn:aws:ec2:*:111122223333:network-interface/*",
    "arn:aws:ec2:*:111122223333:subnet/*",
    "arn:aws:ec2:*:111122223333:security-group/*"
    ]
    },
    {
    "Sid": "allowDescribeEC2",
    "Effect": "Allow",
    "Action": [
    "ec2:Describe*"
    ],
    "Resource": "*"
    },
    {
    "Sid": "allowTagCreation",
    "Effect": "Allow",
    "Action": [
    "ec2:CreateTags"
    ],
    "Resource": "arn:aws:ec2:*:111122223333:network-interface/*",
    "Condition": {
    "StringEquals": {
    "aws:RequestTag/OSISManaged": "true"
    }
    }
    },
    {
    "Sid": "AllowDescribeInstances",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBInstances"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:db:*"
```

```
    ]
    },
    {
    "Sid": "AllowDescribeClusters",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBClusters"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id"
    ]
    },
    {
    "Sid": "AllowSnapshots",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBClusterSnapshots",
    "rds:CreateDBClusterSnapshot",
    "rds:AddTagsToResource"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id",
    "arn:aws:rds:us-east-2:111122223333:cluster-snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowExport",
    "Effect": "Allow",
    "Action": [
    "rds:StartExportTask"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:cluster:DB-id",
    "arn:aws:rds:us-east-2:111122223333:cluster-snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowDescribeExports",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeExportTasks"
    ],
    "Resource": "*",
    "Condition": {
```

```
    "StringEquals": {
    "aws:RequestedRegion": "us-east-2",
    "aws:ResourceAccount": "111122223333"
    }
    }
    },
    {
    "Sid": "AllowAccessToKmsForExport",
    "Effect": "Allow",
    "Action": [
    "kms:Decrypt",
    "kms:Encrypt",
    "kms:DescribeKey",
    "kms:RetireGrant",
    "kms:CreateGrant",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
    ],
    "Resource": [
    "arn:aws:kms:us-east-2:111122223333:key/export-key-id"
    ]
    },
    {
    "Sid": "AllowPassingExportRole",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": [
    "arn:aws:iam::111122223333:role/export-role"
    ]
    },
    {
    "Sid": "SecretsManagerReadAccess",
    "Effect": "Allow",
    "Action": [
    "secretsmanager:GetSecretValue"
    ],
    "Resource": [
    "arn:aws:secretsmanager:*:111122223333:secret:*"
    ]
    }
    ]
    }
```

ステップ 2: パイプラインを作成する

ソースとして Aurora PostgreSQL クラスター を指定する OpenSearch Ingestion パイプラインを次のように設定します。

```
version: "2"
aurora-postgres-pipeline:
  source:
    rds:
      db_identifier: "cluster-id"
      engine: aurora-postgresql
      database: "database-name"
      tables:
        include:
          - "schema1.table1"
          - "schema2.table2"
      s3_bucket: "bucket-name"
      s3_region: "bucket-region"
      s3_prefix: "prefix-name"
      export:
        kms_key_id: "kms-key-id"
        iam_role_arn: "export-role-arn"
      stream: true
      aws:
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        region: "us-east-1"
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
        index: "${getMetadata(\"table_name\")}"
        index_type: custom
        document_id: "${getMetadata(\"primary_key\")}"
        action: "${getMetadata(\"opensearch_action\")}"
        document_version: "${getMetadata(\"document_version\")}"
        document_version_type: "external"
        aws:
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
          region: "us-east-1"
extension:
  aws:
    secrets:
```

```
    secret:
      secret_id: "rds-secret-id"
      region: "us-east-1"
      sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
      refresh_interval: PT1H
```

> **Note**
>
> 事前設定された Amazon Aurora ブループリントを使用して、このパイプラインを作成できます。詳細については、「ブループリントの使用」を参照してください。

Amazon Aurora をソースとして使用するには、パイプラインの VPC アクセスを設定する必要があります。選択する VPC は、Amazon Aurora ソースが使用するのと同じ VPC である必要があります。次に、1 つ以上のサブネットと 1 つ以上の VPC セキュリティグループを選択します。パイプラインには Aurora MySQL データベースへのネットワークアクセスが必要であるため、パイプラインの VPC セキュリティグループからデータベースポートへのインバウンドトラフィックを許可する VPC セキュリティグループが Aurora クラスターに設定されていることも確認する必要があります。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

を使用してパイプライン AWS マネジメントコンソール を作成する場合は、ソースとして Amazon Aurora を使用するには、パイプラインを VPC にアタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] を選択し、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自の CIDR を選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。

カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion と Amazon Aurora 間の IP アドレスの衝突を回避するには、Amazon Aurora VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認します。

詳細については、「Configuring VPC access for a pipeline」を参照してください。

データ整合性

パイプラインは、Amazon Aurora クラスターから変更を継続的にポーリングまたは受信し、OpenSearch インデックス内の対応するドキュメントを更新することで、データ整合性を確保します。

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了としてマークします。OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込む場合は、パイプラインがドキュメント ID を生成しないため、パイプラインシンク設定`document_id: "${getMetadata(\"primary_key\")}"`で を省略する必要があることに注意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、Amazon Aurora 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更と整合されます。

データ型のマッピング

OpenSearch Ingestion パイプラインは、Aurora PostgreSQL データ型を、消費する OpenSearch Service ドメインまたはコレクションに適した表現にマッピングします。OpenSearch でマッピングテンプレートが定義されていない場合、OpenSearch は最初に送信されたドキュメントに基づいて動的マッピングを使用してフィールドタイプを自動的に決定します。マッピングテンプレートを使用して、OpenSearch で最適なフィールドタイプを明示的に定義することもできます。

次の表に、Aurora PostgreSQL データ型と対応する OpenSearch フィールド型を示します。明示的なマッピングが定義されていない場合、デフォルトの OpenSearch フィールドタイプ列には OpenSearch の対応するフィールドタイプが表示されます。この場合、OpenSearch は動的マッピングを使用してフィールドタイプを自動的に決定します。推奨 OpenSearch フィールドタイプ列は、マッピングテンプレートで明示的に指定する、対応する推奨フィールドタイプです。これらのフィールドタイプは、Aurora PostgreSQL のデータ型とより密接に一致しており、通常、OpenSearch で使用できる検索機能を改善できます。

Aurora PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
smallint	long	short
整数	long	整数

Aurora PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
bigint	long	long
decimal	テキスト	double またはキーワード
numeric[(p, s)]	テキスト	double またはキーワード
real	float	float
double precision	float	double
smallserial	long	short
シリアル	long	整数
bigserial	long	long
money	オブジェクト	オブジェクト
character varying(n)	テキスト	テキスト
varchar(n)	テキスト	テキスト
character (n)	テキスト	テキスト
char(n)	テキスト	テキスト
bpchar(n)	テキスト	テキスト
bpchar	テキスト	テキスト
テキスト	テキスト	テキスト

Aurora PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
enum	テキスト	テキスト
bytea	テキスト	バイナリ
timestamp [(p)] [without time zone]	long (エポックミリ秒単位)	date
timestamp [(p)] with time zone	long (エポックミリ秒単位)	date
date	long (エポックミリ秒単位)	date
time [(p)] [without time zone]	long (エポックミリ秒単位)	date
time [(p)] with time zone	long (エポックミリ秒単位)	date
interval [fields] [(p)]	text (ISO8601 形式)	テキスト
boolean	boolean	boolean
point	text (WKT 形式)	geo_shape
LINE	text (WKT 形式)	geo_shape

Aurora PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
lseg	text (WKT 形式)	geo_shape
box	text (WKT 形式)	geo_shape
パス	text (WKT 形式)	geo_shape
Polygon (ポリゴン)	text (WKT 形式)	geo_shape
circle	オブジェクト	オブジェクト
cidr	テキスト	テキスト
inet	テキスト	テキスト
macaddr	テキスト	テキスト
macaddr8	テキスト	テキスト
bit(n)	long	バイト、ショート、整数、ロング (ビット数によって異なります)
bit varying(n)	long	バイト、ショート、整数、ロング (ビット数によって異なります)
json	オブジェクト	オブジェクト
JSONB	オブジェクト	オブジェクト
jsonpath	テキスト	テキスト

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

制限事項

Aurora PostgreSQL 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- 統合は、パイプラインごとに 1 つの Aurora PostgreSQL データベースのみをサポートします。
- 統合は現在、クロスリージョンデータインジェストをサポートしていません。Amazon Aurora クラスターと OpenSearch ドメインは同じ AWS リージョンに存在する必要があります。
- 統合は現在、クロスアカウントデータインジェストをサポートしていません。Amazon Aurora クラスターと OpenSearch Ingestion パイプラインは同じ AWS アカウントに存在する必要があります。
- Amazon Aurora クラスターでサポートされている唯一の認証メカニズムである AWS Secrets Managerを使用して認証が有効になっていることを確認します。
- 既存のパイプライン設定を更新して、別のデータベースや別のテーブルからデータを取り込むことはできません。パイプラインのデータベースやテーブル名を更新するには、パイプラインを停止して、更新された設定で再起動するか、新しいパイプラインを作成する必要があります。
- データ定義言語 (DDL) ステートメントは一般的にサポートされていません。以下の場合、データ整合性は維持されません。
  - プライマリキーが変更されます (追加/削除/名前変更)。
  - テーブルは削除/切り捨てられます。
  - 列名またはデータ型が変更されました。
- 同期する Aurora PostgreSQL テーブルにプライマリキーが定義されていない場合、データ整合性は保証されません。更新/削除を OpenSearch に同期できるようにするには、OpenSearch とシンク設定でカスタム `document_id` オプションを適切に定義する必要があります。
- サポートされているバージョン: Aurora PostgreSQL バージョン 16.4 以降。

推奨される CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント数、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメン

トの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

メトリクス	説明
pipeline-name .rds.credentialsChanged	このメトリクスは、 AWS シークレットがローテーションされる頻度を示します。
pipeline-name .rds.executorRefreshErrors	このメトリクスは、 AWS シークレットの更新に失敗したことを示します。
pipeline-name .rds.exportRecordsTotal	このメトリクスは、Amazon Aurora からエクスポートされたレコードの数を示します。
pipeline-name .rds.exportRecordsProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたレコードの数を示します。
pipeline-name .rds.exportRecordProcessingErrors	このメトリクスは、Amazon Aurora クラスターからデータを読み取る際の OpenSearch Ingestion パイプラインの処理エラーの数を示します。
pipeline-name .rds.exportRecordsSuccessTotal	このメトリクスは、正常に処理されたエクスポートレコードの合計数を示します。
pipeline-name .rds.exportRecordsFailedTotal	このメトリクスは、処理に失敗したエクスポートレコードの合計数を示します。

メトリクス	説明
pipeline-name .rds.bytesReceived	このメトリクスは、OpenSearch Ingestion パイプラインによって受信された合計バイト数を示します。
pipeline-name .rds.bytesProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたバイトの合計数を示します。
pipeline-name .rds.streamRecordsSuccessTotal	このメトリクスは、ストリームから正常に処理されたレコードの数を示します。
pipeline-name .rds.streamRecordsFailedTotal	このメトリクスは、ストリームから処理に失敗したレコードの合計数を示します。

複数のテーブルからのデータを結合して 1 つのドキュメントに取り込む

RDS ソースプラグイン`joins`の設定により、正規化されたリレーショナルテーブルを単一の OpenSearch ドキュメントに自動的に非正規化できます。設定すると、パイプラインは複数の関連テーブルから変更データキャプチャ (CDC) イベントを読み取り、Painless スクリプトベースのアップサートを使用して親ドキュメントにマージします。

このトピックでは、結合タイプ、前提条件、設定構文、バージョン追跡、一般的な問題のトラブルシューティングなど、Aurora および Amazon RDS 取り込みパイプラインでテーブル結合を設定する方法について説明します。

前提条件

- プライマリキー、外部キー、テーブル関係など、リレーショナルデータベースの概念に関する基本的な理解。
- 外部キー関係 (親子) を持つテーブル。
- のすべてのテーブルは、 に`relations`リスト`tables.include`されている必要があります。
- 親テーブルにはプライマリキーが必要です。

- 子テーブルには、親のプライマリキーを参照する列が必要です。
- 子テーブルには、アップサート内の個々のレコードを識別するための独自のプライマリキー (child_primary_key) が必要です。

サポートされている結合パターン

- 親 → 子 (1:1)
- 親 → 子 (1:N)
- 親ごとに複数の子

設定例

次の YAML 設定は、Aurora ソースプラグインの親子テーブル関係を定義する方法を示しています。この例では、異なる結合タイプを使用して 2 つの子テーブルを持つ親テーブルを設定します。

```
version: "2"
aurora-joins-pipeline:
  source:
    rds:
      db_identifier: "my-aurora-cluster"
      engine: "aurora-mysql"
      database: "my_database"
      tables:
        include:
          - "parent_table"
          - "child_table_1"
          - "child_table_2"
      s3_bucket: "my-pipeline-bucket"
      s3_region: "us-east-1"
      s3_prefix: "rds-export"
      export:
        kms_key_id: "my-kms-key-id"
        iam_role_arn: "arn:aws:iam::123456789012:role/my-export-role"
      stream: true
      aws:
        sts_role_arn: "arn:aws:iam::123456789012:role/my-pipeline-role"
        region: "us-east-1"
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
```

```
      joins:
        version_field: "__versions"
        relations:
          - parent: "parent_table"
            child: "child_table_1"
            parent_key: "id"
            child_key: "parent_id"
            child_primary_key: "child_id"
            join_type: "one_to_many"
            max_child_records: 100
          - parent: "parent_table"
            child: "child_table_2"
            parent_key: "id"
            child_key: "parent_id"
            child_primary_key: "child_id"
            join_type: "one_to_one"
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
        index: "my-joined-index"
        document_id: "${getMetadata(\"primary_key\")}"
        action: "${getMetadata(\"opensearch_action\")}"
        aws:
          sts_role_arn: "arn:aws:iam::123456789012:role/my-pipeline-role"
          region: "us-east-1"
extension:
  aws:
    secrets:
      secret:
        secret_id: "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-db-secret"
        region: "us-east-1"
        sts_role_arn: "arn:aws:iam::123456789012:role/my-pipeline-role"
        refresh_interval: PT1H
```

結合の種類

one_to_one

子フィールドは親ドキュメントのルートレベルでフラット化されます。各親に関連する子レコードが 1 つだけある場合に使用します。

```
Parent table: orders (order_id, customer_name, total)
Child table:  shipping (shipping_id, order_id, tracking_number, carrier)
```

```
Result document:
{
  "order_id": 1,
  "customer_name": "Alice",
  "total": 299.99,
  "shipping_id": "1",
  "tracking_number": "TRK-1",
  "carrier": "FedEx"
}
```

one_to_many

子レコードは、ネストされた配列として親ドキュメントに保存されます。各親が複数の関連する子レコードを持つことができる場合に使用します。

```
Parent table: orders (order_id, customer_name, total)
Child table:  order_items (item_id, order_id, product_name, quantity, price)

Result document:
{
  "order_id": 1,
  "customer_name": "Alice",
  "total": 299.99,
  "order_items": [
    {"item_id": 10, "product_name": "Keyboard", "quantity": 1, "price": 149.99},
    {"item_id": 11, "product_name": "Mouse", "quantity": 1, "price": 29.99}
  ]
}
```

ドキュメント構造

ドキュメント ID

OpenSearch ドキュメント_idは、親テーブルのプライマリキーの値に設定されます。同じ親のすべての子レコードは、この 1 つのドキュメントにマージされます。

バージョン追跡

version_field (デフォルト: __versions) は、テーブルごとのバージョンカウンターのマップを保存します。

```
"__versions": {
  "orders": 12345678901,
  "order_items": 12345678902,
  "shipping": 12345678903
}
```

各テーブルのバージョンは、エクスポートタイムスタンプまたはバイナリログタイムスタンプから算出されます。エクスポートまたは CDC イベントが到着すると、Painless スクリプトは、受信バージョンがその特定のテーブルのストアドバージョンよりも新しいかどうかを確認します。古い場合、イベントはスキップされます。これにより、以下が保証されます。

- べき等処理 — 再生されたイベントは安全に無視されます。
- 独立したバージョニング — 注文の更新はアイテムの更新を妨げません。
- 同時安全性 — 同じ親の複数の子挿入が正しく処理されます。

ドキュメント _version

OpenSearch _versionフィールドは、アップサートが成功するたびに増加します。1 つの親 + N 項目 + 1 つの配送レコードを持つドキュメントの場合、 _version = N + 2。

制限事項

- 単一レベルの結合のみ — 親 → 子。マルチレベル (親 → 子 → 孫) はサポートされていません。
- パイプラインごとに 1 つの親テーブル。すべての子テーブルは同じ親に結合されます。
- 子テーブルは、同じパイプライン内の複数の親に属すことはできません。
- max_child_records はone_to_many結合の配列サイズを制限します。この制限を超えるレコードは削除されます。

> **フィールド名の競合**
>
> one_to_one 結合の場合、子フィールドはルートレベルでフラット化されます。子テーブルに親列と同じ名前の列がある場合、子値は親値を上書きします。親テーブルとone_to_one子テーブルで個別の列名を使用します。
> one_to_many 結合の場合、子フィールドは子テーブルにちなんで という名前の配列内にネストされるため、競合は問題ではありません。

トラブルシューティング

欠落している子レコードのドキュメント

- ドキュメント__versionsを確認します。子テーブルのバージョンがない場合、子イベントはまだ処理されていません。
- パイプラインがアクティブで、changeEventsProcessedメトリクスがゼロ以外であることを確認します。

OpenSearch に表示されないドキュメント

- 親レコードがソーステーブルに存在するかどうかを確認します。
- tables.include リストに必要なすべてのテーブルが含まれていることを確認します。
- パイプラインログで取り込みの失敗と設定の問題を確認します: /aws/vendedlogs/*pipeline-log-group*。

OpenSearch Ingestion パイプラインを Amazon DynamoDB で使用する

DynamoDB プラグインを使用して、作成、更新、削除などのテーブルイベントを Amazon OpenSearch Service ドメインと Amazon OpenSearch Serverless コレクションにストリーミングできます。パイプラインは、変更データキャプチャ (CDC) を使用して、大規模で低レイテンシーのストリーミングを行います。

完全な初期スナップショットの有無にかかわらず、DynamoDB データを処理できます。

- 完全なスナップショットの場合 – DynamoDB はポイントインタイムリカバリ (PITR) を使用してバックアップを作成し、Amazon S3 にアップロードします。OpenSearch Ingestion は、スナップショットを 1 つ以上の OpenSearch インデックスにインデックス化します。一貫性を維持するために、パイプラインはすべての DynamoDB の変更を OpenSearch と同期します。このオプションでは、PITR ストリームと DynamoDB ストリームの両方を有効にする必要があります。
- スナップショットなし – OpenSearch Ingestion は新しい DynamoDB イベントのみをストリーミングします。スナップショットがすでにある場合、または履歴データなしでリアルタイムストリーミングが必要な場合は、このオプションを選択します。このオプションでは、DynamoDB Streams のみを有効にする必要があります。

詳細については、「Amazon DynamoDB デベロッパーガイド」の「DynamoDB と Amazon OpenSearch Service のゼロ ETL 統合」を参照してください。

トピック

- 前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- データ整合性
- データ型のマッピング
- Auto Scaling
- 制限事項
- DynamoDB の推奨 CloudWatch アラーム

前提条件

パイプラインを設定するには、DynamoDB Streams が有効になっている DynamoDB テーブルが必要です。ストリームでは NEW_IMAGE ストリームビュータイプを使用する必要があります。ただし、このストリームビュータイプがユースケースに合っている場合、OpenSearch Ingestion パイプラインは NEW_AND_OLD_IMAGES を使用してイベントをストリーミングすることもできます。

また、スナップショットを使用している場合は、テーブルでのポイントインタイムリカバリを有効にする必要があります。詳細については、「Amazon DynamoDB デベロッパーガイド」の「テーブルの作成」、「ポイントインタイムリカバリの有効化」、「ストリームの有効化」を参照してください。

ステップ 1: パイプラインロールを設定する

DynamoDB テーブルを設定したら、パイプライン設定で使用するパイプラインロールを設定し、そのロールに次の DynamoDB 許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowRunExportJob",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:DescribeContinuousBackups",
                "dynamodb:ExportTableToPointInTime"
```

```
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/my-table"
            ]
        },
        {
            "Sid": "allowCheckExportjob",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeExport"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/my-table/export/*"
            ]
        },
        {
            "Sid": "allowReadFromStream",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeStream",
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/my-table/stream/*"
            ]
        },
        {
            "Sid": "allowReadAndWriteToS3ForExport",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/export-folder/*"
            ]
        }
    ]
}
```

AWS KMS カスタマーマネージドキーを使用して、エクスポートデータファイルを暗号化することもできます。エクスポートされたオブジェクトを復号するには、パイプラインのエクスポート設定において、次の形式でキー ID として s3_sse_kms_key_id を指定します: arn:aws:kms:*region*:*account-id*:key/*my-key-id*。次のポリシーには、カスタマーマネージドキーを使用するために必要なアクセス許可が含まれています。

```
{
    "Sid": "allowUseOfCustomManagedKey",
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": arn:aws:kms:region:account-id:key/my-key-id
}
```

ステップ 2: パイプラインを作成する

そして、ソースとして DynamoDB を指定する OpenSearch Ingestion パイプラインを次のように設定できます。このサンプルパイプラインは、PITR スナップショットを使用して table-a からデータを取り込み、続いて DynamoDB Streams からイベントを取り込みます。開始位置が LATEST であることは、パイプラインが DynamoDB Streams から最新のデータを読み取る必要があることを示します。

```
version: "2"
cdc-pipeline:
  source:
    dynamodb:
      tables:
      - table_arn: "arn:aws:dynamodb:region:account-id:table/table-a"
        export:
          s3_bucket: "my-bucket"
          s3_prefix: "export/"
        stream:
          start_position: "LATEST"
      aws:
        region: "us-east-1"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.region.es.amazonaws.com"]
      index: "${getMetadata(\"table-name\")}"
```

```
      index_type: custom
      normalize_index: true
      document_id: "${getMetadata(\"primary_key\")}"
      action: "${getMetadata(\"opensearch_action\")}"
      document_version: "${getMetadata(\"document_version\")}"
      document_version_type: "external"
```

事前設定された DynamoDB ブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

データ整合性

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了としてマークします。

OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込みたい場合は、パイプラインがドキュメント ID を生成しないことに留意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、DynamoDB 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更とリコンサイルされます。

データ型のマッピング

OpenSearch Service は、各着信ドキュメントのデータ型を、DynamoDB の対応するデータ型に動的にマッピングします。次の表は、OpenSearch Service がさまざまなデータ型を自動的にマッピングする方法を示しています。

データ型	OpenSearch	DynamoDB
Number	OpenSearch は、数値データを自動的にマッピングします。数値が整数である場合、OpenSearch はそれをlong 値としてマッピングします。数値が小数である場	DynamoDB は数値をサポートしています。

データ型	OpenSearch	DynamoDB
	合、OpenSearch はそれを float 値としてマッピングします。 OpenSearch は、最初に送信されたドキュメントに基づいてさまざまな属性を動的にマッピングします。DynamoDB の同じ属性についてデータ型が混在している場合 (整数と小数の両方など)、マッピングは失敗する可能性があります。 例えば、最初のドキュメントがある属性を整数として持っており、後のドキュメントがその同じ属性を小数として持っている場合、OpenSearch は 2 つ目のドキュメントの取り込みに失敗します。このような場合は、次のような明示的なマッピングテンプレートを提供する必要があります: `{ "template": { "mappings": { "properties": { "MixedNumberAttribute": { "type": "float" } } } } }` 倍精度が必要な場合は、文字列型のフィールドマッピングを使用します。OpenSearch には、38 桁の精度をサポートする同等の数値型はありません。	

データ型	OpenSearch	DynamoDB
数値セット	OpenSearch は、数値セットを、long 値または浮動小数点値の配列に自動的にマッピングします。スカラー数値の場合と同様、これは、取り込まれた最初の数値が整数または小数のいずれであるかによって異なります。スカラー文字列をマッピングするのと同じ方法で、数値セットのマッピングを提供できます。	DynamoDB は、数値のセットを表す型をサポートしています。
String	OpenSearch は、文字列の値をテキストとして自動的にマッピングします。状況によっては (列挙型の値など)、キーワード型にマッピングできます。 次の例は、PartType という名前の DynamoDB 属性を、OpenSearch キーワードにマッピングする方法を示しています。 `{ "template": { "mappings": { "properties": { "PartType": { "type": "keyword" } } } } }`	DynamoDB は文字列をサポートします。
文字列セット	OpenSearch は、文字列セットを、文字列の配列に自動的にマッピングします。スカラー文字列をマッピングするのと同じ方法で、文字列セットのマッピングを提供できます。	DynamoDB は、文字列のセットを表す型をサポートしています。

データ型	OpenSearch	DynamoDB
バイナリ	OpenSearch は、バイナリデータをテキストとして自動的にマップします。これらを OpenSearch のバイナリフィールドとして書き込むためのマッピングを提供できます。 次の例は、`ImageData` という名前の DynamoDB 属性を、OpenSearch バイナリフィールドにマッピングする方法を示しています。 `{ "template": { "mappings": { "properties": { "ImageData": { "type": "binary" } } } } }`	DynamoDB はバイナリ型の属性をサポートしています。
バイナリセット	OpenSearch は、バイナリセットをテキストとして、バイナリデータの配列に自動的にマッピングします。スカラーバイナリをマッピングするのと同じ方法で、数値セットのマッピングを提供できます。	DynamoDB は、バイナリ値のセットを表す型をサポートしています。
ブール値	OpenSearch は、DynamoDB のブール型を、OpenSearch のブール型にマッピングします。	DynamoDB は、ブール型属性をサポートします。

データ型	OpenSearch	DynamoDB
Null	OpenSearch は、DynamoDB の null 型のドキュメントを取り込むことができます。値は null 値としてドキュメントに保存されます。この型にはマッピングがなく、このフィールドにはインデックスが付けられず、検索もできません。 同じ属性名が null 型に使用され、後で文字列などの別の型に変更される場合、OpenSearch は、最初の null 以外の値について動的マッピングを作成します。後続の値は引き続き DynamoDB の null 値にすることができます。	DynamoDB は、null 型属性をサポートします。

データ型	OpenSearch	DynamoDB
マッピング	OpenSearch は、DynamoDB のマップ属性を、ネストされたフィールドにマッピングします。同じマッピングがネストされたフィールド内でも適用されます。 次の例では、ネストされたフィールドの文字列を、OpenSearch のキーワード型にマッピングします: `{ "template": { "mappings": { "properties": { "AdditionalDescriptions": { "properties": { "PartType": { "type": "keyword" } } } } } } }`	DynamoDB は、マッピング型属性をサポートします。

データ型	OpenSearch	DynamoDB
リスト	OpenSearch は、DynamoDB リストの内容に応じて、そのリストについてのさまざまな結果を提供します。 リストに同じタイプのスカラー型がすべて含まれている場合 (例えば、すべての文字列のリスト)、OpenSearch は、そのリストをその型の配列として取り込みます。これは、文字列、数値、ブール型、および null 型について機能します。これらの各型についての制限は、その型のスカラーについての制限と同じです。 また、マップに使用するのと同じマッピングを使用して、マップのリストのマッピングを提供することもできます。 混合型のリストを提供することはできません。	DynamoDB は、リスト型属性をサポートします。

データ型	OpenSearch	DynamoDB
設定	OpenSearch は、DynamoDB セットの内容に応じて、そのセットについてのさまざまな結果を提供します。 セットに同じタイプのスカラー型がすべて含まれている場合 (例えば、すべての文字列のセット)、OpenSearch は、そのセットをその型の配列として取り込みます。これは、文字列、数値、ブール型、および null 型について機能します。これらの各型についての制限は、その型のスカラーについての制限と同じです。 また、マップに使用するのと同じマッピングを使用して、マップのセットのマッピングを提供することもできます。 混合型のセットを提供することはできません。	DynamoDB は、セットを表す型をサポートします。

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

Auto Scaling

OpenSearch Ingestion の各 OCU は、最大 150 個の DynamoDB ストリームシャードを並行して処理できます。高レイテンシーとデータ損失を防ぐには、最小 OCUs を少なくともオープンシャードの最大数を 150 で割った値に設定し、最も近い整数に切り上げます。OpenSearch Ingestion は、すべてのシャードを並行して処理するために必要な数の OCUs に自動的にスケーリングします。

DynamoDB ストリームのシャード数を確認するには、パイプラインの CloudWatch `total0penShards.max`メトリクスで 15 分間の最大統計を含むメトリクスを表示します。

制限事項

DynamoDB 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- OpenSearch Ingestion と DynamoDB の統合は現在、クロスリージョンの取り込みをサポートしていません。DynamoDB テーブルと OpenSearch Ingestion パイプラインは、同じ AWS リージョンに存在する必要があります。
- DynamoDB テーブルと OpenSearch Ingestion パイプラインは、同じ AWS アカウントに存在する必要があります。
- OpenSearch Ingestion パイプラインは、ソースとして 1 つの DynamoDB テーブルのみをサポートします。
- DynamoDB Streams は、最大 24 時間ログにデータを格納します。大きなテーブルの最初のスナップショットからの取り込みに 24 時間以上かかる場合、初期データの一部が失われます。このデータ損失を軽減するには、テーブルのサイズを見積もり、OpenSearch Ingestion パイプラインの適切なコンピューティングユニットを設定します。

DynamoDB の推奨 CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント量、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメントの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

メトリクス	説明
`dynamodb-pipeline.BlockingBuffer.bufferUsage.value`	使用されているバッファの量を示します。
`dynamodb-pipeline.dynamodb.activeExportS3ObjectConsumers.value`	エクスポートのために Amazon S3 オブジェクトをアクティブに処理している OCU の合計数を表示します。

メトリクス	説明
dynamodb-pipeline.dynamodb.bytesProcessed.count	DynamoDB ソースから処理されたバイト数。
dynamodb-pipeline.dynamodb.changeEventsProcessed.count	DynamoDB ストリームから処理された変更イベントの数。
dynamodb-pipeline.dynamodb.changeEventsProcessingErrors.count	DynamoDB から処理された変更イベントからのエラーの数。
dynamodb-pipeline.dynamodb.exportJobFailure.count	Number of export job submission attempts that have failed.
dynamodb-pipeline.dynamodb.exportJobSuccess.count	Number of export jobs that have been submitted successfully.
dynamodb-pipeline.dynamodb.exportRecordsProcessed.count	エクスポートから処理されたレコードの合計数。
dynamodb-pipeline.dynamodb.exportRecordsTotal.count	DynamoDB からエクスポートされたレコードの合計数。データのエクスポートボリュームの追跡に不可欠です。
dynamodb-pipeline.dynamodb.exportS3ObjectsProcessed.count	Total number of export data files that have been processed successfully from Amazon S3.
dynamodb-pipeline.opensearch.bulkBadRequestErrors.count	Count of errors during bulk requests due to malformed request.
dynamodb-pipeline.opensearch.bulkRequestLatency.avg	Average latency for bulk write requests made to OpenSearch.
dynamodb-pipeline.opensearch.bulkRequestNotFoundErrors.count	Number of bulk requests that failed because the target data could not be found.

メトリクス	説明
dynamodb-pipeline.opensearch.bulkRequestNumberOfRetries.count	Number of retries by OpenSearch Ingestion pipelines to write OpenSearch cluster.
dynamodb-pipeline.opensearch.bulkRequestSizeBytes.sum	Total size in bytes of all bulk requests made to OpenSearch.
dynamodb-pipeline.opensearch.documentErrors.count	Number of errors when sending documents to OpenSearch. The documents causing the errors witll be sent to DLQ.
dynamodb-pipeline.opensearch.documentsSuccess.count	Number of documents successfully written to an OpenSearch cluster or collection.
dynamodb-pipeline.opensearch.documentsSuccessFirstAttempt.count	Number of documents successfully indexed in OpenSearch on the first attempt.
dynamodb-pipeline.opensearch.documentsVersionConflictErrors.count	Count of errors due to version conflicts in documents during processing.
dynamodb-pipeline.opensearch.PipelineLatency.avg	Average latency of OpenSearch Ingestion pipeline to process the data by reading from the source to writint to the destination.
dynamodb-pipeline.opensearch.PipelineLatency.max	Maximum latency of OpenSearch Ingestion pipeline to process the data by reading from the source to writing the destination.
dynamodb-pipeline.opensearch.recordsIn.count	Count of records successfully ingested into OpenSearch. This metric is essential for tracking the volume of data being processed and stored.
dynamodb-pipeline.opensearch.s3.dlqS3RecordsFailed.count	Number of records that failed to write to DLQ.

メトリクス	説明
dynamodb-pipeline.opensearch.s3.dlqS3RecordsSuccess.count	Number of records that are written to DLQ.
dynamodb-pipeline.opensearch.s3.dlqS3RequestLatency.count	Count of latency measurements for requests to the Amazon S3 dead-letter queue.
dynamodb-pipeline.opensearch.s3.dlqS3RequestLatency.sum	Total latency for all requests to the Amazon S3 dead-letter queue
dynamodb-pipeline.opensearch.s3.dlqS3RequestSizeBytes.sum	Total size in bytes of all requests made to the Amazon S3 dead-letter queue.
dynamodb-pipeline.recordsProcessed.count	Total number of records processed in the pipeline, a key metric for overal throughput.
dynamodb.changeEventsProcessed.count	No records are being gathered from DynamoDB streams. This could be due to no activitiy on the table, an export being in progress, or an issue accessing the DynamoDB streams.
dynamodb.exportJobFailure.count	The attempt to trigger an export to S3 failed.
dynamodb-pipeline.opensearch.bulkRequestInvalidInputErrors.count	Count of bulk request errors in OpenSearch due to invalid input, crucial for monitoring data quality and operational issues.
opensearch.EndToEndLatency.avg	The end to end latnecy is higher than desired for reading from DynamoDB streams. This could be due to an underscaled OpenSearch cluster or a maximum pipeline OCU capacity that is too low for the WCU throughput on the DynamoDB table. This end to end latency will be high after an export and should decrease over time as it catches up to the latest DynamoDB streams.

Amazon DocumentDB での OpenSearch Ingestion パイプラインの使用

DocumentDB プラグインを使用して、作成、更新、削除などのドキュメントの変更を Amazon OpenSearch Service にストリーミングできます。パイプラインは、可能な場合は変更データキャプチャ (CDC)、または大規模で低レイテンシーのストリーミングのための API ポーリングをサポートします。

完全な初期スナップショットの有無にかかわらず、データを処理できます。完全なスナップショットは、Amazon DocumentDB コレクション全体をキャプチャし、Amazon S3 にアップロードします。次に、パイプラインは 1 つ以上の OpenSearch インデックスにデータを送信します。スナップショットを取り込むと、パイプラインは継続的な変更を同期して一貫性を維持し、最終的にほぼリアルタイムの更新に追いつきます。

別のソースから完全なスナップショットをすでに取得している場合、または新しいイベントのみを処理する必要がある場合は、スナップショットなしでストリーミングできます。この場合、パイプラインは最初の一括ロードなしで Amazon DocumentDB 変更ストリームから直接読み取ります。

ストリームを有効にする場合は、Amazon DocumentDB コレクションで変更ストリームを有効にする必要があります。ただし、全ロードまたはエクスポートのみを実行する場合は、変更ストリームは必要ありません。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon DocumentDB 開発者ガイドの「Amazon DocumentDB クラスターを作成する」の手順に従って、データを読み取るアクセス許可を持つ Amazon DocumentDB クラスターを作成します。CDC インフラストラクチャを使用する場合は、変更ストリームを発行するように Amazon DocumentDB クラスターを設定します。
2. Amazon DocumentDB クラスターで TLS を有効にします。
3. OpenSearch Ingestion で使用するプライベートアドレススペースの VPC CIDR を設定します。
4. を使用して Amazon DocumentDB クラスターで認証を設定します AWS Secrets Manager。「Automatically rotating passwords for Amazon DocumentDB」の手順に従って、シークレットローテーションを有効にします。詳細については、「Database Access Using Role-Based Access Control」と「Security in Amazon DocumentDB」を参照してください。
5. 変更ストリームを使用して Amazon DocumentDB コレクションのデータ変更をサブスクライブする場合は、`change_stream_log_retention_duration` パラメータを使用して保持期間を最

大 7 日間延長することで、データ損失を回避します。変更ストリームイベントは、イベントが記録されてからデフォルトで 3 時間保存されます。これは、大規模なコレクションには十分な時間ではありません。変更ストリーム保持期間を変更するには、「Modifying the Change Stream Log Retention Duration」を参照してください。

6. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
7. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は Amazon DocumentDB クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
}
```

   コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

Amazon DocumentDB パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、そのロールに次の Amazon DocumentDB 許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowS3ListObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::s3-bucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": "s3-prefix/*"
                }
            }
        },
        {
            "Sid": "allowReadAndWriteToS3ForExportStream",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::s3-bucket/s3-prefix/*"
            ]
        },
        {
            "Sid": "SecretsManagerReadAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
```

```
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret-
name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:network-interface/*",
                "arn:aws:ec2:*:111122223333:subnet/*",
                "arn:aws:ec2:*:111122223333:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/OSISManaged": "true"
                }
```

```
            }
        }
    ]
}
```

OpenSearch Ingestion パイプラインの作成に使用する IAM ロールには、上記の Amazon EC2 アクセス許可を指定する必要があります。これは、パイプラインがこれらのアクセス許可を使用して VPC 内のネットワークインターフェイスを作成および削除するためです。パイプラインは、このネットワークインターフェイスを介してのみ Amazon DocumentDB クラスターにアクセスできます。

ステップ 2: パイプラインを作成する

そして、ソースとして Amazon DocumentDB を指定する OpenSearch Ingestion パイプラインを次のように設定できます。インデックス名を入力するには、getMetadata 関数がメタデータキーとして *documentdb_collection* を使用することに注意してください。getMetadata メソッドなしで別のインデックス名を使用する場合は、設定 index: "*my_index_name*" を使用できます。

```
version: "2"
documentdb-pipeline:
  source:
    documentdb:
      acknowledgments: true
      host: "https://docdb-cluster-id.us-east-1.docdb.amazonaws.com"
      port: 27017
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
      aws:
      s3_bucket: "bucket-name"
      s3_region: "bucket-region"
      s3_prefix: "path" #optional path for storing the temporary data
      collections:
        - collection: "dbname.collection"
          export: true
          stream: true
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      index: "${getMetadata(\"documentdb_collection\")}"
      index_type: custom
      document_id: "${getMetadata(\"primary_key\")}"
```

```
      action: "${getMetadata(\"opensearch_action\")}"
      document_version: "${getMetadata(\"document_version\")}"
      document_version_type: "external"
extension:
  aws:
    secrets:
      secret:
        secret_id: "my-docdb-secret"
        region: "us-east-1"
        refresh_interval: PT1H
```

事前設定された Amazon DocumentDB ブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

を使用してパイプライン AWS マネジメントコンソール を作成する場合は、ソースとして Amazon DocumentDB を使用するには、パイプラインを VPC にアタッチする必要があります。これを行うには、[ソースネットワークオプション] セクションを見つけ、[VPC にアタッチ] チェックボックスをオンにして、提供されたデフォルトオプションのいずれかから CIDR を選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。

カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion と Amazon DocumentDB 間の IP アドレスの衝突を回避するには、Amazon DocumentDB VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

詳細については、「Configuring VPC access for a pipeline」を参照してください。

データ整合性

パイプラインは、Amazon DocumentDB クラスターから変更を継続的にポーリングまたは受信し、OpenSearch インデックス内の対応するドキュメントを更新することで、データ整合性を確保します。

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了としてマークします。

OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込みたい場合は、パ

イプラインがドキュメント ID を生成しないため、パイプラインシンク設定の `document_id: "${getMetadata(\"primary_key\")}"` を省略する必要があることに留意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、Amazon DocumentDB 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更とリコンサイルされます。

データ型のマッピング

OpenSearch Service は、各着信ドキュメントのデータ型を、Amazon DocumentDB の対応するデータ型に動的にマッピングします。次の表は、OpenSearch Service がさまざまなデータ型を自動的にマッピングする方法を示しています。

データ型	OpenSearch	Amazon DocumentDB
整数	OpenSearch は Amazon DocumentDB 整数値を OpenSearch 整数に自動的にマッピングします。 OpenSearch は、最初に送信されたドキュメントに基づいてフィールドを動的にマッピングします。Amazon DocumentDB に同じ属性のデータ型が混在している場合、自動マッピングが失敗する可能性があります。 例えば、最初のドキュメントが long という属性を持っており、後のドキュメントがその同じ属性を integer として持っている場合、OpenSearch は 2 つ目のドキュメントの取り込みに失敗します。このような場合は、次のような、最も柔軟性が高い数値型を選択する明示的なマッピングテンプレートを提供する必要があります。 `{ "template": {`	Amazon DocumentDB は整数をサポートしています。

データ型	OpenSearch	Amazon DocumentDB

	<pre>"mappings": {
	"properties": {
	"MixedNumberField": {
	"type": "float"
	}
	}
	}
	}
	}</pre>	

データ型	OpenSearch	Amazon DocumentDB
Long	OpenSearch は、Amazon DocumentDB の long 値を OpenSearch の long に自動的にマッピングします。 OpenSearch は、最初に送信されたドキュメントに基づいてフィールドを動的にマッピングします。Amazon DocumentDB に同じ属性のデータ型が混在している場合、自動マッピングが失敗する可能性があります。 例えば、最初のドキュメントが long という属性を持っており、後のドキュメントがその同じ属性を integer として持っている場合、OpenSearch は 2 つ目のドキュメントの取り込みに失敗します。このような場合は、次のような、最も柔軟性が高い数値型を選択する明示的なマッピングテンプレートを提供する必要があります。 `{ "template": { "mappings": { "properties": { "MixedNumberField": { "type": "float" } } } } }`	Amazon DocumentDB は long をサポートしています。

データ型	OpenSearch	Amazon DocumentDB
String	OpenSearch は、文字列の値をテキストとして自動的にマッピングします。状況によっては (列挙型の値など)、キーワード型にマッピングできます。 次の例は、`PartType` という名前の Amazon DocumentDB 属性を、OpenSearch キーワードにマッピングする方法を示しています。 `{ "template": { "mappings": { "properties": { "PartType": { "type": "keyword" } } } } }`	Amazon DocumentDB は文字列をサポートしています。

データ型	OpenSearch	Amazon DocumentDB
Double	OpenSearch は、Amazon DocumentDB の double 値を OpenSearch の double に自動的にマッピングします。 OpenSearch は、最初に送信されたドキュメントに基づいてフィールドを動的にマッピングします。Amazon DocumentDB に同じ属性のデータ型が混在している場合、自動マッピングが失敗する可能性があります。 例えば、最初のドキュメントが long という属性を持っており、後のドキュメントがその同じ属性を integer として持っている場合、OpenSearch は 2 つ目のドキュメントの取り込みに失敗します。このような場合は、次のような、最も柔軟性が高い数値型を選択する明示的なマッピングテンプレートを提供する必要があります。 `{ "template": { "mappings": { "properties": { "MixedNumberField": { "type": "float" } } } } }`	Amazon DocumentDB は double をサポートしています。

データ型	OpenSearch	Amazon DocumentDB
日付	デフォルトでは、日付は OpenSearch の整数にマッピングされます。カスタムマッピングテンプレートを定義して、日付を OpenSearch 日付にマッピングできます。 `{ "template": { "mappings": { "properties": { "myDateField": { "type": "date", "format": "epoch_second" } } } } }`	Amazon DocumentDB は日付をサポートしています。
タイムスタンプ	デフォルトでは、タイムスタンプは OpenSearch の整数にマッピングされます。カスタムマッピングテンプレートを定義して、日付を OpenSearch 日付にマッピングできます。 `{ "template": { "mappings": { "properties": { "myTimestampField": { "type": "date", "format": "epoch_second" } } } } }`	Amazon DocumentDB はタイムスタンプをサポートしています。

データ型	OpenSearch	Amazon DocumentDB
ブール値	OpenSearch は、Amazon DocumentDB のブール型を、OpenSearch のブール型にマッピングします。	Amazon DocumentDB はブール型属性をサポートしています。
10 進数	OpenSearch は、Amazon DocumentDB のマップ属性を、ネストされたフィールドにマッピングします。同じマッピングがネストされたフィールド内でも適用されます。 次の例では、ネストされたフィールドの文字列を、OpenSearch のキーワード型にマッピングします: `{ "template": { "mappings": { "properties": { "myDecimalField": { "type": "double" } } } } }` このカスタムマッピングを使用すると、フィールドをクエリして、倍レベルの精度で集計できます。元の値は、OpenSearch ドキュメントの `_source` プロパティの完全な精度を保持します。このマッピングがない場合、OpenSearch はデフォルトでテキストを使用します。	Amazon DocumentDB は小数をサポートしています。

データ型	OpenSearch	Amazon DocumentDB
正規表現	正規表現タイプはネストされたフィールドを作成します。例えば、*<myFieldName>* .patternや*<myFieldName>* .options などです。	Amazon DocumentDB は正規表現をサポートしています。
バイナリデータ	OpenSearch は、Amazon DocumentDB バイナリデータを OpenSearch テキストに自動的にマッピングします。これらを OpenSearch のバイナリフィールドとして書き込むためのマッピングを提供できます。 次の例は、imageData という名前の Amazon DocumentDB フィールドを、OpenSearch バイナリフィールドにマッピングする方法を示しています。 `{ "template": { "mappings": { "properties": { "imageData": { "type": "binary" } } } } }`	Amazon DocumentDB はバイナリデータフィールドをサポートしています。
ObjectId	objectId 型のフィールドは OpenSearch テキストフィールドにマッピングされます。値は objectId の文字列表現になります。	Amazon DocumentDB は objectIds をサポートしています。

データ型	OpenSearch	Amazon DocumentDB
Null	OpenSearch は、Amazon DocumentDB の null 型のドキュメントを取り込むことができます。値は null 値としてドキュメントに保存されます。この型にはマッピングがなく、このフィールドにはインデックスが付けられず、検索もできません。 同じ属性名が null 型に使用され、後で文字列などの別の型に変更される場合、OpenSearch は、最初の null 以外の値について動的マッピングを作成します。後続の値は引き続き Amazon DocumentDB の null 値にすることができます。	Amazon DocumentDB は null 型のフィールドをサポートしています。
未定義	OpenSearch は、Amazon DocumentDB 未定義型のドキュメントを取り込むことができます。値は null 値としてドキュメントに保存されます。この型にはマッピングがなく、このフィールドにはインデックスが付けられず、検索もできません。 同じフィールド名が未定義型に使用され、後で文字列などの別の型に変更される場合、OpenSearch は、最初の未定義以外の値について動的マッピングを作成します。後続の値は引き続き Amazon DocumentDB の未定義の値にすることができます。	Amazon DocumentDB は未定義型のフィールドをサポートしています。

データ型	OpenSearch	Amazon DocumentDB
MinKey	OpenSearch は、Amazon DocumentDB の minKey 型のドキュメントを取り込むことができます。値は null 値としてドキュメントに保存されます。この型にはマッピングがなく、このフィールドにはインデックスが付けられず、検索もできません。 同じフィールド名が minKey 型に使用され、後で文字列などの別の型に変更される場合、OpenSearch は、最初の minKey 以外の値について動的マッピングを作成します。後続の値は引き続き Amazon DocumentDB の minKey 値にすることができます。	Amazon DocumentDB は minKey 型のフィールドをサポートしています。
MaxKey	OpenSearch は、Amazon DocumentDB の maxKey 型のドキュメントを取り込むことができます。値は null 値としてドキュメントに保存されます。この型にはマッピングがなく、このフィールドにはインデックスが付けられず、検索もできません。 同じフィールド名が maxKey 型に使用され、後で文字列などの別の型に変更される場合、OpenSearch は、最初の maxKey 以外の値について動的マッピングを作成します。後続の値は引き続き Amazon DocumentDB の maxKey 値にすることができます。	Amazon DocumentDB は maxKey 型のフィールドをサポートしています。

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

制限事項

Amazon DocumentDB 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- OpenSearch Ingestion と Amazon DocumentDB の統合は現在、クロスリージョンの取り込みをサポートしていません。Amazon DocumentDB クラスターと OpenSearch Ingestion パイプラインは、同じ AWS リージョンに存在する必要があります。
- OpenSearch Ingestion と Amazon DocumentDB の統合は現在、クロスアカウントの取り込みをサポートしていません。Amazon DocumentDB クラスターと OpenSearch Ingestion パイプラインは、同じ AWS アカウントに存在する必要があります。
- OpenSearch Ingestion パイプラインは、ソースとして 1 つの Amazon DocumentDB クラスターのみをサポートします。
- OpenSearch Ingestion と Amazon DocumentDB の統合は、Amazon DocumentDB インスタンスベースのクラスターを特にサポートしています。Amazon DocumentDB エラスティッククラスターはサポートされていません。
- OpenSearch Ingestion 統合は、Amazon DocumentDB クラスターの認証メカニズム AWS Secrets Manager としてのみ をサポートします。
- 既存のパイプライン設定を更新して、別のデータベースまたはコレクションからデータを取り込むことはできません。代わりに、新しいパイプラインを作成する必要があります。

推奨される CloudWatch アラーム

最高のパフォーマンスを得るには、OpenSearch Ingestion パイプラインを作成して Amazon DocumentDB クラスターにソースとしてアクセスするときに、次の CloudWatch アラームを使用することをお勧めします。

CloudWatch アラーム	説明
`<pipeline-name>` .doucmentdb.credentialsChanged	このメトリクスは、 AWS シークレットがローテーションされる頻度を示します。

CloudWatch アラーム	説明
<pipeline-name> .doucmentdb.executorRefreshErrors	このメトリクスは、 AWS シークレットの更新に失敗したことを示します。
<pipeline-name> .doucmentdb.exportRecordsTotal	このメトリクスは、Amazon DocumentDB からエクスポートされたレコードの数を示します。
<pipeline-name> .doucmentdb.exportRecordsProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたレコードの数を示します。
<pipeline-name> .doucmentdb.exportRecordProcessingErrors	このメトリクスは、Amazon DocumentDB クラスターからデータを読み取る際の OpenSearch Ingestion パイプラインの処理エラーの数を示します。
<pipeline-name> .doucmentdb.exportRecordsSuccessTotal	このメトリクスは、正常に処理されたエクスポートレコードの合計数を示します。
<pipeline-name> .doucmentdb.exportRecordsFailedTotal	このメトリクスは、処理に失敗したエクスポートレコードの合計数を示します。
<pipeline-name> .doucmentdb.bytesReceived	このメトリクスは、OpenSearch Ingestion パイプラインによって受信された合計バイト数を示します。
<pipeline-name> .doucmentdb.bytesProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたバイトの合計数を示します。
<pipeline-name> .doucmentdb.exportPartitionQueryTotal	このメトリクスは、エクスポートパーティションの合計を示します。
<pipeline-name> .doucmentdb.streamRecordsSuccessTotal	このメトリクスは、ストリームから正常に処理されたレコードの数を示します。
<pipeline-name> .doucmentdb.streamRecordsFailedTotal	このメトリクスは、ストリームから処理に失敗したレコードの合計数を示します。

OpenSearch Ingestion パイプラインを Confluent Cloud Kafka で使用する

OpenSearch Ingestion パイプラインを使用して、Confluent Cloud Kafka クラスターから Amazon OpenSearch Service ドメインと OpenSearch Serverless コレクションにデータをストリーミングできます。OpenSearch Ingestion は、Confluent Cloud Kafka クラスターから OpenSearch Service または OpenSearch Serverless によって管理されるドメインまたはコレクションにデータをストリーミングするためのパブリックネットワーク設定とプライベートネットワーク設定の両方をサポートしています。

Confluent Cloud パブリック Kafka クラスターへの接続

OpenSearch Ingestion パイプラインを使用して、パブリック設定の Confluent Cloud Kafka クラスターからデータを移行できます。つまり、ドメイン DNS 名をパブリックに解決できます。これを行うには、ソースとして Confluent Cloud パブリック Kafka クラスター、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、セルフマネージド型ソースクラスターから AWSマネージド型送信先ドメインまたはコレクションへのストリーミングデータが処理されます。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. ソースとして機能する Confluent Cloud Kafka クラスターを作成します。クラスターには、OpenSearch Service に取り込むデータが含まれている必要があります。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
3. AWS Secrets Managerを使用して Confluent Cloud Kafka クラスターで認証を設定します。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

Confluent Cloud Kafka クラスターパイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションに書き込むアクセス許可と、Secrets Manager からシークレットを読み取るアクセス許可を追加します。

ネットワークインターフェイスを管理するには、次のアクセス許可が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:network-interface/*",
                "arn:aws:ec2:us-east-1:111122223333:subnet/*",
                "arn:aws:ec2:us-east-1:111122223333:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "arn:aws:ec2:us-east-1:111122223333:subnet/*"
        },
        {
            "Effect": "Allow",
            "Action": [ "ec2:CreateTags" ],
            "Resource": "arn:aws:ec2:us-east-1:111122223333:network-interface/*",
            "Condition": {
               "StringEquals": { "aws:RequestTag/OSISManaged": "true" }
            }
        }
    ]
}
```

以下は、 AWS Secrets Manager サービスからシークレットを読み取るために必要なアクセス許可です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerReadAccess",
            "Effect": "Allow",
            "Action": ["secretsmanager:GetSecretValue"],
            "Resource": ["arn:aws:secretsmanager:us-
east-1:111122223333:secret:,secret-name"]
        }
    ]
}
```

Amazon OpenSearch Service ドメインに書き込むには、次のアクセス許可が必要です。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:role/pipeline-role"
      },
      "Action": ["es:DescribeDomain", "es:ESHttp*"],
      "Resource": "arn:aws:es:region:account-id:domain/domain-name/*"
    }
  ]
}
```

ステップ 2: パイプラインを作成する

そして、ソースとして Confluent Cloud Kafka を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース Confluent Kafka クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。Confluent スキーマレジストリを使用して Confluent スキーマを定義できます。

```
version: "2"
kafka-pipeline:
  source:
    kafka:
      encryption:
        type: "ssl"
      topics:
        - name: "topic-name"
          group_id: "group-id"
      bootstrap_servers:
        - "bootstrap-server.us-east-1.aws.private.confluent.cloud:9092"
      authentication:
        sasl:
          plain:
            username: ${aws_secrets:confluent-kafka-secret:username}
            password: ${aws_secrets:confluent-kafka-secret:password}
      schema:
        type: confluent
        registry_url: https://my-registry.us-east-1.aws.confluent.cloud
        api_key: "${{aws_secrets:schema-secret:schema_registry_api_key}}"
        api_secret: "${{aws_secrets:schema-secret:schema_registry_api_secret}}"
        basic_auth_credentials_source: "USER_INFO"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
  aws:
    secrets:
      confluent-kafka-secret:
        secret_id: "my-kafka-secret"
        region: "us-east-1"
      schema-secret:
        secret_id: "my-self-managed-kafka-schema"
        region: "us-east-1"
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

VPC 内の Confluent Cloud Kafka クラスターへの接続

OpenSearch Ingestion パイプラインを使用して、VPC で実行されている Confluent Cloud Kafka クラスターからデータを移行することもできます。これを行うには、ソースとして Confluent Cloud Kafka クラスター、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、Confluent Cloud Kafka ソースクラスターから AWSマネージド型ターゲットドメインまたはコレクションへのストリーミングデータが処理されます。

OpenSearch Ingestion は、Confluent でサポートされているすべてのネットワークモードで設定された Confluent Cloud Kafka クラスターをサポートします。OpenSearch Ingestion では、ソースとして次のネットワーク設定モードがサポートされています。

- AWS VPC ピアリング接続
- AWS PrivateLink 専用クラスター用
- AWS PrivateLink エンタープライズクラスター用
- AWS Transit Gateway

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. OpenSearch Service に取り込むデータを含む VPC ネットワーク設定を使用して Confluent Cloud Kafka クラスターを作成します。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
3. AWS Secrets Managerを使用して Confluent Cloud Kafka クラスターで認証を設定します。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. Confluent Cloud Kafka クラスターにアクセスできる VPC の ID を取得します。OpenSearch Ingestion で使用する VPC CIDR を選択します。

> **Note**
>
> を使用してパイプライン AWS マネジメントコンソール を作成する場合は、Confluent Cloud Kafka クラスターを使用するにはOpenSearch Ingestion パイプラインを VPC に

アタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] チェックボックスをオンにして、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自のものを選択します。RFC 1918 のベストカレントプラクティスで定義されているように、プライベートアドレス空間から任意の CIDR を使用できます。
カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion とセルフマネージド OpenSearch 間の IP アドレスの衝突を回避するには、セルフマネージド OpenSearch VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

5. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

> **Note**
>
> AWS PrivateLink を使用して Confluent Cloud Kafka を接続する場合は、VPC DHCP オプションを設定する必要があります。[DNS ホスト名] と [DNS 解決] を有効にします。
> 具体的には、次のオプションセット値を使用します。
> エンタープライズクラスター用:
>
> ```
> domain-name: aws.private.confluent.cloud
> domain-name-servers: AmazonProvidedDNS
> ```
>
> 専用クラスター:
>
> ```
> domain-name: aws.confluent.cloud
> domain-name-servers: AmazonProvidedDNS
> ```
>
> この変更により、Confluent PrivateLink エンドポイントの DNS 解決が VPC 内で正しく機能します。

次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、そのロールに次の許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerReadAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
```

```
            ],
            "Resource": ["arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret-name"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals":
                    {
                        "aws:RequestTag/OSISManaged": "true"
                    }
```

```
            }
        }
    ]
}
```

OpenSearch Ingestion パイプラインの作成に使用する IAM ロールには、上記の Amazon EC2 アクセス許可を指定する必要があります。これは、パイプラインがこれらのアクセス許可を使用して VPC 内のネットワークインターフェイスを作成および削除するためです。パイプラインは、このネットワークインターフェイスを介してのみ Kafka クラスターにアクセスできます。

ステップ 2: パイプラインを作成する

そして、ソースとして Kafka を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース Confluent Kafka クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。Confluent スキーマレジストリを使用して Confluent スキーマを定義できます。

```
 version: "2"
kafka-pipeline:
  source:
    kafka:
      encryption:
        type: "ssl"
      topics:
        - name: "topic-name"
          group_id: "group-id"
      bootstrap_servers:
        - "bootstrap-server.us-east-1.aws.private.confluent.cloud:9092"
      authentication:
        sasl:
          plain:
            username: ${aws_secrets:confluent-kafka-secret:username}
            password: ${aws_secrets:confluent-kafka-secret:password}
      schema:
        type: confluent
```

```
        registry_url: https://my-registry.us-east-1.aws.confluent.cloud
        api_key: "${{aws_secrets:schema-secret:schema_registry_api_key}}"
        api_secret: "${{aws_secrets:schema-secret:schema_registry_api_secret}}"
        basic_auth_credentials_source: "USER_INFO"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
      index: "confluent-index"
extension:
  aws:
    secrets:
      confluent-kafka-secret:
        secret_id: "my-kafka-secret"
        region: "us-east-1"
      schema-secret:
        secret_id: "my-self-managed-kafka-schema"
        region: "us-east-2"
```

での OpenSearch Ingestion パイプラインの使用 Amazon Managed Streaming for Apache Kafka

Kafka プラグインを使用して Amazon Managed Streaming for Apache Kafka (Amazon MSK) から OpenSearch Ingestion パイプラインにデータを取り込むことができます。Amazon MSK を使用すると、Apache Kafka をストリーミングデータの処理に使用するアプリケーションを構築および実行できます。OpenSearch Ingestion は AWS PrivateLink 、 を使用して Amazon MSK に接続します。Amazon MSK クラスターと Amazon MSK Serverless クラスターの両方からデータを取り込むことができます。2 つのプロセスの違いは、パイプラインをセットアップする前に実行する必要がある前提条件の手順だけです。

トピック

- プロビジョニングされた Amazon MSK の前提条件
- Amazon MSK Serverless の前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- ステップ 3: (オプション) AWS Glue スキーマレジストリを使用する
- ステップ 4: (オプション) Amazon MSK パイプラインの推奨コンピューティングユニット (OCU) を設定する

プロビジョニングされた Amazon MSK の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon Managed Streaming for Apache Kafka 開発者ガイドの「クラスターの作成」の手順に従って Amazon MSK のプロビジョニングされたクラスターを作成します。[ブローカータイプ] では、t3 タイプは OpenSearch Ingestion ではサポートされていないため、それ以外のオプションを選択します。
2. クラスターのステータスが Active になったら、「マルチ VPC 接続を有効にする」の手順に従います。
3. クラスターとパイプラインが同じ AWS アカウントにあるかどうかに応じて、「クラスターポリシーを MSK クラスターにアタッチする」のステップに従い、以下のポリシーのいずれかをアタッチします。このポリシーでは、OpenSearch Ingestion は Amazon MSK クラスターへの AWS PrivateLink 接続を作成して、Kafka トピックからデータを読み取ることができます。必ず独自の ARN で `resource` を更新してください。

   クラスターとパイプラインが同じ AWS アカウントにある場合は、次のポリシーが適用されます。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis.amazonaws.com"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis-pipelines.amazonaws.com"
      },
```

```
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    }
  ]
}
```

Amazon MSK クラスターがパイプライン AWS アカウント とは異なる にある場合は、代わりに次のポリシーをアタッチします。クロスアカウントアクセスは、プロビジョニングされた Amazon MSK クラスターでのみ可能であり、Amazon MSK Serverless クラスターではできないことに注意してください。の AWS `principal` ARN は、パイプライン設定に提供するのと同じパイプラインロールの ARN である必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis.amazonaws.com"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis-pipelines.amazonaws.com"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeClusterV2"
```

```
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "kafka-cluster:*",
        "kafka:*"
      ],
      "Resource": [
        "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-name/cluster-id",
        "arn:aws:kafka:us-east-1:111122223333:topic/cluster-name/cluster-id/*",
        "arn:aws:kafka:us-east-1:111122223333:group/cluster-name/*"
      ]
    }
  ]
}
```

4. 「トピックの作成」の手順に従って Kafka トピックを作成します。*BootstrapServerString* がプライベートエンドポイント (単一 VPC) のブートストラップ URL の 1 つであることを確認してください。`--replication-factor` の値は、Amazon MSK クラスターのゾーンの数に応じて 2 または 3 を指定します。`--partitions` の値は少なくとも 10 である必要があります。
5. 「データの生成と消費」の手順に従って、データを生成して使用します。*BootstrapServerString* がプライベートエンドポイント (単一 VPC) のブートストラップ URL の 1 つであることを確認してください。

Amazon MSK Serverless の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon Managed Streaming for Apache Kafka 開発者ガイドの「MSK Serverless クラスターの作成」の手順に従って Amazon MSK Serverless クラスターを作成します。
2. クラスターのステータスが [アクティブ] になったら、「クラスターポリシーを MSK クラスターにアタッチする」の手順に従って、次のポリシーをアタッチします。必ず独自の ARN で `resource` を更新してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis.amazonaws.com"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis-pipelines.amazonaws.com"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
    }
  ]
}
```

このポリシーにより、OpenSearch Ingestion は Amazon MSK Serverless クラスター AWS PrivateLink への接続を作成し、Kafka トピックからデータを読み取ることができます。このポリシーは、クラスターとパイプラインが同じ にある場合に適用されます。これは AWS アカウント、Amazon MSK Serverless がクロスアカウントアクセスをサポートしていないためです。

3. 「トピックの作成」の手順に従って Kafka トピックを作成します。*BootstrapServerString* が Simple Authentication and Security Layer (SASL) IAM ブートストラップ URL の 1 つであることを確認します。`--replication-factor` の値は、Amazon MSK Serverless クラスターのゾー

ンの数に応じて 2 または 3 を指定します。--partitions の値は少なくとも 10 である必要があります。

4. 「データの生成と消費」の手順に従って、データを生成して使用します。ここでも、*BootstrapServerString* が Simple Authentication and Security Layer (SASL) IAM ブートストラップ URL の 1 つであることを確認します。

ステップ 1: パイプラインロールを設定する

Amazon MSK をプロビジョニングし、サーバーレスクラスターを設定したら、パイプライン設定で使用するパイプラインロールに次の Kafka アクセス許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster",
                "kafka:DescribeClusterV2",
                "kafka:GetBootstrapBrokers"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:111122223333:cluster/cluster-
name/cluster-id"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:111122223333:topic/cluster-name/cluster-
id/topic-name"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:111122223333:group/cluster-name/*"
            ]
        }
    ]
}
```

ステップ 2: パイプラインを作成する

そして、ソースとして Kafka を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

```
version: "2"
log-pipeline:
  source:
    kafka:
      acknowledgements: true
      topics:
      - name: "topic-name"
        group_id: "grouplambd-id"
      aws:
        msk:
          arn: "arn:aws:kafka:region:account-id:cluster/cluster-name/cluster-id"
        region: "us-west-2"
  processor:
  - grok:
      match:
        message:
        - "%{COMMONAPACHELOG}"
  - date:
      destination: "@timestamp"
      from_time_received: true
  sink:
  - opensearch:
      hosts: ["https://search-domain-endpoint.us-east-1es.amazonaws.com"]
      index: "index_name"
```

```
      aws_region: "region"
      aws_sigv4: true
```

事前設定された Amazon MSK ブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

ステップ 3: (オプション) AWS Glue スキーマレジストリを使用する

Amazon MSK で OpenSearch Ingestion を使用する場合、 AWS Glue スキーマレジストリでホストされているスキーマに AVRO データ形式を使用できます。AWS Glue スキーマレジストリを使用すると、データストリームスキーマを一元的に検出、制御、および展開できます。

このオプションを使用するには、パイプライン設定で `type` スキーマを有効にします。

```
schema:
  type: "aws_glue"
```

また、パイプラインロールで AWS Glue に読み取りアクセス許可を付与する必要があります。AWSGlueSchemaRegistryReadonlyAccess という AWS マネージドポリシーを使用できます。さらに、レジストリは OpenSearch Ingestion パイプラインと同じ AWS アカウント およびリージョンにある必要があります。

ステップ 4: (オプション) Amazon MSK パイプラインの推奨コンピューティングユニット (OCU) を設定する

各コンピューティングユニットには、トピックごとに 1 つのコンシューマーがあります。ブローカーは、特定のトピックについて、これらのコンシューマー間でパーティションのバランスを取ります。ただし、パーティションの数がコンシューマーの数よりも多い場合、Amazon MSK は各コンシューマーで複数のパーティションをホストします。OpenSearch Ingestion には、CPU 使用率またはパイプライン内の保留中のレコード数に基づいてスケールアップまたはスケールダウンする自動スケーリングが組み込まれています。

最適なパフォーマンスを得るには、パーティションを多くのコンピューティングユニットに分散して並列処理を行います。トピックに多くのパーティションがある場合 (パイプラインあたりの最大数である 96 以上の OCU がある場合など)、1 ~ 96 個の OCU でパイプラインを設定することをお勧めします。これは、必要に応じて自動的にスケールするためです。トピックのパーティション数が少ない場合 (96 未満の場合など)、最大コンピューティングユニットをパーティションの数と同じにします。

パイプラインに複数のトピックがある場合は、最大コンピューティングユニットを設定する参照としてパーティション数が最も多いトピックを選択します。新しい OCU セットを含むパイプラインを同じトピックとコンシューマーグループに追加すると、スループットをほぼ直線的にスケールすることができます。

Amazon RDS で OpenSearch Ingestion パイプラインの使用

Amazon RDS または Amazon RDS で OpenSearch インジェストパイプラインを使用すると、既存のデータをエクスポートしたり、変更 (作成、アップデート、削除など) を Amazon OpenSearch Service のドメインやコレクションにストリーミングしたりできます。OpenSearch Ingestion パイプラインには変更データキャプチャ (CDC) インフラストラクチャが組み込まれており、Amazon RDS または Amazon RDS からデータを継続的にストリーミングするための大規模で低レイテンシーの方法を提供します。RDS for MySQL と RDS for PostgreSQL がサポートされています。

データを処理するソースとして Amazon RDS を使用するには、完全な初期スナップショットを使用する方法と使用しない方法の 2 つがあります。完全な初期スナップショットは、指定されたテーブルのスナップショットであり、このスナップショットは Amazon S3 にエクスポートされます。そこから、OpenSearch Ingestion パイプラインによってドメイン内の 1 つのインデックスに送信されるか、またはドメイン内の複数のインデックスにパーティショニングされます。Amazon RDS と OpenSearch のデータの整合性を保つために、パイプラインは、Amazon RDS インスタンス内のテーブルで発生する作成、更新、削除のすべてのイベントを、(1 つまたは複数の) OpenSearch インデックスに保存されているドキュメントと同期します。

完全な初期スナップショットを使用する場合、OpenSearch Ingestion パイプラインは、まずスナップショットを取り込んでから、Amazon RDS 変更ストリームからのデータの読み取りを開始します。最終的には追い付いて、Amazon RDS と OpenSearch の間でほぼリアルタイムのデータ整合性を維持します。

OpenSearch Ingestion と Amazon RDS の統合を使用して、変更データのキャプチャを追跡し、Aurora のすべての更新を OpenSearch に取り込むこともできます。他のメカニズムからの完全なスナップショットが既にある場合、または Amazon RDS インスタンスのデータに対するすべての変更をキャプチャする場合は、このオプションを選択します。

このオプションを選択する場合は、Amazon RDS for MySQL バイナリログ記録を設定するか、Amazon RDS for PostgreSQL DB インスタンスの論理レプリケーションを設定する必要があります。

トピック

- RDS for MySQL

- RDS for PostgreSQL
- 複数のテーブルからのデータの結合

RDS for MySQL

Amazon RDS for MySQL で OpenSearch Ingestion パイプラインを設定するには、次の手順を実行します。

トピック

- RDS for MySQL の前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- データ整合性
- データ型のマッピング
- 制限事項
- 推奨される CloudWatch アラーム

RDS for MySQL の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon RDS でカスタム DB パラメータグループを作成して、バイナリログ記録を設定し、次のパラメータを設定します。

   ```
   binlog_format=ROW
   binlog_row_image=full
   binlog_row_metadata=FULL
   ```

   また、`binlog_row_value_options` パラメータが `PARTIAL_JSON` に設定されていないことも確認してください。

   詳細については、「Aurora MySQL バイナリログの設定」を参照してください。
2. RDS for MySQL DB インスタンスを選択または作成し、前のステップで作成したパラメータグループを DB インスタンスに関連付けます。
3. データベースで自動バックアップが有効になっていることを確認します。詳細については、「自動バックアップの保持」を参照してください。

4. 24 時間など、レプリケーションが発生するのに十分な時間でバイナリログの保持を設定します。詳細については、「Amazon RDS ユーザーガイド」の「バイナリログ構成の設定と表示」を参照してください。
5. Amazon RDS と でパスワード管理を使用して、Amazon RDS インスタンスで AWS Secrets Managerユーザー名とパスワード認証を設定します。Secrets Manager シークレットを作成するで、ユーザー名とパスワードの組み合わせを作成することもできます。
6. 完全な初期スナップショット機能を使用する場合は、 AWS KMS key と Amazon RDS から Amazon S3 にデータをエクスポートするための IAM ロールを作成します。

   IAM ロールには、次の許可ポリシーが必要です。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::s3-bucket-used-in-pipeline",
                "arn:aws:s3:::s3-bucket-used-in-pipeline/*"
            ]
        }
    ]
}
```

   ロールにはまた、次の信頼関係が必要です:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "export.rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

7. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを選択または作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
8. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は Amazon RDS DB インスタンスからドメインまたはコレクションにデータを書き込むことができます。

ステップ 1: パイプラインロールを設定する

Amazon RDS パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定します。また、Amazon RDS ソースの次のアクセス許可をロールに追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
    "Sid": "allowReadingFromS3Buckets",
    "Effect": "Allow",
    "Action": [
    "s3:GetObject",
    "s3:DeleteObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:PutObject"
    ],
    "Resource": [
    "arn:aws:s3:::s3_bucket",
    "arn:aws:s3:::s3_bucket/*"
    ]
```

```
    },
    {
    "Sid": "allowNetworkInterfacesActions",
    "Effect": "Allow",
    "Action": [
    "ec2:AttachNetworkInterface",
    "ec2:CreateNetworkInterface",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:DetachNetworkInterface",
    "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": [
    "arn:aws:ec2:*:111122223333:network-interface/*",
    "arn:aws:ec2:*:111122223333:subnet/*",
    "arn:aws:ec2:*:111122223333:security-group/*"
    ]
    },
    {
    "Sid": "allowDescribeEC2",
    "Effect": "Allow",
    "Action": [
    "ec2:Describe*"
    ],
    "Resource": "*"
    },
    {
    "Sid": "allowTagCreation",
    "Effect": "Allow",
    "Action": [
    "ec2:CreateTags"
    ],
    "Resource": "arn:aws:ec2:*:111122223333:network-interface/*",
    "Condition": {
    "StringEquals": {
    "aws:RequestTag/OSISManaged": "true"
    }
    }
    },
    {
    "Sid": "AllowDescribeInstances",
    "Effect": "Allow",
    "Action": [
```

```
    "rds:DescribeDBInstances"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:db:*"
    ]
    },
    {
    "Sid": "AllowSnapshots",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeDBSnapshots",
    "rds:CreateDBSnapshot",
    "rds:AddTagsToResource"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:db:DB-id",
    "arn:aws:rds:us-east-2:111122223333:snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowExport",
    "Effect": "Allow",
    "Action": [
    "rds:StartExportTask"
    ],
    "Resource": [
    "arn:aws:rds:us-east-2:111122223333:snapshot:DB-id*"
    ]
    },
    {
    "Sid": "AllowDescribeExports",
    "Effect": "Allow",
    "Action": [
    "rds:DescribeExportTasks"
    ],
    "Resource": "*",
    "Condition": {
    "StringEquals": {
    "aws:RequestedRegion": "us-east-2",
    "aws:ResourceAccount": "111122223333"
    }
    }
    },
    {
```

```
    "Sid": "AllowAccessToKmsForExport",
    "Effect": "Allow",
    "Action": [
    "kms:Decrypt",
    "kms:Encrypt",
    "kms:DescribeKey",
    "kms:RetireGrant",
    "kms:CreateGrant",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
    ],
    "Resource": [
    "arn:aws:kms:us-east-2:111122223333:key/export-key-id"
    ]
    },
    {
    "Sid": "AllowPassingExportRole",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": [
    "arn:aws:iam::111122223333:role/export-role"
    ]
    },
    {
    "Sid": "SecretsManagerReadAccess",
    "Effect": "Allow",
    "Action": [
    "secretsmanager:GetSecretValue"
    ],
    "Resource": [
    "arn:aws:secretsmanager:*:111122223333:secret:*"
    ]
    }
    ]
    }
```

ステップ 2: パイプラインを作成する

次のような OpenSearch Ingestion パイプラインを設定します。サンプルパイプラインは、Amazon RDS インスタンスをソースとして指定します。

```
version: "2"
```

```
rds-mysql-pipeline:
  source:
    rds:
      db_identifier: "instance-id"
      engine: mysql
      database: "database-name"
      tables:
        include:
          - "table1"
          - "table2"
      s3_bucket: "bucket-name"
      s3_region: "bucket-region"
      s3_prefix: "prefix-name"
      export:
        kms_key_id: "kms-key-id"
        iam_role_arn: "export-role-arn"
      stream: true
      aws:
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        region: "us-east-1"
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
        index: "${getMetadata(\"table_name\")}"
        index_type: custom
        document_id: "${getMetadata(\"primary_key\")}"
        action: "${getMetadata(\"opensearch_action\")}"
        document_version: "${getMetadata(\"document_version\")}"
        document_version_type: "external"
        aws:
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
          region: "us-east-1"
extension:
  aws:
    secrets:
      secret:
        secret_id: "rds-secret-id"
        region: "us-east-1"
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        refresh_interval: PT1H
```

事前設定された Amazon MSK ブループリントを使用して、このパイプラインを作成できます。詳細については、「ブループリントの使用」を参照してください。

Amazon Aurora をソースとして使用するには、パイプラインの VPC アクセスを設定する必要があります。選択する VPC は、Amazon Aurora ソースが使用するのと同じ VPC である必要があります。次に、1 つ以上のサブネットと 1 つ以上の VPC セキュリティグループを選択します。パイプラインには Aurora MySQL データベースへのネットワークアクセスが必要であるため、パイプラインの VPC セキュリティグループからデータベースポートへのインバウンドトラフィックを許可する VPC セキュリティグループが Aurora クラスターに設定されていることも確認する必要があります。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

を使用してパイプライン AWS マネジメントコンソール を作成する場合は、ソースとして Amazon Aurora を使用するには、パイプラインを VPC にアタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] を選択し、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自の CIDR を選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。

カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion と Amazon RDS 間の IP アドレスの衝突を回避するには、Amazon DRDS VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

詳細については、「Configuring VPC access for a pipeline」を参照してください。

データ整合性

パイプラインは、Amazon RDS インスタンスから変更を継続的にポーリングまたは受信し、OpenSearch インデックス内の対応するドキュメントを更新することで、データ整合性を確保します。

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了としてマークします。OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込む場合は、パイプラインがドキュメント ID を生成しないため、パイプラインシンク設定`document_id: "${getMetadata(\"primary_key\")}"`で を省略する必要があることに注意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、Amazon RDS 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更とリコンサイルされます。

データ型のマッピング

OpenSearch Ingestion パイプラインは、MySQL データ型を、消費する OpenSearch Service ドメインまたはコレクションに適した表現にマッピングします。OpenSearch でマッピングテンプレートが定義されていない場合、OpenSearch は最初に送信されたドキュメントに基づいて動的マッピングを持つフィールドタイプを自動的に決定します。マッピングテンプレートを使用して、OpenSearch で最適なフィールドタイプを明示的に定義することもできます。

次の表に、MySQL データ型と対応する OpenSearch フィールドタイプを示します。明示的なマッピングが定義されていない場合、デフォルトの OpenSearch フィールドタイプ列には OpenSearch の対応するフィールドタイプが表示されます。この場合、OpenSearch は動的マッピングを使用してフィールドタイプを自動的に決定します。推奨 OpenSearch フィールドタイプ列は、マッピングテンプレートで明示的に指定することが推奨される対応するフィールドタイプです。これらのフィールドタイプは MySQL のデータ型とより密接に連携しており、通常、OpenSearch で使用できる検索機能を改善できます。

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
BIGINT	long	long
BIGINT UNSIGNED	long	符号なしロング
BIT	long	ビット数に応じてバイト、ショート、整数、ロング
DECIMAL	テキスト	double またはキーワード
DOUBLE	float	double
FLOAT	float	float
INT	long	整数

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
INT UNSIGNED	long	long
MEDIUMIN	long	整数
MEDIUMIN UNSIGNED	long	整数
NUMERIC	テキスト	double またはキーワード
SMALLINT	long	short
SMALLINT UNSIGNED	long	整数
TINYINT	long	バイト
TINYINT UNSIGNED	long	short
BINARY	テキスト	バイナリ
BLOB	テキスト	バイナリ
CHAR	テキスト	テキスト
ENUM	テキスト	キーワード
LONGBLOE	テキスト	バイナリ
LONGTEXT	テキスト	テキスト
MEDIUMBL B	テキスト	バイナリ
MEDIUMTE T	テキスト	テキスト

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
SET	テキスト	キーワード
TEXT	テキスト	テキスト
TINYBLOB	テキスト	バイナリ
TINYTEXT	テキスト	テキスト
VARBINAR	テキスト	バイナリ
VARCHAR	テキスト	テキスト
DATE	long (エポックミリ秒単位)	date
DATETIME	long (エポックミリ秒単位)	date
TIME	long (エポックミリ秒単位)	date
TIMESTAM	long (エポックミリ秒単位)	date
YEAR	long (エポックミリ秒単位)	date
GEOMETR	text (WKT 形式)	geo_shape
GEOMETR OLLECTIO	text (WKT 形式)	geo_shape
LINESTRIN G	text (WKT 形式)	geo_shape
MULTILINE STRING	text (WKT 形式)	geo_shape
MULTIPOI T	text (WKT 形式)	geo_shape

MySQL のデータ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
MULTIPOLYGON	text (WKT 形式)	geo_shape
POINT	text (WKT 形式)	geo_point または geo_shape
POLYGON	text (WKT 形式)	geo_shape
JSON	テキスト	オブジェクト

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

制限事項

RDS for MySQL 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- 統合は、パイプラインごとに 1 つの MySQL データベースのみをサポートします。
- 統合は現在、クロスリージョンデータインジェストをサポートしていません。Amazon RDS インスタンスと OpenSearch ドメインは同じ AWS リージョンに存在する必要があります。
- 統合は現在、クロスアカウントデータインジェストをサポートしていません。Amazon RDS インスタンスと OpenSearch Ingestion パイプラインが同じ AWS アカウントにある必要があります。
- Amazon RDS インスタンスで、サポートされている唯一の認証メカニズムである Secrets Manager を使用して認証が有効になっていることを確認します。
- 既存のパイプライン設定を更新して、別のデータベースや別のテーブルからデータを取り込むことはできません。パイプラインのデータベース名やテーブル名を更新するには、新しいパイプラインを作成する必要があります。

- データ定義言語 (DDL) ステートメントは一般的にサポートされていません。以下の場合、データ整合性は維持されません。
  - プライマリキーが変更されます (追加/削除/名前変更)。
  - テーブルは削除/切り捨てられます。
  - 列名またはデータ型が変更されました。
- 同期する MySQL テーブルにプライマリキーが定義されていない場合、データ整合性は保証されません。更新/削除を OpenSearch に同期できるようにするには、OpenSearch シンク設定でカスタム `document_id` オプションを適切に定義する必要があります。
- カスケード削除アクションを使用した外部キー参照はサポートされていないため、RDS for MySQL と OpenSearch の間でデータの不整合が発生する可能性があります。
- Amazon RDS マルチアベイラビリティーゾーン DB クラスターはサポートされていません。
- サポートされているバージョン: MySQL バージョン 8.0 以降。

推奨される CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント数、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメントの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

メトリクス	説明
pipeline-name .rds.credentialsChanged	このメトリクスは、 AWS シークレットがローテーションされる頻度を示します。
pipeline-name .rds.executorRefreshErrors	このメトリクスは、 AWS シークレットの更新に失敗したことを示します。
pipeline-name .rds.exportRecordsTotal	このメトリクスは、Amazon Aurora からエクスポートされたレコードの数を示します。

メトリクス	説明
pipeline-name .rds.exportRecordsProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたレコードの数を示します。
pipeline-name .rds.exportRecordProcessingErrors	このメトリクスは、Amazon Aurora クラスターからデータを読み取る際の OpenSearch Ingestion パイプラインの処理エラーの数を示します。
pipeline-name .rds.exportRecordsSuccessTotal	このメトリクスは、正常に処理されたエクスポートレコードの合計数を示します。
pipeline-name .rds.exportRecordsFailedTotal	このメトリクスは、処理に失敗したエクスポートレコードの合計数を示します。
pipeline-name .rds.bytesReceived	このメトリクスは、OpenSearch Ingestion パイプラインによって受信された合計バイト数を示します。
pipeline-name .rds.bytesProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたバイトの合計数を示します。
pipeline-name .rds.streamRecordsSuccessTotal	このメトリクスは、ストリームから正常に処理されたレコードの数を示します。
pipeline-name .rds.streamRecordsFailedTotal	このメトリクスは、ストリームから処理に失敗したレコードの合計数を示します。

RDS for PostgreSQL

Amazon RDS for PostgreSQL で OpenSearch Ingestion パイプラインを設定するには、次の手順を実行します。

トピック

- RDS for PostgreSQL の前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- データ整合性
- データ型のマッピング
- 制限事項
- 推奨される CloudWatch アラーム

RDS for PostgreSQL の前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. Amazon RDS でカスタム DB パラメータグループを作成して、論理レプリケーションを設定します。

   ```
   rds.logical_replication=1
   ```

   詳細については、「Amazon RDS for PostgreSQL の論理レプリケーションの実行」を参照してください。
2. RDS for PostgreSQL DB インスタンスを選択または作成し、ステップ 1 で作成したパラメータグループを DB インスタンスに関連付けます。
3. Amazon RDS と でパスワード管理を使用して、Amazon RDS インスタンスで AWS Secrets Managerユーザー名とパスワード認証を設定します。Secrets Manager シークレットを作成するで、ユーザー名とパスワードの組み合わせを作成することもできます。
4. 完全な初期スナップショット機能を使用する場合は、 AWS KMS key と Amazon RDS から Amazon S3 にデータをエクスポートするための IAM ロールを作成します。

   IAM ロールには、次の許可ポリシーが必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::s3-bucket-used-in-pipeline",
                "arn:aws:s3:::s3-bucket-used-in-pipeline/*"
            ]
        }
    ]
}
```

ロールにはまた、次の信頼関係が必要です:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "export.rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

5. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを選択または作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
6. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は Amazon RDS DB インスタンスからドメインまたはコレクションにデータを書き込むことができます。

ステップ 1: パイプラインロールを設定する

Amazon RDS パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定します。また、Amazon RDS ソースの次のアクセス許可をロールに追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
    "Sid": "allowReadingFromS3Buckets",
    "Effect": "Allow",
    "Action": [
    "s3:GetObject",
    "s3:DeleteObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:PutObject"
    ],
    "Resource": [
    "arn:aws:s3:::s3_bucket",
    "arn:aws:s3:::s3_bucket/*"
    ]
    },
    {
    "Sid": "allowNetworkInterfacesActions",
    "Effect": "Allow",
    "Action": [
    "ec2:AttachNetworkInterface",
    "ec2:CreateNetworkInterface",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteNetworkInterfacePermission",
```

```
        "ec2:DetachNetworkInterface",
        "ec2:DescribeNetworkInterfaces"
        ],
        "Resource": [
        "arn:aws:ec2:*:111122223333:network-interface/*",
        "arn:aws:ec2:*:111122223333:subnet/*",
        "arn:aws:ec2:*:111122223333:security-group/*"
        ]
        },
        {
        "Sid": "allowDescribeEC2",
        "Effect": "Allow",
        "Action": [
        "ec2:Describe*"
        ],
        "Resource": "*"
        },
        {
        "Sid": "allowTagCreation",
        "Effect": "Allow",
        "Action": [
        "ec2:CreateTags"
        ],
        "Resource": "arn:aws:ec2:*:111122223333:network-interface/*",
        "Condition": {
        "StringEquals": {
        "aws:RequestTag/OSISManaged": "true"
        }
        }
        },
        {
        "Sid": "AllowDescribeInstances",
        "Effect": "Allow",
        "Action": [
        "rds:DescribeDBInstances"
        ],
        "Resource": [
        "arn:aws:rds:us-east-2:111122223333:db:*"
        ]
        },
        {
        "Sid": "AllowSnapshots",
        "Effect": "Allow",
        "Action": [
```

```
"rds:DescribeDBSnapshots",
"rds:CreateDBSnapshot",
"rds:AddTagsToResource"
],
"Resource": [
"arn:aws:rds:us-east-2:111122223333:db:DB-id",
"arn:aws:rds:us-east-2:111122223333:snapshot:DB-id*"
]
},
{
"Sid": "AllowExport",
"Effect": "Allow",
"Action": [
"rds:StartExportTask"
],
"Resource": [
"arn:aws:rds:us-east-2:111122223333:snapshot:DB-id*"
]
},
{
"Sid": "AllowDescribeExports",
"Effect": "Allow",
"Action": [
"rds:DescribeExportTasks"
],
"Resource": "*",
"Condition": {
"StringEquals": {
"aws:RequestedRegion": "us-east-2",
"aws:ResourceAccount": "111122223333"
}
}
},
{
"Sid": "AllowAccessToKmsForExport",
"Effect": "Allow",
"Action": [
"kms:Decrypt",
"kms:Encrypt",
"kms:DescribeKey",
"kms:RetireGrant",
"kms:CreateGrant",
"kms:ReEncrypt*",
"kms:GenerateDataKey*"
```

```
    ],
    "Resource": [
    "arn:aws:kms:us-east-2:111122223333:key/export-key-id"
    ]
    },
    {
    "Sid": "AllowPassingExportRole",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": [
    "arn:aws:iam::111122223333:role/export-role"
    ]
    },
    {
    "Sid": "SecretsManagerReadAccess",
    "Effect": "Allow",
    "Action": [
    "secretsmanager:GetSecretValue"
    ],
    "Resource": [
    "arn:aws:secretsmanager:*:111122223333:secret:*"
    ]
    }
    ]
    }
```

ステップ 2: パイプラインを作成する

そして、ソースとして RDS for PostgreSQL インスタンスを指定する OpenSearch Ingestion パイプラインを次のように設定できます。

```
version: "2"
rds-postgres-pipeline:
  source:
    rds:
      db_identifier: "instance-id"
      engine: postgresql
      database: "database-name"
      tables:
        include:
          - "schema1.table1"
          - "schema2.table2"
```

```
      s3_bucket: "bucket-name"
      s3_region: "bucket-region"
      s3_prefix: "prefix-name"
      export:
        kms_key_id: "kms-key-id"
        iam_role_arn: "export-role-arn"
      stream: true
      aws:
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        region: "us-east-1"
      authentication:
        username: ${{aws_secrets:secret:username}}
        password: ${{aws_secrets:secret:password}}
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
        index: "${getMetadata(\"table_name\")}"
        index_type: custom
        document_id: "${getMetadata(\"primary_key\")}"
        action: "${getMetadata(\"opensearch_action\")}"
        document_version: "${getMetadata(\"document_version\")}"
        document_version_type: "external"
        aws:
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
          region: "us-east-1"
extension:
  aws:
    secrets:
      secret:
        secret_id: "rds-secret-id"
        region: "us-east-1"
        sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        refresh_interval: PT1H
```

> **Note**
>
> 事前設定された Amazon MSK ブループリントを使用して、このパイプラインを作成できます。詳細については、「ブループリントの使用」を参照してください。

RDS for PostgreSQL をソースとして使用するには、パイプラインの VPC アクセスを設定する必要があります。選択した VPC は、Amazon RDS ソースが使用する VPC と同じである必要がありま

す。次に、1 つ以上のサブネットと 1 つ以上の VPC セキュリティグループを選択します。パイプラインには RDS for PostgreSQL データベースへのネットワークアクセスが必要であるため、Amazon RDS インスタンスに、パイプラインの VPC セキュリティグループからデータベースポートへのインバウンドトラフィックを許可する VPC セキュリティグループが設定されていることを確認する必要があります。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

を使用してパイプライン AWS マネジメントコンソール を作成する場合は、ソースとして Amazon Aurora を使用するには、パイプラインを VPC にアタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] を選択し、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自の CIDR を選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。

カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion と Amazon RDS 間の IP アドレスの衝突を回避するには、Amazon Aurora VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

詳細については、「Configuring VPC access for a pipeline」を参照してください。

データ整合性

パイプラインは、Amazon RDS インスタンスから変更を継続的にポーリングまたは受信し、OpenSearch インデックス内の対応するドキュメントを更新することで、データ整合性を確保します。

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインがスナップショットまたはストリームを読み取る際に、並列処理用のパーティションが動的に作成されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、パーティションを完了としてマークします。OpenSearch Serverless 検索コレクションに取り込みたい場合は、パイプラインでドキュメント ID を生成できます。OpenSearch Serverless 時系列コレクションに取り込む場合は、パイプラインがドキュメント ID を生成しないため、パイプラインシンク設定`document_id: "${getMetadata(\"primary_key\")}"`で を省略する必要があることに注意してください。

また、OpenSearch Ingestion パイプラインは、着信イベントアクションを、対応する一括インデックス作成アクションにマッピングして、ドキュメントの取り込みをサポートします。これにより、データ整合性が維持され、Amazon RDS 内のすべてのデータ変更が OpenSearch 内の対応するドキュメントの変更とリコンサイルされます。

データ型のマッピング

OpenSearch Ingestion パイプラインは、PostgreSQL データ型を、消費する OpenSearch Service ドメインまたはコレクションに適した表現にマッピングします。OpenSearch でマッピングテンプレートが定義されていない場合、OpenSearch は最初に送信されたドキュメントに基づいて動的マッピングを使用してフィールドタイプを自動的に決定します。マッピングテンプレートを使用して、OpenSearch で最適なフィールドタイプを明示的に定義することもできます。

次の表に、RDS for PostgreSQL データ型と対応する OpenSearch フィールド型を示します。明示的なマッピングが定義されていない場合、デフォルトの OpenSearch フィールドタイプ列には OpenSearch の対応するフィールドタイプが表示されます。この場合、OpenSearch は動的マッピングを使用してフィールドタイプを自動的に決定します。推奨 OpenSearch フィールドタイプ列は、マッピングテンプレートで明示的に指定する、対応する推奨フィールドタイプです。これらのフィールドタイプは RDS for PostgreSQL のデータ型とより密接に連携しており、通常、OpenSearch で使用できる検索機能を改善できます。

RDS PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
smallint	long	short
整数	long	整数
bigint	long	long
decimal	テキスト	double またはキーワード
numeric[(p, s)]	テキスト	double またはキーワード
real	float	float
double precision	float	double
smallserial	long	short

RDS PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
シリアル	long	整数
bigserial	long	long
money	オブジェクト	オブジェクト
character varying(n)	テキスト	テキスト
varchar(n)	テキスト	テキスト
character (n)	テキスト	テキスト
char(n)	テキスト	テキスト
bpchar(n)	テキスト	テキスト
bpchar	テキスト	テキスト
テキスト	テキスト	テキスト
enum	テキスト	テキスト
bytea	テキスト	バイナリ
timestamp [(p)] [without time zone]	long (エポックミリ秒単位)	date
timestamp [(p)] with time zone	long (エポックミリ秒単位)	date

RDS PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
date	long (エポックミリ秒単位)	date
time [(p)] [without time zone]	long (エポックミリ秒単位)	date
time [(p)] with time zone	long (エポックミリ秒単位)	date
interval [fields] [(p)]	text (ISO8601 形式)	テキスト
boolean	boolean	boolean
point	text (WKT 形式)	geo_shape
LINE	text (WKT 形式)	geo_shape
lseg	text (WKT 形式)	geo_shape
box	text (WKT 形式)	geo_shape
パス	text (WKT 形式)	geo_shape
Polygon (ポリゴン)	text (WKT 形式)	geo_shape
circle	オブジェクト	オブジェクト
cidr	テキスト	テキスト

RDS PostgreSQL データ型	デフォルトの OpenSearch フィールドタイプ	推奨される OpenSearch フィールドタイプ
inet	テキスト	テキスト
macaddr	テキスト	テキスト
macaddr8	テキスト	テキスト
bit(n)	long	バイト、ショート、整数、ロング (ビット数によって異なります)
bit varying(n)	long	バイト、ショート、整数、ロング (ビット数によって異なります)
json	オブジェクト	オブジェクト
JSONB	オブジェクト	オブジェクト
jsonpath	テキスト	テキスト

OpenSearch Ingestion パイプラインでデッドレターキュー (DLQ) を設定することをお勧めします。キューが設定済みである場合、OpenSearch Service は、動的マッピングの失敗により取り込めなかったすべての失敗したドキュメントをキューに送信します。

自動マッピングが失敗した場合は、パイプライン設定で `template_type` と `template_content` を使用して明示的なマッピングルールを定義できます。あるいは、パイプラインを開始する前に、検索ドメインまたはコレクションにマッピングテンプレートを直接作成することもできます。

制限事項

RDS for PostgreSQL 用に OpenSearch Ingestion パイプラインを設定する際には、次の制限を考慮してください。

- 統合は、パイプラインごとに 1 つの PostgreSQL データベースのみをサポートします。
- 統合は現在、クロスリージョンデータインジェストをサポートしていません。Amazon RDS インスタンスと OpenSearch ドメインは同じ AWS リージョンに存在する必要があります。

- 統合は現在、クロスアカウントデータインジェストをサポートしていません。Amazon RDS インスタンスと OpenSearch Ingestion パイプラインが同じ AWS アカウントにある必要があります。
- Amazon RDS インスタンスで AWS Secrets Manager、サポートされている唯一の認証メカニズムである を使用して認証が有効になっていることを確認します。
- 既存のパイプライン設定を更新して、別のデータベースや別のテーブルからデータを取り込むことはできません。パイプラインのデータベースやテーブル名を更新するには、パイプラインを停止して、更新された設定で再起動するか、新しいパイプラインを作成する必要があります。
- データ定義言語 (DDL) ステートメントは一般的にサポートされていません。以下の場合、データ整合性は維持されません。
  - プライマリキーが変更されます (追加/削除/名前変更)。
  - テーブルは削除/切り捨てられます。
  - 列名またはデータ型が変更されました。
- 同期する PostgreSQL テーブルにプライマリキーが定義されていない場合、データ整合性は保証されません。更新/削除を OpenSearch に同期できるようにするには、OpenSearch とシンク設定でカスタム `document_id` オプションを適切に定義する必要があります。
- RDS マルチ AZ DB クラスターはサポートされていません。
- サポートされているバージョン: PostgreSQL 16 以降。

推奨される CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント数、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメントの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

メトリクス	説明
pipeline-name .rds.credentialsChanged	このメトリクスは、 AWS シークレットがローテーションされる頻度を示します。
pipeline-name .rds.executorRefreshErrors	このメトリクスは、 AWS シークレットの更新に失敗したことを示します。

メトリクス	説明
pipeline-name .rds.exportRecordsTotal	このメトリクスは、Amazon Aurora からエクスポートされたレコードの数を示します。
pipeline-name .rds.exportRecordsProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたレコードの数を示します。
pipeline-name .rds.exportRecordProcessingErrors	このメトリクスは、Amazon Aurora クラスターからデータを読み取る際の OpenSearch Ingestion パイプラインの処理エラーの数を示します。
pipeline-name .rds.exportRecordsSuccessTotal	このメトリクスは、正常に処理されたエクスポートレコードの合計数を示します。
pipeline-name .rds.exportRecordsFailedTotal	このメトリクスは、処理に失敗したエクスポートレコードの合計数を示します。
pipeline-name .rds.bytesReceived	このメトリクスは、OpenSearch Ingestion パイプラインによって受信された合計バイト数を示します。
pipeline-name .rds.bytesProcessed	このメトリクスは、OpenSearch Ingestion パイプラインによって処理されたバイトの合計数を示します。
pipeline-name .rds.streamRecordsSuccessTotal	このメトリクスは、ストリームから正常に処理されたレコードの数を示します。

メトリクス	説明
pipeline-name .rds.streamRecordsFailedTotal	このメトリクスは、ストリームから処理に失敗したレコードの合計数を示します。

複数のテーブルからのデータの結合

非正規化結合を設定して、複数の関連テーブルのデータを 1 つの OpenSearch ドキュメントに自動的にマージできます。この機能は、親テーブルと子テーブル間の one-to-one 1 および one-to-many の関係をサポートします。パイプラインは、設定されたすべてのテーブルから変更データキャプチャ (CDC) イベントを読み取り、親ドキュメントにマージします。

結合設定は、Amazon RDS ソースと Aurora ソースの両方で同じです。設定の詳細と例については、「」を参照してくださいthe section called “複数のテーブルからのデータの結合”。

OpenSearch Ingestion パイプラインを Amazon S3 で使用する

OpenSearch Ingestion では、Amazon S3 をソースまたは送信先として使用できます。Amazon S3 をソースとして使用する場合は、OpenSearch Ingestion パイプラインにデータを送信します。Amazon S3 を送信先として使用する場合、OpenSearch Ingestion パイプラインから 1 つまたは複数の S3 バケットにデータを書き込みます。

トピック

- ソースとしての Amazon S3
- 送信先としての Amazon S3
- ソースとしての Amazon S3 クロスアカウント

ソースとしての Amazon S3

Amazon S3 をデータ処理のソースとして使用する方法には、S3-SQS 処理とスケジュールされたスキャンの 2 つがあります。

S3 にファイルが書き込まれた後にほぼリアルタイムでファイルをスキャンする必要がある場合は、S3-SQS 処理を使用します。バケット内でオブジェクトが保存または変更されるたびにイベントを発生させるように Amazon S3 バケットを設定できます。S3 バケット内のデータをバッチ処理す

るには、1 回限りのスケジュールされたスキャンまたは反復的なスケジュールされたスキャンを使用します。

トピック

- 前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する

前提条件

スケジュールされたスキャンと S3-SQS 処理の両方の OpenSearch Ingestion パイプラインのソースとして Amazon S3 を使用するには、最初にSS3 バケットを作成します。

> **Note**
>
> OpenSearch Ingestion パイプラインのソースとして使用される S3 バケットが異なる にある場合は AWS アカウント、バケットでクロスアカウント読み取りアクセス許可を有効にする必要もあります。これにより、パイプラインはデータを読み取って処理できるようになります。クロスアカウントアクセス許可を有効にするには、「Amazon S3 User Guide」の「Bucket owner granting cross-account bucket permissions」を参照してください。
> S3 バケットが複数のアカウントにある場合は、`bucket_owners` マップを使用します。例については、OpenSearch ドキュメントの「クロスアカウント S3 アクセス」を参照してください。

S3-SQS 処理をセットアップするには、以下のステップも実行する必要があります。

1. Amazon SQS キューを作成します。
2. SQS キューを送信先とする S3 バケットでイベント通知を有効にします。

ステップ 1: パイプラインロールを設定する

パイプラインにデータをプッシュする他のソースプラグインとは異なり、S3 ソースプラグインは、パイプラインがソースからデータを取得する読み取りベースのアーキテクチャを使用しています。

したがって、パイプラインが S3 から読み取るには、パイプラインの S3 ソース設定で、S3 バケットと Amazon SQS キューの両方にアクセスできるロールを指定する必要があります。キューからデータを読み取るために、パイプラインはこのロールを引き受けます。

> **Note**
>
> S3 ソース設定内で指定するロールは、パイプラインロールである必要があります。したがって、パイプラインロールには 2 つの異なるアクセス許可ポリシーを含める必要があります。1 つはシンクに書き込むポリシーで、もう 1 つは S3 ソースから取得するポリシーです。すべてのパイプラインコンポーネントで同じ `sts_role_arn` を使用する必要があります。

次のサンプルポリシーは、S3 をソースとして使用するために必要なアクセス許可を示しています。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action":[
          "s3:ListBucket",
          "s3:GetBucketLocation",
          "s3:GetObject"
       ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
       "Effect":"Allow",
       "Action":"s3:ListAllMyBuckets",
       "Resource":"arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage",
        "sqs:ChangeMessageVisibility"
      ],
      "Resource": "arn:aws:sqs:us-east-1:111122223333:MyS3EventSqsQueue"
    }
  ]
}
```

S3 ソースプラグイン設定の sts_role_arn オプションで指定した IAM ロールに、これらのアクセス許可をアタッチする必要があります。

```
version: "2"
source:
  s3:
    ...
    aws:
      ...
processor:
  ...
sink:
  - opensearch:
      ...
```

ステップ 2: パイプラインを作成する

アクセス許可を設定したら、Amazon S3 のユースケースに応じて OpenSearch Ingestion パイプラインを設定できます。

S3-SQS 処理

S3-SQS 処理をセットアップするには、パイプラインを設定して S3 をソースとして指定し、Amazon SQS 通知を設定します。

```
version: "2"
s3-pipeline:
  source:
    s3:
      notification_type: "sqs"
      codec:
        newline: null
      sqs:
        queue_url: "https://sqs.us-east-1amazonaws.com/account-id/ingestion-queue"
      compression: "none"
      aws:
        region: "region"
  processor:
  - grok:
      match:
        message:
        - "%{COMMONAPACHELOG}"
  - date:
```

```
      destination: "@timestamp"
      from_time_received: true
  sink:
  - opensearch:
      hosts: ["https://search-domain-endpoint.us-east-1es.amazonaws.com"]
      index: "index-name"
      aws:
        region: "region"
```

Amazon S3 で小さなファイルを処理するときに CPU 使用率が低い場合は、workers オプションの値を変更してスループットを高めることを検討してください。詳細については、「S3 plugin configuration options」を参照してください。

スケジュールされたスキャン

スケジュールされたスキャンをセットアップするには、すべての S3 バケットまたはバケットレベルに適用されるスキャンレベルで、パイプラインにスケジュールを設定します。バケットレベルのスケジュールまたはスキャン間隔の設定は、常にスキャンレベルの設定を上書きします。

スケジュールれたスキャンは、データ移行に理想的な 1 回限りのスキャン、またはバッチ処理に理想的な反復的スキャンで設定できます。

Amazon S3 から読み取るようにパイプラインを設定するには、事前設定された Amazon S3 ブループリントを使用します。パイプライン設定の scan の部分は、スケジュールのニーズに合わせて編集できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

1 回限りのスキャン

1 回限りのスケジュールされたスキャンは 1 回実行されます。パイプライン設定で、start_time および end_time を使用して、バケット内のオブジェクトをスキャンするタイミングを指定できます。range を使用して、バケット内のオブジェクトをスキャンする現在の時刻に相対的な間隔を指定できます。

例えば、範囲を PT4H に設定すると、過去 4 時間に作成されたすべてのファイルがスキャンされます。1 回限りのスキャンを 2 回目に実行するように設定するには、パイプラインを停止して再起動する必要があります。範囲を設定しない場合、開始時間と終了時間も更新する必要があります。

次の設定では、すべてのバケット、およびそれらのバケット内のすべてのオブジェクトを 1 回だけスキャンするようにセットアップされます。

```
version: "2"
log-pipeline:
```

```
source:
  s3:
    codec:
      csv:
    compression: "none"
    aws:
      region: "region"
    acknowledgments: true
    scan:
      buckets:
        - bucket:
            name: my-bucket
            filter:
              include_prefix:
                - Objects1/
              exclude_suffix:
                - .jpeg
                - .png
        - bucket:
            name: my-bucket-2
            key_prefix:
              include:
                - Objects2/
              exclude_suffix:
                - .jpeg
                - .png
    delete_s3_objects_on_read: false
processor:
  - date:
      destination: "@timestamp"
      from_time_received: true
sink:
  - opensearch:
      hosts: ["https://search-domain-endpoint.us-east-1es.amazonaws.com"]
      index: "index-name"
      aws:
        region: "region"
      dlq:
        s3:
          bucket: "dlq-bucket"
          region: "us-east-1"
```

次の設定では、指定した時間ウィンドウですべてのバケットを 1 回だけスキャンするようにセットアップされます。したがって、S3 は、作成時間がこのウィンドウ内に収まるオブジェクトのみを処理します。

```
scan:
  start_time: 2023-01-21T18:00:00.000Z
  end_time: 2023-04-21T18:00:00.000Z
  buckets:
    - bucket:
        name: my-bucket-1
        filter:
          include:
            - Objects1/
          exclude_suffix:
            - .jpeg
            - .png
    - bucket:
        name: my-bucket-2
        filter:
          include:
            - Objects2/
          exclude_suffix:
            - .jpeg
            - .png
```

次の設定では、スキャンレベルとバケットレベルの両方で 1 回限りのスキャンがセットアップされます。バケットレベルの開始時間と終了時間は、スキャンレベルの開始時間と終了時間よりも優先されます。

```
scan:
  start_time: 2023-01-21T18:00:00.000Z
  end_time: 2023-04-21T18:00:00.000Z
  buckets:
    - bucket:
        start_time: 2023-01-21T18:00:00.000Z
        end_time: 2023-04-21T18:00:00.000Z
        name: my-bucket-1
        filter:
          include:
            - Objects1/
          exclude_suffix:
            - .jpeg
```

```
            - .png
    - bucket:
        start_time: 2023-01-21T18:00:00.000Z
        end_time: 2023-04-21T18:00:00.000Z
        name: my-bucket-2
        filter:
          include:
            - Objects2/
          exclude_suffix:
            - .jpeg
            - .png
```

パイプラインを停止すると、停止前にパイプラインによってスキャンされたオブジェクトの既存の参照が削除されます。1 つのスキャンパイプラインが停止すると、既にスキャンされている場合でも、すべてのオブジェクトが起動後に再スキャンされます。1 つのスキャンパイプラインを停止する必要がある場合は、パイプラインを再開する前に時間ウィンドウを変更することをお勧めします。

開始時刻と終了時刻でオブジェクトをフィルタリングする必要がある場合は、パイプラインの停止と開始が唯一のオプションです。開始時刻と終了時刻でフィルタリングする必要がない場合は、名前でオブジェクトをフィルタリングできます。名前で読み取ると、パイプラインを停止して開始する必要はありません。これを行うには、include_prefix と exclude_suffix を使用します。

反復的スキャン

反復的なスケジュールされたスキャンでは、指定した S3 バケットのスキャンがスケジュールされた間隔で定期的に実行されます。個別のバケットレベルの設定はサポートされていないため、これらの間隔はスキャンレベルでのみ設定できます。

パイプライン設定で、interval は反復的スキャンの頻度を 30 秒から 365 日の間で指定します。これらのスキャンのうちの最初のスキャンは、パイプラインの作成時に実行されます。count はスキャンインスタンスの合計数を定義します。

次の設定では、12 時間のスキャン間隔での反復的スキャンがセットアップされます。

```
scan:
  scheduling:
    interval: PT12H
    count: 4
  buckets:
    - bucket:
        name: my-bucket-1
        filter:
```

```
          include:
            - Objects1/
          exclude_suffix:
            - .jpeg
            - .png
    - bucket:
        name: my-bucket-2
        filter:
          include:
            - Objects2/
          exclude_suffix:
            - .jpeg
            - .png
```

送信先としての Amazon S3

OpenSearch Ingestion パイプラインから S3 バケットにデータを書き込むには、事前設定済み S3 ブループリントを使用して S3 シンクを含むパイプラインを作成します。このパイプラインは、選択したデータを OpenSearch シンクにルーティングし、同時にすべてのデータを S3 にアーカイブするために送信します。詳細については、「the section called “ブループリントの使用”」を参照してください。

S3 シンクを作成するとき、さまざまなシンクコーデックから優先フォーマットを指定できます。例えば、列形式でデータを書き込む場合は、Parquet コーデックまたは Avro コーデックを選択します。行ベースの形式の場合は、JSON または NDJSON を選択します。指定したスキーマで S3 にデータを書き込むには、Avro 形式を使用してシンクコーデック内にインラインスキーマを定義することもできます。

次の例では、S3 シンクのインラインスキーマが定義されます。

```
- s3:
  codec:
    parquet:
      schema: >
        {
           "type" : "record",
           "namespace" : "org.vpcFlowLog.examples",
           "name" : "VpcFlowLog",
           "fields" : [
             { "name" : "version", "type" : "string"},
             { "name" : "srcport", "type": "int"},
             { "name" : "dstport", "type": "int"},
```

```
            { "name" : "start", "type": "int"},
            { "name" : "end", "type": "int"},
            { "name" : "protocol", "type": "int"},
            { "name" : "packets", "type": "int"},
            { "name" : "bytes", "type": "int"},
            { "name" : "action", "type": "string"},
            { "name" : "logStatus", "type" : "string"}
          ]
        }
```

このスキーマを定義するとき、パイプラインがシンクに配信するさまざまなタイプのイベントに存在する可能性のあるすべてのキーのスーパーセットを指定します。

例えば、あるイベントでキーが欠落している可能性がある場合は、そのキーに null 値を付けてスキーマに追加します。null 値を宣言すると、(キーを持つイベントと持たないイベントがある) 不均一なデータをスキーマで処理できます。受信イベントにこれらのキーがある場合、その値はシンクに書き込まれます。

このスキーマ定義は、定義済みのキーのみをシンクに送信し、未定義のキーを受信イベントから削除するフィルターとして機能します。

include_keys と exclude_keys をシンクで使用して、他のシンクにルーティングされるデータをフィルタリングすることもできます。この 2 つのフィルターは相互に排他的であるため、スキーマでは一度に 1 つしか使用できません。また、ユーザー定義のスキーマと一緒に使用することもできません。

このようなフィルターでパイプラインを作成するには、事前設定済みのシンクフィルターのブループリントを使用します。詳細については、「the section called “ブループリントの使用”」を参照してください。

ソースとしての Amazon S3 クロスアカウント

OpenSearch Ingestion パイプラインがソースとして別のアカウントの S3 バケットにアクセスできるように、Amazon S3 のアカウント間でアクセスを許可できます。クロスアカウントアクセスを有効にするには、「Amazon S3 User Guide」の「Bucket owner granting cross-account bucket permissions」を参照してください。アクセスを付与したら、パイプラインロールに必要なアクセス許可があることを確認します。

次に、bucket_owners を使用して パイプラインを作成し、Amazon S3 バケットへのクロスアカウントアクセスをソースとして有効にできます。

```
s3-pipeline:
 source:
  s3:
   notification_type: "sqs"
   codec:
    csv:
     delimiter: ","
     quote_character: "\""
     detect_header: True
   sqs:
    queue_url: "https://sqs.ap-northeast-1.amazonaws.com/401447383613/test-s3-queue"
   bucket_owners:
    my-bucket-01: 123456789012
    my-bucket-02: 999999999999
   compression: "gzip"
```

OpenSearch Ingestion パイプラインを Amazon Security Lake で使用する

S3 ソースプラグインを使用して Amazon Security Lake から OpenSearch Ingestion パイプラインにデータを取り込むことができます。Security Lake は、 AWS 環境、オンプレミス環境、SaaS プロバイダーのセキュリティデータを専用のデータレイクに自動的に一元化します。Security Lake から OpenSearch Ingestion パイプラインにデータをレプリケートするサブスクリプションを作成できます。レプリケートされたデータは、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションに書き込まれます。

Security Lake から読み取るようにパイプラインを設定するには、事前設定済みの Security Lake ブループリントを使用します。ブループリントには、Security Lake から Open Cybersecurity Schema Framework (OCSF) Parquet ファイルを取り込むデフォルトの設定が含まれています。詳細については、「the section called “ブループリントの使用”」を参照してください。

トピック

- Amazon Security Lake をソースとして OpenSearch Ingestion パイプラインを使用する
- Amazon Security Lake をシンクとして使用する OpenSearch Ingestion パイプライン

Amazon Security Lake をソースとして OpenSearch Ingestion パイプラインを使用する

OpenSearch Ingestion パイプライン内で Amazon S3 ソースプラグインを使用して Amazon Security Lake からデータを取り込むことができます。Security Lake は、 AWS 環境、オンプレミスシステム、SaaS プロバイダのセキュリティデータを専用のデータレイクに自動的に一元化します。

Amazon Security Lake には、パイプライン内に次のメタデータ属性があります。

- `bucket_name`: セキュリティデータを保存するために Security Lake によって作成された Amazon S3 バケットの名前。
- `path_prefix`: Security Lake IAM ロールポリシーで定義されているカスタムソース名。
- `region`: Security Lake S3 バケット AWS リージョン がある 。
- `accountID`: Security Lake が有効になっている AWS アカウント ID。
- `sts_role_arn`: Security Lake で使用する IAM ロールの ARN。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

- Security Hub を有効にします。
- Security Lake でサブスクライバーを作成します。
  - パイプラインに取り込むソースを選択します。
  - サブスクライバー認証情報には、パイプラインを作成する AWS アカウント の ID を追加します。外部 ID には `OpenSearchIngestion-{accountid}` を指定します。
  - [データアクセスメソッド] には [S3] を選択します。
  - 通知の詳細には、SQS キューを選択します。

サブスクライバーを作成すると、Security Lake は 2 つのインライン許可ポリシーを自動的に作成します。1 つは S3 用、もう 1 つは SQS 用です。ポリシーの形式は `AmazonSecurityLake-amzn-s3-demo-bucket-S3` および `AmazonSecurityLake-AWS Demo-SQS` です。パイプラインがサブスクライバーソースにアクセスできるようにするには、必要なアクセス許可をパイプラインロールに関連付ける必要があります。

パイプラインロールを設定する

Security Lake が自動的に作成した 2 つのポリシーから必要なアクセス許可のみを組み合わせた新しいアクセス許可ポリシーを IAM で作成します。次のポリシー例は、OpenSearch Ingestion パイプラインが複数の Security Lake ソースからデータを読み取るのに必要な最小特権を示しています。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-security-data-lake-us-east-1-abcde/aws/
LAMBDA_EXECUTION/1.0/*",
            "arn:aws:s3:::aws-security-data-lake-us-east-1-abcde/aws/S3_DATA/1.0/
*",
            "arn:aws:s3:::aws-security-data-lake-us-east-1-abcde/aws/
VPC_FLOW/1.0/*",
            "arn:aws:s3:::aws-security-data-lake-us-east-1-abcde/aws/ROUTE53/1.0/
*",
            "arn:aws:s3:::aws-security-data-lake-us-east-1-abcde/aws/
SH_FINDINGS/1.0/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sqs:ReceiveMessage",
            "sqs:DeleteMessage"
         ],
         "Resource":[
            "arn:aws:sqs:us-east-1:111122223333:AmazonSecurityLake-abcde-Main-
Queue"
         ]
      }
   ]
}
```

> **Important**
>
> Security Lake はパイプラインロールポリシーを管理しません。Security Lake サブスクリプションにソースを追加またはそこからソースを削除する場合は、ポリシーを手動で更新する必要があります。Security Lake はログソースごとにパーティションを作成するため、パイプラインロールのアクセス許可を手動で追加または削除する必要があります。

sqs において、S3 ソースプラグイン設定内の sts_role_arn オプションで指定した IAM ロールに、これらの許可をアタッチする必要があります。

```
version: "2"
source:
  s3:
    ...
    sqs:
      queue_url: "https://sqs.us-east-1amazonaws.com/account-id/
AmazonSecurityLake-amzn-s3-demo-bucket-Main-Queue"
    aws:
      ...
processor:
  ...
sink:
  - opensearch:
      ...
```

パイプラインの作成

パイプラインロールにアクセス許可を追加したら、事前設定済みの Security Lake ブループリントを使用してパイプラインを作成します。詳細については、「the section called “ブループリントの使用”」を参照してください。

s3 ソース設定内の queue_url オプションを指定する必要があります (これは、読み取り元の Amazon SQS キューの URL です)。URL をフォーマットするには、サブスクライバー設定で [サブスクリプションエンドポイント] を探し、arn:aws: を https:// に変更します。例えば、https://sqs.*us-east-1*amazonaws.com/*account-id*/AmazonSecurityLake-*AWS Demo*-Main-Queue。

S3 ソース設定内で指定した sts_role_arn は、パイプラインロールの ARN でなければなりません。

Amazon Security Lake をシンクとして使用する OpenSearch Ingestion パイプライン

OpenSearch Ingestion の Amazon S3 シンクプラグインを使用して、サポートされているソースから Amazon Security Lake にデータを送信します。Security Lake は AWS、、オンプレミス環境、SaaS プロバイダーからセキュリティデータを収集し、専用のデータレイクに保存します。

Security Lake にログデータを書き込むようにパイプラインを設定するには、事前設定された [ファイアウォールトラフィックログ] のブループリントを使用します。ブループリントには、Amazon S3 バケットに保存されている raw セキュリティログやその他のデータを取得し、レコードを処理し、正規化するためのデフォルト設定が含まれています。次に、データを Open Cybersecurity Schema Framework (OCSF) にマッピングし、変換された OCSF 準拠のデータを Security Lake に送信します。

パイプラインには次のメタデータ属性があります。

- `bucket_name`: セキュリティデータを保存するために Security Lake によって作成された Amazon S3 バケットの名前。
- `path_prefix`: Security Lake IAM ロールポリシーで定義されているカスタムソース名。
- `region`: Security Lake S3 バケット AWS リージョン がある 。
- `accountID`: Security Lake が有効になっている AWS アカウント ID。
- `sts_role_arn`: Security Lake で使用する IAM ロールの ARN。

前提条件

Security Lake にデータを送信するパイプラインを作成する前に、次の手順を実行します。

- Amazon Security Lake を有効にして設定する: さまざまなソースのセキュリティデータを一元化するように Amazon Security Lake を設定します。手順については、「Enabling Security Lake using the console」を参照してください。

  ソースを選択するときは、[特定の AWS ソースを取り込む] を選択し、取り込む 1 つ以上のログソースとイベントソースを選択します。
- アクセス許可の設定: Security Lake にデータを書き込むために必要なアクセス許可を持つパイプラインロールを設定します。詳細については、「パイプラインロール」を参照してください。

パイプラインの作成

事前設定された Security Lake ブループリントを使用してパイプラインを作成します。詳細については、「ブループリントを使用したパイプラインの作成」を参照してください。

OpenSearch Ingestion パイプラインを Fluent Bit で使用する

この Fluent Bit 設定ファイルのサンプルは、Fluent Bit のログデータを OpenSearch Ingestion パイプラインに送信します。ログデータの取り込みの詳細については、Data Prepper ドキュメントの「ログ分析」を参照してください。

次の点に注意してください。

- host の値はパイプラインエンドポイントにする必要があります。例えば、*pipeline-endpoint*.*us-east-1*osis.amazonaws.com。
- aws_service の値は osis にする必要があります。
- aws_role_arn 値は、署名バージョン 4 AWS 認証のためにクライアントが引き受けて使用する IAM ロールの ARN です。

```
[INPUT]
  name                  tail
  refresh_interval      5
  path                  /var/log/test.log
  read_from_head        true

[OUTPUT]
  Name http
  Match *
  Host pipeline-endpoint.us-east-1osis.amazonaws.com
  Port 443
  URI /log/ingest
  Format json
  aws_auth true
  aws_region region
  aws_service osis
  aws_role_arn arn:aws:iam::account-id:role/ingestion-role
  Log_Level trace
  tls On
```

その後、HTTP をソースとする OpenSearch Ingestion パイプラインを次のように設定できます。

```
version: "2"
unaggregated-log-pipeline:
  source:
    http:
      path: "/log/ingest"
  processor:
    - grok:
        match:
          log:
            - "%{TIMESTAMP_ISO8601:timestamp} %{NOTSPACE:network_node}
 %{NOTSPACE:network_host} %{IPORHOST:source_ip}:%{NUMBER:source_port:int} ->
 %{IPORHOST:destination_ip}:%{NUMBER:destination_port:int} %{GREEDYDATA:details}"
    - grok:
        match:
          details:
            - "'%{NOTSPACE:http_method} %{NOTSPACE:http_uri}' %{NOTSPACE:protocol}"
            - "TLS%{NOTSPACE:tls_version} %{GREEDYDATA:encryption}"
            - "%{NUMBER:status_code:int} %{NUMBER:response_size:int}"
    - delete_entries:
        with_keys: ["details", "log"]

  sink:
    - opensearch:
        hosts: ["https://search-domain-endpoint.us-east-1es.amazonaws.com"]
        index: "index_name"
        index_type: custom
        bulk_size: 20
        aws:
          region: "region"
```

OpenSearch Ingestion パイプラインを Fluentd で使用する

Fluentd は、Fluent Bit などのさまざまな言語やサブプロジェクトに SDK を提供するオープンソースのデータ収集エコシステムです。この Fluentd 設定ファイルのサンプルは、Fluentd のログデータを OpenSearch Ingestion パイプラインに送信します。ログデータの取り込みの詳細については、Data Prepper ドキュメントの「ログ分析」を参照してください。

次の点に注意してください。

- endpoint の値はパイプラインエンドポイントにする必要があります。例えば、*pipeline-endpoint*.*us-east-1*osis.amazonaws.com/apache-log-pipeline/logs。
- aws_service の値は osis にする必要があります。

- aws_role_arn 値は、署名バージョン 4 AWS 認証のためにクライアントが引き受けて使用する IAM ロールの ARN です。

```
<source>
  @type tail
  path logs/sample.log
  path_key log
  tag apache
  <parse>
    @type none
  </parse>
</source>

<filter apache>
  @type record_transformer
  <record>
    log ${record["message"]}
  </record>
</filter>

<filter apache>
  @type record_transformer
  remove_keys message
</filter>

<match apache>
  @type http
  endpoint pipeline-endpoint.us-east-1osis.amazonaws.com/apache-log-pipeline/logs
  json_array true

  <auth>
    method aws_sigv4
    aws_service osis
    aws_region region
    aws_role_arn arn:aws:iam::account-id:role/ingestion-role
  </auth>

  <format>
    @type json
  </format>

  <buffer>
```

```
    flush_interval 1s
  </buffer>
</match>
```

その後、HTTP をソースとする OpenSearch Ingestion パイプラインを次のように設定できます。

```
version: "2"
apache-log-pipeline:
  source:
    http:
      path: "/${pipelineName}/logs"
  processor:
    - grok:
        match:
          log:
            - "%{TIMESTAMP_ISO8601:timestamp} %{NOTSPACE:network_node}
 %{NOTSPACE:network_host} %{IPORHOST:source_ip}:%{NUMBER:source_port:int} ->
 %{IPORHOST:destination_ip}:%{NUMBER:destination_port:int} %{GREEDYDATA:details}"
   sink:
    - opensearch:
        hosts: ["https://search-domain-endpoint.us-east-1es.amazonaws.com"]
        index: "index_name"
        aws_region: "region"
        aws_sigv4: true
```

機械学習オフラインバッチ推論での OpenSearch Ingestion パイプラインの使用

Amazon OpenSearch Ingestion (OSI) パイプラインは、機械学習 (ML) オフラインバッチ推論処理をサポートし、大量のデータを低コストで効率的に強化します。非同期で処理できる大規模なデータセットがある場合は、常にオフラインバッチ推論を使用してください。オフラインバッチ推論は、Amazon Bedrock モデルと SageMaker モデルで機能します。この機能は、OpenSearch Service 2.17 以降のドメインで OpenSearch Ingestion AWS リージョン をサポートするすべての で使用できます。

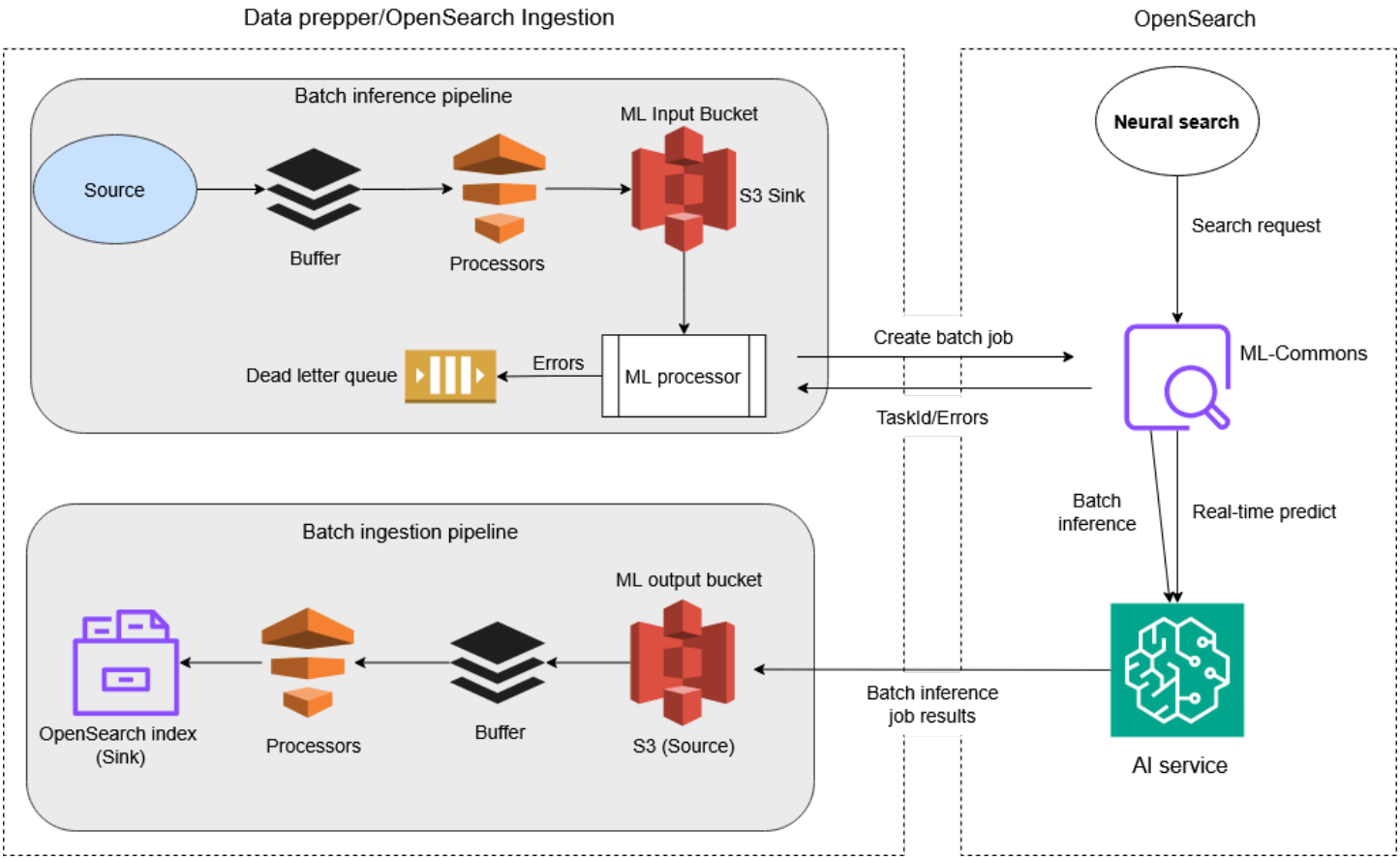
> **Note**
>
> リアルタイム推論処理には、サードパーティプラットフォーム用の Amazon OpenSearch Service ML コネクタ を使用します。

オフラインバッチ推論処理は、ML Commons と呼ばれる OpenSearch の機能を活用します。ML Commons は、トランスポートおよび REST API コールを通じて ML アルゴリズムを提供します。これらの呼び出しは、ML リクエストごとに適切なノードとリソースを選択し、ML タスクを監視してアップタイムを確保します。このように、ML Commons は既存のオープンソースの ML アルゴリズムを活用し、新しい ML 機能の開発に必要な労力を削減できます。ML Commons の詳細については、OpenSearch.org ドキュメントの「Machine learning」をご覧ください。

仕組み

パイプラインに adding a machine learning inference processor することで、OpenSearch Ingestion でオフラインバッチ推論パイプラインを作成できます。このプロセッサを使用すると、パイプラインを SageMaker などの AI サービスに接続してバッチ推論ジョブを実行できます。ターゲットドメインで実行されている AI コネクタ (batch_predict サポート) を介して目的の AI サービスに接続するようにプロセッサを設定できます。

OpenSearch Ingestion は、ML Commons で `ml_inference` プロセッサを使用してオフラインバッチ推論ジョブを作成します。次に、ML Commons は batch_predict API を使用します。この API は、Amazon Bedrock、Amazon SageMaker、Cohere、OpenAI の外部モデルサーバーにデプロイされたモデルを使用して、オフライン非同期モードで大規模なデータセットに対して推論を実行します。次の図は、複数のコンポーネントをオーケストレーションしてこのプロセスをエンドツーエンドで実行する OpenSearch Ingestion パイプラインを示しています。



パイプラインコンポーネントは次のように動作します。

パイプライン 1 (データの準備と変換)*:

- ソース: OpenSearch Ingestion でサポートされている外部ソースからデータがスキャンされます。
- データ処理者: raw データは処理され、統合された AI サービスのバッチ推論に適した形式に変換されます。
- S3 (シンク): 処理されたデータは、統合された AI サービスでバッチ推論ジョブを実行するための入力として機能する Amazon S3 バケットにステージングされます。

パイプライン 2 (トリガー ML batch_inference):

- ソース: パイプライン 1 の出力によって作成された新しいファイルの S3 イベント自動検出。
- Ml_inference プロセッサ: 非同期バッチジョブを介して ML 推論を生成するプロセッサ。ターゲットドメインで実行されている設定済みの AI コネクタを介して AI サービスに接続します。
- タスク ID: 各バッチジョブは、追跡と管理のために ml-commons のタスク ID に関連付けられます。

- OpenSearch ML Commons: ML Commons。リアルタイムニューラル検索用のモデルをホストし、リモート AI サーバーへのコネクタを管理し、バッチ推論とジョブ管理用の API を提供します。
- AI サービス: OpenSearch ML Commons は、Amazon Bedrock や Amazon SageMaker などの AI サービスとやり取りしてデータに対してバッチ推論を実行し、予測やインサイトを生成します。結果は別の S3 ファイルに非同期的に保存されます。

パイプライン 3 (バルク取り込み):

- S3 (ソース): バッチジョブの結果は、このパイプラインのソースである S3 に保存されます。
- データ変換プロセッサ: 取り込み前に、バッチ推論出力にさらに処理と変換が適用されます。これにより、データが OpenSearch インデックスに正しくマッピングされます。
- OpenSearch インデックス (シンク): 処理された結果は OpenSearch にインデックス化され、ストレージ、検索、およびさらなる分析が行われます。

> **Note**
>
> *パイプライン 1 で説明されているプロセスはオプションです。必要に応じて、そのプロセスをスキップし、準備したデータを S3 シンクにアップロードしてバッチジョブを作成できます。

ml_inference プロセッサについて

OpenSearch Ingestion は、バッチ処理に S3 スキャンソースと ML 推論プロセッサ間の特殊な統合を使用します。S3 スキャンはメタデータ専用モードで動作し、実際のファイルの内容を読み取らずに S3 ファイル情報を効率的に収集します。`ml_inference` プロセッサは S3 ファイル URL を使用して、バッチ処理のために ML Commons と調整します。この設計では、スキャンフェーズ中の不要なデータ転送を最小限に抑えることで、バッチ推論ワークフローを最適化します。パラメータを使用して `ml_inference` プロセッサを定義します。以下がその例です。

```
processor:
    - ml_inference:
        # The endpoint URL of your OpenSearch domain
        host: "https://AWS test-offlinebatch-123456789abcdefg.us-
west-2.es.amazonaws.com"

        # Type of inference operation:
```

```
        # - batch_predict: for batch processing
        # - predict: for real-time inference
        action_type: "batch_predict"

        # Remote ML model service provider (Amazon Bedrock or SageMaker)
        service_name: "bedrock"

        # Unique identifier for the ML model
        model_id: "AWS TestModelID123456789abcde"

        # S3 path where batch inference results will be stored
        output_path: "s3://amzn-s3-demo-bucket/"

        # Supports ISO_8601 notation strings like PT20.345S or PT15M
        # These settings control how long to keep your inputs in the processor for
retry on throttling errors
        retry_time_window: "PT9M"

        # AWS configuration settings
        aws:
            # AWS ##### where the Lambda function is deployed
            region: "us-west-2"
            # IAM role ARN for Lambda function execution
            sts_role_arn: "arn:aws::iam::account_id:role/Admin"

        # Dead-letter queue settings for storing errors
        dlq:
          s3:
            region: us-west-2
            bucket: batch-inference-dlq
            key_path_prefix: bedrock-dlq
            sts_role_arn: arn:aws:iam::account_id:role/OSI-invoke-ml

        # Conditional expression that determines when to trigger the processor
        # In this case, only process when bucket matches "amzn-s3-demo-bucket"
        ml_when: /bucket == "amzn-s3-demo-bucket"
```

ml_inference プロセッサを使用した取り込みパフォーマンスの向上

OpenSearch Ingestion `ml_inference` プロセッサは、ML 対応検索のデータインジェストパフォーマンスを大幅に強化します。プロセッサは、セマンティック検索、マルチモーダル検索、ドキュメントエンリッチメント、クエリ理解など、機械学習モデルによって生成されたデータを必要とするユー

スケースに最適です。セマンティック検索では、プロセッサは大量の高次元ベクトルの作成と取り込みを桁違いに高速化できます。

プロセッサのオフラインバッチ推論機能には、リアルタイムモデル呼び出しと比べて明確な利点があります。リアルタイム処理には容量制限のあるライブモデルサーバーが必要ですが、バッチ推論はコンピューティングリソースをオンデマンドで動的にスケーリングし、データを並行して処理します。例えば、OpenSearch Ingestion パイプラインが 10 億のソースデータリクエストを受信すると、ML バッチ推論入力用に 100 個の S3 ファイルが作成されます。次に、`ml_inference` プロセッサは 100 個の `ml.m4.xlarge` Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを使用して SageMaker バッチジョブを開始し、14 時間で 10 億件のリクエストのベクトル化を完了します。これは、リアルタイムモードで達成することが事実上不可能なタスクです。

セマンティック検索のデータリクエストを取り込むように ml_inference プロセッサを設定する

次の手順では、テキスト埋め込みモデルを使用してセマンティック検索用に 10 億のデータリクエストを取り込むように OpenSearch Ingestion `ml_inference` プロセッサを設定および設定するプロセスについて説明します。

トピック

- ステップ 1: OpenSearch でコネクタを作成し、モデルを登録する
- ステップ 2: ML オフラインバッチ推論用の OpenSearch Ingestion パイプラインを作成する
- ステップ 3: 取り込み用にデータを準備する
- ステップ 4: バッチ推論ジョブをモニタリングする
- ステップ 5: 検索を実行する

ステップ 1: OpenSearch でコネクタを作成し、モデルを登録する

次の手順では、ML Commons batch_inference_sagemaker_connector_blueprint を使用して、Amazon SageMaker でコネクタとモデルを作成します。OpenSearch CloudFormation 統合テンプレートを使用する場合は、このセクションの(代替手順) ステップ 1: CloudFormation 統合テンプレートを使用してコネクタとモデルを作成する後半の「」を参照してください。

OpenSearch でコネクタを作成し、モデルを登録するには

1. SageMaker でバッチ変換用の Deep Java Library (DJL) ML モデルを作成します。他の DJL モデルを表示するには、GitHub の「semantic_search_with_CFN_template_for_Sagemaker」を参照してください。

```
POST https://api.sagemaker.us-east-1.amazonaws.com/CreateModel
{
   "ExecutionRoleArn": "arn:aws:iam::123456789012:role/aos_ml_invoke_sagemaker",
   "ModelName": "DJL-Text-Embedding-Model-imageforjsonlines",
   "PrimaryContainer": {
      "Environment": {
         "SERVING_LOAD_MODELS" : "djl://ai.djl.huggingface.pytorch/sentence-
transformers/all-MiniLM-L6-v2"
      },
      "Image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/djl-inference:0.29.0-
cpu-full"
   }
}
```

2. actions フィールドで新しい action タイプとして batch_predict を使用してコネクタを作成します。

```
POST /_plugins/_ml/connectors/_create
{
  "name": "DJL Sagemaker Connector: all-MiniLM-L6-v2",
  "version": "1",
  "description": "The connector to sagemaker embedding model all-MiniLM-L6-v2",
  "protocol": "aws_sigv4",
  "credential": {
  "roleArn": "arn:aws:iam::111122223333:role/SageMakerRole"
},
  "parameters": {
    "region": "us-east-1",
    "service_name": "sagemaker",
    "DataProcessing": {
      "InputFilter": "$.text",
      "JoinSource": "Input",
      "OutputFilter": "$"
    },
    "MaxConcurrentTransforms": 100,
    "ModelName": "DJL-Text-Embedding-Model-imageforjsonlines",
    "TransformInput": {
      "ContentType": "application/json",
      "DataSource": {
        "S3DataSource": {
          "S3DataType": "S3Prefix",
          "S3Uri": "s3://offlinebatch/msmarcotests/"
```

```
        }
      },
      "SplitType": "Line"
    },
    "TransformJobName": "djl-batch-transform-1-billion",
    "TransformOutput": {
      "AssembleWith": "Line",
      "Accept": "application/json",
      "S3OutputPath": "s3://offlinebatch/msmarcotestsoutputs/"
    },
    "TransformResources": {
      "InstanceCount": 100,
      "InstanceType": "ml.m4.xlarge"
    },
    "BatchStrategy": "SingleRecord"
  },
  "actions": [
    {
      "action_type": "predict",
      "method": "POST",
      "headers": {
        "content-type": "application/json"
      },
      "url": "https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/
OpenSearch-sagemaker-060124023703/invocations",
      "request_body": "${parameters.input}",
      "pre_process_function": "connector.pre_process.default.embedding",
      "post_process_function": "connector.post_process.default.embedding"
    },
    {
      "action_type": "batch_predict",
      "method": "POST",
      "headers": {
        "content-type": "application/json"
      },
      "url": "https://api.sagemaker.us-east-1.amazonaws.com/CreateTransformJob",
      "request_body": """{ "BatchStrategy": "${parameters.BatchStrategy}",
 "ModelName": "${parameters.ModelName}", "DataProcessing" :
 ${parameters.DataProcessing}, "MaxConcurrentTransforms":
 ${parameters.MaxConcurrentTransforms}, "TransformInput":
 ${parameters.TransformInput}, "TransformJobName" :
 "${parameters.TransformJobName}", "TransformOutput" :
 ${parameters.TransformOutput}, "TransformResources" :
 ${parameters.TransformResources}}"""
```

```
    },
    {
      "action_type": "batch_predict_status",
      "method": "GET",
      "headers": {
        "content-type": "application/json"
      },
      "url": "https://api.sagemaker.us-east-1.amazonaws.com/DescribeTransformJob",
      "request_body": """{ "TransformJobName" :
 "${parameters.TransformJobName}"}"""
    },
    {
      "action_type": "cancel_batch_predict",
      "method": "POST",
      "headers": {
        "content-type": "application/json"
      },
      "url": "https://api.sagemaker.us-east-1.amazonaws.com/StopTransformJob",
      "request_body": """{ "TransformJobName" :
 "${parameters.TransformJobName}"}"""
    }
  ]
}
```

3. 返されたコネクタ ID を使用して SageMaker モデルを登録します。

```
POST /_plugins/_ml/models/_register
{
    "name": "SageMaker model for batch",
    "function_name": "remote",
    "description": "test model",
    "connector_id": "example123456789-abcde"
}
```

4. batch_predict アクションタイプを使用してモデルを呼び出します。

```
POST /_plugins/_ml/models/teHr3JABBiEvs-eod7sn/_batch_predict
{
  "parameters": {
    "TransformJobName": "SM-offline-batch-transform"
  }
}
```

レスポンスには、バッチジョブのタスク ID が含まれます。

```
{
 "task_id": "exampleIDabdcefd_1234567",
 "status": "CREATED"
}
```

5. タスク ID を使用して Get Task API を呼び出して、バッチジョブのステータスを確認します。

```
GET /_plugins/_ml/tasks/exampleIDabdcefd_1234567
```

レスポンスには、タスクのステータスが含まれます。

```
{
  "model_id": "nyWbv5EB_tT1A82ZCu-e",
  "task_type": "BATCH_PREDICTION",
  "function_name": "REMOTE",
  "state": "RUNNING",
  "input_type": "REMOTE",
  "worker_node": [
    "WDZnIMcbTrGtnR4Lq9jPDw"
  ],
  "create_time": 1725496527958,
  "last_update_time": 1725496527958,
  "is_async": false,
  "remote_job": {
    "TransformResources": {
      "InstanceCount": 1,
      "InstanceType": "ml.c5.xlarge"
    },
    "ModelName": "DJL-Text-Embedding-Model-imageforjsonlines",
    "TransformOutput": {
      "Accept": "application/json",
      "AssembleWith": "Line",
      "KmsKeyId": "",
      "S3OutputPath": "s3://offlinebatch/output"
    },
    "CreationTime": 1725496531.935,
    "TransformInput": {
      "CompressionType": "None",
      "ContentType": "application/json",
      "DataSource": {
```

```
        "S3DataSource": {
          "S3DataType": "S3Prefix",
          "S3Uri": "s3://offlinebatch/sagemaker_djl_batch_input.json"
        }
      },
      "SplitType": "Line"
    },
    "TransformJobArn": "arn:aws:sagemaker:us-east-1:111122223333:transform-job/SM-
offline-batch-transform15",
    "TransformJobStatus": "InProgress",
    "BatchStrategy": "SingleRecord",
    "TransformJobName": "SM-offline-batch-transform15",
    "DataProcessing": {
      "InputFilter": "$.content",
      "JoinSource": "Input",
      "OutputFilter": "$"
    }
  }
}
```

(代替手順) ステップ 1: CloudFormation 統合テンプレートを使用してコネクタとモデルを作成する

必要に応じて、 AWS CloudFormation を使用して、ML 推論に必要なすべての Amazon SageMaker コネクタとモデルを自動的に作成できます。このアプローチでは、Amazon OpenSearch Service コンソールで使用できる事前設定されたテンプレートを使用することで、セットアップを簡素化します。詳細については、「CloudFormation を使用してセマンティック検索用のリモート推論をセットアップする」を参照してください。

必要なすべての SageMaker コネクタとモデルを作成する CloudFormation スタックをデプロイするには

1. Amazon OpenSearch Service コンソールを開きます。
2. ナビゲーションペインで、[統合] を選択します。
3. 検索フィールドに **SageMaker** と入力し、[Amazon SageMaker を介したテキスト埋め込みモデルとの統合] を選択します。
4. [ドメインの設定] を選択し、[VPC ドメインの設定] または [パブリックドメインの設定] を選択します。
5. テンプレートフィールドに情報を入力します。[オフラインバッチ推論を有効化] で、オフラインバッチ処理用にリソースをプロビジョニングするには [true] を選択します。

6. Create を選択して CloudFormation スタックを作成します。
7. スタックを作成したら、 CloudFormation コンソールで [出力] タブを開き、connector_id と model_id を見つけます。これらの値は、後でパイプラインを設定するときに必要になります。

ステップ 2: ML オフラインバッチ推論用の OpenSearch Ingestion パイプラインを作成する

次のサンプルを使用して、ML オフラインバッチ推論用の OpenSearch Ingestion パイプラインを作成します。OpenSearch Ingestion のパイプラインの作成の詳細については、「Amazon OpenSearch Ingestion パイプラインの作成」を参照してください。

[開始する前に]

次の例では、`sts_role_arn` パラメータに IAM ロール ARN を指定します。次の手順を使用して、このロールが OpenSearch の ml-commons にアクセスできるバックエンドロールにマッピングされていることを確認します。

1. OpenSearch Service ドメインの OpenSearch Dashboards プラグインに移動します。OpenSearch Service コンソールのドメインダッシュボードに Dashboards エンドポイントがあります。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[ml_full_access] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. [バックエンドロール] で、ドメインを呼び出すアクセス許可を必要とする Lambda ロールの ARN を入力します。例: arn:aws:iam::*111122223333*:role/*lambda-role*
5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

ML オフラインバッチ推論用の OpenSearch Ingestion パイプラインを作成するためのサンプル

```
version: '2'
extension:
  osis_configuration_metadata:
    builder_type: visual
sagemaker-batch-job-pipeline:
  source:
    s3:
      acknowledgments: true
      delete_s3_objects_on_read: false
      scan:
```

```
        buckets:
          - bucket:
              name: name
              data_selection: metadata_only
              filter:
                include_prefix:
                  - sagemaker/sagemaker_djl_batch_input
                exclude_suffix:
                  - .manifest
          - bucket:
              name: name
              data_selection: data_only
              filter:
                include_prefix:
                  - sagemaker/output/
        scheduling:
          interval: PT6M
      aws:
        region: name
      default_bucket_owner: account_ID
      codec:
        ndjson:
          include_empty_objects: false
      compression: none
      workers: '1'
  processor:
    - ml_inference:
        host: "https://search-AWStest-offlinebatch-123456789abcdef.us-
west-2.es.amazonaws.com"
        aws_sigv4: true
        action_type: "batch_predict"
        service_name: "sagemaker"
        model_id: "model_ID"
        output_path: "s3://AWStest-offlinebatch/sagemaker/output"
        aws:
          region: "us-west-2"
          sts_role_arn: "arn:aws:iam::account_ID:role/Admin"
        ml_when: /bucket == "AWStest-offlinebatch"
        dlq:
          s3:
            region: us-west-2
            bucket: batch-inference-dlq
            key_path_prefix: bedrock-dlq
            sts_role_arn: arn:aws:iam::account_ID:role/OSI-invoke-ml
```

```
    - copy_values:
        entries:
          - from_key: /text
            to_key: chapter
          - from_key: /SageMakerOutput
            to_key: chapter_embedding
          - delete_entries:
            with_keys:
          - text
          - SageMakerOutput
  sink:
    - opensearch:
        hosts: ["https://search-AWStest-offlinebatch-123456789abcdef.us-
west-2.es.amazonaws.com"]
        aws:
          serverless: false
          region: us-west-2
        routes:
          - ml-ingest-route
        index_type: custom
        index: test-nlp-index
  routes:
    - ml-ingest-route: /chapter != null and /title != null
```

ステップ 3: 取り込み用にデータを準備する

ML オフラインバッチ推論処理用にデータを準備するには、独自のツールまたはプロセスを使用してデータを自分で準備するか、OpenSearch Data Prepper を使用します。パイプラインを使用してデータソースのデータを消費するか、機械学習データセットを作成して、データが正しい形式に整理されていることを確認します。

次の例では、自然言語処理タスクの実際のユーザークエリのコレクションを含む MS MARCO データセットを使用します。データセットは JSONL 形式で構造化され、各行は ML 埋め込みモデルに送信されたリクエストを表します。

```
{"_id": "1185869", "text": ")what was the immediate impact of the Paris Peace Treaties
 of 1947?", "metadata": {"world war 2"}}
{"_id": "1185868", "text": "_________ justice is designed to repair the harm to victim,
 the community and the offender caused by the offender criminal act. question 19
 options:", "metadata": {"law"}}
{"_id": "597651", "text": "what is amber", "metadata": {"nothing"}}
{"_id": "403613", "text": "is autoimmune hepatitis a bile acid synthesis disorder",
 "metadata": {"self immune"}}
```

```
...
```

MS MARCO データセットを使用してテストするには、それぞれ 1,000 万件のリクエストを含む 100 個のファイルに分散された 10 億件の入力リクエストを構築するシナリオを想像してください。ファイルは、プレフィックス s3://offlinebatch/sagemaker/sagemaker_djl_batch_input/ で Amazon S3 に保存されます。OpenSearch Ingestion パイプラインは、これら 100 個のファイルを同時にスキャンし、並列処理のために 100 人のワーカーで SageMaker バッチジョブを開始するため、10 億個のドキュメントを効率的にベクトル化して OpenSearch に取り込むことができます。

本番環境では、OpenSearch Ingestion パイプラインを使用して、バッチ推論入力用の S3 ファイルを生成できます。パイプラインはさまざまなデータソースをサポートし、ソースデータを S3 ファイルに継続的に変換するようにスケジュールに従って動作します。これらのファイルは、スケジュールされたオフラインバッチジョブを通じて AI サーバーによって自動的に処理され、継続的なデータ処理と取り込みが保証されます。

ステップ 4: バッチ推論ジョブをモニタリングする

バッチ推論ジョブは、SageMaker コンソールまたは AWS CLIを使用してモニタリングできます。Get Task API を使用してバッチジョブをモニタリングすることもできます。

```
GET /_plugins/_ml/tasks/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "state": "RUNNING"
          }
        }
      ]
    }
  },
  "_source": ["model_id", "state", "task_type", "create_time", "last_update_time"]
}
```

API はアクティブなバッチジョブタスクのリストを返します。

```
{
  "took": 2,
  "timed_out": false,
```

```
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 3,
      "relation": "eq"
    },
    "max_score": 0.0,
    "hits": [
      {
        "_index": ".plugins-ml-task",
        "_id": "nyWbv5EB_tT1A82ZCu-e",
        "_score": 0.0,
        "_source": {
          "model_id": "nyWbv5EB_tT1A82ZCu-e",
          "state": "RUNNING",
          "task_type": "BATCH_PREDICTION",
          "create_time": 1725496527958,
          "last_update_time": 1725496527958
        }
      },
      {
        "_index": ".plugins-ml-task",
        "_id": "miKbv5EB_tT1A82ZCu-f",
        "_score": 0.0,
        "_source": {
          "model_id": "miKbv5EB_tT1A82ZCu-f",
          "state": "RUNNING",
          "task_type": "BATCH_PREDICTION",
          "create_time": 1725496528123,
          "last_update_time": 1725496528123
        }
      },
      {
        "_index": ".plugins-ml-task",
        "_id": "kiLbv5EB_tT1A82ZCu-g",
        "_score": 0.0,
        "_source": {
          "model_id": "kiLbv5EB_tT1A82ZCu-g",
          "state": "RUNNING",
```

```
          "task_type": "BATCH_PREDICTION",
          "create_time": 1725496529456,
          "last_update_time": 1725496529456
        }
      }
    ]
  }
}
```

ステップ 5: 検索を実行する

バッチ推論ジョブをモニタリングして完了を確認したら、セマンティック、ハイブリッド、会話型 (RAG を使用)、ニューラルスパース、マルチモーダルなど、さまざまなタイプの AI 検索を実行できます。OpenSearch Service でサポートされている AI 検索の詳細については、「AI search」を参照してください。

raw ベクトルを検索するには、knn クエリタイプを使用し、vector 配列を入力として指定し、返される結果 k の数を指定します。

```
GET /my-raw-vector-index/_search
{
  "query": {
    "knn": {
      "my_vector": {
        "vector": [0.1, 0.2, 0.3],
        "k": 2
      }
    }
  }
}
```

AI を活用した検索を実行するには、neural クエリタイプを使用します。query_text 入力、OpenSearch Ingestion パイプラインで設定した埋め込みモデルの model_id、および返された結果の k 数を指定します。埋め込みを検索結果から除外するには、_source.excludes パラメータで埋め込みフィールドの名前を指定します。

```
GET /my-ai-search-index/_search
{
  "_source": {
    "excludes": [
      "output_embedding"
    ]
```

```
  },
  "query": {
    "neural": {
      "output_embedding": {
        "query_text": "What is AI search?",
        "model_id": "mBGzipQB2gmRjlv_dOoB",
        "k": 2
      }
    }
  }
}
```

OpenSearch Ingestion パイプラインを OpenTelemetry コレクターで使用する

OpenTelemetry Collector を使用して、OpenSearch Ingestion パイプラインにログ、トレース、メトリクスを取り込むことができます。1 つのパイプラインを使用して、すべてのログ、トレース、メトリクスをドメインまたはコレクション上の異なるインデックスに取り込むことができます。パイプラインを使用して、ログ、トレース、またはメトリクスのみを個別に取り込むこともできます。

トピック

- 前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- クロスアカウント接続
- 制限事項
- OpenTelemetry ソースに推奨される CloudWatch アラーム

前提条件

OpenTelemetry 設定ファイルをセットアップするときに、取り込みを行うには、以下を設定する必要があります。

- 取り込みロールには、パイプラインを操作するための`osis:Ingest`アクセス許可が必要です。詳細については、「取り込みロール」を参照してください。
- エンドポイント値にはパイプラインエンドポイントを含める必要があります。例: `https://pipeline-endpoint.us-east-1.osis.amazonaws.com`.

- サービス値は である必要がありますosis。
- OTLP/HTTP Exporter の圧縮オプションは、パイプラインが選択したソースの圧縮オプションと一致する必要があります。

```
extensions:
    sigv4auth:
        region: "region"
        service: "osis"

exporters:
    otlphttp:
        logs_endpoint: "https://pipeline-endpoint.us-east-1.osis.amazonaws.com/v1/logs"
        metrics_endpoint: "https://pipeline-endpoint.us-east-1.osis.amazonaws.com/v1/
metrics"
        traces_endpoint: "https://pipeline-endpoint.us-east-1.osis.amazonaws.com/v1/
traces"
        auth:
            authenticator: sigv4auth
        compression: none

service:
    extensions: [sigv4auth]
    pipelines:
        traces:
        receivers: [jaeger]
        exporters: [otlphttp]
```

ステップ 1: パイプラインロールを設定する

OpenTelemetry コレクター設定をセットアップしたら、パイプライン設定で使用するパイプラインロールを設定します。パイプラインロールが OTLP ソースに必要な特定のアクセス許可はなく、OpenSearch ドメインまたはコレクションへのアクセス権をパイプラインに付与するアクセス許可のみです。

ステップ 2: パイプラインを作成する

その後、ソースとして OTLP を指定する OpenSearch Ingestion パイプラインを次のように設定できます。OpenTelemetry ログ、メトリクス、トレースを個々のソースとして設定することもできます。

OTLP ソースパイプライン設定:

```
version: 2
otlp-pipeline:
    source:
        otlp:
            logs_path: /otlp-pipeline/v1/logs
            traces_path: /otlp-pipeline/v1/traces
            metrics_path: /otlp-pipeline/v1/metrics
    sink:
        - opensearch:
            hosts: ["https://search-mydomain.region.es.amazonaws.com"]
            index: "ss4o_metrics-otel-%{yyyy.MM.dd}"
            index_type: custom
            aws:
                region: "region"
```

OpenTelemetry Logs パイプライン設定:

```
version: 2
otel-logs-pipeline:
  source:
    otel_logs_source:
        path: /otel-logs-pipeline/v1/logs
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.region.es.amazonaws.com"]
        index: "ss4o_metrics-otel-%{yyyy.MM.dd}"
        index_type: custom
        aws:
            region: "region"
```

OpenTelemetry Metrics パイプライン設定:

```
version: 2
otel-metrics-pipeline:
  source:
    otel_metrics_source:
        path: /otel-metrics-pipeline/v1/metrics
  sink:
```

```
    - opensearch:
        hosts: ["https://search-mydomain.region.es.amazonaws.com"]
        index: "ss4o_metrics-otel-%{yyyy.MM.dd}"
        index_type: custom
        aws:
            region: "region"
```

OpenTelemetry Traces パイプライン設定:

```
version: 2
otel-trace-pipeline:
  source:
    otel_trace_source:
        path: /otel-traces-pipeline/v1/traces
  sink:
    - opensearch:
        hosts: ["https://search-mydomain.region.es.amazonaws.com"]
        index: "ss4o_metrics-otel-%{yyyy.MM.dd}"
        index_type: custom
        aws:
            region: "region"
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

クロスアカウント接続

OpenTelemetry ソースを使用した OpenSearch OpenSearch Ingestion パイプラインには、クロスアカウント取り込み機能があります。Amazon OpenSearch Ingestion を使用すると、仮想プライベートクラウド (VPC) AWS アカウント から別の VPC 内のパイプラインエンドポイントにパイプラインを共有できます。詳細については、「the section called “クロスアカウント取り込み用のパイプラインの設定”」を参照してください。

制限事項

OpenSearch Ingestion パイプラインは、20 mb を超えるリクエストを受信できません。この値は、`max_request_length`オプションでユーザーによって設定されます。このオプションはデフォルトで 10mb です。

OpenTelemetry ソースに推奨される CloudWatch アラーム

取り込みパイプラインのパフォーマンスをモニタリングするには、次の CloudWatch メトリクスをお勧めします。これらのメトリクスは、エクスポートから処理されたデータ量、ストリームから処理されたイベント量、エクスポートとストリームイベントの処理エラー、宛先に書き込まれたドキュメントの数を特定するのに役立ちます。これらのメトリクスの 1 つが、指定された時間にわたって指定された値を超えた場合にアクションを実行するように CloudWatch アラームを設定できます。

OTLP ソースの CloudWatch メトリクスは の形式です`{pipeline-name}.otlp.{logs | traces | metrics}.{metric-name}`。例えば、`otel-pipeline.otlp.metrics.requestTimeouts.count`。

個々の OpenTelemetry ソースを使用する場合、メトリクスは の形式になります`{pipeline-name}.{source-name}.{metric-name}`。例えば、`trace-pipeline.otel_trace_source.requestTimeouts.count`。

3 つの OpenTelemetry データ型はすべて同じメトリクスを持ちますが、簡潔にするために、メトリクスは OTLP ソースログタイプのデータの以下の表にのみ表示されます。

メトリクス	説明
`otel-pipeline.BlockingBuffer.bufferUsage.value`	使用されているバッファの量を示します。
`otel-pipeline.otlp.logs.requestTimeouts.count`	タイムアウトしたリクエストの数。
`otel-pipeline.otlp.logs.requestsReceived.count`	OpenTelemetry Collector が受信したリクエストの数。
`otel-pipeline.otlp.logs.badRequests.count`	OpenTelemetry Collector が受信した不正な形式のリクエストの数。
`otel-pipeline.otlp.logs.requestsTooLarge.count`	OpenTelemetry Collector が受信した最大 20 mb を超えるリクエストの数。
`otel-pipeline.otlp.logs.internalServerError.count`	The number of HTTP 500 errors received from the OpenTelemetry Collector.

メトリクス	説明
otel-pipeline.opensearch.bulkBadRequestErrors.count	Count of errors during bulk requests due to malformed request.
otel-pipeline.opensearch.bulkRequestLatency.avg	Average latency for bulk write requests made to OpenSearch.
otel-pipeline.opensearch.bulkRequestNotFoundErrors.count	Number of bulk requests that failed because the target data could not be found.
otel-pipeline.opensearch.bulkRequestNumberOfRetries.count	Number of retries by OpenSearch Ingestion pipelines to write OpenSearch cluster.
otel-pipeline.opensearch.bulkRequestSizeBytes.sum	Total size in bytes of all bulk requests made to OpenSearch.
otel-pipeline.opensearch.documentErrors.count	Number of errors when sending documents to OpenSearch. The documents causing the errors witll be sent to DLQ.
otel-pipeline.opensearch.documentsSuccess.count	Number of documents successfully written to an OpenSearch cluster or collection.
otel-pipeline.opensearch.documentsSuccessFirstAttempt.count	Number of documents successfully indexed in OpenSearch on the first attempt.
otel-pipeline.opensearch.documentsVersionConflictErrors.count	Count of errors due to version conflicts in documents during processing.
otel-pipeline.opensearch.PipelineLatency.avg	Average latency of OpenSearch Ingestion pipeline to process the data by reading from the source to writing to the destination.
otel-pipeline.opensearch.PipelineLatency.max	Maximum latency of OpenSearch Ingestion pipeline to process the data by reading from the source to writing the destination.

メトリクス	説明
otel-pipeline.opensearch.recordsIn.count	Count of records successfully ingested into OpenSearch. This metric is essential for tracking the volume of data being processed and stored.
otel-pipeline.opensearch.s3.dlqS3RecordsFailed.count	Number of records that failed to write to DLQ.
otel-pipeline.opensearch.s3.dlqS3RecordsSuccess.count	Number of records that are written to DLQ.
otel-pipeline.opensearch.s3.dlqS3RequestLatency.count	Count of latency measurements for requests to the Amazon S3 dead-letter queue.
otel-pipeline.opensearch.s3.dlqS3RequestLatency.sum	Total latency for all requests to the Amazon S3 dead-letter queue
otel-pipeline.opensearch.s3.dlqS3RequestSizeBytes.sum	Total size in bytes of all requests made to the Amazon S3 dead-letter queue.
otel-pipeline.recordsProcessed.count	Total number of records processed in the pipeline, a key metric for overal throughput.
otel-pipeline.opensearch.bulkRequestInvalidInputErrors.count	Count of bulk request errors in OpenSearch due to invalid input, crucial for monitoring data quality and operational issues.

Amazon Managed Service for Prometheus での OpenSearch Ingestion パイプラインの使用

Amazon Managed Service for Prometheus を OpenSearch Ingestion パイプラインの送信先として使用して、メトリクスを時系列形式で保存できます。Prometheus シンクを使用すると、OpenTelemetry メトリクスやその他の時系列データをパイプラインから Amazon Managed Service for Prometheus ワークスペースに送信して、モニタリング、アラート、分析を行うことができます。

prometheus シンクプラグインを使用すると、OpenSearch Ingestion パイプラインは Prometheus リモート書き込みプロトコルを使用して Amazon Managed Service for Prometheus ワークスペースにメトリクスデータを書き込むことができます。この統合により、次のことが可能になります。

- Amazon Managed Service for Prometheus に時系列メトリクスデータを保存する
- Amazon Managed Service for Prometheus と Amazon Managed Grafana を使用してメトリクスをモニタリングおよびアラートする
- 複数の送信先にメトリクスを同時にルーティングする (OpenSearch や Amazon Managed Service for Prometheus など)
- 外部エージェントから OpenTelemetry メトリクスを処理するか、パイプライン内でメトリクスを生成する

トピック

- 前提条件
- ステップ 1: パイプラインロールを設定する
- ステップ 2: パイプラインを作成する
- モニタリングとトラブルシューティング
- 制限事項
- ベストプラクティス

前提条件

Prometheus シンクを設定する前に、以下があることを確認してください。

- Amazon Managed Service for Prometheus ワークスペース: AWS リージョン OpenSearch Ingestion パイプラインと同じ AWS アカウント および にワークスペースを作成します。手順については、「Amazon Managed Service for Prometheus ユーザーガイド」の「ワークスペースの作成」を参照してください。
- IAM アクセス許可: Amazon Managed Service for Prometheus に書き込むアクセス許可を持つ IAM ロールを設定します。詳細については、「the section called “ステップ 1: パイプラインロールを設定する”」を参照してください。

ステップ 1: パイプラインロールを設定する

Prometheus シンクは、認証のためのパイプラインロールの IAM アクセス許可を自動的に継承するため、シンク設定に追加のロール設定 (などsts_role_arn) は必要ありません。

次のサンプルポリシーは、Amazon Managed Service for Prometheus をシンクとして使用するために必要なアクセス許可を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AMPRemoteWrite",
      "Effect": "Allow",
      "Action": [
        "aps:RemoteWrite"
      ],
      "Resource": "arn:aws:aps:region:account-id:workspace/workspace-id"
    }
  ]
}
```

次のプレースホルダーを置き換えます。

- *region*: Your AWS リージョン (例: us-east-1)
- *account-id*: ID AWS アカウント
- *workspace-id*: Amazon Managed Service for Prometheus ワークスペース ID

これらのアクセス許可をパイプラインロールにアタッチする必要があります。

パイプラインロールに OpenSearch Ingestion が引き受けることができる信頼関係があることを確認します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "osis-pipelines.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ステップ 2: パイプラインを作成する

アクセス許可を設定したら、Amazon Managed Service for Prometheus をシンクとして使用するように OpenSearch Ingestion パイプラインを設定できます。

基本的な設定

次の例は、最小限の Prometheus シンク設定を示しています。

```
version: "2"
sink:
  - prometheus:
      url: "https://aps-workspaces.region.amazonaws.com/workspaces/workspace-id/api/v1/
remote_write"
      aws:
        region: "region"
```

Amazon Managed Service for Prometheus リモート書き込みエンドポイントである`prometheus`シンク設定内で `url`オプションを指定する必要があります。URL をフォーマットするには、Amazon Managed Service for Prometheus コンソールでワークスペース ID を見つけ、URL を次のように構築します。 `https://aps-workspaces.region.amazonaws.com/workspaces/workspace-id/api/v1/remote_write`

設定オプション

次のオプションを使用して、Prometheus シンクのバッチ処理とフラッシュの動作を設定します。

Prometheus シンク設定オプション

オプション	必須	型	説明
`max_events`	いいえ	整数	Prometheus にフラッシュする前に累積するイベントの最大数。デフォルトは `1000` です。
`max_request_size`	いいえ	バイト数	フラッシュ前のリクエストペイロードの最大サイズ。デフォルトは `1mb` です。

オプション	必須	型	説明
flush_interval	いいえ	時間	イベントをフラッシュするまでの最大待機時間。デフォルトは 10s です。最大許容値は です60s。

パイプラインの例

例 1: Amazon Managed Service for Prometheus への OpenTelemetry メトリクス

このパイプラインは、外部エージェントから OpenTelemetry メトリクスを受信し、Amazon Managed Service for Prometheus に書き込みます。

```
version: "2"
source:
  otel_metrics_source:
    path: "/v1/metrics"
    output_format: otel

sink:
  - prometheus:
      url: "https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
      aws:
        region: "us-east-1"
```

例 2: デュアルシンク - OpenSearch と Amazon Managed Service for Prometheus

このパイプラインは、OpenSearch と Amazon Managed Service for Prometheus の両方にメトリクスをルーティングします。

```
version: "2"
source:
  otel_metrics_source:
    path: "/v1/metrics"
    output_format: otel

sink:
  - opensearch:
      hosts:
        - "https://search-domain-endpoint.us-east-1.es.amazonaws.com"
```

```
      index: "metrics-%{yyyy.MM.dd}"
      aws:
        region: "us-east-1"
        sts_role_arn: "arn:aws:iam::123456789012:role/OSI-Pipeline-Role"

  - prometheus:
      url: "https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
      aws:
        region: "us-east-1"
```

例 3: フィルタリングを使用したメトリクス

このパイプラインは、Amazon Managed Service for Prometheus に送信する前にメトリクスをフィルタリングします。

```
version: "2"
source:
  otel_metrics_source:
    path: "/v1/metrics"
    output_format: otel

processor:
  - drop_events:
      drop_when: '/name != "http.server.duration" and /name != "http.client.duration"'

sink:
  - prometheus:
      url: "https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
      aws:
        region: "us-east-1"
```

事前設定された Amazon Managed Service for Prometheus ブループリントを使用して、これらのパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

Amazon Managed Service for Prometheus シンクを使用したパイプラインの作成

AWS コンソールの使用

1. OpenSearch Service コンソールに移動します。

2. 取り込みでパイプラインを選択します。
3. パイプラインの作成 を選択します。
4. ブループリントを使用してビルドを選択し、Amazon Prometheus ブループリントへの OpenTelemetry メトリクスを選択します。
5. パイプラインを設定します。
   - Amazon Managed Service for Prometheus ワークスペース ID を入力する
   - パイプラインロール ARN を指定する
   - 必要に応じてソースとプロセッサの設定を構成する
6. パイプラインを確認して作成します。

CLI AWS の使用

必要な設定でパイプライン設定ファイル (などamp-pipeline.yaml) を作成し、以下を実行します。

```
aws osis create-pipeline \
  --pipeline-name my-amp-pipeline \
  --min-units 2 \
  --max-units 4 \
  --pipeline-configuration-body file://amp-pipeline.yaml
```

の使用 AWS CloudFormation

```
Resources:
  MyAMPPipeline:
    Type: AWS::OSIS::Pipeline
    Properties:
      PipelineName: my-amp-pipeline
      MinUnits: 2
      MaxUnits: 4
      PipelineConfigurationBody: |
        version: "2"
        source:
          otel_metrics_source:
            path: "/v1/metrics"
            output_format: otel
        sink:
          - prometheus:
```

```
              url: "https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
              aws:
                region: "us-east-1"
```

モニタリングとトラブルシューティング

CloudWatch メトリクス

CloudWatch メトリクスを使用してパイプラインのパフォーマンスをモニタリングします。

- `DocumentsWritten`: Amazon Managed Service for Prometheus に正常に書き込まれたメトリクスの数
- `DocumentsWriteFailed`: 書き込みに失敗したメトリクスの数
- `RequestLatency`: リモート書き込みリクエストのレイテンシー

一般的な問題

問題: パイプラインが Amazon Managed Service for Prometheus への書き込みに失敗する

解決方法:

- URL のワークスペース ID とリージョンが正しいことを確認します。
- パイプラインロールに アクセス`aps:RemoteWrite`許可があることを確認する
- ワークスペースがサービスマネージド AWS KMS キーを使用していることを確認します。
- パイプラインとワークスペースが同じ にあることを確認する AWS アカウント

問題: 認証エラー

解決方法:

- 信頼関係で がパイプラインロールを引き受け`osis-pipelines.amazonaws.com`ることができることを確認する
- パイプラインロールに必要な`aps:RemoteWrite`アクセス許可があることを確認します。

問題: 高レイテンシーまたはスロットリング

解決方法:

- パイプライン容量ユニットを増やす
- プロセッサにバッチ処理を実装する
- Amazon Managed Service for Prometheus サービスクォータを確認する

制限事項

Amazon Managed Service for Prometheus の OpenSearch Ingestion パイプラインを設定するときは、次の制限を考慮してください。

- Amazon Managed Service for Prometheus ワークスペースでは、サービスマネージド AWS KMS キーを使用する必要があります AWS 。カスタマーマネージド AWS KMS キーは現在サポートされていません。
- パイプラインと Amazon Managed Service for Prometheus ワークスペースは同じ にある必要があります AWS アカウント。

ベストプラクティス

- 同じ IAM ロールを使用する: Prometheus シンクは自動的にパイプラインロールを使用します。他のシンクを使用している場合は、 `sts_role_arn`がパイプラインロールと同じであることを確認します。
- メトリクスのモニタリング: 失敗した書き込みと高レイテンシーの CloudWatch アラームを設定する
- フィルタリングの実装: Amazon Managed Service for Prometheus に送信する前に、プロセッサを使用して不要なメトリクスをフィルタリングする
- 適切なサイズの容量: 最小容量から開始し、メトリクスボリュームに基づいてスケールする
- ブループリントを使用する: 一般的なユースケースで事前設定されたブループリントを活用する

OpenSearch Ingestion パイプラインを Kafka で使用する

Kafka プラグインを使用して、セルフマネージド Kafka クラスターから Amazon OpenSearch Service ドメインおよび OpenSearch Serverless コレクションにデータをストリーミングできます。OpenSearch Ingestion は、パブリックまたはプライベート (VPC) ネットワークで設定された Kafka クラスターからの接続をサポートします。このトピックでは、相互 TLS (mTLS)、SASL/SCRAM、IAM などのネットワーク設定や認証方法の設定など、取り込みパイプラインをセットアップするための前提条件と手順について説明します。

パブリック Kafka クラスターからのデータの移行

OpenSearch Ingestion パイプラインを使用して、パブリックなセルフマネージド Kafka クラスターからデータを移行できます。つまり、ドメイン DNS 名をパブリックに解決できます。これを行うには、ソースとして セルフマネージド Kafka、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、セルフマネージド型ソースクラスターから AWSマネージド型送信先ドメインまたはコレクションへのストリーミングデータが処理されます。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. パブリックネットワーク設定でセルフマネージド Kafka クラスターを作成します。クラスターには、OpenSearch Service に取り込むデータが含まれている必要があります。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
3. でセルフマネージドクラスターの認証を設定します AWS Secrets Manager。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
```

```
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

Kafka パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションに書き込むアクセス許可と、Secrets Manager からシークレットを読み取るアクセス許可を追加します。

ステップ 2: パイプラインを作成する

そして、ソースとして Kafka を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース Confluent Kafka クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。Confluent スキーマレジストリを使用して Confluent スキーマを定義できます。

```
version: "2"
kafka-pipeline:
  source:
    kafka:
      encryption:
        type: "ssl"
      topics:
```

```
        - name: "topic-name"
          group_id: "group-id"
      bootstrap_servers:
        - "bootstrap-server.us-east-1.aws.private.confluent.cloud:9092"
      authentication:
        sasl:
          plain:
            username: ${aws_secrets:confluent-kafka-secret:username}
            password: ${aws_secrets:confluent-kafka-secret:password}
      schema:
        type: confluent
        registry_url: https://my-registry.us-east-1.aws.confluent.cloud
        api_key: "${{aws_secrets:schema-secret:schema_registry_api_key}}"
        api_secret: "${{aws_secrets:schema-secret:schema_registry_api_secret}}"
        basic_auth_credentials_source: "USER_INFO"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
      index: "confluent-index"
extension:
  aws:
    secrets:
      confluent-kafka-secret:
        secret_id: "my-kafka-secret"
        region: "us-east-1"
      schema-secret:
        secret_id: "my-self-managed-kafka-schema"
        region: "us-east-1"
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

VPC 内の Kafka クラスターからのデータの移行

OpenSearch Ingestion パイプラインを使用して、VPC で実行されているセルフマネージド Kafka クラスターからデータを移行することもできます。これを行うには、ソースとして セルフマネージド Kafka、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、セルフマネージド型ソースクラスターから AWS マネージド型送信先ドメインまたはコレクションへのストリーミングデータが処理されます。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. OpenSearch Service に取り込むデータを含む VPC ネットワーク設定を使用してセルフマネージド Kafka クラスターを作成します。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
3. でセルフマネージドクラスターの認証を設定します AWS Secrets Manager。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. セルフマネージド Kafka にアクセスできる VPC の ID を取得します。OpenSearch Ingestion で使用する VPC CIDR を選択します。

   > **Note**
   >
   > を使用してパイプライン AWS マネジメントコンソール を作成する場合は、セルフマネージド Kafka を使用するにはOpenSearch Ingestion パイプラインも VPC にアタッチする必要があります。これを行うには、[ネットワーク設定] セクションを見つけ、[VPC にアタッチ] チェックボックスをオンにして、提供されたデフォルトオプションのいずれかから CIDR を選択するか、独自のものを選択します。RFC 1918 のベストカレントプラクティスで定義されているように、プライベートアドレス空間から任意の CIDR を使用できます。
   >
   > カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion とセルフマネージド OpenSearch 間の IP アドレスの衝突を回避するには、セルフマネージド OpenSearch VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

5. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、そのロールに次の許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerReadAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
```

```
            ],
            "Resource": ["arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret-name"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals":
                    {
                        "aws:RequestTag/OSISManaged": "true"
                    }
```

```
            }
        }
    ]
}
```

OpenSearch Ingestion パイプラインの作成に使用する IAM ロールには、上記の Amazon EC2 アクセス許可を指定する必要があります。これは、パイプラインがこれらのアクセス許可を使用して VPC 内のネットワークインターフェイスを作成および削除するためです。パイプラインは、このネットワークインターフェイスを介してのみ Kafka クラスターにアクセスできます。

ステップ 2: パイプラインを作成する

そして、ソースとして Kafka を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース Confluent Kafka クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。Confluent スキーマレジストリを使用して Confluent スキーマを定義できます。

```
 version: "2"
kafka-pipeline:
  source:
    kafka:
      encryption:
        type: "ssl"
      topics:
        - name: "topic-name"
          group_id: "group-id"
      bootstrap_servers:
        - "bootstrap-server.us-east-1.aws.private.confluent.cloud:9092"
      authentication:
        sasl:
          plain:
            username: ${aws_secrets:confluent-kafka-secret:username}
            password: ${aws_secrets:confluent-kafka-secret:password}
      schema:
        type: confluent
```

```
        registry_url: https://my-registry.us-east-1.aws.confluent.cloud
        api_key: "${{aws_secrets:schema-secret:schema_registry_api_key}}"
        api_secret: "${{aws_secrets:schema-secret:schema_registry_api_secret}}"
        basic_auth_credentials_source: "USER_INFO"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
      index: "confluent-index"
extension:
  aws:
    secrets:
      confluent-kafka-secret:
        secret_id: "my-kafka-secret"
        region: "us-east-1"
      schema-secret:
        secret_id: "my-self-managed-kafka-schema"
        region: "us-east-1"
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

Amazon OpenSearch Ingestion を使用したセルフマネージド OpenSearch クラスターからのデータの移行

Amazon OpenSearch Ingestion パイプラインをセルフマネージド OpenSearch または Elasticsearch で使用して、Amazon OpenSearch Service ドメインと OpenSearch Serverless コレクションにデータを移行できます。OpenSearch Ingestion は、セルフマネージド OpenSearch と Elasticsearch からのデータの移行のためのパブリックネットワーク設定とプライベートネットワーク設定の両方をサポートしています。

パブリック OpenSearch クラスターからの移行

OpenSearch Ingestion パイプラインを使用して、セルフマネージド OpenSearch クラスターまたは Elasticsearch クラスターからパブリック設定でデータを移行できます。つまり、ドメイン DNS 名はパブリックに解決できます。これを行うには、ソースとして セルフマネージド OpenSearch または Elasticsearch、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、セルフマネージド型ソースクラスターから AWSマネージド型送信先ドメインまたはコレクションにデータが効果的に移行されます。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. 移行するデータを含むセルフマネージド OpenSearch または Elastisearch クラスターを作成し、パブリック DNS 名を設定します。手順については、OpenSearch ドキュメントの「Create a cluster」を参照してください。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
3. でセルフマネージドクラスターの認証を設定します AWS Secrets Manager。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/domain-name"
      ]
    }
  ]
```

```
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

OpenSearch パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションに書き込むアクセス許可と、Secrets Manager からシークレットを読み取るアクセス許可を追加します。

ステップ 2: パイプラインを作成する

そして、ソースに OpenSearch を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース OpenSearch または Elasticsearch クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。

```
version: "2"
opensearch-migration-pipeline:
  source:
    opensearch:
      acknowledgments: true
      host: [ "https://my-self-managed-cluster-name:9200" ]
      indices:
        include:
          - index_name_regex: "include-.*"
        exclude:
          - index_name_regex: '\..*'
      authentication:
        username: ${aws_secrets:secret:username}
        password: ${aws_secrets:secret:password}
        scheduling:
           interval: "PT2H"
```

```
            index_read_count: 3
            start_time: "2023-06-02T22:01:30.00Z"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
          #Uncomment the following lines if your destination is an OpenSearch
 Serverless collection
          #serverless: true
          # serverless_options:
          #     network_policy_name: "network-policy-name"
      index: "${getMetadata(\"opensearch-index\")}"
      document_id: "${getMetadata(\"opensearch-document_id\")}"
      enable_request_compression: true
      dlq:
        s3:
          bucket: "bucket-name"
          key_path_prefix: "apache-log-pipeline/logs/dlq"
          region: "us-east-1"
extension:
  aws:
    secrets:
      secret:
        secret_id: "my-opensearch-secret"
        region: "us-east-1"
        refresh_interval: PT1H
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

VPC 内の OpenSearch クラスターからのデータの移行

OpenSearch Ingestion パイプラインを使用して、VPC で実行されているセルフマネージド OpenSearch または Elasticsearch クラスターからデータを移行することもできます。これを行うには、ソースとして セルフマネージド OpenSearch または Elasticsearch、宛先として OpenSearch Service または OpenSearch Serverless を使用して OpenSearch Ingestion パイプラインを設定します。これにより、セルフマネージド型ソースクラスターから AWSマネージド型送信先ドメインまたはコレクションにデータが効果的に移行されます。

前提条件

OpenSearch Ingestion パイプラインを作成する前に、次の手順を実行します。

1. 移行するデータを含む VPC ネットワーク設定を使用して、セルフマネージド OpenSearch または Elastisearch クラスターを作成します。
2. データの移行先となる OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成します。詳細については、「Creating OpenSearch Service domains」および「Creating collections」を参照してください。
3. でセルフマネージドクラスターの認証を設定します AWS Secrets Manager。AWS Secrets Manager シークレットのローテーションの手順に従って、シークレットのローテーションを有効にします。
4. セルフマネージド OpenSearch または Elasticsearch にアクセスできる VPC の ID を取得します。OpenSearch Ingestion で使用する /24 VPC CIDR ブロックを選択します。

   > **Note**
   >
   > を使用してパイプライン AWS マネジメントコンソール を作成する場合は、セルフマネージド OpenSearch または Elasticsearch を使用するには、OpenSearch Ingestion パイプラインを VPC にアタッチする必要があります。これを行うには、[ソースネットワークオプション] セクションを見つけ、[VPC にアタッチ] チェックボックスをオンにして、提供されたデフォルトオプションのいずれかから CIDR を選択します。CIDR ブロックは /24 プレフィックスの長さを使用する必要があります。RFC 1918 のベストプラクティスで定義されているように、プライベートアドレス空間から任意の /24 CIDR を使用できます。カスタム CIDR を指定するには、ドロップダウンメニューから [その他] を選択します。OpenSearch Ingestion とセルフマネージド OpenSearch 間の IP アドレスの衝突を回避するには、セルフマネージド OpenSearch VPC の CIDR が OpenSearch Ingestion の CIDR と異なることを確認してください。

5. リソースベースのポリシーをドメインにアタッチするか、データアクセスポリシーをコレクションにアタッチします。これらのアクセスポリシーにより、OpenSearch Ingestion は自己管理型クラスターからドメインまたはコレクションにデータを書き込むことができます。

   次のドメインアクセスポリシーの例では、次の手順で作成するパイプラインロールに、ドメインへのデータの書き込みが許可されています。必ず独自の ARN で `resource` を更新してください。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/pipeline-role"
      },
      "Action": [
        "es:DescribeDomain",
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-east-1:111122223333:domain/example.com"
      ]
    }
  ]
}
```

コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「the section called “ロールとユーザーの設定”」を参照してください。

ステップ 1: パイプラインロールを設定する

パイプラインの前提条件を設定したら、パイプライン設定で使用するパイプラインロールを設定し、そのロールに次の許可を追加します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerReadAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": ["arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret-name"]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals":
                    {
                        "aws:RequestTag/OSISManaged": "true"
                    }
            }
        }
    ]
}
```

OpenSearch Ingestion パイプラインの作成に使用する IAM ロールには、上記の Amazon EC2 アクセス許可を指定する必要があります。これは、パイプラインがこれらのアクセス許可を使用して VPC 内のネットワークインターフェイスを作成および削除するためです。パイプラインは、このネットワークインターフェイスを介してのみ OpenSearch クラスターにアクセスできます。

ステップ 2: パイプラインを作成する

そして、ソースに OpenSearch を指定する OpenSearch Ingestion パイプラインを次のように設定できます。

複数の OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用すると、条件付きルーティングや、受信データを複数の OpenSearch Service ドメインに複製することができます。

ソース OpenSearch または Elasticsearch クラスターから OpenSearch Serverless VPC コレクションにデータを移行することもできます。パイプライン設定内にネットワークアクセスポリシーを指定していることを確認します。

```
version: "2"
opensearch-migration-pipeline:
  source:
    opensearch:
      acknowledgments: true
      host: [ "https://my-self-managed-cluster-name:9200" ]
      indices:
        include:
          - index_name_regex: "include-.*"
        exclude:
          - index_name_regex: '\..*'
      authentication:
        username: ${aws_secrets:secret:username}
        password: ${aws_secrets:secret:password}
        scheduling:
           interval: "PT2H"
           index_read_count: 3
           start_time: "2023-06-02T22:01:30.00Z"
  sink:
  - opensearch:
      hosts: ["https://search-mydomain.us-east-1.es.amazonaws.com"]
      aws:
          region: "us-east-1"
          #Uncomment the following lines if your destination is an OpenSearch
 Serverless collection
```

```
          #serverless: true
          # serverless_options:
          #     network_policy_name: "network-policy-name"
      index: "${getMetadata(\"opensearch-index\")}"
      document_id: "${getMetadata(\"opensearch-document_id\")}"
      enable_request_compression: true
      dlq:
        s3:
          bucket: "bucket-name"
          key_path_prefix: "apache-log-pipeline/logs/dlq"
          region: "us-east-1"
extension:
  aws:
    secrets:
      secret:
        secret_id: "my-opensearch-secret"
        region: "us-east-1"
        refresh_interval: PT1H
```

事前設定されたブループリントを使用して、このパイプラインを作成できます。詳細については、「the section called “ブループリントの使用”」を参照してください。

Amazon Kinesis Data Streams で OpenSearch Ingestion パイプラインを使用する

Amazon Kinesis Data Streams で OpenSearch Ingestion パイプラインを使用して、複数のストリームから Amazon OpenSearch Service ドメインとコレクションにストリームレコードデータを取り込みます。OpenSearch Ingestion パイプラインには、ストリーミング取り込みインフラストラクチャが組み込まれており、Kinesis からストリームレコードを継続的に取り込むための、高スケールで低レイテンシーな方法を提供します。

トピック

- Amazon Kinesis Data Streams をソースとする場合
- ソースとしての Amazon Kinesis Data Streams クロスアカウント

Amazon Kinesis Data Streams をソースとする場合

次の手順では、Amazon Kinesis Data Streams をデータソースとして使用する OpenSearch Ingestion パイプラインを設定する方法について説明します。このセクションでは、OpenSearch

Service ドメインまたは OpenSearch Serverless コレクションの作成、パイプラインロールの設定とパイプラインの作成の手順など、必要な前提条件について説明します。

前提条件

パイプラインを設定するには、1 つ以上のアクティブな Kinesis Data Streams が必要です。これらのストリームは、レコードを受信するか、他のソースからレコードを受信する準備ができている必要があります。詳細については、「Overview of OpenSearch Ingestion」を参照してください。

パイプラインを設定するには

1. OpenSearch Service ドメインまたは OpenSearch Serverless コレクションを作成する

   ドメインまたはコレクションを作成するには、「Getting started with OpenSearch Ingestion」を参照してください。

   コレクションまたはドメインへの書き込みデータにアクセスするための正しいアクセス許可を持つ IAM ロールを作成するには、「Resource-based policies」を参照してください。
2. アクセス許可を持つパイプラインロールを設定する

   パイプライン設定で使用するパイプラインロールを設定し、そのロールに次のアクセス許可を追加します。*********を、ユーザー自身の情報に置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowReadFromStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:DescribeStreamConsumer",
                "kinesis:DescribeStreamSummary",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards",
                "kinesis:ListStreams",
                "kinesis:ListStreamConsumers",
```

```
                "kinesis:RegisterStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream-name"
            ]
        }
    ]
}
```

ストリームでサーバー側の暗号化が有効になっている場合、次の AWS KMS ポリシーでは、レコードを復号化できるようにします。*#########*を、ユーザー自身の情報に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowDecryptionOfCustomManagedKey",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id"
        }
    ]
}
```

パイプラインがドメインにデータを書き込むには、sts_role_arn パイプラインロールにドメインへのアクセスを許可するドメインレベルのアクセスポリシーが、このドメインに必要になります。

次の例は、前のステップで作成した pipeline-role と言う名前のパイプラインロールに、ingestion-domain と言う名前のドメインへの、データの書き込みが許可されるドメインアクセスポリシーです。*#########*を、ユーザー自身の情報に置き換えます。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::your-account-id:role/pipeline-role"
      },
      "Action": ["es:DescribeDomain", "es:ESHttp*"],
      "Resource": "arn:aws:es:AWS #####:account-id:domain/domain-name/*"
    }
  ]
}
```

3. パイプラインの作成

Kinesis-data-streams をソースとして指定する OpenSearch 取り込みパイプラインを設定します。このようなパイプラインを作成するために、OpenSearch Ingestion Console で用意されたブループリントを見つけることができます。(オプション) を使用してパイプラインを作成するには AWS CLI、「」という名前のブループリントを使用できます**AWS-KinesisDataStreamsPipeline**。*#########*を、ユーザー自身の情報に置き換えます。

```
version: "2"
kinesis-pipeline:
  source:
    kinesis_data_streams:
      acknowledgments: true
      codec:
        # Based on whether kinesis records are aggregated or not, you could choose
 json, newline or ndjson codec for processing the records.
        # JSON codec supports parsing nested CloudWatch Events into individual log
 entries that will be written as documents into OpenSearch.
        # json:
          # key_name: "logEvents"
          # These keys contain the metadata sent by CloudWatch Subscription Filters
```

```
          # in addition to the individual log events:
          # include_keys: [ 'owner', 'logGroup', 'logStream' ]
        newline:
      streams:
        - stream_name: "stream name"
          # Enable this if ingestion should start from the start of the stream.
          # initial_position: "EARLIEST"
          # checkpoint_interval: "PT5M"
          # Compression will always be gzip for CloudWatch, but will vary for other
 sources:
          # compression: "gzip"
        - stream_name: "stream name"
          # Enable this if ingestion should start from the start of the stream.
          # initial_position: "EARLIEST"
          # checkpoint_interval: "PT5M"
          # Compression will always be gzip for CloudWatch, but will vary for other
 sources:
          # compression: "gzip"

        # buffer_timeout: "1s"
        # records_to_accumulate: 100
        # Change the consumer strategy to "polling". Default consumer strategy will
 use enhanced "fan-out" supported by KDS.
        # consumer_strategy: "polling"
        # if consumer strategy is set to "polling", enable the polling config
 below.
        # polling:
          # max_polling_records: 100
          # idle_time_between_reads: "250ms"
      aws:
        # Provide the Role ARN with access to Amazon Kinesis Data Streams. This
 role should have a trust relationship with osis-pipelines.amazonaws.com
        sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
        # Provide the AWS ##### of the Data Stream.
        region: "us-east-1"

  sink:
    - opensearch:
        # Provide an Amazon OpenSearch Serverless domain endpoint
        hosts: [ "https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-
east-1.es.amazonaws.com" ]
        index: "index_${getMetadata(\"stream_name\")}"
        # Ensure adding unique document id as a combination of the metadata
 attributes available.
```

```
        document_id: "${getMetadata(\"partition_key\")}_
${getMetadata(\"sequence_number\")}_${getMetadata(\"sub_sequence_number\")}"
        aws:
          # Provide a Role ARN with access to the domain. This role should have a
 trust relationship with osis-pipelines.amazonaws.com
          sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
          # Provide the AWS ##### of the domain.
          region: "us-east-1"
          # Enable the 'serverless' flag if the sink is an Amazon OpenSearch
 Serverless collection
          serverless: false
          # serverless_options:
            # Specify a name here to create or update network policy for the
 serverless collection
            # network_policy_name: "network-policy-name"
        # Enable the 'distribution_version' setting if the OpenSearch Serverless
 domain is of version Elasticsearch 6.x
        # distribution_version: "es6"
        # Enable and switch the 'enable_request_compression' flag if the default
 compression setting is changed in the domain. See https://docs.aws.amazon.com/
opensearch-service/latest/developerguide/gzip.html
        # enable_request_compression: true/false
        # Optional: Enable the S3 DLQ to capture any failed requests in an S3
 bucket. Delete this entire block if you don't want a DLQ.
        dlq:
          s3:
            # Provide an S3 bucket
            bucket: "your-dlq-bucket-name"
            # Provide a key path prefix for the failed requests
            # key_path_prefix: "kinesis-pipeline/logs/dlq"
            # Provide the region of the bucket.
            region: "us-east-1"
            # Provide a Role ARN with access to the bucket. This role should have a
 trust relationship with osis-pipelines.amazonaws.com
            sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
```

設定オプション

Kinesis 設定オプションについては、OpenSearch ドキュメントの「Configuration options」を参照してください。

使用可能なメタデータ属性

- stream_name – レコードが取り込まれた Kinesis Data Streams の名前
- partition_key – 取り込まれる Kinesis Data Streams レコードのパーティションキー
- sequence_number – 取り込まれる Kinesis Data Streams レコードのシーケンス番号
- sub_sequence_number – 取り込まれる Kinesis Data Streams レコードのサブシーケンス番号

4. (オプション) Kinesis Data Streams パイプラインの推奨コンピューティングユニット (OCU) を設定する

   OpenSearch Kinesis Data Streams ソースパイプラインは、複数のストリームからストリームレコードを取り込むように設定することもできます。取り込みを高速化するには、追加された新しいストリームごとにコンピューティングユニットを追加することをお勧めします。

データ整合性

OpenSearch Ingestion は、データの耐久性を確保するためにエンドツーエンドの確認応答をサポートしています。パイプラインが Kinesis からストリームレコードを読み取ると、ストリームに関連付けられたシャードに基づいてストリームレコードを読み取る作業が動的に分散されます。パイプラインは、OpenSearch ドメインまたはコレクション内のすべてのレコードを取り込んだ後に確認応答を受信すると、自動的にストリームのチェックポイントを作成します。これにより、ストリームレコードの重複処理を回避できます。

ストリーム名に基づいてインデックスを作成するには、opensearch sink セクションのインデックスを "index_${getMetadata(\"stream_name\")}" として定義します。

ソースとしての Amazon Kinesis Data Streams クロスアカウント

OpenSearch Ingestion パイプラインがソースとして別のアカウントの Kinesis Data Streams にアクセスできるように、Amazon Kinesis Data Streams のアカウント間でアクセスを許可できます。クロスアカウントアクセスを有効にするには、以下の手順に従います。

クロスアカウントアクセスの設定

1. Kinesis ストリームがあるアカウントにリソースポリシーを設定する

   *********を、ユーザー自身の情報に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "StreamReadStatementID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Pipeline-Role"
            },
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:444455556666:stream/
stream-name"
        },
        {
            "Sid": "StreamEFOReadStatementID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Pipeline-Role"
            },
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-
east-1:444455556666:stream/stream-name/consumer/consumer-name"
        }
    ]
}
```

2. (オプション) コンシューマーおよびコンシューマーリソースポリシーの設定

   これはオプションのステップであり、ストリームレコードの読み取りに拡張ファンアウトコンシューマー戦略を使用する場合にのみ必要です。詳しくは、「Develop enhanced fan-out consumers with dedicated throughput」を参照してください。

   a. コンシューマーのセットアップ

      既存のコンシューマーを再利用するには、このステップをスキップできます。詳細については、「Amazon Kinesis Data Streams API リファレンス」の「RegisterStreamConsumer」を参照してください。

      次の例に示す CLI コマンドでは、*########*を独自の情報に置き換えます。

      Example CLI コマンドの例:

```
aws kinesis register-stream-consumer \
--stream-arn "arn:aws:kinesis:AWS #####:account-id:stream/stream-name" \
--consumer-name consumer-name
```

   b. コンシューマーリソースポリシーの設定

      次のステートメントでは、*#########*を独自の情報に置き換えます。

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConsumerEFOReadStatementID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Pipeline-Role"
            },
            "Action": [
                "kinesis:DescribeStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
```

```
            "Resource": "arn:aws:kinesis:us-
east-1:444455556666:stream/stream-1/consumer/consumer-name"
        }
    ]
}
```

3. パイプラインの設定

   クロスアカウント取り込みの場合は、各ストリームの kinesis_data_streams に次の属性を追加します。

   - stream_arn – ストリームが存在するアカウントに属するストリームの ARN
   - consumer_arn – これはオプションの属性であり、デフォルトの拡張ファンアウトコンシューマー戦略が選択されている場合は指定する必要があります。このフィールドの実際のコンシューマー ARN を指定します。*#########*を、ユーザー自身の情報に置き換えます。

```
version: "2"
     kinesis-pipeline:
       source:
         kinesis_data_streams:
           acknowledgments: true
           codec:
             newline:
           streams:
             - stream_arn: "arn:aws:kinesis:region:stream-account-id:stream/stream-
name"
               consumer_arn: "consumer arn"
               # Enable this if ingestion should start from the start of the
 stream.
               # initial_position: "EARLIEST"
               # checkpoint_interval: "PT5M"
             - stream_arn: "arn:aws:kinesis:region:stream-account-id:stream/stream-
name"
               consumer_arn: "consumer arn"
                # initial_position: "EARLIEST"

             # buffer_timeout: "1s"
             # records_to_accumulate: 100
             # Enable the consumer strategy to "polling". Default consumer strategy
 will use enhanced "fan-out" supported by KDS.
```

```
            # consumer_strategy: "polling"
            # if consumer strategy is set to "polling", enable the polling config
 below.
            # polling:
              # max_polling_records: 100
              # idle_time_between_reads: "250ms"
          aws:
            # Provide the Role ARN with access to Kinesis. This role should have a
 trust relationship with osis-pipelines.amazonaws.com
            sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
            # Provide the AWS ##### of the domain.
            region: "us-east-1"

      sink:
        - opensearch:
            # Provide an OpenSearch Serverless domain endpoint
            hosts: [ "https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-
east-1.es.amazonaws.com" ]
            index: "index_${getMetadata(\"stream_name\")}"
            # Mapping for documentid based on partition key, shard sequence number
 and subsequence number metadata attributes
            document_id: "${getMetadata(\"partition_key\")}_
${getMetadata(\"sequence_number\")}_${getMetadata(\"sub_sequence_number\")}"
            aws:
              # Provide a Role ARN with access to the domain. This role should
 have a trust relationship with osis-pipelines.amazonaws.com
              sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
              # Provide the AWS ##### of the domain.
              region: "us-east-1"
              # Enable the 'serverless' flag if the sink is an OpenSearch
 Serverless collection
              serverless: false
                # serverless_options:
                # Specify a name here to create or update network policy for the
 serverless collection
              # network_policy_name: network-policy-name
            # Enable the 'distribution_version' setting if the OpenSearch
 Serverless domain is of version Elasticsearch 6.x
            # distribution_version: "es6"
            # Enable and switch the 'enable_request_compression' flag if
 the default compression setting is changed in the domain. See https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/gzip.html
            # enable_request_compression: true/false
```

```
            # Optional: Enable the S3 DLQ to capture any failed requests in an S3
bucket. Delete this entire block if you don't want a DLQ.
            dlq:
              s3:
                # Provide an Amazon S3 bucket
                bucket: "your-dlq-bucket-name"
                # Provide a key path prefix for the failed requests
                # key_path_prefix: "alb-access-log-pipeline/logs/dlq"
                # Provide the AWS ##### of the bucket.
                region: "us-east-1"
                # Provide a Role ARN with access to the bucket. This role should
have a trust relationship with osis-pipelines.amazonaws.com
                sts_role_arn: "arn:aws:iam::111122223333:role/Example-Role"
```

4. OSI パイプラインロール Kinesis Data Streams

   a. IAM ポリシー

   以下のポリシーをパイプラインロールに追加します。*#########*を、ユーザー自身の情報に置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
            "Resource": [
            "arn:aws:kinesis:us-east-1:111122223333:stream/my-stream"
            ]
        },
        {
            "Sid": "allowReadFromStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
```

```
                "kinesis:DescribeStreamSummary",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards",
                "kinesis:ListStreams",
                "kinesis:ListStreamConsumers",
                "kinesis:RegisterStreamConsumer"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/my-stream"
            ]
        }
    ]
}
```

b. 信頼ポリシー

ストリームアカウントからデータを取り込むには、パイプライン取り込みロールとストリームアカウントの間に信頼関係を確立する必要があります。パイプラインロールに以下を追加します。*#########*を、ユーザー自身の情報に置き換えます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111122223333:root"
      },
     "Action": "sts:AssumeRole"
  }]
}
```

次の手順

データをパイプラインにエクスポートすると、パイプラインのシンクとして設定されている OpenSearch Service ドメインからクエリを実行できます。次のリソースを参照してください。

- オブザーバビリティ
- the section called “トレースの検出”
- オブザーバビリティ

での OpenSearch Ingestion パイプラインの使用 AWS Lambda

AWS Lambda プロセッサを使用して、カスタムコードを使用して OpenSearch Ingestion でサポートされている任意のソースまたは送信先からのデータを強化します。Lambda プロセッサを使用すると、独自のデータ変換やエンリッチメントを適用し、処理済みイベントをパイプラインに戻してさらに処理できます。このプロセッサにより、カスタマイズされたデータ処理が可能になり、パイプラインを通過する前にデータがどのように操作されるかを完全に制御できます。

> **Note**
>
> Lambda プロセッサによって処理される単一のイベントのペイロードサイズ制限は 5 MB です。さらに、Lambda プロセッサは JSON 配列形式のレスポンスのみをサポートします。

前提条件

Lambda プロセッサを使用してパイプラインを作成する前に、次のリソースを作成します。

- ソースデータを強化および変換する AWS Lambda 関数。手順については、「最初の Lambda 関数を作成する」を参照してください。
- パイプラインシンクとなる OpenSearch Service ドメインまたは OpenSearch Serverless コレクション。詳細については、「the section called “OpenSearch Service ドメインの作成”」および「the section called “コレクションの作成”」を参照してください。
- ドメインまたはコレクションシンクに書き込むアクセス許可を含むパイプラインロール。詳細については、「the section called “パイプラインロール”」を参照してください。

  パイプラインロールには、パイプライン設定で指定された Lambda 関数を呼び出すことを許可する、アタッチされたアクセス許可ポリシーも必要です。例えば、次のようになります。

  JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "allowinvokeFunction",
            "Effect": "Allow",
            "Action": [
                "lambda:invokeFunction",
                "lambda:InvokeAsync",
                "lambda:ListFunctions"
            ],
            "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:function-name"

        }
    ]
}
```

パイプラインを作成する

プロセッサ AWS Lambda として を使用するには、OpenSearch Ingestion パイプラインを設定し、プロセッサaws_lambdaとして を指定します。[AWS Lambda カスタムエンリッチメント] ブループリントを使用してパイプラインを作成することもできます。詳細については、「the section called “ブループリントの使用”」を参照してください。

次のパイプラインの例では、HTTP ソースからデータを受信し、日付プロセッサと AWS Lambda プロセッサを使用してデータを強化し、処理されたデータを OpenSearch ドメインに取り込みます。

```
version: "2"
lambda-processor-pipeline:
  source:
    http:
      path: "/${pipelineName}/logs"
  processor:
      - date:
        destination: "@timestamp"
        from_time_received: true
    - aws_lambda:
        function_name: "my-lambda-function"

        tags_on_failure: ["lambda_failure"]
        batch:
            key_name: "events"
        aws:
```

```
          region: region
  sink:
    - opensearch:
        hosts: [ "https://search-mydomain.us-east-1es.amazonaws.com" ]
        index: "table-index"
        aws:
          region: "region"
          serverless: false
```

次の AWS Lambda 関数の例では、指定されたイベント配列の各要素に新しいキーと値のペア ("transformed": "true") を追加して受信データを変換し、変更されたバージョンを返します。

```
import json

def lambda_handler(event, context):
    input_array = event.get('events', [])
    output = []
    for input in input_array:
        input["transformed"] = "true";
        output.append(input)

    return output
```

バッチ処理

パイプラインは、バッチ処理されたイベントを Lambda プロセッサに送信し、バッチサイズを動的に調整して 5 MB の制限を下回っないようにします。

以下は、パイプラインバッチの例です。

```
batch:
    key_name: "events"

input_arrary = event.get('events', [])
```

> **Note**
>
> パイプラインを作成するときは、Lambda プロセッサ設定の key_name オプションが Lambda ハンドラーのイベントキーと一致していることを確認してください。

条件付きフィルタリング

条件付きフィルタリングを使用すると、イベントデータの特定の条件に基づいて、 AWS Lambda プロセッサが Lambda 関数を呼び出すタイミングを制御できます。これは、他のイベントを無視しながら特定のタイプのイベントを選択的に処理する場合に特に便利です。

次の設定例では、条件付きフィルタリングを使用しています。

```
processors:
  - aws_lambda:
      function_name: "my-lambda-function"
      aws:
        region: "region"
      lambda_when: "/sourceIp == 10.10.10.10"
```

Amazon OpenSearch Ingestion を使用したドメインとコレクション間のデータの移行

OpenSearch Ingestion パイプラインを使用して、Amazon OpenSearch Service のドメイン間または OpenSearch Serverless VPC コレクション間でデータを移行できます。そのためには、1 つのドメインまたはコレクションをソースとして設定し、別のドメインまたはコレクションをシンクとして設定するパイプラインを設定します。これにより、1 つのドメインまたはコレクションから他のドメインまたはコレクションにデータが効果的に移行されます。

データを移行するには、次のリソースが必要です。

- 送信元の OpenSearch Service ドメインまたは OpenSearch Serverless VPC コレクション。このドメインまたはコレクションには、移行するデータが含まれています。ドメインを使用している場合、OpenSearch 1.0 以降、または Elasticsearch 7.4 以降を実行している必要があります。ドメインには、パイプラインロールに適切なアクセス許可を付与するアクセスポリシーも必要です。
- データを移行する別のドメインまたは VPC コレクション。このドメインまたはコレクションは、パイプラインシンクとして機能します。
- OpenSearch Ingestion がコレクションまたはドメインの読み取りと書き込みに使用するパイプラインロール。このロールの Amazon リソースネーム (ARN) をパイプライン設定に含めます。詳細については、以下のリソースを参照してください。
  - the section called “パイプラインにドメインへのアクセスを付与する”
  - the section called “パイプラインにコレクションへのアクセス権を付与する”

トピック

- 制限事項
- 送信元としての OpenSearch Service
- 複数の OpenSearch Service ドメインシンクの指定
- OpenSearch Serverless VPC コレクションへのデータの移行

制限事項

OpenSearch Service ドメインまたは OpenSearch Serverless コレクションをシンクとして指定する場合、以下の制限が適用されます。

- パイプラインは複数の VPC ドメインに書き込むことはできません。
- VPC アクセスを使用する OpenSearch Serverless コレクションとの間でのデータの移行がサポートされています。パブリックコレクションは一部の設定でも機能する場合がありますが、本番環境での使用には VPC ベースのコレクションが推奨されます。
- 1 つのパイプライン設定で VPC ドメインとパブリックドメインの組み合わせを指定することはできません。
- 1 つのパイプライン設定内で最大 20 個の非パイプラインシンクを持つことができます。
- 1 つのパイプライン設定 AWS リージョン で最大 3 つの異なるシンクを指定できます。
- 複数のシンクを含むパイプラインでは、いずれかのシンクが長時間ダウンしている場合や、受信データを受信するのに十分な容量でプロビジョニングされていない場合、時間の経過に伴って処理速度が低下する可能性があります。

送信元としての OpenSearch Service

ソースとして指定したドメインまたはコレクションは、データの移行元です。

IAM でパイプラインロールを作成する

OpenSearch Ingestion パイプラインを作成するには、最初にパイプラインロールを作成して、ドメインまたはコレクション間の読み取り/書き込みアクセスを付与する必要があります。これを作成するには、次のステップを実行します。

1. IAM で新しいアクセス許可ポリシーを作成してパイプラインロールにアタッチします。ソースから読み取り、シンクに書き込むアクセス許可を付与してください。OpenSearch Service ドメイン

の IAM パイプライン許可の設定の詳細については、「the section called “パイプラインにドメインへのアクセスを付与する”」および「the section called “パイプラインにコレクションへのアクセス権を付与する”」を参照してください。

2. IAM パイプラインロールに次の許可を指定して、送信元からの読み取ります。

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"es:ESHttpGet",
         "Resource":[
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/_cat/
indices",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/_search",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/_search/
scroll",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/*/_search"
         ]
      },
      {
         "Effect":"Allow",
         "Action":"es:ESHttpPost",
         "Resource":[
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/*/_search/
point_in_time",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/*/_search/
scroll"
         ]
      },
      {
         "Effect":"Allow",
         "Action":"es:ESHttpDelete",
         "Resource":[
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/_search/
point_in_time",
            "arn:aws:es:us-east-1:111122223333:domain/domain-name/_search/
scroll"
         ]
      }
```

```
    ]
}
```

パイプラインの作成

パイプラインロールにアポリシーをアタッチしたら、[AWSOpenSearchDataMigrationPipeline] 移行ブループリントを使用してパイプラインを作成します。このブループリントには、OpenSearch Service ドメインまたはコレクション間でデータを移行するためのデフォルト設定が含まれています。詳細については、「the section called “ブループリントの使用”」を参照してください。

> **Note**
>
> OpenSearch Ingestion は、送信元ドメインのバージョンとディストリビューションを使用して、移行に使用するメカニズムを決定します。一部のバージョンでは、`point_in_time` オプションがサポートされています。OpenSearch Serverless は をサポートしていないため、 `search_after`オプションを使用します`scroll`。OpenSearch Serverless は `point_in_time` (PIT) オペレーションをサポートしていることに注意してください。ただし、パイプラインはすべてのソースタイプの互換性`search_after`のために を使用します。

移行プロセス中に新しいインデックスが作成中である場合や、移行中にドキュメントの更新が進行中である場合があります。そのため、新しいデータや更新されたデータを取得するために、ドメインインデックスデータを 1 回だけでなく複数回スキャンする必要がある場合があります。

パイプライン設定で `index_read_count` と `interval` を設定して、スキャンを実行する回数を指定します。次の例では、複数回のスキャンを実行する方法を示しています。

```
scheduling:
    interval: "PT2H"
    index_read_count: 3
    start_time: "2023-06-02T22:01:30.00Z"
```

OpenSearch Ingestion では、次の設定を使用して、データが同じインデックスに書き込まれ、同じドキュメント ID が維持されるようにします。

```
index: "${getMetadata(\"opensearch-index\")}"
document_id: "${getMetadata(\"opensearch-document_id\")}"
```

複数の OpenSearch Service ドメインシンクの指定

複数のパブリック OpenSearch Service ドメインをデータの宛先として指定できます。この機能を使用して、条件付きルーティングを実行することや、受信データを複数の OpenSearch Service ドメインに複製することができます。最大 10 個の異なるパブリック OpenSearch Service ドメインをシンクとして指定できます。

次の例では、受信データは条件付きで異なる OpenSearch Service ドメインにルーティングされます。

```
...
  route:
    - 2xx_status: "/response >= 200 and /response < 300"
    - 5xx_status: "/response >= 500 and /response < 600"
  sink:
    - opensearch:
        hosts: [ "https://search-response-2xx.region.es.amazonaws.com" ]
        aws:
          region: "us-east-1"
        index: "response-2xx"
        routes:
          - 2xx_status
    - opensearch:
        hosts: [ "https://search-response-5xx.region.es.amazonaws.com" ]
        aws:
          region: "us-east-1"
        index: "response-5xx"
        routes:
          - 5xx_status
```

OpenSearch Serverless VPC コレクションへのデータの移行

OpenSearch Ingestion を使用して、送信元の OpenSearch Service のドメインまたは OpenSearch Serverless コレクションから VPC コレクションシンクにデータを移行できます。パイプライン設定内でネットワークアクセスポリシーを指定する必要があります。OpenSearch Serverless VPC コレクションへのデータインジェストの詳細については、「the section called “チュートリアル: コレクションにデータを取り込む”」を参照してください。

VPC コレクションにデータを移行するには

1. OpenSearch Serverless コレクションを作成します。手順については、「the section called “チュートリアル: コレクションにデータを取り込む”」を参照してください。
2. コレクションエンドポイントと Dashboard エンドポイントの両方に対する VPC アクセスを指定するコレクションのネットワークポリシーを作成します。手順については、「the section called “ネットワークアクセス”」を参照してください。
3. パイプラインロールをまだ持っていない場合は作成します。手順については、「the section called “パイプラインロール”」を参照してください。
4. パイプラインを作成します。手順については、「the section called “ブループリントの使用”」を参照してください。

AWS SDKs を使用して Amazon OpenSearch Ingestion を操作する

このセクションでは、 AWS SDKs を使用して Amazon OpenSearch Ingestion を操作する方法の例を示します。コード例は、ドメインとパイプラインを作成し、パイプラインにデータを取り込む方法を示しています。

トピック

- Python

Python

次のサンプルスクリプトでは、AWS SDK for Python (Boto3) を使用して IAM パイプラインロール、データ書き込み先ドメイン、およびデータを取り込むパイプラインを作成します。次に、`requests` HTTP ライブラリを使用してサンプルログファイルをパイプラインに取り込みます。

必要な従属関係をインストールには、次のコマンドを実行します。

```
pip install boto3
pip install botocore
pip install requests
pip install requests-auth-aws-sigv4
```

スクリプト内で、*`account-id`* のすべてのインスタンスを AWS アカウント ID に置き換えます。

```
import boto3
```

```
import botocore
from botocore.config import Config
import requests
from requests_auth_aws_sigv4 import AWSSigV4
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

opensearch = boto3.client('opensearch', config=my_config)
iam = boto3.client('iam', config=my_config)
osis = boto3.client('osis', config=my_config)

domainName = 'test-domain'  # The name of the domain
pipelineName = 'test-pipeline' # The name of the pipeline

def createPipelineRole(iam, domainName):
    """Creates the pipeline role"""
    response = iam.create_policy(
        PolicyName='pipeline-policy',
        PolicyDocument=f'{{\"Version\":\"2012-10-17\",\"Statement\":[{{\"Effect\":
\"Allow\",\"Action\":\"es:DescribeDomain\",\"Resource\":\"arn:aws:es:us-east-1:account-
id:domain\/{domainName}\"}},{{\"Effect\":\"Allow\",\"Action\":\"es:ESHttp*\",\"Resource
\":\"arn:aws:es:us-east-1:account-id:domain\/{domainName}\/*\"}}]}}'
    )
    policyarn = response['Policy']['Arn']

    response = iam.create_role(
        RoleName='PipelineRole',
        AssumeRolePolicyDocument='{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"osis-pipelines.amazonaws.com\"},\"Action\":
\"sts:AssumeRole\"}]}'
    )
    rolename=response['Role']['RoleName']

    response = iam.attach_role_policy(
        RoleName=rolename,
        PolicyArn=policyarn
    )

    print('Creating pipeline role...')
    time.sleep(10)
    print('Role created: ' + rolename)
```

```
def createDomain(opensearch, domainName):
    """Creates a domain to ingest data into"""
    response = opensearch.create_domain(
        DomainName=domainName,
        EngineVersion='OpenSearch_2.3',
        ClusterConfig={
            'InstanceType': 't2.small.search',
            'InstanceCount': 5,
            'DedicatedMasterEnabled': True,
            'DedicatedMasterType': 't2.small.search',
            'DedicatedMasterCount': 3
        },
        # Many instance types require EBS storage.
        EBSOptions={
            'EBSEnabled': True,
            'VolumeType': 'gp2',
            'VolumeSize': 10
        },
        AccessPolicies=f'{{\"Version\":\"2012-10-17\",\"Statement\":[{{\"Effect
\":\"Allow\",\"Principal\":{{\"AWS\":\"arn:aws:iam::account-id:role\/PipelineRole
\"}},\"Action\":\"es:*\",\"Resource\":\"arn:aws:es:us-east-1:account-id:domain\/
{domainName}\/*\"}}]}}',
        NodeToNodeEncryptionOptions={
            'Enabled': True
        }
    )
    return(response)

def waitForDomainProcessing(opensearch, domainName):
    """Waits for the domain to be active"""
    try:
        response = opensearch.describe_domain(
            DomainName=domainName
        )
        # Every 30 seconds, check whether the domain is processing.
        while 'Endpoint' not in response['DomainStatus']:
            print('Creating domain...')
            time.sleep(60)
            response = opensearch.describe_domain(
                DomainName=domainName)

        # Once we exit the loop, the domain is ready for ingestion.
        endpoint = response['DomainStatus']['Endpoint']
```

```
        print('Domain endpoint ready to receive data: ' + endpoint)
        createPipeline(osis, endpoint)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found.')
        else:
            raise error

def createPipeline(osis, endpoint):
    """Creates a pipeline using the domain and pipeline role"""
    try:
        definition = f'version: \"2\"\nlog-pipeline:\n  source:\n    http:\n      path:
 \"/${{pipelineName}}/logs\"\n  processor:\n    - date:\n        from_time_received:
 true\n        destination: \"@timestamp\"\n  sink:\n    - opensearch:\n        hosts:
 [ \"https://{endpoint}\" ]\n        index: \"application_logs\"\n        aws:\n
   region: \"us-east-1\"'
        response = osis.create_pipeline(
            PipelineName=pipelineName,
            MinUnits=4,
            MaxUnits=9,
            PipelineConfigurationBody=definition,
            PipelineRoleArn="arn:aws:iam::account-id:role/PipelineRole"
        )

        response = osis.get_pipeline(
                PipelineName=pipelineName
        )

        # Every 30 seconds, check whether the pipeline is active.
        while response['Pipeline']['Status'] == 'CREATING':
            print('Creating pipeline...')
            time.sleep(30)
            response = osis.get_pipeline(
                PipelineName=pipelineName)

        # Once we exit the loop, the pipeline is ready for ingestion.
        ingestionEndpoint = response['Pipeline']['IngestEndpointUrls'][0]
        print('Pipeline ready to ingest data at endpoint: ' + ingestionEndpoint)
        ingestData(ingestionEndpoint)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceAlreadyExistsException':
            print('Pipeline already exists.')
```

```
            response = osis.get_pipeline(
                PipelineName=pipelineName
            )
            ingestionEndpoint = response['Pipeline']['IngestEndpointUrls'][0]
            ingestData(ingestionEndpoint)
        else:
            raise error


def ingestData(ingestionEndpoint):
    """Ingests a sample log file into the pipeline"""
    endpoint = 'https://' + ingestionEndpoint
    r = requests.request('POST', f'{endpoint}/log-pipeline/logs',

 data='[{"time":"2014-08-11T11:40:13+00:00","remote_addr":"122.226.223.69","status":"404","requ
 http://www.k2proxy.com//hello.html HTTP/1.1","http_user_agent":"Mozilla/4.0
 (compatible; WOW64; SLCC2;)"}]',
    auth=AWSSigV4('osis'))
    print('Ingesting sample log file into pipeline')
    print('Response: ' + r.text)

def main():
    createPipelineRole(iam, domainName)
    createDomain(opensearch, domainName)
    waitForDomainProcessing(opensearch, domainName)

if __name__ == "__main__":
    main()
```

Amazon OpenSearch Ingestion のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – クラウドで AWS AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。「AWS 」 コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。

- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、OpenSearch Ingestion を使用するときに、責任共有モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように OpenSearch Ingestion を設定する方法について説明します。また、OpenSearch Ingestion リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon OpenSearch Ingestion パイプラインの VPC アクセスの設定
- クロスアカウント取り込み用の OpenSearch Ingestion パイプラインの設定
- Amazon OpenSearch Ingestion の アイデンティティとアクセス管理
- を使用した Amazon OpenSearch Ingestion API コールのログ記録 AWS CloudTrail
- Amazon OpenSearch Ingestion とインターフェイスエンドポイント API (AWS PrivateLink)

Amazon OpenSearch Ingestion パイプラインの VPC アクセスの設定

インターフェイス VPC エンドポイントを使用して Amazon OpenSearch Ingestion パイプラインにアクセスできます。VPC は、 専用の仮想ネットワークです AWS アカウント。クラウド内の他の仮想ネットワークから論理的に分離されています AWS 。VPC エンドポイントを介してパイプラインにアクセスすることで、インターネットゲートウェイや NAT デバイス、または VPN 接続を使わなくても、OpenSearch Ingestion と VPC の他のサービスとの間に安全な通信を確立できます。すべてのトラフィックは AWS クラウド内に安全に保持されます。

OpenSearch Ingestion がこのプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。パイプライン作成中に指定した各サブネットでエンドポイントネットワークインターフェイスを作成します。これらは OpenSearch Ingestion パイプラインに送信されるトラフィックのエントリポイントとして機能する、リクエスタマネージド型のネットワークインターフェイスです。また、インターフェイスエンドポイントを自分で作成および管理することもできます。

VPC を使用すれば、データを、パブリックインターネット経由ではなく VPC の境界内の OpenSearch Ingestion パイプラインを通して流すことができます。VPC 内にないパイプラインは、一般向けのエンドポイントとインターネットを経由してデータを送受信します。

VPC アクセスを持つパイプラインは、パブリックまたは VPC OpenSearch Service ドメイン、およびパブリックまたは VPC OpenSearch Serverless コレクションに書き込むことができます。

トピック

- 考慮事項
- 制限事項
- 前提条件
- パイプラインの VPC アクセスを設定する
- セルフマネージド VPC エンドポイント
- VPC アクセス用のサービスにリンクされたロール

考慮事項

パイプラインの VPC アクセスを設定する際は、以下の点に注意してください。

- パイプラインは、そのシンクと同じ VPC 内にある必要はありません。また、2 つの VPC 間の接続を確立する必要もありません。OpenSearch Ingestion が自動的にそれらを接続します。
- パイプラインに指定できる VPC は 1 つのみです。
- パブリックパイプラインとは異なり、VPC パイプラインは書き込み先のドメインまたはコレクションシンクと同じ AWS リージョン になければなりません。クロスリージョンを記述するために、パイプラインの S3 ソースを設定できます。
- パイプラインは、VPC の 1 つ、2 つ、3 つのいずれかのサブネットにデプロイできます。サブネットは、Ingestion OpenSearch Compute Units (OCU) がデプロイされているアベイラビリティーゾーンと同じアベイラビリティーゾーンに分散されます。
- パイプラインを 1 つのサブネットにのみデプロイした場合、そのアベイラビリティーゾーンがダウンするとデータを取り込めなくなります。高可用性を確保するために、パイプラインを 2 つまたは 3 つのサブネットで構成することが推奨されます。
- セキュリティグループの指定はオプションです。セキュリティグループを指定しないと、OpenSearch Ingestion では、VPC で指定されているデフォルトのセキュリティグループが使用されます。

制限事項

VPC 内のパイプラインには、次の制限事項があります。

- パイプラインの作成後は、パイプラインのネットワーク設定を変更できません。VPC 内でパイプラインを起動した場合、後からこれをパブリックエンドポイントに変更することはできず、その逆もまた同様です。
- インターフェイス VPC エンドポイントまたはパブリックエンドポイント内のいずれかでパイプラインを起動することはできますが、両方を実行することはできません。パイプラインを作成するときに、いずれかを選択する必要があります。
- VPC アクセス内でパイプラインをプロビジョニングした後は、そのパイプラインを別の VPC に移動させることはできず、そのサブネットまたはセキュリティグループの設定を変更することはできません。
- パイプラインが VPC アクセスを使用するドメインまたはコレクションシンクに書き込みを行う場合、パイプラインの作成後に、前に戻ってシンク (VPC またはパブリック) を変更することはできません。そのパイプラインを削除し、新しいシンクを使って改めて作成する必要があります。パブリックシンクから VPC アクセスのあるシンクに切り替えることは可能です。
- VPC パイプラインへのアカウントを横断した取り込みアクセスを提供することはできません。

前提条件

VPC アクセス内のパイプラインをプロビジョニングするには、まず次の手順を行います。

- 「VPC を作成する」

  VPC を作成するには、Amazon VPC AWS コンソール、 CLI、またはいずれかの AWS SDKs を使用できます。詳細については、Amazon VPC ユーザーガイドの「VPC の使用」を参照してください。VPC が既にある場合、このステップは省略できます。

- IP アドレスのリザーブ

  OpenSearch Ingestion が、パイプラインの作成時に指定した各サブネットに、elastic network interface を配置します。各ネットワークインターフェースは 1 つの IP アドレスに関連付けられます。ネットワークインターフェイス用にサブネットごとに 1 つの IP アドレスを予約する必要があります。

パイプラインの VPC アクセスを設定する

パイプラインの VPC アクセスは、OpenSearch Service コンソールから、または AWS CLIを使用して許可します。

コンソール

VPC アクセスの設定は、パイプラインの作成時に行います。[ソースネットワークオプション] で [VPC アクセス] を選択し、次のように設定します。

設定	説明
[エンドポイントの管理]	VPC エンドポイントを自分で作成するか、OpenSearch Ingestion で作成するかを選択します。
VPC	使用する仮想プライベートクラウド (VPC) の ID を選択します。VPC とパイプラインは同じ AWS リージョンの中になければなりません。
サブネット	1 つ以上のサブネットを選択します。OpenSearch Service は VPC エンドポイントと Elastic Network Interface (ENI) をサブネットに配置します。
セキュリティグループ	必要なアプリケーションが、パイプラインによって公開されているポート (80 または 443) とプロトコル (HTTP または HTTP) で、OpenSearch Ingestion パイプラインに到達できるよう、1 つ以上の VPC セキュリティグループを選択します。
[VPC アタッチメントオプション]	ソースが Amazon DocumentDB、セルフマネージド OpenSearch、Confluent Kafka などのクロス VPC 通信を必要とする場合、OpenSearch Ingestion は、これらのソースに接続するために指定したサブネットに Elastic Network Interface (ENI) を作成します。OpenSearch Ingestion は ENI を使用して、指定されたソースに到達します。[VPC にアタッチ] オプションにより、OpenSearch Ingestion データプレーン VPC を指定された VPC に接続します。 マネージド VPC の CIDR 予約を選択して、ネットワークインターフェイスをデプロイします。

CLI

を使用して VPC アクセスを設定するには AWS CLI、 `--vpc-options`パラメータを指定します。

```
aws osis create-pipeline \
  --pipeline-name vpc-pipeline \
  --min-units 4 \
```

```
  --max-units 10 \
  --vpc-options
 SecurityGroupIds={sg-12345678,sg-9012345},SubnetIds=subnet-1212234567834asdf \
  --pipeline-configuration-body "file://pipeline-config.yaml"
```

セルフマネージド VPC エンドポイント

パイプラインを作成するときは、エンドポイント管理を使用して、セルフマネージドエンドポイントまたはサービスマネージドエンドポイントを使用してパイプラインを作成できます。エンドポイント管理はオプションで、デフォルトでは OpenSearch Ingestion によって管理されるエンドポイントに設定されます。

でセルフマネージド VPC エンドポイントを使用してパイプラインを作成するには AWS マネジメントコンソール、OpenSearch Service コンソールを使用したパイプラインの作成」を参照してください。でセルフマネージド VPC エンドポイントを使用してパイプラインを作成するには AWS CLI、create-pipeline コマンドで `--vpc-options`パラメータを使用できます。

```
--vpc-options SubnetIds=subnet-abcdef01234567890,VpcEndpointManagement=CUSTOMER
```

エンドポイントサービスを指定すると、パイプラインへのエンドポイントを自分で作成できます。エンドポイントサービスを検索するには、get-pipeline コマンドを使用します。このコマンドは、次のようなレスポンスを返します。

```
"vpcEndpointService" : "com.amazonaws.osis.us-east-1.pipeline-
id-1234567890abcdef1234567890",
"vpcEndpoints" : [
  {
    "vpcId" : "vpc-1234567890abcdef0",
    "vpcOptions" : {
      "subnetIds" : [ "subnet-abcdef01234567890", "subnet-021345abcdef6789" ],
      "vpcEndpointManagement" : "CUSTOMER"
    }
  }
```

レスポンス`vpcEndpointService`の を使用して、 AWS マネジメントコンソール または で VPC エンドポイントを作成します AWS CLI。

セルフマネージド VPC エンドポイントを使用する場合は、VPC で DNS 属性 `enableDnsSupport` と `enableDnsHostnames` を有効にする必要があります。セルフマネージドエンドポイントを持つ

パイプラインを停止して再起動する場合は、アカウントで VPC エンドポイントを再作成する必要があります。

VPC アクセス用のサービスにリンクされたロール

サービスにリンクされたロールは、サービスに権限を委任する一意のタイプの IAM ロールであり、ユーザーに代わってリソースを作成して管理できます。サービスマネージド VPC エンドポイントを選択する場合、OpenSearch Ingestion には、VPC にアクセスし、パイプラインのエンドポイントを作成し、VPC のサブネットにネットワークインターフェースを配置するために、AWSServiceRoleForAmazonOpenSearchIngestionService という、サービスにリンクされたロールが必要です。

セルフマネージド VPC エンドポイントを選択した場合、OpenSearch Ingestion には AWSServiceRoleForOpensearchIngestionSelfManagedVpce という、サービスにリンクされたロールが必要です。これらのロール、そのアクセス許可と、その削除方法の詳細については、「the section called “パイプライン作成ロール”」を参照してください。

Ingestion パイプラインを作成すると、OpenSearch Ingestion がロールを自動的に作成します。この自動作成が成功するには、アカウントに最初のパイプラインを作成するユーザーが、`iam:CreateServiceLinkedRole` アクションに対するアクセス許可を持っている必要があります。詳細については、「IAM User Guide」の「Service-linked role permissions」を参照してください。ロールは、作成後に AWS Identity and Access Management (IAM) コンソールで表示できます。

クロスアカウント取り込み用の OpenSearch Ingestion パイプラインの設定

HTTP や OTel などのプッシュベースのソースの場合、Amazon OpenSearch Ingestion を使用すると、仮想プライベートクラウド (VPC) AWS アカウント から別の VPC のパイプラインエンドポイントにパイプラインを共有できます。組織内の他のチームと分析を共有するチームは、この機能を使用することで、ログ分析の共有などをより合理化された方法で行うことができます。

このセクションでは以下の用語を使用します。

- パイプライン所有者 – OpenSearch Ingestion パイプラインを所有および管理するアカウント。パイプラインを所有できるのは 1 つのアカウントのみです。
- 接続アカウント – 共有パイプラインに接続して使用するアカウント。複数のアカウントが同じパイプラインに接続できます。

OpenSearch Ingestion パイプラインを共有するように VPCs を設定するには AWS アカウント、以下で説明するように、次のタスクを実行します。

- (パイプライン所有者) パイプラインへのアクセスを接続アカウントに許可する
- (接続アカウント) 接続する VPC ごとにパイプラインエンドポイント接続を作成する

[開始する前に]

OpenSearch Ingestion パイプラインを共有するように VPCs を設定する前に AWS アカウント、次のタスクを完了します。

タスク	Details
1 つ以上の OpenSearch Ingestion パイプラインを作成する	OpenSearch コンピューティングユニット (OSU) の最小値を 2 以上に設定します。詳細については、「Amazon OpenSearch Ingestion パイプラインの作成」を参照してください。パイプライン更新の詳細については、「Amazon OpenSearch Ingestion パイプラインの更新」を参照してください。
OpenSearch Ingestion 用の VPC を 1 つ以上作成する	クロスアカウントパイプライン共有を有効にするには、パイプラインとパイプラインエンドポイントに関連する VPC に次の DNS 値を設定する必要があります。 • `enableDnsSupport=true` • `enableDnsHostnames=true` 詳細については、「Amazon VPC ユーザーガイド」の「VPC の DNS 属性」を参照してください。

パイプラインへのアクセスを接続アカウントに許可する

このセクションの手順では、OpenSearch Service コンソールと を使用して AWS CLI 、リソースポリシーを作成してクロスアカウントパイプラインアクセスを設定する方法について説明します。リソースポリシーを使用することで、パイプライン所有者はパイプラインにアクセスできる他のアカウ

ントを指定できます。作成後、パイプラインポリシーは、パイプラインが存在する限り、またはポリシーが削除されるまで維持されます。

> **Note**
>
> リソースポリシーは、IAM アクセス許可を使用した標準の OpenSearch Ingestion 認可に代わるものではありません。リソースポリシーは、クロスアカウントパイプラインアクセスを有効にするための追加の認可メカニズムです。

トピック

- パイプラインへのアクセスを接続アカウントに許可する (コンソール)
- パイプラインへのアクセスを接続アカウントに許可する (CLI)

パイプラインへのアクセスを接続アカウントに許可する (コンソール)

次の手順では、Amazon OpenSearch Service コンソールを使用して接続アカウントにパイプラインへのアクセスを許可します。

パイプラインエンドポイント接続を作成するには

1. Amazon OpenSearch Service コンソールのナビゲーションペインで、[取り込み] を展開し、[パイプライン] を選択します。
2. [パイプライン] セクションで、接続アカウントにアクセス許可を付与するパイプラインの名前を選択します。
3. [VPC エンドポイント] タブを選択します。
4. [認定プリンシパル] セクションで、[アカウントを認証する] を選択します。
5. [AWS アカウント ID] フィールドに 12 桁の数字のアカウント ID を入力し、[承認] を選択します。

パイプラインへのアクセスを接続アカウントに許可する (CLI)

次の手順では、 AWS CLIを使用して接続アカウントにパイプラインへのアクセスを許可します。

接続アカウントにパイプラインへのアクセスを許可するには

1. を最新バージョン (AWS CLI バージョン 2.0) に更新します。詳細については、「AWS CLIの最新バージョンのインストールまたは更新」を参照してください。
2. アカウントで CLI を開き、共有するパイプライン AWS リージョン を使用します。
3. 次のコマンドを実行して、パイプラインのリソースポリシーを作成します。このポリシーは、パイプラインに対する osis:CreatePipelineEndpoint アクセス許可を付与します。ポリシーには、許可する AWS アカウント ID を一覧表示できるパラメータが含まれています。

> **Note**
>
> 次のコマンドでは、12 桁のアカウント ID のみを指定して、短い形式のアカウント ID を使用する必要があります。ARN を使用しても機能しません。また、次に示すように、resource-arn の CLI パラメータおよび、Resource のポリシー JSON でパイプラインの Amazon リソースネーム (ARN) を指定する必要があります。

```
aws --region region osis put-resource-policy \
  --resource-arn arn:aws:osis:region:pipeline-owner-account-ID:pipeline/pipeline-name
  --policy 'IAM-policy'
```

*IAM ####*に次のようなポリシーを使用する

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
  {
  "Sid": "AllowAccess",
  "Effect": "Allow",
  "Principal": {
  "AWS": [
  "111122223333",
  "444455556666"
  ]
  },
  "Action":
```

```
 "osis:CreatePipelineEndpoint",
 "Resource": "arn:aws:osis:us-east-1:123456789012:pipeline/pipeline-name"
 }
 ]
}
```

接続する VPC ごとにパイプラインエンドポイント接続を作成する

パイプライン所有者が前の手順を使用して VPC 内のパイプラインへのアクセスを許可すると、接続アカウントのユーザーは VPC にパイプラインエンドポイントを作成します。このセクションでは、OpenSearch Service コンソールと AWS CLIを使用してエンドポイントを作成する手順について説明します。エンドポイントを作成すると、OpenSearch Ingestion は次のアクションを実行します。

- AWSServiceRoleForAmazonOpenSearchIngestionService サービスにリンクされたロールがまだ存在しない場合は、アカウントにロールを作成します。このロールは、接続アカウントのユーザーに CreatePipelineEndpoint API アクションを呼び出すアクセス許可を付与します。
- パイプラインエンドポイントを作成します。
- パイプライン所有者 VPC の共有パイプラインからデータを取り込むようにパイプラインエンドポイントを設定します。

トピック

- パイプラインエンドポイント接続の作成 (コンソール)
- パイプラインエンドポイント接続の作成 (CLI)

パイプラインエンドポイント接続の作成 (コンソール)

次の手順では、OpenSearch Service コンソールを使用してパイプラインエンドポイント接続を作成します。

パイプラインエンドポイント接続を作成するには

1. Amazon OpenSearch Service コンソールのナビゲーションペインで、[取り込み] を展開し、[VPC エンドポイント] を選択します。
2. [VPC エンドポイント] ページで、[作成] を選択します。

3. [パイプラインの場所]で、オプションを選択します。[現在のアカウント] を選択した場合は、リストからパイプラインを選択します。[クロスアカウント] を選択した場合は、フィールドにパイプライン ARN を入力します。「パイプラインへのアクセスを接続アカウントに許可する」で説明されているように、パイプライン所有者がパイプラインへのアクセスを許可している必要があります。
4. [VPC 設定] セクションの [VPC] で、リストから VPC を選択します。
5. [Subnets (サブネット)] では、1 個のサブネットを選択します。
6. [セキュリティグループ] で、グループを選択します。
7. エンドポイントの作成 を選択します。

作成したエンドポイントのステータスが `ACTIVE` に移行するのを待ちます。パイプラインが `ACTIVE` になると、`ingestEndpointUrl` という名前の新しいフィールドが表示されます。このエンドポイントを使用してパイプラインにアクセスし、FluentBit などのクライアントを使用してデータを取り込みます。FluentBit を使用してデータを取り込む方法については、「OpenSearch Ingestion パイプラインを Fluent Bit で使用する」を参照してください。

> **Note**
>
> `ingestEndpointUrl` は、すべての接続アカウントに対して同じ URL となります。

パイプラインエンドポイント接続の作成 (CLI)

次の手順では、 AWS CLIを使用してパイプラインエンドポイント接続を作成します

パイプラインエンドポイント接続を作成するには

1. まだ更新していない場合は、最新バージョンの に更新します AWS CLI (バージョン 2.0)。詳細については、「AWS CLIの最新バージョンのインストールまたは更新」を参照してください。
2. 共有パイプライン AWS リージョン を持つ の接続アカウントで CLI を開きます。
3. 次のコマンドを実行してパイプラインエンドポイントを作成します。

> **Note**
>
> 接続アカウント VPC には、少なくとも 1 つのサブネットと 1 つのセキュリティグループを指定する必要があります。セキュリティグループにはポート 443 が含まれ、接続アカウント VPC でクライアントをサポートする必要があります。

```
aws osis --region region create-pipeline-endpoint \
  --pipeline-arn arn:aws:osis:region:connecting-account-ID:pipeline/shared-pipeline-name
  --vpc-options SecurityGroupIds={sg-security-group-ID-1,sg-security-group-ID-2},SubnetIds=subnet-subnet-ID
```

4. 次のコマンドを実行して、前のコマンドで指定したリージョンのエンドポイントを一覧表示します。

```
aws osis --region region list-pipeline-endpoints
```

作成したエンドポイントのステータスが ACTIVE に移行するのを待ちます。パイプラインが ACTIVE になると、ingestEndpointUrl という名前の新しいフィールドが表示されます。このエンドポイントを使用してパイプラインにアクセスし、FluentBit などのクライアントを使用してデータを取り込みます。FluentBit を使用してデータを取り込む方法については、「OpenSearch Ingestion パイプラインを Fluent Bit で使用する」を参照してください。

> **Note**
>
> ingestEndpointUrl は、すべての接続アカウントに対して同じ URL となります。

パイプラインエンドポイントの削除

共有パイプラインへのアクセス提供を中止したい場合は、次のいずれかの方法を使用してパイプラインエンドポイントを削除できます。

- パイプラインエンドポイントの削除 (接続アカウント側)。
- パイプラインエンドポイントの取り消し (パイプライン所有者側)。

次の手順では、接続アカウントのパイプラインエンドポイントを削除します。

パイプラインエンドポイントを削除するには (接続アカウント側)

1. 共有パイプライン AWS リージョン を持つ の接続アカウントで CLI を開きます。
2. 次のコマンドを実行して、 リージョンのパイプラインエンドポイントを一覧表示します。

```
aws osis --region region list-pipeline-endpoints
```

削除したいパイプライン ID をメモします。

3. 以下のコマンドを実行して、パイプラインエンドポイントを削除します。

```
aws osis --region region delete-pipeline-endpoint \
  --endpoint-id 'ID'
```

共有パイプラインのパイプライン所有者は、次の手順を使用してパイプラインエンドポイントを取り消すことができます。

パイプラインエンドポイントを取り消すには (パイプライン所有者側)

1. 共有パイプライン AWS リージョン を持つ の接続アカウントで CLI を開きます。
2. 次のコマンドを実行して、リージョンのパイプラインエンドポイント接続を一覧表示します。

```
aws osis --region region list-pipeline-endpoint-connections
```

削除したいパイプライン ID をメモします。

3. 以下のコマンドを実行して、パイプラインエンドポイントを削除します。

```
aws osis --region region revoke-pipeline-endpoint-connections \
  --pipeline-arn pipeline-arn --endpoint-ids ID
```

コマンドでは、エンドポイント ID を 1 つだけ指定できます。

Amazon OpenSearch Ingestion の アイデンティティとアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、どのユーザーに対して認証 (サインイン

受け入れ) を行い、OpenSearch Ingestion リソースの使用を承認 (アクセス許可を付与) するかを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- OpenSearch Ingestion でのアイデンティティベースのポリシー
- OpenSearch Ingestion のポリシーアクション
- OpenSearch Ingestion のポリシーリソース
- Amazon OpenSearch Ingestion のポリシー条件キー
- OpenSearch Ingestion での ABAC
- OpenSearch Ingestion での一時的な認証情報の使用
- OpenSearch Ingestion でのサービスにリンクされたロール
- OpenSearch Ingestion での ID ベースのポリシー例

OpenSearch Ingestion でのアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

OpenSearch Ingestion での ID ベースのポリシー例

OpenSearch Ingestion での ID ベースのポリシー例については、「the section called “アイデンティティベースのポリシーの例”」を参照してください。

OpenSearch Ingestion のポリシーアクション

ポリシーアクションのサポート: あり

JSON ポリシーの Action 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

OpenSearch Ingestion のポリシーアクションでは、アクションの前に次のプレフィックスを使用します。

```
osis
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "osis:action1",
      "osis:action2"
         ]
```

ワイルドカード文字 (*) を使用すると、複数のアクションを指定することができます。例えば、List という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "osis:List*"
```

OpenSearch Ingestion での ID ベースのポリシー例については、「OpenSearch Serverless での ID ベースのポリシー例」を参照してください。

OpenSearch Ingestion のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベ

ルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon OpenSearch Ingestion のポリシー条件キー

サービス固有のポリシー条件キーへのサポート: なし

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

OpenSearch Ingestion の条件キーのリストは、「サービス認証リファレンス」の「Condition keys for Amazon OpenSearch Ingestion」でご確認ください。条件キーの使用が可能なアクションおよびリソースについては、「Amazon OpenSearch Ingestion で定義されるアクション」を参照してください。

OpenSearch Ingestion での ABAC

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM

ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

OpenSearch Ingestion リソースのタグ付けの詳細については、「the section called “パイプラインのタグ付け”」を参照してください。

OpenSearch Ingestion での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

OpenSearch Ingestion でのサービスにリンクされたロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

OpenSearch Ingestion は、`AWSServiceRoleForAmazonOpenSearchIngestionService` と呼ばれるサービスにリンクされたロールを使用します。`AWSServiceRoleForOpensearchIngestionSelfManagedVpce` という名前のサービスにリンクされたロールは、セルフマネージド VPC エンドポイントを持つパイプラインでも使用できます。OpenSearch Ingestion でのサービスにリンクされたロールの作成または管理の詳細については、「the section called “パイプライン作成ロール”」を参照してください。

OpenSearch Ingestion での ID ベースのポリシー例

デフォルトでは、OpenSearch Ingestion リソースを作成または変更する許可は、ユーザーおよびロールに付与されていません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

Amazon OpenSearch Ingestion で定義されるアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認証リファレンス」の「Actions, resources, and condition keys for Amazon OpenSearch Ingestion」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- コンソールでの OpenSearch Ingestion の使用
- OpenSearch Ingestion パイプラインの管理
- OpenSearch Ingestion パイプラインへのデータの取り込み

ポリシーに関するベストプラクティス

アイデンティティベースポリシーは非常に強力です。このポリシーは、アカウント内の OpenSearch Ingestion リソースを作成、アクセス、または削除することが可能なユーザーを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

アイデンティティベースのポリシーは、アカウント内の OpenSearch Ingestion リソースを作成、アクセス、または削除することが可能なユーザーを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセス

を許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。

- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

コンソールでの OpenSearch Ingestion の使用

OpenSearch Ingestion コンソールで OpenSearch Ingestion にアクセスするには、最小限のアクセス許可セットが必要です。これらのアクセス許可により、 AWS アカウントの OpenSearch Ingestion リソースの詳細を一覧表示および表示できます。必要最低限の許可よりも制限が厳しいアイデンティティベースポリシーを作成すると、そのポリシーを持つエンティティ (IAM ロールなど) に対してコンソールが意図したとおりに機能しなくなります。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

次のポリシーでは、OpenSearch Ingestion コンソールで OpenSearch Ingestion にアクセスすることを許可します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "*",
```

```
            "Effect": "Allow",
            "Action": [
                "osis:ListPipelines",
                "osis:GetPipeline",
                "osis:ListPipelineBlueprints",
                "osis:GetPipelineBlueprint",
                "osis:GetPipelineChangeProgress"
            ]
        }
    ]
}
```

または、 the section called “AmazonOpenSearchIngestionReadOnlyAccess” AWS 管理ポリシーを使用して、 のすべての OpenSearch Ingestion リソースへの読み取り専用アクセスを許可することもできます AWS アカウント。

OpenSearch Ingestion パイプラインの管理

このポリシーは、ユーザーに対して Amazon OpenSearch Ingestion パイプラインの管理および運用を許可する「パイプライン管理者」ポリシーの一例です。ユーザーはパイプラインの作成、表示、削除を行うことができます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "arn:aws:osis:us-east-1:111122223333:pipeline/*",
            "Action": [
                "osis:CreatePipeline",
                "osis:DeletePipeline",
                "osis:UpdatePipeline",
                "osis:ValidatePipeline",
                "osis:StartPipeline",
                "osis:StopPipeline"
            ],
            "Effect": "Allow"
        },
        {
            "Resource": "*",
```

```
            "Action": [
                "osis:ListPipelines",
                "osis:GetPipeline",
                "osis:ListPipelineBlueprints",
                "osis:GetPipelineBlueprint",
                "osis:GetPipelineChangeProgress"
            ],
            "Effect": "Allow"
        }
    ]
}
```

OpenSearch Ingestion パイプラインへのデータの取り込み

このポリシー例では、ユーザーまたは他のエンティティに対して、アカウント内の Amazon OpenSearch Ingestion パイプラインにデータを取り込むことを許可します。ユーザーがパイプラインを変更することはできません。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "arn:aws:osis:us-east-1:123456789012:pipeline/*",
            "Action": [
                "osis:Ingest"
            ],
            "Effect": "Allow"
        }
    ]
}
```

を使用した Amazon OpenSearch Ingestion API コールのログ記録 AWS CloudTrail

Amazon OpenSearch Ingestion は AWS CloudTrail、OpenSearch Ingestion のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。

CloudTrail は、OpenSearch Ingestion のすべての API コールをイベントとしてキャプチャします。キャプチャされる対象には、OpenSearch Service コンソールの OpenSearch Ingestion セクションからの呼び出しや、OpenSearch Ingestion API オペレーションへのコード呼び出しなどが含まれます。

証跡を作成すると、OpenSearch Ingestion のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にできます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。

CloudTrail で収集された情報を使用すると、OpenSearch Ingestion に対して発行されたリクエスト、リクエスト元の IP アドレス、リクエスト作成者、リクエスト作成日時、その他の詳細情報などを確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail での OpenSearch Ingestion の情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。OpenSearch Ingestion でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。最近のイベントは、 AWS アカウントで表示、検索、ダウンロードできます。詳細については、 CloudTrail イベント履歴でのイベントの表示を参照してください。

OpenSearch Ingestion のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。

証跡は、 AWS パーティション内のすべてのリージョンからのイベントを ログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての OpenSearch Ingestion アクションは CloudTrail によってログが記録されます。それらのアクションは「OpenSearch Ingestion API リファレンス」に記載されています。たとえば、`CreateCollection`、`ListCollections`、`DeleteCollection` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。この ID 情報は以下のことを確認するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

OpenSearch Ingestion のログファイルエントリの理解

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail ログファイルには、1 つ以上のログエントリがあります。

イベントは、任意の送信元からの単一のリクエストを表します。これには、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

以下の例は、`DeletePipeline` アクションを示す CloudTrail ログエントリです。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "access-key",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
```

```
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-04-21T16:48:33Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-04-21T16:49:22Z",
    "eventSource": "osis.amazonaws.com",
    "eventName": "UpdatePipeline",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "123.456.789.012",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36",
    "requestParameters": {
        "pipelineName": "my-pipeline",
        "pipelineConfigurationBody": "version: \"2\"\nlog-pipeline:\n  source:\n
 http:\n        path: \"/test/logs\"\n  processor:\n    - grok:\n        match:\n
        log: [ '%{COMMONAPACHELOG}' ]\n    - date:\n        from_time_received: true
\n        destination: \"@timestamp\"\n  sink:\n    - opensearch:\n        hosts:
 [ \"https://search-b5zd22mwxhggheqpj5ftslgyle.us-west-2.es.amazonaws.com\" ]\n
  index: \"apache_logs2\"\n        aws_sts_role_arn: \"arn:aws:iam::709387180454:role/
canary-bootstrap-OsisRole-J1BARLD26QKN\"\n        aws_region: \"us-west-2\"\n
 aws_sigv4: true\n"
    },
    "responseElements": {
        "pipeline": {
            "pipelineName": "my-pipeline",sourceIPAddress
            "pipelineArn": "arn:aws:osis:us-west-2:123456789012:pipeline/my-pipeline",
            "minUnits": 1,
            "maxUnits": 1,
            "status": "UPDATING",
            "statusReason": {
                "description": "An update was triggered for the pipeline. It is still
 available to ingest data."
            },
            "pipelineConfigurationBody": "version: \"2\"\nlog-pipeline:\n  source:\n
    http:\n        path: \"/test/logs\"\n  processor:\n    - grok:\n        match:
\n          log: [ '%{COMMONAPACHELOG}' ]\n    - date:\n        from_time_received:
 true\n        destination: \"@timestamp\"\n  sink:\n    - opensearch:\n        hosts:
```

```
 [ \"https://search-b5zd22mwxhggheqpj5ftslgyle.us-west-2.es.amazonaws.com\" ]\n
  index: \"apache_logs2\"\n        aws_sts_role_arn: \"arn:aws:iam::709387180454:role/
canary-bootstrap-OsisRole-J1BARLD26QKN\"\n        aws_region: \"us-west-2\"\n
 aws_sigv4: true\n",
            "createdAt": "Mar 29, 2023 1:03:44 PM",
            "lastUpdatedAt": "Apr 21, 2023 9:49:21 AM",
            "ingestEndpointUrls": [
                "my-pipeline-tu33ldsgdltgv7x7tjqiudvf7m.us-west-2.osis.amazonaws.com"
            ]
        }
    },
    "requestID": "12345678-1234-1234-1234-987654321098",
    "eventID": "12345678-1234-1234-1234-987654321098",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "709387180454",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "osis.us-west-2.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```


Amazon OpenSearch Ingestion とインターフェイスエンドポイント API (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、VPC と OpenSearch Ingestion API エンドポイント間のプライベート接続を確立できます。インターフェイスエンドポイントは を使用しますAWS PrivateLink

AWS PrivateLink を使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続なしで、OpenSearch Ingestion API オペレーションにプライベートにアクセスできます。VPC 内のリソースは、パイプラインを作成、変更、または削除するために OpenSearch Ingestion API エンドポイントと通信するためにパブリック IP アドレスを必要としません。VPC と OpenSearch Ingestion 間のトラフィックは Amazon ネットワークを離れません。

> **Note**
>
> このトピックでは、OpenSearch Ingestion API にアクセスするための VPC エンドポイントについて説明します。これにより、VPC 内からパイプライン (作成、更新、削除) を管理できます。これは、VPC 内のソースからパイプラインにデータを取り込む方法を制御するパイプライン自体の VPC アクセスの設定とは異なります。パイプラインの VPC アクセスの設定については、「」を参照してくださいthe section called “パイプラインの VPC アクセスを設定する”。

各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。Elastic Network Interface の詳細については、Amazon EC2 ユーザーガイドの「Elastic Network Interface」を参照してください。

VPC エンドポイントの詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。OpenSearch Ingestion API オペレーションの詳細については、OpenSearch Ingestion API リファレンスを参照してください。

VPC エンドポイントに関する考慮事項

OpenSearch Ingestion API エンドポイントのインターフェイス VPC エンドポイントを設定する前に、Amazon VPC ユーザーガイドのインターフェイスエンドポイントのプロパティと制限を確認してください。

OpenSearch Ingestion リソースの管理に関連するすべての OpenSearch Ingestion API オペレーションは、 を使用して VPC から使用できます AWS PrivateLink。

VPC エンドポイントポリシーは、OpenSearch Ingestion API エンドポイントでサポートされています。デフォルトでは、OpenSearch Ingestion API オペレーションへのフルアクセスはエンドポイントを介して許可されます。詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

利用可能な状況

OpenSearch Ingestion API は現在、すべての OpenSearch Ingestion リージョンで VPC エンドポイントをサポートしています。

現時点では、FIPS エンドポイントはサポートされていません。

OpenSearch Ingestion API 用のインターフェイス VPC エンドポイントの作成

OpenSearch Ingestion API の VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface () を使用して作成できますAWS CLI。詳細については、「Amazon VPC ユーザーガイド」のインターフェイスエンドポイントの作成を参照してください。

サービス名 を使用して OpenSearch Ingestion API の VPC エンドポイントを作成しますcom.amazonaws.*region*.osis。

エンドポイントのプライベート DNS を有効にすると、 などの AWS リージョンのデフォルト DNS 名を使用して、VPC エンドポイントを使用して OpenSearch Ingestion に API リクエストを行うことができますosis.us-east-1.amazonaws.com。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

OpenSearch Ingestion API 用の VPC エンドポイントポリシーの作成

OpenSearch Ingestion API へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- アクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

例: OpenSearch Ingestion API アクションの VPC エンドポイントポリシー

OpenSearch Ingestion API のエンドポイントポリシーの例を次に示します。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リストされている OpenSearch Ingestion API アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
```

```
            "osis:CreatePipeline",
            "osis:UpdatePipeline",
            "osis:DeletePipeline"
         ],
         "Resource":"*"
      }
   ]
}
```

例: 指定された AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、エンドポイントを使用したリソースへの123456789012すべてのアクセスを AWS アカウントで拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Amazon OpenSearch Ingestion パイプラインのタグ付け

タグを使用すると、Amazon OpenSearch Ingestion パイプラインに任意の情報を割り当て、その情報を分類しフィルタリングすることができます。タグは、ユーザーが割り当てるか、 AWS リソース AWS に割り当てるメタデータラベルです。各タグは、キーと値から構成されます。ユーザーが割り当てるタグでは、ユーザーがキーと値を定義します。たとえば、1 つのリソースのキーを `stage` と定義し、値を `test` と定義します。

タグは、以下のことに役立ちます。

- AWS リソースを特定して整理します。多くの AWS サービスはタグ付けをサポートしているため、異なるサービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。例えば、Amazon OpenSearch Service ドメインに割り当てる OpenSearch Ingestion パイプラインに、同じタグを割り当てることができます。
- AWS コストを追跡します。 AWS Billing and Cost Management ダッシュボードでこれらのタグをアクティブ化します。 はタグ AWS を使用してコストを分類し、毎月のコスト配分レポートを配信します。詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグの使用」(Use Cost Allocation Tags) を参照してください。
- 属性ベースのアクセス制御を使用して、パイプラインへのアクセスを制限します。詳細については、IAM ユーザーガイドのタグキーに基づいたアクセス制御を参照してください。

OpenSearch Ingestion では、主要なリソースはパイプラインです。OpenSearch Service コンソール、 AWS CLI、OpenSearch Ingestion APIs、または AWS SDKs を使用して、パイプラインからタグを追加、管理、削除できます。

トピック

- 必要なアクセス許可
- タグの操作 (コンソール)
- タグの操作 (AWS CLI)

必要なアクセス許可

OpenSearch Ingestion は、パイプラインのタグ付けに次の AWS Identity and Access Management Access Analyzer (IAM) アクセス許可を使用します。

- `osis:TagResource`
- `osis:ListTagsForResource`
- `osis:UntagResource`

各アクセス許可の詳細については、「サービス認証リファレンス」の「Actions, resources, and condition keys for OpenSearch Ingestion」を参照してください。

タグの操作 (コンソール)

コンソールは、パイプラインにタグ付けする際の最も簡単な方法です。

タグを作成するには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. タグを追加するパイプラインを選択し、[タグ] タブに移動します。
3. [管理] を選択して、[新しいタグを追加] を選択します。
4. タグキーとオプションの値を入力します。
5. [保存] を選択します。

タグを削除するには、同じ手順に従って、[タグを管理] ページで [削除] を選択します。

タグを操作するコンソールを使用する方法の詳細については、AWS マネジメントコンソール入門ガイドの「タグエディター」を参照してください。

タグの操作 (AWS CLI)

を使用してパイプラインにタグを付けるには AWS CLI、 `TagResource`リクエストを送信します。

```
aws osis tag-resource
  --arn arn:aws:osis:us-east-1:123456789012:pipeline/my-pipeline
  --tags Key=service,Value=osis Key=source,Value=otel
```

`UntagResource` コマンドを使用してパイプラインからタグを削除します。

```
aws osis untag-resource
  --arn arn:aws:osis:us-east-1:123456789012:pipeline/my-pipeline
  --tag-keys service
```

`ListTagsForResource` コマンドを使用してパイプラインの既存のタグを表示します。

```
aws osis list-tags-for-resource
  --arn arn:aws:osis:us-east-1:123456789012:pipeline/my-pipeline
```

Amazon CloudWatch を使用した Amazon OpenSearch Ingestion のログ記録とモニタリング

Amazon OpenSearch Ingestion は、メトリクスとログを Amazon CloudWatch にパブリッシュします。

トピック

- パイプラインのログのモニタリング
- パイプラインメトリクスのモニタリング

パイプラインのログのモニタリング

Amazon OpenSearch Ingestion パイプラインのログ記録を有効にすることで、パイプラインのオペレーション中や取り込みアクティビティ中に発生した、エラーや警告メッセージを公開できます。OpenSearch Ingestion は、すべてのログを Amazon CloudWatch Logs にパブリッシュします。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。

OpenSearch Ingestion のログに、リクエスト処理の失敗、送信元からシンクへの認証エラー、その他、トラブルシューティングに役立つ警告が表示されることがあります。OpenSearch Ingestion は、ログでは、`INFO`、`WARN`、`ERROR`、`FATAL` のログレベルを使用します。ログのパブリッシュは、すべてのパイプラインで有効にすることが推奨されています。

必要なアクセス許可

OpenSearch Ingestion を有効にしてログを CloudWatch Logs へ送信するには、特定の IAM アクセス権限を持つユーザーとしてサインインする必要があります。

ログ配信リソースを作成し更新するには、次の CloudWatch Logs のアクセス権限が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:DescribeResourcePolicies",
```

```
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
            ]
        }
    ]
}
```

ログパブリッシュの有効化

ログのパブリッシュは、既存のパイプラインで有効にするか、パイプラインの作成中に有効にすることができます。パイプラインの作成中にログのパブリッシュを有効にする手順については、「the section called “パイプラインの作成”」を参照してください。

コンソール

既存のパイプラインでログのパブリッシュを有効にするには

1. https://console.aws.amazon.com/aos/osis/home で Amazon OpenSearch Service コンソールにサインインします。Pipelines ページが表示されます。
2. ログを有効にするパイプラインを開き、[アクション]、[ログ発行のオプションの編集] を選択します。
3. [CloudWatch Logs への発行] を有効にします
4. 新しいロググループを作成するか、既存のロググループを選択します。名前は、`/aws/vendedlogs/OpenSearchIngestion/pipeline-name/audit-logs` のようなパスの形式にすることが推奨されます。この形式を使用すれば、`/aws/vendedlogs/OpenSearchIngestion` のような特定のパスを持つすべてのロググループにアクセス権限を付与する、CloudWatch アクセスポリシーの適用が容易になります。

   > **Important**
   >
   > ロググループ名には、プレフィックス `vendedlogs` を含めます。さもないと作成に失敗します。

5. [保存] を選択します。

CLI

を使用してログ発行を有効にするには AWS CLI、次のリクエストを送信します。

```
aws osis update-pipeline \
  --pipeline-name my-pipeline \
  --log-publishing-options  IsLoggingEnabled=true,CloudWatchLogDestination={LogGroup="/
aws/vendedlogs/OpenSearchIngestion/pipeline-name"}
```

パイプラインメトリクスのモニタリング

Amazon CloudWatch を使って Amazon OpenSearch Ingestion パイプラインをモニタリングすることができます。Amazon CloudWatch が raw データを収集し、読み取り可能な、ほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

OpenSearch Ingestion のコンソールでは、CloudWatch の raw データに基づく一連のグラフが、各パイプラインの [パフォーマンス] タブに表示されます。

OpenSearch Ingestion は、サポートされているほとんどのプラグインのメトリクスを報告します。特定のプラグインに、以下の、固有の表がなければ、プラグインに固有のメトリクスは報告されていないということを意味します。パイプラインのメトリクスは、AWS/OSIS 名前空間にパブリッシュされます。

トピック

- 一般的なメトリクス
- バッファのメトリクス
- Signature V4 メトリクス
- 制限付きブロッキングバッファのメトリクス
- Otel トレースソースメトリクス
- Otel メトリクスソースメトリクス
- HTTP メトリクス
- S3 メトリクス
- メトリクス集約
- 日付メトリクス
- Lambda metrics

- Grok メトリクス
- Otel トレース raw メトリクス
- Otel トレースグループメトリクス
- サービスマップステートフルメトリクス
- OpenSearch メトリクス
- システムと測定メトリクス

一般的なメトリクス

以下は、すべてのプロセッサーとシンクに共通のメトリクスです。

各メトリクスの先頭には、サブパイプライン名とプラグイン名が <*sub_pipeline_name*><*plugin*><*metric_name*> の形式で付きます。例えば、サブパイプライン名が my-pipeline である recordsIn.count メトリクスと date プロセッサの完全名は my-pipeline.date.recordsIn.count になります。

メトリクスのサフィックス	説明
recordsIn.count	パイプラインのコンポーネントへのレコードのイングレス。このメトリクスは、プロセッサとシンクに適用されます。 関連する統計情報: Sum ディメンション:PipelineName
recordsOut.count	パイプラインのコンポーネントへのレコードのエグレス。このメトリクスは、プロセッサと送信元に適用されます。 関連する統計情報: Sum ディメンション:PipelineName
timeElapsed.count	パイプラインコンポーネントの実行中に記録されたデータポイントの数。このメトリクスは、プロセッサとシンクに適用されます。 関連する統計情報: Sum

メトリクスのサフィックス	説明
	ディメンション:PipelineName
timeElapsed.sum	パイプラインコンポーネントの実行中に経過した合計時間。このメトリクスは、プロセッサとシンクにミリ秒単位で適用されます。 関連する統計情報: Sum ディメンション:PipelineName
timeElapsed.max	パイプラインコンポーネントの実行中に経過した最大時間。このメトリクスは、プロセッサとシンクにミリ秒単位で適用されます。 関連する統計情報: Max ディメンション:PipelineName
PipelineLatency.avg	ソース読み取りからシンク書き込みまでパイプライン全体を通過するイベントの平均end-to-end時間をミリ秒単位で表します。 関連する統計: Average ディメンション:PipelineName
PipelineLatency.max	イベントがソース読み取りからシンク書き込みまでパイプライン全体を通過するまでのend-to-end時間。 関連する統計: 最大 ディメンション:PipelineName

バッファのメトリクス

次のメトリクスは、OpenSearch Ingestion がすべてのパイプラインに自動的に設定するデフォルトの制限のあるブロッキングバッファに適用されます。

各メトリクスの先頭には、サブパイプライン名とバッファ名が <*sub_pipeline_name*><*buffer_name*><*metric_name*> の形式で付きます。例えば、サブパイプライン名が my-pipeline である recordsWritten.count メトリクスの完全名は my-pipeline.BlockingBuffer.recordsWritten.count になります。

メトリクスのサフィックス	説明
recordsWritten.count	バッファに書き込まれたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsRead.count	バッファから読み取られたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsInFlight.value	バッファから読み取られた未チェックのレコードの数。 関連する統計: Average ディメンション:PipelineName
recordsInBuffer.value	現在バッファに入っているレコードの数。 関連する統計: Average ディメンション:PipelineName
recordsProcessed.count	バッファから読み取られパイプラインによって処理されたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsWriteFailed.count	パイプラインがシンクへの書き込みに失敗したレコードの数。 関連する統計情報: Sum

メトリクスのサフィックス	説明
	ディメンション:PipelineName
writeTimeElapsed.count	バッファへの書き込み中に記録されたデータポイントの数。 関連する統計情報: Sum ディメンション:PipelineName
writeTimeElapsed.sum	バッファへの書き込み中に経過した合計時間 (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
writeTimeElapsed.max	バッファへの書き込み中に経過した最大時間 (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
writeTimeouts.count	バッファへの書き込みタイムアウトの数。 関連する統計情報: Sum ディメンション:PipelineName
readTimeElapsed.count	バッファへの読み取り中に記録されたデータポイントの数。 関連する統計情報: Sum ディメンション:PipelineName
readTimeElapsed.sum	バッファからの読み取り中に経過した合計時間 (ミリ秒単位) 。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
readTimeElapsed.max	バッファからの読み取り中に経過した最大時間 (ミリ秒単位) 。 関連する統計情報: Max ディメンション:PipelineName
checkpointTimeElapsed.count	チェックポイント中に記録されたデータポイントの数。 関連する統計情報: Sum ディメンション:PipelineName
checkpointTimeElapsed.sum	チェックポイント中に経過した合計時間 (ミリ秒単位)。 関連する統計情報: Sum ディメンション:PipelineName
checkpointTimeElapsed.max	チェックポイント中に経過した最大時間 (ミリ秒単位)。 関連する統計情報: Max ディメンション:PipelineName

Signature V4 メトリクス

次のメトリクスは、パイプラインの取り込みエンドポイントに適用され、ソースのプラグイン (http、otel_trace、otel_metrics) に関連付けられています。取り込みエンドポイントへのリクエストでは、すべて、Signature Version 4 を使用して署名する必要があります。これらのメトリクスは、パイプラインに接続する際の認可の問題を特定したり、認証が正常に行われていることを確認したりするときに役立ちます。

各メトリクスの先頭に、サブパイプライン名と osis_sigv4_auth が付きます。例えば、*sub_pipeline_name*.osis_sigv4_auth.httpAuthSuccess.count。

メトリクスのサフィックス	説明
`httpAuthSuccess.count`	Signature V4 を使用したパイプラインへのリクエストで成功した数。 関連する統計情報: Sum ディメンション:`PipelineName`
`httpAuthFailure.count`	Signature V4 を使用したパイプラインへのリクエストで失敗した数。 関連する統計情報: Sum ディメンション:`PipelineName`
`httpAuthServerError.count`	Signature V4 を使用したパイプラインへのリクエストでサーバーエラーが返された数。 関連する統計情報: Sum ディメンション:`PipelineName`

制限付きブロッキングバッファのメトリクス

次のメトリクスは、制限付きブロッキングバッファに適用されます。各メトリクスの先頭に、サブパイプライン名と `BlockingBuffer` が付きます。例えば、*`sub_pipeline_name`*`.BlockingBuffer.bufferUsage.value`。

メトリクスのサフィックス	説明
`bufferUsage.value`	バッファ内のレコードの数に基づく `buffer_size` の使用率です。 `buffer_size` は、バッファに書き込まれるレコードの最大数、およびチェックが済んでいない実行中のレコードの最大数を表します。 関連する統計: Average ディメンション:`PipelineName`

Otel トレースソースメトリクス

次のメトリクスは、OTel トレースソースに適用されます。各メトリクスの先頭に、サブパイプライン名と otel_trace_source が付きます。例えば、*sub_pipeline_name*.otel_trace_source.requestTimeouts.count。

メトリクスのサフィックス	説明
requestTimeouts.count	タイムアウトしたリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
requestsReceived.count	プラグインによって受信されたリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
successRequests.count	プラグインによって正常に処理されたリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
badRequests.count	プラグインによって処理された、無効な形式のリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
requestsTooLarge.count	コンテンツ内のスパンの数がバッファ容量よりも大きいリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
internalServerError.count	カスタムの例外タイプを持つプラグインによって処理されたリクエストの数。

メトリクスのサフィックス	説明
	関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.count	リクエストをプラグインで処理している最中に記録されたデータポイントの数。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.sum	プラグインによって処理されたリクエストの合計レイテンシー (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.max	プラグインによって処理されたリクエストの最大レイテンシー (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
payloadSize.count	受信リクエストのペイロードサイズの分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
payloadSize.sum	受信リクエストのペイロードサイズの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
payloadSize.max	受信リクエストのペイロードサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName

Otel メトリクスソースメトリクス

次のメトリクスは、OTel メトリクスソースに適用されます。各メトリクスの先頭に、サブパイプライン名と otel_metrics_source が付きます。例えば、*sub_pipeline_name*.otel_metrics_source.requestTimeouts.count。

メトリクスのサフィックス	説明
requestTimeouts.count	タイムアウトしたプラグインへのリクエストの合計数。 関連する統計情報: Sum ディメンション:PipelineName
requestsReceived.count	プラグインによって受信されたリクエストの合計数。 関連する統計情報: Sum ディメンション:PipelineName
successRequests.count	プラグインによって正常に処理されたリクエストの数 (200 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.count	プラグインによって処理されたリクエストのレイテンシー数 (秒)。 関連する統計情報: Sum

メトリクスのサフィックス	説明
	ディメンション:PipelineName
requestProcessDuration.sum	プラグインによって処理されたリクエストの合計レイテンシー (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.max	プラグインによって処理されたリクエストの最大レイテンシー (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
payloadSize.count	受信リクエストのペイロードサイズの分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
payloadSize.sum	受信リクエストのペイロードサイズの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
payloadSize.max	受信リクエストのペイロードサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName

HTTP メトリクス

次のメトリクスは、HTTP ソースに適用されます。各メトリクスの先頭に、サブパイプライン名と http が付きます。例えば、*sub_pipeline_name*.http.requestsReceived.count。

メトリクスのサフィックス	説明
requestsReceived.count	/log/ingest エンドポイントが受信したリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
requestsRejected.count	プラグインによって拒否されたリクエストの数 (429 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
successRequests.count	プラグインによって正常に処理されたリクエストの数 (200 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
badRequests.count	プラグインによって処理された、コンテンツタイプまたは形式が無効なリクエストの数 (400 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
requestTimeouts.count	HTTP ソースサーバーでタイムアウトしたリクエストの数 (415 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
requestsTooLarge.count	コンテンツ内のイベントサイズがバッファ容量よりも大きいリクエストの数 (413 応答状態コード)。 関連する統計情報: Sum

メトリクスのサフィックス	説明
	ディメンション:PipelineName
internalServerError.count	カスタムの例外タイプを持つプラグインによって処理されたリクエストの数 (500 応答状態コード)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.count	プラグインによって処理されたリクエストのレイテンシー数 (秒)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.sum	プラグインによって処理されたリクエストの合計レイテンシー (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
requestProcessDuration.max	プラグインによって処理されたリクエストの最大レイテンシー (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
payloadSize.count	受信リクエストのペイロードサイズの分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
payloadSize.sum	受信リクエストのペイロードサイズの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
payloadSize.max	受信リクエストのペイロードサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName

S3 メトリクス

次のメトリクスは、S3 ソースに適用されます。各メトリクスの先頭に、サブパイプライン名と s3 が付きます。例えば、*sub_pipeline_name*.s3.s3ObjectsFailed.count。

メトリクスのサフィックス	説明
s3ObjectsFailed.count	プラグインが読み取れなかった S3 オブジェクトの合計数。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectsNotFound.count	S3 の Not Found エラーによりプラグインが読み取れなかった S3 オブジェクトの数。これらのメトリクスも、s3ObjectsFailed メトリクスとしてカウントされます。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectsAccessDenied.count	S3 の Access Denied または Forbidden エラーによりプラグインが読み取れなかった S3 オブジェクトの数。

メトリクスのサフィックス	説明
	これらのメトリクスも、s3ObjectsFailed メトリクスとしてカウントされます。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectReadTimeElapsed.count	プラグインが S3 オブジェクトの GET リクエストを実行し、これを解析して、イベントをバッファに書き込むまでに要する時間。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectReadTimeElapsed.sum	プラグインが S3 オブジェクトの GET リクエストを実行し、これを解析して、イベントをバッファに書き込むまでに要する合計時間 (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectReadTimeElapsed.max	プラグインが S3 オブジェクトの GET リクエストを実行し、これを解析して、イベントをバッファに書き込むまでに要する最大時間 (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
s3ObjectSizeBytes.count	S3 オブジェクトサイズの分布の数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
s3ObjectSizeBytes.sum	S3 オブジェクトサイズの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectSizeBytes.max	S3 オブジェクトサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName
s3ObjectProcessedBytes.count	プラグインによって処理された S3 オブジェクトの分布の数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectProcessedBytes.sum	プラグインによって処理された S3 オブジェクトの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectProcessedBytes.max	プラグインによって処理された S3 オブジェクトの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName
s3ObjectsEvents.count	プラグインが受信した S3 イベントの分布の数。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
s3ObjectsEvents.sum	プラグインが受信した S3 イベントの合計分布。 関連する統計情報: Sum ディメンション:PipelineName
s3ObjectsEvents.max	プラグインが受信した S3 イベントの最大分布。 関連する統計情報: Max ディメンション:PipelineName
sqsMessageDelay.count	S3 が、オブジェクトの作成からそれが完全に解析されるまでのイベント時間を記録している最中に、記録されたデータポイントの数。 関連する統計情報: Sum ディメンション:PipelineName
sqsMessageDelay.sum	S3 が、オブジェクトの作成からそれが完全に解析されるまでのイベント時間を記録している間の合計時間 (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
sqsMessageDelay.max	S3 が、オブジェクトの作成からそれが完全に解析されるまでのイベント時間を記録している間の最大時間 (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName

メトリクスのサフィックス	説明
s3ObjectsSucceeded.count	プラグインが正常に読み取った S3 オブジェクトの数。 関連する統計情報: Sum ディメンション:PipelineName
sqsMessagesReceived.count	プラグインがキューから受信した Amazon SQS メッセージの数。 関連する統計情報: Sum ディメンション:PipelineName
sqsMessagesDeleted.count	プラグインがキューから削除した Amazon SQS メッセージの数。 関連する統計情報: Sum ディメンション:PipelineName
sqsMessagesFailed.count	プラグインが解析に失敗した Amazon SQS メッセージの数。 関連する統計情報: Sum ディメンション:PipelineName

メトリクス集約

次のメトリクスは、Aggregate プロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と aggregate が付きます。例えば、*sub_pipeline_name*.aggregate.actionHandleEventsOut.count。

メトリクスのサフィックス	説明
actionHandleEventsOut.count	handleEvent 呼び出しから設定済みアクションに返されたイベントの数。

メトリクスのサフィックス	説明
	関連する統計情報: Sum ディメンション:PipelineName
actionHandleEvents Dropped.count	handleEvent 呼び出しから設定済みアクションに返されたイベントの数。 関連する統計情報: Sum ディメンション:PipelineName
actionHandleEvents ProcessingErrors.count	設定済みアクションのために handleEvent に対して実行され、エラーになった呼び出しの数。 関連する統計情報: Sum ディメンション:PipelineName
actionConcludeGrou pEventsOut.count	concludeGroup 呼び出しから設定済みアクションに返されたイベントの数。 関連する統計情報: Sum ディメンション:PipelineName
actionConcludeGrou pEventsDropped.count	condludeGroup 呼び出しから設定済みアクションに返されなかったイベントの数。 関連する統計情報: Sum ディメンション:PipelineName
actionConcludeGrou pEventsProcessingE rrors.count	設定済みアクションのために concludeGroup に対して実行され、エラーになった呼び出しの数。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
currentAggregateGroups.value	現在のグループの数。この数値は、グループが完了すると減り、イベントが新しいグループの作成を始めると増えます。 関連する統計: Average ディメンション:PipelineName

日付メトリクス

次のメトリクスは、Date プロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と date が付きます。例えば、*sub_pipeline_name*.date.dateProcessingMatchSuccess.count。

メトリクスのサフィックス	説明
dateProcessingMatchSuccess.count	match の設定オプションで指定されたパターンのうちの 1 つ以上に一致しているレコード数。 関連する統計情報: Sum ディメンション:PipelineName
dateProcessingMatchFailure.count	match の設定オプションで指定されたパターンのいずれにも一致していなかったレコード数。 関連する統計情報: Sum ディメンション:PipelineName

Lambda metrics

次のメトリクスは、AWS Lambda プロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と lambda が付きます。例えば、*sub_pipeline_name*.lambda.recordsSuccessfullySentToLambda.count。

メトリクスのサフィックス	説明
recordsSuccessfullySentToLambda.count	Lambda 関数によって正常に処理されたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsFailedToSendToLambda.count	Lambda 関数に送信できなかったレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
lambdaFunctionLatency.avg lambdaFunctionLatency.max	Lambda 関数呼び出しのレイテンシー。 関連する統計情報: Average と Maximum ディメンション:PipelineName
numberOfRequestsSucceeded.count	成功した Lambda 呼び出しリクエストの総数。 関連する統計情報: Sum ディメンション:PipelineName
numberOfRequestsFailed.count	失敗した Lambda 呼び出しリクエストの総数。 関連する統計情報: Sum ディメンション:PipelineName
requestPayloadSize.avg	Lambda に送信されるリクエストペイロードのサイズ。 関連する統計: Average ディメンション:PipelineName
responsePayloadSize.avg	Lambda から受信したレスポンスペイロードのサイズ。 関連する統計: Average

メトリクスのサフィックス	説明
	ディメンション:PipelineName

Grok メトリクス

次のメトリクスは、Grok プロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と grok が付きます。例えば、*sub_pipeline_name*.grok.grokProcessingMatch.count。

メトリクスのサフィックス	説明
grokProcessingMatch.count	match の設定オプションのうち、1 つ以上と一致していることがわかったレコード数。 関連する統計情報: Sum ディメンション:PipelineName
grokProcessingMismatch.count	match の設定オプションで指定されたパターンのいずれにも一致していなかったレコード数。 関連する統計情報: Sum ディメンション:PipelineName
grokProcessingErrors.count	レコード処理エラーの数。 関連する統計情報: Sum ディメンション:PipelineName
grokProcessingTimeouts.count	マッチング中にタイムアウトしたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
grokProcessingTime.count	個々のレコードが match 設定オプションのパターンと照合している間に記録されたデータポイントの数。 関連する統計情報: Sum

メトリクスのサフィックス	説明
	ディメンション:PipelineName
grokProcessingTime.sum	個々のレコードが match 設定オプションのパターンとの照合に要する合計時間 (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
grokProcessingTime.max	個々のレコードが match 設定オプションのパターンとの照合に要する最大時間 (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName

Otel トレース raw メトリクス

次のメトリクスは、OTel トレース raw プロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と otel_trace_raw が付きます。例えば、*sub_pipeline_name*.otel_trace_raw.traceGroupCacheCount.value。

メトリクスのサフィックス	説明
traceGroupCacheCount.value	トレースグループキャッシュ内のトレースグループの数。 関連する統計情報: Sum ディメンション:PipelineName
spanSetCount.value	スパンセットコレクション内のスパンセットの数。 関連する統計情報: Sum ディメンション:PipelineName

Otel トレースグループメトリクス

次のメトリクスは、OTel トレースグループプロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と otel_trace_group が付きます。例えば、*sub_pipeline_name*.otel_trace_group.recordsInMissingTraceGroup.count。

メトリクスのサフィックス	説明
recordsInMissingTraceGroup.count	トレースグループフィールドが欠落しているイングレスレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsOutFixedTraceGroup.count	トレースグループフィールドが正常に入力されているエグレスレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
recordsOutMissingTraceGroup.count	トレースグループフィールドが欠落しているエグレスレコードの数。 関連する統計情報: Sum ディメンション:PipelineName

サービスマップステートフルメトリクス

次のメトリクスは、サービスマップステートフルプロセッサに適用されます。各メトリクスの先頭に、サブパイプライン名と service-map-stateful が付きます。例えば、*sub_pipeline_name*.service-map-stateful.spansDbSize.count。

メトリクスのサフィックス	説明
spansDbSize.value	現在と前のウィンドウ期間における MapDB のスパンのメモリ内バイトサイズ。

メトリクスのサフィックス	説明
	関連する統計: Average ディメンション:PipelineName
traceGroupDbSize.value	現在と前のウィンドウ期間における MapDB のトレースグループのメモリ内バイトサイズ。 関連する統計: Average ディメンション:PipelineName
spansDbCount.value	現在と前のウィンドウ期間における MapDB のスパンの数。 関連する統計情報: Sum ディメンション:PipelineName
traceGroupDbCount.value	現在と前のウィンドウ期間における MapDB のトレースグループの数。 関連する統計情報: Sum ディメンション:PipelineName
relationshipCount.value	現在と前のウィンドウ期間に保存された関係の数。 関連する統計情報: Sum ディメンション:PipelineName

OpenSearch メトリクス

次のメトリクスは、OpenSearch シンクに適用されます。各メトリクスの先頭に、サブパイプライン名と opensearch が付きます。例えば、*sub_pipeline_name*.opensearch.bulkRequestErrors.count。

メトリクスのサフィックス	説明
bulkRequestErrors.count	一括リクエストの送信中に発生したエラーの合計数。 関連する統計情報: Sum ディメンション:PipelineName
documentsSuccess.count	一括リクエストにより OpenSearch Service に正常に送信されたドキュメントの数 (再試行を含む)。 関連する統計情報: Sum ディメンション:PipelineName
documentsSuccessFirstAttempt.count	一括リクエストにより OpenSearch Service に一回で正常に送信されたドキュメントの数。 関連する統計情報: Sum ディメンション:PipelineName
documentErrors.count	一括リクエストで送信できなかったドキュメントの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestFailed.count	送信できなかった一括リクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestNumberOfRetries.count	失敗した一括リクエストの再試行数。 関連する統計情報: Sum ディメンション:PipelineName
bulkBadRequestErrors.count	一括リクエストの送信中に発生した Bad Request エラーの数。

メトリクスのサフィックス	説明
	関連する統計情報: Sum ディメンション:PipelineName
bulkRequestNotAllowedErrors.count	一括リクエストの送信中に発生した Request Not Allowed エラーの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestInvalidInputErrors.count	一括リクエストの送信中に発生した Invalid Input エラーの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestNotFoundErrors.count	一括リクエストの送信中に発生した Request Not Found エラーの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestTimeoutErrors.count	一括リクエストの送信中に発生した Request Timeout エラーの数。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestServerErrors.count	一括リクエストの送信中に発生した Server Error エラーの数。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
bulkRequestSizeBytes.count	一括リクエストのペイロードサイズの分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestSizeBytes.sum	一括リクエストのペイロードサイズの合計分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestSizeBytes.max	一括リクエストのペイロードサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName
bulkRequestLatency.count	リクエストがプラグインに送信されている間に記録されたデータポイントの数 (再試行含む)。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestLatency.sum	プラグインに送信されたリクエストの合計レイテンシー (再試行含む) (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
bulkRequestLatency.max	プラグインに送信されたリクエストの最大レイテンシー (再試行含む) (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName

メトリクスのサフィックス	説明
s3.dlqS3RecordsSuccess.count	S3 デッドレターキューに正常に送信されたレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RecordsFailed.count	S3 デッドレターキューへの送信に失敗したレコードの数。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestSuccess.count	S3 デッドレターキューに正常に送信されたリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestFailed.count	S3 デッドレターキューへの送信に失敗したリクエストの数。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestLatency.count	リクエストが S3 デッドレターキューに送信されている間に記録されたデータポイントの数 (再試行含む)。 関連する統計情報: Sum ディメンション:PipelineName

メトリクスのサフィックス	説明
s3.dlqS3RequestLatency.sum	S3 デッドレターキューに送信されたリクエストの合計レイテンシー (再試行含む) (ミリ秒)。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestLatency.max	S3 デッドレターキューに送信されたリクエストの最大レイテンシー (再試行含む) (ミリ秒)。 関連する統計情報: Max ディメンション:PipelineName
s3.dlqS3RequestSizeBytes.count	S3 デッドレターキューへのリクエストの、ペイロードサイズの分布数 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestSizeBytes.sum	S3 デッドレターキューへのリクエストの、ペイロードサイズの合計分布 (バイト)。 関連する統計情報: Sum ディメンション:PipelineName
s3.dlqS3RequestSizeBytes.max	S3 デッドレターキューへのリクエストの、ペイロードサイズの最大分布 (バイト)。 関連する統計情報: Max ディメンション:PipelineName

システムと測定メトリクス

次のメトリクスは、OpenSearch Ingestion システム全体に適用されます。これらのメトリクスの前には、何も付いていません。

メトリクス	説明
system.cpu.usage.value	すべてのデータノードの、使用可能な CPU 使用量の割合。 関連する統計: Average ディメンション: PipelineName 、area、id
system.cpu.count.value	すべてのデータノードの CPU 使用量の合計。 関連する統計: Average ディメンション: PipelineName 、area、id
jvm.memory.max.value	メモリ管理に使用できるメモリの最大数 (バイト)。 関連する統計: Average ディメンション: PipelineName 、area、id
jvm.memory.used.value	使用されているメモリの合計量 (バイト)。 関連する統計: Average ディメンション: PipelineName 、area、id signa
jvm.memory.committed.value	Java 仮想マシン (JVM) で使用するためにコミットされたメモリーの量 (バイト)。 関連する統計: Average ディメンション: PipelineName 、area、id
computeUnits	パイプラインで使用されている Ingestion OpenSearch Compute Units (Ingestion OCU) の数。 関連する統計: Max、Sum、Average ディメンション:PipelineName

Amazon OpenSearch Ingestion のベストプラクティス

この章では、Amazon OpenSearch Ingestion パイプラインを作成して管理するときのベストプラクティスと、多くのユースケースに適用される一般的なガイドラインを提供します。各ワークロードは一意で、固有の特性があるため、すべてのユースケースに完全に適した一般的な推奨事項はありません。

トピック

- 一般的なベストプラクティス
- 推奨される CloudWatch アラーム

一般的なベストプラクティス

パイプラインの作成と管理には、次の一般的なベストプラクティスが適用されます。

- 高可用性を確保するために、VPC パイプラインを 2 つまたは 3 つのサブネットで構成します。パイプラインを 1 つのサブネットにのみデプロイした場合、そのアベイラビリティーゾーンがダウンするとデータを取り込めなくなります。
- 各パイプライン内では、サブパイプラインの数を 5 つ以下に制限することをお勧めします。
- S3 ソースプラグインを使用している場合は、パフォーマンスを最適化するために均等なサイズの S3 ファイルを使用します。
- S3 ソースプラグインを使用している場合は、パフォーマンスを最適化するために、S3 バケットのファイルサイズ 0.25 GB ごとに 30 秒の可視性タイムアウトを追加します。
- パイプライン設定にデッドレターキュー (DLQ) を含めると、失敗したイベントをオフロードし、分析のためにアクセスできるようになります。誤ったマッピングや他の問題によりシンクがデータを拒否した場合、トラブルシューティングして問題を解決するために、データを DLQ にルーティングできます。
- パイプラインでアグリゲートプロセッサを使用している場合は、パイプラインのパフォーマンスを最適化するために “`local_mode: true`”フラグを使用することをお勧めします。

推奨される CloudWatch アラーム

CloudWatch アラームは、CloudWatch メトリクスがある程度の時間にわたって指定された値を超えたときにアクションを実行します。たとえば、クラスター AWS のヘルスステータスが `red`1 分以上

の場合は、E メールで送信できます。このセクションでは、Amazon OpenSearch Ingestion で推奨されるいくつかのアラームとそのアラームへの対応方法について説明します。

アラームの設定の詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch でのアラームの使用」を参照してください。

アラーム	問題
computeUnits maximum is = the configured maxUnits for 15 minute, 3 consecutive times	パイプラインが最大キャパシティに達したため、maxUnits の更新が必要になります。パイプラインの最大キャパシティを増やしてください。
opensearch.documentErrors.count sum is = *{sub_pipeline_name}* .opensearch.recordsIn.count sum for 1 minute, 1 consecutive time	パイプラインから OpenSearch シンクに書き込めません。パイプラインのアクセス許可をチェックし、ドメインまたはコレクションが正常であることを確かめます。デッドレターキュー (DLQ) が設定されている場合は、失敗したイベントがないかどうかを確認することもできます。
bulkRequestLatency.max max is >= x for 1 minute, 1 consecutive time	パイプラインで OpenSearch シンクにデータを送信するときに高いレイテンシーが発生しています。これは、シンクのサイズが小さすぎるか、シャーディング戦略が不十分でシンクの処理が遅れていることが原因と考えられます。高いレイテンシーが続くとパイプラインのパフォーマンスに影響が及び、クライアントのバックプレッシャーにつながる可能性があります。
httpAuthFailure.count sum >= 1 for 1 minute, 1 consecutive time	取り込みリクエストが認証されていません。すべてのクライアントで Signature Version 4 認証が正しく有効になっていることを確認します。

アラーム	問題
system.cpu.usage.value average >= 80% for 15 minutes, 3 consecutive times	CPU 使用率の高い状態が続くと、問題が生じる可能性があります。パイプラインの最大キャパシティを増やすことを検討してください。
bufferUsage.value average >= 80% for 15 minutes, 3 consecutive times	バッファ使用量の多い状態が続くと、問題が生じる可能性があります。パイプラインの最大キャパシティを増やすことを検討してください。

検討した方が良いその他のアラーム

定期的に使用する Amazon OpenSearch Ingestion の機能に応じて、次のアラームの設定を検討します。

アラーム	問題
dynamodb.exportJobFailure.count sum 1	Amazon S3 へのエクスポートのトリガーの試行が失敗しました。
opensearch.EndtoEndLatency.avg average > X(15 分間)、連続 4 回	EndtoEndLatency が DynamoDB ストリームからの読み取りに必要な値よりも大きくなっています。これは、OpenSearch クラスターのスケールが不十分であるか、またはパイプライン OCU の最大キャパシティが DynamoDB テーブルの WCU スループットに対して少なすぎることが原因である可能性があります。EndtoEndLatency は、エクスポート後には大きくなりますが、時間の経過とともに最新の DynamoDB ストリームに追いつくにつれて小さくなるはずです。
dynamodb.changeEventsProces	DynamoDB ストリームからレコードは収集されていません。これは、テーブル上でアクティビティがないこと、または DynamoDB ストリームに対するアクセスに関する問題が原因である可能性があります。

アラーム	問題
sed.count sum == 0 (X 分間)	
opensearch.s3.dlqS3RecordsSuccess.count sum >= opensearch.documentSuccess.count sum for 1 minute, 1 consecutive time	OpenSearch シンクよりも多くのレコードが DLQ に送信されています。OpenSearch シンクプラグインのメトリクスを確認し、根本原因を調査して特定してください。
grok.grokProcessingTimeouts.count sum = recordsIn.count sum for 1 minute, 5 consecutive times	Grok プロセッサがパターンマッチングを試みている間、すべてのデータがタイムアウトしています。これはパフォーマンスに影響を与え、パイプラインの速度を低下させている可能性があります。タイムアウトを減らすために、パターンの調整を検討してください。
grok.grokProcessingErrors.count sum is >= 1 for 1 minute, 1 consecutive time	Grok プロセッサがパイプライン内のデータとのパターンマッチングに失敗したため、エラーが発生しています。データと Grok プラグインの設定を確認し、パターンマッチングが想定どおりであることを確認してください。
grok.grokProcessingMismatch.count sum = recordsIn.count sum for 1 minute, 5 consecutive times	Grok プロセッサは、パイプライン内のデータにパターンを一致させることができません。データと Grok プラグインの設定を確認し、パターンマッチングが想定どおりであることを確認してください。

アラーム	問題
date.dateProcessingMatchFailure.count sum = recordsIn.count sum for 1 minut, 5 consecutive times	Date プロセッサは、パイプライン内のデータにどのパターンも一致させることができません。データと Date プラグインの設定を確認し、パターンが想定どおりであることを確認してください。
s3.s3ObjectsFailed.count sum >= 1 for 1 minute, 1 consecutive time	この問題は、S3 オブジェクトが存在しないか、パイプラインのアクセス許可が不十分であるために発生しています。s3ObjectsNotFound.count メトリクスと s3ObjectsAccessDenied.count メトリクスを確認し、根本原因を特定してください。S3 オブジェクトが存在することを確認するか、アクセス許可を更新します。
s3.sqsMessagesFailed.count sum >= 1 for 1 minute, 1 consecutive time	S3 プラグインは Amazon SQS メッセージの処理に失敗しました。SQS キューで DLQ が有効になっている場合、失敗のメッセージを確認してください。パイプラインが処理しようとしている無効なデータをキューが受け取っている可能性があります。
http.badRequests.count sum >= 1 for 1 minute, 1 consecutive times	クライアントが不正なリクエストを送信しています。すべてのクライアントが適切なペイロードを送信していることを確認してください。
http.requestsTooLarge.count sum >= 1 for 1 minute, 1 consecutive time	HTTP ソースプラグインからのリクエストに含まれるデータが多すぎるため、バッファ容量を超えています。クライアントのバッチサイズを調整してください。

アラーム	問題
`http.internalServerError.count` sum >= 0 for 1 minute, 1 consecutive time	HTTP ソースプラグインに問題が発生し、イベントを受信できません。
`http.requestTimeouts.count` sum >= 0 for 1 minute, 1 consecutive time	ソースタイムアウトは、パイプラインのプロビジョニングが不十分であることが原因と考えられます。追加のワークロードを処理するために、パイプラインの `maxUnits` を増やすことを検討してください。
`otel_trace.badRequests.count` sum >= 1 for 1 minute, 1 consecutive time	クライアントが不正なリクエストを送信しています。すべてのクライアントが適切なペイロードを送信していることを確認してください。
`otel_trace.requestsTooLarge.count` sum >= 1 for 1 minute, 1 consecutive time	Otel Trace ソースプラグインからのリクエストに含まれるデータが多すぎるため、バッファ容量を超えています。クライアントのバッチサイズを調整してください。
`otel_trace.internalServerError.count` sum >= 0 for 1 minute, 1 consecutive time	Otel Trace ソースプラグインに問題が発生し、イベントを受信できません。

アラーム	問題
otel_trace.requestTimeouts.count sum >= 0 for 1 minute, 1 consecutive time	ソースタイムアウトは、パイプラインのプロビジョニングが不十分であることが原因と考えられます。追加のワークロードを処理するために、パイプラインの maxUnits を増やすことを検討してください。
otel_metrics.requestTimeouts.count sum >= 0 for 1 minute, 1 consecutive time	ソースタイムアウトは、パイプラインのプロビジョニングが不十分であることが原因と考えられます。追加のワークロードを処理するために、パイプラインの maxUnits を増やすことを検討してください。

Amazon OpenSearch Serverless

Amazon OpenSearch Serverless は、Amazon OpenSearch Service 用のオンデマンド自動スケーリング設定です。手動の容量管理を必要とするプロビジョニングされた OpenSearch ドメインとは異なり、OpenSearch Serverless コレクションはアプリケーションのニーズに基づいてコンピューティングリソースを自動的にスケーリングします。

OpenSearch Serverless は、低頻度、断続的、または予測不可能なワークロードに対して費用対効果の高いソリューションを提供します。アプリケーションの使用状況に基づいてコンピューティング容量を自動的にスケーリングすることで、コストを最適化します。Serverless コレクションは、プロビジョニングされた OpenSearch Service ドメインと同じタイプの、大容量かつ分散型で可用性の高いストレージボリュームを使用しています。

OpenSearch Serverless コレクションは常に暗号化されます。暗号化キーは選択できますが、暗号化を無効にすることはできません。詳細については、「the section called “暗号化”」を参照してください。

利点

OpenSearch Serverless には次の利点があります。

- プロビジョニングよりもシンプル – OpenSearch Serverless では、OpenSearch クラスターおよびキャパシティーを管理する上での複雑さが大幅に解消されます。クラスターのサイズ設定および調整を自動的に行い、シャードとインデックスのライフサイクルを管理します。また、サービスソフトウェアの更新や OpenSearch のバージョンアップグレードも管理します。すべての更新とアップグレードは中断されません。
- コスト効率が高い – OpenSearch Serverless を使用する場合、消費したリソースに対してのみ料金が発生します。これにより、ピーク時のワークロードに対する事前のプロビジョニングやオーバープロビジョニングが不要になります。
- 高可用性 – OpenSearch Serverless は、アベイラビリティーゾーンの停止やインフラストラクチャ障害から保護するための冗長性を備えた本番ワークロードをサポートしています。
- スケーラブル – OpenSearch Serverless はリソースを自動的にスケーリングして、一貫して高速なデータインジェスト率とクエリ応答時間を維持します。

Amazon OpenSearch Serverless とは

Amazon OpenSearch Serverless は、Amazon OpenSearch Service のオンデマンドのサーバーレスオプションであり、OpenSearch クラスターのプロビジョニング、設定、チューニングといった複雑な運用を排除します。クラスターを自己管理したくない組織や、大規模なデプロイの運用に必要となる専用リソースや高度な知識を持たない組織に最適です。OpenSearch Serverless を使用することで、基盤となるインフラストラクチャを管理することなく、大容量のデータを検索して分析できます。

OpenSearch Serverless コレクションとは、特定のワークロードやユースケースをサポートするために連携して機能する OpenSearch インデックスのグループを指します。コレクションは、手動のプロビジョニングが必要なセルフマネージド型の OpenSearch クラスターと比べて運用を簡素化します。

コレクションは、プロビジョニングされた OpenSearch Service ドメインと同じタイプの、大容量かつ分散型で可用性の高いストレージボリュームを使用しますが、手動による設定やチューニング操作を減らすことで複雑さが大幅に解消されます。コレクション内のデータは、転送時に暗号化されます。OpenSearch Serverless は、データを分析するためのインターフェイスである OpenSearch Dashboards もサポートしています。

OpenSearch Serverless はオープンソースの OpenSearch と互換性があります。新しいバージョンがリリースされると、OpenSearch Serverless はコレクションを自動的にアップグレードして、新しい機能、バグ修正、パフォーマンス強化を組み込みます。

OpenSearch Serverless では OpenSearch オープンソーススイートと同じ取り込みオペレーションやクエリ API オペレーションがサポートされているため、既存のクライアントやアプリケーションを引き続き使用できます。OpenSearch Serverless を使用するには、クライアントが OpenSearch 3.x と互換性がある必要があります。詳細については、「the section called “コレクションへのデータの取り込み”」を参照してください。

トピック

- OpenSearch Serverless のユースケース
- 仕組み
- コレクションタイプを選択する
- サポートされている AWS リージョン
- 制限事項
- OpenSearch Service と OpenSearch Serverless を比較する

OpenSearch Serverless のユースケース

OpenSearch Serverless は、主に次の 2 つのユースケースをサポートしています。

- ログ分析 – ログ分析セグメントは、オペレーションインサイトとユーザーの行動に関するインサイトを得るために、マシン生成による大量の半構造化された時系列データを分析することに重点を置きます。
- 全文検索 – 全文検索セグメントが、社内ネットワーク内のアプリケーション (コンテンツ管理システム、法的文書) や、e コマースウェブサイトのコンテンツ検索などのインターネット向けアプリケーションを強化します。

コレクションを作成するときは、これらのユースケースのいずれかを選択します。詳細については、「the section called “コレクションタイプを選択する”」を参照してください。

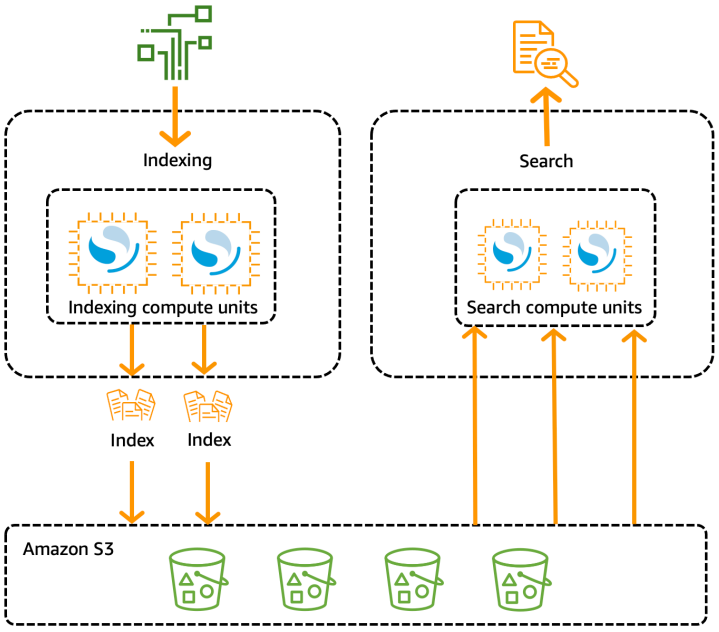
仕組み

従来の OpenSearch クラスターには、インデックス作成オペレーションと検索オペレーションの両方を実行する単一のインスタンスセットがあり、インデックスストレージはコンピューティング性能と密接に結びついています。これに対し、OpenSearch Serverless はインデックス作成 (取り込み) コンポーネントを検索 (クエリ) コンポーネントから分離するクラウドネイティブなアーキテクチャを採用しており、Amazon S3 をインデックスのプライマリデータストレージとして使用します。

この分離されたアーキテクチャでは、検索機能とインデックス作成機能を相互に独立して、さらに S3 のインデックス化されたデータとも無関係にスケーリングできます。また、このアーキテクチャでは、取り込みオペレーションとクエリオペレーションを分離できるため、リソースを競合させることなく同時に実行できます。

コレクションにデータを書き込むと、OpenSearch Serverless はそのデータをインデックス作成用コンピューティングユニットに配信します。インデックス作成用コンピューティングユニットは、受信データを取り込み、インデックスを S3 に移動します。コレクションデータに対して検索を実行すると、OpenSearch Serverless はクエリ実行対象のデータを保持する検索用コンピューティングユニットにリクエストをルーティングします。検索コンピューティングユニットは、インデックス化されたデータを S3 から直接ダウンロードし (まだローカルにキャッシュされていない場合)、検索オペレーションを実行し、集計を実行します。

次の図は、この分離されたアーキテクチャを示しています。



データインジェスト、検索、およびクエリの実行に必要な OpenSearch Serverless のコンピューティング性能は、OpenSearch Compute Unit (OCU) で測定されます。各 OCU は、6 GiB のメモリと対応する仮想 CPU (vCPU)、および Amazon S3 へのデータ転送を組み合わせたものです。

OpenSearch Serverless OCUs を個別にプロビジョニングします。OpenSearch Serverless OCUs を追加します。 ???キャパシティーは、コンピューティング使用量の減少に伴ってスケールダウンされます。

これらの OCUs」を参照してください。 OpenSearch

コレクションタイプを選択する

OpenSearch Serverless は、主に次の 3 つのコレクションタイプをサポートしています。

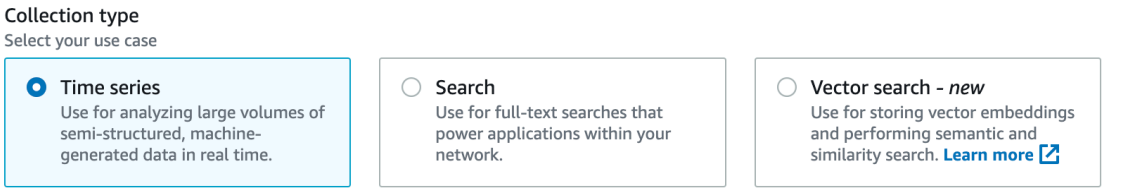
[時系列] – 大量の半構造化マシン生成データをリアルタイムで分析し、運用、セキュリティ、ユーザー行動、およびビジネスパフォーマンスに関するインサイトを提供するログ分析セグメント。

> **Note**
>
> 時系列コレクションは Classic コレクションでのみ使用できます。NextGen コレクションは現在、検索タイプとベクトル検索タイプのみをサポートしています。

[検索] – 社内ネットワーク内のアプリケーション (コンテンツ管理システム、法的ドキュメントレポジトリ) や、e コマースウェブサイト検索やコンテンツ検出などのインターネット向けアプリケーションをサポートする全文検索機能。

[ベクトル検索] – ベクトル埋め込みのセマンティック検索は、ベクトルデータ管理を簡素化し、機械学習 (ML) で拡張された検索エクスペリエンスを実現します。チャットボット、パーソナルアシスタント、不正検出などの生成 AI アプリケーションをサポートします。

コレクションを初めて作成するときに、コレクションタイプを選択します。



選択するコレクションタイプは、コレクションに取り込む予定のデータの種類と、そのデータへのクエリ実行方法によって異なります。コレクションの作成後にコレクションタイプを変更することはできません。

コレクションタイプには、次の大きな違いがあります。

- 検索およびベクトル検索コレクションでは、迅速なクエリ応答時間を確保するため、すべてのデータがホットストレージに保存されます。時系列コレクションは、ホットストレージとウォームストレージを組み合わせて使用します。この場合、より頻繁にアクセスされるデータのクエリ応答時間を最適化するために、最新のデータがホットストレージに保存されます。
- 時系列コレクションの場合、カスタムドキュメント ID でインデックスを作成したり、アップサートリクエストで更新したりすることはできません。この操作は検索のユースケース専用です。代わりにドキュメント ID を使用して更新できます。詳細については、「the section called “サポートされる OpenSearch API オペレーションとアクセス許可”」を参照してください。
- 検索および時系列コレクションの場合、k-NN タイプのインデックスは使用できません。

サポートされている AWS リージョン

OpenSearch Serverless は、OpenSearch Service AWS リージョン が利用可能な のサブセットで使用できます。サポートされているリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon OpenSearch Service エンドポイントとクォータ」を参照してください。

制限事項

OpenSearch Serverless には次の制約事項があります。

- 一部の OpenSearch API オペレーションがサポートされていません。「the section called “サポートされる OpenSearch API オペレーションとアクセス許可”」を参照してください。
- 一部の OpenSearch プラグインがサポートされていません。「the section called “サポートされている OpenSearch プラグイン”」を参照してください。
- 現在、マネージド型の OpenSearch Service ドメインから Serverless コレクションにデータを自動的に移行する方法はありません。ドメインからコレクションにデータを再インデックスする必要があります。
- コレクションへのクロスアカウントアクセスはサポートされていません。他のアカウントからのコレクションを暗号化またはデータアクセスポリシーに含めることはできません。
- カスタムの OpenSearch プラグインはサポートされていません。
- 自動スナップショットは、OpenSearch Serverless コレクションでサポートされています。手動スナップショットはサポートされていません。詳細については、「the section called “スナップショットを使用したコレクションのバックアップ”」を参照してください。
- クロスリージョン検索およびレプリケーションはサポートされていません。
- 1 つのアカウントおよびリージョンで使用できるサーバーレスリソースの数には制限があります。「OpenSearch Serverless のクォータ」を参照してください。
- 検索コレクションと時系列コレクションのインデックスの更新間隔は約 10 秒です。
- シャードの数、間隔の数、更新間隔は変更できず、OpenSearch サーバーレス によって処理されます。シャーディング戦略はコレクションタイプとトラフィックに基づいています。例えば、時系列コレクションでは、書き込みトラフィックのボトルネックに基づいてプライマリシャードをスケーリングします。

OpenSearch Service と OpenSearch Serverless を比較する

OpenSearch Serverless は、プロビジョニングされた OpenSearch Service ドメインのそれぞれ対応する機能と比較すると、概念や機能の一部が異なります。例えば、重要な違いの 1 つは、OpenSearch Serverless にはクラスターやノードという概念がないことです。

次の表では、OpenSearch Serverless の重要な機能と概念が、プロビジョニングされた OpenSearch Service ドメインの同等の機能とどのように異なるかについて説明します。

機能	OpenSearch Service	OpenSearch サーバーレス
ドメインとコレクション	インデックスは、事前にプロビジョニングされた OpenSearch クラスターであるドメインに保持されます。 詳細については、「ドメインの作成と管理」を参照してください。	インデックスはコレクションに保持されます。コレクションとは、特定のワークロードやユースケースを表すインデックスを論理的にグループ化したものです。 詳細については、「the section called “コレクションの管理”」を参照してください。
ノードタイプとキャパシティー管理	コストとパフォーマンスの仕様を満たすノードタイプを使用してクラスターを構築します。独自のストレージ要件を計算し、ドメインのインスタンスタイプを選択する必要があります。 詳細については、「the section called “ドメインのサイジング”」を参照してください。	OpenSearch Serverless は、キャパシティー使用状況に基づいて、アカウントの追加のコンピューティングユニットを自動的にスケーリングおよびプロビジョニングします。 詳細については、「the section called “キャパシティ制限の管理”」を参照してください。
「請求」	1 時間ごとの EC2 インスタンスの使用およびインスタンスにアタッチされたすべての EBS ストレージボリュームの累積サイズに対して料金が発生します。 詳細については、「the section called “料金”」を参照してください。	データインジェストのコンピューティング、検索とクエリのコンピューティング、および S3 に保持されるストレージに対して、OCU 時間単位で課金されます。 詳細については、「Amazon OpenSearch Service の料金」を参照してください。
暗号化	ドメインについては、保管時の暗号化はオプションとなります。	コレクションについては、保管時の暗号化は必須となります。

機能	OpenSearch Service	OpenSearch サーバーレス
	詳細については、「the section called “保管中の暗号化”」を参照してください。	詳細については、「the section called “暗号化”」を参照してください。
データアクセスコントロール	ドメイン内のデータへのアクセスは、IAM ポリシーときめ細かなアクセスコントロールによって決定されます。	コレクション内のデータへのアクセスは、データアクセスポリシーによって決定されます。
サポートされている OpenSearch オペレーション	OpenSearch Service では、すべての OpenSearch API オペレーションのサブセットがサポートされています。 詳細については、「the section called “サポートされているオペレーション”」を参照してください。	OpenSearch Serverless は OpenSearch API オペレーションの異なるサブセットをサポートしています。 詳細については、「the section called “サポートされているオペレーションとプラグイン”」を参照してください。
Dashboards サインイン	ユーザー名とパスワードを使用してサインインします。 詳細については、「the section called “マスターユーザーとして OpenSearch Dashboards にアクセスする”」を参照してください。	AWS コンソールにログインし、ダッシュボード URL に移動すると、自動的にログインします。 詳細については、「the section called “OpenSearch Dashboards へのアクセス”」を参照してください。
API	OpenSearch Service API オペレーションを使用して、OpenSearch Service とプログラムでやり取りします。	OpenSearch Serverless API オペレーションを使用して、OpenSearch Serverless とプログラムでやり取りします。

機能	OpenSearch Service	OpenSearch サーバーレス
ネットワークアクセス	ドメインのネットワーク設定は、OpenSearch Dashboards エンドポイントだけでなくドメインエンドポイントにも適用されます。両方のネットワークアクセスは密接に連携しています。	ドメインエンドポイントと OpenSearch Dashboards エンドポイントのネットワーク設定は分離されています。OpenSearch Dashboards のネットワークアクセスを設定しないことも可能です。 詳細については、「the section called “ネットワークアクセス”」を参照してください。
リクエストへの署名	OpenSearch 高レベルおよび低レベル REST クライアントを使用してリクエストに署名します。サービス名を `es` と指定してください。	現時点では、OpenSearch Serverless は OpenSearch Service でサポートされているクライアントのサブセットをサポートしています。 リクエストに署名するときは、サービス名を `aoss` と指定します。`x-amz-content-sha256` ヘッダーは必須です。詳細については、「the section called “その他のクライアント”」を参照してください。
OpenSearch のバージョンアップグレード	OpenSearch の新しいバージョンが利用可能になったら、ドメインを手動でアップグレードしてください。ユーザーには、ドメインがアップグレード要件を満たしていること、および重要な変更点すべてに対処したことを確認する責任があります。	OpenSearch Serverless は、コレクションを新しい OpenSearch バージョンに自動的にアップグレードします。新しいバージョンがリリースされるとすぐにアップグレードが行われるとは限りません。
サービスソフトウェア更新	サービスソフトウェア更新が利用可能になったら、手動でドメインに適用してください。	OpenSearch Serverless はコレクションを自動的に更新して、最新のバグ修正、機能、およびパフォーマンスの改善を反映させます。

機能	OpenSearch Service	OpenSearch サーバーレス
VPC アクセス	VPC 内でドメインをプロビジョニングできます。 OpenSearch Service で管理される VPC エンドポイントを追加で作成して、ドメインにアクセスすることもできます。	アカウントに OpenSearch Serverless で管理される VPC エンドポイントを 1 つまたは複数作成してください。次に、これらのエンドポイントをネットワークポリシー内に含めます。
SAML 認証	SAML 認証はドメインごとに有効にします。 詳細については、「the section called “OpenSearch Dashboards の SAML 認証”」を参照してください。	アカウントレベルで 1 つまたは複数の SAML プロバイダーを設定し、関連するユーザー ID とグループ ID をデータアクセスポリシーに含めます。 詳細については、「the section called “SAML 認証”」を参照してください。
Transport Layer Security (TLS)	OpenSearch Service は TLS 1.2 をサポートしていますが、TLS 1.3 を使用することをお勧めします。	OpenSearch Serverless は TLS 1.2 をサポートしていますが、TLS 1.3 を使用することをお勧めします。

チュートリアル: Amazon OpenSearch Serverless の開始方法

このチュートリアルでは、Amazon OpenSearch Serverless 検索コレクションをすばやく起動して実行するための基本的な手順を示します。検索コレクションを使用すると、内部ネットワーク内のアプリケーションや、e コマースウェブサイト検索やコンテンツ検索などのインターネット向けアプリケーションを強化できます。

ベクトル検索コレクションの使用方法については、「the section called “ベクトル検索コレクションの使用”」を参照してください。コレクションの使用に関する詳細については、このガイドの「the section called “コレクションの管理”」とその他のトピックを参照してください。

このチュートリアルでは、次の手順を実行します。

1. アクセス許可を設定する
2. コレクションを作成する

3. データをアップロードおよび検索する
4. コレクションを削除する

> **Note**
>
> には ASCII 文字のみを使用してくださいIndexName。に ASCII 文字を使用しない場合IndexName、CloudWatch メトリクスIndexNameの は、非 ASCII 文字の URL エンコード形式に変換されます。

ステップ 1: アクセス許可を設定する

このチュートリアルを完了し、OpenSearch Serverless を一般的に使用するには、正しい IAM アクセス許可が必要です。このチュートリアルでは、コレクションを作成し、データをアップロードして検索を行い、最後にそのコレクションを削除します。

ユーザーまたはロールには、以下の最低限の許可を含む ID ベースのポリシーが、アタッチされている必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "aoss:CreateCollection",
        "aoss:ListCollections",
        "aoss:BatchGetCollection",
        "aoss:DeleteCollection",
        "aoss:CreateAccessPolicy",
        "aoss:ListAccessPolicies",
        "aoss:UpdateAccessPolicy",
        "aoss:CreateSecurityPolicy",
        "aoss:GetSecurityPolicy",
        "aoss:UpdateSecurityPolicy",
        "iam:ListUsers",
        "iam:ListRoles"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

OpenSearch Serverless での IAM アクセス許可の詳細については、「the section called “Identity and Access Management”」を参照してください。

ステップ 2: コレクションを作成する

コレクションとは、連携して特定のワークロードやユースケースをサポートする、OpenSearch インデックスのグループです。

OpenSearch Serverless コレクションを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、コレクションを選択します。
3. [Create collection] (コレクションの作成) を選択します。
4. [名前] に`movies`と入力してください。
5. コレクションタイプで、検索を選択します。詳細については、「Choosing a collection type」(コレクションタイプの選択) を参照してください。
6. コレクション作成メソッドで、Express Create を選択します。
7. デフォルト設定を確認します。これには、コレクショングループ、OpenSearch UI アプリケーション設定、 による暗号化 AWS 所有のキー、パブリックネットワークアクセス、自動生成されたデータアクセスポリシーが含まれます。
8. [Create collection] (コレクションの作成) を選択します。
9. コレクションステータスが に変わるまで待ちます`Active`。この処理には数分かかることがあります。

> **Tip**
>
> このチュートリアルでは、NextGen Express Create メソッドを使用します。ランディングページの情報バナーの Express 作成ボタンから Express Create にアクセスして、より迅速なパスを作成することもできます。暗号化、ネットワーク、データアクセス設定をより詳細に制御するには、標準作成を使用します。Classic コレクション作成ウィザードを使用するに

は、「Classic に切り替える」を選択します。詳細については、「コレクションの作成」を参照してください。

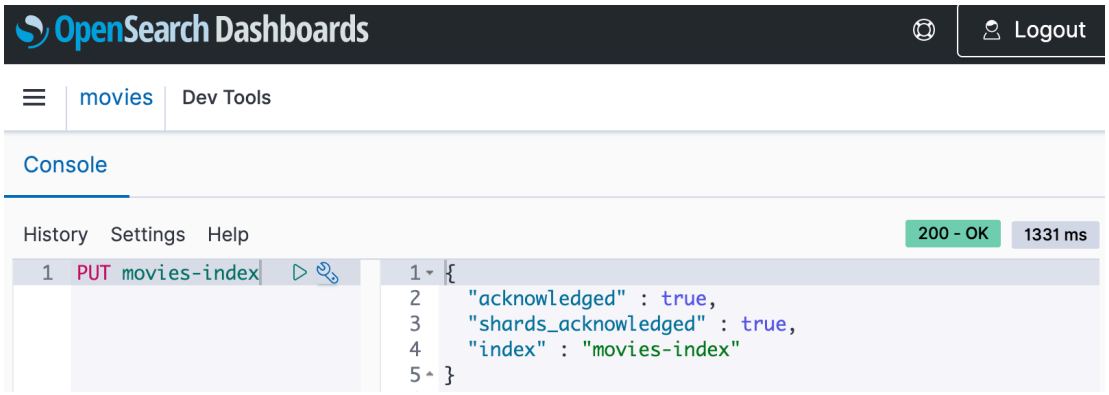
ステップ 3: データをアップロードして検索する

OpenSearch Serverless コレクションへのデータのアップロードには、Postman または cURL が使用できます。わかりやすくするために、これらの例では OpenSearch Dashboards コンソール内で開発ツールを使用します。

movies コレクションのデータをインデックス化して検索するには

1. 左側のナビゲーションペインで [Collections] (コレクション) を選択した後、「movies」コレクションを選択してその詳細ページを開きます。
2. コレクションの OpenSearch Dashboards URL を選択します。この URL の形式は、`https://dashboards.{region}.aoss.amazonaws.com/_login/?collectionId={collection-id}` になります。
3. OpenSearch Dashboards 内で左側のナビゲーションペインを開き、[Dev Tools] (開発ツール) を選択します。
4. movies-index というインデックスを 1 つ作成するには、次のリクエストを送信します。

```
PUT movies-index
```



5. 1 つのドキュメントを movies-index にインデックスするには、次のリクエストを送信します。

```
PUT movies-index/_doc/1
{
  "title": "Shawshank Redemption",
  "genre": "Drama",
```

```
  "year": 1994
}
```

6. OpenSearch Dashboards 内でデータを検索するためには、少なくとも 1 つのインデックスパターンを設定する必要があります。OpenSearch は、そのパターンを使用して、分析する対象のインデックスを特定します。左のナビゲーションパネルを開き、[Stack Management] (スタック管理)、[Index Patterns] (インデックスパターン) の順に選択してから、[Create index pattern] (インデックスパターンを作成) を選択します。このチュートリアルでは、movies と入力します。
7. [次のステップ] を選択してから、[インデックスパターンの作成] を選択します。パターンが作成されたら、`title` および `genre` などのさまざまなドキュメントフィールドを表示できます。
8. データの検索を開始するには、左側のナビゲーションペインを再度開き、検出を選択するか、開発ツール内で検索 API を使用します。

エラー処理

インデックスオペレーションと検索オペレーションを実行すると、次のエラーレスポンスが表示されることがあります。

- `HTTP 507` – 内部サーバーエラーが発生したことを示します。このエラーは通常、OpenSearch コンピューティングユニット (OCU) がリクエスト量またはリクエストの複雑さによって過負荷状態になっていることを示します。OpenSearch Serverless は負荷を管理するために自動的にスケーリングされますが、追加のリソースのデプロイに遅延が生じる可能性があります。

  このエラーを軽減するには、エクスポネンシャルバックオフ再試行ポリシーを実装します。このアプローチにより、リクエストレートを一時的に低減させ、負荷を効果的に管理できます。詳細については、「AWS SDK とツールのリファレンスガイド」の「Retry behavior」を参照してください。
- `HTTP 402` – OpenSearch コンピューティングユニット (OCU) の最大容量制限に達したことを示します。ワークロードを最適化して OCU 使用量を減らすか、クォータの引き上げをリクエストしてください。

ステップ 4: コレクションを削除する

movies コレクションはテスト用であるため、実験が終了したら削除してください。

OpenSearch Serverless コレクションを削除するには

1. Amazon OpenSearch Service コンソールに戻ります。
2. 左側のナビゲーションペインで [Collections] (コレクション) を選択した後、「movies」コレクションを選択します。
3. [削除] を選択して、削除を確認します。

次の手順

コレクションとインデックスデータを作成する方法がわかったので、以下を試すことができます。

- コレクションを作成するためのより高度なオプションについて説明します。詳細については、「the section called “コレクションの管理”」を参照してください。
- セキュリティポリシーを設定して、コレクションセキュリティを大規模に管理します。詳細については、「the section called “OpenSearch Serverless のセキュリティ”」を参照してください。
- 他の方法を使用して、データをコレクションにインデックス化します。詳細については、「the section called “コレクションへのデータの取り込み”」を参照してください。

Amazon OpenSearch Serverless コレクション

Amazon OpenSearch Serverless のコレクションとは、分析ワークロードを表す 1 つまたは複数のインデックスを論理的にグループ化したものです。OpenSearch Serverless は、コレクションを自動的に管理および調整するので、手動入力は最小限で済みます。

トピック

- Amazon OpenSearch Serverless コレクションの管理
- Amazon OpenSearch Serverless のコレクションエンドポイント
- ベクトル検索コレクションの使用
- Amazon OpenSearch Serverless でデータライフサイクルポリシーを使用する
- AWS SDKs を使用して Amazon OpenSearch Serverless を操作する
- CloudFormation を使用して Amazon OpenSearch Serverless コレクションを作成する
- スナップショットを使用したコレクションのバックアップ
- Amazon OpenSearch Serverless での Zstandard Codec サポート
- 派生ソースを使用してストレージを保存する

Amazon OpenSearch Serverless コレクションの管理

Amazon OpenSearch Serverless のコレクションとは、分析ワークロードを表す 1 つまたは複数のインデックスを論理的にグループ化したものです。OpenSearch Serverless は、コレクションを自動的に管理および調整するので、手動入力は最小限で済みます。

トピック

- コレクションのアクセス許可の設定
- Serverless の自動セマンティックエンリッチメント
- コレクションの作成
- OpenSearch Dashboards へのアクセス
- コレクションの表示
- コレクションの削除

コレクションのアクセス許可の設定

OpenSearch Serverless は、コレクションを作成および管理するために次の AWS Identity and Access Management (IAM) アクセス許可を使用します。IAM 条件を指定して、ユーザーを特定のコレクションに制限できます。

- `aoss:CreateCollection` – コレクションを作成します。
- `aoss:ListCollections` – 現在のアカウントのコレクションを一覧表示します。
- `aoss:BatchGetCollection` – 1 つまたは複数のコレクションに関する詳細情報を取得します。
- `aoss:UpdateCollection` – コレクションを変更します。
- `aoss:DeleteCollection` – コレクションを削除します。

次の ID ベースのアクセスポリシーのサンプルでは、ユーザーが Logs という名前の 1 つのコレクションを管理するのに必要な最小限のアクセス許可が付与されています。

```
[
   {
      "Sid":"Allows managing logs collections",
      "Effect":"Allow",
      "Action":[
         "aoss:CreateCollection",
         "aoss:ListCollections",
```

```
            "aoss:BatchGetCollection",
            "aoss:UpdateCollection",
            "aoss:DeleteCollection",
            "aoss:CreateAccessPolicy",
            "aoss:CreateSecurityPolicy"
        ],
        "Resource":"*",
        "Condition":{
            "StringEquals":{
                "aoss:collection":"Logs"
            }
        }
    }
]
```

aoss:CreateAccessPolicy コレクションが正しく機能するためには暗号化、ネットワーク、データアクセスポリシーが必要であるため、 と aoss:CreateSecurityPolicyが含まれています。詳細については、「the section called “Identity and Access Management”」を参照してください。

> **Note**
>
> アカウントで最初のコレクションを作成する場合は、 アクセスiam:CreateServiceLinkedRole許可も必要です。詳細については、「the section called “コレクション作成ロール”」を参照してください。

Serverless の自動セマンティックエンリッチメント

概要

自動セマンティックエンリッチメント機能は、辞書検索よりも検索の関連性を最大 20% 向上させるのに役立ちます。自動セマンティックエンリッチメントにより、独自の ML (機械学習) モデルインフラストラクチャと検索エンジンとの統合を管理するという差別化されていない手間が省けます。この機能は、検索、時系列、ベクトルの 3 つのサーバーレスコレクションタイプすべてで使用できます。

セマンティック検索の概念

従来の検索エンジンは、クエリの結果を見つけるためにword-to-word一致 (辞書検索と呼ばれます) に依存しています。これは、テレビのモデル番号などの特定のクエリでは有効ですが、より抽象的な

検索では関連する結果を返さない場合があります。例えば、「ビーチの靴」を検索する場合、辞書検索はカタログアイテム内の個々の単語「靴」、「ビーチ」、「対象」、「対象」に一致するだけで、正確な検索語を含まない「耐水性サンドラル」や「サーフ対象」などの関連製品が欠落している可能性があります。

セマンティック検索は、キーワードマッチングだけでなく、ユーザーの検索のインテントとコンテキストの意味を組み込んだクエリ結果を返します。例えば、ユーザーが「どのように頭が痛くなるか」を検索すると、セマンティック検索システムが次の結果を返します。

- 片頭痛の治療法
- 疼痛管理技術
- 市販の鎮痛剤

モデルの詳細とパフォーマンスベンチマーク

この機能は、基盤となるモデルを公開せずにバックグラウンドの技術的な複雑さを処理しますが、次のモデルの説明とベンチマークの結果は、重要なワークロードでの機能の採用について情報に基づいた意思決定を行うのに役立ちます。

自動セマンティックエンリッチメントは、カスタムのファインチューニングを必要とせずに効果的に機能するサービスマネージド型の事前トレーニング済みのスパースモデルを使用します。このモデルは、指定したフィールドを分析し、さまざまなトレーニングデータから学習した関連付けに基づいてスパースベクトルに展開します。拡張された用語とその重要度の重みは、効率的な取得のためにネイティブの Lucene インデックス形式で保存されます。このプロセスは、データの取り込み中にのみエンコードが行われるドキュメント専用モードを使用して最適化されています。検索クエリはスパースモデルを介して処理されるのではなくトークン化されるため、ソリューションはコスト効率とパフォーマンスの両方に優れています。

機能開発中のパフォーマンス検証では、平均 334 文字のパッセージを含む MS MARCO パッセージ取得データセットを使用しました。関連性スコアリングでは、使用されたメトリクスは、英語コンテンツの BEIR ベンチマークの最初の 10 件の検索結果 (ndcg@10) の平均正規化割引累積ゲイン (NDCG) と、多言語コンテンツの MIRACL の平均 ndcg@10 でした。レイテンシー評価では、クライアント側の 90 パーセンタイル (p90) 測定値を使用し、検索レスポンス p90 は値を取りました。これらのベンチマークは、検索の関連性と応答時間の両方のベースラインパフォーマンス指標を提供します。以下は、主要なベンチマーク番号です。

- 英語 - 辞書検索よりも 20% の関連性の向上。また、辞書検索よりも P90 検索レイテンシーが 7.7% 減少しました (BM25 は 26 ミリ秒、自動セマンティックエンリッチメントは 24 ミリ秒) 。

- 多言語 - レキシカル検索に比べて関連性が 105% 向上しましたが、P90 検索レイテンシーはレキシカル検索に比べて 38.4% 増加しました (BM25 は 26 ミリ秒、自動セマンティックエンリッチメントは 36 ミリ秒) 。

各ワークロードの固有の性質を考慮すると、実装を決定する前に独自のベンチマーク基準を使用して、開発環境でこの機能を評価できます。

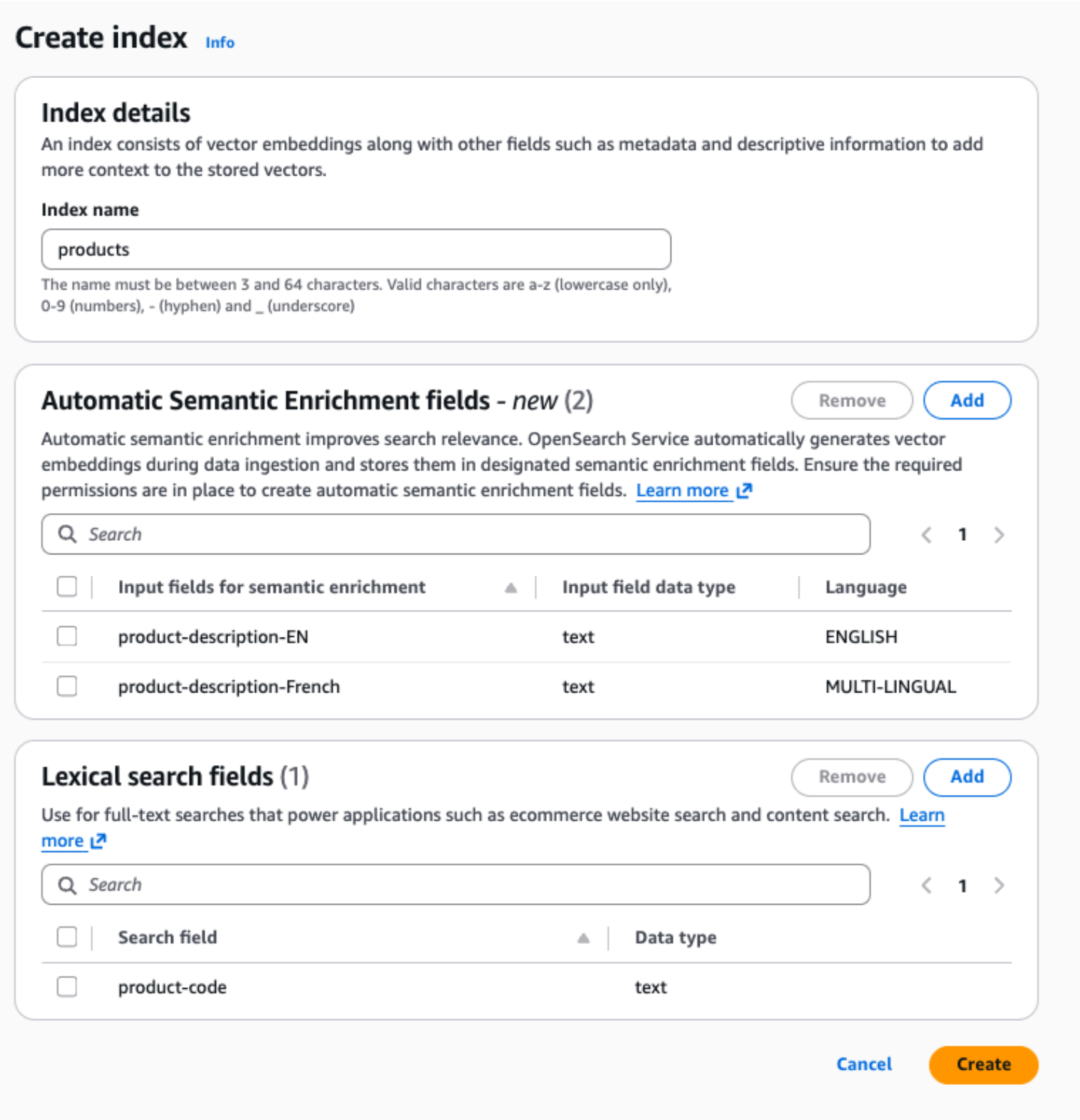
サポートされている言語

この機能は英語をサポートしています。さらに、このモデルはアラビア語、ベンガル語、中国語、フィンランド語、フランス語、ヒンディー語、インドネシア語、日本語、韓国語、ペルシャ語、ロシア語、スペイン語、スワヒリ語、テルグ語もサポートしています。

サーバーレスコレクションの自動セマンティックエンリッチメントインデックスを設定する

新しいインデックスの作成時に、コンソール、APIs、 CloudFormation テンプレートを使用して、テキストフィールドの自動セマンティックエンリッチメントを有効にしてインデックスを設定できます。既存のインデックスに対して有効にするには、テキストフィールドに対してセマンティック自動エンリッチメントを有効にしてインデックスを再作成する必要があります。

AWS コンソールでは、自動セマンティックエンリッチメントフィールドを使用してインデックスを作成できます。コレクションを選択すると、コンソールの上部にインデックスの作成ボタンが表示されます。インデックスの作成を選択すると、コンソールには自動セマンティックエンリッチメントフィールドを定義するオプションが表示されます。1 つのインデックスでは、英語と多言語の自動セマンティックエンリッチメントと、辞書フィールドを組み合わせることができます。



AWS コマンドラインインターフェイス (AWS CLI) を使用して自動セマンティックエンリッチメントインデックスを作成するには、create-index コマンドを使用します。

```
aws opensearchserverless create-index \
--id [collection_id] \
```

```
--index-name [index_name] \
--index-schema [index_body] \
```

次の index-schema の例では、 title_semanticフィールドのフィールドタイプを に設定textし、 パラメータをステータス semantic_enrichmentに設定しますENABLED。semantic_enrichment パラメータを設定すると、 title_semanticフィールドでセマンティックエンリッチメントが自動的に有効になります。language_options フィールドを使用して、 englishまたは を指定できますmulti-lingual。

```
    aws opensearchserverless create-index \
    --id XXXXXXXXX \
    --index-name 'product-catalog' \
    --index-schema '{
    "mappings": {
        "properties": {
            "product_id": {
                "type": "keyword"
            },
            "title_semantic": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "ENABLED",
                    "language_options": "english"
                }
            },
            "title_non_semantic": {
                "type": "text"
            }
        }
    }
}'
```

作成されたインデックスを記述するには、次のコマンドを使用します。

```
aws opensearchserverless get-index \
--id [collection_id] \
--index-name [index_name] \
```

CloudFormation テンプレート (Type:AWS::OpenSearchServerless::CollectionIndex) を使用して、コレクションのプロビジョニング中およびコレクションの作成後にセマンティック検索を作成することもできます。

既存のインデックスを更新する

既存のインデックスを更新して、新しいセマンティックエンリッチメントフィールドを追加したり、既存のフィールドでセマンティックエンリッチメントを有効または無効にしたり、非セマンティックテキストフィールドを追加したりできます。update-index コマンドを使用して、 で変更するフィールドのみを指定しますindex-schema。リクエストに含まれていないフィールドは変更されません。

> **Note**
>
> インデックスは更新settingsできません。リクエストにsettingsブロックを含めると、オペレーションは検証エラーを返します。インデックス設定を変更するには、インデックスを削除して再作成する必要があります。

を使用してインデックスを更新するには AWS CLI、 update-index コマンドを使用します。

```
aws opensearchserverless update-index \
--id [collection_id] \
--index-name [index_name] \
--index-schema [index_body]
```

新しいセマンティックエンリッチメントフィールドを追加する

セマンティックエンリッチメントを有効にした新しいtextフィールドを既存のインデックスに追加できます。このサービスは、必要な ML モデル、取り込みパイプライン、検索パイプラインを自動的にセットアップします。更新後にインデックス付けされた新しいドキュメントは自動的に強化されます。

> **Important**
>
> 既存のドキュメントはバックフィルされません。セマンティックエンリッチメントフィールドを既存のドキュメントに入力するには、更新後に再度取り込む必要があります。再取り込まれるまで、既存のドキュメントは新しいフィールドでのセマンティック検索の恩恵を受けることはありません。

```
aws opensearchserverless update-index \
--id my-collection-id \
--index-name product-catalog \
--index-schema '{
    "mappings": {
        "properties": {
            "description": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "ENABLED",
                    "language_options": "english"
                }
            }
        }
    }
}'
```

フィールドでセマンティックエンリッチメントを無効にする

現在有効になっているフィールドでセマンティックエンリッチメントを無効にするには、 status を に設定しますDISABLED。フィールドは、取り込みパイプラインと検索パイプラインから削除されます。基盤となるテキストフィールドとその埋め込みフィールドはインデックスに残りますが、強化されなくなります。

```
aws opensearchserverless update-index \
--id my-collection-id \
--index-name product-catalog \
--index-schema '{
    "mappings": {
        "properties": {
            "title_semantic": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "DISABLED"
                }
            }
        }
    }
}'
```

更新の制限

以下のオペレーションは ではサポートされていないupdate-indexため、インデックスを削除して再作成する必要があります。

- 現在**language_options**セマンティックエンリッチメントが有効になっているフィールドの変更。最初にフィールドを無効にしてから、新しい言語オプションで再度有効にします。
- ネストされたフィールドの更新。セマンティックエンリッチメントは最上位textフィールドでのみサポートされています。
- インデックス の更新**settings**。

> **Note**
>
> インデックスに、自動セマンティックエンリッチメントによって作成されなかったカスタム取り込みパイプラインまたは検索パイプラインがある場合、更新オペレーションはブロックされます。セマンティックエンリッチメントフィールドを追加する前に、カスタムパイプラインを削除します。

データの取り込みと検索

自動セマンティックエンリッチメントを有効にしてインデックスを作成すると、この機能はデータ取り込みプロセス中に自動的に機能するため、追加の設定は必要ありません。

データインジェスト: インデックスにドキュメントを追加すると、システムは自動的に次の操作を行います。

- セマンティックエンリッチメント用に指定したテキストフィールドを分析します
- OpenSearch Service マネージドスパースモデルを使用してセマンティックエンコーディングを生成します
- これらの強化された表現を元のデータとともに保存します。

このプロセスでは、OpenSearch の組み込み ML コネクタと取り込みパイプラインが使用され、バックグラウンドで自動的に作成および管理されます。

検索: セマンティックエンリッチメントデータは既にインデックス化されているため、クエリは ML モデルを再度呼び出すことなく効率的に実行されます。つまり、追加の検索レイテンシーのオーバーヘッドなしで、検索の関連性が向上します。

自動セマンティックエンリッチメントのアクセス許可の設定

自動セマンティックエンリッチメントインデックスを作成する前に、必要なアクセス許可を設定する必要があります。このセクションでは、必要なアクセス許可とその設定方法について説明します。

IAM ポリシーのアクセス許可

次の AWS Identity and Access Management (IAM) ポリシーを使用して、自動セマンティックエンリッチメントを操作するために必要なアクセス許可を付与します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AutomaticSemanticEnrichmentPermissions",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateIndex",
                "aoss:GetIndex",
                "aoss:UpdateIndex",
                "aoss:DeleteIndex",
                "aoss:APIAccessAll"
            ],
            "Resource": "*"
        }
    ]
}
```

キー許可

- `aoss:*Index` アクセス許可はインデックス管理を有効にします
- `aoss:APIAccessAll` アクセス許可は OpenSearch API オペレーションを許可します
- アクセス許可を特定のコレクションに制限するには、`"Resource": "*"` をそのコレクションの ARN に置き換えます

データアクセス許可を設定する

自動セマンティックエンリッチメントのインデックスを設定するには、インデックス、パイプライン、モデルコレクションリソースへのアクセス許可を付与する適切なデータアクセスポリシーが必要です。データアクセスポリシーの詳細については、「Amazon OpenSearch Serverless のデータアクセスコントロール」を参照してください。データアクセスポリシーを設定する手順については、「データアクセスポリシーの作成 (コンソール)」を参照してください。

データアクセス許可

```
[
    {
        "Description": "Create index permission",
        "Rules": [
            {
                "ResourceType": "index",
                "Resource": ["index/collection_name/*"],
                "Permission": [
                  "aoss:CreateIndex",
                  "aoss:DescribeIndex",
                  "aoss:UpdateIndex",
                  "aoss:DeleteIndex"
                ]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
        ]
    },
    {
        "Description": "Create pipeline permission",
        "Rules": [
            {
                "ResourceType": "collection",
                "Resource": ["collection/collection_name"],
                "Permission": [
                  "aoss:CreateCollectionItems",
                  "aoss:DescribeCollectionItems"
                ]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
```

```
        ]
    },
    {
        "Description": "Create model permission",
        "Rules": [
            {
                "ResourceType": "model",
                "Resource": ["model/collection_name/*"],
                "Permission": ["aoss:CreateMLResource"]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
        ]
    },
]
```

ネットワークアクセス許可

サービス API がプライベートコレクションにアクセスできるようにするには、サービス API とコレクション間の必要なアクセスを許可するネットワークポリシーを設定する必要があります。ネットワークポリシーの詳細については、「Amazon OpenSearch Serverless でのネットワークアクセス」を参照してください。

```
[
   {
      "Description":"Enable automatic semantic enrichment in a private collection",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/collection_name"
            ]
         }
      ],
      "AllowFromPublic":false,
      "SourceServices":[
         "aoss.amazonaws.com"
      ],
   }
]
```

プライベートコレクションのネットワークアクセス許可を設定するには

1. OpenSearch Service コンソール (https://console.aws.amazon.com/aos/home) にサインインします。
2. 左側のナビゲーションペインで、ネットワークポリシーを選択します。次に、以下のいずれかを行ってください。
   - 既存のポリシー名を選択し、[編集] を選択します
   - [ネットワークポリシーを作成] を選択し、ポリシーの詳細設定をおこないます。
3. [アクセスタイプ] エリアで、[プライベート (推奨)] を選択し、[AWS サービスプライベートアクセス] を選択します。
4. 検索フィールドで、[サービス] を選択し、aoss.amazonaws.com を選択します。
5. リソースタイプエリアで、OpenSearch エンドポイントへのアクセスを有効にするチェックボックスをオンにします。
6. [コレクションの検索] または [プレフィックス用語の入力] では、検索フィールドで [コレクション名] を選択します。次に、ネットワークポリシーに関連付けるコレクションの名前を入力または選択します。
7. 新しいネットワークポリシーの場合は [作成]、既存のネットワークポリシーの場合は [更新] を選択します。

サポートされている AWS リージョン

OpenSearch Serverless の自動セマンティックエンリッチメントは、以下で利用できます AWS リージョン。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)

- 欧州 (ストックホルム)
- 欧州 (スペイン)

クエリの書き換え

自動セマンティックエンリッチメントは、クエリの変更を必要とせずに、既存の「一致」クエリをセマンティック検索クエリに自動的に変換します。一致クエリが複合クエリの一部である場合、システムはクエリ構造を横断し、一致クエリを検索して、それらをニューラルスパースクエリに置き換えます。現在、この機能では、スタンドアロンクエリでも複合クエリの一部でも、「一致」クエリの置き換えのみがサポートされています。「multi_match」はサポートされていません。さらに、この機能は、ネストされた一致クエリを置き換えるすべての複合クエリをサポートします。複合クエリには、bool、Boosting、Constant_score、dis_max、 function_score、ハイブリッドなどがあります。

自動セマンティックエンリッチメントの制限

自動セマンティック検索は、映画のタイトル、製品説明、レビュー、概要など、自然言語コンテンツを含むsmall-to-medium規模のフィールドに適用すると最も効果的です。セマンティック検索はほとんどのユースケースの関連性を高めますが、特定のシナリオには適していない場合があります。特定のユースケースに自動セマンティックエンリッチメントを実装するかどうかを決定するときは、次の制限を考慮してください。

- 非常に長いドキュメント – 現在のスパースモデルは、各ドキュメントの最初の 8,192 トークンのみを英語で処理します。多言語ドキュメントの場合、トークンは 512 個です。長い記事の場合は、完全なコンテンツ処理を確保するためにドキュメントチャンキングの実装を検討してください。
- ログ分析ワークロード – セマンティックエンリッチメントはインデックスサイズを大幅に増やします。これは、完全一致が通常十分であるログ分析には不要である可能性があります。セマンティックコンテキストを追加することで、増加したストレージ要件を正当化するのに十分なログ検索の有効性が向上することはめったにありません。
- 自動セマンティックエンリッチメントは、派生ソース機能と互換性がありません。

料金

Amazon OpenSearch Service は、インデックス作成時にスパースベクトル生成中に消費された OpenSearch Compute Units (OCUs) に基づいて、自動セマンティックエンリッチメントを請求します。自動セマンティックエンリッチメントを有効にしたテキストフィールドのインデックス作成中の実際の使用に対してのみ課金されます。1 つのセマンティック検索 OCU は、英語コンテンツ

に対して 1,110 万個のトークンを処理できます。24 億個のトークンを処理するには、約 216 個のセマンティック検索 OCU 時間 (24 億/1,110 万) が必要です。セマンティック検索 OCU 時間あたり 0.24 USD の料金では、自動セマンティック検索用に 10 GB のデータを処理するコストは 51 USD (216 OCU 時間 x 0.24 USD/OCU 時間) になります。検索オペレーション中またはデータストレージには、セマンティック検索 OCU の追加料金はかかりません。

Amazon CloudWatch メトリクス を使用して、この消費量をモニタリングできますSemanticSearchOCU。モデルトークンの制限、OCU あたりのボリュームスループット、およびサンプル計算の例の詳細については、OpenSearch Service の料金」を参照してください。

コレクションの作成

コンソールまたは を使用して AWS CLI 、サーバーレスコレクションを作成できます。以下のステップは、検索コレクションまたは時系列コレクションの作成方法について説明します。ベクトル検索コレクションを作成するには、「the section called “ベクトル検索コレクションの使用”」を参照してください。

コレクションを作成する前に、必要なアクセス許可があることを確認してください。詳細については、「コレクションのアクセス許可の設定」を参照してください。

Amazon OpenSearch Serverless は、次の 2 つのコレクション世代をサポートしています。

- NextGen – 即時自動スケーリングとコスト最適化のためのscale-to-zeroを備えた最新世代の OpenSearch Serverless。コレクショングループを使用して、複数のコレクション間で共有容量を管理します。Express Create オプションと Standard Create オプションを使用して、シンプルな単一ページの作成フローを提供します。
- Classic – コレクションごとのセキュリティ、ネットワーク、暗号化設定でマルチステップウィザードを使用します。各コレクションは個別に設定されます。

コレクションページからコレクションを作成するを選択すると、コンソールはデフォルトで NextGen 作成フォームを開きます。作成フォームのサーバーレス生成フィールドの「Classic に切り替える」またはNextGen に切り替える」リンクを使用して、いつでも世代を切り替えることができます。

- NextGen コレクションを作成する (Express Create)
- NextGen コレクションを作成する (標準作成)
- Classic コレクションを作成する
- コレクション設定を構成する (Classic)

- 追加の検索フィールドを設定する (Classic)

NextGen コレクション設定を構成する

次の手順は、Express Create メソッドと Standard Create メソッドの両方に共通です。

NextGen コレクション設定を構成するには

1. https://console.aws.amazon.com/aos/home/ の Amazon OpenSearch Service コンソールに移動します。
2. 左側のナビゲーションペインで [Serverless] (サーバーレス) を展開し、[Collections] (コレクション) を選択します。
3. [Create collection] (コレクションの作成) を選択します。NextGen コレクション作成フォームがデフォルトで表示されます。

   > **Tip**
   >
   > 代わりに Classic コレクションを作成するには、Serverless generation フィールドで Switch to Classic を選択します。

4. コレクションの名前と説明を入力します。名前は次の基準を満たしている必要があります。
   - アカウントと AWS リージョン
   - 小文字の a ~ z、0 ~ 9 の数字、ハイフン (-) のみ含まれる
   - 3 ~ 32 文字
5. コレクションタイプを次の中から選択します。
   - [Search] (検索) – 社内ネットワークのアプリケーションやインターネットに接続するアプリケーションに使用される全文検索。すべての検索データはホットストレージに保存され、クエリの応答時間を短縮できます。
   - [ベクトル検索] – ベクトルデータ管理を簡素化するベクトル埋め込みのセマンティック検索。機械学習 (ML) 拡張検索エクスペリエンスと生成 AI アプリケーションを強化します。

   詳細については、「the section called “コレクションタイプを選択する”」を参照してください。
6. Express Create または Standard Create のコレクション作成方法を選択します。

NextGen コレクションを作成する (Express Create)

Express Create を使用すると、適切なデフォルトを使用して NextGen コレクションをすばやくセットアップできます。OpenSearch Serverless は、指定したコレクション名とコレクションタイプに基づいて、必要なコレクショングループ、暗号化、ネットワーク、データアクセスポリシーを自動的に作成します。

> **Note**
>
> Express Create には、以下からアクセスすることもできます。
>
> - Amazon OpenSearch Service ランディングページの青い情報バナーにある Express 作成ボタン
> - クイック作成カードの Express 作成ボタン
>
> どちらも、完全な作成ページに移動せずに Express Create モーダルを直接開きます。

モーダルから Express Create を使用すると、完全なコレクション作成フォームに移動せずに、コレクションを数秒で作成できます。コレクション名を入力し、コレクションタイプ (検索またはベクトル検索) を選択し、コレクションの作成を選択します。OpenSearch Serverless は、適切なデフォルトを使用して残りの設定を自動的に設定します。

設定の詳細セクションを展開して、コレクションを作成する前に、コレクショングループ、OpenSearch UI 設定、暗号化、ネットワークアクセス、データアクセスなどの設定を確認または変更できます。一部の設定は、作成後に変更することもできます。

Express Create を使用して NextGen コレクションを作成するには

1. コレクション名、説明、タイプを設定したら (「」を参照the section called “NextGen コレクション設定を構成する”)、コレクション作成方法として Express Create を選択します。
2. OpenSearch Serverless がユーザーに代わって設定するデフォルト設定テーブルを確認します。
   - コレクショングループ – 初めて使用するユーザーの場合、デフォルトのコレクショングループ名はコレクション名に基づいて自動的に生成されます。編集アイコンを選択すると、名前を編集できます。戻りユーザーの場合、選択したタイプの既存の NextGen コレクショングループが表示されます。デフォルトでは 1 つが選択されます。ドロップダウンから他の既存のコレクショングループを選択できます。

- OpenSearch UI 設定 – デフォルトでは、既存の OpenSearch アプリケーションを使用します。アプリケーション名とワークスペース名は、作成後に編集できます。
- 暗号化 – コレクションは で暗号化されます AWS 所有のキー。
- ネットワークアクセス – コレクションはパブリックネットワークからアクセスできます。
- データアクセス – 現在の IAM プリンシパルにコレクションとそのインデックスへのフルアクセスを許可するデータアクセスポリシーが自動的に作成されます。

3. [Create collection] (コレクションの作成) を選択します。

コレクションのステータスは と表示されますCreating。データのインデックス作成を開始する前に、コレクションステータスが に変わるのActiveを待ちます。これには通常数分かかります。

NextGen コレクションを作成する (標準作成)

Standard Create では、コレクショングループ、暗号化、ネットワーク、データアクセス設定を完全に制御できます。

Standard Create を使用して NextGen コレクションを作成するには

1. 「」のステップ 1～5 に従いますNextGen コレクション設定を構成する。
2. コレクション作成メソッドで、標準作成を選択します。

   フォームが展開され、以下のセクションで説明する完全な設定セクションが表示されます。

コレクショングループを設定する

コレクショングループを割り当てる方法を選択します。互換性のあるグループがすでに存在する場合でも、既存の互換性のあるグループを選択するか、新しいグループを作成できます。

- Select existing – ドロップダウンから互換性のあるコレクショングループを選択します。グループは、選択したコレクションタイプをサポートしている必要があります。コンソールには、グループの現在の容量制限 (インデックス作成と検索の最小/最大 OCU) が表示されます。
- 新規作成 – カスタム容量制限を使用して新しいコレクショングループを作成します。
  - コレクショングループ名 – として自動生成されますnextgen-{collection-name}。この名前は編集できます。名前は小文字に正規化され、ハイフンはアンダースコアを置き換え、32 文字に切り捨てられます。
  - 最小インデックス作成容量 (OCUs 単位) – オプション。最小数がない場合は空白のままにします。

- 最大インデックス作成容量 (OCUs) — デフォルトは 96 です。
- 最小検索容量 (OCUs 単位) – オプション。最小数がない場合は空白のままにします。
- 最大検索容量 (OCUs) — デフォルトは 96 です。

暗号化を設定

暗号化オプションを選択します。

- AWS 所有キーを使用する (デフォルト) – Amazon OpenSearch Service は、暗号化キーを追加コストなしで管理します。
- カスタマーマネージドキーを使用する – AWS KMS アカウントからキーを選択します。

(オプション) 暗号化設定をカスタマイズチェックボックスを選択して、別の AWS KMS キーを選択または作成します。

暗号化ポリシーの詳細については、「」を参照してくださいthe section called “暗号化”。

ネットワークアクセスの設定

コレクションのネットワークアクセスを設定します。

- OpenSearch エンドポイントへのパブリックアクセスを有効または無効にします。
- (オプション) VPC エンドポイントルールを追加してアクセスを制限します。

データアクセスポリシーを設定する

データアクセスを設定する方法を選択します。

- 新しいポリシーの作成 – 新しいデータアクセスポリシーを定義します。
  - ポリシー名とオプションの説明を入力します。
  - ビジュアルエディタを使用して、プリンシパルとリソースのアクセス許可を持つステートメントを追加するか、JSON エディタに切り替えてポリシーを直接編集します。
  - プリンシパルを追加する: IAM ユーザーとロール、SAML ユーザーとグループ、またはユーザーとグループ。
  - コレクションとインデックスのリソースアクセス許可を指定します。
- 既存のポリシーに追加する – ドロップダウンから既存のデータアクセスポリシーを選択します。

追加設定を構成する

- (オプション) コレクションにキーと値のペアとしてタグを追加します。

OpenSearch UI 設定を構成する

コレクションの OpenSearch アプリケーションとワークスペースを設定します。

- OpenSearch アプリケーションを選択するには、次のいずれかを選択します。
  - 既存の OpenSearch アプリケーションを選択する – OpenSearch アプリケーション名ドロップダウンから既存のアプリケーションを選択します。次に、Workspace を選択するには、既存のワークスペースを選択するか、新しいワークスペースを作成して名前を入力します。
  - 新しい OpenSearch アプリケーションを作成する – 新しいアプリケーションの名前を入力します。また、新しいワークスペースを作成する必要があります。ワークスペースの名前を入力します。

コレクションを作成するには、設定を確認し、送信を選択します。

送信すると、コレクションは Creating ステータスになります。コンソールは、必要に応じてコレクショングループ (新しい場合) 、暗号化ポリシー、ネットワークポリシー、データアクセスポリシーも作成します。コレクションステータスがアクティブに変わると、使用できる状態になります。これには通常数分かかります。

Classic コレクションを作成する

クラシックコレクションでは、コレクションごとのセキュリティ、ネットワーク、および暗号化設定を持つマルチステップウィザードを使用します。各コレクションは個別に設定されます。

コレクション設定を構成する (Classic)

基本的なコレクション設定を構成するには (Classic)

1. https://console.aws.amazon.com/aos/home/ の Amazon OpenSearch Service コンソールに移動します。左側のナビゲーションペインで、サーバーレスを展開し、コレクションを選択します。
2. [Create collection] (コレクションの作成) を選択します。

   コンソールは、デフォルトで NextGen コレクションの作成フォームを開きます。
3. Serverless generation フィールドで、Classic に切り替えを選択します。

コンソールは Classic コレクション作成ウィザードに移動します。Serverless generation フィールドに Classic と説明が表示されるようになりました。これは Classic コレクションの作成フローです。

> **Tip**
>
> NextGen に戻すには、サーバーレス生成フィールドで NextGen に切り替えるを選択します。

4. コレクションの名前と説明を入力します。名前は次の基準を満たしている必要があります。
   - アカウントと AWS リージョン
   - 小文字 a ~ z、数字 0 ~ 9、ハイフン (-) のみが含まれます。
   - 3 ~ 32 文字
5. コレクションタイプを選択します。
   - [Time series] (時系列) – マシン生成の大量の半構造化データの分析に焦点を当てたログ分析セグメント。少なくとも 24 時間のデータはホットインデックスに保存され、残りはウォームストレージに残ります。
   - [Search] (検索) – 社内ネットワークのアプリケーションやインターネットに接続するアプリケーションに使用される全文検索。すべての検索データはホットストレージに保存され、クエリの応答時間を短縮できます。自動セマンティック検索を有効にする場合は、このオプションを選択します。
   - ベクトル検索 – ベクトルデータの管理を簡素化するベクトル埋め込みのセマンティック検索。機械学習 (ML) 拡張検索エクスペリエンスと、チャットボット、パーソナルアシスタント、不正行為検出などの生成 AI アプリケーションを強化します。
6. (オプション) デプロイオプションを設定します。
   - スタンバイレプリカ – 高可用性のためにスタンバイレプリカを有効にするかどうかを選択します。
   - GPU アクセラレーション – ベクトル検索コレクションでは、GPU アクセラレーションを有効にするかどうかを選択します。
7. [次へ] を選択します。

セキュリティポリシーとアクセスポリシーを設定するには (Classic)

1. 暗号化ポリシー – コレクション内のデータを暗号化する方法を選択します。

   - AWS 所有キーを使用する (デフォルト) – Amazon OpenSearch Service は、暗号化キーを追加コストなしで管理します。
   - カスタマーマネージドキーを使用する – アカウントから暗号化用の AWS KMS キーを選択します。

   (オプション) コレクション名に一致する暗号化ポリシーが既に存在する場合は、既存のポリシーを使用するを選択します。
2. ネットワークアクセス – ユーザーとアプリケーションがコレクションにアクセスする方法を設定します。

   - OpenSearch エンドポイントへのアクセスを有効または無効にします。
   - OpenSearch Dashboards へのアクセスを有効または無効にします。
   - パブリックアクセスを選択するか、VPC エンドポイントを介してアクセスを制限します。
3. データアクセスポリシー – コレクションのデータにアクセスできるプリンシパルを定義します。

   - 新しいポリシーの作成 – ビジュアルエディタまたは JSON エディタを使用して、新しいデータアクセスポリシーを定義します。プリンシパル (IAM ユーザー、IAM ロール、または SAML ユーザーとグループ) を追加し、リソースのアクセス許可を指定します。
   - 既存のポリシーに追加する – ドロップダウンから既存のデータアクセスポリシーを選択します。
4. [次へ] を選択します。

OpenSearch UI を設定する (クラシック)

コレクションの OpenSearch アプリケーションとワークスペースを設定します。

- OpenSearch アプリケーションを選択するには、次のいずれかを選択します。
  - 既存の OpenSearch アプリケーションを選択する – OpenSearch アプリケーション名ドロップダウンから既存のアプリケーションを選択します。次に、Workspace を選択するには、既存のワークスペースを選択するか、新しいワークスペースを作成して名前を入力します。

- 新しい OpenSearch アプリケーションを作成する – 新しいアプリケーションの名前を入力します。また、新しいワークスペースを作成する必要があります。ワークスペースの名前を入力します。

[次へ] を選択します。

追加の検索フィールドを設定する (Classic)

このページに表示されるオプションは、作成するコレクションのタイプによって異なります。

- 検索コレクション – 自動セマンティックエンリッチメントフィールドと辞書検索フィールドを設定します。
- 時系列コレクション – 時系列検索フィールドを設定します。
- ベクトル検索コレクション – フィールド名、ディメンション、距離メトリクスなど、ベクトルフィールドを設定します。

(オプション) コレクションタイプに関連する検索フィールドを設定します。[次へ] を選択します。概要ページのすべての設定を確認し、送信を選択してコレクションを作成します。

送信すると、コレクションは Creating ステータスになります。ステータスがアクティブに変わると、コレクションを使用する準備が整います。

トピック

- 自動セマンティックエンリッチメントを設定する
- 時系列検索フィールドを設定する
- レキシカル検索フィールドを設定する
- ベクトル検索フィールドを設定する

自動セマンティックエンリッチメントを設定する

コレクションを作成または編集するときに、自動セマンティックエンリッチメントを設定できます。これにより、Amazon OpenSearch Service でのセマンティック検索の実装と機能が簡素化されます。セマンティック検索は、キーワードマッチングだけでなく、ユーザーの検索のインテントとコンテキストの意味を組み込んだクエリ結果を返します。詳細については、「Serverless の自動セマンティックエンリッチメント」を参照してください。

自動セマンティックエンリッチメントを設定するには

1. [インデックス詳細] セクションの [インデックス名] で、名前を指定します。
2. [自動セマンティックエンリッチメントフィールド] セクションで、[セマンティック検索フィールドを追加] を選択します。
3. [セマンティックエンリッチメントのフィールド名を入力] フィールドに、エンリッチメントするフィールドの名前を入力します。
4. [データ型] は [Text] です。これを変更することはできません。
5. [言語] では、[英語] または [多言語] を選択します。
6. [Add field (フィールドを追加)] を選択します。
7. コレクションのオプションフィールドの設定が完了したら、[次へ] を選択します。
8. 変更を確認し、[送信] を選択してコレクションを作成します。

時系列検索フィールドを設定する

[時系列検索フィールド] セクションのオプションは、時系列データとデータストリームに関連しています。これらのサブジェクトの詳細については「Data Streams を用いた Amazon OpenSearch Service での時系列データの管理」を参照してください。

時系列検索フィールドを設定するには

1. [時系列検索フィールド] セクションで、[時系列フィールドを追加] を選択します。
2. [フィールド名] に名前を入力します。
3. [データ型] で、リストからデータ型を選択します。
4. [Add field (フィールドを追加)] を選択します。
5. コレクションのオプションフィールドの設定が完了したら、[次へ] を選択します。
6. 変更を確認し、[送信] を選択してコレクションを作成します。

レキシカル検索フィールドを設定する

レキシカル検索は、検索クエリとインデックス付き用語またはキーワードが完全に一致するものを返します。

レキシカル検索フィールドを設定するには

1. [レキシカル検索フィールド] セクションで、[検索フィールドを追加] を選択します。

2. [フィールド名] に名前を入力します。
3. [データ型] で、リストからデータ型を選択します。
4. [Add field (フィールドを追加)] を選択します。
5. コレクションのオプションフィールドの設定が完了したら、[次へ] を選択します。
6. 変更を確認し、[送信] を選択してコレクションを作成します。

ベクトル検索フィールドを設定する

> **Note**
>
> `Engine` プロパティは Classic コレクションでのみ設定でき、NextGen コレクションではサポートされていません。

ベクトル検索フィールドを設定するには

1. [ベクトルフィールド] セクションで[ベクトルフィールドの追加] を選択します。
2. [フィールド名] に名前を入力します。
3. [エンジン] で、リストからタイプを選択します。
4. ディメンションの数を入力します。
5. [距離メトリクス] で、リストからタイプを選択します。
6. コレクションのオプションフィールドの設定が完了したら、[次へ] を選択します。
7. 変更を確認し、[送信] を選択してコレクションを作成します。

コレクションを作成する (CLI)

このセクションの手順では、 AWS CLIを使用して OpenSearch Serverless コレクションを作成します。

トピック

- [開始する前に]
- コレクションの作成
- 自動セマンティックエンリッチメントインデックスを使用したコレクションの作成

[開始する前に]

を使用してコレクションを作成する前に AWS CLI、次の手順を使用してコレクションに必要なポリシーを作成します。

> **Note**
>
> 以下の各手順でコレクションの名前を指定する場合、その名前は次の基準を満たしている必要があります。
>
> - アカウントと AWS リージョン
> - 小文字の a～z、0～9 の数字、ハイフン (-) のみ含まれる
> - 3～32 文字

コレクションに必要なポリシーを作成するには

1. を開き AWS CLI 、次のコマンドを実行して、コレクションの意図した名前と一致するリソースパターンを持つ暗号化ポリシーを作成します。

   ```
   &aws opensearchserverless create-security-policy \
     --name policy name \
     --type encryption --policy "{\"Rules\":[{\"ResourceType\":\"collection\",
   \"Resource\":[\"collection\/collection name\"]}],\"AWSOwnedKey\":true}"
   ```

   例えば、コレクションに「logs-application」という名前を付ける場合は、次のような暗号化ポリシーを作成します。

   ```
   &aws opensearchserverless create-security-policy \
     --name logs-policy \
     --type encryption --policy "{\"Rules\":[{\"ResourceType\":\"collection\",
   \"Resource\":[\"collection\/logs-application\"]}],\"AWSOwnedKey\":true}"
   ```

   このポリシーを別のコレクションにも使用する場合は、`collection/logs*` や `collection/*` のようにルールを拡張しておくことができます。

2. 次のコマンドを実行して、ネットワークポリシーを使用してコレクションのネットワーク設定を行います。コレクションを作成した後でネットワークポリシーを作成できますが、ベストプラクティスとして、事前に実行してください。

```
&aws opensearchserverless create-security-policy \
  --name policy name \
  --type network --policy "[{\"Description\":\"description\",\"Rules\":
[{\"ResourceType\":\"dashboard\",\"Resource\":[\"collection\/collection name\"]},
{\"ResourceType\":\"collection\",\"Resource\":[\"collection\/collection name\"]}],
\"AllowFromPublic\":true}]"
```

例えば前述の「logs-application」であれば、以下のネットワークポリシーを作成できます。

```
&aws opensearchserverless create-security-policy \
  --name logs-policy \
  --type network --policy "[{\"Description\":\"Public access for logs collection
\",\"Rules\":[{\"ResourceType\":\"dashboard\",\"Resource\":[\"collection\/logs-
application\"]},{\"ResourceType\":\"collection\",\"Resource\":[\"collection\/logs-
application\"]}],\"AllowFromPublic\":true}]"
```

コレクションの作成

次の手順では、CreateCollection API アクションを使用して、タイプ SEARCHまたは TIMESERIES のコレクションを作成します。リクエストでコレクションタイプを指定しない場合、デフォルトは TIMESERIES です。これらのタイプの詳細については、「コレクションタイプを選択する」を参照してください。ベクトル検索コレクションを作成するには、「the section called “ベクトル検索コレクションの使用”」を参照してください。

コレクションが で暗号化されている場合 AWS 所有のキー、 kmsKeyArnは ARN autoではなく です。

> **Important**
>
> コレクションを作成した後は、データアクセスポリシーと一致しない限り、そのコレクションにアクセスすることはできません。詳細については、「Amazon OpenSearch Serverless のデータアクセスコントロール」を参照してください。

コレクションを作成するには

1. 「[開始する前に]」で説明されている必須ポリシーを作成できたことを確認します。

2. 以下のコマンドを実行してください。type の場合、SEARCH または TIMESERIES のいずれかを指定します。

```
&aws opensearchserverless create-collection --name "collection name" --
type collection type --description "description"
```

コレクションの作成には通常数分かかります。BatchGetCollection オペレーションを使用して、コレクションのステータスを確認します。

自動セマンティックエンリッチメントインデックスを使用したコレクションの作成

以下の手順を使用して、自動セマンティックエンリッチメント用に設定されたインデックスを持つ新しい OpenSearch Serverless コレクションを作成します。この手順では、OpenSearch Serverless CreateIndex API アクションを使用します。

自動セマンティックエンリッチメント用に設定されたインデックスを使用して新しいコレクションを作成するには

次のコマンドを実行して、コレクションとインデックスを作成します。

```
&aws opensearchserverless create-index \
--region Region ID \
--id collection name --index-name index name \
--index-schema \
'mapping in json'
```

以下に例を示します。

```
&aws opensearchserverless create-index \
--region us-east-1 \
--id conversation_history --index-name conversation_history_index \
--index-schema \
'{
    "mappings": {
        "properties": {
            "age": {
                "type": "integer"
            },
            "name": {
                "type": "keyword"
```

```
            },
            "user_description": {
                "type": "text"
            },
            "conversation_history": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "ENABLED",
                    // Specifies the sparse tokenizer for processing multi-lingual text
                    "language_option": "MULTI-LINGUAL",
                    // If embedding_field is provided, the semantic embedding field
 will be set to the given name rather than original field name + "_embedding"
                    "embedding_field": "conversation_history_user_defined"
                }
            },
            "book_title": {
                "type": "text",
                "semantic_enrichment": {
                    // No embedding_field is provided, so the semantic embedding field
 is set to "book_title_embedding"
                    "status": "ENABLED",
                    "language_option": "ENGLISH"
                }
            },
            "abstract": {
                "type": "text",
                "semantic_enrichment": {
                    // If no language_option is provided, it will be set to English.
                    // No embedding_field is provided, so the semantic embedding field
 is set to "abstract_embedding"
                    "status": "ENABLED"
                }
            }
        }
    }
}'
```

OpenSearch Dashboards へのアクセス

を使用してコレクションを作成したら AWS マネジメントコンソール、コレクションの OpenSearch Dashboards URL に移動できます。Dashboards URL を確認するには、ナビゲーションペインで [Collections] (コレクション) を選択し、コレクションを選択して詳細ページを開きます。この URL の形式は、`https://dashboards.us-east-1.aoss.amazonaws.com/_login/?`

collectionId=*07tjusf2h91cunochc* になります。URL に移動したら、ダッシュボードに自動的にログインします。

OpenSearch Dashboards URL が既にあるが、 にない場合は AWS マネジメントコンソール、ブラウザから Dashboards URL を呼び出すとコンソールにリダイレクトされます。 AWS 認証情報を入力すると、自動的に Dashboards にログインします。SAML のコレクションにアクセスする方法については、「Accessing OpenSearch Dashboards with SAML」(SAML による OpenSearch Dashboards へのアクセス) を参照してください。

OpenSearch Dashboards コンソールのタイムアウトは 1 時間で、設定できません。

> **Note**
>
> 2023 年 5 月 10 日に、OpenSearch は、OpenSearch Dashboards の共通のグローバルエンドポイントを導入しました。今後は、https://dashboards.*us-east-1*.aoss.amazonaws.com/_login/?collectionId=*07tjusf2h91cunochc* の形式をとる URL を使って、ブラウザで OpenSearch Dashboards に移動できます。下位互換性を確保するために、OpenSearch は 形式の既存のコレクション固有の OpenSearch Dashboards エンドポイントを引き続きサポートしますhttps://*07tjusf2h91cunochc*.*us-east-1*.aoss.amazonaws.com/_dashboards。

コレクションの表示

Amazon OpenSearch Service コンソールのコレクションタブ AWS アカウント で、 の既存のコレクションを表示できます。

コレクションとその ID を一覧表示するには、ListCollections リクエストを送信します。

```
&aws opensearchserverless list-collections
```

レスポンス例

```
{
   "collectionSummaries":[
      {
         "arn":"arn:aws:aoss:us-east-1:123456789012:collection/07tjusf2h91cunochc",
```

```
         "id":"07tjusf2h91cunochc",
         "name":"my-collection",
         "status":"CREATING"
      }
   ]
}
```

検索結果を制限するには、コレクションフィルターを使用します。このリクエストは、ACTIVE 状態のコレクションへのレスポンスを次のようにフィルタリングします。

```
&aws opensearchserverless list-collections --collection-filters '{ "status":
 "ACTIVE" }'
```

OpenSearch エンドポイントと OpenSearch Dashboards エンドポイントを含む 1 つまたは複数のコレクションに関する詳細情報を取得するには、BatchGetCollection リクエストを送信します。

```
&aws opensearchserverless batch-get-collection --ids "07tjusf2h91cunochc"
 "1iu5usc4rame"
```

> **Note**
>
> リクエストには --names または --ids を含めることができますが、両方を含めることはできません。

レスポンス例

```
{
   "collectionDetails":[
      {
         "id": "07tjusf2h91cunochc",
         "name": "my-collection",
         "status": "ACTIVE",
         "type": "SEARCH",
         "description": "",
         "arn": "arn:aws:aoss:us-east-1:123456789012:collection/07tjusf2h91cunochc",
         "kmsKeyArn": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "createdDate": 1667446262828,
         "lastModifiedDate": 1667446300769,
```

```
         "collectionEndpoint": "https://07tjusf2h91cunochc.us-
east-1.aoss.amazonaws.com",
         "dashboardEndpoint": "https://07tjusf2h91cunochc.us-east-1.aoss.amazonaws.com/
_dashboards"
      },
      {
         "id": "178ukvtg3i82dvopdid",
         "name": "another-collection",
         "status": "ACTIVE",
         "type": "TIMESERIES",
         "description": "",
         "arn": "arn:aws:aoss:us-east-1:123456789012:collection/178ukvtg3i82dvopdid",
         "kmsKeyArn": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "createdDate": 1667446262828,
         "lastModifiedDate": 1667446300769,
         "collectionEndpoint": "https://178ukvtg3i82dvopdid.us-
east-1.aoss.amazonaws.com",
         "dashboardEndpoint": "https://178ukvtg3i82dvopdid.us-
east-1.aoss.amazonaws.com/_dashboards"
      }
   ],
   "collectionErrorDetails":[]
}
```

コレクションの削除

コレクションを削除すると、コレクション内のすべてのデータとインデックスが削除されます。削除したコレクションを復元することはできません。

コンソールを使用してコレクションを削除するには

1. Amazon OpenSearch Service コンソールの [Collections] (コレクション) パネルから、削除するコレクションを選択します。
2. [削除] を選択して、削除を確認します。

を使用してコレクションを削除するには AWS CLI、DeleteCollection リクエストを送信します。

```
&aws opensearchserverless delete-collection --id 07tjusf2h91cunochc
```

レスポンス例

```
{
   "deleteCollectionDetail":{
      "id":"07tjusf2h91cunochc",
      "name":"my-collection",
      "status":"DELETING"
   }
}
```

Amazon OpenSearch Serverless のコレクションエンドポイント

Amazon OpenSearch Serverless NextGen は、on.awsドメインにリソースベースのエンドポイントを提供します。コレクションごとのエンドポイントを使用して単一のコレクションにアクセスするか、アカウントごとのエンドポイントを使用して AWS アカウント 、1 つのホスト名を介してリージョン内の のすべてのコレクションにアクセスできます。どちらのエンドポイントも標準で動作します AWS PrivateLink 。Amazon VPC コンソールまたは Amazon `CreateVpcEndpoint` Amazon EC2 API を使用して VPC エンドポイントを作成します。

OpenSearch Serverless Classic は、 でコレクションごとのエンドポイントを使用しますaoss.amazonaws.com。

トピック

- エンドポイントタイプ
- コレクションごとのエンドポイント
- アカウントごとのエンドポイント

エンドポイントタイプ

OpenSearch Serverless は、次のコレクションエンドポイント形式をサポートしています。

OpenSearch Serverless コレクションのエンドポイント形式

エンドポイントタイプ	形式	説明
コレクションごと (NextGen)	*collection-id*.aoss.*region*.on.aws	ホスト名からコレクションを識別します。コレクションごとに 1 つのエンドポイント。

エンドポイントタイプ	形式	説明
アカウントごと (NextGen)	*account-id* .aoss.*region*.on.aws	リクエストヘッダーからコレクションを識別します。1 つのエンドポイントは、リージョン内の のすべてのコレクション AWS アカウント を提供します。
コレクションごと (クラシック)	*collection-id*.*region*.aoss.amazonaws.com	ホスト名からコレクションを識別します。Classic コレクションごとに 1 つのエンドポイント。

どちらの NextGen エンドポイント形式も、VPC アクセス AWS PrivateLink に標準を使用します。VPC エンドポイントの作成の詳細については、「the section called “データプレーン VPC エンドポイント”」を参照してください。

FIPS AWS リージョン をサポートする OpenSearch Serverless では、これらのエンドポイントの FIPS 準拠のバリアントも提供しています。詳細については、「the section called “OpenSearch Serverless での FIPS エンドポイントの使用”」を参照してください。

コレクションごとのエンドポイント

コレクションごとのエンドポイントは、コレクション ID を含むホスト名を介して単一のコレクションをターゲットにします。エンドポイントは OpenSearch Serverless コンソールのコレクションの詳細ページに表示され、 `collectionEndpoint`フィールドの `BatchGetCollection` API によって返されます。

形式

- NextGen: *collection-id*.aoss.*region*.on.aws
- クラシック: *collection-id*.*region*.aoss.amazonaws.com

次のリクエストは、ドキュメントを のコレクションの`movies`インデックスにインデックス`1tg2xudton46knx2a95g`化します`us-east-1`。

```
PUT https://1tg2xudton46knx2a95g.aoss.us-east-1.on.aws/movies/_doc/1
{
```

```
  "title": "Shawshank Redemption",
  "year": 1994
}
```

アカウントごとのエンドポイント

アカウントごとのエンドポイントは、リージョン内の のすべてのコレクション AWS アカウント を単一のホスト名でターゲットにします。すべてのコレクションが同じホスト名を共有するため、これらの HTTP ヘッダーのいずれかを使用してターゲットコレクションを識別します。少なくとも 1 つが必要です。両方を含める場合は、同じコレクションを参照する必要があります。

- `x-amz-aoss-collection-name` – お客様が割り当てたコレクション名
- `x-amz-aoss-collection-id` – サービスによって生成されたコレクション ID

アカウントごとのエンドポイントを使用すると、 ヘッダーを介してリクエストごとにターゲットコレクションを切り替えることで、1 つのクライアントが複数のコレクション間で 1 つの接続プールを共有できます。これは、各テナントを個別のコレクションとしてモデル化するマルチテナントアプリケーションに役立ちます。

を使用すると、アプリケーションは既知の名前でリクエストをルーティング`x-amz-aoss-collection-name`できます。サービスで生成されたコレクション ID の保存または検索をスキップします。

アカウントごとのエンドポイントは、OpenSearch Serverless コンソールには表示されません。AWS アカウント ID とリージョンから作成します。

形式

- NextGen: *`account-id`*`.aoss.`*`region`*`.on.aws`

次のリクエストは、 `my-collection`のアカウントのコレクションの`movies`インデックスにドキュメントをインデックス`123456789012`化します`us-east-1`。

```
PUT https://123456789012.aoss.us-east-1.on.aws/movies/_doc/1
x-amz-aoss-collection-name: my-collection

{
  "title": "Shawshank Redemption",
  "year": 1994
```

```
}
```

> **Note**
>
> ヘッダーx-amz-aoss-collection-nameと x-amz-aoss-collection-idヘッダーは SigV4-signed付きである必要があります。 AWS SDKsこれを自動的に実行します。

ベクトル検索コレクションの使用

OpenSearch Serverless のベクトル検索コレクションタイプを使用すると、スケーラブルで高性能な類似度検索を実行できます。基盤となるベクトルデータベースインフラストラクチャを管理することなく、最新の機械学習 (ML) 拡張検索エクスペリエンスと生成人工知能 (AI) アプリケーションを構築できます。

ベクトル検索コレクションのユースケースには、画像検索、ドキュメント検索、音楽検索、製品のレコメンデーション、動画検索、位置ベースの検索、不正検出、異常検出などがあります。

OpenSearch Serverless のベクトルエンジンは、OpenSearch の k 最近傍 (k-NN) 検索機能を使用します。サーバーレス環境のシンプルさで同じ機能を利用できます。エンジンは k-NN プラグイン API をサポートしています。これらのオペレーションでは、全文検索、高度なフィルタリング、集計、地理空間クエリ、ネストされたクエリを使用して、データの取得を高速化し、検索結果を強化できます。

ベクトルエンジンは、ユークリッド距離、コサイン類似度、ドット積などの距離メトリクスを提供し、16,000 次元に対応できます。数値、ブール値、日付、キーワード、ジオポイントなどのさまざまなデータ型を持つフィールドを保存できます。また、保存されたベクトルにさらにコンテキストを追加するための説明情報用のテキストを含むフィールドを保存することもできます。データ型を同一場所に配置することで、複雑さが軽減され、保守性が向上するとともに、データの重複、バージョン互換性の問題、ライセンスの問題を回避できます。

NextGen ベクトル検索コレクション

NextGen ベクトル検索は、ワークロードに基づいてオンデマンドでスケールし、コストとパフォーマンスのバランスを最適化します。アクティブな検索リクエストを処理するために必要なデータブロックのみがメモリにロードされ、ワーカーは必要なメモリと CPU リソースに基づいて動的にスケーリングされます。コレクションが進行中のリクエストなしでアイドル状態の場合、インデックス作成と検索の両方が 0 にスケールされるため、追加のコスト削減が可能になります。デフォルトで

は、NextGen には、コストとレイテンシーを削減しながらリコールを改善する最適化が組み込まれています。

- カスタムドキュメント ID – カスタムドキュメント IDsは NextGen コレクションでサポートされているため、ユーザーはユーザーが指定した IDs。
- 32x 圧縮インデックス – すべてのインデックスは、デフォルトで高度な 32x 圧縮手法で作成されます。デフォルトの圧縮レベルを上書きし、サポートされている圧縮レベルとして 1x、2x、8x、16x、または 32x (デフォルト) を選択できます。
- インデックス構築アクセラレーション – GPU アクセラレーションは、大規模ベクトルインデックスをより迅速かつ効率的に構築できるようにデフォルトで有効になっています。これにより、データをベクトルインデックスにインデックス化するために必要な時間が短縮され、高スループットのインデックス作成エクスペリエンスとコスト削減が実現します。GPU リソースは、インデックス構築オペレーション中に必要な場合にのみプロビジョニングされます。設定 を使用して、GPU の使用状況をインデックスごとに制御できます`index.knn.remote_index_build.enabled`。詳細については、「the section called “ベクトルインデックス作成の GPU アクセラレーション”」を参照してください。
- 簡易 API – NextGen ベクトル検索コレクションでは、インデックスマッピングに `engine` および `mode`パラメータは必要ありません。システムは最適な設定を内部で自動的に決定し、インデックス作成の複雑さを軽減します。
- 最適化検索レスポンス – デフォルトでは、NextGen ベクトルコレクションの検索レスポンスは、結果から元のベクトルを除外します。これにより、エンドツーエンドの検索レイテンシーとレスポンスペイロードのサイズが削減されます。検索レスポンスにベクトルを含めるには、「」を参照してくださいthe section called “ベクトルを含む完全なドキュメントを取得する”。
- NextGen ベクトルコレクションでは、read-after-writeレイテンシー (`refresh_interval`) は 10 秒です。

ベクトル検索コレクションの開始方法

このチュートリアルでは、次のステップを実行して、ベクトル埋め込みをリアルタイムで保存、検索、取得します。

1. アクセス許可を設定する
2. コレクションを作成する
3. データをアップロードおよび検索する
4. コレクションを削除する

ステップ 1: アクセス許可を設定する

このチュートリアルを完了するには (および OpenSearch Serverless を一般的に使用するには)、正しい AWS Identity and Access Management (IAM) アクセス許可が必要です。このチュートリアルでは、コレクションを作成し、データをアップロードして検索を行い、最後にそのコレクションを削除します。

ユーザーまたはロールには、以下の最低限の許可を含む ID ベースのポリシーが、アタッチされている必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "aoss:CreateCollection",
        "aoss:ListCollections",
        "aoss:BatchGetCollection",
        "aoss:DeleteCollection",
        "aoss:CreateAccessPolicy",
        "aoss:ListAccessPolicies",
        "aoss:UpdateAccessPolicy",
        "aoss:CreateSecurityPolicy",
        "iam:ListUsers",
        "iam:ListRoles"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

OpenSearch Serverless での IAM アクセス許可の詳細については、「the section called “Identity and Access Management”」を参照してください。

ステップ 2: コレクションを作成する

新しいコレクションを作成するには、暗号化、ネットワーク、データアクセスポリシーを自動的に設定する統合コレクション作成フロー (NextGen Express Create) に従います。手順については、「NextGen コレクションを作成する (Express Create)」を参照してください。

このチュートリアルの残りの部分では、サンプルコレクションの名前は housingで、NextGen ベクトル検索コレクションです。

> **Note**
>
> 代わりに Classic ベクトルコレクションを作成する場合は、 Classic コレクションに固有の手順Classic ベクトルコレクションの使用については、「」を参照してください。

ステップ 3: データをアップロードして検索する

インデックスとは、ベクトル埋め込みやその他のフィールドを保存、検索、取得する方法を提供する共通のデータスキーマを持ったドキュメントのコレクションです。OpenSearch Dashboards の 開発ツールコンソールや、Postman または awscurl のような HTTP ツールを使用して、OpenSearch Serverless コレクションのインデックスを作成し、データをアップロードできます。このチュートリアルでは、開発ツールを使用します。Python SDK を使用したプログラムによるアクセスについては、「」を参照してくださいthe section called “コレクションへのデータの取り込み”。

住宅コレクション内のデータのインデックス作成と検索を行うには

1. 新しいコレクションのインデックスを作成するには、開発ツールコンソールで次のリクエストを送信します。デフォルトでは、ユークリッド距離と 32 倍圧縮のインデックスが作成されます。

```
PUT housing-index
{
  "settings": {
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "housing-vector": {
        "type": "knn_vector",
        "dimension": 3,
        "space_type": "l2"
      },
```

```
      "title": {
        "type": "text"
      },
      "price": {
        "type": "long"
      },
      "location": {
        "type": "geo_point"
      }
    }
  }
}
```

2. 別の圧縮レベルを使用するには、フィールドマッピングcompression_levelで を設定します。次の例では、 を 1x compression_levelに設定してインデックスを作成します。

```
PUT housing-index-1x
{
  "settings": {
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "housing-vector": {
        "type": "knn_vector",
        "dimension": 3,
        "compression_level": "1x",
        "space_type": "l2"
      },
      "title": {
        "type": "text"
      },
      "price": {
        "type": "long"
      },
      "location": {
        "type": "geo_point"
      }
    }
  }
}
```

サポートされている圧縮レベルは、1x、2x、8x、16x、32x です。

3. ドキュメントを にインデックスを作成するにはhousing-index、システム生成 ID (POST) またはユーザー提供 ID (PUT) を使用できます。

```
# System-generated document ID
POST housing-index/_doc
{
  "housing-vector": [10, 20, 30],
  "title": "2 bedroom in downtown Seattle",
  "price": "2800",
  "location": "47.71, 122.00"
}

# User-provided document ID
PUT housing-index/_doc/100
{
  "housing-vector": [10, 20, 30],
  "title": "2 bedroom in downtown Seattle",
  "price": "2800",
  "location": "47.71, 122.00"
}
```

4. インデックス内のプロパティに似たプロパティを検索するには、次のクエリを送信します。デフォルトでは、検索レスポンスは元のベクトルを から除外_sourceして、レイテンシーとペイロードサイズを減らします。

```
GET housing-index/_search
{
  "size": 5,
  "query": {
    "knn": {
      "housing-vector": {
        "vector": [10, 20, 30],
        "k": 2
      }
    }
  }
}
```

レスポンスは、 housing-vector フィールドを から除外します_source。

```
{
  "took": 10,
```

```
  "timed_out": false,
  "_shards": {
    "total": 0,
    "successful": 0,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "housing-index",
        "_id": "100",
        "_score": 1,
        "_source": {
          "price": "2800",
          "location": "47.71, 122.00",
          "title": "2 bedroom in downtown Seattle"
        }
      }
    ]
  }
}
```

ベクトルを含む完全なドキュメントを取得する

デフォルトの動作を上書きするには、検索リクエストで _source を true に設定します。の includes/excludes オプションを使用して_source、特定のフィールドを取得することもできます。

```
GET housing-index/_search
{
  "size": 5,
  "_source": true,
  "query": {
    "knn": {
      "housing-vector": {
        "vector": [10, 20, 30],
```

```
        "k": 2
      }
    }
  }
}
```

レスポンスに の housing-vectorフィールドが含まれるようになりました_source。

```
{
  "took": 10,
  "timed_out": false,
  "_shards": {
    "total": 0,
    "successful": 0,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "housing-index",
        "_id": "100",
        "_score": 1,
        "_source": {
          "housing-vector": [10, 20, 30],
          "price": "2800",
          "location": "47.71, 122.00",
          "title": "2 bedroom in downtown Seattle"
        }
      }
    ]
  }
}
```

ステップ 4: コレクションを削除する

住宅コレクションはテスト用であるため、実験が終了したら削除してください。

OpenSearch Serverless コレクションを削除するには

1. Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、コレクションを選択し、住宅コレクションを選択します。
3. [削除] を選択し、削除を確定します。

フィルタリングされた検索

フィルターを使用して、セマンティック検索の結果を絞り込むことができます。インデックスを作成し、ドキュメントに対してフィルター検索を実行するには、前のチュートリアルの「データをアップロードおよび検索する」を次の手順に置き換えます。他のステップは同じです。フィルターの詳細については、「フィルターを使用した k-NN 検索」を参照してください。

住宅コレクション内のデータのインデックス作成と検索を行うには

1. コレクション用に単一のインデックスを作成するには、開発ツールコンソールで次のリクエストを送信します。

```
PUT housing-index-filtered
{
  "settings": {
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "housing-vector": {
        "type": "knn_vector",
        "dimension": 3,
        "space_type": "l2",
        "method": {
          "name": "hnsw"
        }
      },
      "title": {
        "type": "text"
      },
      "price": {
        "type": "long"
      },
      "location": {
        "type": "geo_point"
```

```
      }
    }
  }
}
```

2. 1 つのドキュメントを housing-index-filtered にインデックスするために、次のリクエストを送信します。

```
POST housing-index-filtered/_doc
{
  "housing-vector": [
    10,
    20,
    30
  ],
  "title": "2 bedroom in downtown Seattle",
  "price": "2800",
  "location": "47.71, 122.00"
}
```

3. 地理的位置から指定された距離内にある、指定された価格未満のシアトルのアパートのデータを検索するには、次のリクエストを送信します。

```
GET housing-index-filtered/_search
{
  "size": 5,
  "query": {
    "knn": {
      "housing-vector": {
        "vector": [
          0.1,
          0.2,
          0.3
        ],
        "k": 5,
        "filter": {
          "bool": {
            "must": [
              {
                "query_string": {
                  "query": "Find me 2 bedroom apartment in Seattle under $3000 ",
                  "fields": [
                    "title"
```

```
                ]
              }
            },
            {
              "range": {
                "price": {
                  "lte": 3000
                }
              }
            },
            {
              "geo_distance": {
                "distance": "100miles",
                "location": {
                  "lat": 48,
                  "lon": 121
                }
              }
            }
          ]
        }
      }
    }
  }
}
```

制限事項

- 放射検索は、32 倍圧縮を使用する NextGen ベクトルインデックスではサポートされていません。

Classic ベクトルコレクションの使用

クラシックベクトルコレクションは、OpenSearch Serverless ベクトル検索の元の世代です。既存の Classic ベクトルコレクションがある場合は、このセクションの手順を使用します。新しいコレクションについては、NextGen をお勧めします。NextGen ベクトルコレクションを作成するにはコレクションの作成、「」を参照してください。

> **Note**
>
> - Amazon OpenSearch Serverless Classic コレクションは、32 ビット浮動ベクトルと 16 ビットベクトル間の変換を実行する Faiss 16 ビットスカラー量子化をサポートしています。詳細については、「Faiss 16-bit scalar quantization」を参照してください。バイナリベクトルを使用してメモリコストを削減することもできます。詳細については、「Binary vectors」を参照してください。
> - Amazon OpenSearch Serverless Classic コレクションはディスクベースのベクトル検索をサポートしているため、メモリの少ない環境でのベクトルワークロードの運用コストを大幅に削減できます。詳細については、「Disk-based vector search」を参照してください。

データのアップロードと検索 (Classic)

次の例は、デフォルトでnmslibエンジンを使用し、検索レスポンスに元のベクトルを含む Classic ベクトルコレクションに適用されます。

Classic 住宅コレクションのデータのインデックス作成と検索を行うには

1. Classic コレクションのインデックスを作成するには、開発ツールコンソールで次のリクエストを送信します。デフォルトでは、エンジンnmslibとユークリッド距離を持つインデックスが作成されます。

```
PUT housing-index
{
   "settings": {
      "index.knn": true
   },
   "mappings": {
      "properties": {
         "housing-vector": {
            "type": "knn_vector",
            "dimension": 3
         },
         "title": {
            "type": "text"
         },
         "price": {
            "type": "long"
         },
```

```
          "location": {
             "type": "geo_point"
          }
       }
    }
}
```

2. 1 つのドキュメントを housing-index にインデックスするために、次のリクエストを送信します。

```
POST housing-index/_doc
{
  "housing-vector": [
    10,
    20,
    30
  ],
  "title": "2 bedroom in downtown Seattle",
  "price": "2800",
  "location": "47.71, 122.00"
}
```

3. インデックス内のプロパティに類似したプロパティを検索するには、次のクエリを送信します。

```
GET housing-index/_search
{
    "size": 5,
    "query": {
        "knn": {
            "housing-vector": {
                "vector": [
                    10,
                    20,
                    30
                ],
                "k": 5
            }
        }
    }
}
```

フィルタリングされた検索 (Classic)

インデックスを作成し、フィルタリングされた検索を実行するには、次の例を使用します。フィルターの詳細については、「フィルターを使用した k-NN 検索」を参照してください。

Classic 住宅コレクションに対してフィルタリングされた検索をインデックス化して実行するには

1. コレクション用に単一のインデックスを作成するには、開発ツールコンソールで次のリクエストを送信します。

```
PUT housing-index-filtered
{
  "settings": {
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "housing-vector": {
        "type": "knn_vector",
        "dimension": 3,
        "method": {
          "engine": "faiss",
          "name": "hnsw"
        }
      },
      "title": {
        "type": "text"
      },
      "price": {
        "type": "long"
      },
      "location": {
        "type": "geo_point"
      }
    }
  }
}
```

2. 1 つのドキュメントを housing-index-filtered にインデックスするために、次のリクエストを送信します。

```
POST housing-index-filtered/_doc
{
```

```
  "housing-vector": [
    10,
    20,
    30
  ],
  "title": "2 bedroom in downtown Seattle",
  "price": "2800",
  "location": "47.71, 122.00"
}
```

3. 地理的位置から指定された距離内にある、指定された価格未満のシアトルのアパートのデータを検索するには、次のリクエストを送信します。

```
GET housing-index-filtered/_search
{
  "size": 5,
  "query": {
    "knn": {
      "housing-vector": {
        "vector": [
          0.1,
          0.2,
          0.3
        ],
        "k": 5,
        "filter": {
          "bool": {
            "must": [
              {
                "query_string": {
                  "query": "Find me 2 bedroom apartment in Seattle under $3000 ",
                  "fields": [
                    "title"
                  ]
                }
              },
              {
                "range": {
                  "price": {
                    "lte": 3000
                  }
                }
              },
```

```
              {
                "geo_distance": {
                  "distance": "100miles",
                  "location": {
                    "lat": 48,
                    "lon": 121
                  }
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

制限 (Classic)

クラシックベクトルコレクションには以下の制限があります。

- ベクトル検索コレクションは、Apache Lucene ANN エンジンをサポートしていません。
- ベクトル検索コレクションは、Faiss を使用した HNSW アルゴリズムのみをサポートします。IVF または IVFQ はサポートされていません。
- ベクトル検索コレクションは、ウォームアップ、統計、モデルトレーニング API をサポートしていません。
- ベクトル検索コレクションは、インラインスクリプトまたはストアドスクリプトをサポートしていません。
- インデックスカウント情報は、ベクトル検索コレクションの AWS マネジメントコンソール では使用できません。
- ベクトル検索コレクションのインデックスの更新間隔は 60 秒です。

数十億規模のワークロード

クラシックベクトル検索コレクションは、数十億のベクトルを持つワークロードをサポートします。自動スケーリングがこれを行うため、スケーリングの目的でインデックスを再作成する必要はありません。ディメンションの数が多い数百万 (またはそれ以上) のベクトルがあり、200 を超える OCUs

が必要な場合は、 AWS サポートに連絡して、アカウントの OpenSearch Compute Units (OCUs) の最大数を増やしてください。

次の手順

ベクトル検索コレクションとインデックスデータを作成する方法がわかったので、次の演習を試すことができます。

- OpenSearch Python クライアントを使用して、ベクトル検索コレクションを操作します。GitHub でこのチュートリアルを参照してください。
- OpenSearch Java クライアントを使用して、ベクトル検索コレクションを操作します。GitHub でこのチュートリアルを参照してください。
- OpenSearch をベクトルストアとして使用するように LangChain を設定します。LangChain は、言語モデルを利用してアプリケーションを開発するためのオープンソースフレームワークです。詳細については、LangChain ドキュメントを参照してください。

Amazon OpenSearch Serverless でデータライフサイクルポリシーを使用する

Amazon OpenSearch Serverless のデータライフサイクルポリシーは、OpenSearch Serverless が時系列コレクションにデータを保持する期間を定義します。例えば、OpenSearch Serverless がログデータを削除する前に、ログデータを 30 日間保持するようにポリシーを設定できます。

AWS アカウント内の各時系列コレクションのインデックスごとに、個別のポリシーを設定できます。OpenSearch Serverless は、少なくともポリシーで指定した期間ドキュメントを保持します。その後、ベストエフォート方式で、通常は 48 時間以内または保存期間の 10% 以内 (どちらか長い方) にドキュメントを自動的に削除していきます。

データライフサイクルポリシーをサポートしているのは「時系列」コレクションだけです。検索コレクションとベクトル検索コレクションはそうではありません。

トピック

- データライフサイクルポリシー
- 必要なアクセス許可
- ポリシーの優先順位
- ポリシー構文
- データライフサイクルポリシーを作成する

- データライフサイクルポリシーを更新する
- データライフサイクルポリシーを削除する

データライフサイクルポリシー

データライフサイクルポリシーでは、一連の「ルール」を指定します。データライフサイクルポリシーでは、これらのルールに一致するインデックスまたはコレクションに関連するデータの保持期間を管理できます。これらのルールは、インデックスまたはインデックスグループ内のデータの保存期間を定義します。各ルールは、リソースタイプ (index)、保持期間、および保持期間が適用されるリソース (インデックス) のリストで構成されています。

保持期間は、以下のいずれかの形式で定義します。

- `"MinIndexRetention": "24h"` — OpenSearch Serverless は、指定された期間、インデックスデータを時間単位または日単位で保持します。この期間は、`24h` ~ `3650d` で設定できます。
- `"NoMinIndexRetention": true` — OpenSearch Serverless はインデックスデータを無期限に保持します。

次のサンプルポリシーでは、最初のルールがコレクション `marketing` 内のすべてのインデックスに対して 15 日間の保持期間を指定します。2 番目のルールは、`finance` コレクション内の `log` で始まるすべてのインデックス名には保存期間が設定されず、無期限に保持されることを指定しています。

```
{
   "lifeCyclePolicyDetail": {
      "type": "retention",
      "name": "my-policy",
      "policyVersion": "MTY4ODI0NTM2OTk1N18x",
      "policy": {
         "Rules": [
            {
            "ResourceType":"index",
            "Resource":[
               "index/marketing/*"
            ],
            "MinIndexRetention": "15d"
         },
         {
            "ResourceType":"index",
```

```
            "Resource":[
               "index/finance/log*"
            ],
            "NoMinIndexRetention": true
         }
         ]
      },
      "createdDate": 1688245369957,
      "lastModifiedDate": 1688245369957
   }
}
```

以下のサンプルポリシールールでは、OpenSearch Serverless はアカウント内のすべてのコレクションのすべてのインデックスのデータを無期限に保持します。

```
{
   "Rules": [
      {
         "ResourceType": "index",
         "Resource": [
            "index/*/*"
         ]
      }
   ],
   "NoMinIndexRetention": true
}
```

必要なアクセス許可

OpenSearch Serverless のライフサイクルポリシーでは、次の AWS Identity and Access Management (IAM) アクセス許可を使用します。IAM 条件を指定して、特定のコレクションとインデックスに関連付けられたデータライフサイクルポリシーにユーザーを制限できます。

- `aoss:CreateLifecyclePolicy` – データライフサイクルポリシーを作成します。
- `aoss:ListLifecyclePolicies` – 現在のアカウント内のすべてのデータライフサイクルポリシーを一覧表示します。
- `aoss:BatchGetLifecyclePolicy` – アカウントまたはポリシー名に関連付けられたデータライフサイクルポリシーを表示します。
- `aoss:BatchGetEffectiveLifecyclePolicy` – 特定のリソース (`index` がサポートされている唯一のリソース) のデータライフサイクルポリシーを表示します。

- aoss:UpdateLifecyclePolicy – 特定のデータライフサイクルポリシーを変更し、その保持設定またはリソースを変更します。
- aoss:DeleteLifecyclePolicy – データライフサイクルポリシーを削除します。

次の ID ベースのアクセスポリシーにより、ユーザーはすべてのデータライフサイクルポリシーを表示し、リソースパターン index/application-logs を含むポリシーの更新が行えるようになります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aoss:UpdateLifecyclePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aoss:collection": "application-logs"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "aoss:ListLifecyclePolicies",
                "aoss:BatchGetLifecyclePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

ポリシーの優先順位

ポリシー内またはポリシー間に、複数のデータライフサイクルポリシールールが同時に存在する場合があります。この場合、複数のルールに共通するインデックスが存在するのであれば、そのインデックスに対してより具体的なリソース名やパターンを持つルールのほうが、同じインデックスに対してより一般的なリソース名やパターンを持つルールよりも優先されます。

例えば、次のポリシーでは、1 つのインデックス index/sales/logstash に 2 つのルールが適用されます。この場合、index/sales/log* が index/sales/logstash に対する最長の一致なので 2 番目のルールが優先されます。そのため、OpenSearch Serverless はインデックスの保存期間を設定しません。

```
{
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/sales/*",
            ],
            "MinIndexRetention": "15d"
         },
         {
            "ResourceType":"index",
            "Resource":[
               "index/sales/log*",
            ],
            "NoMinIndexRetention": true
         }
      ]
   }
```

ポリシー構文

1 つ以上のルールを指定します。これらのルールは OpenSearch Serverless インデックスのデータライフサイクル設定を定義します。

各ルールには以下の要素が含まれます。MinIndexRetention または NoMinIndexRetention を指定できますが、両方を指定することはできません。

Element	説明
リソースタイプ	ルールが適用されるリソースのタイプ。データライフサイクルポリシーでサポートされる唯一のオプションは index です。
[リソース]	リソース名やパターンのリスト。パターンはプレフィックスとワイルドカード (*) で構成されており、関連付けられたアクセス許可を複数のリソースに適用することができます。例えば、index/*<collection-name|pattern>* /*<index-name|pattern>* 。
MinIndexRetention	ドキュメントをインデックスに保持する最小期間: 日 (d) または時間 (h)。下限は 24h で、上限は 3650d です。
NoMinIndexRetention	true の場合、OpenSearch Serverless はドキュメントを無期限に保持します。

次の例では、最初のルールが autoparts-inventory パターン (index/autoparts-inventory/*) のすべてのインデックスに適用され、削除やアーカイブなどのアクションが発生する前に、少なくとも 20 日間データを保持することを義務付けます。

2 番目のルールは、auto*/gear パターン (index/auto*/gear) に一致するインデックスをターゲットとし、最小保持期間を 24 時間に設定します。

3 番目のルールは特に tires インデックスに適用されるもので、最小保持期間はありません。つまり、このインデックスのデータはすぐに削除またはアーカイブするか、他の基準に基づいて処理することができます。これらのルールは、保持時間が異なる、または保持制限のないインデックスデータの保持を管理するのに役立ちます。

```
{
  "Rules": [
    {
      "ResourceType": "index",
      "Resource": [
        "index/autoparts-inventory/*"
```

```
      ],
      "MinIndexRetention": "20d"
    },
    {
      "ResourceType": "index",
      "Resource": [
        "index/auto*/gear"
      ],
      "MinIndexRetention": "24h"
    },
    {
      "ResourceType": "index",
      "Resource": [
        "index/autoparts-inventory/tires"
      ],
      "NoMinIndexRetention": true
    }
  ]
}
```

データライフサイクルポリシーを作成する

データライフサイクルポリシーを作成するには、指定された基準に基づいてデータの保持と削除を管理するルールを定義します。

コンソール

データライフサイクルポリシーを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで、[データライフサイクルポリシー] を選択します。
3. [データライフサイクルポリシーの作成] を選択します。
4. ポリシーに、わかりやすい名前を入力します。
5. [データライフサイクル] で、[追加] を選択し、ポリシーのコレクションとインデックスを選択します。

   まず、インデックスが属するコレクションを選択します。次に、リストからインデックスを選択するか、インデックスパターンを入力します。すべてのコレクションをソースとして選択するには、アスタリスク (*) を入力します。

6. [データ保持] では、データを無期限に保持するか、[無制限 (削除しない)] の選択を解除して、OpenSearch Serverless が Amazon S3 からデータを自動的に削除するまでの期間を指定できます。
7. [保存]、[作成] の順に選択します。

AWS CLI

を使用してデータライフサイクルポリシーを作成するには AWS CLI、以下のオプションを指定して create-lifecycle-policy コマンドを使用します。

- `--name` – ポリシーの名前。
- `--type` – ポリシーのタイプ。現在、使用できる値は `retention` のみです。
- `--policy` – データライフサイクルポリシー。このパラメータは、インラインポリシーと .json ファイルの両方を受け入れます。インラインポリシーは JSON エスケープ文字列としてエンコードする必要があります。このポリシーをファイルで指定するには、`--policy file://my-policy.json` の形式を使用します。

Example

```
aws opensearchserverless create-lifecycle-policy \
  --name my-policy \
  --type retention \
  --policy "{\"Rules\":[{\"ResourceType\":\"index\",\"Resource\":[\"index/autoparts-inventory/*\"],\"MinIndexRetention\": \"81d\"},{\"ResourceType\":\"index\",\"Resource\":[\"index/sales/orders*\"],\"NoMinIndexRetention\":true}]}"
```

データライフサイクルポリシーを更新する

データライフサイクルポリシーを更新するには、既存のルールを変更して、データ保持または削除の要件の変更を反映させます。これにより、データ管理のニーズの変化に応じてポリシーを調整することができます。

ポリシーを更新してから OpenSearch Serverless が新しい保持期間の適用を開始するまでに数分かかる場合があります。

コンソール

データライフサイクルポリシーを更新するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで、[データライフサイクルポリシー] を選択します。
3. 更新するデータライフサイクルポリシーを選択し、[編集] を選択します。
4. ビジュアルエディタまたは JSON エディタを使用してポリシーを編集します。
5. [保存] を選択します。

AWS CLI

を使用してデータライフサイクルポリシーを更新するには AWS CLI、update-lifecycle-policy コマンドを使用します。

リクエストに `--policy-version` パラメータを含める必要があります。list-lifecycle-policies または batch-get-lifecycle-policy コマンドを使用して、ポリシーバージョンを取得できます。他のユーザーによる変更を誤って上書きしないように、最新のポリシーバージョンを含めることをお勧めします。

次のリクエストは、データライフサイクルポリシーを新しいポリシーの JSON ドキュメントで更新します。

Example

```
aws opensearchserverless update-lifecycle-policy \
  --name my-policy \
  --type retention \
  --policy-version MTY2MzY5MTY1MDA3Ml8x \
  --policy file://my-new-policy.json
```

データライフサイクルポリシーを削除する

データライフサイクルポリシーを削除すると、OpenSearch Serverless はすべての一致するインデックスにそのポリシーを適用しなくなります。

コンソール

データライフサイクルポリシーを削除するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで、[データライフサイクルポリシー] を選択します。
3. 削除するポリシーを選択し、[削除] を選択して削除されたことを確認します。

AWS CLI

を使用してデータライフサイクルポリシーを削除するには AWS CLI、delete-lifecycle-policy コマンドを使用します。

Example

```
aws opensearchserverless delete-lifecycle-policy \
  --name my-policy \
  --type retention
```

AWS SDKs を使用して Amazon OpenSearch Serverless を操作する

このセクションでは、 AWS SDKs を使用して Amazon OpenSearch Serverless を操作する方法の例を示します。これらのコードサンプルには、セキュリティポリシーとコレクションの作成方法が示されています。

> **Note**
> コードサンプルは、現在作成中です。コードサンプル (Java、Go など) を提供したい場合は、GitHub リポジトリ内で直接プルリクエストを開きます。

トピック

- Python
- JavaScript

Python

次のサンプルスクリプトでは、Python の opensearch-py クライアントおよび AWS SDK for Python (Boto3) を使用して、暗号化、ネットワーク、データアクセスポリシーを作成します。また、一致するコレクションを作成し、いくつかのサンプルデータをインデックス化します。

必要な従属関係をインストールには、次のコマンドを実行します。

```
pip install opensearch-py
pip install boto3
pip install botocore
pip install requests-aws4auth
```

スクリプト内の Principal 要素を、リクエストに署名するユーザーまたはロールの Amazon リソースネーム (ARN) に置き換えます。必要に応じて、region を変更することもできます。

```
from opensearchpy import OpenSearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth
import boto3
import botocore
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

client = boto3.client('opensearchserverless')
service = 'aoss'
region = 'us-east-1'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key,
                   region, service, session_token=credentials.token)


def createEncryptionPolicy(client):
    """Creates an encryption policy that matches all collections beginning with tv-"""
    try:
        response = client.create_security_policy(
            description='Encryption policy for TV collections',
            name='tv-policy',
            policy="""
                {
                    \"Rules\":[
```

```
                        {
                            \"ResourceType\":\"collection\",
                            \"Resource\":[
                                \"collection\/tv-*\"
                            ]
                        }
                    ],
                    \"AWSOwnedKey\":true
                }
                """,
            type='encryption'
        )
        print('\nEncryption policy created:')
        print(response)
    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ConflictException':
            print(
                '[ConflictException] The policy name or rules conflict with an existing
 policy.')
        else:
            raise error


def createNetworkPolicy(client):
    """Creates a network policy that matches all collections beginning with tv-"""
    try:
        response = client.create_security_policy(
            description='Network policy for TV collections',
            name='tv-policy',
            policy="""
                [{
                    \"Description\":\"Public access for TV collection\",
                    \"Rules\":[
                        {
                            \"ResourceType\":\"dashboard\",
                            \"Resource\":[\"collection\/tv-*\"]
                        },
                        {
                            \"ResourceType\":\"collection\",
                            \"Resource\":[\"collection\/tv-*\"]
                        }
                    ],
                    \"AllowFromPublic\":true
                }]
```

```
                """,
            type='network'
        )
        print('\nNetwork policy created:')
        print(response)
    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ConflictException':
            print(
                '[ConflictException] A network policy with this name already exists.')
        else:
            raise error


def createAccessPolicy(client):
    """Creates a data access policy that matches all collections beginning with tv-"""
    try:
        response = client.create_access_policy(
            description='Data access policy for TV collections',
            name='tv-policy',
            policy="""
                [{
                    \"Rules\":[
                        {
                            \"Resource\":[
                                \"index\/tv-*\/*\"
                            ],
                            \"Permission\":[
                                \"aoss:CreateIndex\",
                                \"aoss:DeleteIndex\",
                                \"aoss:UpdateIndex\",
                                \"aoss:DescribeIndex\",
                                \"aoss:ReadDocument\",
                                \"aoss:WriteDocument\"
                            ],
                            \"ResourceType\": \"index\"
                        },
                        {
                            \"Resource\":[
                                \"collection\/tv-*\"
                            ],
                            \"Permission\":[
                                \"aoss:CreateCollectionItems\"
                            ],
                            \"ResourceType\": \"collection\"
```

```
                        }
                    ],
                    \"Principal\":[
                        \"arn:aws:iam::123456789012:role\/Admin\"
                    ]
                }]
                """,
            type='data'
        )
        print('\nAccess policy created:')
        print(response)
    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ConflictException':
            print(
                '[ConflictException] An access policy with this name already exists.')
        else:
            raise error


def createCollection(client):
    """Creates a collection"""
    try:
        response = client.create_collection(
            name='tv-sitcoms',
            type='SEARCH'
        )
        return(response)
    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ConflictException':
            print(
                '[ConflictException] A collection with this name already exists. Try
 another name.')
        else:
            raise error


def waitForCollectionCreation(client):
    """Waits for the collection to become active"""
    response = client.batch_get_collection(
        names=['tv-sitcoms'])
    # Periodically check collection status
    while (response['collectionDetails'][0]['status']) == 'CREATING':
        print('Creating collection...')
        time.sleep(30)
```

```
        response = client.batch_get_collection(
            names=['tv-sitcoms'])
    print('\nCollection successfully created:')
    print(response["collectionDetails"])
    # Extract the collection endpoint from the response
    host = (response['collectionDetails'][0]['collectionEndpoint'])
    final_host = host.replace("https://", "")
    indexData(final_host)


def indexData(host):
    """Create an index and add some sample data"""
    # Build the OpenSearch client
    client = OpenSearch(
        hosts=[{'host': host, 'port': 443}],
        http_auth=awsauth,
        use_ssl=True,
        verify_certs=True,
        connection_class=RequestsHttpConnection,
        timeout=300
    )
    # It can take up to a minute for data access rules to be enforced
    time.sleep(45)

    # Create index
    response = client.indices.create('sitcoms-eighties')
    print('\nCreating index:')
    print(response)

    # Add a document to the index.
    response = client.index(
        index='sitcoms-eighties',
        body={
            'title': 'Seinfeld',
            'creator': 'Larry David',
            'year': 1989
        },
        id='1',
    )
    print('\nDocument added:')
    print(response)


def main():
```

```
    createEncryptionPolicy(client)
    createNetworkPolicy(client)
    createAccessPolicy(client)
    createCollection(client)
    waitForCollectionCreation(client)


if __name__ == "__main__":
    main()
```

JavaScript

次のサンプルスクリプトでは、JavaScript の opensearch-js クライアントおよび SDK for JavaScript in Node.js を使用して、暗号化、ネットワーク、データアクセスポリシーを作成します。また、一致するコレクションおよびインデックスを作成して、いくつかのサンプルデータをインデックス化します。

必要な従属関係をインストールには、次のコマンドを実行します。

```
npm i aws-sdk
npm i aws4
npm i @opensearch-project/opensearch
```

スクリプト内の `Principal` 要素を、リクエストに署名するユーザーまたはロールの Amazon リソースネーム (ARN) に置き換えます。必要に応じて、`region` を変更することもできます。

```
var AWS = require('aws-sdk');
var aws4 = require('aws4');
var {
    Client,
    Connection
} = require("@opensearch-project/opensearch");
var {
    OpenSearchServerlessClient,
    CreateSecurityPolicyCommand,
    CreateAccessPolicyCommand,
    CreateCollectionCommand,
    BatchGetCollectionCommand
} = require("@aws-sdk/client-opensearchserverless");
var client = new OpenSearchServerlessClient();

async function execute() {
```

```
    await createEncryptionPolicy(client)
    await createNetworkPolicy(client)
    await createAccessPolicy(client)
    await createCollection(client)
    await waitForCollectionCreation(client)
}

async function createEncryptionPolicy(client) {
    // Creates an encryption policy that matches all collections beginning with 'tv-'
    try {
        var command = new CreateSecurityPolicyCommand({
            description: 'Encryption policy for TV collections',
            name: 'tv-policy',
            type: 'encryption',
            policy: " \
        { \
            \"Rules\":[ \
                { \
                    \"ResourceType\":\"collection\", \
                    \"Resource\":[ \
                        \"collection\/tv-*\" \
                    ] \
                } \
            ], \
            \"AWSOwnedKey\":true \
        }"
        });
        const response = await client.send(command);
        console.log("Encryption policy created:");
        console.log(response['securityPolicyDetail']);
    } catch (error) {
        if (error.name === 'ConflictException') {
            console.log('[ConflictException] The policy name or rules conflict with an
 existing policy.');
        } else
            console.error(error);
    };
}

async function createNetworkPolicy(client) {
    // Creates a network policy that matches all collections beginning with 'tv-'
    try {
        var command = new CreateSecurityPolicyCommand({
            description: 'Network policy for TV collections',
```

```
            name: 'tv-policy',
            type: 'network',
            policy: " \
            [{ \
                \"Description\":\"Public access for television collection\", \
                \"Rules\":[ \
                    { \
                        \"ResourceType\":\"dashboard\", \
                        \"Resource\":[\"collection\/tv-*\"] \
                    }, \
                    { \
                        \"ResourceType\":\"collection\", \
                        \"Resource\":[\"collection\/tv-*\"] \
                    } \
                ], \
                \"AllowFromPublic\":true \
            }]"
        });
        const response = await client.send(command);
        console.log("Network policy created:");
        console.log(response['securityPolicyDetail']);
    } catch (error) {
        if (error.name === 'ConflictException') {
            console.log('[ConflictException] A network policy with that name already
 exists.');
        } else
            console.error(error);
    };
}

async function createAccessPolicy(client) {
    // Creates a data access policy that matches all collections beginning with 'tv-'
    try {
        var command = new CreateAccessPolicyCommand({
            description: 'Data access policy for TV collections',
            name: 'tv-policy',
            type: 'data',
            policy: " \
            [{ \
                \"Rules\":[ \
                    { \
                        \"Resource\":[ \
                            \"index\/tv-*\/*\" \
                        ], \
```

```
                        \"Permission\":[ \
                            \"aoss:CreateIndex\", \
                            \"aoss:DeleteIndex\", \
                            \"aoss:UpdateIndex\", \
                            \"aoss:DescribeIndex\", \
                            \"aoss:ReadDocument\", \
                            \"aoss:WriteDocument\" \
                        ], \
                        \"ResourceType\": \"index\" \
                    }, \
                    { \
                        \"Resource\":[ \
                            \"collection\/tv-*\" \
                        ], \
                        \"Permission\":[ \
                            \"aoss:CreateCollectionItems\" \
                        ], \
                        \"ResourceType\": \"collection\" \
                    } \
                ], \
                \"Principal\":[ \
                    \"arn:aws:iam::123456789012:role\/Admin\" \
                ] \
            }]"
        });
        const response = await client.send(command);
        console.log("Access policy created:");
        console.log(response['accessPolicyDetail']);
    } catch (error) {
        if (error.name === 'ConflictException') {
            console.log('[ConflictException] An access policy with that name already
 exists.');
        } else
            console.error(error);
    };
}

async function createCollection(client) {
    // Creates a collection to hold TV sitcoms indexes
    try {
        var command = new CreateCollectionCommand({
            name: 'tv-sitcoms',
            type: 'SEARCH'
        });
```

```
        const response = await client.send(command);
        return (response)
    } catch (error) {
        if (error.name === 'ConflictException') {
            console.log('[ConflictException] A collection with this name already
 exists. Try another name.');
        } else
            console.error(error);
    };
}

async function waitForCollectionCreation(client) {
    // Waits for the collection to become active
    try {
        var command = new BatchGetCollectionCommand({
            names: ['tv-sitcoms']
        });
        var response = await client.send(command);
        while (response.collectionDetails[0]['status'] == 'CREATING') {
            console.log('Creating collection...')
            await sleep(30000) // Wait for 30 seconds, then check the status again
            function sleep(ms) {
                return new Promise((resolve) => {
                    setTimeout(resolve, ms);
                });
            }
            var response = await client.send(command);
        }
        console.log('Collection successfully created:');
        console.log(response['collectionDetails']);
        // Extract the collection endpoint from the response
        var host = (response.collectionDetails[0]['collectionEndpoint'])
        // Pass collection endpoint to index document request
        indexDocument(host)
    } catch (error) {
        console.error(error);
    };
}

async function indexDocument(host) {

    var client = new Client({
        node: host,
        Connection: class extends Connection {
```

```
            buildRequestObject(params) {
                var request = super.buildRequestObject(params)
                request.service = 'aoss';
                request.region = 'us-east-1'; // e.g. us-east-1
                var body = request.body;
                request.body = undefined;
                delete request.headers['content-length'];
                request.headers['x-amz-content-sha256'] = 'UNSIGNED-PAYLOAD';
                request = aws4.sign(request, AWS.config.credentials);
                request.body = body;

                return request
            }
        }
    });

    // Create an index
    try {
        var index_name = "sitcoms-eighties";

        var response = await client.indices.create({
            index: index_name
        });

        console.log("Creating index:");
        console.log(response.body);

        // Add a document to the index
        var document = "{ \"title\": \"Seinfeld\", \"creator\": \"Larry David\", \"year
\": \"1989\" }\n";

        var response = await client.index({
            index: index_name,
            body: document
        });

        console.log("Adding document:");
        console.log(response.body);
    } catch (error) {
        console.error(error);
    };
}
```

```
execute()
```

CloudFormation を使用して Amazon OpenSearch Serverless コレクションを作成する

CloudFormation を使用して、コレクション、セキュリティポリシー、VPC エンドポイントなどの Amazon OpenSearch Serverless リソースを作成できます。OpenSearch Serverless CloudFormation に関する包括的なリファレンスについては、「CloudFormation ユーザーガイド」の「Amazon OpenSearch Serverless」を参照してください。

次の CloudFormation サンプルテンプレートでは、シンプルなデータアクセスポリシー、ネットワークポリシー、セキュリティポリシー、および一致するコレクションが作成されます。これは、Amazon OpenSearch Serverless をすぐに立ち上げて実行し、コレクションを作成して使用するために必要な要素をプロビジョニングする良い方法です。

> **Important**
>
> この例ではパブリックネットワークアクセスを使用していますが、これは本番環境のワークロードにはお勧めしません。コレクションを保護するために VPC アクセスを使用することをお勧めします。詳細については、「AWS::OpenSearchServerless::VpcEndpoint」と「the section called “データプレーン VPC エンドポイント”」を参照してください。

```
AWSTemplateFormatVersion: 2010-09-09
Description: 'Amazon OpenSearch Serverless template to create an IAM user, encryption
 policy, network policy, data access policy and collection'
Resources:
  IAMUSer:
    Type: 'AWS::IAM::User'
    Properties:
      UserName:  aossadmin
  DataAccessPolicy:
    Type: 'AWS::OpenSearchServerless::AccessPolicy'
    Properties:
      Name: quickstart-access-policy
      Type: data
      Description: Access policy for quickstart collection
      Policy: !Sub >-
        [{"Description":"Access for cfn user","Rules":
[{"ResourceType":"index","Resource":["index/*/*"],"Permission":["aoss:*"]},
```

```
        {"ResourceType":"collection","Resource":["collection/quickstart"],"Permission":
["aoss:*"]}],
        "Principal":["arn:aws:iam::${AWS::AccountId}:user/aossadmin"]}]
  NetworkPolicy:
    Type: 'AWS::OpenSearchServerless::SecurityPolicy'
    Properties:
      Name: quickstart-network-policy
      Type: network
      Description: Network policy for quickstart collection
      Policy: >-
        [{"Rules":[{"ResourceType":"collection","Resource":["collection/
quickstart"]}, {"ResourceType":"dashboard","Resource":["collection/
quickstart"]}],"AllowFromPublic":true}]
  EncryptionPolicy:
    Type: 'AWS::OpenSearchServerless::SecurityPolicy'
    Properties:
      Name: quickstart-security-policy
      Type: encryption
      Description: Encryption policy for quickstart collection
      Policy: >-
        {"Rules":[{"ResourceType":"collection","Resource":["collection/
quickstart"]}],"AWSOwnedKey":true}
  Collection:
    Type: 'AWS::OpenSearchServerless::Collection'
    Properties:
      Name: quickstart
      Type: TIMESERIES
      Description: Collection to holds timeseries data
    DependsOn: EncryptionPolicy
Outputs:
  IAMUser:
    Value: !Ref IAMUSer
  DashboardURL:
    Value: !GetAtt Collection.DashboardEndpoint
  CollectionARN:
    Value: !GetAtt Collection.Arn
```

スナップショットを使用したコレクションのバックアップ

スナップショットは、ディザスタリカバリ機能を提供する Amazon OpenSearch Serverless コレクションのポイントインタイムバックアップです。OpenSearch Serverless は、コレクションのスナップショットを自動的に作成および管理し、ビジネス継続性とデータ保護を確保します。各スナップショットには、インデックスメタデータ (インデックスの設定とマッピング) 、クラスターメタデー

タ (インデックステンプレートとエイリアス) 、インデックスデータ (インデックスに保存されているすべてのドキュメントとデータ) が含まれます。

OpenSearch Serverless は、手動設定なし、メンテナンスオーバーヘッドなし、追加のストレージコストなし、偶発的なデータ損失からの迅速な復旧、スナップショットから特定のインデックスを復元する機能を備えた、1 時間ごとの自動バックアップを提供します。

スナップショットを使用する前に、以下の重要な考慮事項を理解してください。スナップショットの作成には時間がかかり、すぐには作成されません。スナップショット作成時の新しいドキュメントや更新は、スナップショットに含まれません。スナップショットは元のコレクションにのみ復元でき、新しいコレクションには復元できません。復元すると、インデックスは元のバージョンとは異なる新しい UUIDs を受け取ります。OpenSearch Serverless で既存のオープンインデックスに復元すると、新しいインデックス名またはプレフィックスパターンが指定されていない場合、そのインデックスのデータが上書きされます。これは OpenSearch コアの動作とは異なります。一度に実行できる復元オペレーションは 1 つだけであり、同じコレクションで複数の復元オペレーションを同時に開始することはできません。アクティブな復元オペレーション中にインデックスを復元しようとすると、オペレーションは失敗します。復元オペレーション中、インデックスへのリクエストは失敗します。

必要なアクセス許可

スナップショットを使用するには、データアクセスポリシーで次のアクセス許可を設定します。データアクセスポリシーの詳細については、「データアクセスポリシーと IAM ポリシーの比較」を参照してください。

データアクセスポリシー	API
aoss:DescribeSnapshot	GET /_cat/snapshots/aoss-automated GET _snapshot/aoss-automated/*snapshot_id* /
aoss:RestoreSnapshot	POST /_snapshot/aoss-automated/*snapshot_id* /_restore
aoss:DescribeCollectionItems	GET /_cat/recovery

> **Note**
>
> を、 から取得した実際のスナップショット ID <snapshot_id>に置き換えますGET /_cat/snapshots/aoss-automated。

ポリシーは、次の AWS CLI コマンドを使用して設定できます。

1. create-access-policy
2. delete-access-policy
3. get-access-policy
4. update-access-policy
5. list-access-policies

アクセスポリシーを作成するための CLI コマンドの例を次に示します。コマンドで、####コンテンツを特定の情報に置き換えます。

```
aws opensearchserverless create-access-policy \
--type data \
--name Example-data-access-policy \
--region aws-region \
--policy '[
  {
    "Rules": [
      {
        "Resource": [
          "collection/Example-collection"
        ],
        "Permission": [
          "aoss:DescribeSnapshot",
          "aoss:RestoreSnapshot",
          "aoss:DescribeCollectionItems"
        ],
        "ResourceType": "collection"
      }
    ],
    "Principal": [
      "arn:aws:iam::111122223333:user/UserName"
    ],
    "Description": "Data policy to support snapshot operations."
```

```
  }
]'
```

スナップショットの使用

デフォルトでは、新しいコレクションを作成すると、OpenSearch Serverless は 1 時間ごとにスナップショットを自動作成します。ご自身では特に何もする必要はありません。各スナップショットには、コレクション内のすべてのインデックスが含まれます。OpenSearch Serverless がスナップショットを作成したら、次の手順を使用してスナップショットを一覧表示し、スナップショットの詳細を確認できます。

スナップショットを一覧表示する

以下の手順に従って、コレクション内のすべてのスナップショットを一覧表示し、その詳細を確認します。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクションの名前を選択して、詳細ページを開きます。
4. スナップショットタブを選択すると、生成されたすべてのスナップショットが表示されます。
5. 以下を含むスナップショット情報を確認します。
   - スナップショット ID - スナップショットの一意の識別子
   - ステータス - 現在の状態 (使用可能、進行中)
   - 作成日時 - スナップショットが作成された日時

OpenSearch API

- 以下のコマンドを使用して、コレクション内のすべてのスナップショットを一覧表示します。

```
GET /_cat/snapshots/aoss-automated
```

> **Note**
>
> この API コールに末尾のスラッシュを含めないでください。GET /_cat/snapshots/aoss-automated/ (末尾にスラッシュを付けて) を使用すると、404 エラーが返されます。

```
id                                status  start_epoch start_time end_epoch
 end_time    duration    indexes successful_shards failed_shards total_shards
snapshot-ExampleSnapshotID1     SUCCESS 1737964331  07:52:11   1737964382
 07:53:02    50.4s       1
snapshot-ExampleSnapshotID2     SUCCESS 1737967931  08:52:11   1737967979
 08:52:59    47.7s       2
snapshot-ExampleSnapshotID3     SUCCESS 1737971531  09:52:11   1737971581
 09:53:01    49.1s       3
snapshot-ExampleSnapshotID4 IN_PROGRESS 1737975131  10:52:11   -          -
       4.8d       3
```

スナップショットの詳細を取得する

次の手順を使用して、特定のスナップショットに関する詳細情報を取得します。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクションの名前を選択して、詳細ページを開きます。
4. [スナップショット] タブを選択します。
5. スナップショットジョブ ID を選択すると、メタデータ、含まれるインデックス、タイミング情報など、スナップショットに関する詳細情報が表示されます。

OpenSearch API

- スナップショットに関する情報を取得するには、次のコマンドを使用します。コマンドで、####コンテンツを特定の情報に置き換えます。

```
GET _snapshot/aoss-automated/snapshot_id/
```

> **Note**
>
> を、 から取得した実際のスナップショット ID <snapshot_id>に置き換えますGET /_cat/snapshots/aoss-automated。

リクエストの例:

```
GET _snapshot/aoss-automated/snapshot-ExampleSnapshotID1/
```

レスポンスの例:

```
{
    "snapshots": [
        {
            "snapshot": "snapshot-ExampleSnapshotID1-5e01-4423-9833Example",
            "uuid": "Example-5e01-4423-9833-9e9eb757Example",
            "version_id": 136327827,
            "version": "2.11.0",
            "remote_store_index_shallow_copy": true,
            "indexes": [
                "Example-index-0117"
            ],
            "data_streams": [],
            "include_global_state": true,
            "metadata": {},
            "state": "SUCCESS",
            "start_time": "2025-01-27T09:52:11.953Z",
            "start_time_in_millis": 1737971531953,
            "end_time": "2025-01-27T09:53:01.062Z",
            "end_time_in_millis": 1737971581062,
            "duration_in_millis": 49109,
            "failures": [],
            "shards": {
                "total": 0,
                "failed": 0,
                "successful": 0
            }
```

```
        }
    ]
}
```

スナップショットレスポンスには、スナップショットオペレーションの一意の識別子idを提供し、現在の状態SUCCESSまたは statusを返しIN_PROGRESS、スナップショットオペレーションの完了にかかった時間durationを示し、スナップショットに含まれるインデックスの数indexesを返します。

スナップショットからの復元

スナップショットから復元すると、以前に取得したバックアップからデータが復元されます。このプロセスは、OpenSearch Serverless でのディザスタリカバリとデータ管理に不可欠です。復元する前に、復元されたインデックスには元のバージョンとは異なる UUIDs があり、OpenSearch Serverless の既存のオープンインデックスに復元すると、そのインデックスのデータが上書きされます。ただし、新しいインデックス名またはプレフィックスパターンが指定されていない場合、スナップショットは元のコレクションにのみ復元でき (クロスコレクション復元はサポートされていません)、復元オペレーションはクラスターのパフォーマンスに影響するため、それに応じて計画してください。

バックアップされたインデックスをスナップショットから復元するには、次の手順に従います。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクションの名前を選択して、詳細ページを開きます。
4. スナップショットタブを選択して、使用可能なスナップショットを表示します。
5. 復元元のスナップショットを選択し、スナップショットから復元を選択します。
6. スナップショットから復元ダイアログで、次の操作を行います。

    - スナップショット名で、選択したスナップショット ID を確認します。
    - スナップショットスコープでは、次のいずれかを選択します。
        - コレクション内のすべてのインデックス - スナップショットからすべてのインデックスを復元する
        - 特定のインデックス - 復元する個々のインデックスを選択する

- Destination で、復元するコレクションを選択します。
- (オプション) 名前変更設定を設定して、復元されたインデックスの名前を変更します。
  - 名前を変更しない - 元のインデックス名を保持する
  - 復元されたインデックス名にプレフィックスを追加する - 競合を避けるためにプレフィックスを追加します
  - 正規表現を使用して名前を変更する - 高度な名前変更パターンを使用する
- (オプション) 復元が完了したとき、またはエラーが発生したときに通知されるように通知設定を構成します。

7. 保存を選択して復元オペレーションを開始します。

OpenSearch API

1. 以下のコマンドを実行して、適切なスナップショットを特定します。

```
GET /_snapshot/aoss-automated/_all
```

スナップショットのリストを絞り込むには、以下のコマンドを実行します。

```
GET /_cat/snapshots/aoss-automated
```

2. スナップショットを復元する前に、以下のコマンドを実行して詳細を確認します。コマンドで、*####*コンテンツを特定の情報に置き換えます。

```
GET _snapshot/aoss-automated/snapshot-ExampleSnapshotID1/
```

3. 特定のスナップショットから復元するには、以下のコマンドを実行します。

```
POST /_snapshot/aoss-automated/snapshot-ID/_restore
```

リクエスト本文を含めることで、復元オペレーションをカスタマイズできます。以下に例を示します。

```
POST /_snapshot/aoss-automated/snapshot-
ExampleSnapshotID1-5e01-4423-9833Example/_restore
{
  "indices": "opensearch-dashboards*,my-index*",
  "ignore_unavailable": true,
```

```
  "include_global_state": false,
  "include_aliases": false,
  "rename_pattern": "opensearch-dashboards(.+)",
  "rename_replacement": "restored-opensearch-dashboards$1"
}
```

4. 復元の進行状況を表示するには、以下のコマンドを実行します。

```
GET /_cat/recovery
```

> **Note**
>
> リクエスト本文を含むコマンドを使用してスナップショットを復元する場合、いくつかのパラメータを使用して復元動作を制御できます。indices パラメータは、復元するインデックスを指定し、ワイルドカードパターンをサポートします。スナップショットにインデックスがない場合でも復元オペレーションを続行するignore_unavailableように を設定します。include_global_state を使用してクラスターの状態を復元するかどうかを判断し、関連付けられたエイリアスを復元するかどうかinclude_aliasesを制御します。rename_pattern および rename_replacementパラメータは、復元オペレーション中にインデックスの名前を変更します。

Amazon OpenSearch Serverless での Zstandard Codec サポート

インデックスコーデックは、インデックスの保存済みフィールドを圧縮してディスクと S3 に保存する方法を決定します。インデックスコーデックは、圧縮アルゴリズムを指定する静的index.codec設定によって制御されます。この設定は、インデックスシャードサイズとインデックスオペレーションのパフォーマンスの両方に影響します。

デフォルトでは、OpenSearch Serverless のインデックスは LZ4 圧縮アルゴリズムでデフォルトのコーデックを使用します。OpenSearch Serverless はzstd、1 から 6 までの設定可能な圧縮レベルを持つ および zstd_no_dictコーデックもサポートしています。

> **Important**
>
> index.codec は静的設定であるため、インデックスの作成後に変更することはできません。

詳細については、OpenSearch Index Codecs ドキュメントを参照してください。

ZSTD コーデックを使用したインデックスの作成

インデックスの作成時に index.codec設定を使用して ZSTD コーデックを指定できます。

```
PUT /your_index
{
  "settings": {
    "index.codec": "zstd"
  }
}
```

圧縮レベル

ZSTD コーデックは、 index.codec.compression_level設定を介してオプションの圧縮レベルをサポートし、範囲 [1, 6] の整数を受け入れます。圧縮レベルが高いほど、圧縮率 (ストレージが小さい) は向上しますが、圧縮と解凍の速度は遅くなります。デフォルトの圧縮レベルは 3 です。

```
PUT /your_index
{
  "settings": {
    "index.codec": "zstd",
    "index.codec.compression_level": 2
  }
}
```

パフォーマンスベンチマーク

nyc_taxi データセットを使用したベンチマークテストに基づいて、ZSTD 圧縮はzstd、、、zstd_no_dictおよび 圧縮レベルのさまざまな組み合わせでベースラインと比較して 26～32% 優れた圧縮を達成しました。

メトリクス	ZSTD L1	ZSTD L6	ZSTD_NO_DICT L1	ZSTD_NO_DICT L6
インデックスサイズの縮小	28.10%	32%	26.90%	28.70%
スループット変更のインデックス作成	-0.50%	-23.80%	-0.50%	-5.30%

メトリクス	ZSTD L1	ZSTD L6	ZSTD_NO_DICT L1	ZSTD_NO_DICT L6
マッチオールクエリ p90 レイテンシーの改善	-16.40%	29.50%	-16.40%	23.40%
範囲クエリ p90 レイテンシーの改善	90.90%	92.40%	-282.90%	92.50%
距離量 p90 Agg レイテンシーの改善	2%	24.70%	2%	13.80%

詳細については、AWS OpenSearch ブログを参照してください。

派生ソースを使用してストレージを保存する

デフォルトでは、OpenSearch Serverless は取り込まれた各ドキュメントを元の JSON ドキュメント本文を含む _sourceフィールドに格納し、検索する個々のフィールドのインデックスを作成します。_source フィールドは検索できませんが、取得や検索などのフェッチリクエストの実行時に完全なドキュメントが返されるように保持されます。派生ソースが有効になっている場合、OpenSearch Serverless は_sourceフィールドの保存をスキップし、代わりに検索、取得、mget、再インデックス、更新オペレーションなど、オンデマンドで動的に再構築します。派生ソース設定を使用すると、ストレージ使用量を最大 50% 削減できます。

設定

インデックスの派生ソースを設定するには、 index.derived_source.enabled設定を使用してインデックスを作成します。

```
PUT my-index1
{
  "settings": {
    "index": {
      "derived_source": {
        "enabled": true
      }
    }
  }
}
```

重要な考慮事項

- 特定のフィールドタイプのみがサポートされています。サポートされているフィールドと制限のリストについては、OpenSearch ドキュメントを参照してください。派生ソースとサポートされていないフィールドを使用してインデックスを作成すると、インデックスの作成は失敗します。派生ソース対応インデックスでサポートされていないフィールドを持つドキュメントを取り込もうとすると、取り込みは失敗します。この機能は、インデックスに追加されるフィールドタイプを認識している場合にのみ使用します。
- 設定`index.derived_source.enabled`は静的設定です。これは、インデックスの作成後に変更することはできません。

クエリレスポンスの制限

派生ソースを有効にすると、クエリレスポンスの生成と返される方法に特定の制限が課されます。

- 複数の形式が指定されている日付フィールドは、元の取り込み形式に関係なく、リクエストされたすべてのドキュメントのリストの最初の形式を常に使用します。
- ジオポイント値は固定`{"lat": lat_val, "lon": lon_val}`形式で返され、ある程度の精度が失われる可能性があります。
- 複数値の配列はソートされ、キーワードフィールドは重複排除される場合があります。

詳細については、OpenSearch ブログを参照してください。

パフォーマンスベンチマーク

nyc_taxi データセットを使用したベンチマークテストに基づいて、派生ソースはベースラインと比較してインデックスサイズを 58% 削減しました。

メトリクス	派生ソース
インデックスサイズの縮小	58.3%
スループット変更のインデックス作成	3.7%
p90 レイテンシー変更のインデックス作成	6.9%
マッチオールクエリ p90 レイテンシーの改善	19%

メトリクス	派生ソース
範囲クエリ p90 レイテンシーの改善	-18.8%
距離量 p90 Agg レイテンシーの改善	-7.3%

詳細については、OpenSearch ブログを参照してください。

Amazon OpenSearch Serverless コレクショングループ

Amazon OpenSearch Serverless のコレクショングループは、複数のコレクションを整理し、異なる KMS キーを持つコレクション間でコンピューティングリソース共有を有効にします。この共有コンピューティングモデルは、KMS キーごとに個別の OpenSearch Compute Units (OCUs) を不要にすることでコストを削減します。

作成した各 OpenSearch Serverless コレクションは、 を使用して AWS KMS 保管中のデータの暗号化で保護され、暗号化キーを保存および管理します。同じコレクショングループ内のコレクションは、暗号化に異なる KMS キーを使用している場合でも、コンピューティングリソースと OCU メモリスペースを共有します。

コレクショングループは、セキュリティとパフォーマンスの要件を分離します。同じ KMS キーを持つコレクションを単一のコレクショングループにグループ化してセキュリティを分離したり、同じグループ内の異なる KMS キーとコレクションを組み合わせてコストを最適化したりできます。この柔軟性により、セキュリティ要件とリソース効率のバランスをとることができます。

コレクショングループにコレクションを追加すると、OpenSearch Serverless はそれをグループの共有コンピューティングリソースに割り当てます。OpenSearch Serverless は、指定された KMS キーを使用して各コレクションのデータを暗号化することで、セキュリティを維持しながら、これらのリソース全体のワークロードの分散を自動的に管理します。アクセスコントロールは引き続きコレクションレベルで適用され、共有コンピューティングリソースは、グループ内のコレクションを提供するために必要に応じて複数の KMS キーにアクセスします。

リソースの割り当てを制御するには、コレクショングループレベルで最小 OCU 制限と最大 OCU 制限を設定します。これらの制限はグループ内のすべてのコレクションに適用され、一貫したパフォーマンスを確保しながらコストを管理するのに役立ちます。

デフォルトでは、コレクショングループ内のコレクション数、コレクション内のインデックス数、コレクショングループ内の OCUs 数のサービスクォータ (制限) があります。詳細については、OpenSearch Serverless クォータ」を参照してください。

主要なコンセプト

コレクショングループ

1 つ以上のコレクションのコンテナとして機能する AWS リソース。各コレクショングループには一意の識別子があり、独自の容量制限と構成設定を持つことができます。

共有コンピューティング

同じコレクショングループ内のコレクションは、使用する KMS キーに関係なく、同じ OCUs のセットを共有します。この共有により、KMS キーごとに個別のコンピューティングリソースが不要になり、コストを削減できます。

容量制限

コレクショングループレベルで、インデックス作成オペレーションと検索オペレーションの両方に最小 OCU 制限と最大 OCU 制限を設定できます。これらの制限は、コストを制御し、一貫したパフォーマンスを確保するのに役立ちます。

コレクショングループの容量制限

コレクショングループは、最小 OCU 制限と最大 OCU 制限を通じてリソース割り当てをきめ細かく制御します。これらの制限は、グループ内のすべてのコレクションに適用され、アカウントレベルの容量設定とは独立して動作します。

デフォルトでは、コレクショングループ内のコレクション数、コレクション内のインデックス数、コレクショングループ内の OCUs 数のサービスクォータ (制限) があります。詳細については、OpenSearch Serverless クォータ」を参照してください。

コレクショングループの容量制限の概要

コレクショングループレベルで、インデックス作成オペレーションと検索オペレーションの両方に最小 OCU 制限と最大 OCU 制限を設定できます。これらの制限は、OpenSearch Serverless がグループ内のコレクションのリソースをスケーリングする方法を制御します。

- 最小 OCU – OpenSearch Serverless がコレクショングループに対して維持する OCUs の最小数。一貫したベースラインパフォーマンスを確保します。
  - ワークロードに必要な OCUs指定された最小値よりも少ない場合でも、OpenSearch Serverless は指定された最小 OCUs、請求はその最小値を反映します。

- ワークロードが指定された最小値よりも多くの OCUs を必要とする場合、OpenSearch Serverless はワークロードに必要な OCUs の数を維持し、請求にはより高い OCU 使用率が反映されます。
- 最大 OCU – OpenSearch Serverless がコレクショングループに対してスケールアップできる OCUs の最大数。コストの制御に役立ちます。

コレクショングループの容量制限は、アカウントレベルの制限とは独立して動作します。アカウントレベルの最大 OCU 設定は、コレクショングループに関連付けられていないコレクションにのみ適用され、コレクショングループの最大 OCU 設定は、その特定のグループ内のコレクションに適用されます。

有効な容量制限値

コレクショングループの最小 OCU 制限と最大 OCU 制限を設定する場合、1、2、4、8、16、および 16 の倍数 (32、48、64、80、96 など) から最大 1,696 OCUs までの値のみを使用できます。

コレクショングループを作成する場合、最小 OCU 制限と最大 OCU 制限の両方はオプションです。最大 OCU 制限を指定しない場合、OpenSearch Serverless はデフォルト値の 96 OCUs を使用します。

最小 OCU 制限は、最大 OCU 制限以下である必要があります。

アカウントレベルとコレクショングループの OCU 制限

OpenSearch Serverless 容量を計画するときは、アカウントレベルの OCU 制限とコレクショングループの OCU 制限がどのように相互作用するかを理解します。すべてのコレクショングループの最大 OCU 設定とアカウントレベルでの最大 OCU 設定の合計は、アカウントあたりのサービスクォータ制限以下である必要があります。現在の制限値については、OpenSearch Serverless クォータ」を参照してください。

> **Note**
>
> アカウントレベルの最大 OCU 設定は、コレクショングループに関連付けられていないコレクションにのみ適用されます。コレクショングループ内のコレクションは、アカウントレベルの制限ではなく、それぞれのコレクショングループの制限によって管理されます。

この制約は、インデックス作成と検索の両方OCUs に個別に適用されます。例えば、アカウントレベルの設定とコレクショングループを設定する場合、合計が OCUsし、個別に検索 OCUs のサービス

クォータ制限を超えないようにする必要があります。さらに、アカウントごとに最大 300 個のコレクショングループを作成できます。

例: アカウントレベルとコレクショングループの制限を使用してキャパシティを計画する

アカウントレベルの最大検索 OCU を 500 に設定し、サービスクォータ制限を 1,700 に設定した場合:

- また、2 つのコレクショングループを作成します。2 つのコレクショングループの最大 OCU の合計は 1,200 (1,700 ~ 500) 以下にする必要があります。
- 各コレクショングループをデフォルトの最大 OCU 96 (96 + 96 + 500 = 692) のままにして、将来の成長に備えることができます。
- または、サービスクォータで許可されている全容量を使用して、各コレクショングループの最大値を 600 (600 + 600 + 500 = 1,700) に増やすこともできます。

この関係は、キャパシティプランニングに不可欠です。新しいコレクショングループを作成するか、最大 OCU 制限を引き上げる前に、合計割り当てがサービスクォータ制限を超えていないことを確認してください。この制限に達した場合は、既存のコレクショングループの最大 OCU 設定を減らすか、アカウントレベルの最大 OCU 設定を減らして、新しい割り当ての余地を確保する必要があります。

容量制限を設定する

コレクショングループを作成するとき、または後で更新するときに、容量制限を設定できます。を使用して容量制限を設定するには AWS CLI、CreateCollectionGroup コマンドまたは UpdateCollectionGroup コマンドを使用します。

```
aws opensearchserverless create-collection-group \
    --name my-collection-group \
    --capacity-limits
 maxIndexingCapacityInOCU=32,maxSearchCapacityInOCU=32,minIndexingCapacityInOCU=4,minSearchCapa
```

既存のコレクショングループの容量制限を更新するには:

```
aws opensearchserverless update-collection-group \
    --id abcdef123456 \
    --capacity-limits
 maxIndexingCapacityInOCU=48,maxSearchCapacityInOCU=48,minIndexingCapacityInOCU=8,minSearchCapa
```

コレクショングループの容量をモニタリングする

OpenSearch Serverless は、コレクショングループレベルで OCU 使用率と容量制限をモニタリングできるように、次の Amazon CloudWatch Logs メトリクスを 1 分間隔で出力します。

- `IndexingOCU` – コレクショングループが現在使用しているインデックス作成 OCUs の数。
- `SearchOCU` – コレクショングループが現在使用している検索 OCUs の数。

OpenSearch Serverless は、どのコレクショングループにも関連付けられていないコレクションの OCU メトリクスをアカウントレベルで出力します。これらのメトリクスを CloudWatch に集約して、すべてのコレクショングループとアカウントレベルのコレクションにわたる OCUs の合計を視覚化できます。

コレクショングループが容量制限に近づいたときに通知を受信するようにアラームを設定して、必要に応じて設定を調整できます。OpenSearch Serverless メトリクスの詳細については、「」を参照してくださいthe section called “OpenSearch Serverless のモニタリング”。

容量制限の適用

OpenSearch Serverless は、スケーリングオペレーション中にコレクショングループの容量制限を適用します。コレクションに追加のリソースが必要な場合、OpenSearch Serverless は最大 OCU 制限までスケールアップします。需要が低下すると、OpenSearch Serverless はスケールダウンしますが、一貫したパフォーマンスを確保するために少なくとも最小 OCU 制限を維持します。

OpenSearch Serverless は、コレクショングループに少なくとも 1 つのコレクションが含まれている場合にのみ容量制限を適用します。空のコレクショングループは OCUsしたり、容量制限を適用したりしません。

スケーリングオペレーションが最大 OCU 制限を超えるか、最小 OCU 要件に違反する場合、OpenSearch Serverless は設定された制限への準拠を維持するためにオペレーションを拒否します。

コレクショングループの暗号化キーと KMS キー

作成した各 OpenSearch Serverless コレクションは、 を使用して保管中のデータの暗号化で保護 AWS KMS され、暗号化キーを保存および管理します。コレクショングループを使用する場合、コレクションの KMS キーを指定する方法に柔軟性があります。

コレクションに関連付けられた KMS キーは、次の 2 つの方法で指定できます。

- CreateCollection リクエスト – `encryption-config`パラメータを使用してコレクションを作成するときに、KMS キーを直接指定します。
- セキュリティポリシー – 暗号化セキュリティポリシーで KMS キーの関連付けを定義します。

両方の場所で KMS キーを指定すると、CreateCollection リクエストで指定した KMS キーがセキュリティポリシー設定よりも優先されます。

この柔軟性により、特に一意の KMS キーを使用して複数のコレクションを作成する必要がある場合に、大規模なコレクションの管理方法が簡素化されます。数千の暗号化ポリシーを作成して管理する代わりに、コレクションの作成時に KMS キーを直接指定できます。

異なる KMS キー間で OCUs を共有する

コレクショングループを使用すると、異なる KMS キーを持つコレクション間でコンピューティングリソースを共有できます。同じコレクショングループ内のコレクションは、暗号化キーに関係なく OCU メモリ領域を共有します。この共有コンピューティングモデルは、KMS キーごとに個別の OCUs を必要とせずにコストを削減します。

コレクショングループは、セキュリティとパフォーマンスの要件を分離します。同じ KMS キーを持つコレクションを単一のコレクショングループにグループ化してセキュリティを分離したり、同じグループ内の異なる KMS キーとコレクションを組み合わせてコストを最適化したりできます。この柔軟性により、セキュリティ要件とリソース効率のバランスをとることができます。

OpenSearch Serverless は、指定された KMS キーを使用して各コレクションのデータを暗号化することで、セキュリティを維持します。アクセスコントロールは引き続きコレクションレベルで適用され、共有コンピューティングリソースは、グループ内のコレクションを提供するために必要に応じて複数の KMS キーにアクセスします。

必要な KMS アクセス許可

CreateCollection リクエストで KMS キーを指定する場合は、次の追加のアクセス許可が必要です。

- `kms:DescribeKey` – OpenSearch Serverless が KMS キーに関する情報を取得できるようにします。
- `kms:CreateGrant` – OpenSearch Serverless が KMS キーの許可を作成して暗号化オペレーションを有効にすることを許可します。

AWS 所有キーを使用する場合、これらのアクセス許可は必要ありません。

コレクショングループを作成する

このトピックでは、Amazon OpenSearch Serverless でコレクショングループを作成、設定、管理する方法について説明します。コレクショングループを使用してコレクションを整理し、コンピューティングリソースを共有してコストを最適化します。パフォーマンスと支出を制御するために、コレクショングループレベルで最小 OCU 制限と最大 OCU 制限を設定します。

コレクショングループを作成する

次の手順を使用して、新しいコレクショングループを作成し、その設定を構成します。OpenSearch Serverless コンソールまたは SDK を使用してコレクショングループを作成できます。 AWS CLI AWS SDKs コレクショングループを作成するときは、容量制限やその他の設定オプションを指定します。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを展開し、コレクショングループを選択します。
3. コレクショングループの作成 を選択します。NextGen コレクショングループ作成フォームがデフォルトで開きます。

   > **Tip**
   >
   > 代わりに Classic コレクショングループを作成するには、「Classic に切り替える」を選択します。Classic コレクショングループは、NextGen コレクショングループで使用できないデプロイタイプオプションをサポートしています。NextGen フォームに戻すには、NextGen に切り替えるを選択します。

4. コレクショングループ名に、コレクショングループの名前を入力します。名前は次の基準を満たしている必要があります。

   - 小文字 a～z、数字 0～9、ハイフン (-) のみが含まれます。
   - 3～32 文字にする必要があります
5. (オプション) 説明 に、コレクショングループの説明を入力します。
6. キャパシティ管理セクションで、OCU 制限を設定します。

- 最小インデックス作成容量 (OCUs 単位) – オプション。最小数がない場合は空白のままにします。
- 最大インデックス作成容量 (OCUs) — デフォルトは 96 です。
- 最小検索容量 (OCUs 単位) – オプション。最小数がない場合は空白のままにします。
- 最大検索容量 (OCUs) — デフォルトは 96 です。

7. (オプション) タグセクションで、コレクショングループの整理と識別に役立つタグを追加します。
8. コレクショングループの作成 を選択します。

AWS CLI

- create-collection-group コマンドを使用して、新しいコレクショングループを作成します。コマンドで、*####*コンテンツを独自の特定の情報に置き換えます。

```
aws opensearchserverless create-collection-group \
    --name my-collection-group \
    --description "Collection group for production workloads" \
    --capacity-limits
 maxIndexingCapacityInOCU=20,maxSearchCapacityInOCU=20,minIndexingCapacityInOCU=2,minSea
 \
    --tags key=Environment,value=Production key=Team,value=DataEngineering
```

コマンドは、一意の ID や ARN など、作成されたコレクショングループに関する詳細を返します。

コレクショングループにコレクションを追加する

コレクションを作成するときは、コレクショングループに割り当てます。

Console

コレクションをコレクショングループに追加するには

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。

2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. [Create collection] (コレクションの作成) を選択します。
4. コレクション名には、コレクションの名前を入力します。名前は 3～32 文字で、小文字で始まり、小文字、数字、ハイフンのみを使用する必要があります。
5. (オプション) 説明 に、コレクションの説明を入力します。
6. コレクションをコレクショングループに割り当てます。プロセスはコレクションタイプによって異なります。

   - NextGen – Express Create – OpenSearch Serverless はコレクショングループを自動的に割り当てます。初めて使用するユーザーの場合、デフォルトのコレクショングループはコレクション名に基づいて生成されます。戻りユーザーの場合、既存の互換性のあるコレクショングループがデフォルトで選択されます。割り当てを変更するには、ドロップダウンから別のグループを選択します。
   - NextGen – 標準作成 – ドロップダウンから既存の互換性のあるコレクショングループを選択するか、カスタム容量制限を持つ新しいコレクショングループを作成します。
   - Classic – コレクショングループセクションで、コレクションを割り当てるコレクショングループを選択します。コレクションは、一度に 1 つのコレクショングループにのみ属できます。(オプション) 新しいグループを作成するには、新しいグループの作成を選択します。これにより、コレクショングループの作成ワークフローが開きます。
7. ワークフローの残りのステップを完了してコレクションを作成します。

> **Important**
>
> コレクションの作成ワークフローから移動しないでください。

AWS CLI

- create-collection コマンドを使用してコレクションを作成し、既存のコレクショングループに追加します。コマンドで、*####*コンテンツを独自の情報に置き換えます。

```
aws opensearchserverless create-collection \
    --name my-collection \
    --type SEARCH \
    --collection-group-name my-collection-group \
    --description "Collection for search workloads"
```

Amazon OpenSearch Serverless コレクショングループの管理

Amazon OpenSearch Serverless コレクショングループを作成したら、ニーズの変化に応じて設定を変更できます。これらの管理オペレーションを使用して、容量制限を更新し、コレクショングループの詳細を表示します。これらの変更は、リソースの割り当てを最適化し、コレクションの効率的な整理を維持するのに役立ちます。

コレクショングループを表示する

OpenSearch Serverless コレクショングループを表示して、設定、関連するコレクション、現在のステータスを確認します。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクショングループタブを選択します。アカウントのコレクショングループが表示されます。
4. コレクショングループの名前を選択して詳細を表示します。

AWS CLI

- list-collection-groups コマンドを使用して、アカウント内のすべてのコレクショングループを一覧表示します。batchbatch-get-collection-group コマンドを使用して、特定のコレクショングループの詳細を表示します。次のコマンドでは、####コンテンツを独自の特定の情報に置き換えます。

  すべてのコレクショングループを一覧表示するには:

  ```
  aws opensearchserverless list-collection-groups
  ```

  特定のコレクショングループの詳細を取得するには:

```
aws opensearchserverless batch-get-collection-group \
    --names my-collection-group another-group
```

コレクショングループ設定の更新

OpenSearch Serverless コレクショングループ設定を更新して、容量制限や説明などの設定を変更します。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクショングループタブを選択します。アカウントのコレクショングループが表示されます。
4. コレクショングループの名前を選択して詳細を表示します。
5. コレクショングループの詳細で、編集 を選択します。
6. 変更を加え、保存を選択します。

AWS CLI

- update-collection-group コマンドを使用して、既存のコレクショングループの説明と容量制限を更新します。次のコマンドで、####コンテンツを独自の情報に置き換えます。

```
aws opensearchserverless update-collection-group \
    --id abcdef123456 \
    --description "Updated description for production workloads" \
    --capacity-limits
 maxIndexingCapacityInOCU=30,maxSearchCapacityInOCU=30,minIndexingCapacityInOCU=4,minSea
```

容量制限の変更はすぐに有効になり、グループ内のコレクションのスケーリング動作に影響する可能性があります。

コレクショングループを削除する

コレクショングループを削除する前に、まずグループからすべてのコレクションを削除する必要があります。コレクションを含むコレクショングループを削除することはできません。

Console

1. https://console.aws.amazon.com/aos/ で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで、サーバーレスを選択し、コレクションを選択します。
3. コレクショングループタブを選択します。アカウントのコレクショングループが表示されます。
4. 削除するコレクショングループの名前を選択します。

> **Important**
> コレクショングループからすべてのコレクションを削除するには、各コレクションを更新してコレクショングループの関連付けを削除するか、他のコレクショングループに移動します。

5. ページの上部で、[削除] を選択します。
6. 削除を確定し、削除を選択します。

AWS CLI

- deletedelete-collection-group コマンドを使用して、コレクショングループを削除します。

> **Important**
> コレクショングループからすべてのコレクションを削除するには、各コレクションを更新してコレクショングループの関連付けを削除するか、他のコレクショングループに移動します。

次のコマンドで、####コンテンツを独自の情報に置き換えます。

空のコレクショングループを削除します。

```
aws opensearchserverless delete-collection-group \
    --id abcdef123456
```

Amazon OpenSearch Serverless でゼロにスケールする

> **Note**
>
> ゼロへのスケールは、コレクショングループの一部である NextGen コレクションでのみ使用できます。詳細については、「the section called “コレクショングループ”」を参照してください。

ゼロにスケールすると、コレクショングループ内のすべてのコレクションがアイドル状態のときにコンピューティングリソースが自動的にシャットダウンされます。これにより、未使用の容量に対する料金がなくなります。グループ内のコレクションに受信リクエストが 10 分間ない場合、検索ワーカーとインデックス作成ワーカーはゼロ OCU にスケールダウンし、請求は停止します。グループ内のコレクションへのトラフィックが再開されると、ワーカーは自動的にプロビジョニングされ、リクエストパターンに基づいて自動スケーリングが再開されます。

これは、開発環境、バッチ処理ワークロード、予測可能なアイドル期間を持つアプリケーションに最適です。

動作をゼロにスケールする

以下に、コレクションでゼロへのスケールがどのように機能するかを示します。

- NextGen コレクショングループは、特に指定がない限り、インデックス作成と検索の両方でデフォルトで最小 OCU が 0 になります。
- グループ内のすべてのコレクションで 10 分間受信リクエストがない場合、コンピューティングリソースはゼロ OCU にスケールされます。このアイドル期間は設定できません。
- 検索とインデックス作成はゼロにスケールし、個別にウェイクします。各コンポーネントは、独自のトラフィックを受信するまでゼロのままです。
- トラフィックが再開すると、OpenSearch Serverless はゼロscale-to-zero。

  - 検索リクエスト — 2 つの検索ワーカー
  - リクエストのインデックス作成 — 1 つのインデックス作成ワーカー
- 容量が復元される間、各コンポーネントへの最初のリクエストで 10 ~ 30 秒のレイテンシーを想定します。

ゼロへのスケールの有効化

ゼロへのスケールを有効にするには、インデックス作成と検索の両方で最小 OCU が 0 のコレクショングループを作成し、そのグループ内にコレクションを作成します。

ゼロへのスケールの有効化

1. 最小 OCU がゼロのコレクショングループを作成します。

```
aws opensearchserverless create-collection-group \
    --name collection-group-name \
    --standby-replicas ENABLED \
    --generation NEXTGEN \
    --capacity-limits '{
        "maxIndexingCapacityInOCU": 8,
        "maxSearchCapacityInOCU": 8,
        "minIndexingCapacityInOCU": 0,
        "minSearchCapacityInOCU": 0
    }'
```

2. グループにコレクションを作成します。

```
aws opensearchserverless create-collection \
    --name collection-name \
    --type collection-type \
    --collection-group-name collection-group-name \
    --standby-replicas ENABLED
```

ゼロへのスケールアウト

コレクション容量をゼロにスケールしない場合は、最小容量がゼロ以外の値に設定されているコレクショングループの一部であることを確認してください。

Amazon OpenSearch Serverless でのキャパシティ制限の管理

Amazon OpenSearch Serverless では、キャパシティを自分で管理する必要はありません。OpenSearch Serverless は、現在のワークロードに基づいて、アカウントのコンピューティング性能を自動的にスケーリングします。Serverless のコンピューティング性能は、OpenSearch Compute Units (OCU) で測定されます。各 OCU は、6 GiB のメモリと対応する仮想 CPU (vCPU)、および Amazon S3 へのデータ転送を組み合わせたものです。OpenSearch Serverless の分離アーキテクチャの詳細については、「the section called “仕組み”」を参照してください。

最初のコレクションを作成すると、OpenSearch Serverless は冗長性の設定に基づいて OCU をインスタンス化します。デフォルトでは、冗長アクティブレプリカが有効になっています。これにより、インデックス作成には最低 1 OCU (0.5 OCU × 2)、検索には最低 1 OCU (0.5 OCU × 2) がインスタンス化されます。これにより、別のアベイラビリティーゾーンのスタンバイノードで高可用性が確保されます。

開発とテストでは、コレクションの冗長化を有効にする設定を無効にすることができます。これにより、スタンバイレプリカが削除され、2 つの OCUs のみが使用されます (1 つはインデックス作成用、もう 1 つは検索用) 。

これらの OCU は、インデックス作成や検索が行われていない場合でも常に存在します。後続のすべてのコレクションは、独自の OCUs セットをインスタンス化する一意の AWS KMS キーを持つコレクションを除き、これらの OCUs を共有できます。コレクショングループに関連付けられているすべてのコレクションは、同じ OCUs セットを共有できます。1 つのコレクショングループに含めることができるコレクションのタイプ (検索、時系列、またはベクトル検索) は 1 つだけです。詳細については、「the section called “コレクショングループ”」を参照してください。

OpenSearch Serverless は、インデックス作成と検索の使用が増えるにつれて OCUs を自動的にスケールアウトして追加します。トラフィックが減少すると、容量はデータサイズに必要な OCUs の最小数までスケールダウンされます。

検索および時系列コレクションの場合、アイドル状態のときに必要な OCUs の数は、データサイズとインデックス数に比例します。ベクトルコレクションの場合、OCU 要件はベクトルグラフを保存するメモリ (RAM) とインデックスを保存するディスク容量によって異なります。アイドル状態でない場合、OCU 要件は両方の要因を考慮します。

ベクトルコレクションは、インデックスデータを OCU ローカルストレージに保存します。OCU RAM 制限はディスク制限よりも速く到達するため、ベクトル収集は RAM スペースによって制限されます。

冗長性を有効にすると、OCU 容量はインデックス作成用に 1 OCU (0.5 OCU x 2)、検索用に 1 OCU (0.5 OCU x 2) にスケールダウンされます。冗長性を無効にすると、コレクションはインデックス作成用に 0.5 OCU、検索用に 0.5 OCU にスケールダウンできます。

スケーリングでは、コレクションまたはインデックスに必要なシャードの数も考慮されます。各 OCU は指定された数のシャードをサポートし、インデックスの数はシャード数に比例する必要があります。必要な基本 OCUs の合計数は、データ、メモリ、シャード要件の最大数です。詳細については、AWS Big Data Blog の「Amazon OpenSearch Serverless によるあらゆる規模における費用対効果の高い検索機能」を参照してください。

検索およびベクトル検索コレクションでは、迅速なクエリ応答時間を確保するために、すべてのデータがホットインデックスに保存されます。時系列コレクションでは、ホットストレージとウォームストレージを組み合わせて使用し、より頻繁にアクセスされるデータのクエリ応答時間を最適化するために、最新のデータがホットストレージに保存されます。詳細については、「the section called “コレクションタイプを選択する”」を参照してください。

> **Note**
>
> ベクトル検索コレクションが検索コレクションまたは時系列コレクションと同じ KMS キーを使用している場合でも、ベクトル検索コレクションは OCU を検索コレクションおよび時系列コレクションと共有することはできません。最初のベクトルコレクション用に新しい OCU セットが作成されます。ベクトルコレクションの OCU は、同じ KMS キーコレクション間で共有されます。

コレクションのキャパシティを管理してコストを制御するため、現在のアカウントおよびリージョン用にインデックス作成と検索の全体的な最大キャパシティを指定できます。OpenSearch Serverless は、これらの仕様に基づいて、コレクションリソースを自動的にスケールアウトします。

インデックス作成と検索のキャパシティは個別にスケーリングされるため、それぞれにアカウントレベルの制限を指定します。

- インデックス作成の最大キャパシティ — OpenSearch Serverless は、この数の OCU までインデックス作成のキャパシティを増やすことができます。
- 検索の最大キャパシティ — OpenSearch Serverless は、この数の OCU まで検索のキャパシティを増やすことができます。

> **Note**
>
> 現時点では、キャパシティの設定はアカウントレベルでのみ適用されます。コレクションごとにキャパシティの制限を設定することはできません。

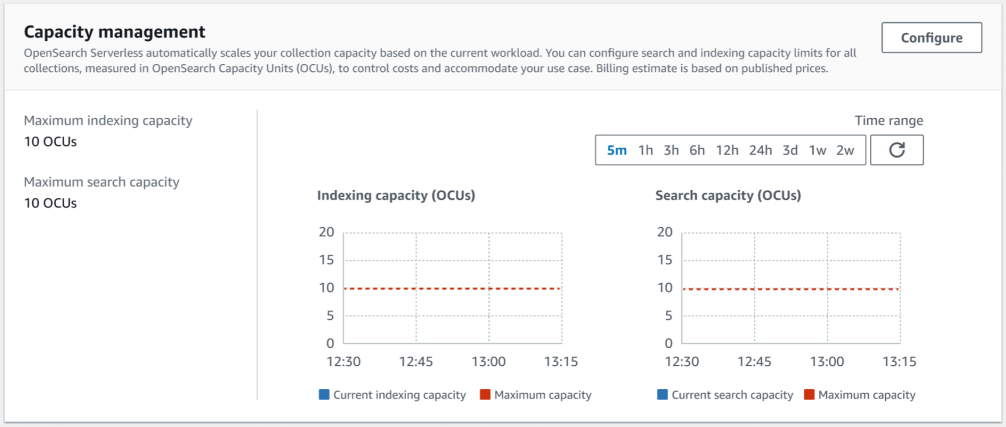
目標は、最大キャパシティがワークロードの急増を処理するために十分な量であるのを確実にすることです。OpenSearch Serverless は、ユーザーの設定に基づいてコレクションの OCU 数を自動的にスケールアウトして、インデックス作成と検索のワークロードを処理します。

トピック

- キャパシティの設定
- 最大キャパシティの制限
- キャパシティ使用量のモニタリング

キャパシティの設定

OpenSearch Serverless コンソールでキャパシティの設定を行うには、左側のナビゲーションペインで [Serverless] を展開し、[Dashboard] (ダッシュボード) を選択します。[Capacity management] (キャパシティ管理) で、インデックス作成と検索の最大キャパシティを指定します。



を使用して容量を設定するには AWS CLI、UpdateAccountSettings リクエストを送信します。

```
aws opensearchserverless update-account-settings \
    --capacity-limits '{ "maxIndexingCapacityInOCU": 8,"maxSearchCapacityInOCU": 9 }'
```

最大キャパシティの制限

コレクションに含めることができるインデックスの最大合計は 1000 です。3 種類のコレクションすべてにおいて、デフォルトの最大キャパシティは、インデックス作成用に 10 OCU、検索用に 10 OCU です。アカウントで利用可能な最小キャパシティは、インデックス作成用に 1 OCU [0.5 OCU x 2]、検索用に 1 OCU [0.5 OCU x 2] です。すべてのコレクションで許可される最大容量は、インデックス作成の場合は 1,700 OCUs検索の場合は 1,700 OCUs。OCU の数は、2 から、最大許容キャパシティ (2 の倍数) までの任意の数に設定できます。

各 OCU には、120 GiB のインデックスデータを保存するのに十分なホットエフェメラルストレージが含まれています。OpenSearch Serverless は、検索およびベクトル検索コレクションのインデックスあたり最大 1 TiB のデータと、時系列コレクションのインデックスあたり最大 100 TiB のホットデータをサポートします。時系列コレクションの場合は、さらに多くのデータを取り込むことができ、S3 にウォームデータとして保存できます。

すべてのクォータのリストについては、「OpenSearch Serverless のクォータ」を参照してください。

キャパシティ使用量のモニタリング

`Search0CU` および `Indexing0CU` におけるアカウントレベルの CloudWatch メトリクスをモニタリングして、コレクションでどのようにスケーリングされているかを把握できます。アカウントがキャパシティに関するメトリクスのしきい値に近づいた際に通知を行うアラームを設定することをお勧めします。そうすることで、状況に応じてキャパシティの設定を調整できます。

最大キャパシティの設定が適切か、または調整が必要どうかを判断するために、これらのメトリクスを使用することもできます。これらのメトリクスを分析することで、コレクションの効率を最適化することに集中できます。OpenSearch Serverless が CloudWatch に送信するメトリクスの詳細については、「the section called “OpenSearch Serverless のモニタリング”」を参照してください。

Amazon OpenSearch Serverless コレクションへのデータの取り込み

これらのセクションでは、Amazon OpenSearch Serverless コレクションへのデータインジェストでサポートされている取り込みパイプラインについて詳しく説明します。OpenSearch API オペレーションの操作に使用できるクライアントの一部についても説明しています。OpenSearch Serverless と統合するには、クライアントが OpenSearch 2.x と互換性がある必要があります。

トピック

- 必要な最小限のアクセス許可
- OpenSearch Ingestion
- Fluent Bit
- Amazon Data Firehose
- Go
- Java
- JavaScript
- Logstash
- Python
- Ruby
- 他のクライアントを使用した HTTP リクエストの署名

必要な最小限のアクセス許可

OpenSearch Serverless コレクションにデータを取り込むには、データを書き込むプリンシパルに、データアクセスポリシーで次の最小限のアクセス許可が割り当てられている必要があります。

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/logs"
            ],
            "Permission":[
               "aoss:CreateIndex",
               "aoss:WriteDocument",
               "aoss:UpdateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:iam::123456789012:user/my-user"
      ]
   }
```

```
]
```

追加のインデックスに書き込む場合は、アクセス許可をより幅広く指定できます。例えば、1 つのターゲットインデックスを指定するのではなく、すべてのインデックス (index/*target-collection*/*) または一部のインデックス (index/*target-collection*/*logs**) へのアクセス許可を付与することができます。

使用可能なすべての OpenSearch API オペレーションとそれに関連するアクセス許可のリファレンスについては、「the section called “サポートされているオペレーションとプラグイン”」を参照してください。

OpenSearch Ingestion

OpenSearch サーバーレスコレクションにデータを直接送信する場合、サードパーティのクライアントを使用する代わりに Amazon OpenSearch Ingestion を使用できます。OpenSearch Ingestion にデータを送信するようにデータプロデューサーを設定すると、指定したコレクションにデータが自動的に送信されます。また、OpenSearch Ingestion で、送信前にデータを変換するように設定することもできます。詳細については、「Amazon OpenSearch Ingestion」を参照してください。

シンクとして設定されている OpenSearch Serverless コレクションに書き込みを行うときは、OpenSearch Ingestion パイプラインにアクセス権限が必要になります。これらのアクセス権限には、コレクションを記述しそこに HTTP リクエストを送信できることが含まれます。OpenSearch Ingestion を使用してコレクションにデータを追加する手順については、「the section called “パイプラインにコレクションへのアクセス権を付与する”」を参照してください。

OpenSearch Ingestion の使用を開始するには、「the section called “チュートリアル: コレクションにデータを取り込む”」を参照してください。

Fluent Bit

OpenSearch Serverless コレクションにデータを取り込むために、AWS for Fluent Bit イメージと OpenSearch 出力プラグインを使用できます。

> **Note**
>
> OpenSearch Serverless と統合するには、 AWS for Fluent Bit イメージのバージョン 2.30.0 以降が必要です。

設定例:

設定ファイルのこのサンプル出力セクションは、OpenSearch Serverless コレクションを宛先として使用する方法を示しています。重要な追加項目は AWS_Service_Name パラメータで、これは aoss です。Host はコレクションエンドポイントです。

```
[OUTPUT]
    Name  opensearch
    Match *
    Host  collection-endpoint.us-west-2.aoss.amazonaws.com
    Port  443
    Index  my_index
    Trace_Error On
    Trace_Output On
    AWS_Auth On
    AWS_Region <region>
    AWS_Service_Name aoss
    tls     On
    Suppress_Type_Name On
```

Amazon Data Firehose

Firehose は配信先として OpenSearch Serverless をサポートしています。OpenSearch Serverless にデータを送信する方法については、「Amazon Data Firehose Developer Guide」の「Creating a Kinesis Data Firehose Delivery Stream」および「Choose OpenSearch Serverless for Your Destination」を参照してください。

配信のために Firehose に提供する IAM ロールは、ターゲットコレクションに対する最小限の aoss:WriteDocument 許可を持つデータアクセスポリシー内で指定する必要があります。また、データを送信する既存のインデックスが必要です。詳細については、「the section called “必要な最小限のアクセス許可”」を参照してください。

データを OpenSearch Serverless に送信する前に、データの変換が必要になる場合があります。Lambda 関数を使用してこのタスクを実行する方法については、このガイドの「Amazon Kinesis Data Firehose データ変換」を参照してください。

Go

以下のサンプルコードでは、Go 用の opensearch-go クライアントを使用して、指定された OpenSearch Serverless コレクションへのセキュアな接続を確立し、単一のインデックスを作成します。region および host の値を指定する必要があります。

```
package main
```

```
import (
  "context"
  "log"
  "strings"
  "github.com/aws/aws-sdk-go-v2/aws"
  "github.com/aws/aws-sdk-go-v2/config"
  opensearch "github.com/opensearch-project/opensearch-go/v2"
  opensearchapi "github.com/opensearch-project/opensearch-go/v2/opensearchapi"
  requestsigner "github.com/opensearch-project/opensearch-go/v2/signer/awsv2"
)

const endpoint = "" // serverless collection endpoint

func main() {
 ctx := context.Background()

 awsCfg, err := config.LoadDefaultConfig(ctx,
  config.WithRegion("<AWS_REGION>"),
  config.WithCredentialsProvider(
   getCredentialProvider("<AWS_ACCESS_KEY>", "<AWS_SECRET_ACCESS_KEY>",
 "<AWS_SESSION_TOKEN>"),
  ),
 )
 if err != nil {
  log.Fatal(err) // don't log.fatal in a production-ready app
 }

 // create an AWS request Signer and load AWS configuration using default config folder
 or env vars.
 signer, err := requestsigner.NewSignerWithService(awsCfg, "aoss") // "aoss" for Amazon
 OpenSearch Serverless
 if err != nil {
  log.Fatal(err) // don't log.fatal in a production-ready app
 }

 // create an opensearch client and use the request-signer
 client, err := opensearch.NewClient(opensearch.Config{
  Addresses: []string{endpoint},
  Signer:    signer,
 })
 if err != nil {
  log.Fatal("client creation err", err)
 }
```

```
 indexName := "go-test-index"

  // define index mapping
 mapping := strings.NewReader(`{
  "settings": {
    "index": {
         "number_of_shards": 4
         }
       }
  }`)

 // create an index
 createIndex := opensearchapi.IndicesCreateRequest{
  Index: indexName,
    Body: mapping,
 }
 createIndexResponse, err := createIndex.Do(context.Background(), client)
 if err != nil {
  log.Println("Error ", err.Error())
  log.Println("failed to create index ", err)
  log.Fatal("create response body read err", err)
 }
 log.Println(createIndexResponse)

 // delete the index
 deleteIndex := opensearchapi.IndicesDeleteRequest{
  Index: []string{indexName},
 }

 deleteIndexResponse, err := deleteIndex.Do(context.Background(), client)
 if err != nil {
  log.Println("failed to delete index ", err)
  log.Fatal("delete index response body read err", err)
 }
 log.Println("deleting index", deleteIndexResponse)
}

func getCredentialProvider(accessKey, secretAccessKey, token string)
 aws.CredentialsProviderFunc {
 return func(ctx context.Context) (aws.Credentials, error) {
  c := &aws.Credentials{
   AccessKeyID:     accessKey,
   SecretAccessKey: secretAccessKey,
```

```
    SessionToken:    token,
   }
   return *c, nil
  }
}
```

Java

以下のサンプルコードでは、Java 用の opensearch-java クライアントを使用して、指定された OpenSearch Serverless コレクションへのセキュアな接続を確立し、単一のインデックスを作成します。region および host の値を指定する必要があります。

OpenSearch Service ドメインとの重要な違いは、サービス名です (es ではなく aoss)。

```
// import OpenSearchClient to establish connection to OpenSearch Serverless collection
import org.opensearch.client.opensearch.OpenSearchClient;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

// Configure credential provider
AwsCredentialsProvider credentialsProvider = DefaultCredentialsProvider.create();

SdkHttpClient httpClient = ApacheHttpClient.builder().build();
// create an opensearch client and use the request-signer
OpenSearchClient client = new OpenSearchClient(
    new AwsSdk2Transport(
        httpClient,
        "...us-west-2.aoss.amazonaws.com", // serverless collection endpoint
        "aoss" // signing service name
        Region.US_WEST_2, // signing service region
        AwsSdk2TransportOptions.builder().build()
    )
);

String index = "sample-index";

// create an index
CreateIndexRequest createIndexRequest = new
 CreateIndexRequest.Builder().index(index).build();
CreateIndexResponse createIndexResponse = client.indices().create(createIndexRequest);
System.out.println("Create index reponse: " + createIndexResponse);

// delete the index
```

```
DeleteIndexRequest deleteIndexRequest = new
 DeleteIndexRequest.Builder().index(index).build();
DeleteIndexResponse deleteIndexResponse = client.indices().delete(deleteIndexRequest);
System.out.println("Delete index reponse: " + deleteIndexResponse);

httpClient.close();
```

次のサンプルコードは、安全な接続を再度確立し、インデックスを検索します。

```
import org.opensearch.client.opensearch.OpenSearchClient;

SdkHttpClient httpClient = ApacheHttpClient.builder().build();

OpenSearchClient client = new OpenSearchClient(
    new AwsSdk2Transport(
        httpClient,
        "...us-west-2.aoss.amazonaws.com", // serverless collection endpoint
        "aoss" // signing service name
        Region.US_WEST_2, // signing service region
        AwsSdk2TransportOptions.builder().build()
    )
);

Response response = client.generic()
    .execute(
        Requests.builder()
            .endpoint("/" + "users" + "/_search?typed_keys=true")
            .method("GET")
            .json("{"
                + "    \"query\": {"
                + "        \"match_all\": {}"
                + "    }"
                + "}")
            .build());

httpClient.close();
```

次のサンプルコードは、安全な接続を確立し、ドキュメントをコレクションにインデックス化します。

```
import org.opensearch.client.opensearch.OpenSearchClient;
import org.opensearch.client.opensearch.core.IndexRequest;
import org.opensearch.client.opensearch.core.IndexResponse;
```

```
import java.util.HashMap;
import java.util.Map;

SdkHttpClient httpClient = ApacheHttpClient.builder().build();

OpenSearchClient client = new OpenSearchClient(
    new AwsSdk2Transport(
        httpClient,
        "...us-west-2.aoss.amazonaws.com", // serverless collection endpoint
        "aoss" // signing service name
        Region.US_WEST_2, // signing service region
        AwsSdk2TransportOptions.builder().build()
    )
);

// index a document
Map<String, Object> document = new HashMap<>();
document.put("title", "The Green Mile");
document.put("director", "Frank Darabont");
document.put("year", "1999");

IndexRequest<Map<String, Object>> indexRequest = IndexRequest.of(i -> i
    .index("books-index")
    .document(document)
);

IndexResponse indexResponse = client.index(indexRequest);
System.out.println("Index response: " + indexResponse.result());

httpClient.close();
```

JavaScript

以下のサンプルコードでは、JavaScript 用の opensearch-js クライアントを使用して、指定された OpenSearch Serverless コレクションへのセキュアな接続の確立、単一のインデックスの作成、ドキュメントの追加、およびインデックスの削除を実行します。node および region の値を指定する必要があります。

OpenSearch Service ドメインとの重要な違いは、サービス名です (es ではなく aoss)。

Version 3

この例では SDK for JavaScript in Node.js の バージョン 3 を使用します。

```
const { defaultProvider } = require('@aws-sdk/credential-provider-node');
const { Client } = require('@opensearch-project/opensearch');
const { AwsSigv4Signer } = require('@opensearch-project/opensearch/aws');

async function main() {
    // create an opensearch client and use the request-signer
    const client = new Client({
        ...AwsSigv4Signer({
            region: 'us-west-2',
            service: 'aoss',
            getCredentials: () => {
                const credentialsProvider = defaultProvider();
                return credentialsProvider();
            },
        }),
        node: '' # // serverless collection endpoint
    });

    const index = 'movies';

    // create index if it doesn't already exist
    if (!(await client.indices.exists({ index })).body) {
        console.log((await client.indices.create({ index })).body);
    }

    // add a document to the index
    const document = { foo: 'bar' };
    const response = await client.index({
        id: '1',
        index: index,
        body: document,
    });
    console.log(response.body);

    // delete the index
    console.log((await client.indices.delete({ index })).body);
}

main();
```

Version 2

この例では SDK for JavaScript in Node.js の バージョン 2 を使用します。

```
const AWS = require('aws-sdk');
const { Client } = require('@opensearch-project/opensearch');
const { AwsSigv4Signer } = require('@opensearch-project/opensearch/aws');

async function main() {
    // create an opensearch client and use the request-signer
    const client = new Client({
        ...AwsSigv4Signer({
            region: 'us-west-2',
            service: 'aoss',
            getCredentials: () =>
                new Promise((resolve, reject) => {
                    AWS.config.getCredentials((err, credentials) => {
                        if (err) {
                            reject(err);
                        } else {
                            resolve(credentials);
                        }
                    });
                }),
        }),
        node: '' # // serverless collection endpoint
    });

    const index = 'movies';

    // create index if it doesn't already exist
    if (!(await client.indices.exists({ index }))).body) {
        console.log((await client.indices.create({
            index
        })).body);
    }

    // add a document to the index
    const document = {
        foo: 'bar'
    };
    const response = await client.index({
        id: '1',
        index: index,
        body: document,
    });
    console.log(response.body);
```

```
    // delete the index
    console.log((await client.indices.delete({ index })).body);
}

main();
```

Logstash

OpenSearch Serverless コレクションにログを公開するときは、Logstash OpenSearch プラグインを使用します。

Logstash を使用して OpenSearch Serverless にデータを送信するには

1. Docker または Linux を使用して logstash-output-opensearch プラグインのバージョン 2.0.0 以降をインストールします。

   Docker

   Docker は、OpenSearch 出力プラグイン (opensearchproject/logstash-oss-with-opensearch-output-plugin) がプリインストールされた Logstash OSS ソフトウェアをホストしています。他のイメージと同じように次のようにイメージをプルできます。

   ```
   docker pull opensearchproject/logstash-oss-with-opensearch-output-plugin:latest
   ```

   Linux

   まだ Logstash の最新バージョンをインストールしていない場合は、インストールします。次に、出力プラグインのバージョン 2.0.0 を次のようにインストールします。

   ```
   cd logstash-8.5.0/
   bin/logstash-plugin install --version 2.0.0 logstash-output-opensearch
   ```

   プラグインが既にインストールされている場合は、次のように最新バージョンに更新します。

   ```
   bin/logstash-plugin update logstash-output-opensearch
   ```

プラグインのバージョン 2.0.0 以降、 AWS SDK はバージョン 3 を使用します。8.4.0 より前のバージョンの Logstash を使用している場合は、プリインストールされている AWS プラグインをすべて削除し、`logstash-integration-aws`プラグインをインストールする必要があります。

```
/usr/share/logstash/bin/logstash-plugin remove logstash-input-s3
/usr/share/logstash/bin/logstash-plugin remove logstash-input-sqs
/usr/share/logstash/bin/logstash-plugin remove logstash-output-s3
/usr/share/logstash/bin/logstash-plugin remove logstash-output-sns
/usr/share/logstash/bin/logstash-plugin remove logstash-output-sqs
/usr/share/logstash/bin/logstash-plugin remove logstash-output-cloudwatch

/usr/share/logstash/bin/logstash-plugin install --version 0.1.0.pre logstash-
integration-aws
```

2. OpenSearch 出力プラグインが OpenSearch Serverless で動作するためには、logstash.conf の `opensearch` 出力セクションに次の変更を加える必要があります。

   - `auth_type` の `service_name` として `aoss` を指定します。
   - `hosts` のコレクションエンドポイントを指定します。
   - パラメータ `default_server_major_version` および `legacy_template` を追加します。これらのパラメータは、プラグインが OpenSearch Serverless で動作するために必要です。

```
output {
  opensearch {
    hosts => "collection-endpoint:443"
    auth_type => {
      ...
      service_name => 'aoss'
    }
    default_server_major_version => 2
    legacy_template => false
  }
}
```

次の設定ファイルの例では、S3 バケット内のファイルから入力を受け取り、OpenSearch Serverless コレクションに送信します。

```
input {
```

```
  s3  {
    bucket => "my-s3-bucket"
    region => "us-east-1"
  }
}

output {
  opensearch {
    ecs_compatibility => disabled
    hosts => "https://my-collection-endpoint.us-east-1.aoss.amazonaws.com:443"
    index => my-index
    auth_type => {
      type => 'aws_iam'
      aws_access_key_id => 'your-access-key'
      aws_secret_access_key => 'your-secret-key'
      region => 'us-east-1'
      service_name => 'aoss'
    }
    default_server_major_version => 2
    legacy_template => false
  }
}
```

3. 次に、次のように新しい設定で Logstash を実行してプラグインをテストします。

```
bin/logstash -f config/test-plugin.conf
```

Python

以下のサンプルコードでは、Python 用の opensearch-py クライアントを使用して、指定された OpenSearch Serverless コレクションへのセキュアな接続の確立、単一のインデックスの作成、そのインデックスの検索を実行します。`region` および `host` の値を指定する必要があります。

OpenSearch Service ドメインとの重要な違いは、サービス名です (`es` ではなく `aoss`)。

```
from opensearchpy import OpenSearch, RequestsHttpConnection, AWSV4SignerAuth
import boto3

host = ''  # serverless collection endpoint, without https://
region = ''  # e.g. us-east-1

service = 'aoss'
```

```
credentials = boto3.Session().get_credentials()
auth = AWSV4SignerAuth(credentials, region, service)

# create an opensearch client and use the request-signer
client = OpenSearch(
    hosts=[{'host': host, 'port': 443}],
    http_auth=auth,
    use_ssl=True,
    verify_certs=True,
    connection_class=RequestsHttpConnection,
    pool_maxsize=20,
)

# create an index
index_name = 'books-index'
create_response = client.indices.create(
    index_name
)

print('\nCreating index:')
print(create_response)

# index a document
document = {
  'title': 'The Green Mile',
  'director': 'Stephen King',
  'year': '1996'
}

response = client.index(
    index = 'books-index',
    body = document,
    id = '1'
)

delete_response = client.indices.delete(
    index_name
)

print('\nDeleting index:')
print(delete_response)
```

> **Note**
>
> この例の `id = '1'`パラメータは、カスタムドキュメント ID を指定します。カスタムドキュメント IDsは、検索コレクションでのみサポートされています。時系列およびベクトル検索コレクションの場合、カスタムドキュメント ID を使用したインデックス作成はサポートされず、エラーが返されます。時系列またはベクトル検索コレクションにインデックスを作成するときは、 `id`パラメータを省略します。

Ruby

`opensearch-aws-sigv4` gem は、追加の設定なしで OpenSearch Serverless とOpenSearch Service へのアクセスを提供します。これには、opensearch-ruby クライアントのすべての機能が備わっています。クライアントがこの gem の依存関係であるためです。

Sigv4 署名者をインスタンス化するときは、`aoss` をサービス名として指定します。

```
require 'opensearch-aws-sigv4'
require 'aws-sigv4'

signer = Aws::Sigv4::Signer.new(service: 'aoss',
                                region: 'us-west-2',
                                access_key_id: 'key_id',
                                secret_access_key: 'secret')

# create an opensearch client and use the request-signer
client = OpenSearch::Aws::Sigv4Client.new(
  { host: 'https://your.amz-opensearch-serverless.endpoint',
    log: true },
  signer)

# create an index
index = 'prime'
client.indices.create(index: index)

# insert data
client.index(index: index, id: '1', body: { name: 'Amazon Echo',
                                            msrp: '5999',
                                            year: 2011 })

# query the index
```

```
client.search(body: { query: { match: { name: 'Echo' } } })

# delete index entry
client.delete(index: index, id: '1')

# delete the index
client.indices.delete(index: index)
```

他のクライアントを使用した HTTP リクエストの署名

以下の要件は、別のクライアントを使用して HTTP リクエストを構築する場合に、OpenSearch Serverless コレクションへのリクエストに署名するときに適用されます。

- サービス名を aoss として指定する必要があります。
- x-amz-content-sha256 ヘッダーは、すべての AWS 署名バージョン 4 リクエストに必要です。これにより、リクエストペイロードのハッシュが指定されます。リクエストペイロードがある場合は、その値をセキュアハッシュアルゴリズム (SHA) 暗号化ハッシュ (SHA256) に設定します。リクエストペイロードがない場合は、値を e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 に設定します。これは空の文字列のハッシュです。

トピック

- cURL でのインデックス作成
- Postman でのインデックス作成

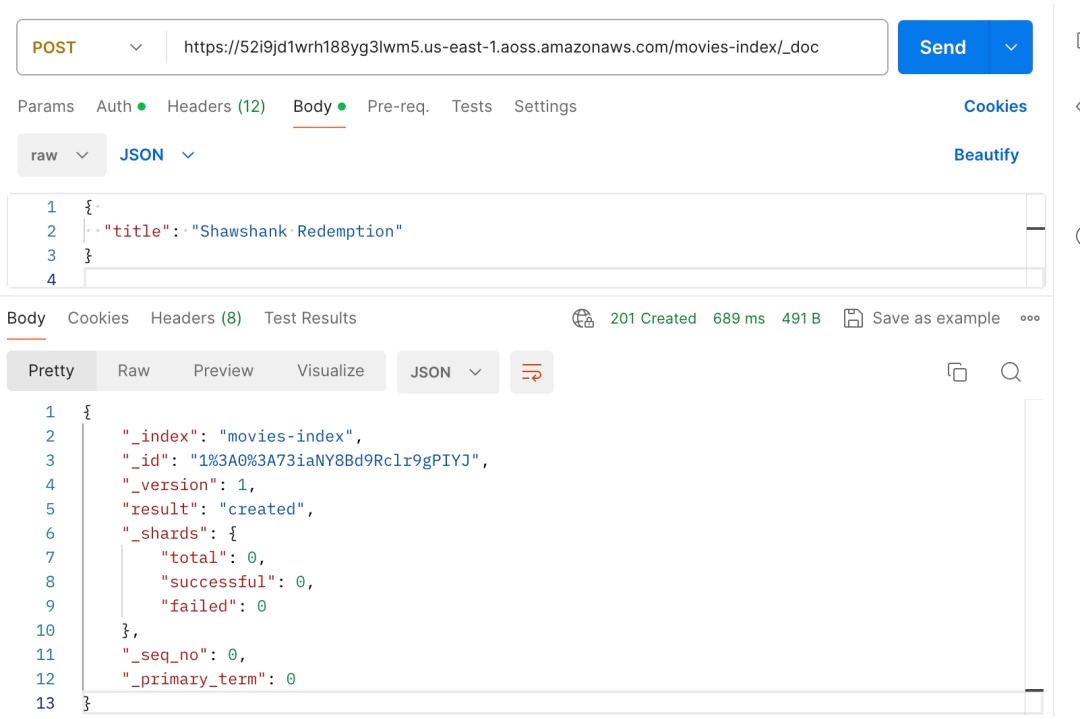
cURL でのインデックス作成

次のリクエスト例では、クライアント URL リクエストライブラリ (cURL) を使用して、コレクション内の movies-index という名前のインデックスに 1 つのドキュメントを送信します。

```
curl -XPOST \
    --user "$AWS_ACCESS_KEY_ID":"$AWS_SECRET_ACCESS_KEY" \
    --aws-sigv4 "aws:amz:us-east-1:aoss" \
    --header "x-amz-content-sha256: $REQUEST_PAYLOAD_SHA_HASH" \
    --header "x-amz-security-token: $AWS_SESSION_TOKEN" \
    "https://my-collection-endpoint.us-east-1.aoss.amazonaws.com/movies-index/_doc" \
    -H "Content-Type: application/json" -d '{"title": "Shawshank Redemption"}'
```

Postman でのインデックス作成

次の図は、Postman を使用してコレクションにリクエストを送信する方法を示しています。認証の手順については、Postman の AWS 「署名付き認証ワークフローによる認証」を参照してください。



Amazon OpenSearch Serverless で機械学習を設定する

機械学習

機械学習 (ML) は、ML アルゴリズムとリモートモデルの形式で ML 機能を提供します。これらのモデルにアクセスすることで、RAG やセマンティック検索など、いくつかの AI ワークフローを実行できます。ML は、コネクタで設定できる最新の外部ホストモデルを使用して、生成 AI ユースケースの実験と本番デプロイをサポートします。コネクタを設定した後は、それをモデルに設定してから、デプロイして予測を実行する必要があります。

Connector

コネクタは、サードパーティーの ML プラットフォームでホストされているモデルへのアクセスを容易にします。OpenSearch クラスターとリモートモデル間のゲートウェイとして機能します。詳細については、次のドキュメントを参照してください。

- 「OpenSearch ドキュメント」ウェブサイトの「Creating connectors for third-party ML platforms」
- 外部プラットフォーム用のコネクタ
- のコネクタ AWS のサービス

> **Important**
>
> - 信頼ポリシーを作成するときは、OpenSearch Service の原則として **`ml.opensearchservice.amazonaws.com`** を追加します。
> - ポリシーでドメインを設定する方法を表示する「コネクタ」ページのステップをスキップします。
> - アクセス許可の設定ステップに `iam:PassRole` ステートメントを追加します。
> - OpenSearch Dashboards の [ML ロールのマッピング] のステップをスキップします。バックエンドロールの設定は必要ありません。これは、 Connectors for AWS のサービス および Connectors for external platform に適用されます。
> - コレクションエンドポイントへの SigV4 リクエストで、サービス名を **`es`** ではなく **`aoss`** に設定します。

モデル

モデルは、さまざまな AI ワークフローで使用される主要な機能です。通常、コネクタを使用して予測を実行するために、コネクタをモデルに関連付けます。モデルがデプロイされた状態になったら、予測を実行できます。詳細については、「OpenSearch ドキュメント」ウェブサイトの「Register a model hosted on a third-party platform」を参照してください。

> **Note**
>
> ローカルモデルなど、すべてのモデル機能が OpenSearch Serverless でサポートされているわけではありません。詳細については、「サポートされていない機械学習 API と機能」を参照してください。

機械学習のアクセス許可を設定する

次のセクションでは、機械学習 (ML) に必要な収集データアクセスポリシーについて説明します。########は、ユーザー固有の情報に置き換えてください。詳細については、「サポートされているポリシーのアクセス許可」を参照してください。

```
{
    "Rules": [
        {
            "Resource": [
                "model/collection_name/*"
            ],
            "Permission": [
                "aoss:DescribeMLResource",
                "aoss:CreateMLResource",
                "aoss:UpdateMLResource",
                "aoss:DeleteMLResource",
                "aoss:ExecuteMLResource"
            ],
            "ResourceType": "model"
        }
    ],
    "Principal": [
        "arn:aws:iam::account_id:role/role_name"
    ],
    "Description": "ML full access policy for collection_name"
}
```

- aoss:DescribeMLResource – コネクタ、モデル、モデルグループを検索およびクエリするアクセス許可を付与します。
- aoss:CreateMLResource – コネクタ、モデル、モデルグループを作成するアクセス許可を付与します。
- aoss:UpdateMLResource – コネクタ、モデル、モデルグループを更新するアクセス許可を付与します。
- aoss:DeleteMLResource – コネクタ、モデル、モデルグループを削除するアクセス許可を付与します。
- aoss:ExecuteMLResource – モデルに対して予測を実行するアクセス許可を付与します。

サポートされていない機械学習 API と機能

サポートされていない API

Amazon OpenSearch Serverless では、次の機械学習 (ML) API はサポートされていません。

- ローカルモデル機能
- モデルトレーニング API
- モデル予測アルゴリズム API
- モデルバッチ予測 API
- エージェント API
- MCP サーバー API
- メモリ API
- コントローラー API
- アルゴリズム API を実行する
- ML プロファイル AP
- ML 統計 API

ML API の詳細については、「OpenSearch ドキュメント」ウェブサイトの「ML APIs」を参照してください。

サポートされていない機能

以下の ML 機能は、Amazon OpenSearch Serverless ではサポートされていません。

- エージェントとツール
- ローカルモデル
- 検索パイプラインと取り込みパイプライン内の ML 推論プロセッサ
  - ML 推論取り込みプロセッサ
  - ML 推論検索レスポンスプロセッサ
  - ML 推論検索リクエストプロセッサ

これらの機能の詳細については、「OpenSearch ドキュメント」ウェブサイトの以下のドキュメントを参照してください。

- 機械学習
- ML 推論プロセッサ
- パイプラインの検索

OpenSearch Serverless でニューラル検索とハイブリッド検索を設定する

Neural Search

Amazon OpenSearch Serverless は、データに対するセマンティック検索オペレーション向けにニューラル検索機能をサポートしています。ニューラル検索は、機械学習モデルを使用してクエリのセマンティックの意味とコンテキストを理解し、従来のキーワードベースの検索よりも関連性の高い検索結果を提供します。このセクションでは、必要なアクセス許可、サポートされているプロセッサ、標準の OpenSearch 実装との主な違いなど、OpenSearch Serverless でニューラル検索を設定する方法について説明します。

ニューラル検索の使用により、データに対してセマンティック検索を実行できます。セマンティック検索では、セマンティックな意味合いを考慮して検索クエリの意図を理解します。この機能は、次のコンポーネントを利用しています。

- テキスト埋め込み取り込みパイプラインプロセッサ
- ニューラルクエリ
- ニューラルスパースクエリ

ハイブリッド検索

ハイブリッド検索を使用してキーワード検索機能とセマンティック検索の各機能を組み合わせることで、検索の関連性を向上させることができます。ハイブリッド検索を使用するには、検索結果を処理しドキュメントスコアを組み合わせるための検索パイプラインを作成します。詳細については、「OpenSearch のドキュメント」ウェブサイトの「Search pipelines」を参照してください。ハイブリッド検索を実装するには、次のコンポーネントを使用します。

- 正規化検索パイプラインプロセッサ

  サポートされている正規化手法

  - `min_max`

- l2

サポートされている組み合わせ手法

- arithmetic_mean
- geometric_mean
- harmonic_mean

正規化と組み合わせの手法の詳細については、「OpenSearch のドキュメント」ウェブサイトの「Request body fields」を参照してください。

- ハイブリッドクエリ

ニューラルクエリとハイブリッドクエリ

デフォルトでは、OpenSearch はキーワードベースの Okapi BM25 アルゴリズムを使用してドキュメントスコアを計算します。これは、キーワードを含む検索クエリに適しています。ニューラル検索では、自然言語クエリの新しいクエリタイプと、セマンティック検索とキーワード検索の両方を組み合わせる機能が提供されます。

Example: neural

```
"neural": {
  "vector_field": {
    "query_text": "query_text",
    "query_image": "image_binary",
    "model_id": "model_id",
    "k": 100
  }
}
```

詳細については、「OpenSearch のドキュメント」ウェブサイトの「Neural query」を参照してください。

Example: hybrid

```
"hybrid": {
      "queries": [
        array of lexical, neural, or combined queries
      ]
    }
```

詳細については、「OpenSearch のドキュメント」ウェブサイトの「Hybrid query」を参照してください。

Amazon OpenSearch Serverless でセマンティック検索コンポーネントを設定するには、「OpenSearch のドキュメント」ウェブサイトの「Neural Search tutorial」の手順に従います。次の重要な違いに注意してください。

- OpenSearch Serverless はリモートモデルのみをサポートします。リモートでホストされたモデルにコネクタを設定する必要があります。リモートモデルをデプロイまたは削除する必要はありません。詳細については、「OpenSearch のドキュメント」ウェブサイトの「Getting started with semantic and hybrid search」を参照してください。
- ベクトルインデックスに対して検索する場合、または最近作成した検索パイプラインと取り込みパイプラインを検索する場合、最大 15 秒のレイテンシーが予想されます。

アクセス許可の設定

OpenSearch Serverless でのニューラル検索には、次のアクセス許可が必要です。詳細については、「サポートされているポリシーのアクセス許可」を参照してください。

Example: ニューラル検索ポリシー

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "NeuralSearch",
            "Effect": "Allow",
            "Action": [
              "aoss:CreateIndex",
              "aoss:CreateCollection",
              "aoss:UpdateCollection",
              "aoss:DeleteIndex",
              "aoss:DeleteCollection"
            ],
            "Resource": "arn:aws:aoss:us-east-1:111122223333:collection/your-collection-name"
        }
    ]
}
```

- aoss:*Index – テキスト埋め込みを保存するベクトルインデックスを作成します。
- aoss:*CollectionItems – 取り込みパイプラインと検索パイプラインを作成します。
- aoss:*MLResource – テキスト埋め込みモデルを作成して登録します。
- aoss:APIAccessAll – 検索オペレーションと取り込みオペレーション用に OpenSearch API へのアクセスを提供します。

以下に、ニューラル検索に必要なコレクションデータアクセスポリシーについて説明します。########は、ユーザー固有の情報に置き換えてください。

```
            {
                "ResourceType": "index",
                "Resource": ["index/collection_name/*"],
                "Permission": [
                  "aoss:CreateIndex",
                  "aoss:DescribeIndex",
                  "aoss:UpdateIndex",
                  "aoss:DeleteIndex"
                ]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
        ]
    },
    {
        "Description": "Create pipeline permission",
        "Rules": [
            {
                "ResourceType": "collection",
                "Resource": ["collection/collection_name"],
                "Permission": [
                  "aoss:CreateCollectionItems",
                  "aoss:DescribeCollectionItems",
                  "aoss:UpdateCollectionItems",
                  "aoss:DeleteCollectionItems"
                ]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
        ]
    },
    {
        "Description": "Create model permission",
        "Rules": [
            {
                "ResourceType": "model",
                "Resource": ["model/collection_name/*"],
                "Permission": ["aoss:CreateMLResources"]
            }
        ],
        "Principal": [
            "arn:aws:iam::account_id:role/role_name"
        ]
    }
]
```

サポートされている AWS リージョン

OpenSearch Serverless のニューラル検索は、以下で利用できます AWS リージョン。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ストックホルム)
- 欧州 (スペイン)

Amazon OpenSearch Serverless でのワークフロー設定

ワークフロー

ワークフロー機能は、OpenSearch でのイノベーティブな AI アプリケーションの構築を目指すビルダーをサポートします。セマンティック検索など、OpenSearch で機械学習 (ML) サービスを使用する現在のプロセスでは、複雑なセットアップと前処理タスクに加えて、詳細なユーザークエリが必要です。どちらも時間がかかり、エラーが発生しやすくなります。ワークフローは、OpenSearch の複数の API コールを連鎖するための簡素化フレームワークです。

セットアップと使用については、OpenSearch ウェブサイトの「Automating configurations」を参照してください。OpenSearch Serverless でワークフローを使用する場合は、次の重要な違いを考慮してください。

- OpenSearch Serverless は、ワークフローステップでリモートモデルのみを使用します。これらのモデルをデプロイする必要はありません。
- OpenSearch Serverless は、再インデックスワークフローステップをサポートしていません。
- 別の API コールを実行した後にワークフローやワークフロー状態を検索する場合、更新が表示されるまでに最大 15 秒のレイテンシーが予想されます。

OpenSearch Serverless コレクションは、OpenSearch UI アプリケーション内でデータソースとして使用される場合に限り、ワークフローをサポートします。詳細については、「Managing data source associations」を参照してください。

アクセス許可の設定

テンプレートを作成してプロビジョニングする前に、必要なアクセス許可があることを確認してください。サポートが必要な場合は、アカウント管理者にお問い合わせください。OpenSearch Serverless ワークフローには、次のアクセス許可が必要です。IAM ポリシーでコレクションリソース ARN を定義することで、特定のコレクションに対するアクセス許可の範囲を指定できます。

Example: ワークフローポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "NeuralSearch",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/Cognito_identitypoolname/Auth_Role"
        ]
      },
      "Action": [
        "aoss:CreateIndex",
        "aoss:CreateCollection",
        "aoss:UpdateCollection",
        "aoss:DeleteIndex",
        "aoss:DeleteCollection"
      ],
      "Resource": "arn:aws:aoss:us-east-1:111122223333:collection/your-collection-name"
    }
  ]
}
```

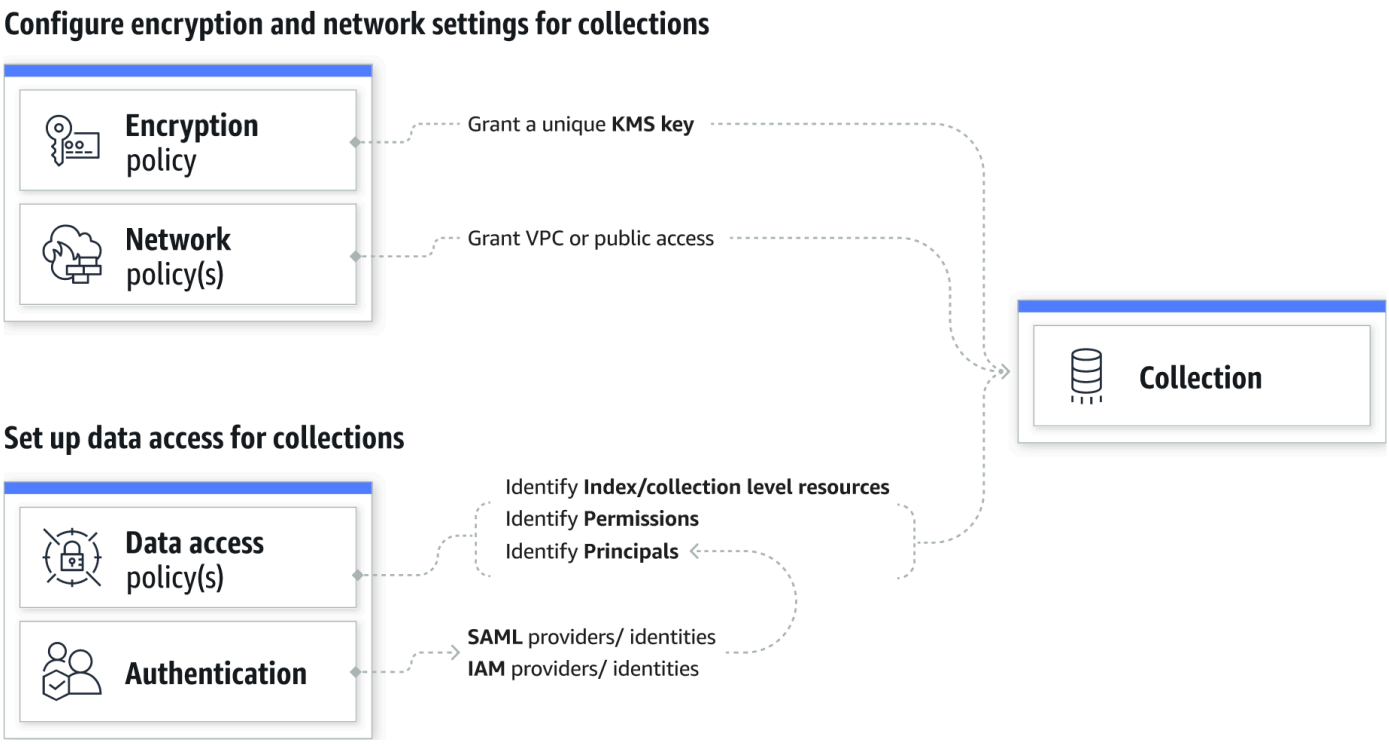
- aoss:*CollectionItems – テンプレートを作成および管理し、検索パイプラインと取り込みパイプラインをプロビジョニングするアクセス許可を付与します。
- aoss:*Index – OpenSearch API オペレーションを使用してインデックスを作成および削除するアクセス許可を付与します。
- aoss:*MLResource – 機械学習の設定を使用するワークフローステップをプロビジョニングするアクセス許可を付与します。

Amazon OpenSearch Serverless のセキュリティの概要

Amazon OpenSearch Serverless のセキュリティは、次の点で Amazon OpenSearch Service のセキュリティと根本的に異なります。

機能	OpenSearch Service	OpenSearch サーバーレス
データアクセスコントロール	きめ細かなアクセスコントロールと IAM ポリシーによりデータアクセスが決定します。	データアクセスポリシーによりデータアクセスが決定します。
保管時の暗号化	ドメインについては、保管時の暗号化はオプションとなります。	コレクションについては、保管時の暗号化は必須となります。
セキュリティの設定および管理者	ネットワーク、暗号化、およびデータアクセスは、ドメインごとに個別に設定する必要があります。	セキュリティポリシーを使用して、複数のコレクションのセキュリティ設定を大規模に管理できます。

次の図は、機能的なコレクションを設定するセキュリティのコンポーネントを示しています。コレクションには、暗号化キー、ネットワークアクセス設定、リソースへのアクセス許可を付与するデータアクセスポリシーが割り当てられている必要があります。



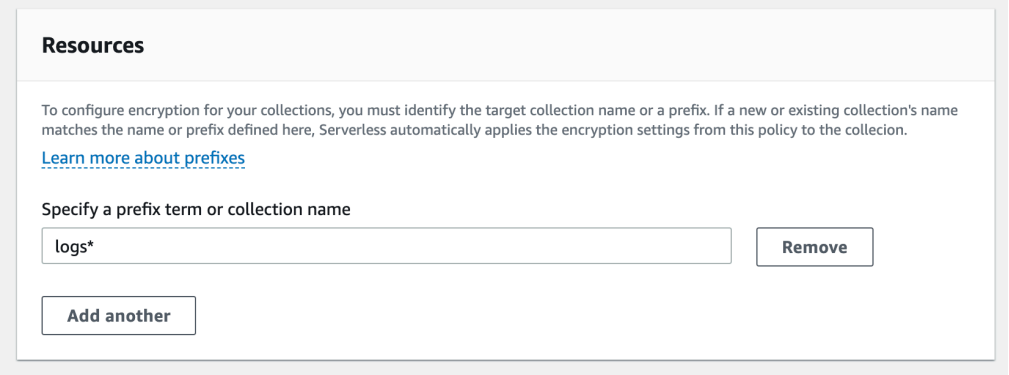
トピック

- 暗号化ポリシー
- ネットワークポリシー
- データアクセスポリシー
- IAM および SAML 認証
- インフラストラクチャセキュリティ
- Amazon OpenSearch Serverless でのセキュリティの開始方法
- Amazon OpenSearch Serverless 向けの アイデンティティとアクセス管理
- Amazon OpenSearch Serverless の暗号化
- Amazon OpenSearch Serverless でのネットワークアクセス
- Amazon OpenSearch Serverless での FIPS コンプライアンス
- Amazon OpenSearch Serverless のデータアクセスコントロール
- を介したデータプレーンアクセス AWS PrivateLink
- を介したコントロールプレーンアクセス AWS PrivateLink
- Amazon OpenSearch Serverless での SAML 認証
- Amazon OpenSearch Serverless のコンプライアンス検証

暗号化ポリシー

暗号化ポリシーは、コレクションを AWS 所有のキー またはカスタマーマネージドキーで暗号化するかどうかを定義します。暗号化ポリシーは、リソースパターンと暗号化キーの 2 つの要素で構成されます。リソースパターンは、ポリシーが適用される 1 つまたは複数のコレクションを定義します。暗号化キーは、関連するコレクションを保護する方法を決定します。

ポリシーを複数のコレクションに適用するには、ポリシーのルールにワイルドカード (*) を含めます。例えば、次のポリシーは、名前が「logs」で始まるすべてのコレクションに適用されます。

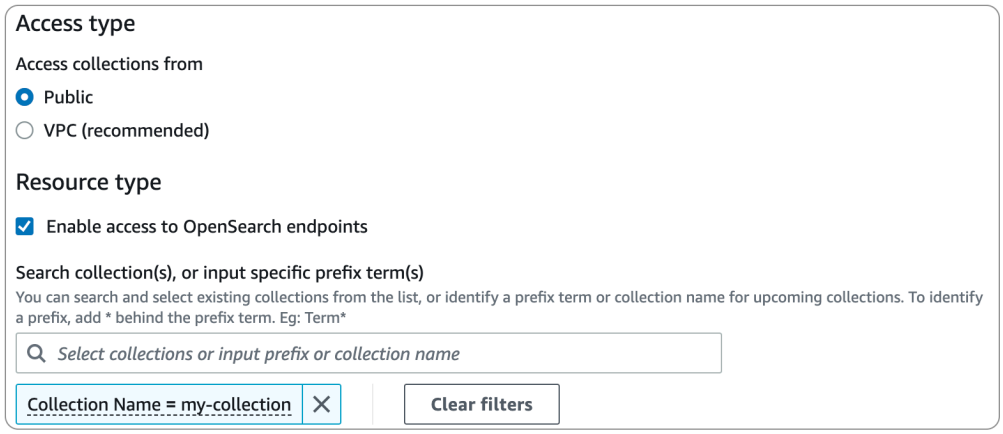


暗号化ポリシーは、特にプログラムで行う場合に、コレクションの作成と管理のプロセスを効率化します。コレクションは名前を指定することで作成でき、作成時に暗号化キーが自動的に割り当てられます。

ネットワークポリシー

ネットワークポリシーは、コレクションにプライベートでアクセスするか、パブリックネットワークからインターネット経由でアクセスするかを定義します。プライベートコレクションには、OpenSearch Serverless が管理する VPC エンドポイント、またはAWS のサービス プライベートアクセスを使用して Amazon Bedrock などの特定の AWS のサービス からアクセスできます。ネットワークポリシーは、暗号化ポリシーと同様に複数のコレクションに適用できるため、多数のコレクションのネットワークアクセスを大規模に管理できます。

ネットワークポリシーは、アクセスタイプとリソースタイプの 2 つの要素で構成されます。アクセスタイプはパブリックでもプライベートでもかまいません。リソースタイプによって、選択したアクセスがコレクションエンドポイント、OpenSearch Dashboards エンドポイント、またはその両方に適用されるかが決まります。



ネットワークポリシー内で VPC アクセスを設定する場合は、まず OpenSearch Serverless マネージド VPC エンドポイントを 1 つ以上作成する必要があります。これらのエンドポイントを使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのように OpenSearch Serverless にアクセスできます。

へのプライベートアクセス AWS のサービス は、コレクションの OpenSearch エンドポイントにのみ適用 AWS のサービス でき、OpenSearch Dashboards エンドポイントには適用されません。OpenSearch Dashboards へのアクセスを許可することはできません。

データアクセスポリシー

データアクセスポリシーは、ユーザーがコレクション内のデータにアクセスする方法を定義します。データアクセスポリシーは、特定のパターンに一致するコレクションとインデックスにアクセス許可を自動的に割り当てることにより、大規模なコレクションを管理するのに役立ちます。複数のポリシーを単一のリソースに適用できます。

データアクセスポリシーはルールのセットで構成され、それぞれに 3 つの構成要素 (リソースタイプ、付与されたリソース、およびアクセス許可のセット) があります。リソースタイプはコレクションでもインデックスでもかまいません。付与されたリソースは、コレクション名またはインデックス名、あるいはワイルドカード (*) 付きのパターンにすることができます。アクセス許可のリストは、ポリシーがアクセスを許可する OpenSearch API オペレーションを指定します。さらにポリシーには、アクセスを許可する IAM ロール、ユーザー、SAML ID を指定するプリンシパルのリストが含まれています。



データアクセスポリシーの形式の詳細については、ポリシー構文を参照してください。

データアクセスポリシーを作成する前に、ポリシーでアクセスを提供するための、1 つ、または複数の IAM ロールもしくはユーザー、または SAML ID が必要です。詳細については、次のセクションを参照ください。

> **Note**
>
> コレクションをパブリックアクセスからプライベートアクセスに切り替えると、OpenSearch Serverless コレクションコンソールの [インデックス] タブが削除されます。

IAM および SAML 認証

IAM プリンシパルおよび SAML ID は、データアクセスポリシーの構成要素の 1 つです。アクセスポリシーの `principal` ステートメントには、IAM ロール、ユーザー、および SAML ID を含めることができます。その後、関連するポリシールールで指定したアクセス許可がこれらのプリンシパルに付与されます。

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/marketing/orders*"
            ],
            "Permission":[
```

```
                "aoss:*"
            ]
         }
      ],
      "Principal":[
         "arn:aws:iam::123456789012:user/Dale",
         "arn:aws:iam::123456789012:role/RegulatoryCompliance",
         "saml/123456789012/myprovider/user/Annie"
      ]
   }
]
```

SAML 認証は、OpenSearch Serverless 内で直接設定します。詳細については、「the section called "SAML 認証"」を参照してください。

インフラストラクチャセキュリティ

Amazon OpenSearch Serverless は AWS グローバルネットワークセキュリティで保護されています。セキュリティサービスと AWS がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で Amazon OpenSearch Serverless にアクセスします。クライアントは Transport Layer Security (TLS) をサポートしている必要があります。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。TLS 1.3 でサポートされている暗号のリストについては、Elastic Load Balancing ドキュメントの「TLS protocols and ciphers」を参照してください。

また、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用してリクエストに署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon OpenSearch Serverless でのセキュリティの開始方法

次のチュートリアルは、Amazon OpenSearch Serverless の使用を開始するのに役立ちます。どちらのチュートリアルも同じ基本的なステップを実行しますが、一方は コンソールを使用し、もう一方は を使用します AWS CLI。

これらのチュートリアルのユースケースは簡素化されています。ネットワークおよびセキュリティポリシーは、かなりオープンです。本番ワークロードでは、SAML 認証、VPC アクセス、制限付きデータアクセスポリシーなど、より堅牢なセキュリティ機能を設定します。

トピック

- チュートリアル: Amazon OpenSearch Serverless でのセキュリティの開始方法 (コンソール)
- チュートリアル: Amazon OpenSearch Serverless (CLI) でのセキュリティの開始方法

チュートリアル: Amazon OpenSearch Serverless でのセキュリティの開始方法 (コンソール)

このチュートリアルでは、Amazon OpenSearch Serverless コンソールを使用してセキュリティポリシーを作成および管理します。

このチュートリアルでは、次の手順を実行します。

1. アクセス許可を設定
2. 暗号化ポリシーを作成
3. ネットワークポリシーを作成する
4. データアクセスポリシーを設定する
5. コレクションを作成
6. データをアップロードおよび検索する

このチュートリアルでは、 を使用してコレクションを設定する方法を示します AWS マネジメントコンソール。を使用した同じ手順については AWS CLI、「」を参照してくださいthe section called “チュートリアル: セキュリティの開始方法 (CLI)”。

ステップ 1: アクセス許可を設定する

> **Note**
>
> `Action":"aoss:*"` や `Action":"*"` など、より広範な ID ベースのポリシーを既に使用している場合は、この手順をスキップできます。ただし、本番環境では、最小特権の原則に従い、タスクを完了するために必要な最小限のアクセス許可のみを割り当てます。

このチュートリアルを完了するには、適切な IAM のアクセス許可を持っている必要があります。ユーザーまたはロールには、以下の最低限の許可を含む ID ベースのポリシーが、アタッチされている必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "aoss:ListCollections",
        "aoss:BatchGetCollection",
        "aoss:CreateCollection",
        "aoss:CreateSecurityPolicy",
        "aoss:GetSecurityPolicy",
        "aoss:ListSecurityPolicies",
        "aoss:CreateAccessPolicy",
        "aoss:GetAccessPolicy",
        "aoss:ListAccessPolicies"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

OpenSearch Serverless のアクセス許可に関する詳細なリストについては、「the section called “Identity and Access Management”」を参照してください。

ステップ 2: 暗号化ポリシーを作成する

暗号化ポリシーは、 AWS KMS OpenSearch Serverless がコレクションの暗号化に使用するキーを指定します。コレクションは、 AWS マネージドキー または別のキーで暗号化できます。このチュートリアルを簡単にするために、コレクションを で暗号化します AWS マネージドキー。

暗号化ポリシーを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。

2. 左側のナビゲーションペインで [Serverless] を展開し、[Encryption policies] (暗号化ポリシー) を選択します。
3. [Create encryption policy] (暗号化ポリシーを作成) を選択します。
4. ポリシーには books-policy という名前を付けます。説明には、「Encryption policy for books collection」を入力します。
5. 「リソース」に「」と入力します。これはbooksコレクションの名前です。より広範囲にする場合は、アスタリスク (books*) を含めて、ポリシーを「books」という単語で始まるすべてのコレクションに適用できます。
6. 暗号化では、 AWS 所有キーの使用 を選択したままにします。
7. [作成] を選択します。

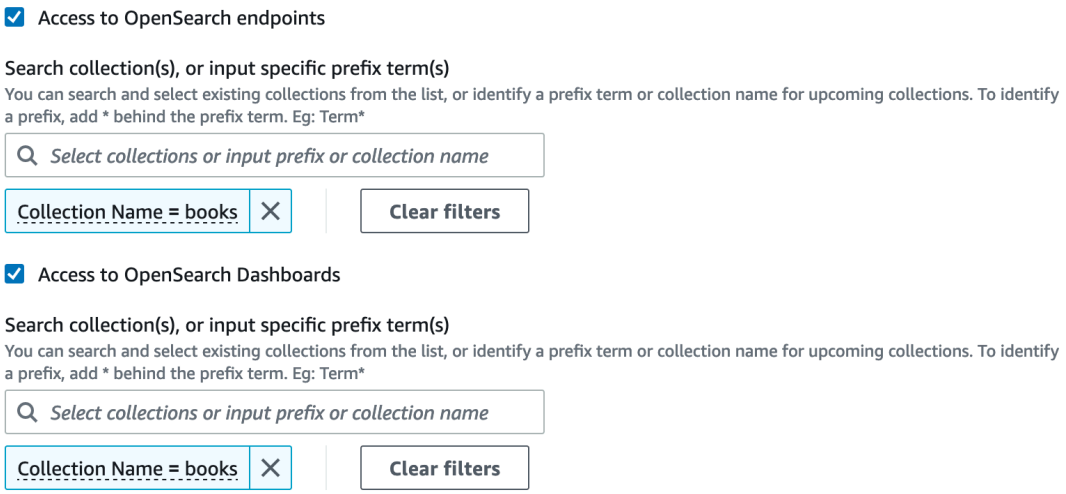
ステップ 3: ネットワークポリシーを作成する

ネットワークポリシーでは、コレクションにパブリックネットワークからインターネット経由でアクセス可能にするか、OpenSearch Serverless が管理する VPC エンドポイント経由でのアクセスが必要にするかを決定できます。このチュートリアルでは、パブリックアクセスを設定します。

ネットワークポリシーを作成するには

1. 左のナビゲーションペインで [Network policies] (ネットワークポリシー) を選択し、[Create network policy] (ネットワークポリシーの作成) を選択します。
2. ポリシーには books-policy という名前を付けます。説明には、「Network policy for books collection」を入力します。
3. ルール 1 で、ルールに という名前を付けますPublic access for books collection。
4. このチュートリアルを簡単にするために、書籍コレクションのパブリックアクセスを設定します。アクセスタイプには、[Public] (パブリック) を選択します。
5. OpenSearch Dashboards からコレクションにアクセスします。そのためには、ダッシュボードと OpenSearch エンドポイントのネットワークアクセスを設定する必要があります。そうしないと、ダッシュボードは機能しません。

   リソースタイプは、[Access to OpenSearch endpoints] (OpenSearch エンドポイントへのアクセス)と [Access to OpenSearch Dashboards] (OpenSearch Dashboardsへのアクセス) の両方を有効にします。
6. 両方の入力ボックスに と入力しますCollection Name = books。このように設定すると、ポリシーの範囲が狭くなり 1 つのコレクション (books) にのみ適用されるようになります。ルールは次のようになります。



7. [作成] を選択します。

ステップ 4: データアクセスポリシーを作成する

データアクセスを設定するまで、コレクションデータにアクセスすることはできません。データアクセスポリシーは、ステップ 1 で設定した IAM ID ベースのポリシーとは別のものです。このポリシーでは、コレクション内の実際のデータにアクセスできます。

このチュートリアルでは、データのインデックス作成に必要なアクセス許可を 1 人のユーザーに付与します。

データアクセスポリシーを作成するには

1. 左のナビゲーションペインで、[Data access policies] (データアクセスポリシー) を選択し、[Create access policy] (アクセスポリシーを作成) を選択します。
2. ポリシーには books-policy という名前を付けます。説明には、「`Data access policy for books collection`」を入力します。
3. ポリシーの定義方法として [JSON] を選択し、JSON エディターに次のポリシーを貼り付けます。

   プリンシパル ARN を、OpenSearch Dashboards へのログインとデータのインデックス作成に使用するアカウントの ARN に置き換えます。

```
[
   {
      "Rules":[
         {
```

```
            "ResourceType":"index",
            "Resource":[
               "index/books/*"
            ],
            "Permission":[
               "aoss:CreateIndex",
               "aoss:DescribeIndex",
               "aoss:ReadDocument",
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:DeleteIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:iam::123456789012:user/my-user"
      ]
   }
]
```

このポリシーは、books コレクション内のインデックス作成や一部のデータのインデックス化、および検索に必要な最低限のアクセス許可を 1 人のユーザーに提供します。

4. [作成] を選択します。

ステップ 5: コレクションを作成する

暗号化ポリシーとネットワークポリシーを設定したら、それらに一致するコレクションを作成できます。OpenSearch Serverless はセキュリティ設定を自動的に適用します。

OpenSearch Serverless コレクションを作成するには

1. 左側のナビゲーションペインで [Collections] (コレクション) 、[Create collection] (コレクションを作成) を選択します。
2. Serverless generation フィールドで、Classic にない場合は Classic に切り替えるを選択します。
3. コレクション名には、 と入力しますbooks。
4. コレクションタイプでは、[Search] (検索) を選択します。
5. [Encryption] (暗号化) で、OpenSearch Serverless によりコレクション名が books-policy 暗号化ポリシーと一致することが通知されます。

6. [Network access settings] (ネットワークアクセス設定) で、OpenSearch Serverless によりコレクション名が `books-policy` ネットワークポリシーと一致することが通知されます。
7. [次へ] を選択します。
8. [Data access policy options] (データアクセスポリシーのオプション) で、OpenSearch Serverless によりコレクション名が `books-policy` データアクセスポリシーと一致することが通知されます。
9. [次へ] を選択します。
10. OpenSearch UI の設定で、コレクションの OpenSearch アプリケーションとワークスペースを設定します。既存の OpenSearch アプリケーションの選択または新しい OpenSearch アプリケーションの作成を選択し、ワークスペースを選択または作成します。[次へ] を選択します。
11. コレクションの設定を確認し、[Submit] (送信) を選択します。通常、コレクションの初期化には 1 分もかかりません。

> **Note**
>
> このチュートリアルでは、 Classic コレクション作成フローを使用して、コレクション作成時に事前設定されたセキュリティポリシーを自動的に一致させる方法を示します。NextGen フローを使用してコレクションを作成する方法については、「」を参照してくださいthe section called “コレクションの作成”。

ステップ 6: データをアップロードおよび検索する

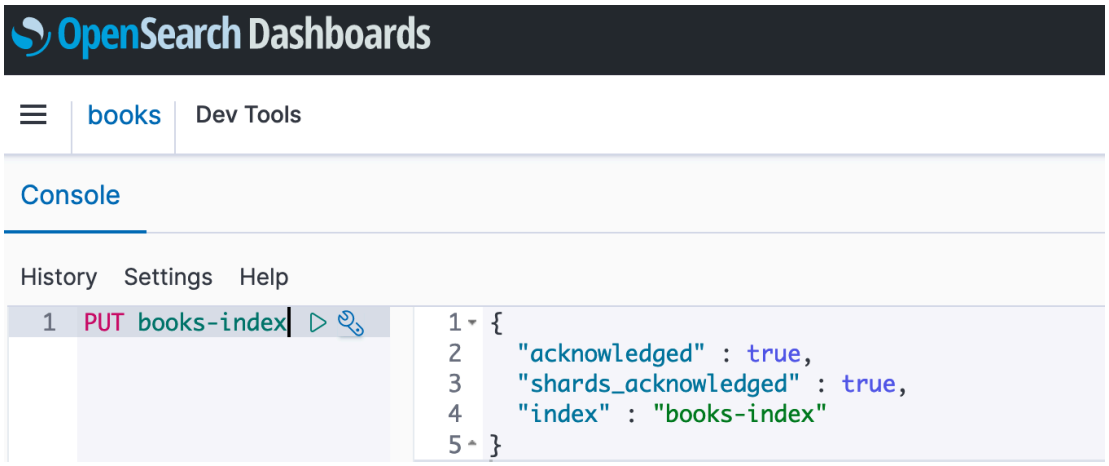
OpenSearch Serverless コレクションへのデータのアップロードには、Postman または curl が使用できます。簡潔に言うと、これらの例では OpenSearch Dashboards コンソール内の開発ツールを使用しています。

コレクション内のデータをインデックス化して検索するには

1. 左側のナビゲーションペインで [Collections] (コレクション) を選択し、books コレクションを選択して詳細ページを開きます。
2. コレクションの OpenSearch Dashboards URL を選択します。この URL の形式は、`https://collection-id.us-east-1.aoss.amazonaws.com/_dashboards` になります。
3. データアクセスポリシーで指定したプリンシパルの AWS アクセスキーとシークレットキーを使用して、OpenSearch Dashboards にサインインします。

4. OpenSearch Dashboards 内で左側のナビゲーションメニューを開き、[Dev Tools] (開発ツール) を選択します。
5. books-index という単一のインデックスを作成するには、次のコマンドを実行します。

```
PUT books-index
```



6. 単一のドキュメントを books-index にインデックス化するには、次のコマンドを実行します。

```
PUT books-index/_doc/1
{
  "title": "The Shining",
  "author": "Stephen King",
  "year": 1977
}
```

7. OpenSearch Dashboards 内でデータを検索するためには、少なくとも 1 つのインデックスパターンを設定する必要があります。OpenSearch は、そのパターンを使用して、分析する対象のインデックスを特定します。Dashboards のメインメニューを開き、[スタック管理] を選択し、[インデックスパターン] を選択してから、[インデックスパターンを作成する] を選択します。このチュートリアルでは、「books-index」と入力します。
8. [次のステップ] を選択してから、[インデックスパターンの作成] を選択します。パターンが作成されたら、`author` および `title` などのさまざまなドキュメントフィールドを表示できます。
9. データの検索を開始するには、メインメニューをもう一度開き [Discover] (検出) を選択するか、検索 API を使用します。

チュートリアル: Amazon OpenSearch Serverless (CLI) でのセキュリティの開始方法

このチュートリアルでは、セキュリティに関するコンソール入門チュートリアルで説明されている手順について説明しますが、OpenSearch Service コンソール AWS CLI ではなく を使用します。

このチュートリアルでは、次の手順を実行します。

1. IAM アクセス権限ポリシーを作成する
2. IAM ポリシーを IAM ロールにアタッチする
3. 暗号化ポリシーを作成する
4. ネットワークポリシーを作成する
5. コレクションを作成
6. データアクセスポリシーを設定する
7. コレクションエンドポイントを取得する
8. コレクションにデータをアップロードする
9. コレクション内のデータを検索する

このチュートリアルの目的は、簡単な暗号化、ネットワーク、データアクセス設定がある 1 つの OpenSearch Serverless コレクションを設定することです。たとえば、パブリックネットワークアクセス、暗号化 AWS マネージドキー 用の 、および単一のユーザーに最小限のアクセス許可を付与する簡略化されたデータアクセスポリシーを設定します。

本番環境では、SAML 認証、カスタム暗号化キー、VPC アクセスなどのより強固な設定を実装することを検討してください。

OpenSearch Serverless でセキュリティポリシーの使用を開始するには

1.

> **Note**
>
> `Action":"aoss:*"` や `Action":"*"` など、より広範な ID ベースのポリシーを既に使用している場合は、この手順をスキップできます。ただし、本番環境では、最小特権の原則に従い、タスクを完了するために必要な最小限のアクセス許可のみを割り当てます。

開始するには、このチュートリアルのステップを実行するために必要な最小限のアクセス許可を持つ AWS Identity and Access Management ポリシーを作成します。ポリシーに という名前を付けますTutorialPolicy。

```
aws iam create-policy \
  --policy-name TutorialPolicy \
  --policy-document "{\"Version\": \"2012-10-17\",\"Statement\":
 [{\"Action\": [\"aoss:ListCollections\",\"aoss:BatchGetCollection\",
\"aoss:CreateCollection\",\"aoss:CreateSecurityPolicy\",\"aoss:GetSecurityPolicy\",
\"aoss:ListSecurityPolicies\",\"aoss:CreateAccessPolicy\",\"aoss:GetAccessPolicy\",
\"aoss:ListAccessPolicies\"],\"Effect\": \"Allow\",\"Resource\": \"*\"}]}"
```

レスポンス例

```
{
    "Policy": {
        "PolicyName": "TutorialPolicy",
        "PolicyId": "ANPAW6WRAECKG6QJWUV7U",
        "Arn": "arn:aws:iam::123456789012:policy/TutorialPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2022-10-16T20:57:18+00:00",
        "UpdateDate": "2022-10-16T20:57:18+00:00"
    }
}
```

2. コレクション内でデータをインデックス化して検索する IAM ロールに TutorialPolicy をアタッチします。この例では、ロールの名前は ですTutorialRole。

```
aws iam attach-role-policy \
  --role-name TutorialRole \
  --policy-arn arn:aws:iam::123456789012:policy/TutorialPolicy
```

3. コレクションを作成する前に、後のステップで作成した書籍コレクション AWS 所有のキー にを割り当てる暗号化ポリシーを作成する必要があります。

   books コレクション用の暗号化ポリシーを作成するには、次のリクエストを送信します。

```
aws opensearchserverless create-security-policy \
  --name books-policy \
  --type encryption --policy "{\"Rules\":[{\"ResourceType\":\"collection\",
\"Resource\":[\"collection\/books\"]}],\"AWSOwnedKey\":true}"
```

レスポンス例

```
{
    "securityPolicyDetail": {
        "type": "encryption",
        "name": "books-policy",
        "policyVersion": "MTY2OTI0MDAwNTk5MF8x",
        "policy": {
            "Rules": [
                {
                    "Resource": [
                        "collection/books"
                    ],
                    "ResourceType": "collection"
                }
            ],
            "AWSOwnedKey": true
        },
        "createdDate": 1669240005990,
        "lastModifiedDate": 1669240005990
    }
}
```

4. books コレクションにパブリックアクセスを提供するネットワークポリシーを作成します。

```
aws opensearchserverless create-security-policy --name books-policy --type network
 \
  --policy "[{\"Description\":\"Public access for books collection\",\"Rules
\":[{\"ResourceType\":\"dashboard\",\"Resource\":[\"collection\/books\"]},
{\"ResourceType\":\"collection\",\"Resource\":[\"collection\/books\"]}],
\"AllowFromPublic\":true}]"
```

レスポンス例

```
{
    "securityPolicyDetail": {
```

```
        "type": "network",
        "name": "books-policy",
        "policyVersion": "MTY2OTI0MDI1Njk1NV8x",
        "policy": [
            {
                "Rules": [
                    {
                        "Resource": [
                            "collection/books"
                        ],
                        "ResourceType": "dashboard"
                    },
                    {
                        "Resource": [
                            "collection/books"
                        ],
                        "ResourceType": "collection"
                    }
                ],
                "AllowFromPublic": true,
                "Description": "Public access for books collection"
            }
        ],
        "createdDate": 1669240256955,
        "lastModifiedDate": 1669240256955
    }
}
```

5. books コレクションを作成します。

```
aws opensearchserverless create-collection --name books --type SEARCH
```

レスポンス例

```
{
    "createCollectionDetail": {
        "id": "8kw362bpwg4gx9b2f6e0",
        "name": "books",
        "status": "CREATING",
        "type": "SEARCH",
        "arn": "arn:aws:aoss:us-
east-1:123456789012:collection/8kw362bpwg4gx9b2f6e0",
        "kmsKeyArn": "auto",
```

```
        "createdDate": 1669240325037,
        "lastModifiedDate": 1669240325037
    }
}
```

6. books コレクション内のデータをインデックス化して検索するための最小限のアクセス許可を付与するデータアクセスポリシーを作成します。プリンシパル ARN をステップ 1 の TutorialRole の ARN に置き換えます。

```
aws opensearchserverless create-access-policy \
  --name books-policy \
  --type data \
  --policy "[{\"Rules\":[{\"ResourceType\":\"index\",\"Resource\":
[\"index\/books\/books-index\"],\"Permission\":[\"aoss:CreateIndex
\",\"aoss:DescribeIndex\",\"aoss:ReadDocument\",\"aoss:WriteDocument
\",\"aoss:UpdateIndex\",\"aoss:DeleteIndex\"]}],\"Principal\":
[\"arn:aws:iam::123456789012:role\/TutorialRole\"]}]"
```

レスポンス例

```
{
    "accessPolicyDetail": {
        "type": "data",
        "name": "books-policy",
        "policyVersion": "MTY2OTI0MDM5NDY1M18x",
        "policy": [
            {
                "Rules": [
                    {
                        "Resource": [
                            "index/books/books-index"
                        ],
                        "Permission": [
                            "aoss:CreateIndex",
                            "aoss:DescribeIndex",
                            "aoss:ReadDocument",
                            "aoss:WriteDocument",
                            "aoss:UpdateDocument",
                            "aoss:DeleteDocument"
                        ],
                        "ResourceType": "index"
                    }
```

```
                ],
                "Principal": [
                    "arn:aws:iam::123456789012:role/TutorialRole"
                ]
            }
        ],
        "createdDate": 1669240394653,
        "lastModifiedDate": 1669240394653
    }
}
```

これで、TutorialRole は books コレクション内のドキュメントをインデックス化して検索できるはずです。

7. OpenSearch API への呼び出しを行うには、コレクションエンドポイントが必要です。次のリクエストを送信して、collectionEndpoint パラメータを取得します。

```
aws opensearchserverless batch-get-collection --names books
```

レスポンス例

```
{
    "collectionDetails": [
        {
            "id": "8kw362bpwg4gx9b2f6e0",
            "name": "books",
            "status": "ACTIVE",
            "type": "SEARCH",
            "description": "",
            "arn": "arn:aws:aoss:us-
east-1:123456789012:collection/8kw362bpwg4gx9b2f6e0",
            "createdDate": 1665765327107,
            "collectionEndpoint": "https://8kw362bpwg4gx9b2f6e0.us-
east-1.aoss.amazonaws.com",
            "dashboardEndpoint": "https://8kw362bpwg4gx9b2f6e0.us-
east-1.aoss.amazonaws.com/_dashboards"
        }
    ],
    "collectionErrorDetails": []
}
```

> **Note**
>
> コレクションエンドポイントは、コレクションステータスが に変わるまで使用できませんACTIVE。コレクションが正常に作成されるまで、ステータスを確認するために複数の呼び出しを行う必要がある場合があります。

8. Postman や curl などの HTTP ツールを使用して、データを books コレクションにインデックス化します。次の例では、 books-index というインデックスを作成し、1 つのドキュメントを追加します。

   TutorialRole の認証情報を使用して、次のリクエストを前の手順で取得したコレクションエンドポイントに送信します。

   ```
   PUT https://8kw362bpwg4gx9b2f6e0.us-east-1.aoss.amazonaws.com/books-index/_doc/1
   {
     "title": "The Shining",
     "author": "Stephen King",
     "year": 1977
   }
   ```

   レスポンス例

   ```
   {
     "_index" : "books-index",
     "_id" : "1",
     "_version" : 1,
     "result" : "created",
     "_shards" : {
       "total" : 0,
       "successful" : 0,
       "failed" : 0
     },
     "_seq_no" : 0,
     "_primary_term" : 0
   }
   ```

9. コレクション内でデータの検索を開始するには、検索 API を使用します。次のクエリは、基本的な検索を実行します。

```
GET https://8kw362bpwg4gx9b2f6e0.us-east-1.aoss.amazonaws.com/books-index/_search
```

レスポンス例

```
{
    "took": 405,
    "timed_out": false,
    "_shards": {
        "total": 6,
        "successful": 6,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": {
            "value": 2,
            "relation": "eq"
        },
        "max_score": 1.0,
        "hits": [
            {
                "_index": "books-index:0::3xJq14MBUaOS0wL26UU9:0",
                "_id": "F_bt4oMBLle5pYmm5q4T",
                "_score": 1.0,
                "_source": {
                    "title": "The Shining",
                    "author": "Stephen King",
                    "year": 1977
                }
            }
        ]
    }
}
```

Amazon OpenSearch Serverless 向けの アイデンティティとアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、どのユーザーを認証 (サインイン受け入れ) し、また OpenSearch Serverless リソースの使用を認可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- OpenSearch Serverless でのアイデンティティベースのポリシー
- OpenSearch Serverless でのポリシーアクション
- OpenSearch Serverless のポリシーリソース
- Amazon OpenSearch Serverless のポリシー条件キー
- ABAC と OpenSearch Serverless
- OpenSearch Serverless での一時的な認証情報の使用
- Amazon OpenSearch Serverless でのサービスにリンクされたロール
- その他のポリシータイプ
- OpenSearch Serverless での ID ベースのポリシー例
- IAM Identity Center による Amazon OpenSearch Serverless のサポート

OpenSearch Serverless でのアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

OpenSearch Serverless での ID ベースのポリシー例

OpenSearch Serverless での ID ベースのポリシー例については、「the section called “アイデンティティベースのポリシーの例”」を参照してください。

OpenSearch Serverless でのポリシーアクション

ポリシーアクションのサポート: あり

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレー

ションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

OpenSearch Serverless のポリシーアクションでは、アクションの前に次のプレフィックスを使用します。

```
aoss
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "aoss:action1",
      "aoss:action2"
         ]
```

ワイルドカード文字 (*) を使用すると、複数のアクションを指定することができます。例えば、Describe という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "aoss:List*"
```

OpenSearch Serverless での ID ベースのポリシー例については、「OpenSearch Serverless での ID ベースのポリシー例」を参照してください。

OpenSearch Serverless のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon OpenSearch Serverless のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

OpenSearch Serverless では、属性ベースのアクセス制御 (ABAC) に加えて、以下の条件キーをサポートしています。

- aoss:collection
- aoss:CollectionId
- aoss:index

これらの条件キーは、アクセスポリシーおよびセキュリティポリシーに許可を与える場合でも使用できます。例えば、次のようになります。

```
[
   {
      "Effect":"Allow",
      "Action":[
         "aoss:CreateAccessPolicy",
         "aoss:CreateSecurityPolicy"
      ],
      "Resource":"*",
      "Condition":{
         "StringLike":{
            "aoss:collection":"log"
         }
      }
   }
]
```

この例の条件は、コレクション名またはパターンに一致するルールを含むポリシーに適用されます。コンディションは、以下の動作を行います。

- `StringEquals` – リソース文字列「log」が完全一致で含まれたルールを使用するポリシーに適用されます (例: `collection/log`)。
- `StringLike` –リソース文字列に文字列「log」が含まれたルールを使用するポリシーに適用されます (例: `collection/log` だけでなく `collection/logs-application` や `collection/applogs123` も含む)。

> **Note**
>
> コレクション条件キーは、インデックスレベルでは適用されません。例として上記のポリシーでは、リソース文字列 `index/logs-application/*` を含むアクセスポリシーまたはセキュリティポリシーに対して、この条件は適用されません。

OpenSearch Serverless の条件キーのリストは、「サービス認証リファレンス」の「Condition keys for Amazon OpenSearch Serverless」を参照してください。条件キーの使用が可能なアクションおよびリソースついては、「Actions defined by Amazon OpenSearch Serverless」(Amazon OpenSearch Serverless で定義されるアクション) を参照してください。

ABAC と OpenSearch Serverless

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`*、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM

ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

OpenSearch Serverless リソースでのタグ付けの詳細については、「the section called “タグコレクション”」を参照してください。

OpenSearch Serverless での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

Amazon OpenSearch Serverless でのサービスにリンクされたロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

OpenSearch Serverless でのサービスにリンクされたロールの作成または管理の詳細については、「the section called “コレクション作成ロール”」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の AWS アカウントをグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各アカウントのルートユーザーを含む、メンバー AWS アカウントのエンティティのアクセス許可を制限します。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースの許可を制限し、組織に属するかどうかにかかわらず、 AWS アカウントルートユーザーを含む ID のための有効な許可に影響を及ぼす可能性があります。RCP をサポートするサービスのリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS

OpenSearch Serverless での ID ベースのポリシー例

デフォルトでは、OpenSearch Serverless リソースを作成または変更する許可は、ユーザーおよびロールに付与されていません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

Amazon OpenSearch Serverless が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式など) の詳細については、「サービス認証リファレンス」の「Actions, resources, and condition keys for Amazon OpenSearch Serverless」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- コンソールでの OpenSearch Serverless の使用
- OpenSearch Serverless コレクションの管理
- OpenSearch Serverless コレクションの表示
- OpenSearch API オペレーションの使用
- OpenSearch API オペレーションの ABAC

ポリシーに関するベストプラクティス

アイデンティティベースポリシーは非常に強力です。このポリシーは、アカウント内の OpenSearch Serverless リソースを作成、アクセス、または削除することが可能なユーザーを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

アイデンティティベースのポリシーは、アカウント内で誰かが OpenSearch Serverless リソースの作成、アクセス、または削除を実行できるどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

コンソールでの OpenSearch Serverless の使用

OpenSearch Service コンソールで OpenSearch Serverless にアクセスするには、最小限の許可セットが必要です。これらのアクセス許可により、 AWS アカウントの OpenSearch Serverless リソースの詳細を一覧表示および表示できます。必要最低限の許可よりも制限が厳しいアイデンティティベースポリシーを作成すると、そのポリシーを持つエンティティ (IAM ロールなど) に対してコンソールが意図したとおりに機能しなくなります。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

以下のポリシーでは、ユーザーが OpenSearch Service コンソール内で OpenSearch Serverless にアクセスすることを許可しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "*",
            "Effect": "Allow",
            "Action": [
                "aoss:ListCollections",
                "aoss:BatchGetCollection",
                "aoss:ListAccessPolicies",
                "aoss:ListSecurityConfigs",
                "aoss:ListSecurityPolicies",
                "aoss:ListTagsForResource",
                "aoss:ListVpcEndpoints",
                "aoss:GetAccessPolicy",
                "aoss:GetAccountSettings",
                "aoss:GetSecurityConfig",
                "aoss:GetSecurityPolicy"
            ]
        }
    ]
}
```

OpenSearch Serverless コレクションの管理

このポリシーは、ユーザーに対し Amazon OpenSearch Serverless コレクションの処理および管理を許可する、「コレクション管理者」ポリシーの一例です。これによりユーザーは、コレクションを作成、表示、削除できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "arn:aws:aoss:us-east-1:111122223333:collection/*",
            "Action": [
                "aoss:CreateCollection",
                "aoss:DeleteCollection",
                "aoss:UpdateCollection"
            ],
            "Effect": "Allow"
        },
        {
            "Resource": "*",
            "Action": [
                "aoss:BatchGetCollection",
                "aoss:ListCollections",
                "aoss:CreateAccessPolicy",
                "aoss:CreateSecurityPolicy"
            ],
            "Effect": "Allow"
        }
    ]
}
```

OpenSearch Serverless コレクションの表示

このポリシー例では、ユーザーに対し、自身のアカウント内のすべての Amazon OpenSearch Serverless コレクションについて、その詳細を表示することを許可します。このユーザーには、コレクション自体や関連するセキュリティポリシーの変更は許可されません。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Resource": "*",
            "Action": [
                "aoss:ListAccessPolicies",
                "aoss:ListCollections",
                "aoss:ListSecurityPolicies",
                "aoss:ListTagsForResource",
                "aoss:BatchGetCollection"
            ],
            "Effect": "Allow"
        }
    ]
}
```

OpenSearch API オペレーションの使用

データプレーン API オペレーションは、OpenSearch Serverless でサービスからリアルタイムの値を引き出すために使用する関数で構成されています。コントロールプレーン API オペレーションは、環境のセットアップに使用する関数で構成されています。

Amazon OpenSearch Serverless データプレーン API と OpenSearch Dashboards にブラウザからアクセスするときは、コレクションリソース用に 2 つの IAM アクセス権限を追加する必要があります。これらのアクセス権限は、`aoss:APIAccessAll` と `aoss:DashboardsAccessAll` です。

> **Note**
>
> 2023 年 5 月 10 日以降、OpenSearch Serverless では、コレクションリソースにこれら 2 つの新しい IAM 許可が必要になります。`aoss:APIAccessAll` のアクセス許可はデータプレーンアクセスを許可し、`aoss:DashboardsAccessAll` アクセス許可はブラウザから OpenSearch Dashboards を許可します。2 つの新しい IAM アクセス権限を追加しなかった場合、403 エラーが表示されます。

こちらのポリシー例では、アカウント内の指定されたコレクションの、データプレーン API へのアクセスと、アカウント内のすべてのコレクションの、OpenSearch Dashboards へのアクセスをユーザーに許可しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": "aoss:APIAccessAll",
            "Resource": "arn:aws:aoss:us-
east-1:111122223333:collection/collection-id"
        },
        {
            "Effect": "Allow",
            "Action": "aoss:DashboardsAccessAll",
            "Resource": "arn:aws:aoss:us-east-1:111122223333:dashboards/default"
        }
    ]
}
```

`aoss:APIAccessAll` と `aoss:DashboardsAccessAll` はどちらも、コレクションリソースに完全な IAM アクセス権限を付与します。Dashboards アクセス権限は、OpenSearch Dashboards アクセスも提供します。各アクセス権限は独立して機能するので、`aoss:APIAccessAll` での明示的な拒否によって開発ツールなどのリソースへの `aoss:DashboardsAccessAll` アクセスがブロックされることはありません。`aoss:DashboardsAccessAll` での拒否についても同じことが言えます。OpenSearch Serverless は、以下のグローバル条件キーをサポートしています。

- `aws:CalledVia`
- `aws:CalledViaAWSService`
- `aws:CalledViaFirst`
- `aws:CalledViaLast`
- `aws:CurrentTime`
- `aws:EpochTime`
- `aws:PrincipalAccount`

- aws:PrincipalArn
- aws:PrincipallsAWSService
- aws:PrincipalOrgID
- aws:PrincipalOrgPaths
- aws:PrincipalType
- aws:PrincipalServiceName
- aws:PrincipalServiceNamesList
- aws:ResourceAccount
- aws:ResourceOrgID
- aws:ResourceOrgPaths
- aws:RequestedRegion
- aws:ResourceTag
- aws:SourceIp
- aws:SourceVpce
- aws:SourceVpc
- aws:userid
- aws:username
- aws:VpcSourceIp

以下は、データプレーン呼び出しのプリンシパルの IAM ポリシーの条件ブロックで aws:SourceIp を使用する例です。

```
"Condition": {
    "IpAddress": {
         "aws:SourceIp": "203.0.113.0"
    }
}
```

以下は、データプレーン呼び出しのプリンシパルの IAM ポリシーの条件ブロックで aws:SourceVpc を使用する例です。

```
"Condition": {
```

```
    "StringEquals": {
        "aws:SourceVpc": "vpc-0fdd2445d8EXAMPLE"
    }
}
```

さらに、以下の OpenSearch Serverless 固有のキーがサポートされています。

- aoss:CollectionId
- aoss:collection

以下は、データプレーン呼び出しのプリンシパルの IAM ポリシーの条件ブロックで aoss:collection を使用する例です。

```
"Condition": {
    "StringLike": {
         "aoss:collection": "log-*"
    }
}
```

OpenSearch API オペレーションの ABAC

アイデンティティベースのポリシーでは、Amazon OpenSearch Serverless データプレーン API へのアクセスの制御にタグを使用できます。以下のポリシーは、コレクションに team:devops タグがある場合に、アタッチされたプリンシパルがデータプレーン API にアクセスすることを許可する例です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "aoss:APIAccessAll",
            "Resource": "arn:aws:aoss:us-
east-1:111122223333:collection/collection-id",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/team": "devops"
                }
```

```
            }
        }
    ]
}
```

以下のポリシーは、コレクションに environment:production タグがある場合に、アタッチされたプリンシパルがデータプレーン API およびダッシュボードにアクセスすることを拒否する例です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "aoss:APIAccessAll",
                "aoss:DashboardsAccessAll"
            ],
            "Resource": "arn:aws:aoss:us-
east-1:111122223333:collection/collection-id"
        }
    ]
}
```

Amazon OpenSearch Serverless は、データプレーン API に対して RequestTag および TagKeys グローバル条件キーをサポートしていません。

IAM Identity Center による Amazon OpenSearch Serverless のサポート

IAM Identity Center による Amazon OpenSearch Serverless のサポート

IAM Identity Center のプリンシパル (ユーザーとグループ) を使用して、Amazon OpenSearch アプリケーションから Amazon OpenSearch Serverless データにアクセスできます。Amazon OpenSearch Serverless の IAM Identity Center に対するサポートを有効にするには、IAM Identity Center の使用を有効にする必要があります。これを行う方法の詳細については、「What is IAM Identity Center?」を参照してください。

> **Note**
>
> IAM Identity Center ユーザーまたはグループを使用して Amazon OpenSearch Serverless コレクションにアクセスするには、OpenSearch UI (アプリケーション) 機能を使用する必要があります。IAM Identity Center 認証情報を使用した OpenSearch Serverless Dashboards への直接アクセスはサポートされていません。詳細については、OpenSearch ユーザーインターフェイスの開始方法」を参照してください。

IAM Identity Center のインスタンスが作成されたら、カスタマーアカウント管理者は Amazon OpenSearch Serverless サービス用の IAM Identity Center アプリケーションを作成する必要があります。これは CreateSecurityConfig を呼び出すことで実行できます。カスタマーアカウント管理者は、リクエストの認可に使用する属性を指定できます。デフォルトで使用される属性は `UserId` と `GroupId.` です。

Amazon OpenSearch Serverless の IAM Identity Center 統合では、次の AWS IAM Identity Center (IAM) アクセス許可を使用します。

- `aoss:CreateSecurityConfig` – IAM Identity Center プロバイダーを作成します
- `aoss:ListSecurityConfig` – 現在のアカウントにおけるすべての IAM Identity Center プロバイダーを一覧表示します。
- `aoss:GetSecurityConfig` – IAM Identity Center のプロバイダー情報を表示します。
- `aoss:UpdateSecurityConfig` – 特定の IAM Identity Center の設定を変更します。
- `aoss:DeleteSecurityConfig` – IAM Identity Center プロバイダーを削除します。

次のアイデンティティベースのアクセスポリシーを使用して、すべての IAM Identity Center 設定を管理できます。

JSON

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Action": [
                "aoss:CreateSecurityConfig",
                "aoss:DeleteSecurityConfig",
```

```
                "aoss:GetSecurityConfig",
                "aoss:UpdateSecurityConfig",
                "aoss:ListSecurityConfigs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Resource 要素はワイルドカードにする必要があります。

IAM Identity Center プロバイダーの作成 (コンソール)

IAM Identity Center プロバイダーを作成して、OpenSearch Application による認証を有効にできます。OpenSearch Dashboards の IAM アイデンティティセンター認証を有効にするには、以下の手順を実行します。

1. [Amazon OpenSearch Service] コンソールにサインインします。
2. 左側のナビゲーションパネルで [サーバーレス] を展開し、[認証] を選択します。
3. [IAM アイデンティティセンター認証] を選択します。
4. [編集] を選択します。
5. [IAM アイデンティティセンターによる認証] の横にあるチェックボックスをオンにします。
6. ドロップダウンメニューから [ユーザーとグループ] 属性キーを選択します。ユーザー属性は、UserName、UserId、Email に基づいてユーザーを認可するために使用されます。グループ属性は、GroupName と GroupId に基づいてユーザーを認証するために使用されます。
7. [IAM アイデンティティセンター] インスタンスを選択します。
8. [保存] を選択します。

IAM アイデンティティセンターのプロバイダーの作成 (AWS CLI)

AWS Command Line Interface (AWS CLI) を使用して IAM Identity Center プロバイダーを作成するには、次のコマンドを使用します。

```
aws opensearchserverless create-security-config \
--region us-east-2 \
--name "iamidentitycenter-config" \
--description "description" \
--type "iamidentitycenter" \
--iam-identity-center-options '{
    "instanceArn": "arn:aws:sso:::instance/ssoins-99199c99e99ee999",
    "userAttribute": "UserName",
    "groupAttribute": "GroupId"
}'
```

IAM アイデンティティセンターが有効化された後、お客様はユーザーとグループの属性のみ変更できます。

```
aws opensearchserverless update-security-config \
--region us-east-1 \
--id <id_from_list_security_configs> \
--config-version <config_version_from_get_security_config> \
--iam-identity-center-options-updates '{
    "userAttribute": "UserId",
    "groupAttribute": "GroupId"
}'
```

を使用して IAM Identity Center プロバイダーを表示するには AWS Command Line Interface、次のコマンドを使用します。

```
aws opensearchserverless list-security-configs --type iamidentitycenter
```

IAM アイデンティティセンターのプロバイダーの削除

IAM アイデンティティセンターには、プロバイダーのインスタンスが 2 つ用意されています。1 つは組織アカウント用、もう 1 つはメンバーアカウント用です。IAM アイデンティティセンターのインスタンスを変更する必要がある場合は、DeleteSecurityConfig API を使用してセキュリティ設定を削除し、新しい IAM アイデンティティセンターのインスタンスを使用して新しいセキュリティ設定を作成する必要があります。以下のコマンドを使用して、IAM アイデンティティセンターのプロバイダーを削除できます。

```
aws opensearchserverless delete-security-config \
--region us-east-1 \
--id <id_from_list_security_configs>
```

IAM アイデンティティセンターに対するコレクションデータへのアクセス権の付与

IAM アイデンティティセンターのプロバイダーが有効になった後は、コレクションのデータアクセスポリシーを更新して IAM アイデンティティセンターのプリンシパルを含めることができます。IAM アイデンティティセンターのプリンシパルは、以下の形式で更新する必要があります。

```
[
   {
"Rules":[
       ...
      ],
      "Principal":[
         "iamidentitycenter/<iamidentitycenter-instance-id>/user/<UserName>",
         "iamidentitycenter/<iamidentitycenter-instance-id>/group/<GroupId>"
      ]
   }
]
```

> **Note**
>
> Amazon OpenSearch Serverless は、すべてのお客様のコレクションに対して IAM アイデンティティセンターの 1 つのインスタンスのみをサポートし、1 人のユーザーに対して最大 100 個のグループをサポートできます。許可されたインスタンスの数を超えて使用しようとすると、データアクセスポリシーの認可処理に不整合が発生し、403 エラーメッセージが表示されます。

アクセス権は、コレクション、インデックス、またはその両方に付与できます。ユーザーごとに異なる権限を持たせたい場合は、複数のルールを作成する必要があります。使用可能なアクセス許可のリストについては、「Amazon OpenSearch Service での Identity and Access Management」を参照してください。アクセスポリシーをフォーマットする方法については、「SAML ID に対するコレクションデータへのアクセス権の付与」を参照してください。

Amazon OpenSearch Serverless の暗号化

保管中の暗号化

ユーザーが作成した各 Amazon OpenSearch Serverless コレクションでは、データへの不正アクセスを防止するための、暗号化を使用したセキュリティ機能により、保管中のデータが保護されます。

保管時の暗号化では AWS Key Management Service 、 (AWS KMS) を使用して暗号化キーを保存および管理します。この暗号化は、256 ビットキーを使用した Advanced Encryption Standard アルゴリズム (AES-256) を使用して実行されます。

トピック

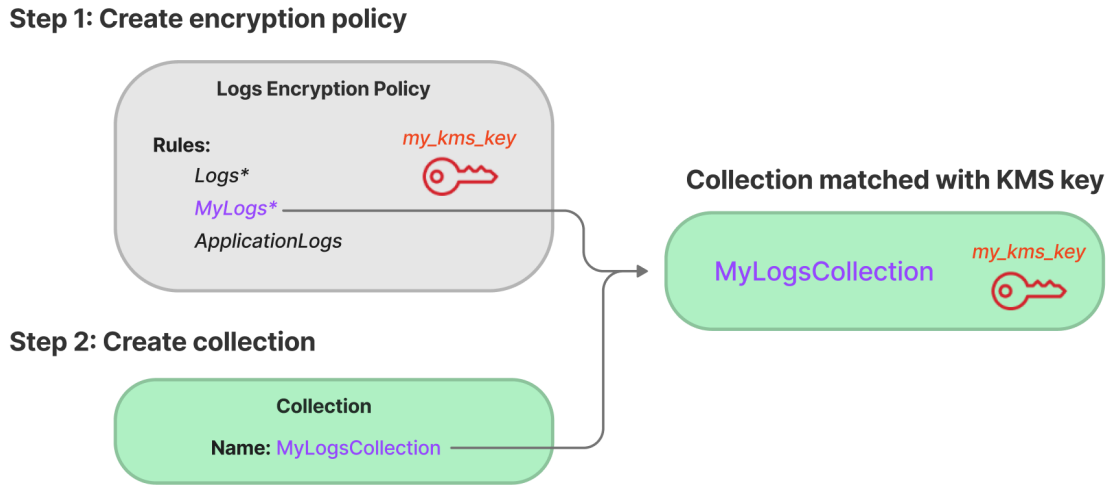
- 暗号化ポリシー
- 考慮事項
- 必要なアクセス許可
- カスタマーマネージドキーのキーポリシー
- OpenSearch Serverless が で許可を使用する方法 AWS KMS
- 暗号化ポリシーの作成 (コンソール)
- 暗号化ポリシーの作成 (AWS CLI)
- 暗号化ポリシーの表示
- 暗号化ポリシーの更新
- 暗号化ポリシーの削除

暗号化ポリシー

暗号化ポリシーを使用すると、新しく作成され特定の名前またはパターンに一致するコレクションに対し、暗号化キーを自動的に割り当てることができ、多くのコレクションを大規模に管理できます。

暗号化ポリシーを作成する際には、プレフィックスでワイルドカードベースのマッチングルール (`MyCollection*` など) を指定するか、コレクション名を 1 つ入力します。次に、その名前またはプレフィックスパターンに一致するコレクションを作成すると、ポリシーとそれに対応する KMS キーが、そのコレクションに対し自動的に割り当てられます。

コレクションを作成するときは、セキュリティポリシーを使用するか、`CreateCollection`リクエスト内で直接行うかの 2 つの方法で AWS KMS キーを指定できます。`CreateCollection` リクエストの一部として AWS KMS キーを指定すると、一致するセキュリティポリシーよりもキーが優先されます。このアプローチにより、必要に応じて特定のコレクションのポリシーベースの暗号化設定を柔軟に上書きできます。



暗号化ポリシーには次の要素が含まれます。

- Rules – 1 つ以上のコレクションマッチングルール。それぞれに次のサブ要素が含まれます。
  - ResourceType – 現在、選択できるオプションは「collection」のみです。暗号化ポリシーは、コレクションリソースにのみ適用されます。
  - Resource – ポリシーが適用される 1 つ以上のコレクション名またはパターンで、形式は collection/<*collection name|pattern*> です。
- AWSOwnedKey – AWS 所有のキーを使用するかどうか。
- KmsARN – AWSOwnedKey に「false」を設定する場合は、KMS キーの Amazon リソースネーム (ARN) を指定して、関連するコレクションを暗号化します。このパラメータを含めると、OpenSearch Serverless は AWSOwnedKey パラメータを無視します。

次のポリシー例では、autopartsinventory という名前で詳細作成されるすべてのコレクションと、「sales」という用語で始まるコレクションに対し、カスタマー管理キーを割り当てます。

```
{
   "Rules":[
      {
         "ResourceType":"collection",
         "Resource":[
            "collection/autopartsinventory",
            "collection/sales*"
         ]
      }
   ],
   "AWSOwnedKey":false,
```

```
   "KmsARN":"arn:aws:kms:us-east-1:123456789012:key/93fd6da4-a317-4c17-
bfe9-382b5d988b36"
}
```

ポリシーがコレクション名と一致していても、リソースパターンにワイルドカード (*) が含まれている場合には、コレクションの作成時にこの自動割り当てを上書きすることが可能です。キーの自動割り当てを上書きすることを選択した場合、OpenSearch Serverless はユーザーに代わり、auto-<*collection-name*> という名前の暗号化ポリシーを作成し、これを対象のコレクションにアタッチします。このポリシーは、初期状態では 1 つのコレクションにのみ適用されますが、他のコレクションを含めるように変更できます。

コンソールを使用してコレクションの作成中に自動暗号化ポリシーの割り当てを上書きするには、コレクションの作成時に暗号化設定をカスタマイズを選択し、任意の AWS KMS キーを選択します。を使用して AWS CLI、`CreateCollection`リクエストに `--encryption-policy-name`パラメータを含めるか、 `kmsKeyArn`を直接指定します。

```
aws opensearchserverless create-collection \
    --name my-collection \
    --type SEARCH \
    --encryption-policy-name my-encryption-policy
```

コレクションと一致しないようにポリシールールを変更しても、関連する KMS キーのコレクションに対する割り当ては、(自動的に) 解除されません。コレクションの暗号化には、常に初期の暗号化キーが使用されます。コレクションに対し異なる暗号化キーを使用した場合には、コレクションを再作成します。

1 つのコレクションに対し複数のポリシーのルールが一致する場合は、より詳細なルールが使用されます。例えば、あるポリシーに `collection/log*` のルールが含まれており、もう 1 つのポリシーには `collection/logSpecial` のルールが含まれている場合には、より詳細である 2 つ目のポリシーの暗号化キーが使用されます。

名前やプレフィックスが既に他のポリシーに含まれている場合、そのポリシーを使用することはできません。異なる暗号化ポリシーで同じリソースパターンを設定しようとすると、OpenSearch Serverless はエラーを表示します。

考慮事項

コレクションの暗号化を設定する際には、次の点を考慮してください。

- すべての Serverless コレクションには、保存時の暗号化が必須です。

- 使用するキーには、カスタマー管理キーと AWS 所有のキーのオプションがあります。カスタマー管理キーを使用する場合は、自動キーローテーションを有効にすることをお勧めします。
- コレクションを作成後、そのコレクションの暗号化キーを変更することはできません。コレクションを初めてセットアップするときに AWS KMS 使用するものを慎重に選択します。
- コレクションに適合できるのは、単一の暗号化ポリシーのみです。
- 一意の KMS キーを持つコレクションは、同じコレクショングループの一部でない限り、OpenSearch Compute Units (OCUs) を他のコレクションと共有することはできません。コレクショングループにない一意のキーを持つ各コレクションには、独自の OCUsが必要です。詳細については、「コレクショングループ」を参照してください。
- 暗号化ポリシーの KMS キーを更新しても、その変更は KMS キーが既に割り当てられている、既存の一致するコレクションには影響しません。
- OpenSearch Serverless は、顧客管理キーにおける許可について明示的なチェックは行いません。データアクセスポリシーを通じてコレクションにアクセスする許可が付与されたユーザーは、関連付けられたキーで暗号化されたデータを、取り込んだりクエリできます。

必要なアクセス許可

OpenSearch Serverless の保管時の暗号化では、次の AWS Identity and Access Management (IAM) アクセス許可を使用します。IAM 条件を指定して、ユーザーを特定のコレクションに制限できます。

- `aoss:CreateSecurityPolicy` – 暗号化ポリシーを作成します。
- `aoss:ListSecurityPolicies` – アタッチされているすべての暗号化ポリシーとコレクションを一覧表示します。
- `aoss:GetSecurityPolicy` – 特定の暗号化ポリシーの詳細を表示します。
- `aoss:UpdateSecurityPolicy` – 暗号化ポリシーを変更します。
- `aoss:DeleteSecurityPolicy` – 暗号化ポリシーを削除します。

以下のアイデンティティベースのアクセスポリシー例では、ユーザーがリソースパターン `collection/application-logs` の暗号化ポリシーを管理する際に必要となる、最小限の許可を付与しています。

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "aoss:CreateSecurityPolicy",
            "aoss:UpdateSecurityPolicy",
            "aoss:DeleteSecurityPolicy",
            "aoss:GetSecurityPolicy"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aoss:collection":"application-logs"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "aoss:ListSecurityPolicies"
         ],
         "Resource":"*"
      }
   ]
}
```

カスタマーマネージドキーのキーポリシー

カスタマーマネージドキーを選択してコレクションを保護する際、OpenSearch Serverless は、選択を行うプリンシパルに代わって KMS キーを使用するための、アクセス許可を取得します。そのプリンシパル、ユーザー、またはロールには、OpenSearch Serverless に必要な KMS キーに対するアクセス許可が付与されている必要があります。れらのアクセス許可は、キーポリシーまたは IAM ポリシーにより付与できます。

OpenSearch Serverless は、自動スケーリング`GenerateDataKey`やソフトウェア更新などのメンテナンスオペレーション中に および `Decrypt` KMS API コールを行います。これらの呼び出しは、一般的なトラフィックパターン外で観察される場合があります。これらの呼び出しは通常のサービスオペレーションの一部であり、アクティブなユーザートラフィックを示すものではありません。

OpenSearch Serverless は、保管中のデータを暗号化する KMS キーにアクセスKMSKeyInaccessibleExceptionできない場合、 をスローします。これは、KMS キーを無効化または削除するか、OpenSearch Serverless がキーを使用できるようにする許可を取り消す場合に発生します。

OpenSearch Serverless には、カスタマー管理について以下のアクセス許可が最低限必要です。

- kms:DescribeKey
- kms:CreateGrant

例えば、次のようになります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Action": "kms:DescribeKey",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Dale"
        },
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "aoss.us-east-1.amazonaws.com"
            }
        }
    },
    {
        "Action": "kms:CreateGrant",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Dale"
        },
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": "aoss.us-east-1.amazonaws.com"
            },
```

```
            "ForAllValues:StringEquals": {
                "kms:GrantOperations": [
                    "Decrypt",
                    "GenerateDataKey"
                ]
            },
            "Bool": {
                "kms:GrantIsForAWSResource": "true"
            }
        }
    }
  ]
}
```

OpenSearch Serverless では、kms:GenerateDataKey および kms:Decrypt のアクセス許可を使用してグラントを作成します。

詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMSでのキーポリシーの使用」を参照してください。

OpenSearch Serverless が で許可を使用する方法 AWS KMS

OpenSearch Serverless がカスタマー管理キーを使用するためには、グラントが必要です。

新しいキーを使用してアカウントに暗号化ポリシーを作成すると、OpenSearch Serverless は CreateGrant リクエストを送信してユーザーに代わって許可を作成します AWS KMS。の権限 AWS KMS は、OpenSearch Serverless に顧客アカウントの KMS キーへのアクセスを許可するために使用されます。

OpenSearch Serverless は、以下の内部オペレーションのために、カスタマー管理キーの使用に関するグラントを必要とします。

- DescribeKey リクエストを に送信 AWS KMS して、指定された対称カスタマーマネージドキー ID が有効であることを確認します。
- オブジェクトを暗号化するためのデータキーを作成するには、GenerateDataKey リクエストを KMS キーに送信します。
- に Decrypt リクエストを送信 AWS KMS して、暗号化されたデータキーを復号して、データの暗号化に使用できるようにします。

グラントへのアクセスの取り消しや、カスタマーマネージドキーに対するサービスのアクセスの取り消しは、いつでもできます。これを行うと、OpenSearch Serverless はカスタマー管理キーによって暗号化されたすべてのデータにアクセスできなくなり、そのデータに依存するすべてのオペレーションが影響を受けます。結果的に、非同期ワークフローで `AccessDeniedException` エラーが発生しその処理に失敗します。

OpenSearch Serverless は、特定の顧客管理キーにおいて、関連付けられたセキュリティポリシーやコレクションが存在しない場合、非同期ワークフローでグラントを廃止します。

暗号化ポリシーの作成 (コンソール)

暗号化ポリシーで、そのポリシーを適用する KMS キーと一連のコレクションパターンを指定します。ポリシーで定義されているパターンのいずれかに一致する新しいコレクションには、その作成時に、対応する KMS キーが割り当てられます。暗号化ポリシーは、コレクションの作成を開始する前に作成することをお勧めします。

OpenSearch Serverless 暗号化ポリシーを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションパネルで [Serverless] (サーバーレス) を展開し、[Encryption policies] (暗号化ポリシー) を選択します。
3. [Create encryption policy] (暗号化ポリシーを作成) を選択します。
4. ポリシーの名前と説明を入力します。
5. [Resources] (リソース) に、この暗号化ポリシーに関する (1 つ以上の) リソースパターンを入力します。現在の AWS アカウント とリージョンにあり、新しく作成され、いずれかのパターンに一致しているコレクションが、自動的にこのポリシーに割り当てられます。例えば、`ApplicationLogs` と (ワイルドカードなしで) 入力し、後にその名前でコレクションを作成した場合、ポリシーおよび対応する KMS キーが、そのコレクションに対し割り当てられます。

   また、`Logs*` などのプレフィックスを指定して、名前が `Logs` で始まる任意の新しいコレクションに対し、ポリシーを割り当てることもできます。ワイルドカードを使用すると、複数のコレクションの暗号化設定を大規模に管理できます。
6. [Encryption] (暗号化) で、使用する KMS キーを選択します。
7. [作成] を選択します。

次のステップ: コレクションを作成する

1 つ以上の暗号化ポリシーを設定したら、それらのポリシーで定義されているルールと一致するコレクションの作成を開始できます。手順については、「the section called “コレクションの作成”」を参照してください。

コレクション作成の [Encryptions] (暗号化) ステップで、暗号化ポリシーで定義されているパターンと一致する名前が入力された場合、OpenSearch Serverless はそれを通知し、対応する KMS キーをコレクションに自動的に割り当てます。リソースパターンにワイルドカード (*) が含まれている場合は、その一致をオーバーライドして独自のキーを選択できます。

暗号化ポリシーの作成 (AWS CLI)

OpenSearch Serverless API オペレーションを使用して暗号化ポリシーを作成する際は、JSON 形式で、リソースパターンと暗号化キーを指定します。CreateSecurityPolicy リクエストは、インラインポリシーと.json ファイルの両方を受け入れます。

暗号化ポリシーは以下の形式になります。このサンプルファイル (my-policy.json) は、将来作成される autopartsinventory という名前のコレクションすべてと、名前が sales で始まるコレクションすべてに一致します。

```
{
   "Rules":[
      {
         "ResourceType":"collection",
         "Resource":[
            "collection/autopartsinventory",
            "collection/sales*"
         ]
      }
   ],
   "AWSOwnedKey":false,
   "KmsARN":"arn:aws:kms:us-east-1:123456789012:key/93fd6da4-a317-4c17-
bfe9-382b5d988b36"
}
```

サービス所有のキーを使用するには、AWSOwnedKey に true を設定します。

```
{
   "Rules":[
      {
         "ResourceType":"collection",
```

```
         "Resource":[
            "collection/autopartsinventory",
            "collection/sales*"
         ]
      }
   ],
   "AWSOwnedKey":true
}
```

次のリクエストは暗号化ポリシーを作成します。

```
aws opensearchserverless create-security-policy \
    --name sales-inventory \
    --type encryption \
    --policy file://my-policy.json
```

その後、CreateCollection API オペレーションを使用して、リソースパターンの 1 つと一致するコレクションを 1 つまたは複数作成します。

暗号化ポリシーの表示

コレクションを作成する際、コレクション名と一致するリソースパターンを持つポリシーを確認するために、アカウント内の既存の暗号化ポリシーを、プレビューする必要が生じることがあります。次の ListSecurityPolicies リクエストでは、アカウント内のすべての暗号化ポリシーが一覧表示されます。

```
aws opensearchserverless list-security-policies --type encryption
```

このリクエストは、すべての設定済みの暗号化ポリシーに関する情報を返します。policy 要素の内容を使用して、ポリシーで定義されているパターンルールを表示します。

```
{
   "securityPolicyDetails": [
      {
         "createdDate": 1663693217826,
         "description": "Sample encryption policy",
         "lastModifiedDate": 1663693217826,
         "name": "my-policy",
         "policy": "{\"Rules\":[{\"ResourceType\":\"collection\",\"Resource\":
[\"collection/autopartsinventory\",\"collection/sales*\"]}],\"AWSOwnedKey\":true}",
         "policyVersion": "MTY2MzY5MzIxNzgyNl8x",
         "type": "encryption"
```

```
      }
   ]
}
```

KMS キーなど、特定のポリシーに関する詳細情報を表示するには、GetSecurityPolicy コマンドを使用します。

暗号化ポリシーの更新

暗号化ポリシー内で KMS キーを更新する場合、その変更は、設定された名前またはパターンと一致する、新規作成のコレクションだけに適用されます。KMS キーが既に割り当て済みの、既存のコレクションには影響は与えません。

同様なことが、ポリシーの一致ルールにも当てはまります。ルールを追加、変更、または削除した場合、その変更は新しく作成されたコレクションにのみ適用されます。ポリシーのルールを変更したために、コレクション名と一致しなくなったとしても、既存のコレクションに割り当てられた KMS キーは失われません。

OpenSearch Serverless コンソールで暗号化ポリシーを更新するには、[Encryption policies] (暗号化ポリシー) を選択し、変更するポリシーを選択した後に、[Edit] (編集) を選択します。変更を行ってから、[保存] を選択します。

OpenSearch Serverless API を使用して暗号化ポリシーを更新する場合は、UpdateSecurityPolicy オペレーションを使用します。次のリクエストでは、新しいポリシー JSON ドキュメントを使用して、暗号化ポリシーを更新します。

```
aws opensearchserverless update-security-policy \
    --name sales-inventory \
    --type encryption \
    --policy-version 2 \
    --policy file://my-new-policy.json
```

暗号化ポリシーの削除

暗号化ポリシーを削除しても、そのポリシーで定義されている KMS キーを現在使用しているコレクションに影響はありません。OpenSearch Serverless コンソールでポリシーを削除するには、ポリシーを選択し、[Delete] (削除) を選択します。

また、DeleteSecurityPolicy オペレーションを使用することもできます。

```
aws opensearchserverless delete-security-policy --name my-policy --type encryption
```

転送中の暗号化

OpenSearch Serverless では、コレクション内のすべてのパスは、業界標準の AES-256 暗号を使用した Transport Layer Security 1.2 (TLS) を使って、転送中に暗号化されます。Opensearch のすべての API と Dashboards へのアクセスも、TLS 1.2 を経由します。TLS は、ネットワーク上でやり取りされる情報の暗号化に使用される、業界標準の暗号化プロトコルのセットです。

Amazon OpenSearch Serverless でのネットワークアクセス

Amazon OpenSearch Serverless コレクションでは、ネットワーク設定によって、そのコレクションに対しインターネット経由でパブリックネットワークからのアクセスが可能か、プライベートでアクセスする必要があるかが決まります。

プライベートアクセスは、次のいずれかまたは両方に適用できます。

- OpenSearch Serverless マネージド VPC エンドポイント
- Amazon Bedrock AWS のサービス などでサポート

コレクションでは、OpenSearch のエンドポイントと、対応する OpenSearch Dashboards のエンドポイントで、個別のネットワークアクセス設定が行えます。

ネットワークアクセスは、さまざまなソースネットワークからのアクセスを、分離して許可するためのメカニズムです。例えば、コレクションの OpenSearch Dashboards エンドポイントはパブリックアクセスが可能で、OpenSearch API エンドポイントはパブリックアクセスが可能でない場合、パブリックネットワークから接続するユーザーは、Dashboards を経由しなければコレクションデータにアクセスできません。パブリックネットワークから直接 OpenSearch API を呼び出そうとしても、ブロックされます。ソースからリソースタイプへのこのような配列には、ネットワーク設定を使用できます。Amazon OpenSearch Serverless は、IPv4 と IPv6 の両方の接続をサポートしています。

トピック

- ネットワークポリシー
- 考慮事項
- ネットワークポリシーの設定に必要なアクセス許可
- ポリシーの優先順位
- ネットワークポリシーの作成 (コンソール)
- ネットワークポリシーの作成 (AWS CLI)
- ネットワークポリシーの表示

- ネットワークポリシーの更新
- ネットワークポリシーの削除

ネットワークポリシー

ネットワークポリシーでは、その中でルールを定義し、そのルールに一致するコレクションに対し、ネットワークアクセス設定を自動的に割り当てることにより、多数のコレクションを大規模に管理できます。

ネットワークポリシーでは、一連のルールを指定します。これらのルールにより、コレクションのエンドポイントと OpenSearch Dashboards のエンドポイントへのアクセス許可を定義します。各ルールは、アクセスタイプ (パブリックまたはプライベート) とリソースタイプ (コレクション、および/または OpenSearch Dashboards のエンドポイント) で構成されます。リソースタイプ (`collection` および `dashboard`) ごとに、一連のルールを指定し、ポリシーを適用する対象となるコレクションを定義します。

このポリシー例では、最初のルールで、語句 `marketing*` で始まるすべてのコレクションについて、コレクションエンドポイントとダッシュボードエンドポイントの両方で、VPC エンドポイントアクセスを使用するように指定しています。また、Amazon Bedrock アクセスも指定します。

> **Note**
>
> Amazon Bedrock AWS のサービス などの へのプライベートアクセスは、コレクションの OpenSearch エンドポイントにのみ適用され、OpenSearch Dashboards エンドポイントには適用されません。`ResourceType` が であってもdashboard、OpenSearch Dashboards へのアクセスを許可 AWS のサービス することはできません。

2 番目のルールでは、`finance` コレクションに対してパブリックアクセスを指定していますが、これには、コレクションエンドポイントのみが使用されます (ダッシュボードからのアクセスは許可されません)。

```
[
   {
      "Description":"Marketing access",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
```

```
                "collection/marketing*"
            ]
         },
         {
            "ResourceType":"dashboard",
            "Resource":[
               "collection/marketing*"
            ]
         }
      ],
      "AllowFromPublic":false,
      "SourceVPCEs":[
         "vpce-050f79086ee71ac05"
      ],
      "SourceServices":[
         "bedrock.amazonaws.com"
      ],
   },
   {
      "Description":"Sales access",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/finance"
            ]
         }
      ],
      "AllowFromPublic":true
   }
]
```

このポリシーでは、OpenSearch Dashboards からのパブリックアクセスは、「finance」で始まるコレクションに対してのみ許可しています。OpenSearch API に直接アクセスする試みは、すべて失敗します。

```
[
  {
    "Description": "Dashboards access",
    "Rules": [
      {
        "ResourceType": "dashboard",
        "Resource": [
```

```
          "collection/finance*"
        ]
      }
    ],
    "AllowFromPublic": true
  }
]
```

ネットワークポリシーは、将来作成するコレクションだけでなく、既存のコレクションにも適用が可能です。コレクションを作成した後に、そのコレクション名と一致するルールを含むネットワークポリシーを作成できます。コレクションを作成する前に、必ずしもネットワークポリシーを用意しておく必要はありません。

考慮事項

コレクションのためにネットワークアクセスを設定する際は、以下の点を考慮してください。

- コレクションのために VPC エンドポイントアクセスを設定する場合は、最初に、少なくとも 1 つの OpenSearch Serverless マネージド VPC エンドポイントを作成しておく必要があります。
- へのプライベートアクセスは、OpenSearch Dashboards エンドポイントではなく、コレクションの OpenSearch エンドポイント AWS のサービス にのみ適用されます。`ResourceType` が であってもdashboard、OpenSearch Dashboards へのアクセスを許可 AWS のサービス することはできません。
- パブリックネットワークからアクセスできるコレクションに対しては、OpenSearch Serverless マネージド VPC エンドポイントとすべての AWS のサービスからもアクセスできます。パブリックネットワークアクセスが有効になっている場合でも、データアクセスポリシーは、コレクションのデータの読み取りと書き込みを行えるユーザーを制御します。ネットワークアクセスは、収集エンドポイントに到達できるネットワークのみを決定します。データアクセスポリシーは、データに対してオペレーションを実行できるプリンシパルを決定します。詳細については、「the section called “データアクセスコントロール”」を参照してください。
- 単一のコレクションに対して、複数のネットワークポリシーを適用できます。詳細については、「the section called “ポリシーの優先順位”」を参照してください。

ネットワークポリシーの設定に必要なアクセス許可

OpenSearch Serverless のネットワークアクセスでは、次の AWS Identity and Access Management (IAM) アクセス許可を使用します。IAM 条件を指定しすることで、特定のコレクションに関連付けられたネットワークポリシーのみをユーザーが使用するように制限できます。

- aoss:CreateSecurityPolicy – ネットワークアクセスポリシーを作成します。
- aoss:ListSecurityPolicies – 現在のアカウントにある、すべてのネットワークポリシーを一覧表示します。
- aoss:GetSecurityPolicy – ネットワークアクセスポリシーの設定を表示します。
- aoss:UpdateSecurityPolicy – 特定のネットワークアクセスポリシーを変更し、VPC ID またはパブリックアクセスの指定を変更します。
- aoss:DeleteSecurityPolicy – ネットワークアクセスポリシーを (すべてのコレクションからデタッチした後に) 削除します。

次の ID ベースのアクセスポリシーにより、ユーザーはすべてのネットワークポリシーを表示し、リソースパターン collection/application-logs を含むポリシーの更新が行えるようになります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aoss:UpdateSecurityPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aoss:collection": "application-logs"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "aoss:ListSecurityPolicies",
                "aoss:GetSecurityPolicy"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

> **Note**
>
> さらに、OpenSearch Serverless には、コレクションリソースの `aoss:APIAccessAll` および `aoss:DashboardsAccessAll` アクセス許可が必要です。詳細については、「the section called “OpenSearch API オペレーションの使用”」を参照してください。

ポリシーの優先順位

ポリシー内またはポリシー間に、複数のネットワークポリシールールが同時に存在する場合があります。この場合、パブリックアクセスを指定するルールとプライベートアクセスを指定するルールでは、(これら両方のルールに共通して適用されるコレクションにおいて) 前者のルールが優先されます。

例えば、次のポリシーでは、2 種類のルールが `finance` コレクションにネットワークアクセスを割り当てています。一方のルールでは VPC アクセスを、もう一方のルールではパブリックアクセスをそれぞれ指定しています。この場合、finance コレクションでのみ、パブリックアクセスは VPC アクセスをオーバーライドします (両方のルールに含まれるのがこのコレクションであるため)。つまり、finance コレクションにアクセスできるのは、パブリックネットワークからとなります。sales コレクションには、指定されたエンドポイントからの VPC アクセスが許可されます。

```
[
   {
      "Description":"Rule 1",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/sales",
               "collection/finance"
            ]
         }
      ],
      "AllowFromPublic":false,
      "SourceVPCEs":[
         "vpce-050f79086ee71ac05"
      ]
```

```
   },
   {
      "Description":"Rule 2",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/finance"
            ]
         }
      ],
      "AllowFromPublic":true
   }
]
```

異なるルールの複数の VPC エンドポイントが 1 つのコレクションに適用されている場合、これらのルールは追加的に認識され、指定されたすべてのエンドポイントから、そのコレクションへのアクセスが可能になります。`AllowFromPublic` を `true` に設定し、さらに 1 つ以上の `SourceVPCEs` または `SourceServices` を指定した場合には、OpenSearch Serverless により VPC エンドポイントは無視され、関連するコレクションにはパブリックアクセスが提供されます。

ネットワークポリシーの作成 (コンソール)

ネットワークポリシーは、将来作成するコレクションだけでなく、既存のコレクションにも適用が可能です。コレクションの作成を開始する前に、ネットワークポリシーを作成しておくことをお勧めします。

OpenSearch Serverless のネットワークポリシーを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションパネルで [Serverless] (サーバーレス) を展開し、[Network policie] (ネットワークポリシー) を選択します。
3. [Create network policy] (ネットワークポリシーを作成) を選択します。
4. ポリシーの名前と説明を入力します。
5. 1 つ以上のルールを指定します。これらのルールは、OpenSearch Serverless コレクションと、その OpenSearch Dashboards エンドポイントへのアクセス許可を定義しています。

   各ルールには以下の要素が含まれます。

Element	説明
ルール名	ルールの内容をわかりやすく示した名前です。例えば、「VPC access for marketing team」などとします。
アクセスタイプ	パブリックアクセスまたはプライベートアクセスを選択します。続いて、次のいずれかを選択します。 • [アクセス用の VPC エンドポイント] – 1 つ以上の OpenSearch Serverless マネージド VPC エンドポイントを指定します。 • AWS のサービス プライベートアクセス — サポートされている を 1 つ以上選択します AWS のサービス。

Element	説明
リソースタイプ	アクセスを提供する対象を、OpenSearch エンドポイント (OpenSearch API への呼び出しを許可) にするか、OpenSearch Dashboards (OpenSearch プラグインのビジュアライゼーションとユーザーインターフェースへのアクセスを許可) にするか、あるいはその両方を対象にするかを選択します。 **Note** AWS のサービス プライベートアクセスは、コレクションの OpenSearch エンドポイントにのみ適用され、OpenSearch Dashboards エンドポイントには適用されません。[OpenSearch Dashboards] を選択した場合でも、 AWS のサービスにはエンドポイントアクセスのみ許可されます。

選択したリソースタイプごとに、ポリシー設定を適用する既存のコレクションを選択すること、および/または 1 つ以上のリソースパターンを作成することができます。リソースパターンはプレフィックスとワイルドカード (*) で構成され、ポリシー設定の適用対象となるコレクションを定義します。

例えば、`Marketing*` というパターンを含めると、名前が「Marketing」で始まる新規または既存のコレクションには、このポリシーのネットワーク設定が自動的に適用されます。ワイルドカード (*) によって、既存のコレクションや将来作成されるコレクションすべてにポリシーが適用されます。

また、ワイルドカードなしで、将来作成されるコレクションの名前 (例: `Finance`) を指定することもできます。OpenSearch Serverless は、これと同じ名前を使用して新しく作成されたすべてのコレクションに、同じポリシー設定を適用します。

6. ポリシー設定を確認し、変更点がない場合は、[Create] (作成) を選択します。

ネットワークポリシーの作成 (AWS CLI)

OpenSearch Serverless API オペレーションを使用してネットワークポリシーを作成するには、JSON 形式でルールを指定します。CreateSecurityPolicy リクエストは、インラインポリシーと.json ファイルの両方を受け入れます。すべてのコレクションとパターンは、`collection/<collection name|pattern>` の形式にする必要があります。

> **Note**
>
> リソースタイプ `dashboards` では OpenSearch Dashboards へのアクセスのみが許可されます。ただし、OpenSearch Dashboards が機能するためには、同じソースからコレクションに対するアクセスも許可する必要があります。例については、以下に示す 2 番目のポリシーを参照してください。

プライベートアクセスを指定するには、次の要素の 1 つまたは両方を含めます。

- `SourceVPCEs` – 1 つ以上の OpenSearch Serverless マネージド VPC エンドポイントを指定します。
- `SourceServices` – サポートされている 1 つ以上の識別子を指定します AWS のサービス。現在、次のサービス識別子がサポートされています。
  - `bedrock.amazonaws.com` – Amazon Bedrock

次のネットワークポリシー例では、VPC エンドポイントと Amazon Bedrock へのプライベートアクセスを、`log*` プレフィックスで始まるコレクションのコレクションエンドポイントにのみ提供しています。認証されたユーザーは OpenSearch Dashboards にサインインできず、コレクションエンドポイントにはプログラムでしかアクセスできません。

```
[
   {
      "Description":"Private access for log collections",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/log*"
            ]
         }
      ],
```

```
      "AllowFromPublic":false,
      "SourceVPCEs":[
         "vpce-050f79086ee71ac05"
      ],
      "SourceServices":[
         "bedrock.amazonaws.com"
      ],
   }
]
```

以下のポリシーでは、finance という名前の単一のコレクションに対して、OpenSearch エンドポイントと OpenSearch Dashboards の両方でのパブリックアクセスを提供します。対象のコレクションが存在しない場合、ネットワーク設定は、そのコレクションが作成された時点で適用されます。

```
[
   {
      "Description":"Public access for finance collection",
      "Rules":[
         {
            "ResourceType":"dashboard",
            "Resource":[
               "collection/finance"
            ]
         },
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/finance"
            ]
         }
      ],
      "AllowFromPublic":true
   }
]
```

次のリクエストにより、上記のネットワークポリシーが作成されます。

```
aws opensearchserverless create-security-policy \
    --name sales-inventory \
    --type network \
    --policy "[{\"Description\":\"Public access for finance collection\",\"Rules
\":[{\"ResourceType\":\"dashboard\",\"Resource\":[\"collection\/finance\"]},
```

```
{\"ResourceType\":\"collection\",\"Resource\":[\"collection\/finance\"]}],
\"AllowFromPublic\":true}]"
```

このポリシーを JSON ファイルで指定するには、`--policy file://my-policy.json` の形式を使用します。

ネットワークポリシーの表示

コレクションを作成する前に、アカウント内の既存のネットワークポリシーをプレビューして、コレクション名と一致するリソースパターンが含まれているポリシーを確認しておきたい場合があります。次の ListSecurityPolicies リクエストにより、アカウント内のすべてのネットワークポリシーが一覧表示されます。

```
aws opensearchserverless list-security-policies --type network
```

このリクエストは、設定されているすべてのネットワークポリシーに関する情報を返します。1 つの特定のポリシーで定義されているパターンルールを表示するには、レスポンスの `securityPolicySummaries` 要素の内容でポリシー情報を探します。このポリシーの `name` と `type` を書き留め、GetSecurityPolicy リクエストでこれらのプロパティを使用して、次のポリシーの詳細を含むレスポンスを受信します。

```
{
    "securityPolicyDetail": [
        {
            "type": "network",
            "name": "my-policy",
            "policyVersion": "MTY2MzY5MTY1MDA3Ml8x",
            "policy": "[{\"Description\":\"My network policy rule\",\"Rules\":
[{\"ResourceType\":\"dashboard\",\"Resource\":[\"collection/*\"]}],\"AllowFromPublic
\":true}]",
            "createdDate": 1663691650072,
            "lastModifiedDate": 1663691650072
        }
    ]
}
```

特定のポリシーの詳細情報を表示するには、GetSecurityPolicy コマンドを使用します。

ネットワークポリシーの更新

ネットワークの VPC エンドポイントまたはパブリックアクセスの指定を変更すると、関連するすべてのコレクションが影響を受けます。OpenSearch Serverless コンソールでネットワークポリシーを更新するには、[Network policies] (ネットワークポリシー) を展開してから変更するポリシーを選択した後、[Edit] (編集) を選択します。変更を行ってから、[保存] を選択します。

OpenSearch Serverless API を使用してネットワークポリシーを更新するには、UpdateSecurityPolicy コマンドを使用します。リクエストには、ポリシーのバージョンを含める必要があります。ポリシーのバージョンは、`ListSecurityPolicies` または `GetSecurityPolicy` コマンドを使用して取得できます。最新のポリシーバージョンを含めると、他のユーザーによる変更を意図せず上書きしてしまうことがなくなります。

次のリクエストは、ネットワークポリシーを新しいポリシーの JSON ドキュメントで更新します。

```
aws opensearchserverless update-security-policy \
    --name sales-inventory \
    --type network \
    --policy-version MTY2MzY5MTY1MDA3Ml8x \
    --policy file://my-new-policy.json
```

ネットワークポリシーの削除

ネットワークポリシーを削除する前に、そのポリシーをすべてのコレクションからデタッチする必要があります。OpenSearch Serverless コンソールでポリシーを削除するには、ポリシーを選択し、[Delete] (削除) を選択します。

DeleteSecurityPolicy コマンドを使用することもできます。

```
aws opensearchserverless delete-security-policy --name my-policy --type network
```

Amazon OpenSearch Serverless での FIPS コンプライアンス

Amazon OpenSearch Serverless は、連邦情報処理規格 (Federal Information Processing Standards/FIPS) 140-2 をサポートしています。これは、機密情報を保護する暗号モジュールのセキュリティ要件を規定する、米国およびカナダ政府の基準です。OpenSearch Serverless を使用して FIPS 対応エンドポイントに接続すると、FIPS 検証済み暗号化ライブラリを使用して暗号化オペレーションが行われます。

OpenSearch Serverless FIPS エンドポイントは、FIPS AWS リージョン がサポートされている で使用できます。これらのエンドポイントは、すべての通信に TLS 1.2 以降と FIPS 検証済み暗号化アルゴリズムを使用します。詳細については、「AWS Verified Access ユーザーガイド」の「FIPS 準拠」を参照してください。

トピック

- OpenSearch Serverless での FIPS エンドポイントの使用
- AWS SDKs
- VPC エンドポイントのセキュリティグループを設定する
- FIPS VPC エンドポイントを使用する
- FIPS コンプライアンスを検証する
- プライベートホストゾーンの FIPS エンドポイント接続の問題を解決する

OpenSearch Serverless での FIPS エンドポイントの使用

FIPS がサポートされている AWS リージョン では、OpenSearch Serverless コレクションには、標準エンドポイントと FIPS 準拠エンドポイントの両方からアクセスできます。FIPS 準拠のバリアントは、OpenSearch Serverless NextGen と Classic の両方のコレクションエンドポイントで使用できます。詳細については、「AWS Verified Access ユーザーガイド」の「FIPS 準拠」を参照してください。

次の例では、 *collection-id*、*account-id*、*region* をコレクション ID、 AWS アカウント ID、および Region に置き換えます。

コレクションあたりの NextGen エンドポイント

- 標準 – **https://*collection-id*.aoss.*region*.on.aws**
- FIPS 準拠 – **https://*collection-id*.aoss-fips.*region*.on.aws**

アカウントあたりの NextGen エンドポイント

- 標準 – **https://*account-id*.aoss.*region*.on.aws**
- FIPS 準拠 – **https://*account-id*.aoss-fips.*region*.on.aws**

コレクションごとのクラシックエンドポイント

- 標準 – **https://*collection-id*.*region*.aoss.amazonaws.com**
- FIPS 準拠 – **https://*collection-id*.*region*.aoss-fips.amazonaws.com**

NextGen FIPS エンドポイントは、非 FIPS エンドポイントと同じ VPC アクセス AWS PrivateLink の標準を使用します。詳細については、「the section called “データプレーン VPC エンドポイント”」を参照してください。

> **Note**
>
> FIPS 対応リージョンでは、標準エンドポイントと FIPS 準拠エンドポイントの両方が FIPS 準拠の暗号化を提供します。FIPS 固有のエンドポイントは、名前に FIPS が使用されているエンドポイントの使用を特に義務付けるコンプライアンス要件を満たすのに役立ちます。

AWS SDKs

AWS SDKs を使用する場合は、クライアントの作成時に FIPS エンドポイントを指定できます。次の例では、*collection_id* と *AWS #####* をご自身のコレクション ID とその AWS リージョンに置き換えます。

```
# Python SDK example
from opensearchpy import OpenSearch, RequestsHttpConnection, AWSV4SignerAuth
import boto3
host = '"https://collection_id.AWS #####.aoss-fips.amazonaws.com"
region = 'us-west-2'
service = 'aoss'
credentials = boto3.Session().get_credentials()
auth = AWSV4SignerAuth(credentials, region, service)
client = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = auth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection,
    pool_maxsize = 20
)
```

VPC エンドポイントのセキュリティグループを設定する

FIPS 準拠の Amazon VPC (VPC) エンドポイントと適切に通信するには、セキュリティグループを作成または変更して、OpenSearch Serverless にアクセスする必要がある VPC 内のリソースからのインバウンド HTTPS トラフィック (TCP ポート 443) を許可します。次に、このセキュリティグループを VPC エンドポイントに関連付けます。これは作成時に行うか、作成後にエンドポイントを変更して行います。詳細については、「Amazon VPC ユーザーガイド」の「セキュリティグループの作成」を参照してください。

FIPS VPC エンドポイントを使用する

FIPS 準拠の VPC エンドポイントを作成したら、それを使用して VPC 内のリソースから OpenSearch Serverless にアクセスできます。API オペレーションにエンドポイントを使用するには、「OpenSearch Serverless での FIPS エンドポイントの使用」セクションの説明に従って、リージョン FIPS エンドポイントを使用するように SDK を設定します。OpenSearch Dashboards にアクセスするには、コレクション固有の Dashboards URL を使用します。この URL は、VPC 内からアクセスすると、FIPS 準拠の VPC エンドポイントを自動的にルーティングします。詳細については、「Amazon OpenSearch Service での OpenSearch Dashboards の使用」を参照してください。

FIPS コンプライアンスを検証する

OpenSearch Serverless への接続で FIPS 準拠の暗号化が使用されていることを確認するには、 AWS CloudTrail を使用して OpenSearch Serverless に対する API コールをモニタリングします。API コールについて、CloudTrail ログの `eventSource` フィールドに `aoss-fips.amazonaws.com` が表示されていることを確認します。

OpenSearch Dashboards にアクセスするには、ブラウザ開発者ツールを使用して TLS 接続の詳細を検査し、FIPS 準拠の暗号スイートが使用されていることを確認します。

プライベートホストゾーンの FIPS エンドポイント接続の問題を解決する

FIPS エンドポイントは、パブリックアクセスを持つ Amazon OpenSearch Serverless コレクションと連携します。新しく作成された VPC エンドポイントを使用する新しく作成された VPC コレクションの場合、FIPS エンドポイントは期待どおりに機能します。他の VPC コレクションでは、FIPS エンドポイントが正しく動作するように手動で設定する必要がある場合があります。

Amazon Route 53 で FIPS プライベートホストゾーンを設定するには

1. Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。
2. ホストゾーンを確認します。

a. コレクションがあるホストゾーンを見つけ AWS リージョン ます。
b. ホストゾーンの命名パターンを確認します。
   - 非 FIPS 形式: *region*.aoss.amazonaws.com。
   - FIPS 形式: *region*.aoss-fips.amazonaws.com。
c. すべてのホストゾーンの [タイプ] が [プライベートホストゾーン] に設定されていることを確認します。

3. FIPS プライベートホストゾーンがない場合:
   a. 対応する非 FIPS プライベートホストゾーンを選択します。
   b. [関連付けられた VPC] 情報をコピーします。例: vpc-1234567890abcdef0 | us-east-2。
   c. ワイルドカードドメインレコードを見つけます。例: *.us-east-2.aoss.amazonaws.com。
   d. [値/トラフィックのルーティング先] 情報をコピーします。例:uoc1c1qsw7poexampleewjeno1pte3rw.3ym756xh7yj.aoss.searchservices.aws
4. FIPS プライベートホストゾーンを作成します。
   a. FIPS 形式で新しいプライベートホストゾーンを作成します。例: us-east-2.aoss-fips.amazonaws.com。
   b. [関連付けられた VPC] には、非 FIPS プライベートホストゾーンからコピーした VPC 情報を入力します。
5. 以下の設定でルールを追加します。
   a. レコード名: *
   b. レコードタイプ: CNAME
   c. 値: 先ほどコピーした [値/トラフィックのルーティング先] 情報を入力します。

一般的な問題

FIPS 準拠の VPC エンドポイントで接続の問題が発生した場合は、次の情報を使用して問題を解決します。

- DNS 解決の失敗 – VPC 内の FIPS エンドポイントドメイン名を解決できない
- 接続タイムアウト – FIPS エンドポイントへのリクエストがタイムアウトする

- アクセス拒否エラー – FIPS エンドポイントの使用時に認証または認可が失敗する
- VPC のみのコレクションにプライベートホストゾーンレコードがない

FIPS エンドポイント接続のトラブルシューティングを行うには

1. プライベートホストゾーンの設定を確認します。

   a. FIPS エンドポイントドメイン (`*.region.aoss-fips.amazonaws.com`) にプライベートホストゾーンが存在することを確認します。

   b. プライベートホストゾーンが正しい VPC に関連付けられていることを確認します。

   詳細については、「Amazon Route 53 デベロッパーガイド」の「Private hosted zones」および「AWS PrivateLink ガイド」の「Manage DNS names」を参照してください。

2. DNS 解決をテストします。

   a. VPC の EC2 インスタンスに接続します。

   b. 次のコマンドを実行します。

   ```
   nslookup collection-id.region.aoss-fips.amazonaws.com
   ```

   c. レスポンスに VPC エンドポイントのプライベート IP アドレスが含まれていることを確認してください。

   詳細については、「Amazon VPC ユーザーガイド」の「Endpoint policies」および「DNS attributes」を参照してください。

3. セキュリティグループの設定を確認します。

   a. VPC エンドポイントにアタッチされたセキュリティグループが、リソースからの HTTPS トラフィック (ポート 443) を許可していることを確認します。

   b. リソースのセキュリティグループが VPC エンドポイントへのアウトバウンドトラフィックを許可していることを確認します。

   詳細については、「AWS PrivateLink ガイド」の「Endpoint policies」、および「Amazon VPC ユーザーガイド」の「Security groups」を参照してください。

4. ネットワーク ACL 設定を確認します。

- ネットワーク ACL によってリソースと VPC エンドポイント間のトラフィックが許可されていることを確認します。

  詳細については、「Amazon VPC ユーザーガイド」の「ネットワーク ACL」を参照してください。

5. エンドポイントポリシーを確認します。

- VPC エンドポイントポリシーが OpenSearch Serverless リソースで必要なアクションを許可していることを確認します。

  詳細については、「AWS PrivateLink ガイド」の「VPC endpoint permissions required」および「Endpoints policies」を参照してください。

> **Tip**
>
> VPC でカスタム DNS リゾルバーを使用する場合は、`*.amazonaws.com`ドメインのリクエストを AWS サーバーに転送するように設定します。

Amazon OpenSearch Serverless のデータアクセスコントロール

Amazon OpenSearch Serverless のデータアクセスコントロールを使用すると、アクセスメカニズムやネットワークソースに関係なく、ユーザーがコレクションやインデックスにアクセスできるようにすることができます。IAM ロールと SAML アイデンティティへのアクセスを提供できます。

アクセス許可は、コレクションとインデックスリソースに適用されるデータアクセスポリシーを通じて管理します。データアクセスポリシーは、特定のパターンに一致するコレクションとインデックスにアクセス許可を自動的に割り当てることにより、大規模なコレクションを管理するのに役立ちます。1 つのリソースに複数のデータアクセスポリシーを適用できます。OpenSearch Dashboards URL にアクセスするには、コレクションのデータアクセスポリシーが必要になります。

トピック

- データアクセスポリシーと IAM ポリシーの比較
- データアクセスポリシーの設定に必要な IAM アクセス許可
- ポリシー構文
- サポートされているポリシーのアクセス許可

- OpenSearch Dashboards のサンプルデータセット
- データアクセスポリシーの作成 (コンソール)
- データアクセスポリシーの作成 (AWS CLI)
- データアクセスポリシーの表示
- データアクセスポリシーの更新
- データアクセスポリシーの削除
- クロスアカウントデータアクセス

データアクセスポリシーと IAM ポリシーの比較

データアクセスポリシーは、 AWS Identity and Access Management (IAM) ポリシーとは論理的に分離されます。IAM アクセス許可は、`CreateCollection` や `ListAccessPolicies` などのサーバーレス API オペレーションへのアクセスを制御します。データアクセスポリシーは、OpenSearch Serverless がサポートする `PUT <index>` や `GET _cat/indices` などの OpenSearch オペレーションへのアクセスを制御します。

`aoss:CreateAccessPolicy` や `aoss:GetAccessPolicy` (次のセクションで説明) などのデータアクセスポリシーの API オペレーションへのアクセスを制御する IAM アクセス許可は、データアクセスポリシーで指定されているアクセス許可には影響しません。

例えば、IAM ポリシーによってユーザーによる `collection-a` に対するデータアクセスポリシーの作成は拒否されているが、すべてのコレクション (*) に対するデータアクセスポリシーの作成は許可されているとします。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "aoss:CreateAccessPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "collection-a"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "aoss:CreateAccessPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

ユーザーがすべてのコレクション (collection/* または index/*/*) に特定のアクセス許可を付与するデータアクセスポリシーを作成した場合、ポリシーはコレクション A を含むすべてのコレクションに適用されます。

> **Important**
>
> OpenSearch Serverless コレクション内のデータにアクセスするためには、データアクセスポリシー内で許可が付与されているだけでは不十分です。関連付けられたプリンシパルにも、IAM アクセス許可 aoss:APIAccessAll および aoss:DashboardsAccessAll に対するアクセスが付与されている必要があります。いずれの許可もコレクションリソースに対するフルアクセスを付与し、Dashboards 許可も OpenSearch Dashboards に対するアクセスを提供します。プリンシパルがこれらの両方の IAM 許可を持っていない場合、コレクションにリクエストを送信しようとすると 403 エラーが表示されます。詳細については、「the section called “OpenSearch API オペレーションの使用”」を参照してください。

データアクセスポリシーの設定に必要な IAM アクセス許可

OpenSearch Serverless のデータアクセスコントロールは、次の IAM アクセス許可を使用します。IAM 条件を指定して、ユーザーを特定のアクセスポリシー名に制限できます。Amazon OpenSearch Service は、 を含むリソースベースのアクセスポリシーでグローバル IAM 条件キーもサポートしていますaws:SecureTransport。を使用してaws:SecureTransport、ドメインへのリクエストを HTTPS 経由で行うことを要求できます。例えば、次のようになります。

```
"Condition": {"Bool": {"aws:SecureTransport": "true"}}
```

- aoss:CreateAccessPolicy – アクセスポリシーを作成します。
- aoss:ListAccessPolicies – すべてのアクセスポリシーを一覧表示します。
- aoss:GetAccessPolicy – 特定のアクセスポリシーの詳細を表示します。
- aoss:UpdateAccessPolicy – アクセスポリシーを変更します。
- aoss:DeleteAccessPolicy – アクセスポリシーを削除します。

次の ID ベースのアクセスポリシーでは、ユーザーはすべてのアクセスポリシーを表示し、リソースパターン collection/logs を含むポリシーを更新できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "aoss:ListAccessPolicies",
                "aoss:GetAccessPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "aoss:UpdateAccessPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aoss:collection": [
                        "logs"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> さらに、OpenSearch Serverless には、コレクションリソースの `aoss:APIAccessAll` および `aoss:DashboardsAccessAll` アクセス許可が必要です。詳細については、「the section called “OpenSearch API オペレーションの使用”」を参照してください。

ポリシー構文

データアクセスポリシーには、次の要素を持つ一連のルールが含まれています。

Element	説明
`ResourceType`	アクセス許可が適用されるリソースのタイプ (コレクションまたはインデックス)。エイリアスとテンプレートのアクセス許可はコレクションレベルで、データの作成、変更、検索のアクセス許可はインデックスレベルです。詳細については、「Supported policy permissions」(サポートされているポリシーのアクセス許可) を参照してください。
`Resource`	リソース名やパターンのリスト。パターンはプレフィックスの後にワイルドカード (*) が続くもので、これによって関連付けられたアクセス許可を複数のリソースに適用できます。 • コレクションの形式は `collection/` *`<name|pattern>`* です。 • インデックスの形式は `index/`*`<collection-name|pattern>`* `/`*`<index-name|pattern/>`* です。
`Permission`	指定されたリソースに付与するアクセス許可のリスト。これによって許可されるアクセス許可とオペレーションの完全なリストについては、「the section called “サポートされる OpenSearch API オペレーションとアクセス許可”」を参照してください。
`Principal`	アクセス権が付与される 1 つ以上のプリンシパルのリスト。プリンシパルは、IAM ロール ARN または SAML アイデンティティにすることができます。これらのプリンシパルは現在の AWS アカウント内にある必要があります。データアクセスポリシーはクロスアカウントアクセスを直接サポートしていませんが、別の のユーザーがコレクション所有アカウントで引き受け AWS アカウント ることができるロールをポリシーに含めることができま

Element	説明
	す。詳細については、「the section called “クロスアカウントデータアクセス”」を参照してください。

次のポリシー例では、autopartsinventory というコレクションおよび sales* というプレフィックスで始まるすべてのコレクションにエイリアスとテンプレートのアクセス許可を付与します。また、autopartsinventory コレクション内のすべてのインデックス、および orders* というプレフィックスで始まる salesorders コレクション内のすべてのインデックスに読み取りおよび書き込みアクセス許可を付与します。

```
[
   {
      "Description": "Rule 1",
      "Rules":[
         {
            "ResourceType":"collection",
            "Resource":[
               "collection/autopartsinventory",
               "collection/sales*"
            ],
            "Permission":[
               "aoss:CreateCollectionItems",
               "aoss:UpdateCollectionItems",
               "aoss:DescribeCollectionItems"
            ]
         },
         {
            "ResourceType":"index",
            "Resource":[
               "index/autopartsinventory/*",
               "index/salesorders/orders*"
            ],
            "Permission":[
               "aoss:*"
            ]
         }
      ],
      "Principal":[
         "arn:aws:iam::123456789012:user/Dale",
         "arn:aws:iam::123456789012:role/RegulatoryCompliance",
```

```
            "saml/123456789012/myprovider/user/Annie",
            "saml/123456789012/anotherprovider/group/Accounting"
        ]
    }
]
```

ポリシー内でアクセスを明示的に拒否することはできません。したがって、ポリシーのアクセス許可はすべて追加的です。例えば、あるポリシーでユーザーに aoss:ReadDocument を付与し、別のポリシーで aoss:WriteDocument を付与した場合、ユーザーには両方のアクセス許可が付与されます。3 番目のポリシーで同じユーザーに aoss:* を付与した場合、そのユーザーは関連付けられたインデックスですべてのアクションを実行できます。制限の厳しいアクセス許可が制限の緩いアクセス許可よりも優先されることはありません。

サポートされているポリシーのアクセス許可

データアクセスポリシーでは、次のアクセス許可がサポートされています。各アクセス許可で実行できる OpenSearch API オペレーションについては、「the section called “サポートされる OpenSearch API オペレーションとアクセス許可”」を参照してください。

コレクションアクセス許可

- aoss:CreateCollectionItems
- aoss:DeleteCollectionItems
- aoss:UpdateCollectionItems
- aoss:DescribeCollectionItems
- aoss:*

インデックスアクセス許可

- aoss:ReadDocument
- aoss:WriteDocument
- aoss:CreateIndex
- aoss:DeleteIndex
- aoss:UpdateIndex
- aoss:DescribeIndex
- aoss:*

OpenSearch Dashboards のサンプルデータセット

OpenSearch Dashboards には、独自のデータを追加する前に Dashboards を調べるのに役立つ、視覚化、ダッシュボード、その他ツールを備えた、サンプルデータセットが用意されています。このサンプルデータからインデックスを作成するには、処理するデータセットへのアクセスを許可する、データアクセスポリシーが必要になります。次のポリシーでは、ワイルドカード (*) を使用して、3 つのサンプルデータセットすべてへのアクセスを許可します。

```
[
  {
    "Rules": [
      {
        "Resource": [
          "index/<collection-name>/opensearch_dashboards_sample_data_*"
        ],
        "Permission": [
          "aoss:CreateIndex",
          "aoss:DescribeIndex",
          "aoss:ReadDocument"
        ],
        "ResourceType": "index"
      }
    ],
    "Principal": [
      "arn:aws:iam::<account-id>:user/<user>"
    ]
  }
]
```

データアクセスポリシーの作成 (コンソール)

ビジュアルエディタを使用して、または JSON 形式で、データアクセスポリシーを作成できます。ポリシーで定義されているパターンのいずれかに一致する新しいコレクションには、対応するアクセス許可がコレクションの作成時に割り当てられます。

OpenSearch Serverless データアクセスポリシーを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで [サーバーレス] を展開し、[セキュリティ] の下の [データアクセスポリシー] を選択します。

3. [Create access policy] (アクセスポリシーの作成) を選択します。
4. ポリシーの名前と説明を入力します。
5. ポリシーの最初のルールの名前を入力します。例えば、「Logs collection access」。
6. [Add principals] (プリンシパルの追加) を選択し、データアクセス権が付与される IAM ロール、または SAML ユーザーとグループを 1 つ、または複数選択します。

> **Note**
>
> ドロップダウンメニューからプリンシパルを選択するには、`iam:ListUsers` および `iam:ListRoles` アクセス許可 (IAM プリンシパルの場合) および `aoss:ListSecurityConfigs` アクセス許可 (SAML ID の場合) が必要です。

7. [Grant] (付与) を選択し、エイリアス、テンプレート、およびインデックスのアクセス許可を選択して、関連するプリンシパルに付与します。アクセス許可とそれによって許可されるアクセスの完全なリストについては、「the section called “サポートされる OpenSearch API オペレーションとアクセス許可”」を参照してください。
8. (オプション) ポリシーに追加のルールを設定します。
9. [作成] を選択します。ポリシーを作成してからアクセス許可が適用されるまでに 1 分程度かかる場合があります。5 分以上かかる場合は、サポート にお問い合わせください。

> **Important**
>
> ポリシーにインデックス許可のみが含まれていてコレクション許可が含まれていない場合でも、`Collection cannot be accessed yet. Configure data access policies so that users can access the data within this collection` といった、一致するコレクションに関するメッセージが表示されることがあります。この警告は無視できます。許可されたプリンシパルは引き続き、それぞれに割り当てられたインデックス関連の操作をコレクションで実行できます。

データアクセスポリシーの作成 (AWS CLI)

OpenSearch Serverless API を使用してデータアクセスポリシーを作成するには、`CreateAccessPolicy` コマンドを使用します。コマンドは、インラインポリシーと .json ファイルの両方を受け入れます。インラインポリシーは JSON エスケープ文字列としてエンコードする必要があります。

次のリクエストで、データアクセスポリシーが作成されます。

```
aws opensearchserverless create-access-policy \
    --name marketing \
    --type data \
    --policy "[{\"Rules\":[{\"ResourceType\":\"collection\",\"Resource\":
[\"collection/autopartsinventory\",\"collection/sales*\"],\"Permission\":
[\"aoss:UpdateCollectionItems\"]},{\"ResourceType\":\"index\",\"Resource\":
[\"index/autopartsinventory/*\",\"index/salesorders/orders*\"],\"Permission
\":[\"aoss:ReadDocument\",\"aoss:DescribeIndex\"]}],\"Principal\":
[\"arn:aws:iam::123456789012:user/Shaheen\"]}]"
```

.json ファイル内でポリシーを指定するには、`--policy file://my-policy.json` の形式を使用します。

ポリシーに含まれるプリンシパルは、アクセス権が付与された OpenSearch オペレーションを使用できるようになりました。

データアクセスポリシーの表示

コレクションを作成する前に、アカウント内の既存のデータアクセスポリシーをプレビューして、コレクション名と一致するリソースパターンがあるポリシーを確認することをお勧めします。次の ListAccessPolicies リクエストでは、アカウント内のすべてのデータアクセスポリシーが一覧表示されます。

```
aws opensearchserverless list-access-policies --type data
```

リクエストは、設定されているすべてのデータアクセスポリシーに関する情報を返します。1 つの特定のポリシーで定義されているパターンルールを表示するには、レスポンスの `accessPolicySummaries` 要素の内容でポリシー情報を探します。このポリシーの `name` と `type` を書き留め、GetAccessPolicy リクエストでこれらのプロパティを使用して、次のポリシーの詳細を含むレスポンスを受信します。

```
{
    "accessPolicyDetails": [
        {
            "type": "data",
            "name": "my-policy",
            "policyVersion": "MTY2NDA1NDE4MDg1OF8x",
            "description": "My policy",
```

```
            "policy": "[{\"Rules\":[{\"ResourceType\":\"collection\",
\"Resource\":[\"collection/autopartsinventory\",\"collection/sales*\"],
\"Permission\":[\"aoss:UpdateCollectionItems\"]},{\"ResourceType\":\"index\",
\"Resource\":[\"index/autopartsinventory/*\",\"index/salesorders/orders*\"],
\"Permission\":[\"aoss:ReadDocument\",\"aoss:DescribeIndex\"]}],\"Principal\":
[\"arn:aws:iam::123456789012:user/Shaheen\"]}]",
            "createdDate": 1664054180858,
            "lastModifiedDate": 1664054180858
        }
    ]
}
```

次のようにリソースフィルターを追加して、結果を特定のコレクションまたはインデックスを含むポリシーに限定できます。

```
aws opensearchserverless list-access-policies --type data --resource
 "index/autopartsinventory/*"
```

特定のポリシーに関する詳細を表示するには、GetAccessPolicy コマンドを使用します。

データアクセスポリシーの更新

データアクセスポリシーを更新すると、関連するすべてのコレクションが影響を受けます。OpenSearch Serverless コンソールでデータアクセスポリシーを更新するには、[Data access control] (データアクセスコントロール) を選択し、変更するポリシーを選択して [Edit] (編集) を選択します。変更を行ってから、[保存] を選択します。

OpenSearch Serverless API を使用してデータアクセスポリシーを更新するには、UpdateAccessPolicy リクエストを送信します。ポリシーバージョンを含める必要があります。ポリシーバージョンは、ListAccessPolicies または GetAccessPolicy コマンドを使用して取得できます。最新のポリシーバージョンを含めると、他のユーザーによる変更を意図せず上書きしてしまうことがなくなります。

次の UpdateAccessPolicy リクエストは、データアクセスポリシーを新しいポリシー JSON ドキュメントで更新します。

```
aws opensearchserverless update-access-policy \
    --name sales-inventory \
    --type data \
    --policy-version MTY2NDA1NDE4MDg1OF8x \
```

```
    --policy file://my-new-policy.json
```

ポリシーを更新してから新しいアクセス許可が適用されるまでに数分かかる場合があります。

データアクセスポリシーの削除

データアクセスポリシーを削除すると、関連するすべてのコレクションは、ポリシーで定義されているアクセス権を失います。ポリシーを削除する前に、IAM ユーザーと SAML ユーザーがコレクションに適切なアクセス権を持っていることを確認してください。OpenSearch Serverless コンソールでポリシーを削除するには、ポリシーを選択し、[Delete] (削除) を選択します。

次のように DeleteAccessPolicy コマンドを使用することもできます。

```
aws opensearchserverless delete-access-policy --name my-policy --type data
```

クロスアカウントデータアクセス

クロスアカウント ID またはクロスアカウントコレクションを使用してデータアクセスポリシーを作成することはできませんが、ロールの引き受けオプションを使用してクロスアカウントアクセスを設定することもできます。例えば、*account-b* がアクセスする必要があるコレクションを *account-a* が所有している場合、*account-b* のユーザーは *account-a* でロールを引き受けることができます。ロールには IAM アクセス許可、`aoss:APIAccessAll` および `aoss:DashboardsAccessAll` があり、*account-a* のデータアクセスポリシーに含まれている必要があります。

を介したデータプレーンアクセス AWS PrivateLink

Amazon OpenSearch Serverless は、コントロールプレーンとデータプレーンオペレーションの 2 種類の AWS PrivateLink 接続をサポートしています。コントロールプレーンオペレーションには、コレクションの作成と削除、アクセスポリシーの管理が含まれます。データプレーンオペレーションは、コレクション内のデータのインデックス作成とクエリに使用します。このページでは、データプレーン VPC エンドポイントについて説明します。コントロールプレーン AWS PrivateLink エンドポイントの詳細については、「」を参照してくださいthe section called “コントロールプレーン VPC エンドポイント”。

を使用して AWS PrivateLink 、VPC と Amazon OpenSearch Serverless の間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのように OpenSearch Serverless にアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても OpenSearch Serverless にアクセスできま

す。VPC ネットワークアクセスの詳細については、「Network connectivity patterns for Amazon OpenSearch Serverless」を参照してください。

このプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに指定した各サブネットに、エンドポイントネットワークインターフェイスを作成します。これらは OpenSearch Serverless に送信されるトラフィックのエントリポイントとして機能する、リクエスタマネージド型のネットワークインターフェイスです。

詳細については「AWS PrivateLink ガイド」の「Access AWS のサービス through AWS PrivateLink」を参照してください。

トピック

- VPC エンドポイントの作成方法
- Classic コレクションエンドポイントの DNS 解決
- VPC とネットワークアクセスポリシー
- VPC とエンドポイントポリシー
- 考慮事項
- 必要なアクセス許可
- 標準 AWS PrivateLink インターフェイスエンドポイントを作成する (OpenSearch Serverless NextGen)
- OpenSearch Serverless マネージドインターフェイスエンドポイントを作成する (OpenSearch Serverless Classic)
- Amazon OpenSearch Serverless の共有 VPC の設定

VPC エンドポイントの作成方法

Amazon OpenSearch Serverless は、アクセスするコレクションエンドポイントのタイプに応じて、データプレーン VPC エンドポイントを作成する 2 つの方法をサポートしています。コレクションのエンドポイント形式の詳細については、「」を参照してくださいthe section called “コレクションエンドポイント”。

- 標準 AWS PrivateLink VPC エンドポイント (NextGen) – このメソッドを使用して、 の OpenSearch Serverless NextGen コレクションエンドポイントにアクセスします on.aws (コレクションごとおよびアカウントごとの形式) 。VPC エンドポイントは、他の のインターフェイス

VPC エンドポイントを作成するのと同じ方法で、Amazon VPC コンソールまたは Amazon EC2 `CreateVpcEndpoint` API を使用して作成します AWS のサービス。このタイプのエンドポイントを作成するには、「」を参照してくださいthe section called “標準インターフェイスエンドポイントを作成する (NextGen)”。

- OpenSearch Serverless マネージド VPC エンドポイント (Classic) – この方法を使用して、 の OpenSearch Serverless Classic コレクションごとのエンドポイントにアクセスしますaoss.amazonaws.com。VPC エンドポイントは、OpenSearch Serverless コンソール、OpenSearch Serverless `CreateVpcEndpoint` API、または OpenSearch Serverless を使用して作成します AWS CLI。このサービスは、ユーザーに代わってインターフェイス VPC エンドポイントと Amazon Route 53 プライベートホストゾーンをアカウントに作成します。このタイプのエンドポイントを作成するには、「」を参照してくださいthe section called “OpenSearch Serverless マネージドインターフェイスエンドポイントを作成する (Classic)”。

Classic コレクションエンドポイントの DNS 解決

OpenSearch Serverless コンソールを使用してデータプレーン VPC エンドポイントを作成すると、サービスは新しい Amazon Route 53 プライベートホストゾーンを作成し、VPC にアタッチします。このプライベートホストゾーンは、OpenSearch Serverless コレクション (`*.us-east-1.aoss.amazonaws.com`) のワイルドカード DNS レコードをエンドポイントに使用されるインターフェイスアドレスに解決するレコードで構成されます。VPC 内に OpenSearch Serverless の VPC エンドポイントが 1 つあれば、各 AWS リージョン のすべてのコレクションと Dashboards にアクセスできます。OpenSearch Serverless のエンドポイントを持つすべての VPC には、独自のプライベートホストゾーンがアタッチされています。

> **Note**
>
> 単一の VPC エンドポイントはマルチ AZ 冗長性を提供しません。高可用性が必要な場合は、異なるアベイラビリティーゾーンにまたがる複数のサブネットに VPC エンドポイントをデプロイすることを検討してください。

OpenSearch Serverless インターフェイスエンドポイントは、リージョン内のすべてのコレクションのパブリック Route 53 ワイルドカード DNS レコードも作成します。DNS 名は OpenSearch Serverless パブリック IP アドレスに解決されます。OpenSearch Serverless の VPC エンドポイントを持たない VPC 内のクライアント、またはパブリックネットワーク内のクライアントは、パブリック Route 53 リゾルバーを使用し、それらの IP アドレスを使用してコレクションと Dashboards にアクセスできます。VPC エンドポイントの IP アドレスタイプ (IPv4、IPv6、またはデュアルスタッ

ク) は、OpenSearch Serverless のインターフェイスエンドポイントを作成するときに提供されるサブネットに基づいて決定されます。

> **Note**
>
> OpenSearch Serverless は、OpenSearch Service ドメイン解決用の追加の Amazon Route 53 プライベートホストゾーン (`<region>.opensearch.amazonaws.com`) を作成します。 AWS CLIの update-vpc-endpoint コマンドを使用して、既存の IPv4 VPC エンドポイントを Dualstack に更新できます。

特定の VPC の DNS リゾルバーアドレスは、VPC CIDR の 2 番目の IP アドレスです。VPC 内のクライアントは、そのリゾルバーを使用して、コレクション用の VPC エンドポイントアドレスを取得する必要があります。リゾルバーは、OpenSearch Serverless によって作成されたプライベートホストゾーンを使用します。任意のアカウントのすべてのコレクション用にそのリゾルバーを使用すれば十分です。一部のコレクションエンドポイントには VPC リゾルバーを使用し、他のコレクションエンドポイントにはパブリックリゾルバーを使用することもできますが、通常は必要ありません。

VPC とネットワークアクセスポリシー

コレクション用に OpenSearch API および Dashboards に対するネットワーク許可を付与するために、OpenSearch Serverless のネットワークアクセスポリシーを使用できます。このネットワークアクセスは、VPC エンドポイントまたはパブリックインターネットのいずれからでも制御できます。ネットワークポリシーはトラフィックの許可のみを制御するため、コレクション内のデータとそのインデックスに対して操作するための許可を指定するデータアクセスポリシーも設定する必要があります。OpenSearch Serverless の VPC エンドポイントはサービスへのアクセスポイント、ネットワークアクセスポリシーはコレクションおよび Dashboards へのネットワークレベルのアクセスポイント、データアクセスポリシーはコレクション内のデータに対するあらゆる操作についてのきめ細かいアクセスコントロールのためのアクセスポイントとお考えください。

ネットワークポリシーでは複数の VPC エンドポイント ID を指定できるため、コレクションにアクセスする必要がある VPC ごとに VPC エンドポイントを作成することをお勧めします。これらの VPCsは、OpenSearch Serverless コレクションおよびネットワークポリシーを所有する AWS アカウントとは異なるアカウントに属することができます。あるアカウントの VPC-to-VPCピアリングやその他のプロキシソリューションを作成することはお勧めしません。これは、各 VPC が独自のエンドポイントを持つ場合よりも安全性とコスト効率が低くなります。最初の VPC は、ネットワークポリシーでその VPC のエンドポイントへのアクセスを設定した他の VPC の管理者には簡単に表示されません。

VPC とエンドポイントポリシー

Amazon OpenSearch Serverless は、VPC 用のエンドポイントポリシーをサポートします。エンドポイントポリシーは、VPC エンドポイントにアタッチして、エンドポイント AWS を使用して AWS サービスにアクセスできるプリンシパルを制御する IAM リソースベースのポリシーです。詳細については、「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

エンドポイントポリシーを使用するには、まずインターフェイスエンドポイントを作成する必要があります。OpenSearch Serverless コンソールまたは OpenSearch Serverless API を使用して、インターフェイスエンドポイントを作成できます。インターフェイスエンドポイントを作成した後、エンドポイントポリシーをエンドポイントに追加する必要があります。詳細については、「the section called “OpenSearch Serverless マネージドインターフェイスエンドポイントを作成する (Classic)”」を参照してください。

> **Note**
>
> OpenSearch Service コンソールでエンドポイントポリシーを直接定義することはできません。

エンドポイントポリシーは、設定済みの他の ID ベースポリシー、リソースベースポリシー、ネットワークポリシー、またはデータアクセスポリシーをオーバーライドしたり、これらに置き換わったりすることはありません。エンドポイントポリシーの更新の詳細については、「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

デフォルトでは、エンドポイントポリシーにより、VPC エンドポイントに対するフルアクセスが付与されます。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

デフォルトの VPC エンドポイントポリシーにより、エンドポイントに対するフルアクセスが付与されますが、特定のロールおよびユーザーに対するアクセスを許可するように VPC エンドポイントポリシーを設定できます。これを実行するには、次の例を参照してください:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "123456789012",
                    "987654321098"
                ]
            },
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

VPC エンドポイントポリシーで、OpenSearch Serverless コレクションを条件要素として含めるように指定できます。これを実行するには、次の例を参照してください:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aoss:collection": [
                        "coll-abc"
```

```
                    ]
                }
            }
        }
    ]
}
```

aoss:CollectionId のサポートがサポートされています。

```
Condition": {
         "StringEquals": {
               "aoss:CollectionId": "collection-id"
          }
}
```

VPC エンドポイントポリシーで SAML ID を使用して、VPC エンドポイントアクセスを決定できます。VPC エンドポイントポリシーのプリンシパルセクションではワイルドカード (*) を使用する必要があります。これを実行するには、次の例を参照してください:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "saml:cn": [
                        "saml/111122223333/idp123/group/football",
                        "saml/111122223333/idp123/group/soccer",
                        "saml/111122223333/idp123/group/cricket"
                    ]
                }
            }
        }
    ]
```

```
}
```

さらに、特定の SAML プリンシパルポリシーを含めるようにエンドポイントポリシーを設定できます。これを実行するには、次を参照してください:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/Department": [
                        "Engineering"]
                    }
                }
            }
        ]
    }
```

Amazon OpenSearch Serverless での SAML 認証の使用の詳細については、「Amazon OpenSearch Serverless での SAML 認証」を参照してください。

また、IAM ユーザーと SAML ユーザーを同じ VPC エンドポイントポリシーに含めることもできます。これを実行するには、次の例を参照してください:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "saml:cn": [
                        "saml/111122223333/idp123/group/football",
                        "saml/111122223333/idp123/group/soccer",
                        "saml/111122223333/idp123/group/cricket"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

インターフェイス VPC エンドポイントを介して Amazon EC2 から Amazon OpenSearch Serverless コレクションにアクセスすることもできます。詳細については、「Access an OpenSearch Serverless collection from Amazon EC2 (via interface VPC endpoints)」を参照してください。

考慮事項

OpenSearch Serverless のインターフェイスエンドポイントを設定する前に、次の点を考慮してください。

- OpenSearch Serverless は、サポートされているすべての OpenSearch API オペレーション (設定 API オペレーションではない) をインターフェイスエンドポイント経由で呼び出すことをサポートしています。
- OpenSearch Serverless のインターフェイスエンドポイントを作成したら、Serverless コレクションにアクセスできるようにするために、そのエンドポイントをネットワークアクセスポリシーに含める必要があります。

- デフォルトでは、インターフェイスエンドポイント経由で OpenSearch Serverless へのフルアクセスが許可されます。セキュリティグループをエンドポイントのネットワークインターフェイスに関連付けて、インターフェイスエンドポイント経由で OpenSearch Serverless へのトラフィックを制御できます。
- 1 つの に最大 50 個の OpenSearch Serverless VPC エンドポイント AWS アカウント を設定できます。
- ネットワークポリシーでコレクションの API またはダッシュボードへのパブリックインターネットアクセスを有効にすると、コレクションには任意の VPC およびパブリックインターネットからアクセスできます。
- オンプレミスで VPC の外部にいる場合、OpenSearch Serverless の VPC エンドポイント解決に DNS リゾルバーを直接使用することはできません。VPN アクセスが必要な場合、VPC には外部クライアントが使用できる DNS プロキシリゾルバーが必要です。Route 53 は、オンプレミスネットワークまたは別の VPC から VPC への DNS クエリを解決するために使用できるインバウンドエンドポイントオプションを提供します。
- OpenSearch Serverless が作成して VPC にアタッチするプライベートホストゾーンは、 サービスによって管理されますが、 Amazon Route 53 リソースに表示され、アカウントに請求されます。
- その他の考慮事項については、「AWS PrivateLink ガイド」の「Considerations」を参照してください。

必要なアクセス許可

OpenSearch Serverless の VPC アクセスでは、次の AWS Identity and Access Management (IAM) アクセス許可を使用します。IAM 条件を指定して、ユーザーを特定のコレクションに制限できます。

- `aoss:CreateVpcEndpoint` – VPC エンドポイントを作成します。
- `aoss:ListVpcEndpoints` – すべての VPC エンドポイントを一覧表示します。
- `aoss:BatchGetVpcEndpoint` – VPC エンドポイントのサブセットに関する詳細を参照してください。
- `aoss:UpdateVpcEndpoint` – VPC エンドポイントを変更します。
- `aoss:DeleteVpcEndpoint` – VPC エンドポイントを削除します。

さらに、VPC エンドポイントを作成するには、以下の Amazon EC2 許可と Route 53 許可が必要です。

- ec2:CreateTags
- ec2:CreateVpcEndpoint
- ec2:DeleteVpcEndPoints
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcEndpoints
- ec2:DescribeVpcs
- ec2:ModifyVpcEndPoint
- route53:AssociateVPCWithHostedZone
- route53:ChangeResourceRecordSets
- route53:CreateHostedZone
- route53:DeleteHostedZone
- route53:GetChange
- route53:GetHostedZone
- route53:ListHostedZonesByName
- route53:ListHostedZonesByVPC
- route53:ListResourceRecordSets

標準 AWS PrivateLink インターフェイスエンドポイントを作成する (OpenSearch Serverless NextGen)

の OpenSearch Serverless NextGen コレクションエンドポイントの場合on.aws、VPC エンドポイントを標準インターフェイス VPC エンドポイントとして作成します。これは、他のエンドポイントの場合と同じです AWS のサービス。Amazon VPC コンソールまたは Amazon EC2 CreateVpcEndpoint API を使用できます。これらのエンドポイントに OpenSearch Serverless CreateVpcEndpoint API を使用する必要はありません。

エンドポイントは、 のリージョン *.aoss.*region*.on.aws (および FIPS がサポートされているリージョン) のすべてのコレクション*.aoss-fips.*region*.on.awsを解決します。

VPC エンドポイントを作成するときは、次のサービス名を使用します。

```
com.amazonaws.region.aoss-data
```

たとえば、*us-east-1*リージョンでは、サービス名は ですcom.amazonaws.*us-east-1*.aoss-data。

OpenSearch Serverless NextGen の標準インターフェイス VPC エンドポイントを作成するには

1. 「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイント AWS のサービス を使用して にアクセスする」の手順に従います。
2. サービス名で、 を選択しますcom.amazonaws.*region*.aoss-data。
3. エンドポイントの VPC、サブネット、セキュリティグループを選択します。セキュリティグループが、エンドポイントを使用するリソースからのインバウンド HTTPS (ポート 443) トラフィックを許可していることを確認します。
4. エンドポイントのプライベート DNS を有効にして、OpenSearch Serverless NextGen コレクションのホスト名に AWS PrivateLinkマネージド DNS 解決を使用します。

エンドポイントを作成したら、OpenSearch Serverless ネットワークアクセスポリシーの vpce-*abc123def4EXAMPLE* ID を使用して、OpenSearch Serverless マネージド VPC エンドポイントの場合と同様に、エンドポイントにコレクションへのアクセスを許可できます。

OpenSearch Serverless マネージドインターフェイスエンドポイントを作成する (OpenSearch Serverless Classic)

の OpenSearch Serverless Classic コレクションごとのエンドポイントの場合aoss.amazonaws.com、OpenSearch Serverless コンソールまたは OpenSearch Serverless API を使用して OpenSearch Serverless マネージドインターフェイスエンドポイントを作成します。

OpenSearch Serverless Classic コレクションのインターフェイスエンドポイントを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインで [Serverless] (サーバーレス) を展開し、[VPC endpoints] (VPC エンドポイント) を選択します。
3. [Create VPC endpoint] (VPC エンドポイントの作成) を選択します。
4. エンドポイントの名前を入力します。
5. [VPC] で OpenSearch Serverless にアクセスする VPC を選択します。
6. [Subnet] (サブネット) で、OpenSearch Serverless にアクセスするサブネットを 1 つ選択します。

- エンドポイントの IP アドレスと DNS タイプはサブネットタイプに基づいています
  - Dualstack: すべてのサブネットに IPv4 と IPv6 の両方のアドレス範囲がある場合
  - IPv6: すべてのサブネットが IPv6 のみのサブネットの場合
  - IPv4: すべてのサブネットに IPv4 アドレス範囲がある場合

7. [Security groups] (セキュリティグループ) で、エンドポイントネットワークインターフェイスに関連付けるセキュリティグループを選択します。これは、エンドポイントに承認するインバウンドトラフィックのポート、プロトコル、ソースを制限する重要なステップです。OpenSearch Serverless との通信に VPC エンドポイントを使用するリソースがエンドポイントのネットワークインターフェイスと通信できるように、セキュリティグループルールが設定されていることを確認します。
8. エンドポイントの作成 を選択します。

OpenSearch Serverless API を使用して VPC エンドポイントを作成する場合は、`CreateVpcEndpoint` コマンドを使用します。

> **Note**
>
> エンドポイントを作成したら、その ID を書き留めます (例: `vpce-abc123def4EXAMPLE`)。コレクションへのエンドポイントアクセスを提供するには、この ID を 1 つまたは複数のネットワークアクセスポリシーに含める必要があります。

インターフェイスエンドポイントを作成したら、ネットワークアクセスポリシーを使用して、そのエンドポイントにコレクションへのアクセスを提供する必要があります。詳細については、「the section called “ネットワークアクセス”」を参照してください。

Amazon OpenSearch Serverless の共有 VPC の設定

Amazon Virtual Private Cloud (VPC) を使用して、VPC サブネットを組織 AWS アカウント 内の他のと共有したり、複数の のリソース間で VPN などのネットワークインフラストラクチャを共有したりできます AWS アカウント。

現在、Amazon OpenSearch Serverless は、 AWS PrivateLink VPC の所有者でない限り、共有 VPC への接続の作成をサポートしていません。 AWS PrivateLink また、 は、その VPC 間の接続の共有もサポートしていません AWS アカウント。

ただし、OpenSearch Serverless の柔軟なモジュール式のアーキテクチャに基づいて、共有 VPC をセットアップすることはできます。これは、OpenSearch Serverless ネットワークインフラストラクチャが個々のコレクション (OpenSearch Service) インフラストラクチャとは異なるためです。したがって、VPC が配置されている 1 つのアカウントの AWS PrivateLink VPCe エンドポイントを作成し、他のアカウントのネットワークポリシーで VPCe ID を使用して、トラフィックをその共有 VPC からのみに制限できます。

以下の手順では、所有者アカウントとコンシューマーアカウントについて説明します。

所有者アカウントは、VPC をセットアップして他のアカウントと共有するための共通のネットワークアカウントとして機能します。コンシューマーアカウントは、所有者アカウントによって共有されている VPC で OpenSearch Serverless コレクションを作成および維持するアカウントです。

前提条件

共有 VPC を設定する前に、以下の要件が満たされていることを確認してください。

- 目的の所有者アカウントは、Amazon Virtual Private Cloud で VPC、サブネット、ルートテーブル、およびその他の必要なリソースを既に設定している必要があります。詳細については、Amazon VPC ユーザーガイドを参照してください。
- 目的の所有者アカウントとコンシューマーアカウントは、 AWS Organizationsの同じ組織に属している必要があります。詳細については、「AWS Organizations ユーザーガイド」を参照してください。

所有者アカウント/共通ネットワークアカウントで共有 VPC を設定するには。

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 「the section called “OpenSearch Serverless マネージドインターフェイスエンドポイントを作成する (Classic)”」のステップを実行してください。その際、以下の選択を行います。
   - 組織内のコンシューマーアカウントと共有されている VPC とサブネットを選択します。
3. エンドポイントを作成したら、生成された VPCe ID を書き留めて、コンシューマーアカウントでセットアップタスクを実行する管理者に渡します。

   VPCe ID は `vpce-abc123def4EXAMPLE` 形式になっています。

コンシューマーアカウントで共有 VPC を設定するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. コレクションがまだない場合は、the section called “コレクションの管理” の情報を使用してコレクションを作成します。
3. the section called “ネットワークポリシーの作成 (コンソール)” の情報を使用して、ネットワークポリシーを作成します。その際、以下の選択を行います。

   > **Note**
   >
   > この目的のために、既存のネットワークポリシーを更新することもできます。

   a. [アクセスタイプ] に [VPC (推奨)] を選択します。
   b. [アクセス用の VPC エンドポイント] の場合は、所有者アカウントから提供された VPCe ID (`vpce-abc123def4EXAMPLE` 形式) を選択します。
   c. [リソースタイプ] エリアで、以下を実行します。
      - [OpenSearch エンドポイントへのアクセスを有効にする] ボックスを選択し、その共有 VPC からのアクセスを有効にするのに使用するコレクション名またはコレクションパターンを選択します。
      - [OpenSearch Dashboard へのアクセスを有効にする] ボックスを選択し、その共有 VPC からのアクセスを有効にするのに使用するコレクション名またはコレクションパターンを選択します。
4. 新しいポリシーの場合は、[作成] を選択します。既存のポリシーの場合は、[更新] を選択します。

を介したコントロールプレーンアクセス AWS PrivateLink

Amazon OpenSearch Serverless は、コントロールプレーンとデータプレーンオペレーションの 2 種類の AWS PrivateLink 接続をサポートしています。コントロールプレーンオペレーションには、コレクションの作成と削除、アクセスポリシーの管理が含まれます。データプレーンオペレーションは、コレクション内のデータのインデックス作成とクエリに使用します。このページでは、コントロールプレーン AWS PrivateLink エンドポイントについて説明します。データプレーン VPC エンド

ポイントの詳細については、「」を参照してくださいthe section called “データプレーン VPC エンドポイント”。

コントロールプレーン AWS PrivateLink エンドポイントの作成

インターフェイス VPC エンドポイントを使用するように OpenSearch Serverless を設定することで、VPC のセキュリティ体制を改善できます。インターフェイスエンドポイントは AWS PrivateLink を利用しています。このテクノロジーにより、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで、OpenSearch Serverless APIs にプライベートにアクセスできます。

AWS PrivateLink と VPC エンドポイントの詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイント」を参照してください。

考慮事項

- VPC エンドポイントは、同じリージョン内でのみサポートされます。
- VPC エンドポイントでは、Amazon Route 53 を介して Amazon 提供の DNS のみがサポートされています。
- VPC エンドポイントは、OpenSearch Serverless コレクション、ポリシー、および VpcEndpoints へのアクセスを制御するエンドポイントポリシーをサポートします。
- OpenSearch Serverless はインターフェイスエンドポイントのみをサポートします。ゲートウェイエンドポイントはサポートされていません。

VPC エンドポイントの作成

Amazon OpenSearch Serverless のコントロールプレーン VPC エンドポイントを作成するには、「Amazon VPC デベロッパーガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」の手順を使用します。次のエンドポイントを作成します。

- `com.amazonaws.region.aoss`

コンソールを使用してコントロールプレーン VPC エンドポイントを作成するには

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. ナビゲーションペインで、[Endpoints] (エンドポイント) を選択します。
3. [エンドポイントの作成] を選択します。

4. [サービスカテゴリ] で、[AWS のサービス] を選択します。
5. サービス で、 を選択しますcom.amazonaws.*region*.aoss。例えば、com.amazonaws.us-east-1.aoss。
6. [VPC] で、エンドポイントを作成する VPC を選択します。
7. [サブネット] で、エンドポイントネットワークインターフェイスを作成する先のサブネット (アベイラビリティーゾーン) を選択します。
8. セキュリティグループで、エンドポイントネットワークインターフェイスに関連付けるセキュリティグループを選択します。HTTPS (ポート 443) が許可されていることを確認します。
9. ポリシー では、フルアクセスを選択してすべてのオペレーションを許可するか、カスタムを選択してカスタムポリシーをアタッチします。
10. エンドポイントの作成 を選択します。

エンドポイントポリシーの作成

Amazon OpenSearch Serverless へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- アクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

Example OpenSearch Serverless の VPC エンドポイントポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "aoss:ListCollections",
        "aoss:BatchGetCollection"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

Exampleリストオペレーションのみを許可する制限ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "aoss:ListCollections",
      "Resource": "*"
    }
  ]
}
```

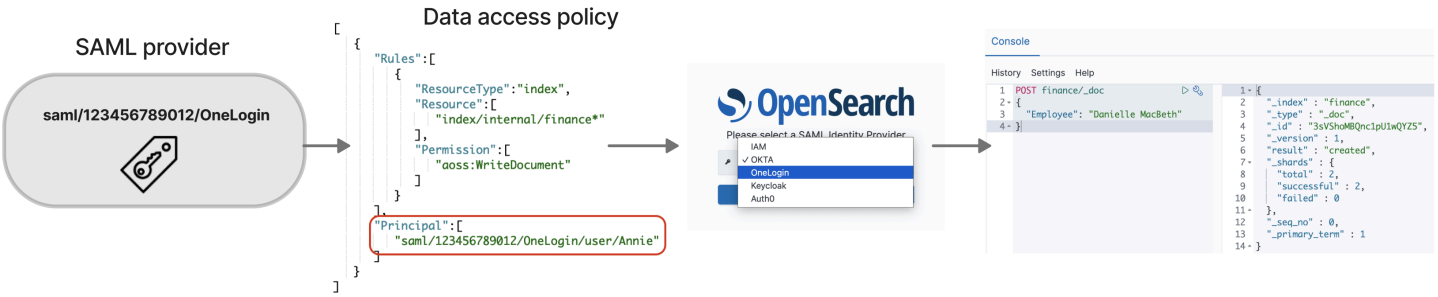
Amazon OpenSearch Serverless での SAML 認証

Amazon OpenSearch Serverless で SAML 認証を使用すると、Serverless コレクションの OpenSearch Dashboards エンドポイントに対し、既存の ID プロバイダーによるシングルサインオン (SSO) を提供できます。

SAML 認証により、ユーザーはサードパーティの ID プロバイダーを使用して OpenSearch Dashboards にサインインし、データのインデックス化と検索を行えます。OpenSearch Serverless では、IAM Identity Center、Okta、Keycloak、Active Directory Federation Services (AD FS)、Auth0 など、SAML 2.0 標準を使用するプロバイダーをサポートしています。Okta、OneLogin 、Microsoft Entra ID などの他の ID ソースからのユーザーとグループを同期するように IAM Identity Center を設定できます。IAM Identity Center でサポートされている ID ソースのリストと設定手順については、「IAM Identity Center ユーザーガイド」の「Getting started tutorials」を参照してください。

> **Note**
>
> SAML 認証は、ウェブブラウザから OpenSearch Dashboards にアクセスする場合のみに使用します。この認証を受けたユーザーは、OpenSearch Dashboards の [Dev Tools] (開発ツール) を通じてのみ、OpenSearch API オペレーションにリクエストを行うことができます。ユーザーが、SAML 認証情報により、OpenSearch API オペレーションに直接 HTTP リクエストを発行することはできません。

SAML 認証の使用を開始するには、まず SAML の ID プロバイダー (IdP) を設定します。次に、その IdP から 1 人以上のユーザーを、データアクセスポリシーに含めます。このポリシーにより、コレクションやインデックスに対する特定のアクセス許可が付与されます。その後、ユーザーは、OpenSearch Dashboards にサインインして、データアクセスポリシーで許可されているアクションを実行できます。



トピック

- 考慮事項
- 必要なアクセス許可
- SAML プロバイダーの作成 (コンソール)
- OpenSearch Dashboards へのアクセス
- SAML ID に対するコレクションデータへのアクセス権の付与
- SAML プロバイダの作成 (AWS CLI)
- SAML プロバイダーの表示
- SAML プロバイダの更新
- SAML プロバイダーの削除

考慮事項

SAML 認証を設定する際には、以下を考慮します。

- 署名済みおよび暗号化されたリクエストはサポートされていません。
- 暗号化されたアサーションはサポートされていません。
- 認証とサインアウトを IdP が開始することはサポートされていません。
- サービスコントロールポリシー (SCP) は、IAM 以外の ID (Amazon OpenSearch Serverless & SAML における SAML、あるいは Amazon OpenSearch Service の 基本的な内部ユーザー許可など) の場合には適用または評価されません。

必要なアクセス許可

OpenSearch Serverless の SAML 認証では、次の AWS Identity and Access Management (IAM) アクセス許可を使用します。

- `aoss:CreateSecurityConfig` – SAML プロバイダーを作成します。
- `aoss:ListSecurityConfig` – 現在のアカウントのすべての SAM Lプロバイダーを一覧表示します。
- `aoss:GetSecurityConfig` – SAML プロバイダーの情報を表示します。
- `aoss:UpdateSecurityConfig` – 特定の SAML プロバイダーの (XML メタデータを含む) 設定を変更します。
- `aoss:DeleteSecurityConfig` – SAML プロバイダーを削除します。

次の ID ベースのアクセスポリシーでは、すべての IdP 設定を管理することを、ユーザーに対し許可します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "aoss:CreateSecurityConfig",
                "aoss:DeleteSecurityConfig",
                "aoss:GetSecurityConfig",
                "aoss:UpdateSecurityConfig",
                "aoss:ListSecurityConfigs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

`Resource` 要素はワイルドカードにする必要がある点に、注意してください。

SAML プロバイダーの作成 (コンソール)

以下の手順では、SAML プロバイダーを作成する方法について説明します。この手順により、SAML 認証がサービスプロバイダー (SP) を使用して、OpenSearch Dashboards に対する認証を開始できるようになります。認証を IdP が開始することはサポートされていません。

OpenSearch Dashboards の SAML 認証を有効にするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションパネルで [Serverless] (サーバーレス) を展開し、[SAML authentication] (SAML 認証) を選択します。
3. [Add SAML provider] (SAML プロバイダーを追加) を選択します。
4. プロバイダーの名前と説明を入力します。

   > **Note**
   >
   > ここで指定した名前はパブリックにアクセス可能で、OpenSearch Dashboards にサインインしたユーザーは、ドロップダウンメニューに表示することができます。名前は認識しやすく、また、ID プロバイダーに関する機密情報が明らかにならないものにしてください。

5. [Configure your IdP] (IDP の設定) で、アサーションコンシューマーサービス (ACS) のURL をコピーします。
6. ここでコピーした ACS URL は、ID プロバイダーを設定するために使用します。用語と手順はプロバイダーによって異なります。プロバイダーのドキュメントを参照してください。

   例えば、Okta では「SAML 2.0 ウェブアプリケーション」を作成し、ACS URL を、[Single Sign On URL] (シングルサインオン URL)、[Recipient URL] (受信者 URL)、[Destination URL] (送信先 URL) として指定します。Auth0 の場合は、この情報を [Allowed Callback URLs] (許可するコールバック URL) の中で指定します。
7. IdP にオーディエンス制限用のフィールドがある場合は、オーディエンス制限を設定します。オーディエンス制限は SAML アサーション内の値であり、これによりアサーションの対象者を指定します。OpenSearch Serverless では、以下を実行できます。次のコード例の######独自の AWS アカウント ID に置き換えてください。

   1. デフォルトのオーディエンス制限 `:opensearch:111122223333` を使用します。

2. (オプション) AWS CLIを使用して、カスタムオーディエンス制限を設定します。詳細については、「SAML プロバイダの作成 (AWS CLI)」を参照してください。

   オーディエンス制限フィールドの名前は、プロバイダーによって異なります。Okta の場合は、[Audience URI (SP Entity ID)] (オーディエンス URI (SP エンティティ ID)) です。IAM ID センターでは、[Application SAML audience] (アプリケーション SAML オーディエンス) になります。

8. IAM ID センターを使用している場合は、属性マッピング (unspecified の形式を使用する Subject=${user:name}) も指定する必要があります。

9. ID プロバイダーを設定すると、IdP メタデータファイルが生成されます。この XML ファイルには、TLS 証明書、Single Sign-On エンドポイント、ID プロバイダーのエンティティ ID など、プロバイダーに関する情報が含まれています。

   IdP メタデータファイル内のテキストをコピーして、[Provide metadata from your IdP] (IdP からメタデータを提供) フィールドに貼り付けます。または、[XML ファイルからインポート] を選択し、ファイルをアップロードします。メタデータファイルは、次のように表示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="entity-id"
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  <md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>tls-certificate</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>s
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</
md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</
md:NameIDFormat>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST" Location="idp-sso-url"/>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="idp-sso-url"/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

10. ユーザーネームに対する SAML アサーションの NameID 要素を使用するために、[Custom user ID attribute] (カスタムユーザー ID 属性) フィールドは空のままにしておきます。アサーションでこの標準エレメントを使用せず、代わりにユーザーネームをカスタム属性として含める場合は、ここでその属性を指定します。属性では、大文字と小文字が区別されます。シングルユーザー属性のみがサポートされています。

    次の例では、SAML アサーション内の、NameID に対するオーバーライド属性を示しています。

    ```
    <saml2:Attribute Name="UserId" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
    format:basic">
      <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="xs:string">annie</saml2:AttributeValue>
    </saml2:Attribute>
    ```

11. (オプション) [Group attribute] (グループ属性) フィールドで、カスタム属性 (role または groupなど) を指定します。グループ属性は、 1 つだけがサポートされます。グループ属性には、デフォルト値はありません。これを指定しない場合、データアクセスポリシーにはユーザープリンシパルのみが含まれます。

    次に、SAML アサーションのグループ属性での例を示します。

    ```
    <saml2:Attribute Name="department"
     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xs:string">finance</saml2:AttributeValue>
    </saml2:Attribute>
    ```

12. OpenSearch Dashboards のデフォルトでは、 24 時間を経過したユーザーはログアウトされます。[OpenSearch Dashboards のタイムアウト] を指定することで、この値を 1 ~ 12 時間 (15 ~ 720 分) の任意の数値に設定することができます。タイムアウトを 15 分以下に設定しようとすると、セッションは 1 時間にリセットされます。
13. [Create SAML provider] (SAML プロバイダーを作成) を選択します。

OpenSearch Dashboards へのアクセス

SAML プロバイダーを設定すると、そのプロバイダーに関連付けられたすべてのユーザーとグループは、OpenSearch Dashboards エンドポイントを使用できるようになります。Dashboards URL は、すべてのコレクションで *collection-endpoint*/_dashboards/ の形式になっています。

SAML が有効になっている場合、 のリンクを選択すると IdP 選択ページ AWS マネジメントコンソール が表示され、SAML 認証情報を使用してサインインできます。まず、ドロップダウンから ID プロバイダーを選択します。



次に、自分の IdP 認証情報を使用してサインインします。

SAML が有効になっていない場合、 のリンクを選択すると、SAML のオプションなしで IAM ユーザーまたはロールとしてログインするように AWS マネジメントコンソール 指示されます。

SAML ID に対するコレクションデータへのアクセス権の付与

SAML プロバイダーを作成した後も、基盤となるユーザーとグループに対して、コレクション内のデータへのアクセス権を付与する必要があります。アクセス権は、データアクセスポリシーを介して付与します。アクセスが付与されるまで、ユーザーはコレクション内のデータの読み取り、書き込み、削除を行うことはできません。

アクセスを許可するには、データアクセスポリシーを作成し、Principal ステートメントの中で、SAML ユーザーおよび/またはグループ ID を指定します。

```
[
   {
      "Rules":[
       ...
      ],
      "Principal":[
         "saml/987654321098/myprovider/user/Shaheen",
         "saml/987654321098/myprovider/group/finance"
      ]
   }
```

```
]
```

アクセス権は、コレクション、インデックス、またはその両方に付与できます。ユーザーごとに異なる権限を持たせたい場合は、複数のルールを作成します。使用可能な許可のリストについては、「Supported policy permissions」(サポートされるポリシーの許可) を参照してください。アクセスポリシーの形式の詳細については、「Policy syntax」(ポリシーの構文) を参照してください。

SAML プロバイダの作成 (AWS CLI)

OpenSearch Serverless API を使用して SAML プロバイダーを作成するには、CreateSecurityConfig リクエストを送信します。

```
aws opensearchserverless create-security-config \
    --name myprovider \
    --type saml \
    --saml-options file://saml-auth0.json
```

.json ファイル内のキーと値のマップとして (メタデータ XML を含めながら) saml-options を指定します。メタデータ XML は、JSON エスケープ文字列としてエンコードする必要があります。

```
{
   "sessionTimeout": 70,
   "groupAttribute": "department",
   "userAttribute": "userid",
   "openSearchServerlessEntityId": "aws:opensearch:111122223333:app1",
   "metadata": "EntityDescriptor xmlns=\"urn:oasis:names:tc:SAML:2.0:metadata
\" ... ... ... IDPSSODescriptor\r\n\/EntityDescriptor"
}
```

> **Note**
>
> (オプション) AWS CLIを使用して、カスタムオーディエンス制限を設定します。詳細については、「SAML プロバイダの作成 (AWS CLI)」を参照してください。

SAML プロバイダーの表示

次の ListSecurityConfigs リクエストは、アカウント内のすべての SAML プロバイダーを一覧表示します。

```
aws opensearchserverless list-security-configs --type saml
```

このリクエストでは、既存のすべての SAML プロバイダーに関する (ID プロバイダーが生成する完全な IdP メタデータを含む) 情報を返します。

```
{
   "securityConfigDetails": [
      {
         "configVersion": "MTY2NDA1MjY4NDQ5M18x",
         "createdDate": 1664054180858,
         "description": "Example SAML provider",
         "id": "saml/111122223333/myprovider",
         "lastModifiedDate": 1664054180858,
         "samlOptions": {
            "groupAttribute": "department",
            "metadata": "EntityDescriptorxmlns=\"urn:oasis:names:tc:SAML:2.0:metadata
\" ..... ...IDPSSODescriptor\r\n/EntityDescriptor",
            "sessionTimeout": 120,
            "openSearchServerlessEntityId": "aws:opensearch:111122223333:app1",
            "userAttribute": "userid"
         }
      }
   ]
}
```

将来の更新ににおける configVersion を含め、特定のプロバイダの詳細を表示するには、GetSecurityConfig リクエストを送信します。

SAML プロバイダの更新

OpenSearch Serverless コンソールを使用して SAML プロバイダーを更新するには、[SAML authentication] (SAML 認証) を選択し、ID プロバイダーを選択した後に、[Edit] (編集) を選択します。メタデータやカスタム属性を含め、すべてのフィールドを変更可能です。

OpenSearch Serverless API を使用してプロバイダーを更新するには、更新するポリシーの識別子を指定しながら、UpdateSecurityConfig リクエストを送信します。また、設定のバージョン (ListSecurityConfigs または GetSecurityConfig コマンドで取得可能) も含める必要があります。最新バージョンを含めると、他のユーザーが行った変更を不注意に上書きしてしまうことを防げます。

次のリクエストは、プロバイダーの SAML オプションを更新します。

```
aws opensearchserverless update-security-config \
    --id saml/123456789012/myprovider \
    --type saml \
    --saml-options file://saml-auth0.json \
    --config-version MTY2NDA1MjY4NDQ5M18x
```

SAML 設定オプションは、.json ファイル内のキーと値のマップとして指定します。

> **Important**
>
> SAML オプションの更新を段階的に実行することはできません。更新時に、SAMLOptions オブジェクト内でパラメータの値を指定しない場合、既存の値は空の値で上書きされます。例えば、現在の構成で userAttribute に値が指定されていて、この値を含めずに更新を行った場合、その値は構成から削除されます。GetSecurityConfig オペレーション呼び出しにより更新を実行する前に、既存の値が何であるかを確認してください。

SAML プロバイダーの削除

SAML プロバイダーを削除した場合、データアクセスポリシー内で関連付けられたユーザーやグループへの参照は、以後、機能しなくなります。混乱を避けるため、エンドポイントを削除する前に、アクセスポリシー内で (指定している) エンドポイントへの参照を、すべて削除することをお勧めします。

OpenSearch Serverless コンソールを使用して SAML プロバイダーを削除するには、[Authentication] (認証) を選択し、プロバイダーを選択した後に、[Delete] (削除) を選択します。

OpenSearch サーバーServerless API を使用してプロバイダーを削除するには、DeleteSecurityConfig リクエストを送信します。

```
aws opensearchserverless delete-security-config --id saml/123456789012/myprovider
```

Amazon OpenSearch Serverless のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS コンプライアンスプログラムの一環として Amazon OpenSearch Serverless のセキュリティとコンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA が含まれます。

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照し

て、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

Amazon OpenSearch Serverless コレクション

タグを使用すると、Amazon OpenSearch Serverless コレクションに任意の情報を割り当てて、その情報を分類してフィルタリングできます。タグは、 AWS AWS リソースに割り当てるメタデータラベルです。

各タグは、キーと値から構成されます。ユーザーが割り当てるタグでは、ユーザーがキーと値を定義します。たとえば、1 つのリソースのキーを `stage` と定義し、値を `test` と定義します。

タグを使用すると、 AWS リソースを識別して整理できます。多くの AWS サービスはタグ付けをサポートしているため、異なるサービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。例えば、Amazon OpenSearch Service ドメインに割り当てる OpenSearch Serverless コレクションに同じタグを割り当てることができます。

OpenSearch Serverless では、プライマリリソースはコレクションです。OpenSearch Service コンソール、、 AWS CLI OpenSearch Serverless API オペレーション、または AWS SDKs を使用して、コレクションからタグを追加、管理、削除できます。

必要なアクセス許可

OpenSearch Serverless は、コレクションのタグ付けに次の AWS Identity and Access Management Access Analyzer (IAM) アクセス許可を使用します。

- `aoss:TagResource`
- `aoss:ListTagsForResource`
- `aoss:UntagResource`

コレクションのタグ付け (コンソール)

コンソールは、コレクションにタグを付けるうえで最も簡単な方法です。

タグを作成するには (コンソール)

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで [Serverless] (サーバーレス) を展開し、[Collections] (コレクション) を選択します。
3. タグを追加するコレクションを選択し、[Tags] (タグ) タブに移動します。
4. [管理] を選択して、[新しいタグを追加] を選択します。
5. タグキーとオプションの値を入力します。
6. [保存] を選択します。

タグを削除するには、同じ手順に従って、[タグを管理] ページで [削除] を選択します。

タグを操作するコンソールを使用する方法の詳細については、AWS マネジメントコンソール入門ガイドの「タグエディター」を参照してください。

コレクションのタグ付け (AWS CLI)

を使用してコレクションにタグを付けるには AWS CLI、TagResource リクエストを送信します。

```
aws opensearchserverless tag-resource
  --resource-arn arn:aws:aoss:us-east-1:123456789012:collection/my-collection
  --tags Key=service,Value=aoss Key=source,Value=logs
```

ListTagsForResource コマンドを使用して、コレクションの既存のタグを表示します。

```
aws opensearchserverless list-tags-for-resource
  --resource-arn arn:aws:aoss:us-east-1:123456789012:collection/my-collection
```

UntagResource コマンドを使用してコレクションのタグを削除します。

```
aws opensearchserverless untag-resource
  --resource-arn arn:aws:aoss:us-east-1:123456789012:collection/my-collection
  --tag-keys service
```

Amazon OpenSearch Serverless でサポートされているオペレーションとプラグイン

Amazon OpenSearch Serverless は、インデックス作成、検索、メタデータ API オペレーションのサブセットの他にも、OpenSearch での利用が可能な各種の OpenSearch プラグインをサポートしています。データアクセスポリシー内にある表で、左側の列にアクセス許可を含めることで、アクセス権限を特定のオペレーションに制限できます。

トピック

- サポートされる OpenSearch API オペレーションとアクセス許可
- サポートされている OpenSearch プラグイン

サポートされる OpenSearch API オペレーションとアクセス許可

次の表に、OpenSearch Serverless がサポートする API オペレーションと、それに対応するデータアクセスポリシー許可を一覧で示します。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:CreateIndex`	PUT <index>	インデックスを作成します。詳細については、「Create index」(インデックスの作成) を参照してください。 **Note** このアクセス権限は、OpenSearch Dashboards のサンプルデータを使ってインデックスを作成するときにも適用されます。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:DescribeIndex`	• GET <index> • GET <index>/_mapping • GET <index>/_mappings • GET <index>/_setting • GET <index>/_setting/<setting> • GET <index>/_settings • GET <index>/_settings/<setting> • GET _cat/indices • GET _mapping • GET _mappings • GET _resolve/index/<index> • HEAD <index>	インデックスの詳細を表示します。詳細については、以下のリソースを参照してください。 • Get index (インデックスを取得する) • Get a mapping (マッピングを取得する) • Get settings (設定を取得する) • インデックスが存在する • CAT インデックス (レスポンスには `health` または `status` フィールドは含まれません)。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:WriteDocument`	• DELETE <index>/_doc/<id> • POST <index>/_bulk • POST <index>/_create/<id> (検索コレクションタイプのみ) • POST <index>/_doc • POST <index>/_update/<id> (検索コレクションタイプのみ) • POST _bulk • PUT <index>/_create/<id> (検索コレクションタイプのみ) • PUT <index>/_doc/<id> (検索コレクションタイプのみ)	ドキュメントを作成および更新する。詳細については、以下のリソースを参照してください。 • 一括 • データインデックス **Note** 一部の操作は、タイプ SEARCH のコレクションに対してのみ利用可能です。詳細については、「the section called “コレクションタイプを選択する”」を参照してください。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:ReadDocument`	• GET <index>/_analyz • GET <index>/_doc/<id> • GET <index>/_explain/<id> • GET <index>/_mget • GET <index>/_source/<id> • GET <index>/_count • GET <index>/_field_caps • GET <index>/_msearch • GET <index>/_rank_eval • GET <index>/_search • GET <index>/_validate/<query> • GET _analyze • GET _field_caps • GET _mget • GET _search • GET /_search/point_in_time/_all • HEAD <index>/_doc/<id> • HEAD <index>/_source/<id> • POST /_plugins/_sql • POST /_plugins/_ppl • POST /_plugins/_sql/_explain • POST /_plugins/_ppl/_explain • POST /_plugins/_ppl/_close • POST <index>/_analyze • POST /_search/point_in_time • POST <index>/_explain/<id> • POST <index>/_count • POST <index>/_field_caps	ドキュメントを読み取ります。詳細については、以下のリソースを参照してください。 • Perform text analysis (テキスト分析を実行する) • Get document (ドキュメントを取得する) • Count (カウント) • Query DSL (DSL をクエリする) • Ranking evaluation (ランキングの評価) • Analyze API (API を分析する) • Explain (説明する) • ポイントインタイム • SQL と PPL

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
	• POST <index>/_rank_eval • POST <index>/_search • POST _analyze • POST _field_caps • POST _search • DELETE /_search/point_in_time/_all • DELETE /_search/point_in_time	
`aoss:DeleteIndex`	DELETE <target>	インデックスを削除します。詳細については、「Delete index」(インデックスを削除する) を参照してください。
`aoss:UpdateIndex`	• POST _mapping • POST <index>/_mapping/ • POST <index>/_mappings/ • POST <index>/_setting • POST <index>/_settings • POST _setting • POST _settings • PUT _mapping • PUT <index>/_mapping • PUT <index>/_mappings/ • PUT <index>/_setting • PUT <index>/_settings • PUT _setting • PUT _settings	インデックス設定を更新します。詳細については、以下のリソースを参照してください。 • Mapping (マッピング) • Update settings (設定を更新する)

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:CreateCollectionItems`	• POST _aliases • POST /_plugins/_flow_framework/workflow • * POST /_plugins/_flow_framework/workflow/<workflow_id>/_provision • PUT _ingest/pipeline/<pipeline-id> • PUT _search/pipeline/<pipeline-id>	• インデックスエイリアス、パイプライン、テンプレートを作成します。エイリアスの詳細については、「Create aliases」(エイリアスを作成する) を参照してください。 • * テンプレートをプロビジョニングまたは再プロビジョニングします。ML Commons Client および OpenSearch Serverless サービスは、依存ポリシーを管理します。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:DescribeCollectionItems`	• GET <index>/_alias/<alias> • GET _alias • GET _alias/<alias> • GET _cat/aliases • GET _cat/templates • GET _cat/templates/<template_name> • GET _component_template • GET _component_template/<component-template> • GET _index_template • GET _index_template/<index-template> • GET _ingest/pipeline/<pipeline-id> • GET _ingest/pipeline/_simulate • GET /_plugins/_flow_framework/workflow/<workflow-id> • GET /_plugins/_flow_framework/workflow/_search • GET /_plugins/_flow_framework/workflow/<workflow-id>/_status • GET /_plugins/_flow_framework/workflow/state/_search • GET /_plugins/_flow_framework/workflow/_steps • GET /_plugins/_flow_framework/workflow/_step?workflow_step=<step_name> • GET _search/pipeline/<pipeline-id>	エイリアス、インデックスとフレームワークのテンプレート、パイプラインを操作する方法について説明します。詳細については、以下のリソースを参照してください。 • Manage aliases (エイリアスを管理する) • Index templates (インデックステンプレート)

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
	• HEAD _alias/<alias> • HEAD _component_template/<component-template> • HEAD _index_template/<name> • HEAD <index>/_alias/<alias> • POST _ingest/pipeline/_simulate • POST /_plugins/_flow_framework/workflow/_search • POST /_plugins/_flow_framework/workflow/state/_search	
`aoss:UpdateCollectionItems`	• POST <index>/_alias/<alias> • POST <index>/_aliases/<alias> • POST _component_template/<component-template> • POST _index_template/<index-template> • * POST /_plugins/_flow_framework/workflow/<workflow_id>/_deprovision • PUT <index>/_alias/<alias> • PUT <index>/_aliases/<alias> • PUT _component_template/<component-template> • PUT _index_template/<index-template> • PUT /_plugins/_flow_framework/workflow/<workflow_id>	エイリアス、インデックステンプレート、フレームワークテンプレートを更新します。詳細については、以下のリソースを参照してください。 • Index aliases (インデックスエイリアス) • Index templates (インデックステンプレート) * テンプレートのプロビジョニングを解除する API。ML Commons Client および OpenSearch Serverless サービスは、依存ポリシーを管理します。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:DeleteCollectionItems`	• DELETE <index>/_alias/<alias> • DELETE _component_template/<component-template> • DELETE _index_template/<index-template> • DELETE <index>/_aliases/<alias> • DELETE _search/pipeline/<pipeline-id> • DELETE _ingest/pipeline/<pipeline-id> • DELETE /_plugins/_flow_framework/workflow/<workflow_id>	エイリアス、インデックスとフレームワークのテンプレート、パイプラインを削除します。詳細については、以下のリソースを参照してください。 • Delete aliases (エイリアスを削除する) • Delete a template (テンプレートを削除する)

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:DescribeMLResource`	• GET /_plugins/_ml/models/<model_id> • GET /_plugins_ml/models/_search • GET /_plugins/_ml/model_groups/<model_group_id> • GET /_plugins/_ml/model_groups/_search • GET /_plugins/_ml/connectors/<connector_id> • GET /_plugins/_ml/connectors/_search • GET /_plugins/_ml/profile/tasks/<task_id> • POST /_plugins/_ml/models/_search • POST /_plugins/_ml/model_groups/_search • POST /_plugins/_ml/connectors/_search	モデルとコネクタに関する情報を取得するための GET および検索 API について説明します。
`aoss:CreateMLResource`	• POST /_plugins/_ml/models/_register • POST /_plugins/_ml/model_groups/_register • POST /_plugins/_ml/connectors/_create	ML リソースを作成するアクセス許可を付与します。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
aoss:UpdateMLResource	• PUT /_plugins/_ml/models/<model_id> • POST /_plugins/_ml/models/<model_id>/_deploy • POST /_plugins/_ml/models/<model_id>/_undeploy • PUT /_plugins/_ml/model_groups/<model_group_id> • PUT /_plugins/_ml/connectors/<connector_id>	既存の ML リソースを更新するアクセス許可を付与します。
aoss:DeleteMLResource	• DELETE /_plugins/_ml/models/<model_id> • DELETE _plugins/_ml/model_groups/<model_group_id> • DELETE /_plugins/_ml/connectors/<connector_id> • DELETE /_plugins/_ml/tasks/<task_id>	ML リソースを削除するアクセス許可を付与します。
aoss:ExecuteMLResource	• POST /_plugins/_ml/models/<model_id>/_predict	モデルを実行するアクセス許可を提供します。
aoss:CreateAgent	• POST /_plugins/_ml/agents/_register	エージェントを作成するアクセス許可を付与します。
aoss:DescribeAgent	• GET /_plugins/_ml/agents/<agent_id>	エージェント情報を取得するアクセス許可を提供します。
aoss:UpdateAgent	• PUT /_plugins/_ml/agents/<agent_id>	既存のエージェントを更新するアクセス許可を付与します。

データアクセスポリシーでの許可	OpenSearch API オペレーション	説明と注意事項
`aoss:DeleteAgent`	• DELETE /_plugins/_ml/agents/<agent_id>	エージェントを削除するアクセス許可を付与します。
`aoss:InvokeAgent`	• POST /_plugins/_ml/agents/<agent_id>/_execute	エージェントを呼び出し (実行) するアクセス許可を提供します。
`aoss:SearchAgents`	• POST /_plugins/_ml/agents/_search	エージェントを検索するアクセス許可を付与します。

サポートされている OpenSearch プラグイン

OpenSearch Serverless コレクションには、OpenSearch コミュニティから提供される以下のプラグインが、事前にパッケージされて提供されます。Serverless は、自動的にプラグインのデプロイと管理を実行します。

分析プラグイン

- ICU Analysis
- Japanese (kuromoji) Analysis
- Korean (Nori) Analysis
- Phonetic Analysis
- Smart Chinese Analysis
- Stempel Polish Analysis
- Ukrainian Analysis

マッパープラグイン

- Mapper Size
- Mapper Murmur3
- Mapper Annotated Text

スクリプトプラグイン

- Painless (検索クエリと集計のインラインスクリプトのみ。保存されたスクリプトと/_scriptsエンドポイントは OpenSearch Serverless ではサポートされていません)
- Expression
- Mustache

これらに加えて、OpenSearch Serverless には、モジュールとして出荷されるすべてのプラグインが含まれています。

Amazon OpenSearch Serverless のモニタリング

モニタリングは、Amazon OpenSearch Serverless およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 には、OpenSearch Serverless をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツール AWS が用意されています。

- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。

  例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- AWS CloudTrail は、 AWS アカウントによって行われた、またはそのアカウントに代わって実行された API コールと関連イベントをキャプチャします。このツールでは、ユーザーが指定した Amazon S3 バケットにログファイルを配信します。呼び出し元のユーザーとアカウント AWS、呼び出し元の送信元 IP アドレス、呼び出しの発生日時を特定できます。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。
- Amazon EventBridge は、OpenSearch Service ドメインの変更を記述したシステムイベントのストリームをほぼリアルタイムに配信します。特定のイベントを監視するルールを作成し、これらのイベントが発生した AWS のサービス ときに他の で自動アクションをトリガーできます。詳細については、Amazon EventBridge ユーザーガイドを参照してください。

Amazon CloudWatch を使用した OpenSearch Serverless のモニタリング

CloudWatch を使用して Amazon OpenSearch Serverless をモニタリングし、収集した未加工データを、ほぼリアルタイムの読み取り可能なメトリクスに変換できます。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。

また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

OpenSearch Serverless は、AWS/AOSS 名前空間に以下のメトリクスをレポートします。

メトリクス	説明
ActiveCollection	コレクションが有効化されているかどうかを示します。値が 1 の場合は、コレクションのステータスが ACTIVE であることを意味します。この値は、コレクションが正しく作成された時点で出力され、そのコレクションを削除するまで 1 を保ちます。このメトリクスでは、値が 0 になることはありません。 関連する統計情報: Max ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
DeletedDocuments	削除されたドキュメントの総数。 関連する統計情報: Average、Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒
IndexingOCU	コレクションデータを取り込むために使用された OpenSearch Compute Units (OCU) の数。このメトリクス

メトリクス	説明
	は、アカウントレベルで適用されます。どのコレクショングループにも属さないコレクションの使用のみを表します。 関連する統計情報: Sum ディメンション: ClientId 頻度: 60 秒
IndexingOCU	コレクションデータを取り込むために使用された OpenSearch Compute Units (OCU) の数。このメトリクスは、コレクショングループレベルに適用されます。 関連する統計情報: Sum ディメンション: ClientId、CollectionGroupId 、CollectionGroupName 頻度: 60 秒
IngestionDataRate	コレクションまたはインデックスに対するインデックス化レート (GiB /秒)。このメトリクスは、一括インデックス作成リクエストにのみ適用されます。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒

メトリクス	説明
IngestionDocumentErrors	コレクションまたはインデックスに対する取り込み中に発生した、ドキュメントエラーの総数。一括インデックス作成リクエストが成功すると、ライターはそのリクエストを処理し、リクエスト内で失敗したすべてのドキュメントについてエラーを出力します。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒
IngestionDocumentRate	コレクションまたはインデックスへの、ドキュメントの取り込みにおける 1 秒あたりのレート。このメトリクスは、一括インデックス作成リクエストにのみ適用されます。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒
IngestionRequestErrors	コレクションに対して発生した、一括インデックス作成リクエストエラーの総数。OpenSearch Serverless は、認証や可用性の問題など、何らかの理由で一括インデックス作成リクエストが失敗した場合に、このメトリックを出力します。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒

メトリクス	説明
IngestionRequestLatency	コレクションへの一括書き込みオペレーションにおけるレイテンシー (秒単位)。 関連する統計情報: Minimum、Maximum、Average ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
IngestionRequestRate	コレクションが受信した、一括書き込みオペレーションの総数。 関連する統計情報: Minimum、Maximum、Average ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
IngestionRequestSuccess	コレクションに対して成功した、インデックス作成オペレーションの総数。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
SearchableDocuments	コレクションまたはインデックス内にある、検索可能なドキュメントの合計数。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒

メトリクス	説明
SearchRequestErrors	コレクションに対し発生したクエリエラーの、1 分あたりの総数。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
SearchRequestLatency	コレクションに対し、検索オペレーションを完了するために必要な平均時間 (ミリ秒)。 関連する統計情報: Minimum、Maximum、Average ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
SearchOCU	コレクションデータの検索に使用された OpenSearch Compute Units (OCU) の数。このメトリクスは、アカウントレベルで適用されます。どのコレクショングループにも属さないコレクションの使用のみを表します。 関連する統計情報: Sum ディメンション: ClientId 頻度: 60 秒

メトリクス	説明
SearchOCU	コレクションデータの検索に使用された OpenSearch Compute Units (OCU) の数。このメトリクスは、コレクショングループレベルに適用されます。 関連する統計情報: Sum ディメンション: ClientId、CollectionGroupId 、CollectionGroupName 頻度: 60 秒
SearchRequestRate	コレクションに対する検索リクエストの、1 分あたりの総数。 関連する統計情報: Average、Maximum、Sum ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒
StorageUsedInS3	使用した Amazon S3 ストレージの量 (バイト)。OpenSearch Serverless は、インデックス付きデータをAmazon S3 に保存します。正確な値を取得するには、1 分単位で期間を選択します。この値は、消費された物理 S3 ストレージの合計を表します。これには、レプリケーション、メタデータ、内部インデックス作成構造のオーバーヘッドが含まれます。これは、 によって報告される論理インデックスサイズGET _cat/indices や、 マネジメントコンソールに表示される合計サイズとは異なる場合があります。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 、IndexId、IndexName 頻度: 60 秒

メトリクス	説明
VectorIndexBuildAcceleration0CU	ベクトルインデックス作成の高速化に使用される OpenSearch コンピューティングユニット (OCUs) の数。このメトリクスはコレクションレベルに適用されます。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 頻度: 60 秒
2xx, 3xx, 4xx, 5xx	コレクションに対するリクエストのうち、特定の HTTP レスポンスコード (2xx、3xx、4xx、5xx) で応答したものの数。 関連する統計情報: Sum ディメンション: ClientId、CollectionId 、CollectionName 頻度: 60 秒

を使用した OpenSearch Serverless API コールのログ記録 AWS CloudTrail

Amazon OpenSearch Serverless は AWS CloudTrail、Serverless のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。

CloudTrail は、OpenSearch Serverless のすべての API 呼び出しをイベントとしてキャプチャします。キャプチャされる対象としては、OpenSearch Service コンソールの Service セクションからの呼び出しや、OpenSearch Serverless API オペレーションへのコード呼び出しなどが含まれます。

証跡を作成する場合は、OpenSearch Serverless のイベントなど、Amazon S3 バケットに対する CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。

CloudTrail で収集された情報を使用して、OpenSearch Serverless に対し発行されたリクエスト、リクエスト元の IP アドレス、リクエスト作成者、リクエスト作成日時、その他の詳細情報などを確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail での OpenSearch Serverless の情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。OpenSearch Serverless でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

OpenSearch Serverless のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。

証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての OpenSearch Serverless アクションは CloudTrail によってログ記録されます。これらは、「OpenSearch Service API reference」(OpenSearch Service API リファレンス) に記載されています。たとえば、`CreateCollection`、`ListCollections`、`DeleteCollection` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。この ID 情報は以下のことを確認するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。

- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

CloudTrail での OpenSearch Serverless データイベント

データイベントでは、リソース上またはリソース内で実行されるリソースオペレーション (OpenSearch Serverless コレクションへの検索やインデックス作成など) についての情報が得られます。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、CloudTrail はデータイベントを口グ記録しません。CloudTrail [イベント履歴] にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail コンソール、または CloudTrail CloudTrail API オペレーションを使用して AWS CLI、`AWS::AOSS::Collection`リソースタイプのデータイベントを記録できます。データイベントを口グに記録する方法の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS マネジメントコンソールを使用したデータイベントの口グ記録」および「AWS Command Line Interfaceを使用したデータイベントの口グ記録」を参照してください。

`eventName`、`readOnly`、および `resources.ARN` フィールドでフィルタリングして、自分にとって重要なイベントのみを口グに記録するように高度なイベントセレクタを設定できます。オブジェクトの詳細については、「AWS CloudTrail API リファレンス」の「AdvancedFieldSelector」を参照してください。

Amazon OpenSearch Serverless のデータイベントエントリを理解する

以下の例で、次の操作を行います。

- `requestParameters` フィールドには、コレクションに対して行われた API コールに関する詳細が含まれます。これには、基本リクエストパス (クエリパラメータなし) が含まれます。
- `responseElements` フィールドには、リソースの変更時にリクエストの結果を示すステータスコードが含まれます。このステータスコードは、変更が正常に処理されたか、注意が必要かを追跡するのに役立ちます。
- OpenSearch Serverless は、IAM 認証を正常に完了したリクエストに対してのみ CloudTrail データイベントを記録します。

```
        "accessKeyId": "access-key",
        "userName": "",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AROA123456789EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "attributes": {
            "creationDate": "2025-08-15T22:57:38Z",
            "mfaAuthenticated": "false"
          },
          "sourceIdentity": "",
          "ec2RoleDelivery": "",
          "assumedRoot": ""
        },
        "identityProvider": "",
        "credentialId": ""
      },
      "eventTime": "2025-08-15T22:58:00Z",
      "eventSource": "aoss.amazonaws.com",
      "eventName": "Search",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "AWS Internal",
      "userAgent": "python-requests/2.32.3",
      "requestParameters": {
        "pathPrefix": "/_search"
      },
      "responseElements": null,
      "requestID": "2cfee788-EXAM-PLE1-8617-4018cEXAMPLE",
      "eventID": "48d43617-EXAM-PLE1-9d9c-f7EXAMPLE",
      "readOnly": true,
      "resources": [
        {
          "type": "AWS::AOSS::Collection",
          "ARN": "arn:aws:aoss:us-east-1:111122223333:collection/aab9texampletu45xh77"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "111122223333",
      "eventCategory": "Data"
    }
  ]
}
```

OpenSearch Serverless Management Events エントリを理解する

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail ログファイルには、1 つ以上のログエントリがあります。

イベントは、任意の送信元からの単一のリクエストを表します。これには、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

以下の例は、`CreateCollection` アクションを示す CloudTrail ログエントリです。


```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"AIDACKCEVSQ6C2EXAMPLE",
      "arn":"arn:aws:iam::123456789012:user/test-user",
      "accountId":"123456789012",
      "accessKeyId":"access-key",
      "sessionContext":{
         "sessionIssuer":{
            "type":"Role",
            "principalId":"AIDACKCEVSQ6C2EXAMPLE",
            "arn":"arn:aws:iam::123456789012:role/Admin",
            "accountId":"123456789012",
            "userName":"Admin"
         },
         "webIdFederationData":{

         },
         "attributes":{
            "creationDate":"2022-04-08T14:11:34Z",
            "mfaAuthenticated":"false"
         }
      }
   },
   "eventTime":"2022-04-08T14:11:49Z",
   "eventSource":"aoss.amazonaws.com",
   "eventName":"CreateCollection",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"AWS Internal",
```

```
   "userAgent":"aws-cli/2.1.30 Python/3.8.8 Linux/5.4.176-103.347.amzn2int.x86_64 exe/
x86_64.amzn.2 prompt/off command/aoss.create-collection",
   "errorCode":"HttpFailureException",
   "errorMessage":"An unknown error occurred",
   "requestParameters":{
      "accountId":"123456789012",
      "name":"test-collection",
      "description":"A sample collection",
      "clientToken":"d3a227d2-a2a7-49a6-8fb2-e5c8303c0718"
   },
   "responseElements": null,
   "requestID":"12345678-1234-1234-1234-987654321098",
   "eventID":"12345678-1234-1234-1234-987654321098",
   "readOnly":false,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "recipientAccountId":"123456789012",
   "eventCategory":"Management",
   "tlsDetails":{
      "clientProvidedHostHeader":"user.aoss-sample.us-east-1.amazonaws.com"
   }
}
```


Amazon EventBridge を用いた OpenSearch Serverless イベントのモニタリング

Amazon OpenSearch Service は Amazon EventBridge と統合され、ドメインに影響する特定のイベントを通知します。 AWS サービスからのイベントは、ほぼリアルタイムで EventBridge に配信されます。同じイベントが Amazon EventBridge の先行である Amazon CloudWatch Events にも送信されます。ルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。自動的にアクティブ化できるアクションの例には、以下が含まれます。

- AWS Lambda 関数の呼び出し
- Amazon EC2 Run Command の呼び出し
- Amazon Kinesis Data Streams へのイベントの中継
- AWS Step Functions ステートマシンのアクティブ化
- Amazon SNS トピックまたは Amazon SQS キューの通知

詳細については、Amazon EventBridge ユーザーガイドの「Amazon EventBridge の開始方法」を参照してください。

通知の設定

AWS ユーザー通知を使用して、OpenSearch Serverless イベントが発生したときに通知を受け取ることができます。イベントは、OpenSearch Serverless 環境内の変化 (OCU 使用量の上限に達したときなど) のインジケータです。 Amazon EventBridge はイベントを受け取り、通知を AWS マネジメントコンソール 通知センターとユーザーが指定した配信チャネルにルーティングします。指定したルールにイベントが一致すると、通知を受け取ります。

OpenSearch コンピューティングユニット (OCU) イベント

OpenSearch Serverless は、OCU 関連の次のイベントのいずれかが発生したときに、EventBridge にイベントを送信します。

OCU の使用量が上限に近づいている

OpenSearch Serverless は、検索またはインデックス OCU の使用量が容量制限の 75% に達するとこのイベントを送信します。OCU の使用量は、設定した容量制限と現在の OCU 消費量に基づいて計算されます。

例

このタイプのイベント (検索 OCU) の例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "OCU Utilization Approaching Max Limit",
  "source": "aws.aoss",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "eventTime" : 1678943345789,
    "description": "Your search OCU usage is at 75% and is approaching the configured
 maximum limit."
  }
}
```

このタイプのイベント (インデックス OCU) の例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "OCU Utilization Approaching Max Limit",
  "source": "aws.aoss",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "eventTime" : 1678943345789,
    "description": "Your indexing OCU usage is at 75% and is approaching the configured
 maximum limit."
  }
```

OCU の使用量が上限に達した

OpenSearch Serverless は、検索またはインデックス OCU の使用量が容量制限の 100% に達するとこのイベントを送信します。OCU の使用量は、設定した容量制限と現在の OCU 消費量に基づいて計算されます。

例

このタイプのイベント (検索 OCU) の例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "OCU Utilization Reached Max Limit",
  "source": "aws.aoss",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "eventTime" : 1678943345789,
    "description": "Your search OCU usage has reached the configured maximum limit."
  }
}
```

このタイプのイベント (インデックス OCU) の例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "OCU Utilization Reached Max Limit",
  "source": "aws.aoss",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "eventTime" : 1678943345789,
    "description": "Your indexing OCU usage has reached the configured maximum limit."
  }
}
```

Amazon OpenSearch Service ドメインの作成と管理

この章では、Amazon OpenSearch Service ドメインを作成して管理する方法について説明します。ドメインは、オープンソースの OpenSearch クラスターと同等の AWSがプロビジョニングされたものです。ドメインを作成するときは、その設定、インスタンスタイプ、インスタンス数、ストレージ割り当てを指定します。オープンソースクラスターの詳細については、OpenSearch ドキュメントの「Creating a cluster」を参照してください。

開始方法チュートリアルの簡単な手順とは異なり、この章ではすべてのオプションについて説明し、関連するリファレンス情報を提供します。OpenSearch Service コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDKs の手順を使用して、各手順を完了できます。

OpenSearch Service ドメインの作成

このセクションでは、OpenSearch Service コンソールを使用するか、 `create-domain` コマンド AWS CLI で を使用して OpenSearch Service ドメインを作成する方法について説明します。

OpenSearch Service ドメインの作成 (コンソール)

次の手順に従って、コンソールを使用することで OpenSearch Service ドメインを作成します。

OpenSearch Service ドメインを作成するには (コンソール)

1. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. [ドメインの作成] を選択します。
4. [ドメイン名] には、ドメイン名を入力します。名前は次の基準を満たしている必要があります。
   - アカウントと に固有 AWS リージョン
   - 先頭が小文字
   - 3～28 文字
   - 小文字の a～z、0～9 の数字、ハイフン (-) のみ含まれる
5. ドメインの作成方法は、[標準作成] を選択します。
6. [テンプレート] で、ドメインの目的に最も一致するオプションを選択します。

- 高可用性とパフォーマンスを必要とするワークロード向けの [本番] ドメイン。これらのドメインは、可用性を高めるために、マルチ AZ (スタンバイの有無にかかわらず) と専用マスターノードを使用します。
- 開発またはテスト向けの [開発/テスト]。これらのドメインは、マルチ AZ (スタンバイの有無にかかわらず) または単一のアベイラビリティーゾーンを使用できます。

> **Important**
>
> デプロイタイプが異なると、後続のページのオプションが異なります。これらのステップにはすべてのオプションが含まれています。

7. [デプロイオプション] で、[スタンバイが有効のドメイン] を選択して 3-AZ ドメインを設定し、いずれかのゾーンのノードがスタンバイとして予約されます。このオプションでは、指定されたデータノード数、マスターノード数、インスタンスタイプ、レプリカ数、ソフトウェアアップデート設定など、いくつかのベストプラクティスが適用されます。
8. バージョンでは、使用する OpenSearch また従来の Elasticsearch OSS のバージョンを選択します。最新バージョンの OpenSearch を選択することをお勧めします。詳細については、「the section called “サポートされているバージョンの Elasticsearch および OpenSearch”」を参照してください。

   (オプション) ドメインに OpenSearch バージョンを選択した場合は、[互換モードを有効にする] を選択して OpenSearch のバージョンを 7.10 と報告します。これにより、接続する前にバージョンをチェックする特定の Elasticsearch OSS クライアントおよびプラグインが、サービスとの動作を続行できます。
9. [インスタンスタイプ] では、データノードのインスタンスタイプを選択します。詳細については、「the section called “サポートされるインスタンスタイプ”」を参照してください。

   > **Note**
   >
   > すべてのアベイラビリティーゾーンですべてのインスタンスタイプがサポートされているわけではありません。スタンバイ付きまたはスタンバイなしのマルチ AZ を選択する場合は、R5 や I3 などの現行世代のインスタンスタイプを選択することをお勧めします。
10. [ノードの数] で、データノードの数を選択します。

最大値については、「OpenSearch Service のドメインとインスタンスクォータ」を参照してください。単一ノードのクラスターは、開発とテスト用に適切ですが、本稼働のワークロードには使用しないでください。ガイダンスについては、「the section called “ドメインのサイジング”」および「the section called “マルチ AZ ドメインの設定”」を参照してください。

> **Note**
>
> (オプション) 専有コーディネーターノードは、すべての OpenSearch バージョンと ElasticSearch バージョン 6.8～7.10 をサポートしています。専用コーディネーターノードは、専用クラスターマネージャーが有効になっているドメインで使用できます。専用コーディネーターノードを有効にするには、インスタンスのタイプと数を選択します。ベストプラクティスとして、専用コーディネーターノードのインスタンスファミリーをデータノード (Intel ベースのインスタンスまたは Graviton ベースのインスタンス) と同じままにする必要があります。

11. [ストレージタイプ] で、[Amazon EBS] を選択します。一覧で利用できるボリュームタイプは、選択したインスタンスタイプに応じて異なります。特に大きなドメインを作成する際のガイダンスについては、「the section called “ペタバイトスケール”」を参照してください。
12. [EBS] ストレージでは、次の追加設定を行います。選択したボリュームのタイプによっては、一部の設定が表示されない場合があります。

設定	説明
EBS ボリュームタイプ	汎用 (SSD) – gp3 と 汎用 (SSD) – gp2 のいずれか、または旧世代の プロビジョンド IOPS (SSD)、およびマグネティック (標準) を選択します。
ノードあたりの EBS ストレージサイズ	各データノードに接続する EBS ボリュームのサイズを入力します。 [EBS ボリュームサイズ] はノードあたりのサイズです。OpenSearch Service ドメインの合計クラスターサイズは、データノードの数を EBS ボリュームサイズで乗算して計算できます。EBS ボリュームの最小サイズと最大サイズは、指定された EBS ボリュームタイプとそれがアタッチされるインスタンスタ

設定	説明
	イプの両方によって異なります。詳細については、「EBS ボリュームサイズの制限」を参照してください。
プロビジョンド IOPS	Provisioned IOPS SSD ボリュームタイプを選択した場合は、ボリュームがサポートできる I/O オペレーション/秒 (IOPS) を入力します。

13. (オプション) gp3 ボリュームタイプを選択した場合は、[詳細設定] を展開し、ストレージの料金に含まれるものを超えて、追加の IOPS (データノードあたりプロビジョニングされる 3 TiB ボリュームサイズそれぞれに対して最大 16,000 MiB/秒) とスループット (データノードあたりプロビジョニングされる 3 TiB ボリュームサイズそれぞれに対して最大 1,000) を指定します。これには追加料金がかかります。詳細については、「Amazon OpenSearch Service の料金」を参照してください。
14. (オプション) UltraWarm ストレージを有効にするには、[UltraWarm データノードを有効にする] を選択します。各インスタンスタイプには、アドレス可能なストレージの最大容量があります。この量に、アドレス可能なウォームストレージの合計のウォームデータノードの数を乗算します。
15. (オプション) [コールドストレージ] を有効にするには、[コールドストレージを有効にする] を選択します。コールドストレージを有効にするには、UltraWarm を有効にする必要があります。
16. スタンバイ付きマルチ AZ を使用する場合、3 つの専用マスターノードが既に有効になっています。必要なマスターノードのタイプを選択します。スタンバイなしのマルチ AZ のドメインを選択した場合は、[専用マスターノードを有効にする] を選択し、必要なマスターノードのタイプと数を選択します。専用マスターノードは、クラスターの安定性を高めます。また、インスタンス数が 10 を超えるドメインに必要です。本番稼働用ドメインには、3 つの専用マスターノードをお勧めします。

   > **Note**
   >
   > 専用マスターノードおよびデータノードの異なるインスタンスタイプを選択できます。たとえば、データノードの汎用またはストレージ最適化インスタンスを選択できますが、専用マスターノードのコンピューティング最適化インスタンスは選択できません。

17. (オプション) OpenSearch または Elasticsearch 5.3 以降を実行しているドメインでは、[スナップショット設定] は無関係です。自動化されたスナップショットの詳細については、「the section called “インデックススナップショットの作成”」を参照してください。

18. 標準エンドポイントの `https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-east-1.es.amazonaws.com` ではなく、カスタムエンドポイントを使用する場合、[カスタムエンドポイントを有効にする] を選択し、名前と証明書を入力します。詳細については、「the section called “カスタムエンドポイントの作成”」を参照してください。
19. [ネットワーク] セクションで、[VPC アクセス] または [パブリックアクセス] を選択します。[パブリックアクセス] を選択した場合は、次のステップに進みます。[VPC access] (VPC アクセス) を選択する場合は、[prerequisites] (前提条件) を満たしていることを確認してから、次の設定を行います。

設定	説明
VPC	使用する仮想プライベートクラウド (VPC) の ID を選択します。VPC とドメインが同じ AWS リージョンに存在している必要があり、テナントが [デフォルト] に設定されている VPC を選択する必要があります。OpenSearch Service は専有テナントを使用する VPC をまだサポートしていません。
サブネット	サブネットを選択します。マルチ AZ を有効にした場合、2 つまたは 3 つのサブネットを選択する必要があります。OpenSearch Service は VPC エンドポイントと Elastic Network Interface (ENI) をサブネットに配置します。 サブネット内のネットワークインターフェイス用に十分な IP アドレスを予約する必要があります。詳細については、「VPC サブネットで IP アドレスをリザーブする」を参照してください。
セキュリティグループ	必要なアプリケーションが、ドメインによって公開されているポート (80 または 443) およびプロトコル (HTTP または HTTPS) で OpenSearch Service ドメインにアクセスできるようにする、1 つ以上の VPC セキュリティグループを選択します。詳細については、「the section called “VPC サポート”」を参照してください。
IAM ロール	デフォルトのロールを維持します。OpenSearch Service では、この事前定義されたロール (サービスにリンクされたロールとも呼ばれる) を使用して VPC にアクセスし、VPC のサブネットに VPC エンドポイントとネットワークインターフェースを配置します。詳細については、「VPC アクセス用のサービスにリンクされたロール」を参照してください。

設定	説明
VPC 出力	(オプション) Egress を有効にする を選択して、パブリックインターネットではなく VPC 経由でドメインの Egress トラフィックをルーティングします。詳細については、「the section called “VPC を介したドメイン出力トラフィックのルーティング”」を参照してください。
IP アドレスタイプ	IP アドレスタイプとして、デュアルスタックまたは IPv4 を選択します。デュアルスタックでは、IPv4 と IPv6 のアドレスタイプ間でドメインリソースを共有できます。これが推奨オプションです。IP アドレスタイプをデュアルスタックに設定した場合、後でアドレスタイプを変更することはできません。

20. きめ細かなアクセスコントロールを有効または無効にします。

    - ユーザー管理に IAM を使用する場合は、[IAM ARN をマスターユーザーとして設定] を選択し、IAM ロールの ARN を指定します。
    - 内部ユーザーデータベースを使用する場合は、[マスターユーザーの作成] を選択し、ユーザー名とパスワードを指定します。

    どちらのオプションを選択しても、マスターユーザーはクラスター内のすべてのインデックスとすべての OpenSearch API にアクセスできます。選択するオプションのガイダンスについては、「the section called “主要なコンセプト”」を参照してください。

    きめ細かなアクセスコントロールを無効にしても、ドメインを VPC 内に配置するか、制限付きアクセスポリシーを適用するか、またはその両方を行うことで、ドメインへのアクセスをコントロールできます。きめ細かなアクセスコントロールを使用するには、ノード間の暗号化と保存時の暗号化を有効にする必要があります。

    > **Note**
    >
    > ドメイン上のデータを保護するために、きめ細かなアクセスコントロールを有効にすることを強くお勧めします。きめ細かなアクセスコントロールにより、クラスター、インデックス、ドキュメント、フィールドの各レベルでセキュリティが提供されます。

21. (オプション) OpenSearch Dashboards で SAML 認証を使用する場合は、[SAML 認証を有効にする] を選択して、ドメインの SAML オプションを設定します。手順については、「the section called “OpenSearch Dashboards の SAML 認証”」を参照してください。

22. (オプション) OpenSearch Dashboards の Amazon Cognito 認証を使用する場合は、[Amazon Cognito 認証を有効にする] を選択します。次に、OpenSearch Dashboards の認証に使用する Amazon Cognito ユーザープールと ID プールを選択します。これらのリソースの作成のガイダンスについては、「the section called “OpenSearch Dashboards の Amazon Cognito 認証”」を参照してください。
23. (オプション) IAM Identity Center (IDC) 認証を使用して既存の ID ソースを接続し、ユーザーの共通ビューを AWS アプリケーションに付与する場合は、IAM Identity Center で認証された API アクセスを有効にする を選択します。詳細については、「IAM アイデンティティセンターユーザーガイド」の「Trusted identity propagation overview」を参照してください。
24. (オプション) これらの機能を使用する場合は、[高度な機能] セクションで [自然言語クエリ生成と Amazon Q Developer 機能を有効にする] をオンにしておきます。

    a. 拡張ベクトル検索オプションで [エンジンオプションとして S3 ベクトルを有効にする] を選択します。詳細については、「Amazon S3 ベクトルエンジンを使用した高度な検索機能」を参照してください。

    b. 拡張ベクトル検索オプションで GPU アクセラレーションを有効にするを選択します。詳細については、「ベクトルインデックス作成の GPU アクセラレーション」を参照してください。
25. [アクセスポリシー] で、アクセスポリシーを選択するか、独自のポリシーを設定します。カスタムポリシーを作成することを選択した場合は、自分で設定することも、別のドメインからインポートすることもできます。詳細については、「the section called “Identity and Access Management”」を参照してください。

    > **Note**
    >
    > VPC アクセスを有効にした場合、IP ベースのポリシーは使用できません。代わりに、どの IP アドレスがドメインにアクセスできるかを制御するセキュリティグループを使用できます。詳細については、「the section called “VPC ドメインのアクセスポリシーについて”」を参照してください。
26. (オプション) ドメインへのすべてのリクエストが HTTPS 経由で到着することを要求するには、[ドメインへのすべてのトラフィックに HTTPS が必要] を選択します。ノード間の暗号化を有効にするには、[ノード間の暗号化] を選択します。詳細については、「the section called “ノード間の暗号化”」を参照してください。保管中のデータの暗号化を有効にするには、[保管時のデータの暗号化を有効にする] を選択します。スタンバイ付きマルチ AZ デプロイオプションを選択した場合、これらのオプションは既に選択されています。

27. (オプション) AWS 所有キーを使用する を選択して、OpenSearch Service がユーザーに代わって AWS KMS 暗号化キーを作成する (または作成済みのキーを使用する) ようにします。それ以外の場合は、独自の KMS キーを選択します。詳細については、「the section called “保管中の暗号化”」を参照してください。
28. [オフピークウィンドウ] で、ブルー/グリーンデプロイを必要とするサービスソフトウェアの更新と自動調整の最適化をスケジュールする開始時刻を選択します。オフピークの時間帯に更新を行うことにより、トラフィックの多い時間帯にクラスターの専用マスターノードにかかる負荷を、最小限に抑えることができます。
29. Auto-Tune では、速度と安定性を向上させるために、OpenSearch Service がドメインに対するメモリ関連の設定変更を提案できるようにするかどうかを選択します。詳細については、「the section called “Auto-Tune”」を参照してください。

    (オプション) [オフピークウィンドウ] を選択すると、自動調整によってドメインが更新される定期的なウィンドウがスケジュールされます。
30. (オプション) [自動ソフトウェア更新] を選択して、自動ソフトウェア更新を有効にします。
31. (オプション) ドメインを説明するタグを追加して、その情報を分類およびフィルタリングできるようにします。詳細については、「the section called “ドメインのタグ付け”」を参照してください。
32. (オプション) クラスターの 詳細設定 を展開して設定します。これらのオプションの概要については、「the section called “高度なクラスター設定”」を参照してください。
33. [作成] を選択します。

OpenSearch Service ドメインの作成 (AWS CLI)

コンソールを使用して OpenSearch Service ドメインを作成する代わりに、 AWS CLIを使用できます。構文については、AWS CLI コマンドリファレンスの「Amazon OpenSearch Service」を参照してください。

コマンド例

この最初の例は、以下の OpenSearch Service ドメイン設定を示しています。

- OpenSearch バージョン 1.2 で、mylogs という名前の OpenSearch Service ドメインを作成します。
- `r6g.large.search` インスタンスタイプの 2 つのインスタンスをドメインに追加する
- 各データノードのストレージに 100 GiB 汎用 (SSD) gp3 EBS ボリュームを使用する

- 単一の IP アドレス 192.0.2.0/32 からのみ匿名アクセスを許可する

```
aws opensearch create-domain \
    --domain-name mylogs \
    --engine-version OpenSearch_1.2 \
    --cluster-config  InstanceType=r6g.large.search,InstanceCount=2 \
    --ebs-options
 EBSEnabled=true,VolumeType=gp3,VolumeSize=100,Iops=3500,Throughput=125 \
    --access-policies '{"Version": "2012-10-17",        "Statement": [{"Action":
 "es:*", "Principal":"*","Effect": "Allow", "Condition": {"IpAddress":{"aws:SourceIp":
["192.0.2.0/32"]}}}]}'
```

次の例は、以下の OpenSearch Service ドメイン設定を示しています。

- Elasticsearch バージョン 7.10 では、mylogs という名前の OpenSearch Service ドメインが作成されます。
- r6g.large.search インスタンスタイプの 6 つのインスタンスをドメインに追加する
- 各データノードのストレージに 100 GiB 汎用 (SSD) gp2 EBS ボリュームを使用する
- サービスへのアクセスを、ユーザーの AWS アカウント ID で識別される 1 人のユーザーに制限します: 555555555555
- 3 つのアベイラビリティーゾーンへのインスタンスを分散する

```
aws opensearch create-domain \
    --domain-name mylogs \
    --engine-version Elasticsearch_7.10 \
    --cluster-config
 InstanceType=r6g.large.search,InstanceCount=6,ZoneAwarenessEnabled=true,ZoneAwarenessConfig={A
 \
    --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=100 \
    --access-policies '{"Version": "2012-10-17",        "Statement": [ { "Effect":
 "Allow", "Principal": {"AWS": "arn:aws:iam::555555555555:root" }, "Action":"es:*",
 "Resource": "arn:aws:es:us-east-1:555555555555:domain/mylogs/*" } ] }'
```

次の例は、以下の OpenSearch Service ドメイン設定を示しています。

- OpenSearch バージョン 1.0 で、mylogs という名前の OpenSearch Service ドメインを作成します。
- r6g.xlarge.search インスタンスタイプの 10 個のインスタンスをドメインに追加する

- 専用マスターノードとして機能する r6g.large.search インスタンスタイプの 3 つのインスタンスをドメインに追加する
- ストレージに 100 GiB プロビジョンド IOPS EBS ボリュームを使用し、各データノードに 1000 IOPS のベースラインパフォーマンスを設定する
- アクセスを単一のユーザーと単一のサブリソース _search API に制限する

```
aws opensearch create-domain \
    --domain-name mylogs \
    --engine-version OpenSearch_1.0 \
    --cluster-config
 InstanceType=r6g.xlarge.search,InstanceCount=10,DedicatedMasterEnabled=true,DedicatedMasterTyp
 \
    --ebs-options EBSEnabled=true,VolumeType=io1,VolumeSize=100,Iops=1000 \
    --access-policies '{"Version": "2012-10-17",        "Statement": [ { "Effect":
 "Allow", "Principal": { "AWS": "arn:aws:iam::555555555555:root" }, "Action": "es:*",
 "Resource": "arn:aws:es:us-east-1:555555555555:domain/mylogs/_search" } ] }'
```

> **Note**
>
> OpenSearch Service ドメインを作成する際に、同じ名前のドメインが既に存在しても、CLI からエラーは報告されません。その代わりに、既存ドメインの詳細が返されます。

OpenSearch Service ドメイン (AWS SDKsの作成

AWS SDKs (Android および iOS SDKs を除く) は、 を含む Amazon OpenSearch Service API リファレンスで定義されているすべてのアクションをサポートしますCreateDomain。サンプルコードについては、「the section called “ AWS SDKsの使用”」を参照してください。 AWS SDKsAWS「 Software Development Kits」を参照してください。

OpenSearch Service ドメインの作成 (AWS CloudFormation)

OpenSearch Service は と統合されています。これは AWS CloudFormation、 AWS リソースとインフラストラクチャの作成と管理に費やす時間を短縮できるように、リソースのモデル化とセットアップに役立つサービスです。作成する OpenSearch ドメインを説明するテンプレートを作成すれば、CloudFormation がユーザーに向けてドメインのプロビジョニングと設定を自動的に行います。OpenSearch ドメインの JSON テンプレートと YAML テンプレートの例を含む詳細について

は、AWS CloudFormation ユーザーガイドの「Amazon OpenSearch Service リソースタイプのリファレンス」を参照してください。

アクセスポリシーの設定

Amazon OpenSearch Service には、OpenSearch Service ドメインへのアクセスを設定するいくつかの方法が用意されています。詳細については、「the section called “Identity and Access Management”」および「the section called “きめ細かなアクセスコントロール”」を参照してください。

コンソールは、ドメインの特定のニーズに対応してカスタマイズできる事前定義のアクセスポリシーを提供します。他の OpenSearch Service ドメインからアクセスポリシーをインポートすることもできます。上記のアクセスポリシーが VPC アクセスを操作する方法についての詳細は、「the section called “VPC ドメインのアクセスポリシーについて”」を参照してください。

アクセスポリシーを設定するには (コンソール)

1. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. ナビゲーションペインの [ドメイン] で、更新するドメインを選択します。
4. [アクション] から [セキュリティ設定の編集] を選択します。
5. アクセスポリシーの JSON を編集するか、事前設定済みのオプションをインポートします。
6. [Save changes] (変更の保存) をクリックします。

Migration Assistant を使用した OpenSearch Service への移行

Migration Assistant for Amazon OpenSearch Service は、セルフマネージド型の Elasticsearch クラスターまたは OpenSearch クラスターから OpenSearch Service に移行するプロセスを簡素化する包括的なソリューションです。このツールキットは、移行作業の複雑さに対処しながら、データの整合性を確保し、移行後のパフォーマンスを検証します。

概要

でのproof-of-conceptの設定 AWS、本番ワークロードの移行、最新の OpenSearch バージョンへのアップグレードなど、Migration Assistant には、OpenSearch 移行の可能性を最大限に活用するためのstep-by-stepガイダンス、ベストプラクティス、ツールが用意されています。

Migration Assistant は、主に以下の機能を提供します。

- メタデータの移行 – インデックス設定、タイプマッピング、インデックステンプレート、エイリアスなどのクラスターメタデータを移行させます
- データ移行 – 既存のデータをレガシークラスターから OpenSearch Service ドメインに移行させます
- ライブトラフィック処理 – 最小限のレイテンシーで、セルフマネージドクラスターからのライブトラフィックをインターセプトして OpenSearch Service ドメインにリダイレクトします
- トラフィックレプリケーション – ターゲットクラスターの本番稼働用トラフィックをレプリケートして、精度とパフォーマンスを検証します
- パフォーマンステスト – リクエストパターンのキャプチャと再生により実際のトラフィックをシミュレーションし、これによりシステムパフォーマンスをファインチューニングします
- グローバル可用性 - グローバルリーチとスケーラビリティのために最も一般的な AWS リージョンにデプロイします

Migration Assistant は、Elasticsearch バージョン 6.x および 7.x、OpenSearch 1.x および 2.x からの移行をサポートします。詳細については、「Supported migration paths」を参照してください。

> **Note**
>
> Migration Assistant に関する以下の追加情報を参照してください。
>
> - このツールは、マルチホップ移行 (1 ホップでの Elasticsearch 5.x から OpenSearch Service 3.x への移行など) をサポートしています。
> - 移行はロールバック可能です。
> - 一部のユースケースでは、ツールはダウンタイムをほとんど、またはまったく必要としません。
> - このツールは、ソースクラスターに影響を与えることなく、高性能なバックフィルを提供します。

移行シナリオ

Migration Assistant は、以下の移行シナリオを処理するように設計されています。

メタデータの移行

インデックス設定、エイリアス、テンプレートなどのクラスターメタデータをソースクラスターからターゲット OpenSearch Service ドメインに移行します。

バックフィル移行

既存のデータまたは履歴データをソースクラスターからターゲット OpenSearch Service ドメインに移行し、移行中にもすべての重要なデータが保持されるようにします。

ライブトラフィックの移行

ソースクラスターからターゲット OpenSearch Service ドメインに継続中のライブトラフィックをレプリケートすることで、移行中のサービスの可用性が維持されます。

> **Important**
>
> 移行戦略はすべてのケースに適用できる訳ではありません。Migration Assistant はエンジニアリングのベストプラクティスに基づくガイダンスを提供しますが、本番環境のワークロードを移行する前に、特定の要件を評価し、徹底的にテストする必要があります。

Migration Assistant の開始方法

Migration Assistant for Amazon OpenSearch Service は、包括的なドキュメント、デプロイテンプレート、ソースコードを含む AWS ソリューションとして利用できます。Migration Assistant の使用を開始するには

Migration Assistant リソースにアクセスするには

1. 「Migration Assistant for Amazon OpenSearch Service solution overview」のソリューションドキュメント全体を確認してください。
2. デプロイ計画ガイドを読んで、コストと要件を理解してください。
3. デプロイ手順を使用してソリューションをデプロイし、使用ガイドに従って移行を実行します。

デベロッパーおよび上級ユーザー向けの Migration Assistant ソースコードと追加ドキュメントは、OpenSearch 移行 GitHub リポジトリで入手できます。

ソリューションアーキテクチャ

Migration Assistant は AWS、 にデプロイすると、複数の AWS サービスを使用して包括的な移行ソリューションを提供します。

- AWS CloudFormation – Migration Assistant をデプロイして設定するための Infrastructure as Code (IaC) テンプレートを提供します
- OpenSearch Service – 移行する検索および分析ワークロードの対象となるサービス
- Amazon Managed Streaming for Apache Kafka – 耐久性のあるストレージと HTTP トラフィックの再利用のためのストリーム処理機能を提供します
- Amazon Elastic Container Service – 安全かつスケーラブルなコンテナ内で移行マネジメントコンソールとトラフィックリプレイヤーを実行します
- Amazon Elastic File System – ソースクラスターとターゲットクラスターからのリクエストデータとレスポンスデータを格納するスケーラブルな永続的ストレージを提供します
- Amazon Simple Storage Service – 過去のバックフィルタスクと Infrastructure as Code コンテンツのスナップショットを保存します

アーキテクチャの詳細については、「Migration Assistant architecture details」を参照してください。

高度なクラスター設定

詳細オプションを使用して、次のように設定します。

リクエストボディのインデックス

HTTP リクエストボディ内で、インデックスへの明示参照を許可するかどうかを指定します。このプロパティを `false` に設定すると、ユーザーがサブリソースのアクセスコントロールをバイパスできなくなります。デフォルトでは、値は `true` に設定されます。詳細については、「the section called “詳細オプションと API に関する考慮事項”」を参照してください。

フィールドデータのキャッシュ割り当て

フィールドデータに割り当てられる Java ヒープスペースの割合を指定します。デフォルトでは、この設定は JVM ヒープの 20% です。

> **Note**
>
> 多くのお客様が、ローテーションするインデックスのクエリを毎日実行しています。`indices.fielddata.cache.size` を使用してベンチマークテストを始めることをお勧めします。これらのほとんどのユースケースでは JVM ヒープを 40% に設定してください。非常に大きいインデックスでは、さらに大きいフィールドデータキャッシュが必要になることがあります。

句の最大数

Lucene のブールクエリで許可される句の最大数を指定します。デフォルト値は 1,024 です。クエリに含まれる句の数が最大数を超えていると、`TooManyClauses` エラーが発生します。詳細については、Lucene のドキュメントを参照してください。

Amazon OpenSearch Service の自動セマンティックエンリッチメント

序章

Amazon OpenSearch Service は、他の従来の検索エンジンと同様に、word-to-word一致 (辞書検索) を使用して結果を検索します。このアプローチは、製品コードやモデル番号などの特定のクエリに適していますが、ユーザーの意図を理解することが重要になる抽象検索には苦労します。例えば、「ビーチの靴」を検索すると、辞書検索はカタログアイテム内の個々の単語「靴」、「ビーチ」、「for」、「the」に一致し、正確な検索語を含まない「耐水性サンドラル」や「サーフの綱領」などの関連製品が欠落している可能性があります。

自動セマンティックエンリッチメントは、キーワード一致と検索の背後にあるコンテキスト上の意味の両方を考慮することで、この制限を解決します。この機能は検索インテントを理解し、検索の関連性を最大 20% 向上させます。インデックスのテキストフィールドに対してこの機能を有効にして、検索結果を強化します。

> **Note**
>
> 自動セマンティックエンリッチメントは、バージョン 2.19 以降を実行している OpenSearch Service ドメインで使用できます。さらに、OpenSearch バージョン 2.19 のドメインも、最

新のサービスソフトウェアバージョン更新に含まれている必要があります。現在、機能はパブリックドメインで使用でき、VPC ドメインはサポートされていません。

モデルの詳細とパフォーマンスベンチマーク

この機能は、基盤となるモデルを公開せずにバックグラウンドの技術的な複雑さを処理しますが、簡単なモデルの説明とベンチマーク結果を通じて透明性を提供し、重要なワークロードでの機能の採用について情報に基づいた意思決定に役立ちます。

自動セマンティックエンリッチメントは、カスタムのファインチューニングを必要とせずに効果的に機能する、事前トレーニング済みのサービスマネージドスパースモデルを使用します。このモデルは、指定したフィールドを分析し、さまざまなトレーニングデータから学習した関連付けに基づいてスパースベクトルに展開します。拡張された用語とその重要度の重みは、効率的な取得のためにネイティブの Lucene インデックス形式で保存されます。このプロセスは、データの取り込み中にのみエンコードが行われるドキュメント専用モードを使用して最適化されています。検索クエリは、スパースモデルを介して処理されるのではなく、単にトークン化されるため、ソリューションのコスト効率とパフォーマンスの両方が向上します。

機能開発中のパフォーマンス検証では、平均 334 文字のパッセージを含む MS MARCO パッセージ取得データセットを使用しました。関連性スコアについては、英語コンテンツの BEIR ベンチマークでは最初の 10 件の検索結果 (ndcg@10) の平均正規化割引累積ゲイン (NDCG) を、多言語コンテンツの MIRACL では平均 ndcg@10 を測定しました。クライアント側の 90 パーセンタイル (p90) 測定値と検索レスポンス p90 が値を取り、レイテンシーを評価しました。これらのベンチマークは、検索の関連性と応答時間の両方のベースラインパフォーマンス指標を提供します。主なベンチマーク番号は次のとおりです -

- 英語 - 辞書検索よりも 20% の関連性の向上。また、辞書検索よりも P90 検索レイテンシーが 7.7% 短縮されました (BM25 は 26 ミリ秒、自動セマンティックエンリッチメントは 24 ミリ秒) 。
- 多言語 - レキシカル検索に比べて関連性が 105% 向上しましたが、P90 検索レイテンシーはレキシカル検索に比べて 38.4% 増加しました (BM25 は 26 ミリ秒、自動セマンティックエンリッチメントは 36 ミリ秒) 。

各ワークロードの固有の性質を考慮して、実装を決定する前に、独自のベンチマーク基準を使用して開発環境でこの機能を評価することをお勧めします。

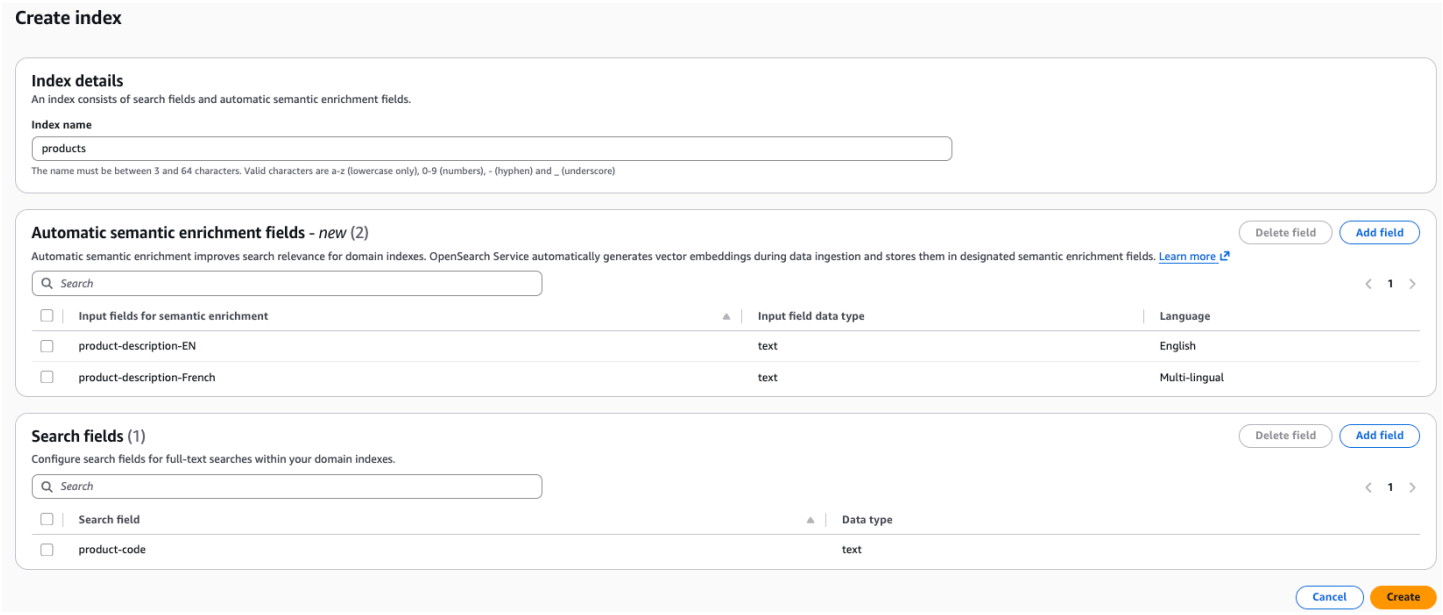
サポートされている言語

この機能は英語をサポートしています。さらに、このモデルはアラビア語、ベンガル語、中国語、フィンランド語、フランス語、ヒンディー語、インドネシア語、日本語、韓国語、ペルシャ語、ロシア語、スペイン語、スワヒリ語、テルグ語もサポートしています。

ドメインの自動セマンティックエンリッチメントインデックスを設定する

テキストフィールドの自動セマンティックエンリッチメントを有効にしてインデックスを設定することは簡単で、新しいインデックスの作成時にコンソール、APIs、 CloudFormation テンプレートを使用して管理できます。既存のインデックスに対して有効にするには、テキストフィールドに対してセマンティック自動エンリッチメントを有効にしてインデックスを再作成する必要があります。

コンソールエクスペリエンス - AWS コンソールでは、自動セマンティックエンリッチメントフィールドを使用してインデックスを簡単に作成できます。ドメインを選択すると、コンソールの上部にインデックスの作成ボタンが表示されます。インデックスの作成ボタンをクリックすると、自動セマンティックエンリッチメントフィールドを定義するオプションが表示されます。1 つのインデックスで、英語と多言語の自動セマンティックエンリッチメントと、辞書フィールドを組み合わせることができます。



API エクスペリエンス - AWS コマンドラインインターフェイス (AWS CLI) を使用して自動セマンティックエンリッチメントインデックスを作成するには、create-index コマンドを使用します。

```
aws opensearch create-index \
--domain-name [domain_name] \
```

```
--index-name [index_name] \
--index-schema [index_body] \
```

次の index-schema の例では、 title_semantic フィールドのフィールドタイプを text に設定し、パラメータ semantic_enrichment を status ENABLED に設定します。semantic_enrichment パラメータを設定すると、 title_semantic フィールドで自動セマンティックエンリッチメントが有効になります。language_options フィールドを使用して、英語または MULTI-LINGUAL を指定できます。

```
    aws opensearch create-index \
    --id XXXXXXXXX \
    --index-name 'product-catalog' \
    --index-schema '{
    "mappings": {
        "properties": {
            "product_id": {
                "type": "keyword"
            },
            "title_semantic": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "ENABLED",
                    "language_options": "english"
                }
            },
            "title_non_semantic": {
                "type": "text"
            }
        }
    }
}'
```

作成されたインデックスを記述するには、次のコマンドを使用します。

```
aws opensearch get-index \
--domain-name [domain_name] \
--index-name [index_name] \
```

既存のインデックスを更新する

既存のインデックスを更新して、新しいセマンティックエンリッチメントフィールドを追加したり、既存のフィールドでセマンティックエンリッチメントを有効または無効にしたり、非セマンティックテキストフィールドを追加したりできます。update-index コマンドを使用して、 で変更するフィールドのみを指定しますindex-schema。リクエストに含まれていないフィールドは変更されません。

> **Note**
>
> インデックスは更新settingsできません。リクエストにsettingsブロックを含めると、オペレーションは検証エラーを返します。インデックス設定を変更するには、インデックスを削除して再作成する必要があります。

を使用してインデックスを更新するには AWS CLI、 update-index コマンドを使用します。

```
aws opensearch update-index \
--domain-name [domain_name] \
--index-name [index_name] \
--index-schema [index_body]
```

新しいセマンティックエンリッチメントフィールドを追加する

セマンティックエンリッチメントを有効にした新しいtextフィールドを既存のインデックスに追加できます。サービスは、必要な ML モデル、取り込みパイプライン、検索パイプラインを自動的にセットアップします。更新後にインデックス付けされた新しいドキュメントは自動的に強化されます。

> **Important**
>
> 既存のドキュメントはバックフィルされません。セマンティックエンリッチメントフィールドを既存のドキュメントに入力するには、更新後に再度取り込む必要があります。再取り込まれるまで、既存のドキュメントは新しいフィールドでのセマンティック検索の恩恵を受けることはありません。

```
aws opensearch update-index \
--domain-name my-domain \
```

```
--index-name product-catalog \
--index-schema '{
    "mappings": {
        "properties": {
            "description": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "ENABLED",
                    "language_options": "english"
                }
            }
        }
    }
}'
```

フィールドでセマンティックエンリッチメントを無効にする

現在有効になっているフィールドでセマンティックエンリッチメントを無効にするには、 status を に設定しますDISABLED。フィールドは、取り込みパイプラインと検索パイプラインから削除されます。基盤となるテキストフィールドとその埋め込みフィールドはインデックスに残りますが、強化されなくなります。

```
aws opensearch update-index \
--domain-name my-domain \
--index-name product-catalog \
--index-schema '{
    "mappings": {
        "properties": {
            "title_semantic": {
                "type": "text",
                "semantic_enrichment": {
                    "status": "DISABLED"
                }
            }
        }
    }
}'
```

更新の制限

以下のオペレーションは ではサポートされていないupdate-indexため、インデックスを削除して再作成する必要があります。

- 現在**language_options**セマンティックエンリッチメントが有効になっているフィールドの変更。最初に フィールドを無効にしてから、新しい言語オプションで再度有効にします。
- ネストされたフィールドの更新。セマンティックエンリッチメントは最上位textフィールドでのみサポートされています。
- インデックス の更新**settings**。

> **Note**
>
> インデックスに、自動セマンティックエンリッチメントによって作成されなかったカスタム取り込みパイプラインまたは検索パイプラインがある場合、更新オペレーションはブロックされます。セマンティックエンリッチメントフィールドを追加する前に、カスタムパイプラインを削除します。

データの取り込みと検索

自動セマンティックエンリッチメントを有効にしてインデックスを作成すると、この機能はデータ取り込みプロセス中に自動的に機能するため、追加の設定は必要ありません。

データインジェスト: インデックスにドキュメントを追加すると、システムは自動的に次の操作を行います。

- セマンティックエンリッチメント用に指定したテキストフィールドを分析します
- OpenSearch Service マネージドスパースモデルを使用してセマンティックエンコーディングを生成します
- これらの強化された表現を元のデータとともに保存します。

このプロセスでは、OpenSearch の組み込み ML コネクタと取り込みパイプラインが使用され、バックグラウンドで自動的に作成および管理されます。

検索: セマンティックエンリッチメントデータは既にインデックス化されているため、クエリは ML モデルを再度呼び出すことなく効率的に実行されます。つまり、追加の検索レイテンシーのオーバーヘッドなしで、検索の関連性が向上します。

自動セマンティックエンリッチメントのアクセス許可の設定

自動セマンティックエンリッチメントを使用してインデックスを作成する前に、必要なアクセス許可を設定する必要があります。このセクションでは、さまざまなインデックスオペレーションに必要なアクセス許可と、 AWS Identity and Access Management (IAM) ときめ細かなアクセスコントロールシナリオの両方でそれらを設定する方法について説明します。

IAM アクセス許可

自動セマンティックエンリッチメントオペレーションには、次の IAM アクセス許可が必要です。これらのアクセス許可は、実行する特定のインデックスオペレーションによって異なります。

CreateIndex API アクセス許可

自動セマンティックエンリッチメントを使用してインデックスを作成するには、次の IAM アクセス許可が必要です。

- `es:CreateIndex` – セマンティックエンリッチメント機能を備えたインデックスを作成します。
- `es:ESHttpHead` – HEAD リクエストを実行してインデックスの存在を確認します。
- `es:ESHttpPut` – インデックス作成の PUT リクエストを実行します。
- `es:ESHttpPost` – インデックスオペレーションの POST リクエストを実行します。

UpdateIndex API アクセス許可

自動セマンティックエンリッチメントで既存のインデックスを更新するには、次の IAM アクセス許可が必要です。

- `es:UpdateIndex` – インデックス設定とマッピングを更新します。
- `es:ESHttpPut` – インデックス更新の PUT リクエストを実行します。
- `es:ESHttpGet` — GET リクエストを実行してインデックス情報を取得します。
- `es:ESHttpPost` – インデックスオペレーションの POST リクエストを実行します。

GetIndex API アクセス許可

自動セマンティックエンリッチメントを使用するインデックスに関する情報を取得するには、次の IAM アクセス許可が必要です。

- `es:GetIndex` – インデックス情報と設定を取得します。

- `es:ESHttpGet` — GET リクエストを実行してインデックスデータを取得します。

DeleteIndex API アクセス許可

自動セマンティックエンリッチメントを含むインデックスを削除するには、次の IAM アクセス許可が必要です。

- `es:DeleteIndex` – インデックスとそのセマンティックエンリッチメントコンポーネントを削除します。
- `es:ESHttpDelete` – インデックス削除の DELETE リクエストを実行します。

サンプルの IAM ポリシー

次の ID ベースのアクセスポリシーのサンプルは、ユーザーが自動セマンティックエンリッチメントでインデックスを管理するために必要なアクセス許可を提供します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSemanticEnrichmentIndexOperations",
            "Effect": "Allow",
            "Action": [
                "es:CreateIndex",
                "es:UpdateIndex",
                "es:GetIndex",
                "es:DeleteIndex",
                "es:ESHttpHead",
                "es:ESHttpGet",
                "es:ESHttpPut",
                "es:ESHttpPost",
                "es:ESHttpDelete"
            ],
            "Resource": "arn:aws:es:aws-region:111122223333:domain/domain-name"
        }
    ]
}
```

aws-region、*111122223333*、*domain-name* を特定の値に置き換えます。リソース ARN で特定のインデックスパターンを指定することで、アクセスをさらに制限できます。

きめ細かなアクセス制御のアクセス許可

Amazon OpenSearch Service ドメインできめ細かなアクセスコントロールが有効になっている場合は、IAM アクセス許可以外の追加のアクセス許可が必要です。インデックスオペレーションごとに次のアクセス許可が必要です。

CreateIndex API アクセス許可

きめ細かなアクセスコントロールが有効になっている場合、自動セマンティックエンリッチメントでインデックスを作成するには、次の追加のアクセス許可が必要です。

- `indices:admin/create` – インデックスオペレーションを作成します。
- `indices:admin/mapping/put` – インデックスマッピングを作成および更新します。
- `cluster:admin/opensearch/ml/create_connector` – セマンティック処理用の機械学習コネクタを作成します。
- `cluster:admin/opensearch/ml/register_model` – セマンティックエンリッチメント用の機械学習モデルを登録します。
- `cluster:admin/ingest/pipeline/put` – データ処理用の取り込みパイプラインを作成します。
- `cluster:admin/search/pipeline/put` – クエリ処理用の検索パイプラインを作成します。

UpdateIndex API アクセス許可

きめ細かなアクセスコントロールが有効になっている場合、自動セマンティックエンリッチメントでインデックスを更新するには、次の追加のアクセス許可が必要です。

- `indices:admin/get` – インデックス情報を取得します。
- `indices:admin/settings/update` – インデックス設定を更新します。
- `indices:admin/mapping/put` – インデックスマッピングを更新します。
- `cluster:admin/opensearch/ml/create_connector` – 機械学習コネクタを作成します。
- `cluster:admin/opensearch/ml/register_model` – 機械学習モデルを登録します。
- `cluster:admin/ingest/pipeline/put` – 取り込みパイプラインを作成します。
- `cluster:admin/search/pipeline/put` – 検索パイプラインを作成します。
- `cluster:admin/ingest/pipeline/get` – 取り込みパイプライン情報を取得します。
- `cluster:admin/search/pipeline/get` – 検索パイプライン情報を取得します。

GetIndex API アクセス許可

きめ細かなアクセスコントロールが有効になっている場合、自動セマンティックエンリッチメントを持つインデックスに関する情報を取得するには、次の追加のアクセス許可が必要です。

- `indices:admin/get` – インデックス情報を取得します。
- `cluster:admin/ingest/pipeline/get` – 取り込みパイプライン情報を取得します。
- `cluster:admin/search/pipeline/get` – 検索パイプライン情報を取得します。

DeleteIndex API アクセス許可

きめ細かなアクセスコントロールが有効になっている場合、自動セマンティックエンリッチメントでインデックスを削除するには、次の追加のアクセス許可が必要です。

- `indices:admin/delete` – インデックスオペレーションを削除します。

クエリの書き換え

自動セマンティックエンリッチメントは、クエリの変更を必要とせずに、既存の「一致」クエリをセマンティック検索クエリに自動的に変換します。一致クエリが複合クエリの一部である場合、システムはクエリ構造を横断し、一致クエリを検索して、それらをニューラルスパースクエリに置き換えます。現在、この機能はスタンドアロンクエリでも複合クエリの一部でも、「一致」クエリの置き換えのみをサポートしています。「multi_match」はサポートされていません。さらに、この機能は、ネストされた一致クエリを置き換えるすべての複合クエリをサポートします。複合クエリには、bool、Boosting、Constant_score、dis_max、 function_score、ハイブリッドなどがあります。

自動セマンティックエンリッチメントの制限

自動セマンティック検索は、映画のタイトル、製品説明、レビュー、概要など、自然言語コンテンツを含むsmall-to-medium規模のフィールドに適用すると最も効果的です。セマンティック検索はほとんどのユースケースの関連性を高めますが、特定のシナリオには適していない場合があります。特定のユースケースに自動セマンティックエンリッチメントを実装するかどうかを決定するときは、次の制限を考慮してください。

- 非常に長いドキュメント – 現在のスパースモデルは、各ドキュメントの最初の 8,192 トークンのみを英語で処理します。多言語ドキュメントの場合、トークンは 512 個です。長い記事については、完全なコンテンツ処理を確保するためにドキュメントチャンキングの実装を検討してください。

- ログ分析ワークロード – セマンティックエンリッチメントはインデックスサイズを大幅に増やします。これは、完全一致が通常十分であるログ分析には不要である可能性があります。セマンティックコンテキストを追加すると、増加したストレージ要件を正当化するのに十分なログ検索の有効性が向上することはめったにありません。
- 自動セマンティックエンリッチメントは、派生ソース機能と互換性がありません。
- スロットリング — OpenSearch Service ドメインのインデックス推論リクエストは現在 200 TPS に制限されています。これはソフト制限です。より高い制限については AWS サポートにお問い合わせください。

料金

Amazon OpenSearch Service は、インデックス作成時にスパースベクトル生成中に消費された OpenSearch Compute Units (OCUs) に基づいて、自動セマンティックエンリッチメントを請求します。自動セマンティックエンリッチメントを有効にしたテキストフィールドのインデックス作成中の実際の使用に対してのみ課金されます。1 つのセマンティック検索 OCU は、英語コンテンツに対して 1,110 万トークンを処理できます。24 億個のトークンを処理するには、約 216 個のセマンティック検索 OCU 時間 (24 億/1,110 万) が必要です。セマンティック検索 OCU 時間あたり 0.24 USD の料金では、自動セマンティック検索のために 10 GB のデータを処理するコストは 51 USD (216 OCU 時間 x 0.24 USD/OCU 時間) になります。検索オペレーション中またはデータストレージには、セマンティック検索 OCU の追加料金はかかりません。

Amazon CloudWatch メトリクス を使用して、この消費量をモニタリングできますSemanticSearchOCU。モデルトークンの制限、OCU あたりのボリュームスループット、およびサンプル計算の例の詳細については、OpenSearch Service の料金」を参照してください。

サポートされる AWS リージョン

自動セマンティックエンリッチメントは、以下で使用できます AWS リージョン。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)

- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ストックホルム)
- 欧州 (スペイン)

Amazon OpenSearch Service で設定変更を行う

Amazon OpenSearch Service では、ドメインの更新時に Blue/Green デプロイプロセスが使用されます。ブルー/グリーンデプロイでは、本番環境をコピーするドメイン更新用のアイドル環境が作成され、更新が完了した後にユーザーが新しい環境にルーティングされます。ブルー/グリーンデプロイでは、ブルー環境が現在の本稼働環境です。グリーン環境はアイドル環境です。

データはブルー環境からグリーン環境に移行されます。新しい環境の準備ができたら、OpenSearch Service が環境を切り替えて、グリーン環境を新しい本稼働環境にプロモートします。データ損失なしでスイッチオーバーが行われます。この方法では、新しい環境へのデプロイに失敗しても、ダウンタイムを最小限に抑えることができ、元の環境を維持することができます。

トピック

- 通常は blue/green デプロイの原因となる変更
- 通常は blue/green デプロイが発生しない変更
- ブルー/グリーンデプロイオプション
- 変更によってブルー/グリーンデプロイが実行されるかどうかを判断する
- 設定変更の追跡
- 設定変更のステージ
- ブルー/グリーンデプロイのパフォーマンスへの影響
- 設定変更に関連する料金
- 検証エラーのトラブルシューティング

通常は blue/green デプロイの原因となる変更

次のオペレーションにより、Blue/Green デプロイが発生します。

- インスタンスタイプを変更する
- きめ細かなアクセスコントロールの有効化

- サービスソフトウェアのアップデートを行う
- 専用マスターノードを有効または無効にする
- スタンバイなしのマルチ AZ を有効または無効にする
- ストレージタイプ、ボリュームタイプ、ボリュームサイズの変更
- 別の VPC のサブネットを選択する
- VPC セキュリティグループを追加または削除する
- 専用コーディネーターノードの追加または削除
- OpenSearch Dashboards の Amazon Cognito 認証を有効または無効にする
- 別の Amazon Cognito ユーザープールまたは ID プールを選択する
- アドバンスト設定を変更する
- 新しい OpenSearch バージョンへのアップグレード (OpenSearch Dashboards は、アップグレードの一部またはすべてで使用できない場合があります)
- 保管中のデータの暗号化およびノード間の暗号化を有効にする
- UltraWarm またはコールドストレージを有効または無効にする
- Auto-Tune を無効にし、変更をロールバックする
- ドメインへのオプションプラグインの関連付けとドメインからのオプションプラグインの関連付けの解除
- 専用マスターノードが 2 つある Multi-AZ ドメインの専用マスターノード数の増加
- EBS ボリュームサイズの縮小
- 最後に行った変更が進行中であるか、6 時間以内に発生した場合、EBS ボリュームサイズ、IOPS、またはスループットを変更する
- CloudWatch への監査ログの発行を有効にします。

スタンバイが有効のマルチ AZ では、1 回につき 1 つの変更リクエストしか行えません。変更が既に実行中の場合、新しいリクエストは拒否されます。現在の変更ステータスは、`DescribeDomainChangeProgress` API を使用して確認できます。

通常は blue/green デプロイが発生しない変更

ほとんどの場合、次のオペレーションにより、Blue/Green デプロイは発生しません。

- アクセスポリシーの変更
- カスタムエンドポイントの変更

- Transport Layer Security (TLS) ポリシーを変更する
- 自動スナップショットの時間を変更する
- [HTTPS が必要] を有効または無効にする
- 変更内容をロールバックせずに Auto-Tune を有効または無効にする
- ドメインに専用マスターノードがある場合に、データノードまたは UltraWarm ノード数を変更する
- ドメインに専用マスターノードがある場合、専用マスターインスタンスのタイプまたは数を変更する (専用ノードが 2 つある Multi-AZ ドメインを除く)
- エラーログ、またはスローログの CloudWatch への公開の有効化または無効化
- CloudWatch への監査ログの発行の無効化
- gp3 EBS タイプのボリュームサイズ、IOPS、またはスループットを増やす。

> **Note**
>
> 2026 年 3 月 10 日以前は、インプレースボリュームの増加は gp3 で最大 3 TiB のボリュームでのみサポートされていました。2026 年 3 月 10 日、この制限は削除され、インプレースボリュームは 3 TiB を超えて増加しました。ただし、この制限が削除される前にクラスターのボリュームサイズが 3 TiB を超えた場合、最初のボリュームを増やすには Blue/Green デプロイが必要です。そのクラスターの後続のすべてのボリュームの増加はインプレース更新として実行され、ブルー/グリーンデプロイは必要ありません。

- タグの追加と削除

> **Note**
>
> サービスソフトウェアのバージョンによっては、一部の例外があります。変更によるブルー/グリーンデプロイが起らないことを確実にしたい場合で、このオプションが利用可能な場合は、ドライランを実行してからドメインを更新してください。一部の変更では、ドライランオプションが提供されません。通常、ピークトラフィック時間外にクラスターを変更することをお勧めします。

ブルー/グリーンデプロイオプション

更新時に十分な容量が利用できない場合にクラスターがデプロイを処理する方法を制御するデプロイ戦略を選択します。

1. Full Swap Blue/Green — デフォルトのデプロイ動作。インスタンスのフルキャパシティーを事前に要求し、キャパシティーが利用可能な場合に最速のデプロイを確保します。十分な容量を割り当てることができない場合、デプロイは続行されません。
2. 容量最適化 — 30 個以上のデータノードを持つクラスターに推奨されます。最初に完全なブルー/グリーンスワップを試み、容量が不十分な場合は、バッチでのデプロイに進みます。容量が制限されていてもデプロイを完了できるようにします。デプロイはバッチで行われるため、完了時間が長くなる場合があります。

Console

ドメインフローの編集の場合:

1. で Amazon OpenSearch Service コンソールを開きますhttps://console.aws.amazon.com/aos/home。
2. ナビゲーションペインのドメインで、ドメイン名を選択してクラスター設定を開きます。
3. クラスター設定タブの右側にある編集ボタンをクリックします。
4. デプロイ戦略オプションから、ドメイン更新に必要な設定を選択します。

    1. Full Swap Blue/Green — デフォルトのデプロイ動作。インスタンスのフルキャパシティーを事前に要求し、キャパシティーが利用可能な場合に最速のデプロイを確保します。十分な容量を割り当てることができない場合、デプロイは続行されません。
    2. 容量最適化 — 30 個以上のデータノードを持つクラスターに推奨されます。最初に完全なブルー/グリーンスワップを試み、容量が不十分な場合は、バッチでのデプロイに進みます。容量が制限されていてもデプロイを完了できるようにします。デプロイはバッチで行われるため、完了時間が長くなる場合があります。
5. [Save changes] (変更の保存) をクリックします。

ドメインフローの作成の場合:

1. で Amazon OpenSearch Service コンソールを開きますhttps://console.aws.amazon.com/aos/home。

2. 左側のナビゲーションペインで [ドメイン] を選択します。
3. ドメインの作成ボタンをクリックします。
4. ドメインに必要な設定をすべて選択します。
5. デプロイ戦略オプションから、ドメイン更新に必要な設定を選択します。
    1. Full Swap Blue/Green — デフォルトのデプロイ動作。インスタンスのフルキャパシティーを事前に要求し、キャパシティーが利用可能な場合に最速のデプロイを確保します。十分な容量を割り当てることができない場合、デプロイは続行されません。
    2. 容量最適化 — 30 個以上のデータノードを持つクラスターに推奨されます。最初に完全なブルー/グリーンスワップを試み、容量が不十分な場合は、バッチでのデプロイに進みます。容量が制限されていてもデプロイを完了できるようにします。デプロイはバッチで行われるため、完了時間が長くなる場合があります。
6. 右側のドメイン概要パネルの「作成」ボタンをクリックします。

API

UpdateDomainConfig API を使用してデプロイ戦略を設定できます。

容量最適化オプション

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/
config
{
   "DeploymentStrategyOptions": {
    "DeploymentStrategy": "CapacityOptimized"
   }
}
```

デフォルトオプション

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/
config
{
   "DeploymentStrategyOptions": {
    "DeploymentStrategy": "Default"
   }
}
```

変更によってブルー/グリーンデプロイが実行されるかどうかを判断する

一部のタイプの計画された設定の変更をテストして、これらの変更をコミットせずにブルー/グリーンデプロイが実行されるかどうかを判断できます。設定の変更を開始する前に、コンソールまたは API を使用して検証チェックを実行し、ドメインが更新の対象であることを確認してください。

Console

設定の変更を検証するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで [Domains] (ドメイン) を選択します。
3. 設定の変更を実行するドメインを選択します。選択すると、ドメインの詳細ページが開きます。[Actions] (アクション) ドロップダウンメニューを選択してから、[Edit cluster configuration] (クラスター設定の編集) を選択します。
4. インスタンスタイプやノード数の変更など、ドメインを変更します。
5. [ドライラン分析] で、[実行] を選択します。ドライランにより、設定変更のエラーを検証し、ブルー/グリーンデプロイが必要かどうかを判断します。
6. ドライランが完了したら、その結果がドライラン ID とともに、ページの最下部に表示されます。分析結果は、設定変更にブルー/グリーンデプロイが必要かどうかを示します。

   新たにドライランを実行すると、前のドライラン結果は上書きされます。各ドライラン実行の詳細を保持したい場合は、そのドライラン ID を保存しておきます。ドライランは 90 日間、または設定を更新するまで繰り返し利用できます。
7. 設定の更新を続行するには、[Save changes] (変更の保存) を選択します。それ以外の場合は、[Cancel] (キャンセル) を選択します。どちらのオプションを選択しても、[Cluster configuration] (クラスター設定) タブに戻ります。このタブで [Dry run details] (ドライランの詳細) を選択すると、最新のドライランの詳細が確認できます。このページには、ドライラン前とドライラン後の設定の対照比較も含まれています。

API

設定 API を使用してドライラン検証を実行できます。API を使用して変更をテストするには、`DryRun` を `true`、`DryRunMode` を `Verbose` に設定します。Verbose モードは、変更によってブルー/グリーンデプロイが開始されるかどうかの判断に加えて、検証チェックも実行しま

す。例えば、この UpdateDomainConfig リクエストは、UltraWarm を有効化した結果としてのデプロイタイプをテストします。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/
config
{
   "ClusterConfig": {
    "WarmCount": 3,
    "WarmEnabled": true,
    "WarmType": "ultrawarm1.large.search"
   },
   "DryRun": true,
   "DryRunMode": "Verbose"
}
```

リクエストは検証チェックを実行し、変更に起因するデプロイのタイプを返しますが、実際の更新は行いません。

```
{
   "ClusterConfig": {
     ...
    },
   "DryRunResults": {
      "DeploymentType": "Blue/Green",
      "Message": "This change will require a blue/green deployment."
    }
}
```

考えられるデプロイタイプは、次のとおりです。

- Blue/Green – 変更によって、ブルー/グリーンデプロイが発生します。
- DynamicUpdate – 変更によって、ブルー/グリーンデプロイは発生しません。
- Undetermined – ドメインがまだ処理状態であるため、デプロイのタイプを特定できません。
- None – 設定変更はありません。

検証が失敗すると、検証失敗のリストが返されます。

```
{
   "ClusterConfig":{
```

```
      "..."
   },
   "DryRunProgressStatus":{
      "CreationDate":"2023-01-12T01:14:33.847Z",
      "DryRunId":"db00ca39-48b2-4774-bbd3-252cf094d205",
      "DryRunStatus":"failed",
      "UpdateDate":"2023-01-12T01:14:33.847Z",
      "ValidationFailures":[
         {
            "Code":"Cluster.Index.WriteBlock",
            "Message":"Cluster has index write blocks."
         }
      ]
   }
}
```

ステータスが引き続き pending になっている場合は、後続の DescribeDryRunProgress コールの UpdateDomainConfig レスポンス内にあるドライラン ID を使用して、検証のステータスを確認できます。

```
GET https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/
dryRun?dryRunId=my-dry-run-id
{
    "DryRunConfig": null,
    "DryRunProgressStatus": {
        "CreationDate": "2023-01-12T01:14:42.998Z",
        "DryRunId": "db00ca39-48b2-4774-bbd3-252cf094d205",
        "DryRunStatus": "succeeded",
        "UpdateDate": "2023-01-12T01:14:49.334Z",
        "ValidationFailures": null
    },
    "DryRunResults": {
        "DeploymentType": "Blue/Green",
        "Message": "This change will require a blue/green deployment."
    }
}
```

検証チェックなしでドライラン分析を実行するには、設定 API を使用するときに DryRunMode を Basic に設定します。

Python

次の Python コードは、UpdateDomainConfig API を使用してドライラン検証チェックを実行し、チェックが成功した場合は、ドライランなしで同じ API を呼び出して更新を開始します。チェックが失敗した場合、スクリプトはエラーを出力して停止します。

```
import time
import boto3

client = boto3.client('opensearch')

response = client.UpdateDomainConfig(
    ClusterConfig={
        'WarmCount': 3,
        'WarmEnabled': True,
        'WarmCount': 123,
    },
    DomainName='test-domain',
    DryRun=True,
    DryRunMode='Verbose'
)

dry_run_id = response.DryRunProgressStatus.DryRunId

retry_count = 0

while True:

    if retry_count == 5:
        print('An error occured')
        break

    dry_run_progress_response = client.DescribeDryRunProgress('test-domain',
 dry_run_id)
    dry_run_status = dry_run_progress_response.DryRunProgressStatus.DryRunStatus

    if dry_run_status == 'succeeded':
        client.UpdateDomainConfig(
            ClusterConfig={
            'WarmCount': 3,
            'WarmEnabled': True,
            'WarmCount': 123,
        })
```

```
        break

    elif dry_run_status == 'failed':
        validation_failures_list =
dry_run_progress_response.DryRunProgressStatus.ValidationFailures
        for item in validation_failures_list:
            print(f"Code: {item['Code']}, Message: {item['Message']}")
        break

    retry_count += 1
    time.sleep(30)
```

設定変更の追跡

一度に 1 つの設定変更をリクエストしたり、1 つのリクエストで複数の設定変更をグループ化することもできます。コンソールの [ドメイン処理ステータス] と [設定変更ステータス] フィールドを使用して、設定変更を追跡できます。追加の変更をリクエストする前に、ドメインのステータスが Active になるまで待ちます。

ドメインの処理ステータスは次のとおりです。

- Active – 進行中の設定変更はありません。新しい設定変更リクエストを送信できます。
- Creating – ドメインを作成中です。
- Modifying – 新しいデータノードの追加、EBS、gp3、IOPS プロビジョニング、KMS キー設定などの設定変更が進行中です。
- Upgrading engine version – エンジンバージョンのアップグレードが進行中です。
- Updating service software – サービスソフトウェアの更新が進行中です。
- Deleting – ドメイン削除中です。
- Isolated – ドメインは一時停止状態です。

ドメインには、次の設定変更ステータスがあります。

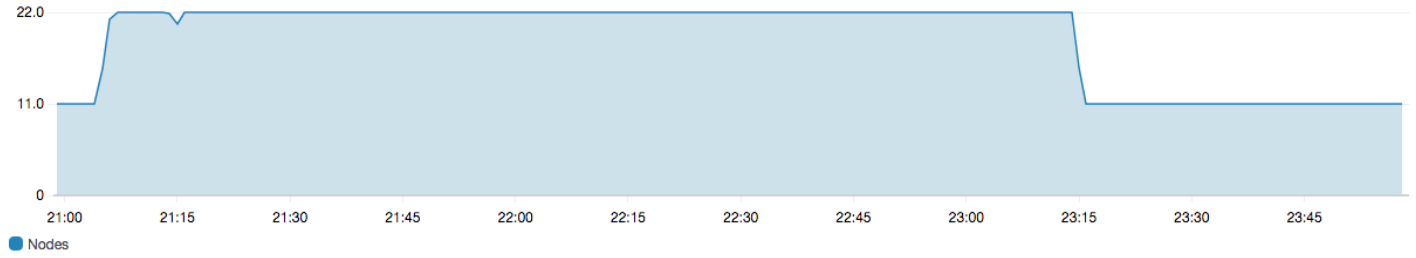
- Pending – 設定変更リクエストが送信されました。
- Initializing – サービスは設定変更リクエストを初期化しています。
- Validating – サービスは、リクエストされた変更と必要なリソースを検証しています。
- Awaiting user inputs – サービスは、インスタンスタイプの変更などの、設定変更の進行待ち状態です。設定変更を編集できます。

- `Applying changes` – サービスはリクエストされた設定変更を適用しています。
- `Cancelled` – 設定変更はキャンセルされました。[キャンセル] を選択するとすべての変更をロールバックできます。
- `Completed` – リクエストされた設定変更は正常に完了しました。
- `Validation failed` – リクエストされた設定変更を完了できませんでした。設定の変更は適用されませんでした。

> **Note**
>
> 検証の失敗は、ドメイン内に存在する赤いインデックス、選択したインスタンスタイプの可用性の欠如、またはディスク容量不足の結果である可能性があります。検証エラーのリストについては、「the section called “検証エラーのトラブルシューティング”」を参照してください。検証失敗イベント中に、設定の変更をキャンセル、再試行、編集できます。

設定の変更が完了すると、ドメインのステータスは `Active` に戻ります。

クラスターヘルスと Amazon CloudWatch メトリクスを調べると、ドメイン更新中はクラスター内のノード数が一時的に増加 (しばしば倍増) していることを確認できます。次の図では、設定の変更中にノード数が 11 から 22 に倍増し、更新が完了すると 11 に戻っていることがわかります。



この一時的な増加により、管理対象のノード数がかなり増えるため、クラスターの専用マスターノードへの負荷が高くなります。また、OpenSearch Service が古いクラスターから新しいクラスターにデータをコピーするため、検索とインデックス作成のレイテンシーが増加する可能性もあります。クラスターでは、Blue/Green デプロイに関連するオーバーヘッドを処理するための十分な容量を維持することが重要です。

> **Important**
>
> 設定の変更およびサービスのメンテナンス中に、追加料金は発生しません。課金対象となるのは、クラスター用にリクエストしたノード数のみです。詳細については、「the section called “設定変更に関連する料金”」を参照してください。

専用マスターノードに過度な負荷がかかることを防ぐには、Amazon CloudWatch メトリクスを使用して使用率をモニタリングできます。推奨される最大値については、「the section called “推奨される CloudWatch アラーム”」を参照してください。

設定変更のステージ

設定の変更を開始すると、OpenSearch Service は一連の手順を経てドメインを更新します。コンソールの [設定変更ステータス] で設定変更の進行状況を表示できます。更新が実行される正確な手順は、変更の種類によって異なります。また、DescribeDomainChangeProgress API 操作を使用して設定の変更をモニタリングすることもできます。

設定変更中に更新が実行される可能性のあるステージを次に示します。

ステージ名	説明
検証	ドメインが更新の要件を満たしていることを検証し、必要に応じて検証の問題を明らかにします。
新しい環境の作成	blue/green デプロイを開始するために必要な前提条件を完了し、必要なリソー

ステージ名	説明
	スを作成します。
新しいノードのプロビジョニング	新しい環境での新しいインスタンスのセットの作成。
新しいノードでのトラフィックルーティング	新しく作成されたデータノードにトラフィックをリダイレクトします。
古いノードでのトラフィックルーティング	古いデータノードでトラフィックを無効にする。
削除するノードの準備	ノードを削除する準備をしています。このステップは、ドメインを縮小している場合にのみ発生します (例えば、8 ノードから 6 ノードへ)。

ステージ名	説明
シャードの新しいノードへのコピー	古いノードから新しいノードにシャードを移動します。
ノードの終了	シャードを削除した後、古いノードを終了して削除する。
古いリソースの削除	古い環境 (ロードバランサーなど) に関連付けられているリソースの削除。
動的更新	更新が blue/green デプロイを必要とせず、動的に適用できる場合に表示されます。
専用マスター関連の変更の適用	専用マスターインスタンスのタイプまたは数が変更されたときに表示されます。

ステージ名	説明
ボリューム関連の変更の適用	ボリュームのサイズ、タイプ、IOPS、スループットが変更されたときに表示されます。

ブルー/グリーンデプロイのパフォーマンスへの影響

ブルー/グリーンデプロイ中は、Amazon OpenSearch Service クラスターが受信検索リクエストとインデックス作成リクエストに使用できます。ただし、次のようなパフォーマンスの問題が発生する場合があります。

- クラスターの管理するノードが増えるにつれて、リーダーノードの使用量が一時的に増加する。
- OpenSearch Service が古いノードから新しいノードにデータをコピーするにつれて、検索とインデックス作成のレイテンシーが増加した。
- ブルー/グリーンデプロイ中にクラスターの負荷が増加するにつれて、受信リクエストの拒否が増加した。
- レイテンシーの問題やリクエストの拒否を回避するには、クラスターが正常でネットワークトラフィックが少ない場合にブルー/グリーンデプロイを実行する必要があります。

設定変更に関連する料金

ドメインの設定を変更すると、「the section called “設定変更”」で説明されているように、OpenSearch Service で新しいクラスターが作成されます。古いものから新しいものへの移行中、以下の料金が発生します。

- インスタンスタイプを変更した場合、両方のクラスター (最初の 1 時間) が課金されます。最初の 1 時間の後は、新しいクラスターに対してのみ料金が発生します。EBS ボリュームはクラスターの一部であるため、2 回課金されることはありません。そのため、課金はインスタンスの課金に従います。

例: 3 つの m3.xlarge インスタンスから 4 つの m4.large インスタンスに設定を変更した場合。最初の 1 時間は、両方のクラスターに課金されます (3 * m3.xlarge + 4 * m4.large)。最初の 1 時間の後は、新しいクラスターに対してのみ料金が発生します (4 * m4.large)。

- インスタンスタイプを変更しない場合は、最大のクラスターに対してのみ課金されます (最初の 1 時間)。最初の 1 時間の後は、新しいクラスターに対してのみ料金が発生します。

  例: 6 つの m3.xlarge インスタンスから 3 つの m3.xlarge インスタンスに設定を変更した場合。最初の 1 時間は、最大のクラスターに課金されます (6 * m3.xlarge)。最初の 1 時間の後は、新しいクラスターに対してのみ料金が発生します (3 * m3.xlarge)。

検証エラーのトラブルシューティング

設定の変更を開始するか、OpenSearch または Elasticsearch のバージョンアップグレードを実行すると、OpenSearch Service は最初に一連の検証チェックを実行して、ドメインが更新のための要件を満たしていることを確認します。これらのチェックのいずれかが失敗した場合、ドメインを更新する前に修正する必要がある特定の問題を含む通知がコンソールで表示されます。

OpenSearch Service は、ドメインが 60 日以上使用不能状態のままである場合にドメインを分離します。これらの問題を解決するための通知リマインダーが送信されます。この期間内に問題を解決しない場合、OpenSearch Service はドメインとそのデータを削除します。

OpenSearch Service が明らかにする可能性のあるドメインの問題と、それらを解決するためのステップを次の表に示します。

問題	エラーコード	トラブルシューティングのステップ
セキュリティグループが見つかりません	SecurityGroupNotFound	OpenSearch Service ドメインに関連付けられているセキュリティグループが存在しません。この問題を解決するには、指定された名前でセキュリティ グループを作成します。
サブネットが見つかりません	SubnetNotFound	OpenSearch Service ドメインに関連付けられているサブネットが存在しません。この問題を解決するには、VPC でサブネットを作成します。

問題	エラーコード	トラブルシューティングのステップ
サービスにリンクされたロールが設定されていません	SLRNotConfigured	OpenSearch Service のためにサービスにリンクされたロールが設定されていません。サービスにリンクされたロールは OpenSearch Service によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。ロールが存在しない場合は、手動で作成する必要がある場合があります。
十分な IP アドレスがありません	InsufficientFreeIPsForSubnets	1 つ以上の VPC サブネットに、ドメインを更新するのに十分な IP アドレスがありません。必要な IP アドレスの数を計算するには、「the section called “VPC サブネットで IP アドレスをリザーブする”」を参照してください。
Cognito ユーザープールが存在しません	CognitoUserPoolNotFound	OpenSearch Service は Amazon Cognito ユーザープールを見つけることができません。ユーザープールを作成し、正しい ID が設定されていることを確認します。ID を見つけるには、Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用できます。 `aws cognito-idp list-user-pools --max-results 60 --region us-east-1`
Cognito ID プールが存在しません	CognitoIdentityPoolNotFound	OpenSearch Service は Cognito ID プールを見つけることができません。ユーザープールを作成し、正しい ID が設定されていることを確認します。ID を見つけるには、Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用できます。 `aws cognito-identity list-identity-pools --max-results 60 --region us-east-1`
ユーザープールの Cognito ドメインが見つかりません	CognitoDomainNotFound	ユーザープールにドメイン名がありません。Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用して設定できます。 `aws cognito-idp create-user-pool-domain --domain my-domain --user-pool-id id`

問題	エラーコード	トラブルシューティングのステップ
Cognito ロールが設定されていません	`CognitoRoleNotConfigured`	Amazon Cognito ユーザープールと ID プールを設定し、それらを認証に使用するための許可を OpenSearch Service に付与する IAM ロールが設定されていません。適切な許可セットと信頼関係を使用してロールを設定します。コンソールを使用してデフォルトの CognitoAccessForAmazonOpenSearch ロールを作成することも、AWS CLI または AWS SDK を使用してロールを手動で設定することもできます。
ユーザープールを記述できません	`UserPoolNotDescribable`	指定された Amazon Cognito ロールには、ドメインに関連付けられたユーザープールを記述するための許可がありません。ロールの許可ポリシーで `cognito-identity:DescribeUserPool` アクションが許可されていることを確認してください。完全な許可ポリシーについては、「the section called “CognitoAccessForAmazonOpenSearch ロールについて”」を参照してください。
ID プールを記述できません	`IdentityPoolNotDescribable`	指定された Amazon Cognito ロールには、ドメインに関連付けられた ID プールを記述するための許可がありません。ロールの許可ポリシーで `cognito-identity:DescribeIdentityPool` アクションが許可されていることを確認してください。完全な許可ポリシーについては、「the section called “CognitoAccessForAmazonOpenSearch ロールについて”」を参照してください。
ユーザーおよび ID プールを記述できません	`CognitoPoolsNotDescribable`	指定された Amazon Cognito ロールには、ドメインに関連付けられたユーザーおよび ID プールを記述するための許可がありません。ロールの許可ポリシーで `cognito-identity:DescribeIdentityPool` および `cognito-identity:DescribeUserPool` アクションが許可されていることを確認してください。完全な許可ポリシーについては、「the section called “CognitoAccessForAmazonOpenSearch ロールについて”」を参照してください。

問題	エラーコード	トラブルシューティングのステップ
KMS キーが有効になっていません	KMSKeyNotEnabled	ドメインの暗号化に使用される AWS Key Management Service (AWS KMS) キーは無効になっています。すぐにキーを再度有効にします。
カスタム証明書が [ISSUED] (発行済み) の状態ではありません	InvalidCertificate	ドメインがカスタムエンドポイントを使用している場合は、 AWS Certificate Manager (ACM) で SSL 証明書を生成するか、独自の証明書をインポートして保護します。証明書のステータスは [Issued] (発行済み) である必要があります。このエラーが発生した場合は、ACM コンソールで証明書のステータスを確認してください。ステータスが [Expired] (期限切れ)、[Failed] (失敗)、[Inactive] (非アクティブ)、または [Pending validation] (検証の保留中) の場合は、ACM のトラブルシューティングドキュメントを参照して問題を解決してください。
選択したインスタンスタイプを起動するのに十分なキャパシティがありません	InsufficientInstanceCapacity	リクエストされたインスタンスタイプのキャパシティは利用できません。例えば、5 個の i3.16xlarge.search ノードをリクエストしたが、OpenSearch Service で使用できる十分な i3.16xlarge.search ホストがないため、リクエストを満たすことができない場合があります。OpenSearch Service でサポートされているインスタンスタイプを確認し、別のインスタンスタイプを選択します。
クラスター内の赤いインデックス	RedCluster	クラスター内の 1 つ以上のインデックスが赤のステータスになり、全体的に赤のクラスターステータスになります。この問題のトラブルシューティングおよび修復については、「the section called “赤のクラスター状態”」を参照してください。
メモリサーキットブレーカー、リクエストが多すぎます	TooManyRequests	ドメインに対する検索および書き込みリクエストが多すぎるため、OpenSearch Service は設定を更新できません。リクエストの数を減らしたり、インスタンスを最大 64 GiB の RAM まで垂直方向にスケールしたり、インスタンスを追加して水平方向にスケールしたりできます。

問題	エラーコード	トラブルシューティングのステップ
新しい設定はデータを保持できません (ディスク容量不足)	InsufficientStorageCapacity	設定されたストレージサイズでは、ドメインのすべてのデータを保持できません。この問題を解決するには、より大きなボリュームを選択するか、未使用のインデックスを削除するか、クラスター内のノードの数を増やしてディスク容量をすぐに解放します。

特定のノードにシャードが固定されています	ShardMovementBlocked	ドメイン内の 1 つ以上のインデックスが特定のノードにアタッチされており、再割り当てできません。これは、シャード割り当てのフィルタリングを設定したために発生する可能性が最も高いです。このフィルタリングは、特定のインデックスのシャードをホストすることを許可するノードを指定できるようにします。
		
		この問題を解決するには、影響を受けるすべてのインデックスからシャード割り当てフィルターを削除します。
		
		`PUT my-index/_settings`
		`{`
		`  "settings": {`
		`  "index.routing.allocation.require._name": null`
		`  }`
		`}`

新しい設定はすべてのシャードを保持できません (シャード数)	TooManyShards	ドメインのシャード数が多すぎるため、OpenSearch Service がそれらを新しい設定に移行できません。この問題を解決するには、現在のクラスターノードと同じ設定タイプのノードを追加して、ドメインを水平にスケールします。EBS ボリュームの最大サイズは、ノードのインスタンスタイプによって異なることに注意してください。 今後この問題が発生しないようにするには、「the section called “シャード数の選択”」を参照して、ユースケースに適したシャーディング戦略を定義してください。

問題	エラーコード	トラブルシューティングのステップ
ドメインに関連付けられたサブネットは IPv4 アドレスをサポートしていません	`ResultCodeIPv4BlockNotExists`	この問題を解決するには、ドメインの設定された IP アドレスタイプに応じて、VPC でサブネットを作成するか、既存のサブネットを更新します。ドメインで [IPv4 のみ] のアドレスタイプを使用している場合は、IPv4 のみのサブネットを使用します。ドメインで [デュアルスタックモード] を使用している場合は、デュアルスタックサブネットを使用します。
ドメインに関連付けられたサブネットは IPv6 アドレスをサポートしていません	`ResultCodeIPv6BlockNotExists`	この問題を解決するには、ドメインの設定された IP アドレスタイプに応じて、VPC でサブネットを作成するか、既存のサブネットを更新します。ドメインで [IPv4 のみ] のアドレスタイプを使用している場合は、IPv4 のみのサブネットを使用します。ドメインで [デュアルスタックモード] を使用している場合は、デュアルスタックサブネットを使用します。

Amazon OpenSearch Service でのサービスソフトウェア更新

> **Note**
>
> 各メジャーな (パッチでない) サービスのソフトウェア更新で行われた変更および追加に関する説明については、リリースノートを参照してください。

Amazon OpenSearch Service では、機能追加またはドメイン強化を行うサービスソフトウェア更新を定期的にリリースしています。コンソールの [Notifications] (通知) パネルを使用すると、更新がリリースされているかどうかの確認や、更新のステータスのチェックを簡単に行うことができます。各通知には、サービスソフトウェア更新に関する詳細が含まれます。ソフトウェア更新では、ダウンタイムを最小限に抑えるためブルー/グリーンデプロイが使用されます。

サービスソフトウェアの更新は、OpenSearch Service のバージョンアップグレードとは異なります。新しいバージョンの OpenSearch Service へのアップグレードについては、「」を参照してくださいthe section called “ドメインのアップグレード”。

OpenSearch Service では、必要なサービスソフトウェアの更新を可用性から 30 日以内に適用する必要があります。これらの更新は、セキュリティコンプライアンスを維持する上で重要です。

30 日以内に必要な更新を適用しない場合、30 日間 15 日ごとにリマインダー通知が送信されます。コンプライアンスのないこの期間を過ぎると、ドメインは次の効果で分離されます。

- ドメインへのすべてのネットワークアクセスが削除されます
- ドメインのステータスが分離された状態に変更される
- ドメインは、必要な更新を適用するまで使用できません

隔離中は、60 日間 15 日ごとにリマインダー通知が引き続き届きます。この期間内に必要な更新を適用しない場合、OpenSearch Service ドメインと関連するすべてのデータは完全に削除されます。詳細については、「検証エラーのトラブルシューティング」を参照してください。

オプションの更新と必須の更新

OpenSearch Service のサービスソフトウェア更新には、大きく分けて次の 2 つのカテゴリがあります。

オプションの更新

オプションのサービスソフトウェア更新には、通常、機能強化や新機能のサポートが含まれます。オプションの更新はドメインには適用されず、インストールの厳しい締切もありません。更新の有無は、メールとコンソールの通知でユーザーに通知されます。更新は、すぐに適用することもできれば、別の日時にスケジュール変更することもできます。ドメインのオフピークの時間帯にスケジュールすることも可能です。ソフトウェア更新の大半は、任意です。

更新をスケジュールするかどうかにかかわらず、`UseLatestServiceSoftwareForBlueGreen`設定が有効で、ブルー/グリーンデプロイを引き起こすドメインに変更を加えると、OpenSearch Service は自動的にサービスソフトウェアを更新します。この設定が無効になっている場合、ブルー/グリーンデプロイ中にサービスソフトウェアは自動的に更新されません。

自分のドメインで、オフピークの時間帯にオプションの更新を自動適用するように、設定することが可能です。このオプションをオンにすると、OpenSearch Service はオプションの更新が利用可能に

なってから少なくとも 13 日間待機し、7 日後に更新をスケジュールします。更新がスケジュールされると、コンソールに通知が届きます。別の日にスケジュール変更することも可能です。

自動ソフトウェア更新を有効にするには、ドメインを作成または更新する際に、[Enable automatic software update] (自動ソフトウェア更新を有効にする) を選択します。を使用して同じ設定を構成するには AWS CLI、ドメインを作成または更新`true`するときに `--software-update-options`をに設定します。

必須の更新

必須のサービスソフトウェア更新には、通常、ドメインの整合性と機能とを維持するための、重要なセキュリティ修正やその他必要な更新が含まれます。必須の更新の例には、Log4j の Common Vulnerabilities and Exposures (CVE) や、Instance Metadata Service Version 2 (IMDSv2) の適用などがあります。必須の更新が行われる回数は、通常、年に 3 回未満です。

OpenSearch Service はこれらの更新を自動的にスケジュールし、スケジュールされた更新の 7 日前に E メールとコンソール通知で通知します。必須の更新は、すぐに適用することもできれば、指定された期間内の別の日時にスケジュール変更することもできます。ドメインの、次回のオフピークの時間帯にスケジュールすることも可能です。ユーザーが必須の更新に関して何もしなかったり、ブルー/グリーンデプロイにつながるドメインの変更を行わなかったりすると、OpenSearch Service は、指定された期限 (通常は更新の提供から 14 日目) 以降随時、ドメインのオフピークの時間帯に更新を開始します。

更新がいつ予定されているかにかかわらず、`UseLatestServiceSoftwareForBlueGreen`設定が有効で、ブルー/グリーンデプロイを引き起こすドメインに変更を加えると、OpenSearch Service は自動的にドメインを更新します。この設定が無効になっている場合、ブルー/グリーンデプロイ中にサービスソフトウェアは自動的に更新されません。

パッチ更新

末尾が「-P」と数字のサービスソフトウェアバージョン (例: R20211203-*P4*) がパッチリリースです。パッチには、パフォーマンスの改善、マイナーなバグ修正、セキュリティ修正または体制の改善が含まれる可能性があります。パッチリリースには新機能や重大な変更は含まれず、通常は、ユーザーに直接的なまたは顕著な影響はありません。サービスソフトウェアの通知を見ると、パッチリリースがオプションか必須かがわかります。

考慮事項

ドメインを更新するかどうかを決定するときには、以下を考慮します。

- ドメインを手動で更新すると、新しい機能をより迅速に活用できます。[更新] を選択すると、OpenSearch Service はリクエストをキューに入れ、時間があるときに更新を開始します。
- サービスソフトウェアのアップデートを開始すると、OpenSearch Service はアップデートの開始時と完了時に通知を送信します。
- ソフトウェア更新では、ダウンタイムを最小化するためにブルー/グリーンデプロイが使用されます。更新によってクラスターの専用マスターノードに一時的に負荷がかかることがあるため、関連するオーバーヘッドを処理するのに十分な容量を確保してください。
- 更新は通常、数分以内に完了しますが、システムに負荷がかかっている場合は、数時間または数日かかることがあります。ドメインを更新するときは、長時間の更新を避けるため、あらかじめ設定したオフピークの時間帯に行うことを検討します。

> **Note**
>
> `UseLatestServiceSoftwareForBlueGreen` 設定はデフォルトで無効になっています。有効にすると、ドメインへのブルー/グリーンデプロイによって、利用可能な最新のサービスソフトウェア更新と設定変更が自動的に適用されます。この設定を有効にするには、`UpdateDomainConfig` API または を使用します AWS CLI。

サービスのソフトウェア更新を開始する

OpenSearch Service コンソール、、またはいずれかの SDKs を使用して AWS CLI、サービスソフトウェアの更新をリクエストできます。

コンソール

サービスのソフトウェア更新をリクエストするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [Actions] (アクション)、[Update] (更新) の順に選択し、次のオプションの中から 1 つを選択します。

   - [Apply update now] (今すぐ更新を適用) - 容量に空きがあれば、アクションを現在の時間で行うよう、すぐにスケジュールします。容量に空きがなければ、別の他の時間帯を選択できます。

- [Schedule it in off-peak window] (オフピークウィンドウにスケジュールする) — ドメインでオフピークウィンドウが有効になっている場合のみ選択できます。ドメインで設定されたオフピークの時間帯に更新を行うようにスケジュールします。更新が、その次の時間帯に行われるという保証はありません。容量によっては翌日になる場合もあります。詳細については、「the section called “オフピークウィンドウ”」を参照してください。
- [Schedule for specific date and time] (特定の日時にスケジュールする) — 特定の日時に更新が行われるようにスケジュールします。指定した日時が容量上の理由で利用できない場合は、別の時間帯を選択できます。

更新を後日 (ドメインのオフピークウィンドウ中かそれ以外) にスケジュールした場合は、いつでもスケジュール変更できます。手順については、「the section called “アクションのスケジュール変更”」を参照してください。

4. [確認] を選択します。

AWS CLI

startstart-service-software-update AWS CLI リクエストを送信して、サービスソフトウェアの更新を開始します。こちらの例では、更新がすぐにキューに追加されています。

```
aws opensearch start-service-software-update \
  --domain-name my-domain \
  --schedule-at "NOW"
```

レスポンス:

```
{
    "ServiceSoftwareOptions": {
        "CurrentVersion": "R20220928-P1",
        "NewVersion": "R20220928-P2",
        "UpdateAvailable": true,
        "Cancellable": true,
        "UpdateStatus": "PENDING_UPDATE",
        "Description": "",
        "AutomatedUpdateDate": "1969-12-31T16:00:00-08:00",
        "OptionalDeployment": true
    }
}
```

> **Tip**
>
> 更新をリクエストした後に、わずかな時間ですが、これをキャンセルできる時間帯があります。このPENDING_UPDATE状態の期間は大きく異なり、 AWS リージョン と OpenSearch Service が実行している同時更新の数によって異なります。更新をキャンセルするには、 コンソールまたは `cancel-service-software-update` AWS CLI コマンドを使用します。

リクエストが `BaseException` により失敗した場合は、指定した日時が容量上の理由で利用できないことを意味し、別の日時を指定する必要があります。OpenSearch Service が、応答で、利用可能な別の時間帯を提案します。

AWS SDKs

このサンプル Python スクリプトは、 の describe_domain メソッドと start_service_software_update メソッド AWS SDK for Python (Boto3) を使用して、ドメインがサービスソフトウェア更新の対象であるかどうかを確認し、対象である場合は更新を開始します。`domain_name` の値を指定する必要があります。

```
import boto3
from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

my_config = Config(
    # Optionally lets you specify a Region other than your default.
    region_name='us-east-1'
)

domain_name = ''  # The name of the domain to check and update

client = boto3.client('opensearch', config=my_config)


def getUpdateStatus(client):
    """Determines whether the domain is eligible for an update"""
    response = client.describe_domain(
        DomainName=domain_name
```

```
    )
    sso = response['DomainStatus']['ServiceSoftwareOptions']
    if sso['UpdateStatus'] == 'ELIGIBLE':
        print('Domain [' + domain_name + '] is eligible for a service software update
 from version ' +
              sso['CurrentVersion'] + ' to version ' + sso['NewVersion'])
        updateDomain(client)
    else:
        print('Domain is not eligible for an update at this time.')


def updateDomain(client):
    """Starts a service software update for the eligible domain"""
    response = client.start_service_software_update(
        DomainName=domain_name
    )
    print('Updating domain [' + domain_name + '] to version ' +
          response['ServiceSoftwareOptions']['NewVersion'] + '...')
    waitForUpdate(client)


def waitForUpdate(client):
    """Waits for the domain to finish updating"""
    response = client.describe_domain(
        DomainName=domain_name
    )
    status = response['DomainStatus']['ServiceSoftwareOptions']['UpdateStatus']
    if status == 'PENDING_UPDATE' or status == 'IN_PROGRESS':
        time.sleep(30)
        waitForUpdate(client)
    elif status == 'COMPLETED':
        print('Domain [' + domain_name +
              '] successfully updated to the latest software version')
    else:
        print('Domain is not currently being updated.')

def main():
    getUpdateStatus(client)
```

オフピークの時間帯にソフトウェア更新をスケジュールする

2023 年 2 月 16 日以降に作成された各 OpenSearch Service ドメインでは、現地時間の毎日午後 10 時から午前 8 時までの 10 時間が、オフピークの時間帯とみなされています。OpenSearch Service

は、この時間帯を利用して、ドメインにサービスソフトウェアの更新をスケジュールします。オフピークの時間帯に更新を行うことにより、トラフィックの多い時間帯にクラスターの専用マスターノードにかかる負荷を、最小限に抑えることができます。OpenSearch Service は、ユーザーの同意なしに、この 10 時間以外の時間帯に更新を開始することはできません。

- オプションの更新の場合、OpenSearch Service は、更新が提供されたことをユーザーに通知し、次回のオフピークウィンドウに更新をスケジュールするように指示します。
- 必須の更新の場合、OpenSearch Service は、次回のオフピークウィンドウに自動的に更新をスケジュールし、更新の 3 日前にユーザーに通知します。この更新は (オフピークウィンドウ中かそれ以外に) スケジュール変更できますが、更新を完了すべき時間枠内に限られます。

ドメインごとに、デフォルトの午後 10 時の開始時刻を、カスタムの時刻で上書きすることができます。手順については、「the section called “カスタムのオフピークウィンドウの設定”」を参照してください。

コンソール

次回のオフピークウィンドウに更新をスケジュールするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [アクション]、[更新] の順に選択します。
4. [Schedule it in off-peak window] (オフピークの時間帯に更新をスケジュール) を選択します。
5. [確認] を選択します。

スケジュール済みのアクションは [Off-peak window] (オフピークウィンドウ) タブに表示され、いつでもスケジュール変更できます。「the section called “スケジュール済みのアクションを表示する”」を参照してください。

CLI

を使用して今後のオフピークウィンドウ中に更新をスケジュールするには AWS CLI、StartServiceSoftwareUpdate リクエストを送信し、 `--schedule-at`パラメータ`OFF_PEAK_WINDOW`に を指定します。

```
aws opensearch start-service-software-update \
  --domain-name my-domain \
```

```
  --schedule-at "OFF_PEAK_WINDOW"
```

サービスソフトウェア更新のモニタリング

OpenSearch Service はサービスのソフトウェア更新が利用可能になったとき、必要なとき、開始したとき、完了したとき、または失敗したときに、通知を送信します。これらの通知は、OpenSearch Service コンソールの [Notifications] (通知) パネルで確認できます。通知の重要度は、それが必要な場合に更新がオプションおよび High の場合、Informational です。

OpenSearch Service は、サービスソフトウェアイベントの、Amazon EventBridge への送信も行います。EventBridge を使用して、イベントの受信時に E メールを送信したり、特定のアクションを実行したりするルールを設定できます。ウォークスルーの例については、「」を参照してくださいthe section called “チュートリアル: 利用可能な更新に関する SNS アラートの送信”。

Amazon EventBridge に送信される各サービスソフトウェアイベントの形式を表示する場合は、「the section called “サービスソフトウェア更新イベント”」を参照してください。

ドメインが更新の対象でない場合

次のテーブルに示されているいずれかの状態にある場合、ドメインはサービスソフトウェア更新の対象外である可能性があります。

状態	説明
処理中のドメイン	ドメインは、設定変更中です。オペレーション完了後、更新の適格性を確認します。
赤のクラスター状態	クラスター内の 1 つ以上のインデックスが赤です。トラブルシューティングステップについては、「the section called “赤のクラスター状態”」を参照してください。
高いエラー率	OpenSearch クラスターはリクエストの処理を試みましたが、大量の 5xx エラーを返しています。この問題は通常、同時読み取りまたは書き込みリクエストが多すぎることで起こります。クラスターへのトラフィックを減らすか、ドメインのスケーリングを検討してください。
スプリットブレイン	スプリットブレインとは、OpenSearch クラスターに 1 つ以上のマスターノードがあり、決して独自に再結合することのない 2 つのクラスターに分割されているという意味です。専用マスターノードの推奨値を

状態	説明
	使用することで、スプリットブレインを回避できます。スプレッドブレインからの復旧方法については、サポート にお問い合わせください。
Amazon Cognito の統合の問題	ドメインで OpenSearch Dashboards の認証を使用しており、OpenSearch Service では 1 つ以上の Amazon Cognito リソースが見つかりません。この問題は通常、Amazon Cognito ユーザープールがない場合に発生します。この問題を修正するには、見つからないリソースを再作成し、そのリソースを使用するように OpenSearch Service ドメインを設定します。
その他の サービスの問題	OpenSearch Service 自体の問題により、ドメインが更新対象外として表示される可能性があります。前の条件がドメインに当てはまらず、問題が 1 日以上続く場合は、サポート にお問い合わせください。

サービスソフトウェア更新のロールバック

Amazon OpenSearch Service は、サービスソフトウェア更新のロールバックをサポートしています。OpenSearch Service コンソール、、 AWS CLIまたはいずれかの AWS SDKs を使用してロールバックを開始できます。ソフトウェアの更新は、ブルー/グリーンデプロイを使用してロールバックされます。

> **Note**
>
> ロールバックは、2026 年 4 月 24 日以降に適用されるサービスソフトウェアの更新でのみサポートされます。エンジンバージョンのアップグレードはロールバックできません。

ロールバックの適格性と考慮事項

ロールバックを使用するには、ドメインが以下の条件をすべて満たしている必要があります。

- ドメインがアクティブ – ドメインは `Active`状態である必要があります。ロールバックは、設定変更、ソフトウェア更新、またはその他の Blue/Green デプロイの進行中は使用できません。
- ロールバック時間ウィンドウ内 – ソフトウェアの更新は、許可されたロールバック期間内に適用されている必要があります。「the section called “ロールバック時間枠”」を参照してください。

- 更新後の設定変更なし – ソフトウェア更新の適用後にドメインの設定変更を行った場合、ロールバックはブロックされます。設定の変更は、以前のソフトウェアバージョンと互換性のない方法でドメイン設定に影響を与える可能性があります。例えば、インスタンスタイプの変更、レプリカの有効化、更新後のストレージ設定の調整は、ロールバックをブロックします。
- 更新がサービスによって自動適用されなかった – 30 日間の可用性ウィンドウ内に必須の更新を実行せず、OpenSearch Service が自動的に更新を適用した場合、その更新でロールバックは使用できません。
- 更新はエンジンバージョンのアップグレードではありません – エンジンバージョンの変更 (OpenSearch 1.3 から OpenSearch 2.11 へのアップグレードなど) は元に戻せません。サービスソフトウェアバージョンのロールバックのみがサポートされています。
- 以前のソフトウェアバージョンが存在する — ロールバックは、この機能が一般公開 (GA) された後に適用されるソフトウェア更新でのみ使用できます。GA 以降にまだソフトウェア更新を行っていないドメインには、ロールバックする以前のバージョンがありません。
- 更新ごとに許可されるロールバックは 1 つだけ – ロールバックが完了すると、ドメインはロールバック状態と見なされます。新しいソフトウェア更新が正常に適用されるまで、ロールバックすることはできません。
- ソフトウェア更新が明示的に開始された – ロールバックは、ソフトウェア更新がユーザーによって明示的に開始されたか、必須またはオプションの更新の一部として OpenSearch Service によって自動的にスケジュールされた場合にのみ使用できます。ロールバックは、次の場合には使用できません。
  - ソフトウェア更新は、設定変更によってトリガーされたブルー/グリーンデプロイ中に適用されます。これには、`UseLatestServiceSoftwareForBlueGreen`設定が有効になっている場合も含まれます。これにより、最新のソフトウェア更新と設定変更がバンドルされます。この設定はデフォルトで無効になっています。
  - 更新は、インフラストラクチャの復旧、自動修復、ドメインのヘルスと可用性を維持するために OpenSearch Service によって実行されるその他の内部オペレーションなど、ドメインのサービス開始メンテナンスオペレーション中に適用されました。

ロールバック時間枠

ロールバックウィンドウは、適用されたソフトウェア更新のタイプによって異なります。

[Update type] (更新タイプ)	ロールバックウィンドウ	注意事項
オプションです。	更新が適用された日から 15 日間	更新を手動で適用したか、スケジュールしたか、オフピーク時間に自動的に適用したかにかかわらず適用されます。
必須	更新が適用された日から 15 日間	更新を 30 日間の可用性ウィンドウ内に自分で適用した場合にのみ使用できます。期限後にサービスが更新を自動適用した場合、ロールバックは使用できません。

ロールバックウィンドウの有効期限が切れると、セルフサービスのロールバックは使用できなくなります。ウィンドウの閉鎖後に重大な問題が発生した場合は、 サポート にお問い合わせください。

ロールバックをリクエストする方法

`RollbackServiceSoftwareUpdate` API を使用してロールバックをリクエストできます。

[リクエスト]

```
POST /2021-01-01/opensearch/serviceSoftwareUpdate/rollback
```

リクエスト本文:

```
{
    "DomainName": "your-domain-name"
}
```

[応答]

API は、エラー以外のすべてのケースで HTTP 200 レスポンスを返します。レスポンス本文の `RollbackAvailable`フィールドは、ロールバックが開始されたかどうかを示します。

```
{
    "RollbackServiceSoftwareOptions": {
        "CurrentVersion": "string",
        "NewVersion": "string",
```

```
        "RollbackAvailable": boolean,
        "Description": "string"
    }
}
```

レスポンスフィールド:

フィールド	タイプ	説明
CurrentVersion	文字列	ドメインで現在実行されているソフトウェアバージョン。
NewVersion	String	ドメインがロールバックされるソフトウェアバージョン。ロールバックが利用できない場合は null を返します。
RollbackAvailable	ブール値	true ロールバックが正常に開始されたかどうか。falseドメインがロールバックの対象でない場合。
Description	String	結果またはロールバックが使用できない理由を説明する人間が読めるメッセージ。

エラーレスポンス

エラー	説明
ResourceNotFoundException	指定されたドメインは存在しません。
ValidationException	ドメインがアクティブ状態ではありません。進行中の変更が完了するのを待ってから再試行します。
InternalException	サービスで内部エラーが発生しました。リクエストを再試行します。
DisabledOperationException	ロールバックオペレーションは、このドメインではサポートされていません。

レスポンスメッセージ

次の表は、 RollbackServiceSoftwareUpdate API によって返される可能性のあるレスポンスメッセージを示しています。

ケース	説明
Success	ロールバックが正常に開始されました。ドメインは、######から######にロールバックされます。
機能が有効になっていない	ロールバックは使用できません。サポートサポートが必要な場合は、 にお問い合わせください。
以前のソフトウェアは利用できません	ロールバックに使用できる以前のソフトウェアバージョンはありません。サポートサポートが必要な場合は、 にお問い合わせください。
以前のソフトウェアは現在のソフトウェアと同じです	ロールバックは使用できません。ロールバックに使用できる以前のソフトウェアバージョンはありません。サポートサポートが必要な場合は、 にお問い合わせください。
ドメインは既にロールバックされています	ドメインは既にロールバック状態です。
ロールバックは使用できません	ロールバックは使用できません。サービス側の設定により、このドメインのロールバックが妨げられています。サポートサポートが必要な場合は、 にお問い合わせください。
時間枠外	ロールバックは使用できません。15 日間のロールバックウィンドウの有効期限が切れています。サポートサポートが必要な場合は、 にお問い合わせください。
必須のサービス更新の自動適用	ロールバックは使用できません。現在のバージョンは、必須のサービス更新を通じて適用されました。サポートサポートが必要な場合は、 にお問い合わせください。
エンジンバージョンのアップグレード	ロールバックは使用できません。エンジンバージョンのアップグレードは元に戻せません。サポートサポートが必要な場合は、 にお問い合わせください。

ケース	説明
更新後に設定が変更されました	ロールバックは使用できません。クラスター設定は、前回のソフトウェア更新以降に変更されました。サポートサポートが必要な場合は、 にお問い合わせください。

レスポンスの例:

```
{
    "RollbackServiceSoftwareOptions": {
        "CurrentVersion": "OpenSearch_2_11_R20240115",
        "NewVersion": "OpenSearch_2_11_R20231023",
        "RollbackAvailable": true,
        "Description": "Rollback initiated successfully. The domain will be rolled back
 from OpenSearch_2_11_R20240115 to OpenSearch_2_11_R20231023."
    }
}
```

ロールバックが利用できない場合、 は false RollbackAvailableを返し、 Descriptionフィールドには上記の表の理由が含まれます。

Amazon OpenSearch Service のオフピークウィンドウの定義

Amazon OpenSearch Service ドメインを作成するときは、オフピークの時間として、毎日 10 時間の時間帯を定義します。OpenSearch Service は、サービスソフトウェアの更新や、ブルー/グリーンデプロイを必要とする Auto-Tune の最適化を、トラフィックが比較的少ないこの時間帯に可能な限りスケジュールします。ブルー/グリーンとは、ドメインの更新用に新しい環境を作成し、この更新の完了後にユーザーを新しい環境にルーティングするプロセスを指します。

ブルー/グリーンデプロイは無停止ですが、リソースがブルー/グリーンデプロイに使用されている間の、パフォーマンスへの潜在的影響を最小限に抑えるため、このようなデプロイは、ドメインに設定されているオフピークウィンドウ中にスケジュールすることが推奨されます。ノードの交換や、ドメインにただちにデプロイすべき更新などでは、オフピークウィンドウは使用しません。

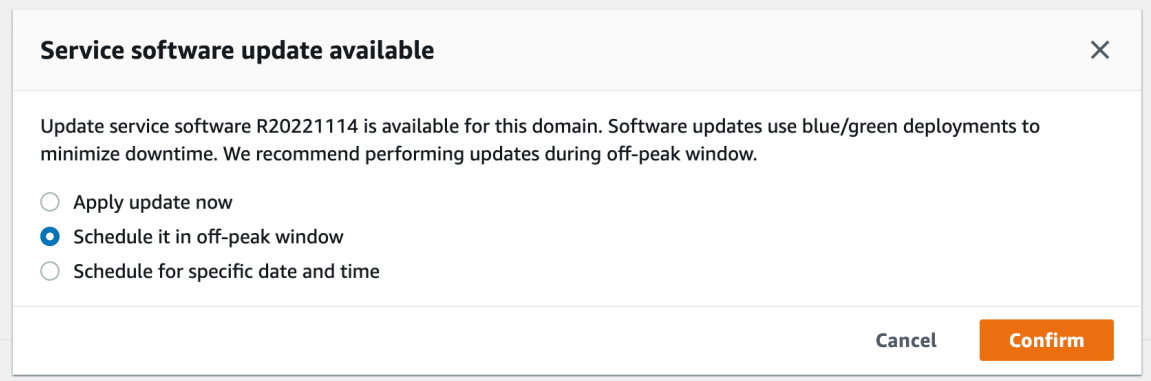
オフピークウィンドウの開始時刻は変更が可能ですが、ウィンドウの所要時間は変更できません。

> **Note**
>
> オフピークウィンドウは 2023 年 2 月 16 日に導入されました。この日付以前に作成されたすべてのドメインでは、オフピークウィンドウはデフォルトで無効になっています。これらのドメインでは、オフピークウィンドウを手動で有効にし、設定する必要があります。この日付以後に作成されたすべてのドメインで、オフピークウィンドウはデフォルトで有効になっています。ドメインのオフピークウィンドウは、いったん有効にした後は、無効にできません。

オフピーク時のサービスソフトウェアの更新

OpenSearch Service のサービスソフトウェア更新には、オプションと必須という 2 つのカテゴリがあります。どちらのカテゴリにもブルー/グリーンデプロイが必要です。オプションの更新はお使いのドメインに強制されることはありませんが、必須の更新は、指定された期限 (通常は提供開始の 2 週間後) までにアクションを実行しないと自動的にインストールされます。詳細については、「the section called “オプションの更新と必須の更新”」を参照してください。

オプションの更新を実行するときは、更新をすぐに適用するか、次回のオフピークウィンドウ時にスケジュールする、またはカスタムの日時を指定して適用するかの、いずれかを選択できます。



必須の更新の場合、OpenSearch Service がオフピークウィンドウ中の日時で更新を自動的にスケジュールします。ユーザーには、スケジュールされた更新の 3 日前に通知が届きます。必須のデプロイの期間内であれば、更新の日時を後の日時に変更することも可能です。手順については、「the section called “アクションのスケジュール変更”」を参照してください。

オフピーク時の Auto-Tune の最適化

これまで、Auto-Tune では、ブルー/グリーンデプロイが必要なスケジュールを変更する場合、メンテナンスウィンドウが使用されていました。オフピークウィンドウが導入される前に、Auto-Tune と

メンテナンスウィンドウが有効になっていたドメインでは、オフピークウィンドウを使用するようにドメインを移行しない限り、更新には引き続きメンテナンスウィンドウが使用されます。

当社では、オフピークウィンドウを使用するようにドメインを移行することを推奨しています。サービスソフトウェアの更新など、ドメインの他のアクティビティのスケジュールに使用するためです。手順については、「the section called “Auto-Tune のメンテナンスウィンドウからの移行”」を参照してください。ドメインをオフピークウィンドウに移行した後は、メンテナンスウィンドウの使用に戻ることはできません。

2023 年 2 月 16 日以降に作成されたすべてのドメインでは、ブルー/グリーンデプロイのスケジュールでは、従来のメンテナンスウィンドウではなくオフピークウィンドウを使用します。ドメインのオフピークウィンドウは無効にできません。ブルー/グリーンデプロイを必要とする Auto-Tune 最適化のリストについては、「the section called “変更のタイプ”」を参照してください。

オフピークウィンドウの有効化

2023 年 2 月 16 日 (オフピークウィンドウの導入日) 以前に作成されたドメインでは、この機能はデフォルトで無効になっています。これらのドメインでは、オフピークウィンドウを手動で有効にする必要があります。ドメインのオフピークウィンドウは、有効にした後は無効にできません。

コンソール

ドメインのオフピークウィンドウを有効にするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [オフピークウィンドウ] タブに移動し、[編集] を選択します。
4. 協定世界時 (UTC) でカスタムの開始時刻を指定します。例えば、開始時刻を米国西部 (オレゴン) リージョンの午後 11 時 30 分に設定するときは、[07:30] と指定します。
5. [Save changes] (変更の保存) をクリックします。

CLI

を使用してオフピークウィンドウを変更するには AWS CLI、UpdateDomainConfig リクエストを送信します。

```
aws opensearch update-domain-config \
```

```
  --domain-name my-domain \
  --off-peak-window-options 'Enabled=true,
OffPeakWindow={WindowStartTime={Hours=02,Minutes=00}}'
```

カスタムのウィンドウ開始時刻を指定しない場合、00:00 UTC がデフォルトの時刻となります。

カスタムのオフピークウィンドウの設定

ドメインのカスタムオフピークウィンドウを協定世界時 (UTC) で指定します。例えば、オフピークウィンドウを米国東部 (バージニア北部) リージョンのドメインで午後 11 時に開始するときは、[04:00 UTC] と指定します。

コンソール

ドメインのオフピークウィンドウを変更するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [オフピークウィンドウ] タブに移動します。設定済みのオフピークウィンドウと、今後予定されているドメインのアクション一覧が表示されます。
4. [編集] を選択し、新しい開始時刻を UTC で指定します。例えば、開始時刻を米国東部 (バージニア北部) リージョンの午後 9 時に設定するときは、[02:00 UCT] と指定します。
5. [Save changes] (変更の保存) をクリックします。

CLI

を使用してカスタムオフピークウィンドウを設定するには AWS CLI、UpdateDomainConfig リクエストを送信し、24 時間形式で時間と分を指定します。

例えば、次のリクエストを送信すると、ウィンドウの開始時刻が UTC の午前 2 時に変更されます。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --off-peak-window-options 'OffPeakWindow={WindowStartTime={Hours=02,Minutes=00}}'
```

ウィンドウの開始時刻を指定しなかった場合、ドメインが作成された AWS リージョン の現地時間午後 10 時がデフォルトで開始時刻となります。

スケジュール済みのアクションを表示する

各ドメインの現在スケジュール済みのアクション、進行中のアクション、保留中のアクションは、すべて表示できます。アクションには、HIGH、MEDIUM、LOW の重要度のレベルがあります。

アクションのステータスは次の通りです。

- `Pending update` — アクションはキューに入って処理待ちです。
- `In progress` — アクションは進行中です。
- `Failed` - アクションの実行に失敗しました。
- `Completed` – アクションは正常に完了しました。
- `Not eligible` — サービスソフトウェアの更新のみです。クラスターが異常な状態であるため、更新を続行できません。
- `Eligible` — サービスソフトウェアの更新のみです。このドメインは更新の対象になっています。

コンソール

OpenSearch Service コンソールには、ドメイン設定内でスケジュールされているすべてのアクションが、各アクションの重要度や現在のステータスと併せて表示されます。

ドメインでスケジュールされているアクションを表示するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [オフピークウィンドウ] タブに移動します。
4. [スケジュールされたアクション] の下に、ドメインで現在スケジュールされているアクション、進行中のアクション、保留中のアクションがすべて表示されます。

CLI

を使用してスケジュールされたアクションを表示するには AWS CLI、ListScheduledActions リクエストを送信します。

```
aws opensearch list-scheduled-actions \
  --domain-name my-domain
```

レスポンス:


```
{
    "ScheduledActions": [
        {
            "Cancellable": true,
            "Description": "The Deployment type is : BLUE_GREEN.",
            "ID": "R20220721-P13",
            "Mandatory": false,
            "Severity": "HIGH",
            "ScheduledBy": "CUSTOMER",
            "ScheduledTime": 1.673871601E9,
            "Status": "PENDING_UPDATE",
            "Type": "SERVICE_SOFTWARE_UPDATE",
        },
        {
            "Cancellable": true,
            "Description": "Amazon Opensearch will adjust the young generation JVM
 arguments on your domain to improve performance",
            "ID": "Auto-Tune",
            "Mandatory": true,
            "Severity": "MEDIUM",
            "ScheduledBy": "SYSTEM",
            "ScheduledTime": 1.673871601E9,
            "Status": "PENDING_UPDATE",
            "Type": "JVM_HEAP_SIZE_TUNING",
        }
    ]
}
```


アクションのスケジュール変更

OpenSearch Service からユーザーに、サービスソフトウェアの更新と Auto-Tune の最適化がスケジュールされている旨の通知が届きます。スケジュールの変更は、すぐに適用することもできれば、別の日時に変えることもできます。

> **Note**
>
> OpenSearch Service では、選択した時刻から 1 時間以内にアクションをスケジュールできます。例えば、更新を午後 5 時に適用する場合、午後 5 時から 6 時の間に適用できます。

コンソール

アクションのスケジュールを変更するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ドメイン名を選択し、設定を開きます。
3. [オフピークウィンドウ] タブに移動します。
4. [スケジュールされたアクション] の下で、アクションを選択し、[スケジュール変更] を選択します。
5. 以下のオプションのいずれかを選択してください。
   - [今すぐ更新を適用] - 容量に空きがあれば、アクションを現在の時間で行うよう、すぐにスケジュールします。容量に空きがなければ、別の他の時間帯を選択できます。
   - [オフピークウィンドウでのスケジュール設定] - 次回のオフピークウィンドウ中にピックアップするようにアクションにマークします。変更が、次回のウィンドウ中に適用されるという保証はありません。容量によっては翌日になる場合もあります。
   - [この更新のスケジュールを変更する] - 変更を適用する日付と時刻をカスタムで指定できます。指定した日時が容量上の理由で利用できない場合は、別の時間帯を選択できます。
   - [スケジュールされた更新のキャンセル] - 更新をキャンセルします。このオプションを利用できるのは、オプションのサービスソフトウェア更新のみです。Auto-Tune アクション、または必須のソフトウェア更新には利用できません。
6. [Save changes] (変更の保存) をクリックします。

CLI

を使用してアクションを再スケジュールするには AWS CLI、UpdateScheduledAction リクエストを送信します。アクション ID を取得するには、`ListScheduledActions` リクエストを送信します。

次のリクエストを送信すると、サービスソフトウェアの更新が特定の日時にスケジュール変更されます。

```
aws opensearch update-scheduled-action \
  --domain-name my-domain \
  --action-id R20220721-P13 \
  --action-type "SERVICE_SOFTWARE_UPDATE" \
  --desired-start-time 1677348395000 \
```

```
  --schedule-at TIMESTAMP
```

レスポンス:

```
{
   "ScheduledAction": {
      "Cancellable": true,
      "Description": "Cluster status is updated.",
      "Id": "R20220721-P13",
      "Mandatory": false,
      "ScheduledBy": "CUSTOMER",
      "ScheduledTime": 1677348395000,
      "Severity": "HIGH",
      "Status": "PENDING_UPDATE",
      "Type": "SERVICE_SOFTWARE_UPDATE"
   }
}
```

リクエストが SlotNotAvailableException により失敗した場合は、指定した日時が容量上の理由で利用できないことを意味し、別の日時を指定する必要があります。OpenSearch Service が、応答で、利用可能な別の時間帯を提案します。

Auto-Tune のメンテナンスウィンドウからの移行

ドメインが 2023 年 2 月 16 日以前に作成された場合、ブルー/グリーンデプロイを必要とする Auto-Tune 最適化のスケジュールに、メンテナンスウィンドウを使用できます。既存の Auto-Tune ドメインを移行して、代わりにオフピークウィンドウを使用します。

> **Note**
>
> オフピークウィンドウを使用するようにドメインを移行した後は、メンテナンスウィンドウの使用に戻ることはできません。

コンソール

オフピークウィンドウを使用するようにドメインを移行するには

1. Amazon OpenSearch Service コンソールで、ドメインの名前を選択し、設定を開きます。
2. [Auto-Tune] タブに進み、[編集] を選択します。

3. [オフピークウィンドウへ移行] を選択します。
4. [開始時刻 (UTC)] に、オフピークウィンドウの開始時刻を協定世界時 (UTC) で入力します。
5. [Save changes] (変更の保存) をクリックします。

CLI

を使用して Auto-Tune メンテナンスウィンドウからオフピークウィンドウに移行するには AWS CLI、UpdateDomainConfig リクエストを送信します。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --auto-tune-options
 DesiredState=ENABLED,UseOffPeakWindow=true,MaintenanceSchedules=[]
```

ドメインを Auto-Tune メンテナンスウィンドウからオフピークウィンドウへ移行するには、オフピークウィンドウをオンにします。オフピークウィンドウは、個別のリクエストで、または同じリクエストで有効にすることができます。手順については、「the section called “オフピークウィンドウの有効化”」を参照してください。

Amazon OpenSearch Service での通知

Amazon OpenSearch Service の通知には、ドメインのパフォーマンスとヘルスに関する重要な情報が含まれています。OpenSearch Service は、サービスソフトウェアの更新、自動チューニングの強化、クラスターヘルスイベント、およびドメインエラーについて通知します。通知は、すべてのバージョンの OpenSearch および Elasticsearch OSS で利用できます。

通知は、OpenSearch Service コンソールの [Notifications] (通知) パネルで確認できます。OpenSearch サービスの通知はすべて Amazon EventBridge にも表示されます。通知とサンプルイベントの完全なリストについては、「the section called “イベントのモニタリング”」を参照してください。

通知の使用開始

通知は、ドメインの作成時に自動的に有効になります。OpenSearch Service コンソールの通知パネルに進み、通知のモニタリングと確認を行います。各通知には、通知が投稿された時刻、関連するドメイン、重要度とステータスレベル、簡単な説明などの情報が含まれます。コンソールでは、最大 90 日間の履歴通知を表示できます。

[通知] パネルにアクセスした後または通知を確認した後に、es:ListNotificationsV2 または es:UpdateNotificationStatus を実行するアクセス許可がないことを示すエラーメッセージが表示されることがあります。この問題を解決するには、IAM でユーザーまたはロールに次のアクセス許可を付与します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "es:DescribeDomain",
      "es:ListDomainNames"
    ],
    "Resource": "arn:aws:es:*:111122223333:domain/*"
  }]
}
```

IAM コンソールは、エラー (「IAM does not recognize one or more actions.」) をスローしますが、このエラーは安全に無視できます。また、特定のドメインに対する es:UpdateNotificationStatus アクションを制限することもできます。詳細についてはthe section called “ポリシーエレメントのリファレンス”を参照してください。

通知重要度

OpenSearch サービスでの通知は、すでに実行されているアクションまたはドメインのオペレーションに関連する [情報]、または特定のアクション (必須のセキュリティパッチの適用など) を取る必要のある [実用的] です。各通知には、重要度 (Informational、Low、Medium、High、または Critical) が関連付けられています。次の表に重大度の一覧を示します。

緊急度	説明	例
Informational	ドメインのオペレーションに関連する情報。	• サービスソフトウェア更新が利用可能 • Auto-Tune が開始されました
Low	推奨されるアクションですが、アクションを行わない	• Auto-Tune がキャンセルされました

緊急度	説明	例
	場合、ドメインの可用性やパフォーマンスに悪影響を及ぼしません。	• シャードカウントが高い警告
Medium	推奨されるアクションが実行されないが、アクションが実行されるまでの時間枠が長くなった場合、影響が生じる可能性があります。	• サービスソフトウェアの更新に失敗しました • シャード数の制限を超えました
High	悪影響を避けるためには、緊急のアクションが必要です。	• サービスソフトウェア更新が必要 • KMS キーにアクセスできない
Critical	悪影響を避けるために、またはそこから回復するために、即時のアクションが必要です。	現在利用可能なものはありません

サンプル EventBridge イベント

次の例は、Amazon EventBridge に送信される OpenSearch Service の通知イベントを示しています。更新はオプションでなので、通知の重要度は Informational です。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Available",
    "severity": "Informational",
    "description": "Service software update [R20200330-p1] available."
```

```
  }
}
```

Amazon OpenSearch Service でのマルチ AZ ドメインの設定

サービス中断が発生した場合にデータの損失を防ぎ、Amazon OpenSearch Service クラスターのダウンタイムを最小限に抑えるため、同じリージョン内の 2 つまたは 3 つのアベイラビリティーゾーン間にノードを分散できます。これは、マルチ AZ と呼ばれる設定です。アベイラビリティーゾーンは、各 AWS リージョン内の独立した場所です。

プロダクションワークロードを実行するドメインでは、Multi-AZ with Standby デプロイのオプションが推奨されます。このオプションが次の構成を作成します。

- ドメインが 3 つのゾーン間にデプロイされました。
- 専用マスターノードおよびデータノードに、現行世代のインスタンスタイプを選択します。
- 3 つの専用マスターノードと 3 つ (または 3 の倍数) のデータノード。
- ドメイン内の各インデックスに 2 つ以上のレプリカ、または 3 の倍数のデータコピー (プライマリノードとレプリカの両方を含む)。

このセクションの残りの部分では、これらの構成とそのコンテキストについて説明します。

Multi-AZ with Standby

Multi-AZ with Standby は、99.99% の可用性、プロダクションワークロードでの一貫したパフォーマンス、シンプルなドメイン設定と管理とを実現した、Amazon OpenSearch Service ドメインのデプロイオプションです。Multi-AZ with Standby を使用したドメインは、インフラストラクチャの障害に対して回復力があり、パフォーマンスや可用性に一切影響を与えません。このデプロイオプションは、指定されたデータノード数、マスターノード数、インスタンスタイプ、レプリカ数、ソフトウェア更新設定、Auto-Tune の有効化など、数多くのベストプラクティスを義務付けることでこの標準を達成しています。

> **Note**
>
> スタンバイ付きマルチ AZ は OpenSearch バージョン 1.3 以降で利用可能で、少なくとも 3 つのアベイラビリティーゾーンを持つリージョンが必要です。

Multi-AZ with Standby を使用すると、OpenSearch Service が 3 つのアベイラビリティーゾーンを横断してドメインを作成し、データの完全なコピーが各ゾーンに追加され、データは各ゾーンに均等に分散されます。ドメインは、これらのゾーンのいずれかの 1 つのノードをスタンバイとして予約します。このノードは、検索リクエストの処理を行いません。OpenSearch Service が基盤のインフラストラクチャで障害を検出すると、スタンバイノードが 1 分以内に自動的にアクティブ化します。ドメインは、引き続きインデックス作成と検索リクエストを処理し、その影響は、フェイルオーバーの実行に必要な時間に限られます。データやリソースの再配布は行われないため、クラスターのパフォーマンスには影響せず、可用性が下がるリスクもありません。Multi-AZ with Standby は、追加コストなしでご利用いただけます。

AWS マネジメントコンソールでスタンバイ付きのドメインを作成するには、2 つの方法があります。1 つ目では、[簡易作成] の作成方法を使用してドメインを作成します。OpenSearch Service は、次を含む、あらかじめ決められた設定を自動的に使用します。

- 3 つのアベイラビリティーゾーン、1 つはスタンバイとして機能
- 3 つの専用マスターノードとデータノード
- Auto-Tune がドメインで有効になっている
- データノード用の GP3 ストレージ

また、[標準作成] の作成方法を選択し、デプロイオプションとして [スタンバイ状態のドメイン] を選択することもできます。これにより、ドメインをカスタマイズしながら、引き続き 3 つのゾーンと 3 つのマスターノードなどスタンバイの主な機能を要求することができます。データノードの数は 3 の倍数 (アベイラビリティーゾーンの数) を選ぶことが推奨されます。

ドメインを作成したら、ドメインの詳細ページに移動し、[クラスター設定] タブで、アベイラビリティーゾーンに「スタンバイ状態の 3 つの AZ」と表示されていることを確認します。

既存のドメインを Multi-AZ with Standby へ移行する際に問題が生じた場合は、トラブルシューティングガイドの「Multi-AZ with Standby に移行する際のエラー」を参照してください。

制限事項

Multi-AZ with Standby のドメインを設定するときは、次の制限を考慮します。

- ノード上のシャードの合計数は 1000 を超えることはできず、クラスター上のシャードの合計数は 75000 を超えることはできず、1 つのシャードの容量は 65 GB を超えることはできません。

- スタンバイ付きマルチ AZ は、m5、、c5、r5、r6g、c6g、m6g、r6gd、r7gd、、、および r7g c7g m7gi3インスタンスタイプでのみ機能します。サポートされているインスタンスタイプの詳細については、「サポートされるインスタンスタイプ」を参照してください。
- スタンバイではプロビジョンド IOPS SSD、汎用 SSD (GP3)、または instance-backed ストレージのみを使用できます。
- スタンバイ付きマルチ AZ ドメインで UltraWarm を有効にする場合、ウォームノードの数は、使用されるアベイラビリティーゾーンの数の倍数である必要があります。

Multi-AZ without Standby

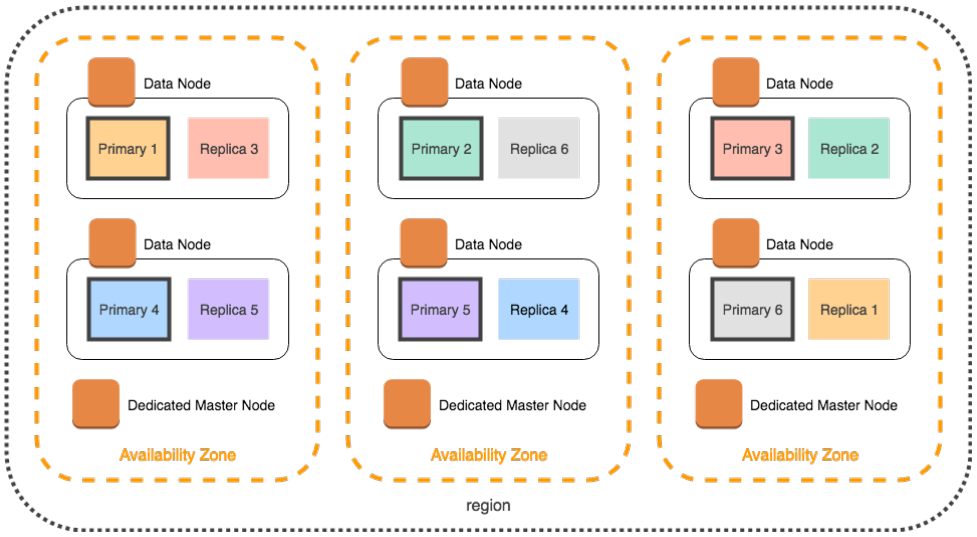
OpenSearch Service は、Multi-AZ without Standby もサポートしています。こちらの可用性は 99.9% です。ノードはアベイラビリティーゾーン全体に分散しており、可用性は、アベイラビリティーゾーンとデータコピーの数に依存します。スタンバイ付きの場合はドメインをベストプラクティスで設定する必要がありますが、スタンバイが付いていない場合は、アベイラビリティゾーン、ノード、レプリカの数を自分で選択できます。この方法は、スタンバイ付きのドメインを作成した場合に既存のワークフローが中断する可能性がない限り、推奨されません。

この方法を選択した場合でも、ノード、ディスク、シングル AZ での障害に対する回復力を維持するため、3 つのアベイラビリティーゾーンを選択することが引き続き推奨されます。障害が発生すると、クラスターは、可用性と冗長性を保つため他のリソースにデータを再分散します。このデータ移動によりクラスターでのリソースの使用量が増加し、パフォーマンスに影響する可能性があります。クラスターの容量が適切でないと可用性は低下します。これでは、マルチ AZ の目的が大きく損なわれかねません。

でスタンバイなしでドメインを設定する唯一の方法は AWS マネジメントコンソール 、標準作成方法を選択し、スタンバイなしでドメインをデプロイオプションとして選択することです。

シャード分散

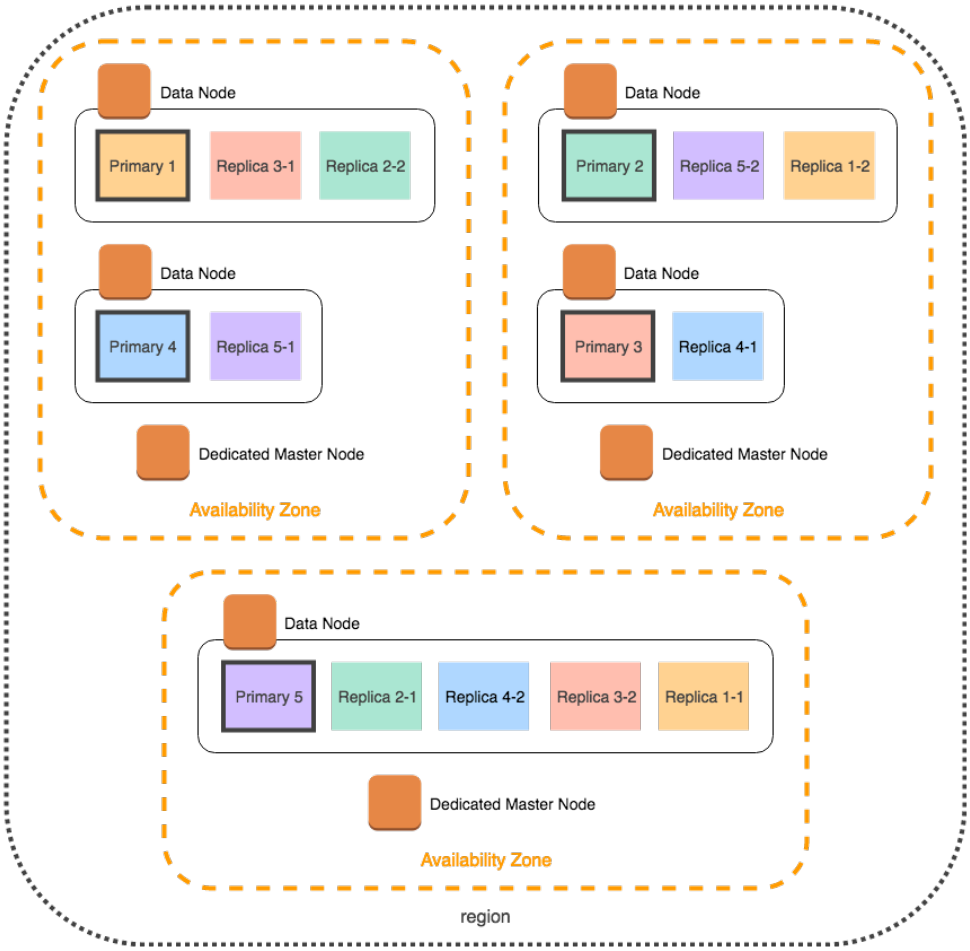
スタンバイの付いていないマルチ AZ を有効にする場合は、クラスターの各インデックスに 1 つ以上のレプリカを作成します。レプリカがないと、OpenSearch Service はデータのコピーを他のアベイラビリティーゾーンに分散することができません。さいわい、すべてのインデックスのデフォルト設定で、レプリカの数は 1 つです。次の図に示すように、OpenSearch Service は、最善の方法でプライマリシャードとそれに対応するレプリカシャードを異なるゾーンに分散します。



シャードをアベイラビリティーゾーンごとに分散することに加えて、OpenSearch Service はノードごとにシャードを分散します。それでも、特定のドメイン設定ではシャード数が不均等にある場合があります。次のドメインを考えます。

- データノード x 5
- プライマリシャード x 5
- レプリカ x 2
- アベイラビリティーゾーン x 3

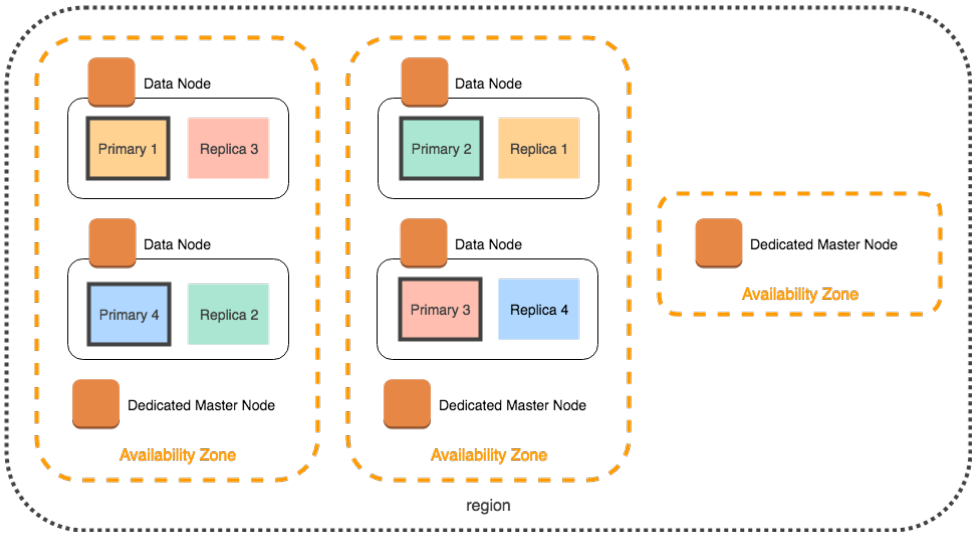
この状況では、次の図に示すように、OpenSearch Service はゾーン間でプライマリシャードとレプリカシャードを分散するために 1 つのノードをオーバーロードする必要があります。



こうした、個々のノードに負荷がかかってパフォーマンスが低下する状況を避けるため、Multi-AZ with Standby を選択するか、インデックスごとのレプリカ数を 2 つ以上にする場合は インスタンスの数を 3 の倍数にすることが推奨されます。

専用マスターノード分散

ドメインを設定するときに 2 つのアベイラビリティーゾーンを選択した場合でも、OpenSearch Service は 3 つのアベイラビリティーゾーン間に専用マスターノードを自動的に分散します。この分散により、ゾーンでサービス中断が発生した場合にクラスターのダウンタイムを防ぐことができます。推奨される 3 つの専用マスターノードを使用していて、1 つのアベイラビリティーゾーンがダウンした場合でも、クラスターには専用マスターノードのクォーラム (2) が残り、新しいマスターを選択できます。この設定は以下の図のようになります。



3 つのアベイラビリティーゾーンで使用できない旧世代のインスタンスタイプを選択した場合、以下のシナリオが適用されます。

- ドメインに 3 つのアベイラビリティーゾーンを選択した場合、OpenSearch Service はエラーをスローします。異なるインスタンスタイプを選択し、もう一度試します。
- ドメインに 2 つのアベイラビリティーゾーンを選択した場合、OpenSearch Service は 2 つのゾーン間に専用マスターノードを分散します。

アベイラビリティーゾーンの中断

アベイラビリティーゾーンの中断が発生することはほとんどありませんが、起こり得ます。以下の表に、各種マルチ AZ 設定と中断時の動作を示します。表の最後の行は Multi-AZ with Standby に適用され、それ以外のすべての行は Multi-AZ without Standby にのみ適用される構成になっています。

リージョン内のアベイラビリティーゾーン数	選択したアベイラビリティーゾーンの数	専用マスターノードの数	1 つのアベイラビリティーゾーンで中断が発生した場合の動作
2 以上	2	0	ダウンタイム。クラスターのデータノードの半分が失われます。マスターを選択する前に残りのアベイラビリティーゾーンで少なくとも 1 つを置き換える必要があります。

リージョン内のアベイラビリティーゾーン数	選択したアベイラビリティーゾーンの数	専用マスターノードの数	1 つのアベイラビリティーゾーンで中断が発生した場合の動作
2	2	3	ダウンタイムの可能性は五分五分です。OpenSearch Service は、2 つの専門マスターノードを 1 つのアベイラビリティーゾーンに、もう 1 つを他のアベイラビリティーゾーンに分散します。 • 専用マスターノードが 1 つのアベイラビリティーゾーンで中断が発生した場合、残りのアベイラビリティーゾーン内の 2 つの専用マスターノードがマスターを選択できます。 • 専用マスターノードが 2 つのアベイラビリティーゾーンで中断が発生した場合、残りのアベイラビリティーゾーンが回復するまでクラスターは使用できません。
3 以上	2	3	ダウンタイムはありません。OpenSearch Service は、専用マスターノードを 3 つのアベイラビリティーゾーン間で自動的に分散するため、残りの 2 つの専用マスターノードがマスターを選択できます。
3 以上	3	0	ダウンタイムはありません。データノードの約 3 分の 2 が引き続きマスターを選択できます。
3 以上	3	3	ダウンタイムはありません。残りの 2 つの専用マスターノードがマスターを選択できます。

原因に関係なくあらゆる設定において、ノードの障害が発生すると、欠落したノードを置き換えるために OpenSearch Service が自動的に新しいノードを設定する間、クラスターの残りのデータノードで非常に大きな負荷がかかる期間が発生する可能性があります。

たとえば、3 ゾーン設定でアベイラビリティーゾーンが中断した場合、3 分の 2 に相当するデータノードがクラスターへのリクエストを処理する必要があります。これらのリクエストを処理するとき、残りのノードはオンラインになる新しいノードにもシャードをレプリケートするため、さらにパフォーマンスに影響が及びます。ワークロードにとって可用性が重要な場合、クラスターにリソースを追加してこの問題を緩和することを検討してください。

> **Note**
>
> OpenSearch Service はマルチ AZ ドメインを透過的に管理するため、アベイラビリティーゾーンの中断を手動でシミュレートすることはできません。

VPC 内で Amazon OpenSearch Service ドメインを起動する

Amazon OpenSearch Service ドメインなどの AWS リソースを Virtual Private Cloud (VPC) で起動できます。VPC は、 専用の仮想ネットワークです AWS アカウント。クラウド内の他の仮想ネットワークから論理的に分離されています AWS 。OpenSearch Service ドメインを VPC に配置すると、インターネットゲートウェイ、NAT デバイスや VPN 接続なしで、OpenSearch Service と他のサービス間に VPC 内で安全な通信ができるようになります。すべてのトラフィックは AWS クラウド内に安全に保持されます。

> **Note**
>
> OpenSearch Service ドメインを VPC 内に配置する場合、コンピュータが VPC に接続できる必要があります。多くの場合、この接続では、VPN、Transit Gateway、マネージド型のネットワーク、またはプロキシサーバーを使用します。VPC の外部からドメインに直接アクセスすることはできません。

VPC 対パブリックドメイン

以下に、VPC ドメインとパブリックドメインの違いを示します。それぞれの違いについては、後半で説明します。

- 論理的な隔離により、VPC 内に存在するドメインには、パブリックエンドポイントを使用するドメインに比較して、より拡張されたセキュリティレイヤーがあります。
- パブリックドメインはインターネットに接続されたあらゆるデバイスからアクセスできますが、VPC ドメインには何らかの形式の VPN またはプロキシが必要です。

- パブリックドメインと比較すると、コンソールに表示される VPC ドメインの情報は少なくなります。具体的には、[クラスターヘルス] タブにはシャード情報が含まれておらず、[インデックス] タブは存在しません。
- ドメインエンドポイントは、異なる形式 (`https://search-domain-name` 対 `https://vpc-domain-name`) を取ります。
- セキュリティグループには IP ベースのアクセス権限ポリシーですでに強化されているため、VPC 内に存在するドメインに IP ベースのアクセス権限ポリシーを適用することはできません。

制限事項

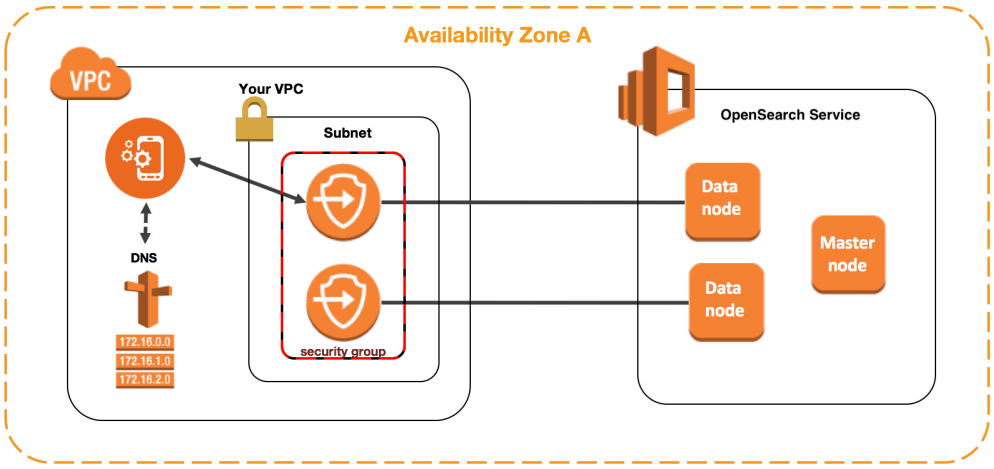
VPC 内の OpenSearch Service ドメイン運用には次の制限があります。

- VPC 内に新規ドメインを起動する場合、後でパブリックエンドポイントの使用に切り替えることはできません。その逆も同じ結果となります。パブリックエンドポイントでドメインを作成する場合、後で VPC に配置することはできません。代わりに、新規のドメインを作成して、データを移行する必要があります。
- VPC 内でドメインを起動すること、あるいはパブリックエンドポイントを使用することができますが、両方を実行することはできません。ドメイン作成時にどちらかを選択する必要があります。
- 専有テナントを使用している VPC 内でドメインを起動することはできません。テナントを [デフォルト] に設定した VPC を使用する必要があります。
- VPC 内にドメインをセットした後で、そのドメインを別の VPC に移動することはできませんが、サブネットとセキュリティグループの設定は変更可能です。
- VPC 内に存在するドメイン用に OpenSearch Dashboards のデフォルトインストールにアクセスするには、VPC へのアクセスを保持している必要があります。このプロセスはネットワーク構成によって異なりますが、VPN への接続、ネットワークの管理あるいはプロキシサーバーまたは Transit Gateway の使用が必要となる場合がほとんどです。詳細については、「the section called “VPC ドメインのアクセスポリシーについて”」、「Amazon VPC ユーザーガイド」および「the section called “Dashboards へのアクセスの制御”」を参照してください。
- VPC を介して Egress を有効にする場合、ドメインの Egress トラフィックは VPC の DNS とネットワークコントロールによって異なります。それに応じてサブネット IP 容量を計画します。「the section called “制限と考慮事項”」を参照してください。

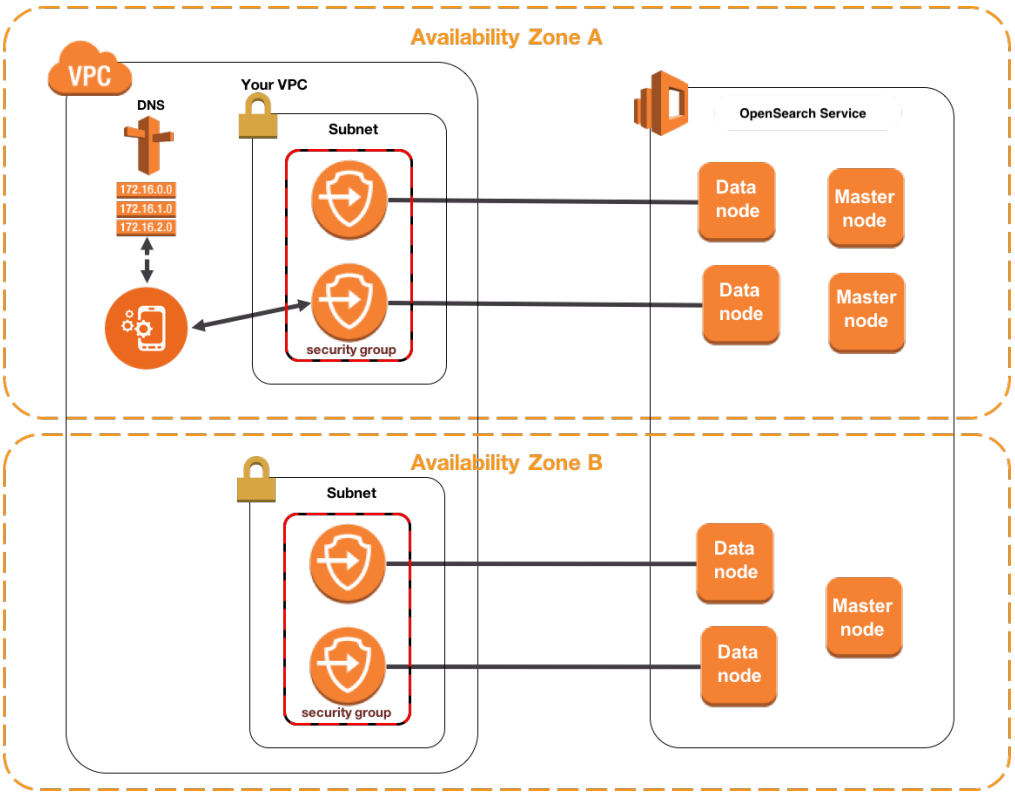
アーキテクチャ

VPC をサポートするために、OpenSearch Service はエンドポイントを VPC の 1 つ、2 つ、または 3 つのサブネットに配置します。ドメインに対して複数のアベイラビリティーゾーンを有効にする場合、各サブネットは同じリージョンの異なるアベイラビリティーゾーンに存在する必要があります。アベイラビリティーゾーンを 1 つしか使用しない場合、OpenSearch Service は 1 つのサブネットのみにエンドポイントを配置します。

次の図は、1 つのアベイラビリティーゾーンの VPC アーキテクチャを示しています。



次の図は、2 つのアベイラビリティーゾーンの VPC アーキテクチャを示しています。



また、OpenSearch Service は各データノード用に VPC 内に Elastic Network Interface (ENI) を配置します。OpenSearch Service は、各 ENI にサブネットの IPv4 アドレス範囲内のプライベート IP アドレスを割り当てます。また、このサービスは、IP アドレスにパブリック DNS ホスト名 (これは、ドメインエンドポイントです) も割り当てます。データノード用に適確な IP アドレスでエンドポイント (DNS ホスト名) を解決するために、パブリック DNS サービスを使用する必要があります。

- VPC が `enableDnsSupport` オプションを `true` (デフォルト値) に設定して Amazon が提供する DNS サーバーを使用する場合、OpenSearch Service エンドポイントの解決は成功します。
- VPC がプライベート DNS サーバーを使用し、このサーバーが DNS ホスト名を解決するためにパブリック権限の DNS サーバーに到達できる場合にも、OpenSearch Service エンドポイントの解決は成功します。

IP アドレスは変更する可能性があるため、ドメインのエンドポイントを定期的に解決して常に正しいデータノードにアクセスできるようにすることが重要です。DNS 解決間隔を 1 分間に設定することが推奨されます。クライアントを使用している場合には、クライアント側の DNS キャッシュもクリアしていることも確認する必要があります。

パブリックアクセスから VPC アクセスに移行する

ドメインを作成するときは、このドメインがパブリックエンドポイントにあるか、あるいは VPC 内に存在するかを指定します。一度作成すると、選択を別のものに切り替えることはできません。代わりに、新規のドメインを作成して、手動で再度インデックスするか、またはデータを移行する必要があります。スナップショットは、データの移行に便利な手段です。スナップショットの作成と復元の詳細については、「the section called “インデックススナップショットの作成”」を参照してください。

VPC ドメインのアクセスポリシーについて

VPC 内に OpenSearch Service ドメインを配置すると、内在する強力なセキュリティ層を提供します。パブリックアクセスのドメインを作成する場合、エンドポイントは次の形式になります。

```
https://search-domain-name-identifier.region.es.amazonaws.com
```

「パブリック」ラベルが示すように、このエンドポイントはすべてのインターネット接続デバイスからアクセスできますが、このアクセスを制御でき、制御する必要があります。ウェブブラウザでエンドポイントにアクセスするときに `Not Authorized` メッセージを受信することがありますが、リクエストはドメインに届きます。

VPC アクセスを使用してドメインを作成する場合、このエンドポイントはパブリックエンドポイントに類似しています。

```
https://vpc-domain-name-identifier.region.es.amazonaws.com
```

ウェブブラウザでエンドポイントにアクセスしようとすると、リクエストがタイムアウトする場合があります。基本的な `GET` リクエストを実行するときでも、コンピューターが VPC に接続していることが必要です。多くの場合、この接続では、VPN、Transit Gateway、マネージド型のネットワーク、またはプロキシサーバーを使用します。使用できるさまざまな形式の詳細については、Amazon VPC ユーザーガイドの「VPC の例」を参照してください 。開発に焦点を置いた例については、「the section called “VPC ドメインをテストする”」を参照してください。

この接続要件に加えて、VPC はセキュリティグループを使用したドメインへのアクセス管理を可能にします。多くのユースケースでは、このセキュリティ機能の組み合わせで十分となり、ドメインにオープンなアクセスポリシーを安心して適用できます。

オープンなアクセスポリシーで操作することは、インターネット上の誰でも OpenSearch Service ドメインにアクセスできることを意味するものではありません。むしろこれは、リクエストが

OpenSearch Service ドメインに到達し、関連付けられたセキュリティグループがそれを許可することは、ドメインがこのリクエストを受け入れることを意味します。唯一の例外は、きめ細かなアクセスコントロール、または IAM ロールを指定するアクセスポリシーを使用している場合です。これらの状況では、ドメインがリクエストを受信するには、セキュリティグループがそのリクエストを許可し、そしてこのリクエストが有効な認証情報で署名されていることが必要です。

> **Note**
>
> セキュリティグループには IP ベースのアクセス権限ポリシーですでに強化されているため、VPC 内に存在する OpenSearch Service ドメインに IP ベースのアクセス権限ポリシーを適用することはできません。パブリックエンドポイントを使用する場合、IP ベースのポリシーを引き続き利用できます。

開始する前に: VPC アクセスの前提条件

VPC と新規の OpenSearch Service ドメイン間の接続を有効にするためには、次を実行する必要があります。

- 「VPC を作成する」

  VPC を作成するには、Amazon VPC AWS コンソール、 CLI、またはいずれかの AWS SDKs を使用できます。詳細については、Amazon VPC ユーザーガイドの「VPC の使用」を参照してください。VPC が既にある場合、このステップは省略できます。

- IP アドレスのリザーブ

  OpenSearch Service は、ネットワークインターフェースを VPC のサブネットに配置することで、VPC をドメインに接続できるようにします。各ネットワークインターフェースは 1 つの IP アドレスに関連付けられます。ネットワークインターフェースのためにサブネットで十分な数の IP アドレスをリザーブする必要があります。詳細については、「VPC サブネットで IP アドレスをリザーブする」を参照してください。

VPC ドメインをテストする

VPC のセキュリティを強化することで、ドメインへの接続や、基本的なテスト実行を行うことができます。OpenSearch Service VPC ドメインがすでにあり、VPC サーバーを作成しない場合は、次のプロセスをお試しください。

1. ドメインのアクセスポリシーに [きめ細かなアクセスコントロールのみを使用] を選択します。この設定は、テスト完了後にいつでも更新できます。
2. OpenSearch Service ドメインと同じ VPC、サブネット、およびセキュリティグループに Amazon Linux Amazon EC2 インスタンスを作成します。

   このインスタンスはテストを目的としており、必要な作業はわずかであるため、安価なインスタンスタイプ (例: `t2.micro`) を選択します。インスタンスにパブリック IP アドレスを割り当てたら、新しいキーペアを作成するか、既存のキーペアを選択します。新しいキーを作成する場合は、`~/.ssh` ディレクトリにダウンロードします。

   インスタンス作成の詳細については、「Amazon EC2 Linux インスタンスの開始方法」を参照してください。
3. インターネットゲートウェイを VPC に追加します。
4. VPC のルートテーブルで、新しいルートを追加します。[送信先] で、コンピュータのパブリック IP アドレスを含む CIDR ブロックを指定します。[ターゲット] で、先ほど作成したインターネットゲートウェイを指定します。

   たとえば、コンピュータが 1 つのみの場合は `123.123.123.123/32`、複数の場合には `123.123.123.0/24` を指定します。
5. セキュリティグループで、2 つのインバウンドルールを指定します。

タイプ	プロトコル	ポート範囲	ソース
SSH (22)	TCP (6)	22	*your-cidr-block*
HTTPS (443)	TCP (6)	443	*your-security-group-id*

   最初のルールでは、EC2 インスタンスに SSH 接続できます。2 番目のルールでは、EC2 インスタンスから OpenSearch Service ドメインに HTTPS 経由で通信できるようになります。
6. ターミナルから、次のコマンドを実行します。

```
ssh -i ~/.ssh/your-key.pem ec2-user@your-ec2-instance-public-ip -N -L
 9200:vpc-domain-name-identifier.region.es.amazonaws.com:443
```

このコマンドでは、https://localhost:9200 へのリクエストを EC2 インスタンスから OpenSearch Service ドメインに転送する SSH トンネルが作成されます。コマンドでポート 9200 を指定すると、OpenSearch のローカルインストールがシミュレートされますが、任意のポートを使用します。OpenSearch サービスでは、ポート 80 (HTTP) または 443 (HTTPS) 経由でのみ接続を受け入れます。

このコマンドではフィードバックは返らず、無限に実行されます。停止するには、`Ctrl + C` を押します。

7. ウェブブラウザで https://localhost:9200/_dashboards/ に移動します。セキュリティ例外の承認が必要な場合があります。

   https://localhost:9200curl、Postman、またはお気に入りのプログラミング言語を使用して、 にリクエストを送信することもできます。

   > **Tip**
   >
   > 証明書の不一致が原因で curl のエラーが発生した場合は、`--insecure` フラグを試してください。

VPC サブネットで IP アドレスをリザーブする

OpenSearch Service は、ネットワークインターフェイスを VPC の 1 つのサブネット (複数のアベイラビリティーゾーンを有効にした場合には VPC の複数のサブネット) に配置することでドメインを VPC に接続します。各ネットワークインターフェースは 1 つの IP アドレスに関連付けられます。OpenSearch Service ドメインを作成する前に、ネットワークインターフェースを対応するために十分な数の IP アドレスを各サブネットで有効にしておく必要があります。

基本的な計算として、各サブネットで OpenSearch Service が予約する IP アドレスの数は、データノードの数に 3 をかけた数をアベイラビリティーゾーンの数で割った数になります。

例

- 3 つのアベイラビリティーゾーンで 1 つのドメインに 9 つのデータノードがある場合、サブネットあたりの IP アドレスの数は、9 x 3 ÷ 3 = 9 になります。
- 2 つのアベイラビリティーゾーンで 1 つのドメインに 8 つのデータノードがある場合、サブネットあたりの IP アドレスの数は、8 x 3 ÷ 2 = 12 になります。

- 1 つのアベイラビリティーゾーンで 6 つのドメインに 1 つのデータノードがある場合、サブネットあたりの IP アドレスの数は、6 x 3 ÷ 1 = 18 になります。

ドメインを作成する場合、OpenSearch Service は IP アドレスを予約し、一部をドメインに使用して、残りは Blue/Green デプロイ用に予約します。Amazon EC2 コンソール の [ネットワークインターフェース] セクションからネットワークインターフェースと関連する IP アドレスを表示できます。[説明] 列は、ネットワークインターフェイスのどの OpenSearch Service ドメインが関連付けられているかを示しています。

> **Tip**
>
> OpenSearch Service リザーブド IP アドレス用の専用サブネットを作成することが推奨されます。専用サブネットを使用することで、他のアプリケーションやサービスとの重複を回避でき、将来的にクラスターをスケールする必要が生じた場合に追加の IP アドレスをリザーブできることを確保します。詳細については、「VPC でサブネットを作成する」を参照してください。

VPC ドメインに必要なプライベート IP アドレス予約の数を減らすために、専用コーディネーターノードのプロビジョニングを検討することもできます。OpenSearch は、データノードではなく、専用コーディネーターノードに Elastic Network Interface (ENI) をアタッチします。専用コーディネーターノードは通常、データノード全体の約 10% を表します。その結果、VPC ドメイン用に予約されるプライベート IP アドレスの数が少なくなります。

VPC アクセス用のサービスにリンクされたロール

サービスにリンクされたロールは、サービスに権限を委任する一意のタイプの IAM ロールであり、ユーザーに代わってリソースを作成して管理できます。OpenSearch Service には、VPC にアクセスし、ドメインのエンドポイントを作成し、VPC のサブネットにネットワークインターフェースを配置するために、サービスにリンクされたロールが必要です。

OpenSearch サービスコンソールを使用して VPC 内にドメインを作成する場合、OpenSearch Service は自動的にこのロールを作成します。この自動作成を成功させるには、`iam:CreateServiceLinkedRole` アクションへのアクセス許可が必要です。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールの許可」を参照してください。

OpenSearch Service がロールを作成すると、IAM コンソールを使用してこのロールを表示することができます (`AWSServiceRoleForAmazonOpenSearchService`)。

このロールのアクセス権限およびその削除方法の詳細については、「the section called “サービスにリンクされたロールの使用”」を参照してください。

送信トラフィックオプション

デフォルトでは、VPC ドメインは VPC からの進入を受け入れます。ドメインで設定したカスタムエンドポイントへのトラフィックなど、ドメインからのエグレストラフィックは、パブリックインターネットを経由します。ワークロードで、代わりに独自の VPC を介して送信トラフィックフローが必要な場合は、ドメインで Egress オプションを有効にします。詳細については、「the section called “VPC を介したドメイン出力トラフィックのルーティング”」を参照してください。

VPC を介したドメイン出力トラフィックのルーティング

Amazon OpenSearch Service VPC ドメインからの出力トラフィックを、パブリックインターネットではなく独自の VPC 経由でルーティングする方法について説明します。

> **Note**
>
> このオプションは、ドメインからの出力トラフィックにのみ影響します。ドメインへの進入は、ポート 80 と 443 でも同じように機能します。

概要

デフォルトでは、VPC ドメインからカスタムエンドポイントへの出力はパブリックインターネット経由で送信されます。

VPC を介して Egress を有効にすると、ドメインからの Egress トラフィックが VPC に入り、ルートテーブル、セキュリティグループ、ネットワーク ACLs の対象となります。このオプションは、VPC を介してドメインからの出力を制御する必要がある場合や、ドメインから VPC エンドポイントなどのプライベートエンドポイントに到達する必要がある場合に使用します。

仕組み

VPC ドメインで Egress を有効にすると、OpenSearch Service はドメインに指定する各サブネットに追加の Elastic Network Interface (ENI) を配置します。ドメインからのエグレストラフィックは、これらのエグレス ENIs。

出力 ENIsはリクエスタマネージド型です。OpenSearch Service は、ユーザーに代わって作成、設定、削除します。アカウントから変更することはできません。

VPC 内のコンポーネント

出力が有効になっている場合、VPC には 2 つのリソースタイプが含まれます。

- ドメイン ENIs。ドメインへのインバウンドトラフィック用に OpenSearch Service によって作成および管理されます。これらは、出力が有効になっているかどうかにかかわらず、任意の VPC ドメインに存在します。
- Egress ENIs サービスにリンクされたロールを介して OpenSearch Service によって作成され、OpenSearch Service ネットワークプレーンによって管理されます。これらは、ドメインからの Egress トラフィックを VPC に伝送します。

マルチ AZ ドメインでは、エグレス ENIs はアベイラビリティーゾーンごとにプロビジョニングされ、ドメインに選択したサブネットと完全に一致します。

出力の DNS 解決

VPC 経由の出力が有効になっている場合、ドメインはデフォルトの VPC リゾルバー (VPC CIDR の「+2」アドレス) を介してホスト名を解決します。カスタム DNS リゾルバーは起動時にサポートされていません。

ドメインは VPC リゾルバーを使用するため、以下を解決できます。

- VPC に関連付けられた Amazon Route 53 プライベートホストゾーンのレコード。
- VPC 内の VPC エンドポイントのプライベート DNS 名。

> **Important**
>
> VPC DNS にアクセスできないか、設定が間違っている場合、ドメインの出力統合は失敗します。「the section called “トラブルシューティング”」を参照してください。

前提条件

VPC 経由で Egress を有効にする前に、VPC が次の要件を満たしていることを確認してください。

- DNS 解決と DNS ホスト名はどちらも VPC で有効になっています。
- デフォルトの VPC リゾルバー (VPC CIDR の「+2」アドレス) は、ドメインに使用するサブネットからアクセスできます。

サブネット IP 容量。ドメイン ENIs「」を参照the section called “VPC サブネットで IP アドレスをリザーブする”) と、送信 ENIs。

サービスにリンクされたロール。既存の Amazon OpenSearch Service サービスにリンクされたロールは、出力 ENIs を作成および管理するために必要なアクセス許可を取得します。すでに VPC ドメインを使用している場合は、ロールを再作成する必要はありません。詳細についてはthe section called “サービスにリンクされたロールの使用”を参照してください。

リージョンの可用性。VPC 経由のエグレスは、Amazon OpenSearch Service エンドポイントとクォータページに記載されているリージョンで利用できます。

ドメインで Egress を有効にする

VPC ドメインの作成時、または既存の VPC ドメインを更新することで、VPC ドメインで Egress を有効にできます。パブリックエンドポイントドメインで出力を有効にすることはできません。このオプションを有効または無効にすると、ブルー/グリーンデプロイがトリガーされます。

コンソール

1. Amazon OpenSearch Service コンソールを開きます。
2. 新しいドメインの作成を開始するか、既存の VPC ドメインを選択して編集を選択します。
3. ネットワークで、VPC アクセスを選択し、現在と同じように VPC、サブネット、セキュリティグループを選択します。
4. VPC Egress で、Enable Egress を選択します。
5. 残りのステップを完了し、変更を送信します。

AWS CLI

出力を有効にしてドメインを作成するには、 `EgressEnabled` に を含めます`--vpc-options`。

```
aws opensearch create-domain \
  --domain-name example-domain \
  --engine-version OpenSearch_2.15 \
  --cluster-config
 InstanceType=r6g.large.search,InstanceCount=3,ZoneAwarenessEnabled=true \
  --vpc-options '{
      "SubnetIds": ["subnet-EXAMPLEAZ1", "subnet-EXAMPLEAZ2", "subnet-EXAMPLEAZ3"],
      "SecurityGroupIds": ["sg-EXAMPLE"],
      "EgressEnabled": true
  }'
```

既存の VPC ドメインで出力を切り替えるには、 を使用しますupdate-domain-config。

```
aws opensearch update-domain-config \
  --domain-name example-domain \
  --vpc-options '{
      "SubnetIds": ["subnet-EXAMPLEAZ1", "subnet-EXAMPLEAZ2", "subnet-EXAMPLEAZ3"],
      "SecurityGroupIds": ["sg-EXAMPLE"],
      "EgressEnabled": true
  }'
```

API

VPC 経由で出力を有効にするには、 CreateDomain または VPCOptionsの true で EgressEnabled を に設定しますUpdateDomainConfig。値は、 DescribeDomainと VPCOptionsの で返されますDescribeDomainConfig。

```
{
  "SubnetIds": ["subnet-EXAMPLEAZ1", "subnet-EXAMPLEAZ2", "subnet-EXAMPLEAZ3"],
  "SecurityGroupIds": ["sg-EXAMPLE"],
  "EgressEnabled": true
}
```

完全なスキーマについては、Amazon OpenSearch Service API リファレンスのVPCOptions」を参照してください。

更新または無効化

Egress は、ドメインの作成時または作成後にいつでもオンにでき、有効になっているドメインではオフにできます。エグレスを有効にしたまま、アベイラビリティーゾーンを追加または削除することもできます。への変更は、他の VPC 設定の変更と同様に、ブルー/グリーンデプロイをEgressEnabledトリガーします。詳細については、「the section called “設定変更”」を参照してください。

出力を無効にすると、OpenSearch Service は VPC から出力 ENIs と関連するサービスマネージドリソースを削除します。ドメインを削除すると、すべての出力リソースが自動的にクリーンアップされます。

検証とモニタリング

Egress を有効にしたら、Amazon EC2 コンソールで表示して、Egress ENIs が選択したサブネットに存在することを確認します。これらの説明は OpenSearch Service ドメインを識別しま

す。ドメインから送信トラフィックを監視するには、送信 ENIs で VPC フローログを有効にします。OpenSearch Service コンソールでドメインの状態を確認し、エグレス統合 (アラートの送信先、機械学習コネクタ、スナップショットリポジトリ) からの既存の成功および失敗シグナルに基づいて、統合レベルのステータスを確認します。

トラブルシューティング

Egress を有効にした後、Egress 統合が機能しなくなりました。VPC ルートテーブルが送信 ENIs から送信先へのトラフィックを許可し、VPC リゾルバーに到達可能であり、送信先ホスト名を解決できることを確認します。

ホスト名の解決が失敗します。VPC で DNS 解決と DNS ホスト名が有効になっていることを確認します。Route 53 プライベートホストゾーンまたは Route 53 Resolver アウトバウンドエンドポイントを使用する場合は、関連するルールが送信先をカバーしていることを確認します。

サブネットに十分な IP アドレスがありません。サブネットを展開するか、ドメイン専用のサブネットを使用します。「the section called “VPC サブネットで IP アドレスをリザーブする”」を参照してください。

サービスにリンクされたロールにアクセス許可がありません。サービスにリンクされたロールを再作成するか、更新されたポリシーをアタッチします。「the section called “サービスにリンクされたロールの使用”」を参照してください。

パブリックエンドポイントドメインで出力を有効にすることはできません。VPC 経由の Egress は、VPC ドメインでのみ使用できます。最初にドメインを変換します。「the section called “パブリックアクセスから VPC アクセスに移行する”」を参照してください。

> **Warning**
>
> 出力 ENIsはサービスマネージド型です。手動でデタッチまたは削除しないでください。それらを削除するには、ドメインの Egress オプションを無効にするか、ドメインを削除します。

制限と考慮事項

VPC 経由のエグレスは、Amazon OpenSearch Service エンドポイントとクォータページに記載されているリージョンで利用できます。VPC が有効になっている Amazon OpenSearch Service ドメインでサポートされています。

使用状況と請求

VPC を介した出力に関連するデータ転送をモニタリングするには、AWS 請求ダッシュボードで使用タイプ DataTransfer-Regional-Bytes、オペレーション VPCConnectionUsage、製品コードを確認しますAmazonES。現在の料金については、「Amazon OpenSearch Service の料金」を参照してください。

Amazon OpenSearch Service でのインデックススナップショットの作成

Amazon OpenSearch Service でのスナップショットはクラスターのインデックスと状態のバックアップです。状態には、クラスター設定、ノード情報、インデックス設定、シャードの割り当てなどが含まれます。

OpenSearch Service のスナップショットには、次のような形式が用意されています。

- 自動スナップショットは、クラスターの復元専用です。クラスターのステータスが赤くなった場合や、データが失われた場合に、ドメインを復元するために使用できます。詳細については、「スナップショットの復元」を参照してください。OpenSearch Service では、事前設定された Amazon S3 バケットに自動スナップショットを追加料金なしで保存します。
- 手動スナップショットは、クラスターの復元、またはクラスター間でのデータの移行に使用します。手動スナップショットを開始する必要があります。これらのスナップショットは独自の Amazon S3 バケットに保存され、標準の S3 料金が適用されます。セルフマネージド OpenSearch クラスターのスナップショットがある場合、そのスナップショットを使用して OpenSearch Service ドメインに移行できます。詳細については、「Amazon OpenSearch Service への移行」を参照してください。

すべての OpenSearch Service ドメインで自動スナップショットを取得しますが、頻度は以下の点で異なります。

- OpenSearch または Elasticsearch 5.3 以降を実行しているドメインでは、OpenSearch Service は 1 時間ごとに自動的にスナップショットを取得し、最大 336 個のスナップショットを 14 日間保持します。時間単位のスナップショットは増分的な性質があるため、中断が少なくなります。また、ドメインの問題が発生した場合に、より最近のリカバリポイントを提供します。
- Elasticsearch 5.1 以前を実行しているドメインでは、OpenSearch Service は 1 日ごとに指定する時刻に自動でスナップショットを取得し、14 個のスナップショットを 保持しますが、30 日間を超えてスナップショットデータを保持することはありません。

クラスターのステータスが赤になると、クラスターのステータスが維持されている間、すべての自動スナップショットが失敗します。2 週間以内にこの問題を解決しない場合、クラスターのデータは永遠に失われます。トラブルシューティングステップについては、「the section called “赤のクラスター状態”」を参照してください。

前提条件

スナップショットを手動で作成するには、IAM および Amazon S3 を使用して作業する必要があります。スナップショットの取得を試す前に、次の前提条件を満たしていることを確認します。

前提条件	説明
S3 バケット	OpenSearch Service ドメイン用の手動スナップショットを保存するための S3 バケットを作成します。手順については、「Amazon Simple Storage Service ユーザーガイド」の「汎用バケットの作成」を参照してください。 次の場所で使用するには、バケットの名前を覚えておいてください。 • IAM ロールにアタッチされた IAM ポリシーの `Resource` ステートメント • スナップショットリポジトリの登録に使用する Python クライアント (この方法を使用する場合) ⚠ Important Amazon Glacier ライフサイクルルールをこのバケットに適用しないでください。手動スナップショットは、Amazon Glacier ストレージクラスをサポートしていません。
IAM ロール	OpenSearch Service に許可を委任する IAM ロールを作成します。手順については、IAM ユーザーガイドの「IAM ロールの作成 (コンソール)」を参照してください。この章の残りの部分では、このロールを `TheSnapshotRole` と呼びます。 IAM ポリシーのアタッチ 次のポリシーを `TheSnapshotRole` にアタッチして、S3 バケットへのアクセスを許可します。

前提条件	説明

```
{
  "Version":"2012-10-17",
  "Statement": [{
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3::: amzn-s3-demo-bucket  "
      ]
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3::: amzn-s3-demo-bucket  /*"
      ]
    }
  ]
}
```

ポリシーをロールにアタッチする方法については、「IAM ユーザーガイド」の「IAM ID アクセス許可の追加 (コンソール)」を参照してください。

信頼関係を編集する

TheSnapshotRole の信頼関係を編集して、次の例に示すように、Principal ステートメントで OpenSearch Service を指定します。

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Sid": "",
    "Effect": "Allow",
```

前提条件	説明

	"Principal": {
	"Service": "es.amazonaws.com"
	},
	"Action": "sts:AssumeRole"
	}]
	
	}
	
	信頼関係を編集する手順については、「IAM ユーザーガイド」の「ロール信頼ポリシーを更新する」を参照してください。

前提条件	説明
アクセス許可	スナップショットリポジトリを登録するには、TheSnapshotRole を OpenSearch Service に渡すことができる必要があります。さらに、es:ESHttpPut アクションへのアクセスも必要です。これらの両方の許可を付与するには、リクエストの署名に認証情報が使用されている IAM ロールまたはユーザーに次のポリシーをアタッチします。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::  123456789012 :role/TheSnapsh
otRole "
    },
    {
      "Effect": "Allow",
      "Action": "es:ESHttpPut",
      "Resource": "arn:aws:es:  us-east-1 :123456789
012 :domain/domain-name /*"
    }
  ]
}
```

ユーザーまたはロールが TheSnapshotRole を渡すための iam:PassRole 許可を持っていない場合、次のステップでリポジトリを登録しようとすると、次の一般的なエラーが発生することがあります。

```
$ python register-repo.py
{"Message":"User: arn:aws:iam::   123456789012 :user/MyUserAccount
is not authorized to perform: iam:PassRole on resource:
arn:aws:iam:: 123456789012 :role/TheSnapshotRole "}
```

手動スナップショットレポジトリの登録

手動インデックススナップショットを取得する前に、OpenSearch Service を用いてスナップショットレポジトリを登録する必要があります。この 1 回限りのオペレーションでは、「」で説明されているようにTheSnapshotRole、 へのアクセスが許可されている認証情報を使用して AWS リクエストに署名する必要がありますthe section called “前提条件”。

ステップ 1: OpenSearch Dashboards でスナップショットロールをマッピングする (きめ細かなアクセスコントロールを使用している場合)

きめ細かなアクセスコントロールにより、リポジトリの登録時に追加のステップが導入されます。HTTP 基本認証を他のすべての目的で使用する場合でも、TheSnapshotRole を渡すための `iam:PassRole` 許可を持っている IAM ロールまたはユーザーに `manage_snapshots` ロールをマップする必要があります。

1. OpenSearch Service ドメインの OpenSearch Dashboards プラグインに移動します。OpenSearch Service コンソールのドメインダッシュボードに Dashboards エンドポイントがあります。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[manage_snapshots] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. TheSnapshotRole を渡すための許可を持っているロールの ARN を追加します。[Backend roles] (バックエンドロール) の下にロール ARN を配置します

   ```
   arn:aws:iam::123456789123:role/role-name
   ```

5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

ステップ 2: リポジトリを登録する

次の [Snapshots] (スナップショット) のタブには、スナップショットディレクトリの登録方法が記されています。手動スナップショットの暗号化に関するオプションと、新しいドメインに移行した後にスナップショットを登録する際のオプションについては、関連するタブを参照してください。

Snapshots

スナップショットリポジトリを登録するには、OpenSearch Service ドメインエンドポイントに PUT リクエストを送信します。curl、サンプル Python クライアント、Postman、またはその他の方法を使用して署名付きリクエストを送信し、スナップショットリポジトリを登録できます。OpenSearch Dashboards コンソールでは、PUT リクエストを使用してリポジトリを登録することはできませんのでご注意ください。

リクエストは以下のような形式です。

```
PUT domain-endpoint/_snapshot/my-snapshot-repo-name
{
  "type": "s3",
  "settings": {
    "bucket": "amzn-s3-demo-bucket",
    "base_path": "my/snapshot/directory",
    "region": "region",
    "role_arn": "arn:aws:iam::123456789012:role/TheSnapshotRole"
  }
}
```

> **Note**
>
> リポジトリ名は「cs-」で始めることはできません。さらに、複数のドメインから同じリポジトリに書き込まない必要があります。リポジトリへの書き込みアクセス権を持つドメインは、1 つだけにする必要があります。

ドメインが Virtual Private Cloud (VPC) に存在する場合は、リクエストが正常にスナップショットレポジトリに登録するようにお使いのコンピュータが VPC に接続されていることが必要です。VPC へのアクセスはネットワーク構成によって異なりますが、VPN あるいは社内ネットワークへの接続を含む場合がよくあります。OpenSearch Service ドメインにアクセスできるかを確認するには、ウェブブラウザで `https://your-vpc-domain.region.es.amazonaws.com` を開き、デフォルトの JSON 応答を受信していることを確認します。

Amazon S3 バケットが OpenSearch ドメイン AWS リージョン 以外の にある場合は、リクエスト`"endpoint": "s3.amazonaws.com"`に パラメータを追加します。

Encrypted snapshots

現在、 AWS Key Management Service (KMS) キーを使用して手動スナップショットを暗号化することはできませんが、サーバー側の暗号化 (SSE) を使用して保護できます。

スナップショットリポジトリとして使用しているバケットの S3 管理のキーを使って SSE を有効にするには、PUT リクエストの `"settings"` ブロックに `"server_side_encryption": true` を追加します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) の使用」を参照してください。

または、スナップショットリポジトリとして使用する S3 バケットのサーバー側の暗号化に AWS KMS キーを使用することもできます。この方法を使用する場合は、S3 バケットの暗号化に使用される AWS KMS キーにアクセスTheSnapshotRole許可を付与してください。詳細については、「AWS KMSのキーポリシー」を参照してください。

Domain migration

スナップショットリポジトリの登録は 1 回限りのオペレーションです。ただし、1 つのドメインから別のドメインに移行するには、古いドメインと新しいドメインで同じスナップショットレポジトリを登録する必要があります。リポジトリ名は任意です。

新しいドメインに移行する場合、または同じリポジトリを複数のドメインに登録する場合は、次のガイドラインを考慮してください。

- 新しいドメインにリポジトリを登録する場合は、`"readonly": true` を PUT リクエストの `"settings"` ブロックに追加します。この設定により、古いドメインのデータが誤って上書きされるのを防ぐことができます。リポジトリへの書き込みアクセス権を持つドメインは、1 つだけにする必要があります。
- データを別の AWS リージョンへ移行する場合 (us-east-2 にある古いドメインとバケットから us-west-2 にある新しいドメインへ等) は、`"region": "region"` を PUT ステートメントの `"endpoint": "s3.amazonaws.com"` に置き換えて、リクエストを再送信します。

サンプル Python クライアントの使用

Python クライアントは、シンプルな HTTP リクエストよりも自動化が容易で、再利用性が向上します。この方法を使用してスナップショットリポジトリを登録する場合は、次のサンプル Python コードを `register-repo.py` などの Python ファイルとして保存します。クライアントでは、AWS SDK for Python (Boto3)、リクエストおよび requests-aws4auth パッケージが必要になります。クライアントには、他のスナップショットオペレーションのコメントアウトされた例が含まれています。

サンプルコードで、次の変数を更新します: host、region、path、および payload。

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

host = '' # domain endpoint
region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

# Register repository

path = '/_snapshot/my-snapshot-repo-name' # the OpenSearch API endpoint
url = host + path

payload = {
  "type": "s3",
  "settings": {
    "bucket": "amzn-s3-demo-bucket",
    "base_path": "my/snapshot/directory",
    "region": "us-west-1",
    "role_arn": "arn:aws:iam::123456789012:role/snapshot-role"
  }
}

headers = {"Content-Type": "application/json"}

r = requests.put(url, auth=awsauth, json=payload, headers=headers)

print(r.status_code)
print(r.text)

# # Take snapshot
#
# path = '/_snapshot/my-snapshot-repo-name/my-snapshot'
# url = host + path
#
# r = requests.put(url, auth=awsauth)
#
# print(r.text)
#
```

```
# # Delete index
#
# path = 'my-index'
# url = host + path
#
# r = requests.delete(url, auth=awsauth)
#
# print(r.text)
#
# # Restore snapshot (all indexes except Dashboards and fine-grained access control)
#
# path = '/_snapshot/my-snapshot-repo-name/my-snapshot/_restore'
# url = host + path
#
# payload = {
#   "indices": "-.kibana*,-.opendistro_security,-.opendistro-*",
#   "include_global_state": False
# }
#
# headers = {"Content-Type": "application/json"}
#
# r = requests.post(url, auth=awsauth, json=payload, headers=headers)
#
# print(r.text)
#
# # Restore snapshot (one index)
#
# path = '/_snapshot/my-snapshot-repo-name/my-snapshot/_restore'
# url = host + path
#
# payload = {"indices": "my-index"}
#
# headers = {"Content-Type": "application/json"}
#
# r = requests.post(url, auth=awsauth, json=payload, headers=headers)
#
# print(r.text)
```

手動スナップショットの作成

スナップショットは瞬時に取得されません。完了するまでには時間がかかり、クラスターの完全なポイントインタイムビューを表すわけではありません。スナップショットが進行中の間も、ドキュメントのインデックス作成や、クラスターへの他のリクエストを行うことはできますが、新しいドキュメ

ントおよび既存のドキュメントの更新は一般的にスナップショットに含まれません。スナップショットには、OpenSearch によってスナップショットが開始された際に存在したプライマリシャードが含まれます。スナップショットのスレッドプールのサイズによっては、わずかな時間の違いで、スナップショットにさまざまなシャードが含まれることがあります。スナップショットのベストプラクティスについては、「the section called “スナップショットパフォーマンスの向上”」を参照してください。

> **Warning**
>
> 手動スナップショットには、UltraWarm またはコールドストレージ階層に保存されているデータは含まれません。ドメインが UltraWarm またはコールドストレージを使用している場合は、そのデータを保持する必要がある場合は、手動スナップショットを作成する前に、それらのインデックスをホットストレージに移行します。

スナップショットのストレージとパフォーマンス

OpenSearch スナップショットは増分です。つまり、最後にスナップショットが取得されてから変更されたデータのみ保存されます。増分のみ保存されるため、スナップショットの取得頻度が高い場合でも低い場合でも、ディスク使用量を最小限に抑えることができます。つまり、スナップショットを 1 時間ごとに 1 週間 (合計 168 個) 取得すると、週末に 1 つのスナップショットを取得する場合よりも、使用するディスク容量は少なくなる場合があります。また、スナップショットの取得頻度が高くなるほど、完了までにかかる時間は短くなります。例えば、日次スナップショットの完了には 20 ~ 30 分かかる場合がありますが、時間単位のスナップショットは数分以内に完了することもあります。中には、30 分ごとにスナップショットを取得している OpenSearch ユーザーもいます。

スナップショットを取得する

スナップショットを作成するときは、以下の情報を指定します。

- スナップショットリポジトリの名前
- スナップショットの名前

この章の例では、わかりやすく簡潔にするために、一般的な HTTP クライアントである curl を使用します。curl リクエストにユーザー名とパスワードを渡す方法については、「開始方法のチュートリアル」を参照してください。

アクセスポリシーでユーザーまたはロールを指定する場合は、スナップショットリクエストに署名する必要があります。curl の場合、--aws-sigv4 のオプション (バージョン 7.75.0 以降) を使用できます。また、サンプル Python クライアントのコメントアウトされた例を使って、curl コマンドが使用しているのと同じエンドポイントに、署名された HTTP リクエストを行うこともできます

手動スナップショットを作成するには、次のステップを実施します。

1. 現在進行中のスナップショットは取得できません。確認するには、以下のコマンドを実行します。

   ```
   curl -XGET 'domain-endpoint/_snapshot/_status'
   ```

2. 次のコマンドを実行して、手動でスナップショットを取得します。

   ```
   curl -XPUT 'domain-endpoint/_snapshot/repository-name/snapshot-name'
   ```

   特定のインデックスを包含または除外したり、他の設定を指定したりするには、リクエスト本文を追加します。リクエストの構造については、OpenSearch ドキュメントの「Take snapshots」(スナップショットを取得する) を参照してください。

> **Note**
>
> OpenSearch Service ドメインの容量が大きくなると、スナップショットの取得に必要な時間は長くなります。長時間実行しているスナップショット操作の場合は、504 GATEWAY_TIMEOUT エラーが発生する場合があります。通常、このエラーは無視して、オペレーションが正常に完了するのを待ってかまいません。次のコマンドを実行して、ドメインのすべてのスナップショットの状態を確認します。
>
> ```
> curl -XGET 'domain-endpoint/_snapshot/repository-name/_all?pretty'
> ```

スナップショットからのデータの復元

スナップショットを復元する前に、スタンバイ付きマルチ AZ を使用してドメインに復元するには、復元されたインデックスにスタンバイ設定と互換性のあるレプリカ設定が必要であることに注意してください。スナップショットに互換性のないレプリカ数を持つインデックスが含まれている場合、復元中に設定を上書きしない限り、復元は失敗します。失敗を回避するには、復元されたインデックス

のレプリカ数がドメインのスタンバイ設定と互換性があることを確認するか、 index_settingsパラメータを使用して復元オペレーション中にレプリカ設定を上書きします。

> **Warning**
>
> インデックスのエイリアスを使用する場合は、インデックスを削除する前に、エイリアスへの書き込みリクエストを中止するか、エイリアスを別のインデックスに切り替える必要があります。書き込みリクエストを中止すると、次のシナリオの回避に有効です。
>
> 1. インデックスを削除すると、そのエイリアスも削除される。
> 2. 削除したばかりのエイリアスに対する障害のある書き込みリクエストにより、エイリアスと同じ名前で新しいインデックスが作成される。
> 3. 新しいインデックスとの命名の競合により、エイリアスを使用できなくなる。エイリアスを別のインデックスに切り替えた場合は、スナップショットから復元するときに `"include_aliases": false` を指定します。

スナップショットを復元するには

1. 復元するスナップショットを特定します。カスタムアナライザーパッケージや割り当て要件設定など、このインデックスのすべての設定がドメインと互換性があることを確認してください。すべてのスナップショットレポジトリを表示するには、次のコマンドを実行します。

   ```
   curl -XGET 'domain-endpoint/_snapshot?pretty'
   ```

   リポジトリを識別した後、次のコマンドを実行してすべてのスナップショットを表示します。

   ```
   curl -XGET 'domain-endpoint/_snapshot/repository-name/_all?pretty'
   ```

   > **Note**
   >
   > ほとんどの自動スナップショットは、cs-automated リポジトリに保存されます。ドメインで暗号化された保管中のデータは cs-automated-enc リポジトリに保存されます。検索する手動スナップショットレポジトリが表示されない場合は、ドメインにその登録をしたことを確認します。

2. (オプション) クラスター上のインデックスとスナップショット内のインデックスの間に名前の競合がある場合は、OpenSearch Service ドメイン内の 1 つ以上のインデックスを削除するか、名前を変更します。OpenSearch クラスターに同じ名前のインデックスが既にある場合は、インデックスのスナップショットを復元できません。

   インデックスの付けた名前の競合がある場合は、次のオプションがあります。

   - 既存の OpenSearch Service ドメインにあるインデックスを削除して、スナップショットを復元します。
   - スナップショットから復元する際、インデックスの名前を変更し、その後インデックスを再作成します。インデックスの名前を変更する方法については、OpenSearch ドキュメントの「this example request」を参照してください。
   - 別の OpenSearch Service ドメインにスナップショットを復元します (手動スナップショットでのみ可能)。

   次のコマンドは、ドメイン内の既存のインデックスをすべて削除します。

   ```
   curl -XDELETE 'domain-endpoint/_all'
   ```

   ただし、すべてのインデックスを復元する予定がない場合は、インデックスを 1 つだけ削除できます。

   ```
   curl -XDELETE 'domain-endpoint/index-name'
   ```

3. スナップショットを復元するには、次のコマンドを実行します。

   ```
   curl -XPOST 'domain-endpoint/_snapshot/repository-name/snapshot-name/_restore'
   ```

   OpenSearch Dashboards ときめ細かなアクセスコントロールインデックスに対する特別な許可により、特に自動スナップショットから復元しようとする場合は、すべてのインデックスを復元しようとすると失敗することがあります。次の例では、1 つのインデックスである `my-index` を `2020-snapshot` から `cs-automated` スナップショットレポジトリで復元します。

   ```
   curl -XPOST 'domain-endpoint/_snapshot/cs-automated/2020-snapshot/_restore' \
   -d '{"indices": "my-index"}' \
   -H 'Content-Type: application/json'
   ```

または、Dashboards ときめ細かなアクセスコントロールインデックスを除くすべてのインデックスを復元することもできます。

```
curl -XPOST 'domain-endpoint/_snapshot/cs-automated/2020-snapshot/_restore' \
-d '{"indices": "-.kibana*,-.opendistro*"}' \
-H 'Content-Type: application/json'
```

> **Note**
>
> OpenSearch のバージョンによっては、 `-.opensearch-observability*`や など、追加のシステムインデックスを除外する必要もあります`-.plugins-ml-config*`。これらのインデックスがターゲットドメインに既に存在する場合、復元オペレーションが失敗することがあります。除外するには、`indices`除外リストに追加します。例: `"indices": "-.kibana*,-.opendistro*,-.opensearch-observability*,-.plugins-ml-config*"`。

`rename_pattern` と `rename_replacement` のパラメータを使用すると、データを削除することなくスナップショットを復元することができます。これらのパラメータの詳細については、OpenSearch ドキュメントの「Restore Snapshot API」(スナップショット API の復元) 内、「Request fields」(リクエストフィールド) および「Example request」(リクエスト例) を参照してください。

> **Note**
>
> 関連するインデックスに対してすべてのプライマリシャードを使用できなかった場合、スナップショットが `PARTIAL` の `state` になっている可能性があります。この値は、1 つ以上のシャードからのデータが正しく保存されていないことを示します。部分スナップショットからの復元も可能ですが、不足しているインデックスの復元に古いスナップショットの使用が必要になる場合もあります。

手動スナップショットの削除

次のコマンドを実行して、手動スナップショットを削除します。

```
DELETE _snapshot/repository-name/snapshot-name
```

Snapshot Management を用いたスナップショットの自動化

OpenSearch Dashboards で Snapshot Management (SM) ポリシーを設定すると、スナップショットの定期的な作成と削除を自動化できます。SM は、インデックスのグループのスナップショットを作成できますが、Index State Management は、インデックスごとに 1 つのスナップショットしか作成できません。OpenSearch Service で SM を使用するときは、ユーザー独自の Amazon S3 リポジトリを登録する必要があります。リポジトリの登録手順については、「手動スナップショットレポジトリの登録」を参照してください。

> **Note**
>
> スナップショット管理は、OpenSearch バージョン 2.5 以降でサポートされています。以前のバージョンを実行しているドメインは、この機能をサポートしていません。

SM が登場する以前は、OpenSearch Service は自動スナップショット機能を無料で提供していました。こちらは、現在もデフォルトで有効になっています。この機能は、サービスが管理する `cs-*` リポジトリにスナップショットを送信します。この機能を無効にする場合は、 サポートまでお問い合わせください。

SM 機能の詳細については、OpenSearch ドキュメントの「Snapshot Management 」(スナップショットの管理) を参照してください。

SM は、現在は、複数のインデックスタイプでのスナップショット作成をサポートしていません。例えば、* を使って複数のインデックスでスナップショットを作成する際に、一部のインデックスがウォーム層にある場合、スナップショットの作成は失敗します。スナップショットに複数のインデックスタイプを含める必要がある場合は、SM でこのオプションがサポートされるまでは、ISM スナップショットアクションを使用します。

アクセス許可の設定

OpenSearch Service の以前のドメインバージョンから 2.5 にアップグレードした場合、Snapshot Management のセキュリティ権限がドメインで定義されない場合があります。きめ細かいアクセスコントロールを使用して、ドメインでスナップショット管理を使用するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。スナップショット管理ロールを手動で作成するときは、次の手順を実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	許可
`snapshot_management_full_access`	• `cluster:admin/opensearch/snapshot_management/*` • `cluster:admin/opensearch/notifications/feature/publish` • `cluster:admin/repository/*` • `cluster:admin/snapshot/*`
`snapshot_management_read_access`	• `cluster:admin/opensearch/snapshot_management/policy/get` • `cluster:admin/opensearch/snapshot_management/policy/search` • `cluster:admin/opensearch/snapshot_management/policy/explain` • `cluster:admin/repository/get` • `cluster:admin/snapshot/get`

3. [ロール]、[ロールの作成] の順に選択します。
4. ロールに snapshot_management_role という名前を付けます。
5. [Cluster permissions] (クラスターのアクセス権限) で、`snapshot_management_full_access` または `snapshot_management_read_access` を選択します。
6. [作成] を選択します。
7. ロールを作成したら、任意のユーザー、またはスナップショットを管理するバックエンドロールにそれをマッピングします。

考慮事項

Snapshot Management を設定するときは、次の点を考慮します。

- リポジトリごとに使用できるポリシーは 1 つです。
- 1 つのポリシーで最大 400 のスナップショットを作成できます。

- この機能は、ドメインのステータスが赤色の場合、JVM の負荷が高い (85% 以上) 場合、スナップショット機能が停止している場合は、実行されません。クラスターの全体的なインデックス作成と検索のパフォーマンスが影響を受けていると、SM も影響を受けている可能性があります。
- スナップショットのオペレーションは、前のオペレーションが完了した後に始まるため、1 つのポリシーでスナップショットのオペレーションが並行して実行されることはありません。
- 同じスケジュールのポリシーが複数存在すると、リソースが急増する可能性があります。ポリシーのスナップショット化したインデックスが重複している場合、シャードレベルのスナップショットオペレーションは順番にしか実行されないため、パフォーマンスの問題が連鎖的に発生する可能性があります。ポリシーがリポジトリを共有すると、そのリポジトリへの書き込みオペレーションが急増します。
- 特別なユースケースでない限り、スナップショットオペレーションの自動化は 1 時間に 1 回以下でスケジュールすることが推奨されます。

自動スナップショットの一時停止

OpenSearch Service では、ドメインの自動スナップショットを一時的に一時停止できます。これは、データ移行を実行したり、メンテナンスオペレーションを実行したり、ピークトラフィック期間中にスナップショットのオーバーヘッドを回避したりする必要がある場合に便利です。

ドメイン設定AutomatedSnapshotPauseOptionsで を使用して一時停止を設定します。オプションには、 enabledフラグ、StartTime、および が含まれますEndTime。時間は 秒単位の Unix エポックタイムスタンプであり、1 回の一時停止の最大期間は 72 時間を超えることはできません。は 90 日以上後に設定EndTimeすることはできません。

を省略するとStartTime、OpenSearch Service は直ちに一時停止を開始します。一時停止が現在アクティブで、 を省略した場合StartTime、既存の開始時間は保持されます。

一時停止ウィンドウには次の状態があります。

- DISABLED — 一時停止は設定されていません。
- SCHEDULED — 一時停止は、将来の開始時刻で設定されます。
- ACTIVE — 一時停止は現在有効です。
- COMPLETED — 一時停止ウィンドウが経過しました。

> **Important**
>
> 自動スナップショットが一時停止している間、ドメインのデータ保護が低下します。一時停止ウィンドウ中は、どの時点にも復元できません。この機能は、短期的な運用上のニーズにのみ使用します。

自動スナップショットを一時停止するには、ドメイン設定を更新するときに次のパラメータを含めます。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
  "AutomatedSnapshotPauseOptions": {
    "Enabled": true,
    "StartTime": 1737100800,
    "EndTime": 1737360000
  }
}
```

StartTime は省略可能です。これを省略すると、一時停止はすぐに開始されます。 EndTimeは Enabled でtrue、OpenSearch Service が自動スナップショットの作成を再開するタイミングを指定するときに必要です。

一時停止が終了する前にキャンセルするには、 Enabled または falseなしで を StartTime に設定しますEndTime。無効化時に開始時刻または終了時刻を指定することは許可されず、検証エラーが発生します。

一時停止が完了したら、新しい一時停止の は、前の一時停止の から少なくとも 1 時間後StartTimeである必要がありますEndTime。これにより、OpenSearch Service は連続する一時停止ウィンドウ間で少なくとも 1 つのスナップショットを作成します。

一時停止が の間はACTIVE、 のみを更新EndTimeしてウィンドウを短縮または拡張できます。一時停止中は、 を変更StartTimeできません。更新したウィンドウの合計期間は引き続き 72 時間を超えることはできません。

スナップショットが一時停止されると、OpenSearch Service コンソールのドメイン詳細ページに、一時停止終了時間とデータ保護ステータスの低下を示す警告バナーが表示されます。

インデックスステート管理を用いたスナップショットの自動化

Index State Management (ISM) スナップショット オペレーションを使用して、経過時間、サイズ、ドキュメント数の変化に基づいてインデックスのスナップショットを自動的にトリガーできます。ISM は、インデックスごとに 1 つのスナップショットが必要な場合に最適です。インデックスグループのスナップショットが必要な場合は、「Snapshot Management を用いたスナップショットの自動化」を参照してください。

OpenSearch Service で SM を使用するときは、ユーザー独自の Amazon S3 リポジトリを登録する必要があります。snapshot オペレーションを使用した ISM ポリシーの例については、「サンプルポリシー」を参照してください。

スナップショットの Curator の使用

ISM がインデックスとスナップショットの管理に機能しない場合は、代わりに Curator を使用できます。複雑なクラスターでの管理タスクの単純化に役立つ、アドバンストフィルタリング機能が提供されています。pip を使用して Curator をインストールします。

```
pip install elasticsearch-curator
```

Curator はコマンドラインインターフェース (CLI) または Python API として使用できます。Python API を使用する場合は、従来の elasticsearch-py クライアントのバージョン 7.13.4 以前を使用する必要があります。opensearch-py クライアントは、サポートされていません。

CLI を使用する場合は、コマンドラインで認証情報をエクスポートして curator.yml を次のように設定します。

```
client:
  hosts: search-my-domain.us-west-1.es.amazonaws.com
  port: 443
  use_ssl: True
  aws_region: us-west-1
  aws_sign_request: True
  ssl_no_validate: False
  timeout: 60

logging:
  loglevel: INFO
```

Amazon OpenSearch Service ドメインのアップグレード

> **Note**
>
> OpenSearch および Elasticsearch バージョンのアップグレードは、サービスソフトウェア更新とは異なります。OpenSearch Service ドメインのサービスソフトウェアの更新については、「the section called “サービスソフトウェア更新”」を参照してください。

Amazon OpenSearch Service では、OpenSearch 1.0 以降または Elasticsearch 5.1 以降を実行するドメインに対して、インプレースアップグレードを提供しています。Amazon Data Firehose や Amazon CloudWatch Logs などのサービスを使用して OpenSearch Service にデータをストリーミングする場合は、移行前にこれらのサービスが OpenSearch の新しいバージョンをサポートしていることを確認します。

サポートされているアップグレードパス

現在、OpenSearch Service は次のアップグレードパスをサポートしています。

元のバージョン	目的のバージョン
OpenSearch 1.3、2.x、または 3.x	OpenSearch 3.x OpenSearch 1.3 または 2.x を実行している場合は、OpenSearch 3.x にアップグレードする前に、まず OpenSearch 2.19 にアップグレードする必要があります。 OpenSearch 2.19 から OpenSearch 3.x にアップグレードする場合、次の互換性のないインデックス設定に対処する必要があります。これらは 2.x の前半で廃止され、削除しないとアップグレードチェックの検証に失敗します。 • index.knn.algo_param.ef_construction • index.knn.algo_param.m • index.knn.space_type • index.store.hybrid.mmap.extensions

元のバージョン	目的のバージョン
	廃止されたインデックス設定とその修正手順の詳細については、「Index settings」を参照してください。OpenSearch 3.x の重大な変更の完全なリストについては、「3.0.0」を参照してください。 ドメインに OpenSearch 1.3、Elasticsearch 7.10、またはそれ以前のバージョンで作成されたインデックスが含まれている場合は、OpenSearch 3.x にアップグレードする前にインデックスを再作成する必要があります。OpenSearch 3.x は、ホットストレージ、UltraWarm ストレージ、コールドストレージのいずれにあるかにかかわらず、これらの古いインデックスをサポートしていません。互換性のない UltraWarm インデックスまたはコールドインデックスを再インデックスするには: 1. インデックスをホットストレージに移行します。 2. データのインデックスを再作成します。 3. インデックスをウォームストレージまたはコールドストレージに戻します。 または、不要になったインデックスを削除します。

元のバージョン	目的のバージョン
OpenSearch 1.3 または 2.x	OpenSearch 2.x OpenSearch 2.17 は、ドメインが次の条件を満たす場合、デフォルトで自動モードで同時セグメント検索を有効にします。 • 以前の同時検索設定は明示的に設定されていません。 • すべてのデータインスタンス (ホットおよびウォーム) はインスタンスタイプ 2.xl 以上です。 • 1 週間以上のデータインスタンス (ホットおよびウォーム) の平均 p90 cpu 使用率は 45% 未満です。 ここでの同時セグメント検索設定の詳細については、「Concurrent segment search」を参照してください。 バージョン 2.3 には、次のような重要な変更点があります。 • type パラメータは、バージョン 2.0 ですべての OpenSearch API エンドポイントから削除されました。詳細については、「breaking changes」(重要な変更点) を参照してください。 • ドメインに Elasticsearch 6.8 で最初に作成されたインデックス (ホット、UltraWarm、またはコールド) が含まれている場合、それらのインデックスは OpenSearch 2.3 と互換性がありません。 バージョン 2.3 にアップグレードする前に、互換性のないインデックスを再インデックスする必要があります。互換性のない UltraWarm またはコールドインデックスについては、ホットストレージに移行し、データのインデックスを再作成してから、ウォームまたはコールドストレージに戻します。または、インデックスが不要になった場合は削除することもできます。 最初にこれらの手順を実行することなく、誤ってドメインをバージョン 2.3 にアップグレードした場合、互換性のないインデックスを現在のストレージ階層から移行することができなくなります。唯一の選択肢はインデックスを削除することです。

元のバージョン	目的のバージョン
OpenSearch 1.x	OpenSearch 1.x
Elasticsearch 7.x	Elasticsearch 7.x または OpenSearch 1.x **Important** OpenSearch 1.x では多数の画期的な変更が行われています。詳細については、「Amazon OpenSearch Service 名称変更」を参照してください。
Elasticsearch 6.8	Elasticsearch 7.x または OpenSearch 1.x **Important** Elasticsearch 7.0 と OpenSearch 1.0 には、数多くの大きな変更が含まれています。インプレースアップグレードを開始する前に、6.x ドメインの手動スナップショットを作成し、それをテスト用の 7.x または OpenSearch 1.x ドメインで復元して、そのテストドメインを使用してアップグレードの潜在的な問題を特定することが推奨されます。OpenSearch 1.0 での変更については、「Amazon OpenSearch Service 名称変更」を参照してください。 Elasticsearch 6.x と同様に、インデックスに含めることができるマッピングタイプは 1 つだけですが、そのタイプには _doc という名前を付ける必要があります。その結果、特定の API では、リクエスト本文にマッピングタイプ (_bulk API など) が不要になりました。 新しいインデックスの場合、セルフホストの Elasticsearch 7.x と OpenSearch 1.x のデフォルトのシャード数は 1 です。Elasticsearch 7.x以降での OpenSearch Service ドメインでは、以前のデフォルトの 5 のままです。
Elasticsearch 6.x	Elasticsearch 6.x

元のバージョン	目的のバージョン
Elasticsearch 5.6	Elasticsearch 6.x **Important** バージョン 6.x で作成されたインデックスは、複数のマッピングタイプをサポートしなくなりました。バージョン 5.x で作成されたインデックスは、6.x クラスターに復元された場合でも、複数のマッピングタイプをサポートします。クライアントコードによって作成されるマッピングタイプがインデックスごとに 1 種類のみであることを確認します。Elasticsearch 5.6 から 6.x へのアップグレード中のダウンタイムを最小限に抑えるため、OpenSearch Service は `.kibana` インデックスを `.kibana-6` に再生成し、`.kibana` を削除して `.kibana` という名前でエイリアスを作成します。それから、新しいインデックスを新しいエイリアスにマッピングします。
Elasticsearch 5.x	Elasticsearch 5.x

アップグレードプロセスには、3 つの手順が含まれます。

1. アップグレード前の確認 – OpenSearch Service は、アップグレードをブロックする可能性のある問題を確認し、これらの確認が成功しない限り、次のステップに進みません。
2. スナップショット – OpenSearch Service は、OpenSearch または Elasticsearch クラスターのスナップショットを取得し、スナップショットが成功しない限り次のステップに進みません。アップグレードが失敗した場合、OpenSearch Service はこのスナップショットを使用してクラスターを元の状態に復元します。詳細については、「the section called “アップグレード後にダウングレードできない”」を参照してください。
3. アップグレード – OpenSearch Service がアップグレードを開始すると、完了まで 15 分から数時間かかります。OpenSearch Dashboards は、アップグレードの一部または全期間で使用できなくなる場合があります。

ドメインのアップグレード (コンソール)

アップグレードプロセスは元に戻せず、一時停止またはキャンセルすることはできません。アップグレード中、ドメインの設定を変更することはできません。アップグレードを開始する前に、続行するかどうかを再度確認します。これらの同じ手順を使用して、アップグレードを実際に開始することなくアップグレード前の確認を実行することができます。

クラスタに専用のマスターノードがある場合、OpenSearch のアップグレードはダウンタイムなしで完了します。ただし、マスターノードの再選択中に短時間の 1 秒未満の中断 (約 300 ミリ秒) が発生する場合があります。ない場合、クラスターがマスターノードを選択している間、アップグレード後数秒間応答しなくなることがあります。

ドメインを新しいバージョンの OpenSearch または Elasticsearch にアップグレードするには

1. ドメインの手動スナップショットを作成します。以前のバージョンの Opensearch の使用に戻る場合、このスナップショットは、新しいドメインで復元できるバックアップとして機能します。
2. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
3. [分析] の下で、[Amazon OpenSearch Service] を選択します。
4. ナビゲーションペインの [Domains] (ドメイン) で、アップグレードするドメインを選択します。
5. [アクション] から [更新] を選択します。
6. アップグレードするバージョンを選択します。OpenSearch バージョンにアップグレードする場合、[互換モードを有効にする] オプションが表示されます。この設定を有効にすると、OpenSearch はそのバージョンを 7.10 とレポートし、Elasticsearch の OSS クライアントや Logstash などのプラグインが Amazon OpenSearch Service での使用を継続できるようにします。この設定は後で無効にできます
7. [アップグレード] を選択します。
8. ドメインダッシュボードのステータスをチェックし、アップグレードのステータスをモニタリングします。

ドメインのアップグレード (CLI)

以下のオペレーションを使用して、ドメイン用の正しい OpenSearch または Elasticsearch バージョンを識別できます。インプレースアップグレードを開始し、アップグレード前のチェックを実行して進捗状況を表示できます。

- `get-compatible-versions` (`GetCompatibleVersions`)

- `upgrade-domain` (`UpgradeDomain`)
- `get-upgrade-status` (`GetUpgradeStatus`)
- `get-upgrade-history` (`GetUpgradeHistory`)

詳細については、「AWS CLI コマンドリファレンス」と「Amazon OpenSearch Service API リファレンス」を参照してください。

ドメインのアップグレード (SDK)

このサンプルでは、 の OpenSearchService 低レベル Python クライアント AWS SDK for Python (Boto) を使用して、ドメインが特定のバージョンへのアップグレードの対象であるかどうかを確認し、アップグレードし、アップグレードステータスを継続的にチェックします。

```
import boto3
from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

DOMAIN_NAME = ''  # The name of the domain to upgrade
TARGET_VERSION = ''  # The version you want to upgrade the domain to. For example,
 OpenSearch_1.1

my_config = Config(
    # Optionally lets you specify a Region other than your default.
    region_name='us-east-1'
)
client = boto3.client('opensearch', config=my_config)


def check_versions():
    """Determine whether domain is eligible for upgrade"""
    response = client.get_compatible_versions(
        DomainName=DOMAIN_NAME
    )
    compatible_versions = response['CompatibleVersions']
    for i in range(len(compatible_versions)):
        if TARGET_VERSION in compatible_versions[i]["TargetVersions"]:
            print('Domain is eligible for upgrade to ' + TARGET_VERSION)
```

```
            upgrade_domain()
            print(response)
        else:
            print('Domain not eligible for upgrade to ' + TARGET_VERSION)


def upgrade_domain():
    """Upgrades the domain"""
    response = client.upgrade_domain(
        DomainName=DOMAIN_NAME,
        TargetVersion=TARGET_VERSION
    )
    print('Upgrading domain to ' + TARGET_VERSION + '...' + response)
    time.sleep(5)
    wait_for_upgrade()


def wait_for_upgrade():
    """Get the status of the upgrade"""
    response = client.get_upgrade_status(
        DomainName=DOMAIN_NAME
    )
    if (response['UpgradeStep']) == 'UPGRADE' and (response['StepStatus']) ==
 'SUCCEEDED':
        print('Domain successfully upgraded to ' + TARGET_VERSION)
    elif (response['StepStatus']) == 'FAILED':
        print('Upgrade failed. Please try again.')
    elif (response['StepStatus']) == 'SUCCEEDED_WITH_ISSUES':
        print('Upgrade succeeded with issues')
    elif (response['StepStatus']) == 'IN_PROGRESS':
        time.sleep(30)
        wait_for_upgrade()


def main():
    check_versions()


if __name__ == "__main__":
    main()
```

検証障害のトラブルシューティング

OpenSearch または Elasticsearch のバージョンアップグレードを開始すると、OpenSearch Service は最初に一連の検証チェックを実行して、ドメインがアップグレードのための要件を満たしていることを確認します。これらのチェックのいずれかが失敗した場合、ドメインを更新する前に修正する必要がある特定の問題を含む通知が表示されます。発生する可能性のある問題とその解決手順の一覧については、the section called “検証エラーのトラブルシューティング” を参照してください。

アップグレードのトラブルシューティング

インプレース アップグレードには、正常なドメインが必要です。ドメインがアップグレードの対象とならなかったり、さまざまな理由でアップグレードに失敗する場合があります。次の表は、最も一般的な問題を示しています。

問題	説明
オプションのプラグインはサポートされていません	オプションプラグインを使用してドメインをアップグレードすると、OpenSearch Service はプラグインも自動的にアップグレードします。そのため、ドメインのターゲットバージョンは、これらのオプションプラグインもサポートしている必要があります。ターゲットバージョンでは使用できないオプションのプラグインがドメインにインストールされている場合、アップグレードリクエストは失敗します。
ノードあたりのシャードが多すぎます	OpenSearchだけでなく、7.x バージョンの Elasticsearch のデフォルト設定も、ノードあたり 1,000 シャード以下です。現在のクラスター内のノードがこの設定を超えている場合、OpenSearch Service はアップグレードを許可しません。トラブルシューティングのオプションについては、「the section called “シャードの最大制限を超えました”」を参照してください。
処理中のドメイン	ドメインは、設定変更中です。オペレーション完了後、アップグレードの適格性を確認します。
赤のクラスター状態	クラスター内の 1 つ以上のインデックスが赤です。トラブルシューティングステップについては、「the section called “赤のクラスター状態”」を参照してください。
高いエラー率	クラスターはリクエストの処理を試みましたが、大量の 5xx エラーを返しています。この問題は通常、同時読み取りまたは書き込みリクエス

問題	説明
	トが多すぎることで起こります。クラスターへのトラフィックを減らすか、ドメインのスケーリングを検討してください。
スプリットブレイン	スプリットブレインとは、 クラスターに 1 つ以上のマスターノードがあり、決して独自に再結合することのない 2 つのクラスターに分割されているという意味です。専用マスターノードの推奨値を使用することで、スプリットブレインを回避できます。スプレッドブレインからの復旧方法については、サポート にお問い合わせください。
マスターノードが見つからない	OpenSearch Service は、クラスターのマスターノードを見つけることができません。ドメインでマルチ AZ を使用している場合、アベイラビリティーゾーンの障害によってクラスターがクォーラムを失い、新しいマスターノードを選択できなくなる可能性があります。問題が自動的に解決されない場合、サポート にお問い合わせください。
保留中のタスクが多すぎる	マスターノードに高い負荷がかかっており、たくさんの保留中のタスクがあります。クラスターへのトラフィックを減らすか、ドメインのスケーリングを検討してください。
障害が発生したストレージボリューム	1 つ以上のノードのディスクボリュームが正常に機能していません。この問題は、高いエラー率や保留中のタスクが多すぎるなど、他の問題とともに頻繁に発生します。問題が単独で発生し、自動的に解決されない場合は、サポート にお問い合わせください。
KMS キーの問題	ドメインの暗号化に使用する KMS キーがアクセス不可能であるか、または存在しません。詳細については、「the section called “保管中のデータを暗号化するドメインのモニタリング”」を参照してください。
スナップショットを作成中	ドメインは現在、スナップショットを作成しています。スナップショットの完了後に、アップグレードの適格性を確認します。手動スナップショットのリポジトリが一覧表示できること、それらのリポジトリ内のスナップショットが一覧表示できること、および手動スナップショットが作成できることも確認してください。OpenSearch Service が、スナップショットが作成中かどうか確認できない場合、アップグレードは失敗します。

問題	説明
スナップショットのタイムアウトまたは障害	アップグレード前のスナップショットの完了まで時間がかかりすぎたか、失敗しました。クラスター状態を確認して、再度お試しください。問題が解決しない場合は、サポート までお問い合わせください。
互換性のないインデックス	1 つ以上のインデックスがターゲットバージョンと互換性がありません。この問題は、以前のバージョンの OpenSearch または Elasticsearch からインデックスを移行した場合に発生する可能性があります。インデックスを再作成して、再試行してください。
高いディスク使用率	クラスターのディスク使用率が 90% を超えています。データを削除またはドメインをスケーリングして、再度お試しください。
高い JVM 使用率	JVM メモリプレッシャーが 75% を超えています。クラスターへのトラフィックを削減するか、ドメインをスケーリングして、再度お試しください。
Opensearch Dashboards エイリアスの問題	`.dashboards` はエイリアスとして既に設定されており、互換性のないインデックスにマッピングされています。おそらく、以前のバージョンの OpenSearch Dashboards からのインデックスです。インデックスを再作成して、再度お試しください。
赤の Dashboards ステータス	Opensearch Dashboards のステータスが赤くなっています。アップグレードが完了したら、Dashboards を使用してみてください。赤いステータスが続く場合は、手動で解決してから再度お試しください。
クラスター間の互換性	アップグレードできるのは、アップグレード後にソースドメインとデスティネーションドメインの間で、クラスター間の互換性が維持されている場合のみです。アップグレードプロセス中に、互換性のない接続があれば特定されます。続行するには、リモートドメインをアップグレードするか、互換性のない接続を削除します。ドメインでレプリケーションがアクティブな場合は、接続を削除した後でレプリケーションを再開できないことに注意してください。

問題	説明
OpenSearch Service のサービスに関するその他の問題	OpenSearch Service 自体の問題により、ドメインがアップグレード対象外として表示される可能性があります。前述の条件がドメインに当てはまらず、問題が 1 日以上続く場合は、サポート にお問い合わせください。

スナップショットを使用してデータを移行する

インプレースアップグレードは、ドメインを新しい OpenSearch または Elasticsearch バージョンにアップグレードする従来よりも簡単で迅速な、信頼性の高い方法です。スナップショットは、Elasticsearch の 5.1 より前のバージョンから移行する必要がある場合や、まったく新しいクラスターに移行する場合に適しています。

次の表は、スナップショットを使用して、OpenSearch または Elasticsearch の異なるバージョンを使用するドメインにデータを移行する方法を示しています。スナップショットの作成と復元の詳細については、「the section called “インデックススナップショットの作成”」を参照してください。

元のバージョン	目的のバージョン	移行プロセス

OpenSearch 1.3 または 2.x	OpenSearch 2.x	1. OpenSearch 2.3 の重大な変更を確認して、インデックスまたはアプリケーションを調整する必要があるかどうかを確認します。
		2. 1.3 または 2.x ドメインの手動スナップショットを作成します。
		3. 元の 1.3 または 2.x ドメインよりも上位のバージョンである 2.x ドメインを作成します。
		4. 元のドメインのスナップショットを 2.x ドメインに復元します。オペレーション中に、新しい名前で `.opensearch` インデックスを復元することが必要になる場合があります。
		`POST _snapshot/ <repository-name> /<snapshot-name>/_restore`
		`{`
		`  "indices": "*",`

元のバージョン	目的のバージョン	移行プロセス

		<pre>"ignore_unavailable": true,
		"rename_pattern": ".opensearch",
		"rename_replacement": ".backup-opensearch"
		}</pre>
		次に、新しいドメインで .backup-opensearch のインデックスを再作成し、エイリアス .opensearch を付けることができます。_restore のデフォルトは false であるため、_restore REST 呼び出しには include_global_state は含まれません。その結果、テストドメインにはインデックステンプレートが含まれず、バックアップの完全な状態も含まれません。
		5. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス

OpenSearch 1.x	OpenSearch 1.x	1. 1.x ドメインの手動スナップショットを作成します。
		2. 元の 1.x ドメインよりも上位のバージョンである 1.x ドメインを作成します。
		3. 元のドメインのスナップショットを新しい 1.x ドメインに復元します。オペレーション中に、新しい名前で .opensearch インデックスを復元することが必要になる場合があります。
		`POST _snapshot/ <repository-name> /<snapshot-name>/_restore`
		`{`
		`  "indices": "*",`
		`  "ignore_unavailable": true,`
		`  "rename_pattern": ".opensearch",`
		`  "rename_replacement": ".backup-opensearch"`
		`}`
		次に、新しいドメインで .backup-opensearch のインデックスを再作成し、エイリアス .opensearch を付けることができます。_restore のデフォルトは false であるため、_restore REST 呼び出しには include_global_state は含まれません。その結果、テストドメインにはインデックステンプレートが含まれず、バックアップの完全な状態も含まれません。
		4. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス

Elasticsearch 6.x または 7.x	OpenSearch 1.x	1. OpenSearch 1.0 の重大な変更を確認して、インデックスまたはアプリケーションを調整する必要があるかどうかを確認します。
		2. Elasticsearch 7.x または 6.x ドメインの手動スナップショットを作成します。
		3. OpenSearch 1.x ドメインを作成します。
		4. Elasticsearch ドメインから OpenSearch ドメインにスナップショットを復元します。オペレーション中に、新しい名前で `.elasticsearch` インデックスを復元することが必要になる場合があります。
		`POST _snapshot/ <repository-name> /<snapshot-name>/_restore`
		`{`
		`  "indices": "*",`
		`  "ignore_unavailable": true,`
		`  "rename_pattern": ".elasticsearch",`
		`  "rename_replacement": ".backup-opensearch"`
		`}`
		次に、新しいドメインで `.backup-opensearch` のインデックスを再作成し、エイリアス `.elasticsearch` を付けることができます。`_restore` のデフォルトは false であるため、`_restore` REST 呼び出しには `include_global_state` は含まれません。その結果、テストドメインにはインデックステンプレートが含まれず、バックアップの完全な状態も含まれません。
		5. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス

Elasticsearch 6.x	Elasticsearch 7.x	1. 7.0 の重大な変更を確認して、インデックスまたはアプリケーションを調整する必要があるかどうかを確認します。
		2. 6.x ドメインの手動スナップショットを作成します。
		3. 7.x ドメインを作成します。
		4. 元のドメインのスナップショットを 7.x ドメインに復元します。オペレーション中に、新しい名前で `.opensearch` インデックスを復元することが必要になる場合あります。
		
		`POST _snapshot/ <repository-name> /<snapshot-name>/_restore`
		`{`
		`  "indices": "*",`
		`  "ignore_unavailable": true,`
		`  "rename_pattern": ".elasticsearch",`
		`  "rename_replacement": ".backup-elasticsearch"`
		`}`
		
		次に、新しいドメインで `.backup-elasticsearch` のインデックスを再作成し、エイリアス `.elasticsearch` を付けることができます。`_restore` のデフォルトは false であるため、`_restore` REST 呼び出しには `include_global_state` は含まれません。その結果、テストドメインにはインデックステンプレートが含まれず、バックアップの完全な状態も含まれません。
		5. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス
Elasticsearch 6.x	Elasticsearch 6.8	1. 6.x ドメインの手動スナップショットを作成します。 2. 6.8 ドメインを作成します。 3. 元のドメインのスナップショットを 6.8 ドメインに復元します。 4. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。
Elasticsearch 5.x	Elasticsearch 6.x	1. インデックスやアプリケーションを調整する必要があるかどうかを確認するには、「6.0 の重要な変更」を参照します。 2. 5.x ドメインの手動スナップショットを作成します。 3. 6.x ドメインを作成します。 4. 元のドメインのスナップショットを 6.x ドメインに復元します。 5. 5.x ドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。
Elasticsearch 5.x	Elasticsearch 5.6	1. 5.x ドメインの手動スナップショットを作成します。 2. 5.6 ドメインを作成します。 3. 元のドメインのスナップショットを 5.6 ドメインに復元します。 4. 元のドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス
Elasticsearch 2.3	Elasticsearch 6.x	Elasticsearch 2.3 のスナップショットは 6.x とは互換性がありません。データを 2.3 から 6.x に直接移行するには、新しいドメインでインデックスを手動で再作成する必要があります。 または、この表の 2.3 から 5.x のステップに従い、新しい 5.x ドメインで `_reindex` オペレーションを実行して、2.3 インデックスを 5.x インデックスに変換してから、5.x から 6.x のステップに従うこともできます。
Elasticsearch 2.3	Elasticsearch 5.x	1. 5.0 の重大な変更を確認して、インデックスまたはアプリケーションを調整する必要があるかどうかを確認します。 2. 2.3 ドメインの手動スナップショットを作成します。 3. 5.x ドメインを作成します。 4. 2.3 ドメインのスナップショットを 5.x ドメインに復元します。 5. 2.3 ドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

元のバージョン	目的のバージョン	移行プロセス
Elasticsearch 1.5	Elasticsearch 5.x	Elasticsearch 1.5 のスナップショットは 5.x とは互換性がありません。データを 1.5 から 5.x に移行するには、新しいドメインでインデックスを手動で再作成する必要があります。 **Important** 1.5 のスナップショットには 2.3 との互換性がありますが、OpenSearch Service 2.3 のドメインでは `_reindex` オペレーションがサポートされていません。それらのインデックスを再作成できないため、1.5 ドメインで作成されたインデックスは、2.3 スナップショットから 5.x ドメインへの復元に失敗します。
Elasticsearch 1.5	Elasticsearch 2.3	1. 移行プラグインを使用して、バージョン 2.3 に直接アップグレードできるかどうかを確認します。移行前にデータの変更が必要になる場合もあります。 a. ウェブブラウザで、`http://`*domain-endpoint*`/_plugin/migration/` を開きます。 b. [今すぐ確認を実行] を選択します。 c. 結果を確認し、必要に応じて、手順に従いデータを変更します。 2. 1.5 ドメインの手動スナップショットを作成します。 3. 2.3 ドメインを作成します。 4. 1.5 ドメインのスナップショットを 2.3 ドメインに復元します。 5. 1.5 ドメインが不要であれば、削除します。削除しなかった場合は、ドメインの利用料金が引き続き発生します。

Amazon OpenSearch Service 用のカスタムエンドポイントの作成

Amazon OpenSearch Service ドメインのカスタムエンドポイントを作成すると、OpenSearch および OpenSearch Dashboards の URL を簡単に参照できるようになります。会社のブランディングを含めることや、標準のエンドポイントではなく短く覚えやすいエンドポイントを使用することができます。

新しいドメインに切り替える必要がある場合は、新しい URL を指すように DNS を更新し、以前と同じエンドポイントを引き続き使用します。

カスタムエンドポイントを保護するには、 AWS Certificate Manager (ACM) で証明書を生成するか、独自の証明書をインポートします。

新しいドメインのカスタムエンドポイント

OpenSearch Service コンソール、または設定 API を使用して AWS CLI、新しい OpenSearch Service ドメインのカスタムエンドポイントを有効にできます。

エンドポイントをカスタマイズするには (コンソール)

1. OpenSearch Service コンソールから [ドメインの作成] を選択し、ドメインの名前を入力します。
2. [カスタムエンドポイント] の下で、[カスタムエンドポイントを有効にする] を選択します。
3. [カスタムホスト名] では、希望するカスタムエンドポイントのホスト名を入力します。ホスト名は、www.yourdomain.com や example.yourdomain.com のような完全修飾ドメイン名 (FQDN) である必要があります。

   > **Note**
   >
   > 「ワイルドカード証明書」がない場合は、カスタムドメインのサブドメイン用に新しい証明書を取得する必要があります。

4. [AWS 証明書] では、ドメインに使用する SSL 証明書を選択します。利用可能な証明書がない場合は、証明書を ACM にインポートするか、ACM を使用して証明書をプロビジョニングできます。詳細については、AWS Certificate Manager ユーザーガイドの「証明書の発行と管理」を参照してください。

> **Note**
>
> 証明書にはカスタムエンドポイント名があり、OpenSearch Service ドメインと同じアカウントに存在する必要があります。証明書のステータスは「ISSUED」である必要があります。

- 残りのステップに従って、ドメインを作成し、[作成] を選択します。
- 処理が完了したら、ドメインを選択して、カスタムエンドポイントを表示します。

CLI または設定 API を使用するには、`CreateDomain` および `UpdateDomainConfig` オペレーションを使用します。詳細については、「AWS CLI コマンドリファレンス」と「Amazon OpenSearch Service API リファレンス」を参照してください。

既存のドメインのカスタムエンドポイント

既存の OpenSearch Service ドメインにカスタムエンドポイントを追加するには、[編集] をクリックし、上記のステップ 2～4 を実行します。

CNAME マッピング

OpenSearch Service ドメインのカスタムエンドポイントを有効にしたら、Amazon Route 53 (または希望する DNS サービスプロバイダー) で CNAME マッピングを作成することができます。CNAME マッピングを作成すると、カスタムエンドポイントとそのサブドメインにトラフィックをルーティングできます。このマッピングがないと、トラフィックをカスタムエンドポイントにルーティングできなくなります。Route 53 でこのマッピングを作成する手順については、「Configuring DNS routing for a new domain」および「Creating a new hosted zone for a subdomain」を参照してください。他のプロバイダについては、そのドキュメントを参照してください。

カスタムエンドポイントが自動生成されたドメインエンドポイントを指す CNAME レコードを作成します。ドメインがデュアルスタックの場合は、サービスによって生成された 2 つのエンドポイントのいずれかに CNAME レコードを指定できます。カスタムエンドポイントのデュアルスタック機能は、サービスによって指定された CNAME レコードの指定先のエンドポイントに応じて異なります。カスタムエンドポイントホスト名は CNAME レコードの名前であり、ドメインエンドポイントホスト名は CNAME レコードの値です。

OpenSearch Dashboards の SAML 認証を使用する場合は、IdP を新しい SSO URL で更新する必要があります。

Amazon Route 53 を使用してエイリアスレコードタイプを作成し、ドメインのカスタムエンドポイントをデュアルスタック検索エンドポイントにポイントできます。エイリアスレコードタイプを作成するには、デュアルスタック IP アドレスタイプを使用するようにドメインを設定する必要があります。これは、Route 53 API を使用して行うこともできます。

Route 53 API を使用してエイリアスレコードタイプを作成するには、ドメインのエイリアスターゲットを指定します。ドメインのエイリアスターゲットは、OpenSearch Service コンソールのカスタムエンドポイントセクションの[ホストゾーン (デュアルスタック)] フィールド内で、または `DescribeDomain` API を使用して `DomainEndpointV2HostedZoneId` の値をコピーすることで見つけることができます。

Amazon OpenSearch Service の Auto-Tune

Amazon OpenSearch Service の Auto-Tune では、OpenSearch クラスターのパフォーマンスと使用状況のメトリクスを使用して、メモリ関連の設定の変更を提案します。これには、キューとキャッシュのサイズ、ノードの Java 仮想マシン (JVM) 設定が含まれます。これらのオプションの変更により、クラスターの速度と安定性が向上します。

変更は、すぐにデプロイされるものもあれば、ドメインのオフピークウィンドウにスケジュールされるものもあります。いつでもデフォルトの OpenSearch Service 設定に戻すことができます。Auto-Tune がドメインのパフォーマンスメトリクスの収集および分析を行うと、通知ページの OpenSearch Service コンソールでレコメンデーションを確認できます。

> **Important**
>
> ドメインにカスタムオフピークウィンドウを設定していない場合、Auto-Tune はデフォルトのオフピークウィンドウを使用します。このウィンドウには、ブルー/グリーンデプロイの変更 (JVM ヒープサイズや JVM の新世代調整など) が適用されます。Auto-Tune を有効にして予期しないブルー/グリーンデプロイを回避する前に、オフピークウィンドウ設定を確認して設定していることを確認してください。詳細については、「Amazon OpenSearch Service のオフピークウィンドウ」を参照してください。

Auto-Tune は、サポートされているインスタンスタイプで任意の OpenSearch バージョンまたは Elasticsearch 6.7 以降を実行しているドメインで使用できます。

変更のタイプ

Auto-Tune には、大きく分けて次の 2 つのカテゴリがあります。

- クラスターの実行時に適用される、無停止での変更。
- ブルー/グリーンデプロイを必要とする変更。これは、ドメインのオフピークウィンドウに適用されます。

Auto-Tune では、ドメインのパフォーマンスメトリクスに基づいて、次の設定の調整を提案できます。

タイプの変更	Category	説明
JVM ヒープサイズ	Blue/Green	デフォルトでは、OpenSearch Service は JVM ヒープ (最大 32 GiB のヒープサイズ) にインスタンスの RAM の 50% を使用します。 この割合を増やすと、OpenSearch のメモリは増えますが、オペレーティングシステムやその他のプロセスでは減ります。値を大きくすると、ガベージコレクションの一時停止の数は減りますが、一時停止の長さが増えます。
JVM 新世代設定	Blue/Green	JVM「新世代」設定は、マイナーガベージコレクションの頻度に影響します。マイナーコレクションを頻繁に行うと、メジャーコレクションの数が減り、一時停止することがあります。
キューサイズ	無停止	デフォルトでは、検索キューのサイズは `1000` であり、書き込みキューのサイズは `10000` です。リクエストを処理するために追加のヒープが利用可能な場合、Auto-Tune は検索キューと書き込みキューを自動的にスケーリングします。
キャッシュサイズ	無停止	フィールドキャッシュは、ヒープ上のデータ構造をモニタリングするので、キャッシュの使用をモニタリングすることが重要です。Auto-Tune は、フィールドデータキャッシュサイズをスケーリングして、メモリ不足や回路ブレーカーの問題を回避します。 シャードリクエストキャッシュはノードレベルで管理され、デフォルトの最大サイズはヒープの 1% です。Auto-Tune は、シャードリ

タイプの変更	Category	説明
		クエストキャッシュのサイズをスケーリングして、設定されたクラスターが処理できるよりも多くの検索およびインデックスリクエストを受け入れます。
リクエストサイズ	無停止	デフォルトでは、実行中のリクエストの集計サイズが合計 JVM の 10% を超えている場合 (t2 インスタンスタイプでは 2%、t3.small では 1%)、既存のリクエストが完了するまで、OpenSearch はすべての新しい _search および _bulk をスロットルします。 Auto-Tune は、システム上で現在占有されている JVM の量に基づいて、このしきい値 (通常は 5 ~ 15%) を自動的に調整します。例えば、JVM のメモリ負荷が高い場合、Auto-Tune はしきい値を 5% に減らすことがあり、その時点で、クラスターが安定してしきい値が増加するまで、拒否が多くなる可能性があります。

Auto-Tune を有効または無効にする

OpenSearch Service では、新しいドメインで、デフォルトで Auto-Tune が有効になります。既存のドメインで Auto-Tune を有効または無効にするには、コンソールを使用することをお勧めします。これにより、プロセスが簡素化されます。Blue/Green デプロイを必要とする変更

現在、 AWS CloudFormationを使用して Auto-Tune を有効または無効にすることはできません。

コンソール

既存のドメインで自動調整を有効にするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ナビゲーションペインの [ドメイン] で、ドメイン名を選択して [クラスター設定] を開きます。
3. ドメインの詳細ページで、Auto-Tune タブを選択し、編集を選択してドメインの Auto-Tune をオンまたはオフにします。

4. 必要に応じて [オフピークウィンドウ] を選択し、ドメインに設定されたオフピークウィンドウにブルー/グリーンデプロイを要求する最適化をスケジュールします。詳細については、「the section called “Auto-Tune の機能強化のスケジューリング”」を参照してください。
5. [Save changes] (変更の保存) をクリックします。

CLI

を使用して Auto-Tune を有効にするには AWS CLI、UpdateDomainConfig リクエストを送信します。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --auto-tune-options DesiredState=ENABLED
```

Auto-Tune の機能強化のスケジューリング

2023 年 2 月 16 日以前、Auto-Tune はメンテナンスウィンドウを使用して、ブルー/グリーンデプロイを必要とする変更をスケジュールしていました。現在ではメンテナンスウィンドウが廃止され、オフピークウィンドウが採用されています。これは、1 日の中でドメインのトラフィックが一般的に少なくなる 10 時間の時間ブロックです。オフピークウィンドウのデフォルトの開始時間は変更できますが、時間の長さを変更することはできません。

2023 年 2 月 16 日にオフピークウィンドウが導入される前に Auto-Tune メンテナンスウィンドウを有効にしていたドメインは、従来のメンテナンスウィンドウを中断することなく引き続き使用できます。ですが、既存のドメインを移行して、代わりにドメインメンテナンスのオフピークウィンドウを使用することをお勧めします。手順については、「the section called “Auto-Tune のメンテナンスウィンドウからの移行”」を参照してください。

コンソール

オフピークウィンドウに Auto-Tune アクションをスケジュールするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ナビゲーションペインの [ドメイン] で、ドメイン名を選択して [クラスター設定] を開きます。
3. [Auto-Tune] タブに進み、[編集] を選択します。
4. ドメインの詳細ページで、Auto-Tune タブを選択し、編集を選択してドメインの Auto-Tune をオンまたはオフにします。

5. [オフピークウィンドウ中に最適化をスケジュールする] で、[オフピークウィンドウ] をクリックします。
6. [Save changes] (変更の保存) をクリックします。

CLI

設定されたオフピークウィンドウ中に Auto-Tune アクションをスケジュールするようにドメインを設定するには、UpdateDomainConfig リクエストに `UseOffPeakWindow` を含めてください。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --auto-tune-options
 DesiredState=ENABLED,UseOffPeakWindow=true,MaintenanceSchedules=null
```

オートチューンの変更の監視

Amazon CloudWatchでオートチューンの統計情報を監視できます。メトリクスの一覧については、「the section called “Auto-Tune メトリクス”」を参照してください。

OpenSearch Service はオートチューンイベントを Amazon EventBridge に送信します。EventBridge を使用して、イベントの受信時に E メールを送信したり、特定のアクションを実行したりするルールを設定できます。EventBridge に送信される各オートチューンイベントのフォーマットを見るには、「the section called “Auto-Tune イベント”」を参照してください。

Amazon OpenSearch Service ドメインのタグ付け

タグを使用すると、Amazon OpenSearch Service ドメインに任意の情報を割り当てて、その情報を分類してフィルタリングできます。タグとは、OpenSearch Service ドメインを定義して関連付けるキーと値のペアです。これらのタグを使用して、同様にタグ付けされたリソースの費用をグループ化することでコストを追跡できます。 AWS はタグに意味論的な意味を適用しません。タグは単なる文字列として解釈されます。タグには、次の要素があります。

タグ要素	説明	必須
タグキー	タグキーは、タグ名です。キーは、アタッチされている OpenSearch Service ドメインに対して一意である必要があります。タグキーと値の基本的な制限のリストについては、「ユーザー定義タグの制限」を参照してください。	はい

タグ要素	説明	必須
タグ値	タグ値は、タグの文字列値です。タグ値は null を指定できます。また、タグセット内で一意である必要はありません。例えば、project/Trinity と cost-center/Trinity のタグセット内に 1 つのキーと値のペアを使用できます。タグキーと値の基本的な制限のリストについては、「ユーザー定義タグの制限」を参照してください。	いいえ

各 OpenSearch Service ドメインにはタグセットがあり、それぞれの OpenSearch Service ドメインに割り当てられているすべてのタグが含まれています。 AWS は、OpenSearch Service ドメインにタグを自動的に割り当てません。タグセットには、0～50 個のタグを含めることができます。ドメインに追加したタグのキーが既存のタグのキーと同じ場合、既存の値は新しい値によって上書きされます。

タグ付けの例

キーを使用してカテゴリを定義し、タグ値をそのカテゴリの項目にすることができます。具体的には、「project」というタグキーと「Salix」というタグ値を定義して、OpenSearch Servic ドメインが Salix プロジェクトに割り当てられていることを示すことができます。また、タグキーとして environment=test や environment=production などを使用して、OpenSearch Service ドメインがテスト用なのか本稼働用なのかを示すこともできます。OpenSearch Service ドメインに関連付けられているメタデータの追跡が簡単になるように、一貫した一連のタグキーを使用することを試してください。

タグを使用して AWS 請求書を整理し、独自のコスト構造を反映することもできます。これを行うには、サインアップしてタグキー値を含む AWS アカウント 請求書を取得します。その後、同じタグキー値を持つリソースに従って請求情報を整理し、結合したリソースのコストを確認します。例えば、複数の OpenSearch Service ドメインにキーと値のペアをタグ付けし、請求情報を整理して複数のサービスにおける各ドメインの合計コストを確認できます。詳細については、https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-alloc-tags.html 請求情報とコスト管理の「AWS コスト配分タグの使用」を参照してください。

> **Note**
>
> タグは承認用にキャッシュに格納されます。そのため、OpenSearch Service ドメインに対するタグの追加や更新には数分かかることがあります。

ドメインのタグ付け (コンソール)

コンソールは、ドメインにタグを付ける最も簡単な方法です。

タグを作成するには (コンソール)

1. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. タグを追加するドメインを選択し、[タグ] タブに移動します。
4. [管理] を選択して、[新しいタグを追加] を選択します。
5. タグキーとオプションの値を入力します。
6. [保存] を選択します。

タグを削除するには、同じ手順に従って、[タグを管理] ページで [削除] を選択します。

タグを操作するコンソールを使用する方法の詳細については、AWS マネジメントコンソール入門ガイドの「タグエディター」を参照してください。

ドメインのタグ付け (AWS CLI)

--add-tags コマンド AWS CLI で を使用してリソースタグを作成できます。

[Syntax] (構文)

```
add-tags --arn=<domain_arn> --tag-list Key=<key>,Value=<value>
```

パラメータ	説明
`--arn`	タグが添付される OpenSearch Service ドメインの Amazon リソースネーム。
`--tag-list`	スペースで区切られたキーと値のペアの以下の形式のセット: `Key=<key>,Value=<value>`

例

次の例では、[ログ] ドメイン用に 2 つのタグを作成します。

```
aws opensearch add-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs --tag-list
 Key=service,Value=OpenSearch Key=instances,Value=m3.2xlarge
```

--remove-tags コマンドを使用して OpenSearch Service ドメインからタグを削除できます。

構文

```
remove-tags --arn=<domain_arn> --tag-keys Key=<key>,Value=<value>
```

パラメータ	説明
`--arn`	タグが添付される OpenSearch Service ドメインの Amazon リソースネーム (ARN)。
`--tag-keys`	OpenSearch Service ドメインから削除するスペース区切りのキーと値のペアのセット。

例

次の例は、[ログ] ドメインから前述の例で作成した 2 つのタグを削除します。

```
aws opensearch remove-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs --tag-
keys service instances
```

--list-tags コマンドで OpenSearch Service ドメインの既存のタグを表示できます。

[Syntax] (構文)

```
list-tags --arn=<domain_arn>
```

パラメータ	説明
`--arn`	タグが添付される OpenSearch Service ドメインの Amazon リソースネーム (ARN)。

例

次の例は、[ログ] ドメインのすべてのリソースタグをリスト表示します。

```
aws opensearch list-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs
```

ドメインのタグ付け (AWS SDKs)

AWS SDKs (Android および iOS SDKs を除く) は、、、ListTagsおよび RemoveTagsオペレーションを含むAddTags、Amazon OpenSearch Service API リファレンスで定義されているすべてのアクションをサポートします。 AWS SDKsAWS 「 Software Development Kits」を参照してください。

Python

この例では、 AWS SDK for Python (Boto) の OpenSearchService 低レベル Python クライアントを使用して、ドメインにタグを追加し、ドメインにアタッチされたタグを一覧表示し、ドメインからタグを削除します。DOMAIN_ARN、TAG_KEY および TAG_VALUE の値を指定する必要があります。

```
import boto3
from botocore.config import Config  # import configuration

DOMAIN_ARN = ''  # ARN for the domain. i.e "arn:aws:es:us-east-1:123456789012:domain/
my-domain
TAG_KEY = ''  # The name of the tag key. i.e 'Smileyface'
TAG_VALUE = ''  # The value assigned to the tag. i.e 'Practicetag'

# defines the configurations parameters such as region

my_config = Config(region_name='us-east-1')
client = boto3.client('opensearch', config=my_config)


# defines the client variable

def addTags():
    """Adds tags to the domain"""

    response = client.add_tags(ARN=DOMAIN_ARN,
                               TagList=[{'Key': TAG_KEY,
                                         'Value': TAG_VALUE}])

    print(response)
```

```
def listTags():
    """List tags that have been added to the domain"""

    response = client.list_tags(ARN=DOMAIN_ARN)
    print(response)


def removeTags():
    """Remove tags that have been added to the domain"""

    response = client.remove_tags(ARN=DOMAIN_ARN, TagKeys=[TAG_KEY])

    print('Tag removed')
    return response
```

Amazon OpenSearch Service ドメインでの管理アクションの実行

Amazon OpenSearch Service では、ドメインの問題をトラブルシューティングする必要がある場合に、きめ細かな制御を可能にする複数の管理オプションが提供されています。これらのオプションには、データノードで OpenSearch プロセスを再起動する機能や、データノードを再起動する機能が含まれます。

OpenSearch Service はノードの状態パラメータを監視し、異常がある場合はドメインを安定させるための是正措置を講じます。ノード上の OpenSearch プロセスを再起動し、ノード自体を再起動する管理オプションを使用すると、これらの緩和アクションの一部を制御できます。

これらのアクションを実行するには AWS マネジメントコンソール、 AWS CLI、、または AWS SDK を使用できます。以下のセクションでは、コンソールでこれらのアクションを実行する方法について説明します。

Amazon OpenSearch Service のノード上で OpenSearch プロセスを再起動する

ノード上の OpenSearch プロセスを再起動するには

1. https://console.aws.amazon.com/aos/ で OpenSearch Service コンソール に移動します。
2. 左側のナビゲーションペインで [ドメイン] を選択します。作業するドメインの名前を選択します。
3. ドメインの詳細ページが開いたら、[インスタンスの正常性] タブに移動します。

4. [データノード] で、プロセスを再起動するノードの横にあるボタンを選択します。
5. [アクション] ドロップダウンを選択し、[OpenSearch/Elasticsearch プロセスを再起動] を選択します。
6. モーダルで [確認] を選択します。
7. 開始したアクションのステータスを確認するには、ノードの名前を選択します。ノードの詳細ページが開いたら、ノード名の下にある [イベント] タブを選択して、そのノードに関連するイベントのリストが表示します。

Amazon OpenSearch Service でのデータノードの再起動

データノードを再起動するには

1. https://console.aws.amazon.com/aos/ で OpenSearch Service コンソール に移動します。
2. 左側のナビゲーションペインで [ドメイン] を選択します。作業するドメインの名前を選択します。
3. ドメインの詳細ページが開いたら、[インスタンスの正常性] タブに移動します。
4. [データノード] で、プロセスを再起動するノードの横にあるボタンを選択します。
5. [アクション] ドロップダウンを選択し、[ノードの再起動] を選択します。
6. モーダルで [確認] を選択します。
7. 開始したアクションのステータスを確認するには、ノードの名前を選択します。ノードの詳細ページが開いたら、ノード名の下にある [イベント] タブを選択して、そのノードに関連するイベントのリストが表示します。

Amazon OpenSearch Service のノードにおける OpenSearch Dashboards プロセスの再起動

OpenSearch Dashboards (旧 Kibana) プロセスを再起動することで、インターフェイスのフリーズ、ロードエラー、応答しない視覚化などの問題から復旧できます。OpenSearch Dashboards を再起動するオプションは、Dashboards プロセスをアクティブに実行しているノードでのみ使用できます。ほとんどの OpenSearch Service ドメインでは、このプロセスはデータノードではなく専用コーディネーターノードで実行されます。その結果、コンソールで [アクション] ドロップダウンを開くと、通常、オプションはコーディネーターノードでのみ表示されます。詳細については、「the section called “専用コーディネーターノード”」を参照してください。

この動作は、ドメインの設定方法によって異なります。

- 専用コーディネーターノードを持つドメイン – Dashboards プロセスはそれらのノードでのみ実行され、再起動オプションが表示されるのはそれらのノードのみです。
- 専用コーディネーターノードのないドメイン – より単純な設定や古い設定では、代わりに Dashboards がデータノードで実行され、再起動オプションが表示されます。
- マスターノード – これらのノードは、クラスター状態と選択項目の管理専用に使用されます。Dashboards は実行されず、再起動オプションも表示されません。

Dashboards プロセスを実行しているノードを確認するには、ドメインの [クラスター設定] セクションに移動し、ノードロールを確認します。再起動するオプションは、Dashboards プロセスをホストするノードでのみ使用できます。

ノード上の Dashboard プロセスを再起動するには

1. https://console.aws.amazon.com/aos/ で OpenSearch Service コンソール に移動します。
2. 左側のナビゲーションペインで [ドメイン] を選択します。作業するドメインの名前を選択します。
3. ドメインの詳細ページが開いたら、[インスタンスの正常性] タブに移動します。
4. Dashboards プロセスを実行しているノードのセクションで、プロセスを再起動するノードの横にあるボタンを選択します。
5. [アクション] ドロップダウンを選択し、[Dashboard/Kibana プロセスを再起動] を選択します。
6. モーダルで [確認] を選択します。
7. 開始したアクションのステータスを確認するには、ノードの名前を選択します。ノードの詳細ページが開いたら、ノード名の下にある [イベント] タブを選択して、そのノードに関連するイベントのリストが表示します。

制限事項

管理オプションには以下の制限があります。

- 管理オプションは Elasticsearch バージョン 7.x 以降でサポートされています。
- 管理オプションは、スタンバイ付きマルチ AZ のドメインをサポートしていません。
- OpenSearch と Elasticsearch プロセスが再起動して、データノードの再起動が、3 つ以上のデータノードがあるドメインでサポートされています。
- Dashboards と Kibana のプロセスサポートは、2 つ以上のデータノードがあるドメインでサポートされます。

- ノードで OpenSearch プロセスを再起動する場合、またはノードを再起動する場合、ドメインが赤色の状態でないこと、およびすべてのインデックスにレプリカが設定されていることが必要です。

Amazon OpenSearch Service ダイレクトクエリの使用

Amazon OpenSearch Service ダイレクトクエリを使用して、取り込みパイプラインを構築せずに Amazon CloudWatch Logs、Amazon S3、Amazon Security Lake、Amazon Managed Service for Prometheus 内のデータを分析します。このゼロ ETL 統合により、PromQL、PPL、または SQL を使用して所定の場所にデータをクエリし、Discover で調べることができます。

Amazon Managed Service for Prometheus、CloudWatch Logs、または Security Lake の使用を開始するには、 AWS マネジメントコンソールでデータソースを設定します。Amazon S3 の場合は、ドメイン接続を使用し、Query Workbench で SQL を使用してテーブルを作成します。CloudWatch Logs と Security Lake は、事前設定されたデータソースとスキーマを使用します。Amazon S3 と Security Lake の場合、データは AWS Glue Data Catalog テーブルを使用してカタログ化されます。Amazon S3 ではこれらのテーブルを手動で作成する必要がありますが、Security Lake では取り込みプロセスの一部として自動的に設定されます。

OpenSearch Service における Amazon S3 データのダイレクトクエリ

このセクションでは、Amazon OpenSearch Service でデータソース統合を作成および設定するプロセスについて説明します。これにより、Amazon S3 に保存されているデータを効率的にクエリおよび分析できます。

以下のページでは、 AWS マネジメントコンソール と OpenSearch Service API の両方を使用して、Amazon S3 ダイレクトクエリデータソースをセットアップし、必要な前提条件を操作して、段階的な手順に従う方法について説明します。また、 AWS Glue Data Catalog ロールのマッピングや OpenSearch Dashboards でのアクセス制御の設定など、重要な次のステップについても説明します。

トピック

- OpenSearch Service における Amazon S3 データソース統合の作成
- OpenSearch Dashboards での S3 データソースの設定とクエリ
- 料金
- 制限事項
- 推奨事項
- クォータ

- サポートされている AWS リージョン

OpenSearch Service における Amazon S3 データソース統合の作成

OpenSearch Service 用の新しい Amazon S3 ダイレクトクエリデータソースは、 AWS マネジメントコンソール または API を使用して作成できます。各新しいデータソースは AWS Glue Data Catalog を使用して、Amazon S3 バケットを表すテーブルを管理します。

トピック

- 前提条件
- 手順
- 次の手順
- AWS Glue Data Catalog ロールをマッピングする
- その他のリソース

前提条件

作業を始める前に、以下の文書を確認したか確かめてください:

- the section called “制限事項”
- the section called “推奨事項”
- the section called “クォータ”

データソースを作成する前に、 に次のリソースが必要です AWS アカウント。

- バージョン 2.13 以降の OpenSearch ドメイン これは、ダイレクトクエリ統合を設定するための基盤です。これをセットアップする手順については、「the section called “OpenSearch Service ドメインの作成”」を参照してください。
- 1 つ以上の S3 バケット。クエリするデータを含むバケットと、クエリのチェックポイントを保存するバケットを指定する必要があります。S3 バケットの作成方法については、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。
- (オプション) 1 つ以上の AWS Glue テーブル。 Amazon S3 でデータをクエリするには、S3 データを指すように AWS Glue Data Catalog にテーブルを設定しておく必要があります。OpenSearch Query Workbench を使用してテーブルを作成する必要があります。既存の Hive テーブルには互換性がありません。

Amazon S3 データソースを初めて設定する場合は、管理者データソースを作成してすべての AWS Glue Data Catalog テーブルを設定する必要があります。これを行うには、すぐに使用できる OpenSearch 統合をインストールするか、OpenSearch Query Workbench を使用して高度なユースケース用のカスタム SQL テーブルを作成します。VPC、CloudTrail、および AWS WAF ログのテーブルを作成する例については、GitHub for VPC、CloudTrail、および のドキュメントを参照してくださいAWS WAF。テーブルを作成したら、新しい Amazon S3 データソースを作成し、アクセスを限定されたテーブルのみに制限できます。

- (オプション) 手動で作成された IAM ロール。このロールを使用して、データソースへのアクセスを管理できます。または、OpenSearch Service で必要なアクセス許可を持つロールを自動的に作成することも可能です。手動で作成した IAM ロールを使用する場合は、the section called “手動で作成された IAM ロールに必要なアクセス許可” にあるガイダンスに従ってください。

手順

AWS マネジメントコンソール または OpenSearch Service API を使用して、ドメインに直接クエリデータソースを設定できます。

を使用してデータソースを設定するには AWS マネジメントコンソール

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで [Domains] (ドメイン) を選択します。
3. 新しいデータソースを設定するドメインを選択します。選択すると、ドメインの詳細ページが開きます。
4. 一般的なドメインの詳細の下にある [接続] タブを選択し、[ダイレクトクエリ] セクションを見つけます。
5. [データソースの設定] を選択します。
6. 新しいデータソースの名前と適宜説明を入力します。
7. で Amazon S3 AWS Glue Data Catalogを選択します。
8. [IAM アクセス許可の設定] で、アクセスを管理する方法を選択します。

    a. このデータソースのロールを自動的に作成する場合は、次の手順に従います。

        i. [新しいロールを作成する] を選択します。

        ii. IAM ロールの名前を入力します。

iii. クエリするデータを含む S3 バケットを 1 つ以上選択します。

iv. クエリチェックポイントを保存するチェックポイント S3 バケットを選択します。

v. クエリできるデータを定義する AWS Glue データベースまたはテーブルを 1 つ以上選択します。テーブルがまだ作成されていない場合は、デフォルトのデータベースへのアクセスを提供します。

b. 自分で管理している既存のロールを使用する場合は、次の手順に従います。

i. [既存のロールを使用] を選択します。

ii. ドロップダウンメニューから既存のロールを選択します。

> **Note**
>
> 独自のロールを使用する場合は、IAM コンソールから必要なポリシーをアタッチして、すべての必要なアクセス許可があることを確認する必要があります。詳細については、「the section called “手動で作成された IAM ロールに必要なアクセス許可”」内のサンプルポリシーを参照してください。

9. [設定] を選択します。これにより、OpenSearch Dashboards URL を含むデータソースの詳細の画面が開きます。この URL に移動して、次のステップを完了できます。

OpenSearch Service API

AddDataSource API オペレーションを使用して、ドメインに新しいデータソースを作成します。

```
POST https://es.region.amazonaws.com/2021-01-01/opensearch/domain/domain-name/
dataSource

{
   "DataSourceType": {
        "S3GlueDataCatalog": {
            "RoleArn": "arn:aws:iam::account-id:role/role-name"
        }
    }
   "Description": "data-source-description",
   "Name": "my-data-source"
}
```

次の手順

OpenSearch Dashboards にアクセスする

データソースを作成すると、OpenSearch Service は OpenSearch Dashboards のリンクを提供します。これを使用して、アクセス制御の設定、テーブルの定義、設定不要な統合のインストール、データのクエリを行うことができます。

詳細については、「the section called “S3 データソースの設定”」を参照してください。

AWS Glue Data Catalog ロールをマッピングする

データソースの作成後にきめ細かなアクセスコントロールを有効にしている場合は、直接クエリを実行するために、管理者以外のユーザーを AWS Glue Data Catalog アクセス権を持つ IAM ロールにマッピングする必要があります。IAM ロールにマッピングできるバックエンド `glue_access` ロールを手動で作成するには、次のステップを実行します。

> **Note**
>
> インデックスは、データソースに対するクエリに使用されます。特定のデータソースのリクエストインデックスの読み取りアクセス権を持つユーザーは、そのデータソースに対するすべてのクエリを読み取ることができます。結果インデックスの読み取りアクセス権を持つユーザーは、そのデータソースに対するすべてのクエリの結果を読み取ることができます。

1. OpenSearch Dashboards のメインメニューから、[セキュリティ]、[ロール]、[ロールを作成] の順に選択します。
2. ロールに glue_access という名前を付けます。
3. [クラスターの許可] で、`indices:data/write/bulk*`、`indices:data/read/scroll`、または `indices:data/read/scroll/clear` を選択します。
4. [インデックス] で、ロールアクセスをユーザーに付与する次のインデックスを入力します:
   - `.query_execution_request_<name of data source>`
   - `query_execution_result_<name of data source>`
   - `.async-query-scheduler`
   - `flint_*`
5. [インデックスの許可] で、`indices_all` を選択します。
6. [作成] を選択します。

7. [マッピングされたユーザー]、[マッピングの管理] を選択します。
8. [バックエンドロール] で、ドメインを呼び出す許可を必要とする AWS Glue ロールの ARN を追加します。

```
arn:aws:iam::account-id:role/role-name
```

9. マップを選択し、ロールがマッピングされたユーザーに表示されていることを確認します。

ロールのマッピングの詳細については、「the section called “ユーザーへのロールのマッピング”」を参照してください。

その他のリソース

手動で作成された IAM ロールに必要なアクセス許可

お使いのドメインのデータソースを作成するときは、データへのアクセスを管理する IAM ロールを選択します。これには 2 つのオプションがあります。

1. 新しい IAM ロールを自動で作成する
2. 手動で作成した既存の IAM ロールを使用する

手動で作成したロールを使用する場合は、そのロールに正しいアクセス許可をアタッチする必要があります。そのアクセス許可は、特定のデータソースへのアクセスを許可するものであり、OpenSearch Service がロールを引き継ぐことを許可する必要があります。これは、OpenSearch Service がデータに安全にアクセスして操作できるようにするために必要です。

次のサンプルポリシーは、データソースの作成と管理に必要な最小特権の許可を示しています。`s3:*` または `AdminstratorAccess` ポリシーなどのより広範な許可を持っている場合、これらの許可にはサンプルポリシーの最小特権が含まれます。

次のサンプルポリシーでは、*`placeholder text`* をユーザー自身の情報に置き換えます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"HttpActionsForOpenSearchDomain",
```

```
         "Effect":"Allow",
         "Action":"es:ESHttp*",
"Resource":"arn:aws:es:us-east-1:111122223333:domain/example.com/*"
      },
      {
         "Sid":"AmazonOpenSearchS3GlueDirectQueryReadAllS3Buckets",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:ListBucket"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceAccount":"111122223333"
            }
         },
         "Resource":"*"
      },
      {
         "Sid":"AmazonOpenSearchDirectQueryGlueCreateAccess",
         "Effect":"Allow",
         "Action":[
            "glue:CreateDatabase",
            "glue:CreatePartition",
            "glue:CreateTable",
            "glue:BatchCreatePartition"
         ],
         "Resource":"*"
      },
      {
         "Sid":"AmazonOpenSearchS3GlueDirectQueryModifyAllGlueResources",
         "Effect":"Allow",
         "Action":[
            "glue:DeleteDatabase",
            "glue:DeletePartition",
            "glue:DeleteTable",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:GetTable",
            "glue:GetTableVersions",
            "glue:GetTables",
```

```
            "glue:UpdateDatabase",
            "glue:UpdatePartition",
            "glue:UpdateTable",
            "glue:BatchGetPartition",
            "glue:BatchDeletePartition",
            "glue:BatchDeleteTable"
         ],
         "Resource":[
            "arn:aws:glue:us-east-1:111122223333:table/*",
            "arn:aws:glue:us-east-1:111122223333:database/*",
            "arn:aws:glue:us-east-1:111122223333:catalog",
            "arn:aws:es:us-east-1:111122223333:domain/domain_name"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceAccount":"111122223333"
            }
         }
      },
      {
         "Sid":"ReadAndWriteActionsForS3CheckpointBucket",
         "Effect":"Allow",
         "Action":[
            "s3:ListMultipartUploadParts",
            "s3:DeleteObject",
            "s3:GetObject",
            "s3:PutObject",
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceAccount":"111122223333"
            }
         },
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
      }
   ]
}
```

異なるアカウントで Amazon S3 バケットをサポートするには、Amazon S3 ポリシーに条件を含め、適切なアカウントを追加する必要があります。

次の例では、*placeholder text* をユーザー自身の情報に置き換えます。

```
"Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "{{accountId}}"
                }
```

また、ロールには、ターゲット ID を指定する次の信頼ポリシーが必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
       {
          "Effect":"Allow",
          "Principal":{
             "Service": "directquery.opensearchservice.amazonaws.com"
          },
          "Action":"sts:AssumeRole"
       }
     ]
}
```

ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成」を参照してください。

OpenSearch Service できめ細かなアクセスコントロールが有効になっている場合、データソースに対して新しい OpenSearch のきめ細かなアクセスコントロールロールが自動的に作成されます。新しいきめ細かなアクセスコントロールロールの名前は `AWS OpenSearchDirectQuery <name of data source>` になります。

デフォルトでは、ロールは直接クエリデータソースインデックスにのみアクセスできます。データソースへのアクセスを制限または許可するようにロールを設定できますが、このロールのアクセスは調整しないことをお勧めします。データソースを削除すると、このロールは削除されます。これにより、他のユーザーがロールにマッピングされている場合、そのユーザーのアクセスは削除されます。

OpenSearch Dashboards での S3 データソースの設定とクエリ

データソースを作成したので、セキュリティ設定の構成、Amazon S3 テーブルの定義、高速データインデックスの設定を行うことができます。このセクションでは、データをクエリする前に、OpenSearch Dashboards でのデータソースを使用するさまざまなユースケースについて説明します。

次のセクションを設定するには、まず OpenSearch Dashboards でデータソースに移動する必要があります。左側のナビゲーションの [管理] で、[データソース] を選択します。[データソースを管理] で、コンソールで作成したデータソースの名前を選択します。

Query Workbench を使用して Spark テーブルを作成する

OpenSearch Service から Amazon S3 へのダイレクトクエリは、 AWS Glue Data Catalog内の Spark テーブルを使用します。OpenSearch Dashboards を離れることなく、Query Workbench 内からテーブルを作成できます。

データソース内の既存のデータベースとテーブルを管理したり、直接クエリを使用する新しいテーブルを作成するには、左側のナビゲーションから [Query Workbench] を選択して、データソースのドロップダウンから Amazon S3 データソースを選択します。

Parquet 形式で S3 に保存されている VPC フローログのテーブルを設定するには、次のクエリを実行します。

```
CREATE TABLE
datasourcename.gluedatabasename.vpclogstable (version INT, account_id STRING,
 interface_id STRING,
srcaddr STRING, dstaddr STRING, srcport INT, dstport INT, protocol INT, packets
 BIGINT,
bytes BIGINT, start BIGINT, end BIGINT, action STRING, log_status STRING,
`aws-account-id` STRING, `aws-service` STRING, `aws-region` STRING, year STRING,
month STRING, day STRING, hour STRING)

USING parquet PARTITIONED BY (aws-account-id, aws-service, aws-region, year, month,
day, hour)

LOCATION "s3://accountnum-vpcflow/AWSLogs"
```

テーブルを作成した後、次のクエリを実行して、テーブルがダイレクトクエリと互換性があることを確認します:

```
MSCK REPAIR TABLE  datasourcename.databasename.vpclogstable
```

一般的な AWS ログタイプの統合をセットアップする

Amazon S3 に保存されている AWS ログタイプを OpenSearch Service と統合できます。OpenSearch Dashboards を使用して、 AWS Glue Data Catalog テーブル、保存されたクエリ、ダッシュボードを作成する統合をインストールします。これらの統合では、インデックス付きビューを使用してダッシュボードを最新の状態に維持します。

統合をインストールする手順については、OpenSearch ドキュメントの「Installing an integration asset」を参照してください。

統合を選択するときは、`S3 Glue` タグがあることを確認してください。

統合を設定するときは、接続タイプに S3 接続を指定します。次に、統合用のデータソース、データの Amazon S3 の場所、アクセラレーションインデックスを管理するチェックポイント、そしてユースケースに必要なアセットを選択します。

> **Note**
>
> チェックポイント用の S3 バケットに、チェックポイントの保存場所への書き込み権限があることを確認します。これらの権限がないと、統合のアクセラレーションは失敗します。

アクセス制御を設定する

データソースの詳細ページで、[アクセスコントロール] セクションを見つけて、[編集] を選択します。ドメインで詳細なアクセス制御が有効になっている場合は、[制限付き] を選択し、新しいデータソースへのアクセスを許可するロールを選択します。管理者のみがデータソースにアクセスできるようにする場合は、[管理者のみ] を選択することもできます。

> **Important**
>
> インデックスは、データソースに対するクエリに使用されます。特定のデータソースのリクエストインデックスの読み取りアクセス権を持つユーザーは、そのデータソースに対するすべてのクエリを読み取ることができます。結果インデックスの読み取りアクセス権を持つユーザーは、そのデータソースに対するすべてのクエリの結果を読み取ることができます。

OpenSearch Discover での S3 データのクエリ

テーブルを設定し、必要なオプションのクエリアクセラレーションを構成したら、データの分析を開始できます。データのクエリを実行するには、ドロップダウンメニューからデータソースを選択します。Amazon S3 と OpenSearch Dashboards を使用している場合は、Discover に移動し、データソース名を選択します。

スキップインデックスを使用している場合、またはインデックスを作成していない場合は、SQL または PPL を使用してデータをクエリできます。マテリアライズドビューまたはカバーインデックスを設定している場合は、インデックスが既に存在するため、Dashboard 全体で Dashboards Query Language (DQL) を使用できます。オブザーバビリティプラグインで PPL を使用したり、Query Workbench プラグインで SQL を使用したりすることもできます。現在、PPL と SQL をサポートしているのは、オブザーバビリティプラグインと Query Workbench プラグインのみです。OpenSearch Service API を使用したデータのクエリについては、「非同期 API ドキュメント」を参照してください。

> **Note**
>
> すべての SQL および PPL ステートメント、コマンド、関数がサポートされているわけではありません。サポートされているコマンドのリストについては、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。
> マテリアライズドビューまたはカバリングインデックスを作成した場合は、DQL を使用して、インデックス化済みのデータをクエリできます。

トラブルシューティング

結果が期待どおりに返されない場合があります。問題が発生した場合は、the section called “推奨事項” に従っているかを確認してください。

料金

Amazon OpenSearch Service は、Amazon S3 ダイレクトクエリに対して OpenSearch Compute Unit (OCU) の料金を提供します。 Amazon S3 直接クエリを実行すると、1 時間あたりの OCUs に対して料金が発生し、請求には DirectQuery OCU の使用タイプとして表示されます。また、データストレージには Amazon S3 とは別の料金が発生します。

ダイレクトクエリには、インタラクティブビュークエリとインデックス付きビュークエリの 2 つのタイプがあります。

- インタラクティブクエリは、データセレクタに入力し、Amazon S3 のデータに対して分析を実行するために使用されます。Discover から新しいクエリを実行すると、OpenSearch Service は最低 3 分間続く新しいセッションを開始します。OpenSearch Service は、後続のクエリが迅速に実行されるように、このセッションをアクティブに維持します。
- インデックス付きビュークエリは、コンピューティングを使用して OpenSearch Service のインデックス付きビューを維持します。これらのクエリは通常、名前付きインデックスに取り込むデータ量が一定でないため、処理に時間がかかります。Amazon S3 データソースの場合、インデックス化されたデータは、購入したインスタンスタイプに基づいてドメインに保存されます。

詳細については、「Amazon OpenSearch Service の料金」の「ダイレクトクエリとサーバーレス」セクションを参照してください。

制限事項

Amazon S3 の直接クエリには、次の制限が適用されます。

- S3 の直接クエリは、OpenSearch バージョン 2.13 以降を実行している OpenSearch Service ドメインでのみ使用でき、 へのアクセスが必要です AWS Glue Data Catalog。既存の AWS Glue Data Catalog テーブルは、OpenSearch Query Workbench で SQL を使用して再作成する必要があります。
- S3 のダイレクトクエリでは、Amazon S3 でチェックポイントバケットを指定する必要があります。このバケットは、最終更新時間や最新の取り込みデータなど、インデックス付きビューの状態を維持します。
- OpenSearch ドメイン と は同じ にある AWS Glue Data Catalog 必要があります AWS アカウント。S3 バケットは別のアカウントにあることができますが (IAM ポリシーに条件を追加する必要があります)、ドメイン AWS リージョン と同じ にある必要があります。
- S3 での OpenSearch Service のダイレクトクエリは、Query Workbench から生成される Spark テーブルのみをサポートします。 AWS Glue Data Catalog または Athena 内で生成されたテーブルは、インデックス付きビューを維持するために必要な Spark ストリーミングではサポートされていません。
- OpenSearch インスタンスタイプには、選択した特定のインスタンスタイプに応じて、10 MiB または 100 MiB のネットワークペイロード制限があります。
- 一部のデータ型はサポートされていません。サポートされるデータ型は、Parquet、CSV、および JSON に限定されます。
- データの構造が時間の経過とともに変化する場合は、インデックス付きビューまたは追加設定なしの統合機能を、データの構造の変化に対応するように更新する必要があります。

- AWS CloudFormation テンプレートはまだサポートされていません。
- OpenSearch SQL ステートメントと OpenSearch PPL ステートメントは、OpenSearch インデックスを使用する場合とダイレクトクエリを使用する場合とでは制限が異なります。ダイレクトクエリでは、JOIN、サブクエリ、ルックアップなどの高度なコマンドがサポートされていますが、OpenSearch インデックスでのこれらのコマンドのサポートは制限されているか、存在していません。詳細については、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

推奨事項

Amazon S3 で直接クエリを使用する場合は、次のことをお勧めします。

- 年、月、日、時間のパーティション形式を使用してデータを Amazon S3 に取り込み、クエリを高速化します。
- スキップインデックスを構築するときは、高いカーディナリティを持つフィールドにはブルームフィルターを使用し、値の範囲が広いフィールドには最小/最大インデックスを使用します。カーディナリティの高いフィールドでは、値ベースのアプローチを使用してクエリ効率を向上させることを検討してください。
- インデックスステート管理を使用して、マテリアライズドビューとカバリングインデックスのストレージを維持します。
- 欠落している列を処理し、結果が返されるようにするには、`COALESCE` SQL 関数を使用します。
- クエリの制限を使用して、データが多すぎないようにします。

クォータ

Amazon S3 データソースにクエリを実行するたびに、OpenSearch Service はセッションを開き、少なくとも 3 分間セッションを維持します。これにより、後続のクエリにおけるセッション開始時間がなくなり、クエリのレイテンシーが短縮されます。

説明	最大値	上書き可能
ドメインあたりの接続数	10	[Yes (はい)]
ドメインあたりのデータソース	20	はい
ドメインあたりのインデックス	5	はい

説明	最大値	上書き可能
データソースあたりの同時実行セッション	10	[Yes (はい)]
クエリあたりの最大 OCU	60	はい
最大クエリ実行時間 (分)	30	はい
アクセラレーションあたりの最大 OCU	20	はい
最大エフェメラルストレージ	20	はい

サポートされている AWS リージョン

Amazon S3 の直接クエリでは、以下 AWS リージョン がサポートされています。

- アジアパシフィック (香港)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ストックホルム)
- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)

OpenSearch Service での Amazon CloudWatch Logs データのダイレクトクエリ

このセクションでは、Amazon OpenSearch Service でデータソース統合を作成および設定するプロセスについて説明します。これにより、CloudWatch Logs に保存されているデータを効率的にクエリおよび分析できます。

以下のページでは、CloudWatch Logs のダイレクトクエリ用データソースをセットアップ方法、必要な前提条件の確認方法、 AWS マネジメントコンソールを使用する手順をステップごとに説明します。

トピック

- OpenSearch Service での Amazon CloudWatch Logs データソース統合の作成
- OpenSearch Dashboards での CloudWatch Logs データソースの設定およびクエリ
- 料金
- 制限事項
- 推奨事項
- クォータ
- サポートされている AWS リージョン

OpenSearch Service での Amazon CloudWatch Logs データソース統合の作成

オブザーバビリティのニーズに合わせて Amazon OpenSearch Serverless を使用する場合、OpenSearch Service にデータをコピーまたは取り込まなくても Amazon CloudWatch Logs を分析できるようになりました。この機能は、OpenSearch Service から Amazon S3 のデータを分析するのと同様に、ダイレクトクエリを活用してデータをクエリします。まず、 AWS マネジメントコンソールから新しい接続されたデータソースを作成します。

CloudWatch Logs で運用ログをダイレクトクエリするために、Amazon OpenSearch Serverless を構築する必要はなく、CloudWatch Logs データを分析する新しいデータソースを作成できます。これにより、OpenSearch Service の外部にあるアクセス済みの運用データを分析できます。OpenSearch Service と CloudWatch Logs をまたいでクエリを実行することで、まずは CloudWatch Logs のログの分析を開始し、ツールを切り替えることなく OpenSearch のデータソースのモニタリングに戻ることができます。

この機能を使用するには、 AWS マネジメントコンソールを使用して OpenSearch Service の CloudWatch Logs ダイレクトクエリデータソースを作成します。

トピック

- 前提条件
- 手順
- 次の手順
- その他のリソース

前提条件

作業を始める前に、以下の文書を確認したか確かめてください:

- the section called “制限事項”
- the section called “推奨事項”
- the section called “クォータ”

データソースを作成する前に、 に次のリソースが必要です AWS アカウント。

- CloudWatch ログを有効化します。OpenSearch リソース AWS アカウント と同じ でログを収集するように CloudWatch Logs を設定します。 OpenSearch 手順については、「Amazon CloudWatch Logs ユーザーガイド」の「Getting started with CloudWatch Logs」を参照してください。
- 1 つ以上の CloudWatch ロググループ。クエリするデータを含むロググループを指定できます。CloudWatch Logs グループの作成方法については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs にロググループを作成する」を参照してください。
- (オプション) 手動で作成された IAM ロール。このロールを使用して、データソースへのアクセスを管理できます。または、OpenSearch Service で必要なアクセス許可を持つロールを自動的に作成することも可能です。手動で作成した IAM ロールを使用する場合は、the section called “手動で作成された IAM ロールに必要なアクセス許可” にあるガイダンスに従ってください。

手順

AWS マネジメントコンソールを使用してコレクションレベルのクエリデータソースを設定できます。

を使用してコレクションレベルのデータソースを設定するには AWS マネジメントコンソール

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで、[中央管理] に移動し、[接続されたデータソース] を選択します。
3. 接続 を選択します。
4. データソースタイプとして [CloudWatch] を選択します。
5. [次へ] を選択します。
6. [データ接続の詳細] で、名前とオプションの説明を入力します。
7. [IAM ロール] で、ロググループへのアクセスを管理する方法を選択します。
   a. このデータソースのロールを自動的に作成する場合は、次の手順に従います。
      i. [新しいロールを作成する] を選択します。
      ii. IAM ロールの名前を入力します。
      iii. 1 つ以上のロググループを選択して、クエリできるデータを定義します。
   b. 自分で管理している既存のロールを使用する場合は、次の手順に従います。
      i. [既存のロールを使用] を選択します。
      ii. ドロップダウンメニューから既存のロールを選択します。

   > **Note**
   > 独自のロールを使用する場合は、IAM コンソールから必要なポリシーをアタッチして、すべての必要なアクセス許可があることを確認する必要があります。詳細については、「the section called “手動で作成された IAM ロールに必要なアクセス許可”」を参照してください。

8. (オプション) アクセスポリシーで、データソースのアクセスポリシーを設定します。アクセスポリシーは、OpenSearch Service ダイレクトクエリデータソースへのリクエストを受け入れるか拒否するかを制御します。アクセスポリシーを設定しない場合、データソース所有者のみがアクセスできます。アクセスポリシーを設定してクロスアカウントアクセスを有効にし、他の のプリンシパル AWS アカウント がデータソースにアクセスできるようにします。

アクセスポリシーを作成するには、ビジュアルエディタを使用するか、JSON ポリシードキュメントを指定します。ビジュアルエディタでは、プリンシパル AWS アカウント ID、アカウント ARN、IAM ユーザー ARN、IAM ロール ARN、ソース IP アドレス、または CIDR ブロックを指定することで、アクセスを許可または拒否できます。ビジュアルエディタは、最大 10 個の要素をサポートします。10 個を超える要素を含むポリシーを定義するには、JSON エディタを使用します。

インポートポリシーを選択して、別のデータソースから既存のアクセスポリシーをインポートすることもできます。

9. (オプション) [タグ] で、データソースにタグを追加します。
10. [次へ] を選択します。
11. [OpenSearch のセットアップ] で、OpenSearch のセットアップ方法を選択します。

    a. デフォルトの設定を使用します。

    - デフォルトのリソース名とデータ保持設定を確認します。カスタム名を使用することをお勧めします。

      デフォルト設定を使用すると、新しい OpenSearch アプリケーションと専用の Essentials ワークスペースが追加料金なしで作成されます。OpenSearch を使用すると、複数のデータソースを分析できます。これにはワークスペースが含まれており、一般的なユースケースに合わせてカスタマイズされたエクスペリエンスを提供します。WorkSpaces はアクセス制御に対応しているため、ユースケースに応じたプライベートスペースを作成し、共同作業者とのみ共有することができます。

    b. カスタマイズされた設定を使用します:

    i. [カスタマイズ] を選択します。

    ii. 必要に応じて、コレクション名とデータ保持設定を編集します。

    iii. 使用する OpenSearch アプリケーションとワークスペースを選択します。

12. [次へ] を選択します。
13. 選択内容を確認し、変更を加える必要がある場合は [編集] を選択します。
14. [コネクタ] を選択して、データソースをセットアップします。データソースの作成中は、このページにとどまってください。準備ができたら、データソースの詳細ページに移動します。

次の手順

OpenSearch Dashboards にアクセスする

データソースを作成すると、OpenSearch Service は OpenSearch Dashboards URL を提供します。これを使用して、アクセス制御の設定、テーブルの定義、一般的なログタイプ用のログタイプベースのダッシュボードの設定、および SQL や PPL を使用したデータのクエリを実行できます。

詳細については、「the section called “CloudWatch Logs データソースの設定”」を参照してください。

その他のリソース

手動で作成された IAM ロールに必要なアクセス許可

データソースを作成するときは、データへのアクセスを管理する IAM ロールを選択します。これには 2 つのオプションがあります。

1. 新しい IAM ロールを自動で作成する
2. 手動で作成した既存の IAM ロールを使用する

手動で作成したロールを使用する場合は、そのロールに正しいアクセス許可をアタッチする必要があります。そのアクセス許可は、特定のデータソースへのアクセスを許可するものであり、OpenSearch Service がロールを引き継ぐことを許可する必要があります。これは、OpenSearch Service がデータに安全にアクセスして操作できるようにするために必要です。

次のサンプルポリシーは、データソースの作成と管理に必要な最小特権の許可を示しています。`logs:*` または `AdminstratorAccess` ポリシーなどのより広範な許可を持っている場合、これらの許可にはサンプルポリシーの最小特権が含まれます。

次のサンプルポリシーでは、*`placeholder text`* をユーザー自身の情報に置き換えます。

JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
```

```
            "Sid": "AmazonOpenSearchDirectQueryAllLogsAccess",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:StartQuery",
                "logs:GetLogGroupFields"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            },
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        }
    ]
}
```

また、ロールには、ターゲット ID を指定する次の信頼ポリシーが必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TrustPolicyForAmazonOpenSearchDirectQueryService",
            "Effect": "Allow",
            "Principal": {
                "Service": "directquery.opensearchservice.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:opensearch:us-
east-1:111122223333:datasource/rolename"
                }
            }
        }
    ]
```

```
}
```

ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成」を参照してください。

デフォルトでは、ロールは直接クエリデータソースインデックスにのみアクセスできます。データソースへのアクセスを制限または許可するようにロールを設定できますが、このロールのアクセスは調整しないことをお勧めします。データソースを削除すると、このロールは削除されます。これにより、他のユーザーがロールにマッピングされている場合、そのユーザーのアクセスは削除されます。

ダイレクトクエリデータソースのサンプルアクセスポリシー

直接クエリデータソースのアクセスポリシーは、IAM ポリシー構文に従います。ポリシードキュメントは有効な JSON 形式である必要があります。次のポリシー例では、直接クエリデータソースへの特定の AWS アカウント アクセスを許可します。

次のサンプルポリシーでは、*placeholder text* をユーザー自身の情報に置き換えます。

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
     "AWS": "arn:aws:iam::account-id:root"
     },
     "Action": [
       "opensearch:StartDirectQuery",
       "opensearch:GetDirectQuery",
       "opensearch:CancelDirectQuery",
       "opensearch:GetDirectQueryResult"
     ],
     "Resource": "arn:aws:opensearch:region:account-id:datasource/data-source-name"
   }
 ]
}
```

アクセスポリシーを設定しない場合、データソース所有者のみがデータソースにアクセスできます。

OpenSearch Dashboards での CloudWatch Logs データソースの設定およびクエリ

データソースを作成したら、OpenSearch Dashboards の使用を開始できます。このセクションでは、OpenSearch Dashboards でデータソースを使用するさまざまなユースケースについて説明します。

Discover ページからロググループをクエリする

OpenSearch Discover ページで、CloudWatch Logs ロググループのクエリ用に設定した新しいダイレクトクエリデータソースを使用できます。これを行うには、[Explore logs] を選択し、検索バーを使用して SQL または PPL を使用してクエリを構築します。ロググループから返されたデータをフィルタリング、ソート、視覚化できます。CloudWatch Logs 統合でサポートされているステートメント、コマンド、制限については、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

データソースのダッシュボードビューを作成する

OpenSearch Service を使用すると、構築済みのダッシュボードテンプレートを使用して、一般的な AWS ログタイプをすばやく分析できます。CloudWatch Logs には、VPC、CloudTrail、 AWS WAF、および Network Firewall ログ用のテンプレートがあります。これらのテンプレートを使用すると、特定のデータに合わせたダッシュボードをすばやく作成できます。これには、その特定のログタイプに合わせてカスタマイズされたダッシュボードが含まれます。これにより、すべてをゼロから構築することなく、これらの一般的な AWS ログソースの分析を迅速に開始して実行できます。

> **Note**
>
> ダッシュボードはインデックス付きビューを使用しており、インデックス付きビューは、ダイレクトクエリ用の OpenSearch Compute Units (OCU) とサーバーレスコレクションの indexingOCU、SearchOCU、ストレージを使用して CloudWatch Logs からデータを取り込みます。

これらの構築済みテンプレートのいずれかを使用してダッシュボードを作成するには、次のステップに従います。これにより、データの探索と分析をすぐに開始することができます。

ダッシュボードビューを作成するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインから、[中央管理]、[接続済みデータソース] を選択します。
3. データソースを選択して詳細ページを開きます。
4. [ダッシュボードを作成] を選択します。
5. 作成するダッシュボードのタイプを選択します。
6. ダッシュボードの名前を入力します。
7. ダッシュボードのオプションの説明を入力します。
8. ダッシュボードに表示するロググループを 1 つ以上選択します。
9. ダッシュボード内のデータを更新する頻度を選択します。
10. 使用する OpenSearch ワークスペースを選択します。
    a. 新しいワークスペースを作成するには、[新しいワークスペースを作成] を選択し、名前を入力します。
    b. 既存のワークスペースを使用するには、[既存のワークスペースを選択] を選択します。
11. [ダッシュボードを作成] を選択します。

OpenSearch Discover での CloudWatch Logs データのクエリ

データのクエリを実行するには、ドロップダウンメニューからデータソースを選択します。CloudWatch Logs を使用している場合は、Essentials ワークスペースから Discover に移動し、OpenSearch SQL または Piped Processing Language (PPL) を使用してデータのクエリを開始します。サポートされているコマンドのリストについては、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

> **Note**
>
> マテリアライズドビューが作成してある場合は、DQL を使用してインデックス化済みのデータをクエリできます。

トラブルシューティング

結果が期待どおりに返されない場合があります。問題が発生した場合は、the section called “推奨事項” に従っているかを確認してください。

料金

Amazon OpenSearch Service は、CloudWatch Logs の直接クエリに対して OpenSearch Compute Unit (OCU) の料金を提供します。直接クエリを実行すると、1 時間あたりの OCUs に対して料金が発生し、請求には DirectQuery OCU の使用タイプとして表示されます。また、Amazon CloudWatch Logs とは別の料金が発生します。

ダイレクトクエリには、インタラクティブビュークエリとインデックス付きビュークエリの 2 つのタイプがあります。

- インタラクティブクエリは、データセレクタに入力し、CloudWatch Logs のデータに対して分析を実行するために使用されます。OpenSearch Service は、拡張セッションを維持せずに、個別の事前ウォームされたジョブで各クエリを処理します。
- インデックス付きビュークエリは、コンピューティングを使用して OpenSearch Service のインデックス付きビューを維持します。これらのクエリは通常、名前付きインデックスに取り込むデータ量が一定でないため、処理に時間がかかります。CloudWatch Logs に接続されたデータソースの場合、インデックス化されたデータは OpenSearch Serverless コレクションに保存され、インデックス化されたデータ (IndexingOCU)、検索されたデータ (SearchOCU)、GB で保存されたデータに対して課金されます。

詳細については、「Amazon OpenSearch Service の料金」の「ダイレクトクエリとサーバーレス」セクションを参照してください。

制限事項

CloudWatch Logs の直接クエリには、次の制限が適用されます。

- CloudWatch Logs とのダイレクトクエリ統合は、OpenSearch Service コレクションと OpenSearch ユーザーインターフェイスでのみ使用できます。
- OpenSearch Serverless コレクションには、100 MiB のネットワークペイロード制限があります。
- CloudWatch Logs は、コンソールからインストールされた VPC フロー、CloudTrail、 AWS WAF ダッシュボードの統合をサポートします。

- AWS CloudFormation テンプレートはまだサポートされていません。
- OpenSearch SQL ステートメントと OpenSearch PPL ステートメントは、OpenSearch インデックスを使用する場合とダイレクトクエリを使用する場合とでは制限が異なります。ダイレクトクエリでは、JOIN、サブクエリ、ルックアップなどの高度なコマンドがサポートされていますが、OpenSearch インデックスでのこれらのコマンドのサポートは制限されているか、存在していません。詳細については、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

推奨事項

CloudWatch Logs で直接クエリを使用する場合は、次のことをお勧めします。

- 1 つのクエリで複数のロググループを検索する場合は、適切な構文を使用します。詳細については、「the section called “マルチロググループ関数”」を参照してください。
- SQL コマンドまたは PPL コマンドを使用する場合は、特定のフィールドをバッククオートで囲んでクエリを正常に実行します。特殊文字 (アルファベットや数字でないもの) を含むフィールドにはバッククオートが必要です。例えば、`@message`、`Operation.Export,`、`Test::Field` をバッククオートで囲みます。列名がアルファベットのみの場合はバッククオートで囲む必要はありません。

  シンプルなフィールドを含むクエリの例:

  ```
  SELECT SessionToken, Operation, StartTime  FROM `LogGroup-A`
  LIMIT 1000;
  ```

  バッククオートが付け足された類似のクエリ:

  ```
  SELECT `@SessionToken`, `@Operation`, `@StartTime` FROM `LogGroup-A`
  LIMIT 1000;
  ```

- クエリの制限を使用して、データが多すぎないようにします。
- フィールド名が同一であるが、大文字や小文字のみが異なる (`field1` および `FIELD1` など) クエリはサポートされていません。

  例として、以下のクエリはサポートされていません。

  ```
  Select AWSAccountId, AwsAccountId from LogGroup
  ```

```
Select a.@LogStream, b.@logStream from Table A INNER Join Table B ona.id = b.id
```

ただし、このフィールド名 (@logStream) は両方のロググループで同一であるため、次のクエリがサポートされています。

```
Select a.@logStream, b.@logStream from Table A INNER Join Table B on a.id = b.id
```

- 関数と式はフィールド名に対して操作し、FROM 句で指定されたロググループを持つ SELECT ステートメントの一部である必要があります。

  例として、次のクエリはサポートされていません。

  ```
  SELECT cos(10) FROM LogGroup
  ```

  次のクエリはサポートされています。

  ```
  SELECT cos(field1) FROM LogGroup
  ```

クォータ

> **Note**
>
> CloudWatch Logs Insights を使用してダイレクトクエリを実行する場合は、必ず「the section called “CloudWatch Logs Insights”」を参照してください。

説明	値	ソフト制限?	注意事項
ダイレクトクエリ API 全体のアカウントレベルの TPS 制限	3 TPS	はい	
データソースの最大数	20	はい	制限は ごとです AWS アカウント。

説明	値	ソフト制限?	注意事項
自動更新インデックスまたはマテリアライズドビューの最大数	30	はい	制限はデータソースごとに適用されます。
最大同時クエリ数	30	はい	制限はデータソースごとであり、保留中または実行中状態のクエリに適用されます。 インタラクティブクエリ (などのデータ取り出しコマンドなどSELECT) とインデックスクエリ (CREATE/ などのオペレーションなど) が含まれますALTER。
1 クエリあたりの最大同時 OCU	512	はい	OpenSearch コンピューティングユニット (OCU) 15 個のエグゼキューターと 1 個のドライバーに基づいて制限します。各ドライバーには 16 個の vCPU と 32 GB のメモリがあります。同時処理能力を表します。
最大クエリ実行時間 (分)	60	いいえ	制限は、CloudWatch Logs Insights の OpenSearch PPL/SQL クエリに適用されます。
古いクエリ ID を削除する期間	90 日間	はい	これは、OpenSearch Service が古いエントリのクエリメタデータを消去するまでの期間です。例えば、90 日以上経過したクエリでは、GetDirectQuery または GetDirectQueryResult の呼び出しは失敗します。

サポートされている AWS リージョン

CloudWatch Logs の直接クエリでは、以下 AWS リージョン がサポートされています。

- アジアパシフィック (ムンバイ)
- アジアパシフィック (香港)
- アジアパシフィック (大阪)

- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- 欧州 (スペイン)
- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- 米国西部 (北カリフォルニア)
- 欧州 (パリ)
- 欧州 (ロンドン)
- 南米 (サンパウロ)

OpenSearch Service での Amazon Security Lake データのダイレクトクエリ

このセクションでは、Amazon OpenSearch Service でデータソース統合を作成および設定するプロセスについて説明します。これにより、Security Lake に保存されているデータを効率的にクエリおよび分析できます。

以下のページでは、Security Lake のダイレクトクエリ用データソースをセットアップ方法、必要な前提条件の確認方法、 AWS マネジメントコンソールを使用する手順をステップごとに説明します。

トピック

- OpenSearch Service での Amazon Security Lake データソース統合の作成
- OpenSearch Dashboards での Security Lake データソースの設定とクエリ
- 料金

- 制限事項
- 推奨事項
- クォータ
- サポートされている AWS リージョン

OpenSearch Service での Amazon Security Lake データソース統合の作成

Amazon OpenSearch Serverless を使用して、Amazon Security Lake のセキュリティデータを直接クエリできます。これを行うには、Security Lake データに対して OpenSearch ゼロ ETL 機能を使用できるようにするデータソースを作成します。データソースを作成すると、Security Lake に保存されているデータを直接検索、インサイトの取得、分析できます。オンデマンドインデックス作成を使用することで、一部の Security Lake データセットに対するクエリのパフォーマンスを加速させ、高度な OpenSearch 分析を使用できます。

トピック

- 前提条件
- 手順
- 次の手順
- その他のリソース

前提条件

作業を始める前に、以下の文書を確認したか確かめてください:

- the section called “制限事項”
- the section called “推奨事項”
- the section called “クォータ”

データソースを作成する前に、Security Lake で次のアクションを実行します。

- Security Hub を有効にします。OpenSearch リソース AWS リージョン と同じ でログを収集するように Security Lake を設定します。 OpenSearch 手順については、「Amazon Security Lake ユーザーガイド」の「Amazon Security Lake の開始方法」を参照してください。
- Security Lake のアクセス許可を設定します。リソース管理のためにサービスにリンクされたロールのアクセス許可を受け入れ、コンソールで問題ページに問題が表示されていないことを確認

します。詳細については、「Amazon Security Lake ユーザーガイド」の「Service-linked role for Security Lake」を参照してください。

- Security Lake データソースを共有します。Security Lake と同じアカウント内で OpenSearch にアクセスする場合は、Security Lake コンソールに Lake Formation への Security Lake バケット登録を促すメッセージが表示されていないことを確認します。クロスアカウント OpenSearch アクセスの場合は、Security Lake コンソールで Lake Formation クエリサブスクライバーを設定します。OpenSearch リソースに関連付けられたアカウントをサブスクライバーとして使用します。詳細については、「Amazon Security Lake ユーザーガイド」の「Subscriber management in Security Lake」を参照してください。

さらに、 AWS アカウントには次のリソースも必要です。

- (オプション) 手動で作成された IAM ロール。このロールを使用して、データソースへのアクセスを管理できます。または、OpenSearch Service で必要なアクセス許可を持つロールを自動的に作成することも可能です。手動で作成した IAM ロールを使用する場合は、the section called “手動で作成された IAM ロールに必要なアクセス許可” にあるガイダンスに従ってください。

手順

データソースを設定して、 AWS マネジメントコンソール内から Security Lake データベースに接続できます。

を使用してデータソースを設定するには AWS マネジメントコンソール

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで、[中央管理] に移動し、[接続されたデータソース] を選択します。
3. 接続 を選択します。
4. データソースタイプとして [Security Lake] を選択します。
5. [次へ] を選択します。
6. [データ接続の詳細] で、名前とオプションの説明を入力します。
7. [IAM アクセス許可の設定] で、データソースへのアクセスを管理する方法を選択します。
    a. このデータソースのロールを自動的に作成する場合は、次の手順に従います。
        i. [新しいロールを作成する] を選択します。

ii. IAM ロールの名前を入力します。

iii. 1 つ以上の AWS Glue テーブルを選択して、クエリできるデータを定義します。

b. 自分で管理している既存のロールを使用する場合は、次の手順に従います。

i. [既存のロールを使用] を選択します。

ii. ドロップダウンメニューから既存のロールを選択します。

> **Note**
>
> 独自のロールを使用する場合は、IAM コンソールから必要なポリシーをアタッチして、すべての必要なアクセス許可があることを確認する必要があります。詳細については、「the section called “手動で作成された IAM ロールに必要なアクセス許可”」を参照してください。

8. (オプション) [タグ] で、データソースにタグを追加します。
9. [次へ] を選択します。
10. [OpenSearch のセットアップ] で、OpenSearch のセットアップ方法を選択します。

- デフォルトのリソース名とデータ保持設定を確認します。

  デフォルト設定を使用すると、新しい OpenSearch アプリケーションと専用の Essentials ワークスペースが追加料金なしで作成されます。OpenSearch を使用すると、複数のデータソースを分析できます。これにはワークスペースが含まれており、一般的なユースケースに合わせてカスタマイズされたエクスペリエンスを提供します。WorkSpaces はアクセス制御に対応しているため、ユースケースに応じたプライベートスペースを作成し、共同作業者とのみ共有することができます。

11. カスタマイズされた設定を使用します:

a. [カスタマイズ] を選択します。

b. 必要に応じて、コレクション名とデータ保持設定を編集します。

c. 使用する OpenSearch アプリケーションとワークスペースを選択します。

12. [次へ] を選択します。
13. 選択内容を確認し、変更を加える必要がある場合は [編集] を選択します。
14. [コネクタ] を選択して、データソースをセットアップします。データソースの作成中は、このページにとどまってください。準備ができたら、データソースの詳細ページに移動します。

次の手順

OpenSearch Dashboards にアクセスしてダッシュボードを作成する

データソースを作成すると、OpenSearch Service は OpenSearch Dashboards URL を提供します。これを使用することで、SQL または PPL を使用してデータをクエリします。Security Lake 統合には、ログの分析をすぐに始められるように SQL と PPL 用のパッケージ化されたクエリテンプレートが付属しています。

詳細については、「the section called “Security Lake データソースの設定”」を参照してください。

その他のリソース

手動で作成された IAM ロールに必要なアクセス許可

データソースを作成するときは、データへのアクセスを管理する IAM ロールを選択します。これには 2 つのオプションがあります。

1. 新しい IAM ロールを自動で作成する
2. 手動で作成した既存の IAM ロールを使用する

手動で作成したロールを使用する場合は、そのロールに正しいアクセス許可をアタッチする必要があります。アクセス許可は、特定のデータソースへのアクセスを許可するとともに、OpenSearch Service がロールを引き受けて、OpenSearch Service がデータに安全にアクセスして操作できるようにする必要があります。さらに、クエリを実行するデータベースとテーブルのロールに LakeFormation アクセス許可を付与します。ダイレクトクエリ接続からクエリを実行する SecurityLake データベースのロールに `DESCRIBE` のアクセス許可を付与します。データベース内のテーブルに対して、少なくとも `SELECT and DESCRIBE` のアクセス許可をデータソースロールに付与します。

次のサンプルポリシーは、データソースの作成と管理に必要な最小特権の許可を示しています。`AdminstratorAccess` ポリシーなどのより広範な許可を持っている場合、これらの許可にはサンプルポリシーの最小特権が含まれます。

次のサンプルポリシーでは、*`placeholder text`* をユーザー自身の情報に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AmazonOpenSearchDirectQueryServerlessAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll",
                "aoss:DashboardsAccessAll"
            ],
            "Resource": "arn:aws:aoss:us-
east-1:111122223333:collection/collectionname/*"
        },
        {
            "Sid": "AmazonOpenSearchDirectQueryGlueAccess",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:GetTable",
                "glue:GetTableVersions",
                "glue:GetTables",
                "glue:SearchTables",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:table/databasename/*",
                "arn:aws:glue:us-east-1:111122223333:database/databasename",
                "arn:aws:glue:us-east-1:111122223333:catalog",
                "arn:aws:glue:us-east-1:111122223333:database/default"
            ]
        },
        {
            "Sid": "AmazonOpenSearchDirectQueryLakeFormationAccess",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

また、ロールには、ターゲット ID を指定する次の信頼ポリシーが必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "directquery.opensearchservice.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成」を参照してください。

デフォルトでは、ロールは直接クエリデータソースインデックスにのみアクセスできます。データソースへのアクセスを制限または許可するようにロールを設定できますが、このロールのアクセスは調整しないことをお勧めします。データソースを削除すると、このロールは削除されます。これにより、他のユーザーがロールにマッピングされている場合、そのユーザーのアクセスは削除されます。

カスタマーマネージドキーで暗号化された Security Lake データのクエリ

データ接続に関連付けられた Security Lake バケットが、カスタマーマネージド AWS KMS keyによるサーバー側の暗号化を使用して暗号化されている場合は、LakeFormation サービスロールをキーポリシーに追加する必要があります。これにより、サービスはクエリ用にデータにアクセスして読み取ることができます。

次のサンプルポリシーでは、*placeholder text* をユーザー自身の情報に置き換えます。

```
{
    "Sid": "Allow LakeFormation to access the key",
    "Effect": "Allow",
```

```
    "Principal": {
        "AWS": "arn:aws:iam::account:role/aws-service-role/lakeformation.amazonaws.com/
AWSServiceRoleForLakeFormationDataAccess"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

OpenSearch Dashboards での Security Lake データソースの設定とクエリ

データソースを作成したら、OpenSearch Dashboards で設定できます。

> **Note**
>
> Discover の DirectQuery は、IAM 認証を使用して OpenSearch UI アプリケーションにログインする場合にのみサポートされます。IAM Identity Center (IAM Identity Center) は DirectQuery ではサポートされていません。DirectQuery Security Lake は Discover からのみ使用できるため、IAM Identity Center を使用しているお客様は DirectQuery Security Lake を使用できません。

このセクションでは、データをクエリする前に、OpenSearch Dashboards でのデータソースを使用するさまざまなユースケースについて説明します。開始するには、OpenSearch Dashboards でデータソースに移動する必要があります。左側のメニューの [管理] で、[データソース] を選択します。次に、OpenSearch Service コンソールで先ほど作成したデータソースの名前を選択します。

Discover から Security Lake テーブルをクエリする

Security Lake ログに基づいてテーブルを作成した場合は、OpenSearch Discover から直接それらのテーブルをクエリできるようになりました。これにより、使い慣れた Discover インターフェイスから直接、Security Lake に保存されているデータにシームレスにアクセスして分析できます。Discover から直接 Security Lake にクエリを実行することで、データを別の検索インデックスに手動で抽出、変換、ロードする必要がなくなります。ログの分析をすばやく始められるように、Discover には PPL および SQL の保存済みクエリが一式含まれています。

まずは、設定したデータソースを選択します。クエリを実行する関連データベースとテーブルを選択し、検索バーを使用してテーブルに対してクエリを書き込みます。Security Lake 統合でサポートされているステートメント、コマンド、制限については、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

Security Lake で使用できる構築済みのクエリを活用するには、Discover の右上にある [...] に移動し、[クエリを開く] を選択し、[テンプレート] を選択します。Security Lake でサポートされているログソースには、事前に構築されたクエリが多数あります。ユースケースに一致するテンプレートを検索し、検索バーで使用するクエリをコピーして、テンプレート化されたフィールド (リージョンやアクションなど) を独自の情報に置き換えます。

Discover からのデータの高速化

パフォーマンスを向上させ、OpenSearch で後続のクエリや分析を高速化するには、Discover からのクエリの結果を OpenSearch インデックス付きビューに取り込むことができます。

インデックス付きビューを作成するには

1. Discover から、[インデックス付きビューを作成] を選択します。
2. クエリエディタで、目的のクエリを入力します。ここで新しいクエリを作成するか、過去の検索から既存のクエリを使用できます。
3. 新しいインデックス付きビューの名前を指定します。後でビューを識別しやすくなるよう、わかりやすい名前を選択します。
4. インデックス付きビューのデータ保持設定を構成します。インデックスにデータを保持する期間を指定することで、パフォーマンスとストレージコストのバランスを取ることができます。
5. インデックス付きビューを作成します。作成後、インデックス付きビューを使用してクエリと分析を高速化できます。

インデックス付きビューを以前に作成してある場合は、Discover からアクセスできます。

既存のインデックスビューを使用するには

1. Discover から [インデックス付きビューを選択] を選択すると、Security Lake の既存のインデックス付きビューのリストが表示されます。
2. 使用するインデックス付きビューを選択します。これにより、現在のクエリにビューが適用され、データの取得と分析が大幅に高速化される可能性があります。

データソースのダッシュボードビューを作成する

OpenSearch Service を使用すると、構築済みのダッシュボードテンプレートを使用して一般的な AWS ログタイプを分析できます。Security Lake には、VPC、CloudTrail、および WAF のログ用テンプレートがあります。これらのテンプレートを使用すると、特定のデータに合わせたダッシュボードを作成できます。これには、その特定のログタイプに合わせてカスタマイズされた事前構築済みのクエリとダッシュボードが含まれます。これにより、すべてをゼロから構築することなく、これらの一般的な AWS ログソースの分析を迅速に開始して実行できます。

> **Note**
>
> ダッシュボードはインデックス付きビューを使用しており、Security Lake からデータを取り込んで、ダイレクトクエリやコレクションのコンピューティングに活用されます。

これらの構築済みテンプレートのいずれかを使用してダッシュボードを作成するには、次のステップに従います。これにより、データの探索と分析をすぐに開始することができます。

ダッシュボードビューを作成するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインから、[中央管理]、[接続済みデータソース] を選択します。
3. データソースを選択して詳細ページを開きます。
4. [ダッシュボードを作成] を選択します。
5. 作成するダッシュボードのタイプを選択します。
6. ダッシュボードの名前を入力します。
7. ダッシュボードのオプションの説明を入力します。
8. ダッシュボードに表示する AWS Glue テーブルを 1 つ以上選択します。
9. ダッシュボード内のデータを更新する頻度を選択します。
10. 使用する OpenSearch ワークスペースを選択します。
    a. [新しいワークスペースを作成] を選択して、新しいワークスペースを作成します。
    b. 既存のワークスペースを使用するには、[既存のワークスペースを選択] を選択します。
11. ワークスペースの名前を入力します。

12. [ダッシュボードを作成] を選択します。

トラブルシューティング

結果が期待どおりに返されない場合があります。問題が発生した場合は、the section called “推奨事項” に従っているかを確認してください。

料金

Amazon OpenSearch Service は、Security Lake ダイレクトクエリに対して OpenSearch Compute Unit (OCU) の料金を提供します。直接クエリを実行すると、1 時間あたりの OCUs に対して料金が発生し、請求には DirectQuery OCU の使用タイプとして表示されます。また、Amazon Security Lake とは別の料金が発生します。

ダイレクトクエリには、インタラクティブビュークエリとインデックス付きビュークエリの 2 つのタイプがあります。

- インタラクティブクエリは、データセレクタに入力し、Security Lake のデータに対して分析を実行するために使用されます。OpenSearch Service は、拡張セッションを維持せずに、個別の事前ウォームされたジョブで各クエリを処理します。
- インデックス付きビュークエリは、コンピューティングを使用して OpenSearch Service のインデックス付きビューを維持します。これらのクエリは通常、名前付きインデックスに取り込むデータ量が一定でないため、処理に時間がかかります。Security Lake に接続されたデータソースの場合、インデックス化されたデータは OpenSearch Serverless コレクションに保存され、インデックス化されたデータ (IndexingOCU)、検索されたデータ (SearchOCU)、GB で保存されたデータに対して課金されます。

詳細については、「Amazon OpenSearch Service の料金」の「ダイレクトクエリとサーバーレス」セクションを参照してください。

制限事項

Security Lake の直接クエリには、次の制限が適用されます。

- Security Lake とのダイレクトクエリ統合は、OpenSearch Service コレクションと OpenSearch ユーザーインターフェイスでのみ使用できます。
- OpenSearch Serverless コレクションには、100 MiB のネットワークペイロード制限があります。

- Security Lake のテーブル管理は Lake Formation で実行されます。
- Security Lake は、インデックス付きビューとしてマテリアライズドビューのみをサポートします。カバリングインデックスはサポートされていません。
- AWS CloudFormation テンプレートはまだサポートされていません。
- OpenSearch SQL ステートメントと OpenSearch PPL ステートメントは、OpenSearch インデックスを使用する場合とダイレクトクエリを使用する場合とでは制限が異なります。ダイレクトクエリでは、JOIN、サブクエリ、ルックアップなどの高度なコマンドがサポートされていますが、OpenSearch インデックスでのこれらのコマンドのサポートは制限されているか、存在していません。詳細については、「the section called “サポートされている SQL コマンドと PPL コマンド”」を参照してください。

推奨事項

Security Lake で直接クエリを使用する場合は、次のことをお勧めします。

- Security Lake のステータスを確認し、問題なくスムーズに実行されていることを確認します。トラブルシューティングの詳細な手順については、「Amazon Security Lake ユーザーガイド」の「Troubleshooting Lake Formation issues」を参照してください。
- クエリアクセスを確認します。
  - Security Lake の委任管理者アカウントとは異なるアカウントから Security Lake にクエリを実行する場合は、Security Lake でクエリアクセス権を持つサブスクライバーを設定します。
  - 同じアカウントから Security Lake にクエリを実行する場合は、マネージド S3 バケットを LakeFormation に登録する必要がある旨のメッセージが Security Lake に表示されていないかを確認します。
- クエリテンプレートと構築済みのダッシュボードを調べて、分析をすぐに開始します。
- Open Cybersecurity Schema Framework (OCSF) と Security Lake について説明します。
  - OCSF GitHub リポジトリ内の AWS ソースのスキーママッピングの例を確認する
  - AWS ソースバージョン 2 (OCSF 1.1.0) の Security Lake クエリにアクセスして、Security Lake を効果的にクエリする方法について説明します。
  - パーティションを使用してクエリのパフォーマンスを向上させる: `accountid`、`region`、および `time_dt`
- Security Lake がクエリ用にサポートしている SQL 構文を使いこなす。詳細については、「the section called “サポートされている SQL コマンド”」を参照してください。
- クエリの制限を使用して、データが多すぎないようにします。

クォータ

説明	値	ソフト制限?	注意事項
ダイレクトクエリ API 全体のアカウントレベルの TPS 制限	3 TPS	はい	
データソースの最大数	20	はい	制限は ごとです AWS アカウント。
自動更新インデックスまたはマテリアライズドビューの最大数	30	はい	制限はデータソースごとに適用されます。 自動更新が true に設定されているインデックスとマテリアライズドビュー (MV) のみが含まれます。
最大同時クエリ数	30	はい	制限は、保留中または実行中状態のクエリに適用されます。 インタラクティブクエリ (SELECT のようなデータ取り出しコマンドなど) とインデックスクエリ (CREATE/ALTER/DROP などのオペレーション) が含まれます。
1 クエリあたりの最大同時 OCU	512	はい	OpenSearch コンピューティングユニット (OCU) 15 個のエグゼキューターと 1 個のドライバーに基づいて制限します。各ドライバーには 16 個の vCPU と 32 GB のメモリがあります。同時処理能力を表します。
最大クエリ実行時間 (分)	30	いいえ	インタラクティブクエリ (SELECT のようなデータ取得コマンドなど) にのみ適用されます。REFRESH クエリの場合、制限は 6 時間です。
古いクエリ ID を削除する期間	90 日間	はい	これは、OpenSearch Service が古いエントリのクエリメタデータを消去するまでの期間です。例えば、90 日以上経過したクエリでは、GetDi

説明	値	ソフト制限?	注意事項
			rectQuery または GetDirectQueryResult の呼び出しは失敗します。

サポートされている AWS リージョン

Security Lake の直接クエリでは、以下 AWS リージョン がサポートされています。

- アジアパシフィック (ムンバイ)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ストックホルム)
- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- 欧州 (パリ)
- 欧州 (ロンドン)
- 南米 (サンパウロ)

Amazon Managed Service for Prometheus の直接クエリ

Amazon OpenSearch Service を使用して、Amazon Managed Service for Prometheus に保存されている運用メトリクスを直接クエリできます。この統合により、OpenSearch UI 内のログとトレースとともに Prometheus 時系列データを分析および視覚化し、統一されたオブザーバビリティエクスペリエンスを実現できます。

ストレージベースの直接クエリ (Amazon S3 や CloudWatch Logs など) とは異なり、Prometheus 統合はライブコールアーキテクチャを使用します。OpenSearch Service は直接クライアントとして機

能し、クエリを翻訳し、Prometheus ワークスペースにライブ API コールを行います。OpenSearch Service はデータをスキャンするための一時的なコンピューティングをプロビジョニングしないため、これらのクエリに対して OpenSearch Compute Unit (OCU) 料金は発生しません。

この統合を設定して使用するには、まずデータソースを作成し、データをクエリするようにワークスペースを設定する必要があります。

トピック

- Amazon Managed Service for Prometheus データソースの作成
- Prometheus メトリクスのクエリ
- 料金
- 制限事項
- 推奨事項
- クォータ
- サポートされる AWS リージョン

Amazon Managed Service for Prometheus データソースの作成

Amazon Managed Service for Prometheus データソースを作成するには、アクティブなワークスペースと、メトリクスをクエリするために必要なアクセス許可を OpenSearch Service に付与する IAM ロールが必要です。

前提条件

データソースを接続する前に、以下があることを確認してください。

- Prometheus ワークスペース – アクティブな Amazon Managed Service for Prometheus ワークスペース。WorkSpace ID とその場所を書き AWS リージョン 留めます。
- IAM ロール – `directquery.opensearchservice.amazonaws.com`サービスプリンシパルが引き受けることを許可する信頼ポリシーを持つ AWS Identity and Access Management ロール。

データソースの接続

前提条件が満たされたら、OpenSearch Service コンソールを使用してデータソースを接続できます。

Amazon Managed Service for Prometheus データソースを設定するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで、[中央管理] に移動し、[接続されたデータソース] を選択します。
3. 新しいデータソースの接続 を選択します。
4. データソースタイプとして Amazon Managed Service for Prometheus を選択します。
5. [次へ] を選択します。
6. [データ接続の詳細] で、名前とオプションの説明を入力します。
7. IAM ロールで、アクセスを管理する方法を選択します。
   - このデータソースのロールを自動的に作成するには:
     a. [新しいロールを作成する] を選択します。
     b. IAM ロールの名前を入力します。
     c. 1 つ以上のワークスペースを選択して、クエリできるデータを定義します。
   - 自分で管理する既存のロールを使用するには:
     a. [既存のロールを使用] を選択します。
     b. ドロップダウンメニューから既存のロールを選択します。

   > **Note**
   > 独自のロールを使用する場合は、IAM コンソールから必要なポリシーをアタッチして、必要なすべてのアクセス許可があることを確認してください。詳細については、「the section called “手動で作成された IAM ロールに必要なアクセス許可”」を参照してください。

8. (オプション) アクセスポリシーで、データソースのアクセスポリシーを設定します。アクセスポリシーは、OpenSearch Service ダイレクトクエリデータソースへのリクエストを受け入れるか拒否するかを制御します。アクセスポリシーを設定しない場合、データソース所有者のみがアクセスできます。アクセスポリシーを設定してクロスアカウントアクセスを有効にし、他の のプリンシパル AWS アカウント がデータソースにアクセスできるようにします。

   アクセスポリシーを作成するには、ビジュアルエディタを使用するか、JSON ポリシードキュメントを指定します。ビジュアルエディタでは、プリンシパル AWS アカウント ID、アカウント

ARN、IAM ユーザー ARN、IAM ロール ARN、ソース IP アドレス、または CIDR ブロックを指定することで、アクセスを許可または拒否できます。ビジュアルエディタは、最大 10 個の要素をサポートします。10 個を超える要素を含むポリシーを定義するには、JSON エディタを使用します。

インポートポリシーを選択して、別のデータソースから既存のアクセスポリシーをインポートすることもできます。

9. (オプション) [タグ] で、データソースにタグを追加します。
10. [次へ] を選択します。
11. OpenSearch のセットアップ で、OpenSearch UI のセットアップ方法を選択します。

    a. アカウントに OpenSearch UI アプリケーションが存在しない場合は、新しい OpenSearch アプリケーションを作成します。既存の OpenSearch アプリケーションが存在する場合は、それを選択します。

    b. 新しいアプリケーションを作成する場合は、新しいオブザーバビリティワークスペースを作成します。既存のアプリケーションを選択した場合は、新しいオブザーバビリティワークスペースを作成するか、既存のワークスペースを選択します。Amazon Managed Service for Prometheus は、オブザーバビリティワークスペースでのみ使用できます。

12. [次へ] を選択します。
13. 選択内容を確認し、変更を加える必要がある場合は [編集] を選択します。
14. [コネクタ] を選択して、データソースをセットアップします。データソースの作成中は、このページにとどまってください。準備ができたら、データソースの詳細ページに移動します。

次の手順

OpenSearch UI にアクセスする

データソースを作成すると、OpenSearch Service は OpenSearch UI アプリケーション URL を提供します。これを使用して、OpenSearch UI にアクセスできるユーザーを設定し、PromQL で Discover Metrics を使用して Amazon Managed Service for Prometheus データを分析します。

その他のリソース

手動で作成された IAM ロールに必要なアクセス許可

データソースを作成するときは、データへのアクセスを管理する IAM ロールを選択します。これには 2 つのオプションがあります。

- 新しい IAM ロールを自動で作成する
- 手動で作成した既存の IAM ロールを使用する

手動で作成したロールを使用する場合は、そのロールに正しいアクセス許可をアタッチする必要があります。アクセス許可は、特定のデータソースへのアクセスを許可し、OpenSearch Service がロールを引き受けることを許可する必要があります。これは、OpenSearch Service がデータに安全にアクセスして操作できるようにするために必要です。

次のサンプルポリシーは、データソースの作成と管理に必要な最小特権の許可を示しています。aps:* や AdministratorAccessポリシーなど、より広範なアクセス許可がある場合、これらのアクセス許可にはサンプルポリシーの最小特権のアクセス許可が含まれます。

次のサンプルポリシーでは、*placeholder text* をユーザー自身の情報に置き換えます。

サンプルの IAM ポリシー

次のアクセス許可を IAM ロールにアタッチして、OpenSearch Service がメトリクスメタデータを取得してクエリを実行できるようにします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonOpenSearchDirectQueryPrometheusAccess",
            "Effect": "Allow",
            "Action": [
                "aps:DeleteAlertManagerSilence",
                "aps:GetAlertManagerSilence",
                "aps:GetAlertManagerStatus",
                "aps:GetLabels",
                "aps:GetMetricMetadata",
                "aps:GetSeries",
                "aps:ListAlertManagerAlertGroups",
                "aps:ListAlertManagerAlerts",
                "aps:ListAlertManagerReceivers",
                "aps:ListAlertManagerSilences",
                "aps:ListAlerts",
                "aps:QueryMetrics",
                "aps:PutAlertManagerSilences",
                "aps:DescribeAlertManagerDefinition",
                "aps:CreateRuleGroupsNamespace",
                "aps:DeleteRuleGroupsNamespace",
```

```
                "aps:ListRuleGroupsNamespaces",
                "aps:DescribeRuleGroupsNamespace",
                "aps:PutRuleGroupsNamespace"
            ],
            "Resource": "arn:aws:aps:region:account-id:workspace/workspace-id"
        },
        {
            "Sid": "AmazonOpenSearchDirectQueryPrometheusListAccess",
            "Effect": "Allow",
            "Action": [
                "aps:ListWorkspaces"
            ],
            "Resource": "*"
        }
    ]
}
```

サンプル信頼ポリシー

IAM ロールに次の信頼ポリシーをアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TrustPolicyForAmazonOpenSearchDirectQueryService",
            "Effect": "Allow",
            "Principal": {
                "Service": "directquery.opensearchservice.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:opensearch:region:account-
id:datasource/data-source-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "account-id"
                }
            }
        }
    ]
}
```

Prometheus メトリクスのクエリ

Amazon OpenSearch Service では、オブザーバビリティインターフェイスから直接 PromQL (Prometheus クエリ言語) を使用して Prometheus データをクエリできます。Prometheus データソースに対して PromQL クエリを実行すると、OpenSearch Service はクエリを実行のためにワークスペース API に直接渡します。

PromQL クエリの実行

クエリを実行するには:

1. OpenSearch UI アプリケーションとオブザーバビリティワークスペースを開きます。
2. オブザーバビリティに移動し、メトリクスの検出を選択します。
3. データソースドロップダウンで、Prometheus データソースを選択します。
4. クエリバーに PromQL クエリを入力します。

たとえば、特定のポッドの 5 分間のウィンドウにおける 1 秒あたりの平均 CPU 使用率を確認するには、次のようにします。

```
avg(rate(container_cpu_usage_seconds_total{pod="payment-service-pod"}[5m])) by (pod)
```

> **Note**
>
> タイムピッカーを関連する狭いウィンドウ (過去 1 時間など) に設定して、API のパフォーマンスを最適化し、タイムアウトを防止します。

ダッシュボードでのメトリクスの視覚化

PromQL 駆動型メトリクスの視覚化を既存のオブザーバビリティダッシュボードに追加して、ログとトレースデータと関連付けることができます。

1. メトリクスの検出に移動し、データソースドロップダウンから Prometheus ワークスペースを選択し、PromQL クエリを実行します。
2. Discover Metrics のビジュアライゼーションタブを使用してビジュアライゼーションを作成し、ビジュアライゼーションタイプを定義します。
3. 視覚化をダッシュボードに保存します。

> **Note**
>
> メトリクスの視覚化は、Discover Metrics からのみ追加できます。ビジュアライゼーションタブにあるビジュアライゼーションは、ログに対してのみ最適化されます。

料金

Amazon Managed Service for Prometheus の場合、OpenSearch Service は Prometheus ワークスペースに直接ライブ呼び出しを行い、データを取得します。OpenSearch Service は、これらのクエリを実行するために個別のコンピューティングリソースをプロビジョニングしないため、OpenSearch Compute Unit (OCU) 料金は発生しません。Amazon Managed Service for Prometheus に関連する標準のクエリコストは、お客様の負担となります。

詳細については、「Amazon OpenSearch Service の料金」の「ダイレクトクエリとサーバーレス」セクションを参照してください。

制限事項

Amazon Managed Service for Prometheus の直接クエリには、次の制限が適用されます。

- 時間範囲の制約 — ライブクエリは運用ダッシュボード用に最適化されています。非常に長い期間 (複数か月にまたがるなど) にわたって非常に詳細でダウンサンプリングされていないメトリクスをクエリすると、ペイロードサイズ制限に達したり、Prometheus API からタイムアウトが発生する可能性があります。
- Prometheus API クォータ – クエリには、処理されたクエリサンプル (QSP) の制限や同時クエリ実行の制限など、標準の Amazon Managed Service for Prometheus サービスクォータが適用されます。
- クエリタイムアウト – 実行時間が長いクエリは 30 秒後にタイムアウトします。

推奨事項

Amazon Managed Service for Prometheus で直接クエリを使用する場合は、次のことをお勧めします。

- 長期間の記録ルールを使用する – ダイレクトクエリはワークスペースに対してライブ API コールを行うため、長期間 (複数か月など) にわたって非常に詳細なデータをクエリすると、API タイム

アウトやペイロード制限が発生する可能性があります。Amazon Managed Service for Prometheus の記録ルールを使用して、履歴分析用にダウンサンプリングされたメトリクスを作成します。

- 狭い時間フィルターを適用する – Prometheus API から動的に取得されるデータの量を最小限に抑えるために、OpenSearch UI クエリで常に目標時間範囲を指定します。
- Prometheus クォータのモニタリング – これらのライブクエリでは OpenSearch Compute Unit (OCU) 料金が発生しないため、代わりに Amazon Managed Service for Prometheus の使用状況をモニタリングします。ワークスペースレベルでスロットリングが発生しないように、クエリサンプル処理 (QSP) と同時クエリ制限をモニタリングします。

クォータ

他の直接クエリデータソースとは異なり、Amazon Managed Service for Prometheus へのクエリを開始するたびに、OpenSearch Service は Amazon Managed Service for Prometheus をライブ呼び出します。セッションは作成されません。アカウントとリージョンごとに、次のクォータが適用されます。たとえば、アカウントには us-east-1 に 20 個のデータソースが割り当てられ、us-east-2 に 20 個のデータソースが割り当てられます。

説明	最大値	上書き可能
データソース	20	はい
クエリの実行 – インスタントクエリと範囲クエリ (TPS)	500	はい
リソースの読み取り – ラベル、メトリクス、アラート、ルール、アラートマネージャー (TPS)	50	はい
リソースの書き込み – サイレンスの作成/更新、ルールの作成/更新 (TPS)	50	はい
最大クエリ実行時間 (秒)	30	いいえ

サポートされる AWS リージョン

Amazon Managed Service for Prometheus の直接クエリでは、以下 AWS リージョン がサポートされています。

- アジアパシフィック (香港)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- 南米 (サンパウロ)
- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)

Amazon OpenSearch Service のデータソースの管理

データソースの管理は、直接クエリデータソース AWS やその他のソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS には、モニタリング、問題発生時の報告、必要に応じて自動アクションを実行するための以下のツールが用意されています。

トピック

- CloudWatch メトリクスデータソースによるモニタリング
- データソースの有効化と無効化
- AWS Budget によるモニタリング
- データソースの削除

CloudWatch メトリクスデータソースによるモニタリング

CloudWatch を使用して直接クエリをモニタリングできます。CloudWatch は、raw データを収集して、それを読み取り可能なほぼリアルタイムのメトリクスに変換します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。

また、特定のしきい値をモニタリングするアラームを設定し、しきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch とは」を参照してください。

Amazon S3 は、次のメトリクスを報告します。

メトリクス	説明
AsyncQueryCreateAPI	非同期クエリを作成するために API に対して行われたリクエストの合計数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒
AsyncQueryGetApiRequestCount	非同期クエリ結果を取得するために API に対して行われたリクエストの合計数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒
AsyncQueryCancelApiRequestCount	非同期クエリをキャンセルするために API に対して行われたリクエストの合計数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒

メトリクス	説明
AsyncQueryGetApiFailedRequestCusErrCount	顧客関連のエラー (無効なクエリ ID など) によって非同期クエリ結果の取得時に失敗したリクエストの数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒
AsyncQueryCancelApiFailedRequestCusErrCount	顧客関連のエラー (無効なクエリ ID など) によって非同期クエリ結果の取得時に失敗したリクエストの数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒
AsyncQueryCancelApiFailedRequestSysErrCount	顧客関連のエラーによって非同期クエリの作成時に失敗したリクエストの数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒
AsyncQueryGetApiFailedRequestSysErrCount	システム関連のエラーによって非同期クエリ結果の取得時に失敗したリクエストの数。 関連する統計情報: Average、Maximum、Sum ディメンション: `ClientId`、`DomainName` 頻度: 60 秒

CloudWatch Logs と Security Lake は、次のメトリクスを報告します。

メトリクス	説明
DirectQueryRate	データソースに対して行われたリクエストレート。 関連する統計: Sum、Maximum、Minimum、Average ディメンション: `DataSourceName` 頻度: 60 秒
DirectQueryLatency	データソースでクエリを実行する際に観察されるレイテンシー。 関連する統計: Average、P90、P99、Sum、Minimum、Maximum ディメンション: `DataSourceName` 頻度: 60 秒
FailedDirectQueries	データソースのクエリ実行時に観測されたクエリ失敗の合計数。 関連する統計: Sum、Maximum、Minimum、Average ディメンション: `DataSourceName` 頻度: 60 秒
DirectQueryConsumedOCU	データソースでクエリを実行する際に消費される OCU の数。 関連する統計: Average、P90、P99、Sum、Minimum、Maximum ディメンション: `DataSourceName` 頻度: 60 秒

データソースの有効化と無効化

> **Note**
>
> 以下の情報は、Amazon S3 データソースにのみ適用されます。

データソースの直接クエリの使用を停止する場合は、データソースを無効にするように選択できます。データソースを無効にすると、既存のクエリの実行が終了し、新しいクエリを実行できなくなります。

データソースが無効になると、インデックスのスキップ、マテリアライズドビュー、カバリングインデックスなど、クエリパフォーマンスを向上させるためのアクセラレーション設定が手動に設定されます。データソースが無効になった後にアクティブに設定すると、ユーザークエリは期待どおりに実行されます。以前にセットアップされ、手動に設定されているクセラレーションは、スケジュールに従って再度実行するように手動で設定する必要があります。

AWS Budget によるモニタリング

Amazon OpenSearch Service は、アカウントレベルで OCU 使用状況データを Billing and Cost Management の Cost Explorer に入力しています。OCU 使用量はアカウントレベルで管理でき、しきい値を設定して、超過時にアラートを発することができます。

Cost Explorer でフィルタリングする使用タイプの形式は、RegionCode-DirectQueryOCU (OCU 時間単位) のようになります。DirectQueryOCU (OCU 時間単位) の使用量がしきい値に達したときに通知を受け取ることを希望する場合は、 AWS Budgets アカウントを作成し、設定したしきい値に基づいてアラートを設定できます。Amazon S3 に対しては、オプションで Amazon SNS トピックを設定することができ、しきい値の条件が満たされた場合にデータソースを無効化することができます。

> **Note**
>
> AWS Budgets の使用データはリアルタイムではなく、最大 8 時間遅延する可能性があります。

データソースの削除

データソースを削除すると、Amazon OpenSearch Service はそのデータソースをドメインまたはコレクションから削除します。また、OpenSearch Service は、データソースに関連付けられているイ

ンデックスも削除します。トランザクションデータは他方から削除されませんが AWS のサービス、他方 AWS のサービス は新しいデータを OpenSearch Service に送信しません。

データソース統合は、 AWS マネジメントコンソール または OpenSearch Service API を使用して削除できます。

AWS マネジメントコンソール

Amazon S3 データソースを削除するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインから [ドメイン] を選択します。
3. データソースを削除するドメインを選択します。選択すると、ドメインの詳細ページが開きます。全般情報の下にある [接続] タブを選択し、[ダイレクトクエリ] セクションを見つけます。
4. 削除するデータソースを選択し、[削除] を選択して、削除を確認します。

CloudWatch Logs または Security Lake データソースを削除するには

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインから、[中央管理]、[接続済みデータソース] を選択します。
3. 削除するデータソースを選択し、[削除] を選択して、削除を確認します。

OpenSearch Service API

Amazon S3 データソースを削除するには、DeleteDataSource API オペレーションを使用します。

```
POST https://es.region.amazonaws.com/2021-01-01/opensearch/domain/domain-name/
dataSource/data-source-name
```

CloudWatch Logs または Security Lake データソースを削除するには、DeleteDirectQueryDataSource API オペレーションを使用します。

サポートされている SQL コマンドと PPL コマンド

OpenSearch SQL および OpenSearch Pipeline Processing Language (PPL) は、OpenSearch、CloudWatch Logs Insights、および Security Lake でデータをクエリ、分析、

処理するための言語です。OpenSearch Discover で OpenSearch SQL と OpenSearch PPL を使用して、CloudWatch Logs、Amazon S3、または Security Lake 内のデータをクエリできます。CloudWatch Logs Insights では、OpenSearch CloudWatch Logs を分析するための専用のクエリ言語である Logs Insights QL に加えて、OpenSearch PPL と OpenSearch SQL の両方のクエリ言語もサポートされています。

- OpenSearch SQL: OpenSearch SQL は、リレーショナルデータベースの使用に慣れているユーザーにとっては使いやすいオプションを提供します。OpenSearch SQL は SQL 機能のサブセットを提供しており、アドホッククエリやデータ分析タスクの実行に適しています。OpenSearch SQL では、SELECT、FROM、WHERE、GROUP BY、HAVING などのコマンドに加え、SQL で利用可能なその他のさまざまな SQL コマンドや関数を使用できます。テーブル (またはロググループ) 間で JOINs を実行し、サブクエリを使用してテーブル (またはロググループ) 間でデータの関連性を分析し、さらに、JSON 関数、数学関数、文字列関数、条件関数など、豊富な SQL 関数のセットを使用して、ログとセキュリティデータに対して強力な分析を実行できます。
- OpenSearch PPL (Piped Processing Language): OpenSearch PPL を使用すると、パイプでつなげたコマンドを使用してデータを取得、クエリ、分析できるため、複雑なクエリも理解しやすく、構築しやすくなります。この構文は Unix パイプに基づいており、コマンドを連鎖してデータの変換と処理を行えます。PPL を使用すると、データをフィルタリングして集計し、JOIN、サブクエリ、LOOKUP などのコマンドに加え、数学、文字列、日付、条件付きなどの関数の豊富なセットを活用して分析を行うことができます。

OpenSearch PPL および OpenSearch SQL のクエリ言語に含まれるコマンドの多くは、CloudWatch Logs と OpenSearch で共通していますが、これらの各サービスでサポートされているコマンドと関数のセットにはいくつかの違いがあります。詳細については、以下のページの表を参照してください。

- the section called “サポートされている SQL コマンド”
  - the section called “CloudWatch Logs Insights”
  - the section called “SQL の一般的な制限”
- the section called “サポートされている PPL コマンド”
  - the section called “OpenSearch PPL を使用する CloudWatch Logs Insights ユーザー向けの追加情報”

サポートされている OpenSearch SQL コマンドと関数

次の参照表は、Amazon S3、Security Lake、または CloudWatch Logs でデータをクエリするために OpenSearch Discover でサポートされている SQL コマンドと、CloudWatch Logs Insights でサポートされている SQL コマンドを示しています。CloudWatch Logs Insights でサポートされる SQL 構文と CloudWatch Logs のクエリ用に OpenSearch Discover でサポートされる SQL 構文は同じであり、次の表では CloudWatch Logs として参照されます。

> **Note**
>
> OpenSearch には、OpenSearch に取り込まれ、インデックスに保存されたデータをクエリするための SQL サポートもあります。この SQL ダイアレクトは、ダイレクトクエリで使用される SQL とは異なり、インデックス上の OpenSearch SQL と呼ばれます。

トピック

- コマンド
- 関数
- SQL の一般的な制限
- OpenSearch SQL を使用する CloudWatch Logs Insights ユーザーの追加情報

コマンド

> **Note**
>
> コマンドの例の列では、クエリするデータソースに応じて、*<tableName/logGroup>* を適宜置き換えます。
>
> - コマンドの例: `SELECT Body , Operation FROM` **`<tableName/logGroup>`**
> - Amazon S3 または Security Lake をクエリする場合は、`SELECT Body , Operation FROM` **`table_name`** を使用します。
> - CloudWatch Logs をクエリする場合は、`SELECT Body , Operation FROM` **`` `LogGroupA` ``** を使用します。

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “SELECT 句”	予測値を表示します。	サポート対象	サポート対象	サポート対象	`SELECT method, status FROM <tableName/logGroup>`
the section called “WHERE 句”	指定されたフィールド基準に基づいてログイベントをフィルタリングします。	サポート対象	サポート対象	サポート対象	`SELECT * FROM <tableName/logGroup> WHERE status = 100`
the section called “GROUP BY 句”	グループはカテゴリに基づいてイベントを口グに記録し、統計に基づいて平均を見つけます。	サポート対象	サポート対象	サポート対象	`SELECT method, status, COUNT(*) AS request_count, SUM(bytes) AS total_bytes FROM <tableName/logGroup> GROUP BY method, status`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “HAVING 句”	グループ化条件に基づいて結果をフィルタリングします。	サポート対象	サポート対象	サポート対象	`SELECT method, status, COUNT(*) AS request_count, SUM(bytes) AS total_bytes FROM <tableName/logGroup> GROUP BY method, status HAVING COUNT(*) > 5`
the section called “ORDER BY 句”	order 句のフィールドに基づいて結果が並べ替えられます。昇順または降順のいずれかで並べ替えることができます。	サポート対象	サポート対象	サポート対象	`SELECT * FROM <tableName/logGroup> ORDER BY status DESC`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例

the section called “JOIN 句”
 	
(INNER | CROSS | LEFT OUTER)
	共通フィールドに基づいて 2 つのテーブルの結果を結合します。	対応 (結合には Inner および Left Outer キーワードを使用する必要があります。SELECT ステートメントでサポートされている JOIN 操作は 1 回のみです)	対応 (結合には Inner、Left Outer、Oss のキーワードを使用する必要があります)	対応 (結合には Inner、Left Outer、Oss のキーワードを使用する必要があります)	`SELECT`
					`  A.Body,`
					`  B.Timestamp`
					`FROM`
					`  <tableNameA/logGroupA>  AS A`
					`INNER JOIN`
					`  <tableNameB/logGroupB>  AS B`
					`  ON A.`requestId` = B.`requestId``

the section called “LIMIT 句”	結果を最初の N 行に制限します。	サポート対象	サポート対象	サポート対象	`SELECT`
					`  *`
					`FROM`
					`  <tableName/logGroup>`
					`LIMIT`
					`  10`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例

the section called “CASE 句”	条件を評価し、最初の条件が満たされたときに値を返します。	サポート対象	サポート対象	サポート対象	SELECT
					    method,
					    status,
					    CASE
					        WHEN status BETWEEN 100 AND 199 THEN 'Informational'
					        WHEN status BETWEEN 200 AND 299 THEN 'Success'
					        WHEN status BETWEEN 300 AND 399 THEN 'Redirection'
					        WHEN status BETWEEN 400 AND 499 THEN 'Client Error'
					        WHEN status BETWEEN 500 AND 599 THEN 'Server Error'
					        ELSE 'Unknown Status'
					    END AS status_category,
					    CASE method
					        WHEN 'GET' THEN 'Read Operation'
					        WHEN 'POST' THEN 'Create Operation'
					        WHEN 'PUT' THEN 'Update Operation'
					        WHEN 'PATCH' THEN 'Partial Update Operation'
					        WHEN 'DELETE' THEN 'Delete Operation'
					        ELSE 'Other Operation'
					    END AS operation_type,
					    bytes,
					    datetime
					FROM <tableName/logGroup>

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例

the section called “共通テーブル式”	SELECT、INSERT、UPDATE、DELETE、または MERGE ステートメント内に一時的な名前付き結果セットを作成します。	サポート対象外	サポート対象	サポート対象	`WITH RequestStats AS (`
					`  SELECT`
					`  method,`
					`  status,`
					`  bytes,`
					`  COUNT(*) AS request_count`
					`  FROM`
					`  tableName`
					`  GROUP BY`
					`  method,`
					`  status,`
					`  bytes`
					`)`
					`SELECT`
					`  method,`
					`  status,`
					`  bytes,`
					`  request_count`
					`FROM`
					`  RequestStats`
					`WHERE`
					`  bytes > 1000`

the section called “EXPLAIN”	実際に実行せずに SQL ステートメントの実行プランを表示します。	サポート対象外	サポート対象	サポート対象	`EXPLAIN`
					`SELECT`
					`  k,`
					`  SUM(v)`
					`FROM`
					`  VALUES`
					`  (1, 2),`
					`  (1, 3) AS t(k, v)`
					`GROUP BY`
					`  k`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “LATERAL SUBQUERY 句”	FROM 句のサブクエリが、同じ FROM 句内の前述の項目の列を参照できるようにします。	サポート対象外	サポート対象	サポート対象	`SELECT * FROM tableName LATERAL ( SELECT * FROM t2 WHERE t1.c1 = t2.c1 )`
the section called “LATERAL VIEW 句”	テーブル生成関数をベーステーブルの各行に適用して仮想テーブルを生成します。	サポート対象外	サポート対象	サポート対象	`SELECT * FROM tableName LATERAL VIEW EXPLODE(ARRAY(30, 60)) tableName AS c_age LATERAL VIEW EXPLODE(ARRAY(40, 80)) AS d_age`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “LIKE 述語”	ワイルドカードの文字を使用して文字列をパターンと照合します。	サポート対象	サポート対象	サポート対象	`SELECT method, status, request, host FROM <tableName/logGroup> WHERE method LIKE 'D%'`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例

the section called “OFFSET”	クエリから行を返し始める前にスキップする行数を指定します。	クエリ内でLIMIT句と組み合わせて使用する場合にサポートされます。例えば、次のようになります。
		• サポート対象: SELECT * FROM Table LIMIT 100 OFFSET 10
		• サポート対象
			サポート対象	サポート対象	<pre>SELECT
					    method,
					    status,
					    bytes,
					    datetime
					FROM
					    <tableName/logGroup>
					ORDER BY
					    datetime
					OFFSET
					    10</pre>

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
		外: `SELEC * FROM Table OFFSE 10`			
the section called “PIVOT 句”	行を列に変換し、データを行ベースの形式から列ベースの形式にローテーションします。	サポート対象外	サポート対象	サポート対象	`SELECT * FROM ( SELECT method, status, bytes FROM <tableName/logGroup> ) AS SourceTable PIVOT ( SUM(bytes) FOR method IN ('GET', 'POST', 'PATCH', 'PUT', 'DELETE') ) AS PivotTable`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “セット演算子”	2 つ以上の SELECT ステートメント (UNION、TERSEC XCEPT など) の結果を結合します。	サポート対象	サポート対象	サポート対象	`SELECT method, status, bytes FROM <tableName/logGroup> WHERE status = '416' UNION SELECT method, status, bytes FROM <tableName/logGroup> WHERE bytes > 20000`
the section called “SORT BY 句”	クエリ結果を返す順序を指定します。	サポート対象	サポート対象	サポート対象	`SELECT method, status, bytes FROM <tableName/logGroup> SORT BY bytes DESC`

コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “UNPIVOT”	列を行に変換し、列ベースの形式から行ベースの形式にデータをローテーションします。	サポート対象外	サポート対象	サポート対象	`SELECT status, REPLACE(method, '_bytes', '') AS request_method, bytes, datetime FROM PivotedData UNPIVOT ( bytes FOR method IN ( GET_bytes, POST_bytes, PATCH_bytes, PUT_bytes, DELETE_bytes ) ) AS UnpivotedData`

関数

> **Note**
>
> コマンドの例の列では、クエリするデータソースに応じて、*<tableName/logGroup>* を適宜置き換えます。
>
> - コマンドの例: `SELECT Body , Operation FROM` **`<tableName/logGroup>`**
> - Amazon S3 または Security Lake をクエリする場合は、`SELECT Body , Operation FROM` **`table_name`** を使用します。
> - CloudWatch Logs をクエリする場合は、`SELECT Body , Operation FROM` **`` `LogGroupA` ``** を使用します。

使用可能なSQL文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “文字列関数”	SQL クエリ内の文字列およびテキストデータを操作および変換できる組み込み関数。例えば、大文字と小文字の変換、文字列の組み合わせ、パーツの抽出、テキストのクリーニングなどが挙げられます。	サポート対象	サポート対象	サポート対象	`SELECT UPPER(method) AS upper_method, LOWER(host) AS lower_host FROM <tableName/logGroup>`
the section called “日付および時刻関数”	クエリ内で日付とタイムスタンプデータを処理および変換するための組み込み関数。例えば、date_add、at、	サポート対象	サポート対象	サポート対象	`SELECT TO_TIMESTAMP(datetime) AS timestamp, TIMESTAMP_SECONDS(UNIX_TIMESTAMP(datetime)) AS from_seconds, UNIX_TIMESTAMP(datetime) AS to_unix, FROM_UTC_TIMESTAMP(datetime, 'PST') AS to_pst, TO_UTC_TIMESTAMP(datetime, 'EST') AS from_est`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
	datediff、curre ate などが挙げられます。				FROM `<tableName/logGroup>`
the section called “集計関数”	複数の行で計算を実行して 1 つの要約値を生成する組み込み関数。例えば、sum、cou などが挙げられます。	サポートされている	サポート対象	サポート対象	SELECT COUNT(*) AS total_records, COUNT(DISTINCT method) AS unique_methods, SUM(bytes) AS total_bytes, AVG(bytes) AS avg_bytes, MIN(bytes) AS min_bytes, MAX(bytes) AS max_bytes FROM `<tableName/logGroup>`
the section called “条件関数”	指定された条件に基づいてアクションを実行するか、式を条件付きで評価する組み込み関数。例えば、CASE と IF が挙げられます。	サポート対象	サポート対象	サポート対象	SELECT CASE WHEN method = 'GET' AND bytes < 1000 THEN 'Small Read' WHEN method = 'POST' AND bytes > 10000 THEN 'Large Write' WHEN status >= 400 OR bytes = 0 THEN 'Problem' ELSE 'Normal' END AS request_type FROM `<tableName/logGroup>`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例

the section called “JSON 関数”	SQL クエリ内で JSON 形式のデータを解析、抽出、変更、クエリするための組み込み関数 (from_json、to_json、get_json_object、json_tuple など) により、データセット内の JSON 構造を操作できます。	サポート対象	サポート対象	サポート対象	<pre>SELECT
					    FROM_JSON(
					        @message,
					        'STRUCT<
					            host: STRING,
					            user-identifier: STRING,
					            datetime: STRING,
					            method: STRING,
					            status: INT,
					            bytes: INT
					        >'
					    ) AS parsed_json
					FROM
					    <tableName/logGroup></pre>

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “配列関数”	SQL クエリで配列型の列を操作するための組み込み関数により、配列データ へのアクセス、変更、分析などの操作を可能にします (例: size、explode、array_contains など)。	サポート対象	サポート対象	サポート対象	`SELECT scores, size(scores) AS length, array_contains(scores, 90) AS has_90 FROM <tableName/logGroup>`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “Window 関数”	現在の行 (ウィンドウ) に関連する指定された行セット間で計算を実行する組み込み関数により、ランク付け、実行合計、移動平均などの操作を可能にします (例: ROW_NUMBER、RANK、LAG、LEAD など)	サポート対象	サポート対象	サポート対象	`SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) AS field2Rank FROM <tableName/logGroup>`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “変換関数”	SQL クエリ内でデータをあるタイプから別のタイプに変換する組み込み関数により、データタイプの変換と形式変換を可能にします (CAST、TO_DATE、TO_TIMESTAMP、BINARY など)	サポート対象	サポート対象	サポート対象	`SELECT CAST('123' AS INT) AS converted_number, CAST(123 AS STRING) AS converted_string FROM <tableName/logGroup>`

使用可能なSQL文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “述語関数”	条件を評価し、指定された条件またはパターンに基づいてブール値 (true/false) を返す組み込み関数 (IN、LIKE、BETWEEN、IS NULL、EXISTS など)。	サポート対象	サポート対象	サポート対象	`SELECT * FROM <tableName/logGroup> WHERE id BETWEEN 50000 AND 75000`
the section called “マップ関数”	指定された関数をコレクション内の各要素に適用し、データを新しい値のセットに変換します。	サポート対象外	サポート対象	サポート対象	`SELECT MAP_FILTER( MAP( 'method', method, 'status', CAST(status AS STRING), 'bytes', CAST(bytes AS STRING) ), (k, v) -> k IN ('method', 'status') AND v != 'null' ) AS filtered_map FROM <tableName/logGroup> WHERE status = 100`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “数学関数”	平均、合計、三角値の計算など、数値データに対して数学演算を実行します。	サポート対象	サポート対象	サポート対象	`SELECT bytes, bytes + 1000 AS added, bytes - 1000 AS subtracted, bytes * 2 AS doubled, bytes / 1024 AS kilobytes, bytes % 1000 AS remainder FROM <tableName/logGroup>`
the section called “マルチロググループ関数”	ユーザーが SQL SELECT ステートメントで複数のロググループを指定できます	サポート対象	該当しない	該当しない	`SELECT lg1.Column1, lg1.Column2 FROM `logGroups(logGroupIdentifier: ['LogGroup1', 'LogGroup2'])` AS lg1 WHERE lg1.Column3 = "Success"`

使用可能な SQL 文法	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “ジェネレータ関数”	値のシーケンスを生成するイテレーターオブジェクトを作成し、大規模なデータセットに対して効率的なメモリ使用を可能にします。	サポート対象外	サポート対象	サポート対象	`SELECT explode(array(10, 20))`

SQL の一般的な制限

CloudWatch Logs、Amazon S3、および Security Lake で OpenSearch SQL を使用する場合、次の制限が適用されます。

1. SELECT ステートメントで使用できる JOIN 操作は 1 回のみです。
2. ネストされたサブクエリは 1 段階までしかサポートされていません。
3. セミコロンで区切られた複数のステートメントクエリはサポートされていません。
4. フィールド名が同一であるが、大文字や小文字のみが異なる (field1 および FIELD1 など) クエリはサポートされていません。

   例として、以下のクエリはサポートされていません。

   ```
   Select AWSAccountId, awsaccountid from LogGroup
   ```

ただし、次のクエリは、フィールド名 (@logStream) が両方のロググループが同じであるため、サポートされています。

```
Select a.`@logStream`, b.`@logStream` from Table A INNER Join Table B on a.id = b.id
```

5. 関数と式はフィールド名に対して操作を行う必要があり、FROM 句で指定されたロググループを持つ SELECT ステートメントの一部である必要があります。

   例として、次のクエリはサポートされていません。

   ```
   SELECT cos(10) FROM LogGroup
   ```

   次のクエリはサポートされています。

   ```
   SELECT cos(field1) FROM LogGroup
   ```

OpenSearch SQL を使用する CloudWatch Logs Insights ユーザーの追加情報

CloudWatch Logs は、Logs Insights コンソール、API、および CLI で OpenSearch SQL クエリをサポートします。SELECT、FROM、WHERE、GROUP BY、HAVING、JOINS、ネストされたクエリなどのほとんどのコマンドに加え、JSON 関数、数学関数、文字列関数、条件関数などをサポートしています。ただし、CloudWatch Logs は読み取り操作のみをサポートしているため、DDL または DML ステートメントは使用できません。サポートされているコマンドと関数の完全なリストについては、前のセクションの表を参照してください。

マルチロググループ関数

CloudWatch Logs Insights は、複数のロググループをクエリする機能をサポートしています。SQL でこのユースケースに対処するには、`logGroups` コマンドを使用できます。このコマンドは、1 つ以上のロググループに関連する CloudWatch Logs Insights 内のデータをクエリする際に特有のものです。この構文を使用すると、各ロググループごとにクエリを記述して `UNION` コマンドと組み合わせるのではなく、コマンド内で複数のロググループを指定して一括でクエリを実行できます。

構文:

```
`logGroups(
    logGroupIdentifier: ['LogGroup1','LogGroup2', ...'LogGroupn']
)
```

この構文では、logGroupIndentifier パラメータに最大 50 個のロググループを指定できます。モニタリングアカウントのロググループを参照するには、LogGroup の名前の代わりに ARN を使用します。

クエリの例:

```
SELECT LG1.Column1, LG1.Column2 from `logGroups(
    logGroupIdentifier: ['LogGroup1', 'LogGroup2']
)` as LG1
WHERE LG1.Column1 = 'ABC'
```

CloudWatch Logs をクエリする際は、FROM ステートメントの後に複数のロググループを指定する次の構文はサポートされていません。

```
SELECT Column1, Column2 FROM 'LogGroup1', 'LogGroup2', ...'LogGroupn'
WHERE Column1 = 'ABC'
```

制限事項

SQL コマンドまたは PPL コマンドを使用する場合は、特定のフィールドをバッククオートで囲み、クエリを実行します。特殊文字 (アルファベットや数字でないもの) を含むフィールドにはバッククオートが必要です。例えば、@message、Operation.Export,、Test::Field をバッククオートで囲みます。列名がアルファベットのみの場合はバッククオートで囲む必要はありません。

シンプルなフィールドを含むクエリの例:

```
SELECT SessionToken, Operation, StartTime  FROM `LogGroup-A`
LIMIT 1000;
```

バッククオートが付け足された同一のクエリ:

```
SELECT `SessionToken`, `Operation`, `StartTime`  FROM `LogGroup-A`
LIMIT 1000;
```

CloudWatch Logs に固有ではないその他の一般的な制限については、「the section called “SQL の一般的な制限”」を参照してください。

サンプルクエリとクォータ

> **Note**
>
> 以下は、CloudWatch Logs Insights ユーザーと CloudWatch データをクエリする OpenSearch ユーザーの両方に適用されます。

CloudWatch Logs で使用できるサンプル SQL クエリの例については、Amazon CloudWatch Logs Insights コンソールの「Saved and sample queries」を参照してください。

OpenSearch Service から CloudWatch Logs をクエリするときに適用される制限については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs quotas」を参照してください。制限には、クエリできる CloudWatch Log グループの数、同時実行できる最大クエリ数、最大クエリ実行時間、結果として返される最大行数が含まれます。CloudWatch Logs のクエリに使用する言語 (OpenSearch PPL、SQL、および Logs Insights) に関係なく、制限は同じです。

SQL コマンド

トピック

- 文字列関数
- 日付および時刻関数
- 集計関数
- 条件関数
- JSON 関数
- 配列関数
- Window 関数
- 変換関数
- 述語関数
- マップ関数
- 数学関数
- ジェネレータ関数
- SELECT 句
- WHERE 句
- GROUP BY 句

- HAVING 句
- ORDER BY 句
- JOIN 句
- LIMIT 句
- CASE 句
- 共通テーブル式
- EXPLAIN
- LATERAL SUBQUERY 句
- LATERAL VIEW 句
- LIKE 述語
- OFFSET
- PIVOT 句
- セット演算子
- SORT BY 句
- UNPIVOT

文字列関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
ascii(str)	`str` の最初の文字の数値を返します。
base64(bin)	引数をバイナリ `bin` からベース 64 文字列に変換します。
bit_length(expr)	文字列データの場合はビット長、バイナリデータの場合はビット数を返します。。

関数	説明
btrim(str)	`str` から先頭と末尾のスペース文字を削除します。
btrim(str, trimStr)	`str` から先頭と末尾の `trimStr` 文字を削除します。
char(expr)	`expr` と同等のバイナリを持つ ASCII 文字を返します。n が 256 より大きい場合、結果は chr(n % 256) と同等になります。
char_length(expr)	文字列データの場合は文字数、バイナリデータの場合はバイト数を返します。文字列データの長さには、末尾のスペースも含まれます。バイナリデータの長さには、バイナリゼロも含まれます。
character_length(expr)	文字列データの場合は文字数、バイナリデータの場合はバイト数を返します。文字列データの長さには、末尾のスペースも含まれます。バイナリデータの長さには、バイナリゼロも含まれます。
chr(expr)	`expr` と同等のバイナリを持つ ASCII 文字を返します。n が 256 より大きい場合、結果は chr(n % 256) と同等になります。
concat_ws(sep[, str | array(str)]+)	null 値をスキップして、`sep` で区切られた文字列の連結を返します。
contains(left, right)	ブール値を返します。left 中に right が見つかった場合、値は True になります。いずれかの入力式が NULL の場合、NULL を返します。それ以外の場合は、False を返します。left または right の両方が STRING または BINARY 型である必要があります。

関数	説明
decode(bin, charset)	2 番目の引数文字セットを使用して最初の引数をデコードします。
decode(expr, search, result [, search, result] ... [, default])	expr を各検索値と順番に比較します。expr が検索値と等しい場合、デコードは対応する結果を返します。一致が見つからない場合は、デフォルト値を返します。デフォルト値を省略すると、null が返されます。
elt(n, input1, input2, ...)	n 番目の入力を返します。例えば、n が 2 の場合は `input2` を返します。
encode(str, charset)	2 番目の引数文字セットを使用して最初の引数をエンコードします。
endswith(left, right)	ブール値を返します。left の末尾が right と一致する場合、値は True になります。いずれかの入力式が NULL の場合、NULL を返します。それ以外の場合は、False を返します。left または right の両方が STRING または BINARY 型である必要があります。
find_in_set(str, str_array)	カンマ区切りリスト (`str_array`) 内の指定された文字列 (`str`) のインデックス (1 ベース) を返します。文字列が見つからない場合、または指定された文字列 (`str`) にカンマが含まれている場合は、0 を返します。
format_number(expr1, expr2)	'#,###,###.##' のような数値 `expr1` をフォーマットし、`expr2` 小数点以下を四捨五入します。`expr2` が 0 の場合、結果には小数点または端数部分はありません。`expr2` はユーザー指定の形式も受け入れます。これは MySQL の FORMAT のように機能するはずです。

関数	説明
format_string(strfmt, obj, ...)	printf 形式の文字列からフォーマットされた文字列を返します。
initcap(str)	各単語の最初の文字を大文字にして str を返します。その他の文字はすべて小文字です。単語はスペースで区切られます。
instr(str, substr)	str で substr が最初に出現したときの (1 ベース) インデックスを返します。
lcase(str)	すべての文字を小文字に変更して str を返します。
left(str、len)	文字列 str から左端の len 文字 (len は文字列型でも可) を返します。len が 0 以下の場合、結果は空の文字列になります。
len(expr)	文字列データの場合は文字数、バイナリデータの場合はバイト数を返します。文字列データの長さには、末尾のスペースも含まれます。バイナリデータの長さには、バイナリゼロも含まれます。
length(expr)	文字列データの場合は文字数、バイナリデータの場合はバイト数を返します。文字列データの長さには、末尾のスペースも含まれます。バイナリデータの長さには、バイナリゼロも含まれます。
levenshtein(str1, str2[, threshold])	指定された 2 つの文字列間の Levenshtein 距離を返します。しきい値を設定し、それを超える距離がある場合は、-1 を返します。
locate(substr, str[, pos])	位置 pos の後で str の中に substr が最初に出現した位置を返します。指定された pos および戻り値は 1 ベースです。

関数	説明
lower(str)	すべての文字を小文字に変更して str を返します。
lpad(str, len[, pad])	str を pad で左詰めして、len の長さになるようして返します。str が len より長い場合、戻り値は len の文字数やバイト数まで短縮されます。pad が指定されていない場合、str は文字列の場合はスペース文字で左詰めされ、バイト列の場合はゼロで左詰めされます。
ltrim(str)	str から先頭のスペース文字を削除します。
luhn_check(str)	Luhn アルゴリズムに従って、数字の文字列が有効であることを確認します。このチェックサム関数は、クレジットカード番号や政府発行の身分証明番号などに広く利用されており、正しい番号と打ち間違いって不正確な番号を区別します。
mask(input[, upperChar, lowerChar, digitChar, otherChar])	指定された文字列値をマスクします。この関数は、文字を 'X' または 'x' に置き換え、数字を 'n' に置き換えます。これは、機密情報が削除されたテーブルのコピーを作成する際に役立ちます。
octet_length(expr)	文字列データのバイト長またはバイナリデータのバイト数を返します。
overlay(input, replace, pos[, len])	input を pos で始まり、長さが len である replace に置き換えます。
position(substr, str[, pos])	位置 pos の後で str の中に substr が最初に出現した位置を返します。指定された pos および戻り値は 1 ベースです。

関数	説明
printf(strfmt, obj, ...)	printf 形式の文字列からフォーマットされた文字列を返します。
regexp_count(str、regexp)	正規表現パターン `regexp` が文字列 `str` 内で一致した回数をカウントして返します。
regexp_extract(str, regexp[, idx])	`regexp` 式に一致し、正規表現グループインデックスに対応する `str` の最初の文字列を抽出します。
regexp_extract_all(str, regexp[, idx])	`regexp` 式に一致し、正規表現グループインデックスに対応する `str` 内のすべての文字列を抽出します。
regexp_instr(str, regexp)	文字列で正規表現を検索し、一致するサブ文字列の開始位置を示す整数を返します。位置は 1 ベースであり、0 ベースではありません。一致が見つからない場合は 0 を返します。
regexp_replace(str, regexp, rep[, position])	`str` の中で `regexp` に一致するすべてのサブ文字列を `rep` に置き換えます。
regexp_substr(str, regexp)	文字列 `str` 内の正規表現 `regexp` に一致する部分文字列を返します。正規表現が見つからない場合、結果は null になります。
repeat(str, n)	指定された文字列値を n 回繰り返す文字列を返します。
replace(str, search[, replace])	`search` のすべての出現を `replace` に置き換えます。
right(str, len)	文字列 `str` から右端の `len` 文字 (`len` は文字列型でも可) を返します。`len` が 0 以下の場合、結果は空の文字列になります。

関数	説明
rpad(str, len[, pad])	str を pad で右詰めして、len の長さになるようして返します。str が len より長い場合、戻り値は len の文字数まで短縮されます。pad が指定されていない場合、str は文字列の場合はスペース文字で右詰めされ、バイナリ列の場合はゼロで右詰めされます。
rtrim(str)	str から末尾のスペース文字を削除します。
sentences(str[, lang, country])	str を単語単位で分割した配列の配列に分割します。
soundex(str)	文字列の Soundex コードを返します。
space(n)	n スペースで構成される文字列を返します。
split(str, regex, limit)	文字列 str を regex と一致する箇所で分割し、長さが最大で limit の配列を返します。
split_part(str, delimiter, partNum)	str を区切り記号で分割し、分割されたうちの指定された部分 (1 ベース) を返します。いずれかの入力が null の場合、null を返します。partNum が分割部分の範囲外の場合、空の文字列を返します。partNum が 0 の場合、エラーをスローします。partNum が負の場合、文字列の末尾からその部分を数えます。delimiter が空の文字列の場合、str は分割されません。
startswith(left, right)	ブール値を返します。left の先頭が right と一致する場合、値は True になります。いずれかの入力式が NULL の場合、NULL を返します。それ以外の場合は、False を返します。left または right の両方が STRING または BINARY 型である必要があります。

関数	説明
substr(str, pos[, len])	pos で始まり、長さ len の str の部分文字列、または pos で始まり、長さ len のバイト配列のスライスを返します。
substr(str FROM pos[FOR len]])	pos で始まり、長さ len の str の部分文字列、または pos で始まり、長さ len のバイト配列のスライスを返します。
substring(str, pos[, len])	pos で始まり、長さ len の str の部分文字列、または pos で始まり、長さ len のバイト配列のスライスを返します。
substring(str FROM pos[FOR len]])	pos で始まり、長さ len の str の部分文字列、または pos で始まり、長さ len のバイト配列のスライスを返します。
substring_index(str, delim, count)	区切り文字 delim が count 回数出現する前の str のサブ文字列を返します。count が正の場合、最後の区切り文字の左にあるものがすべて返されます (左からカウント)。count が負の場合、最後の区切り文字の右側にあるすべて (右からカウント) が返されます。関数 substring_index は、delim を検索するときに大文字と小文字を区別する照合を行います。
to_binary(str[, fmt])	入力 str を指定された fmt に基づいてバイナリ値に変換します。fmt は、大文字小文字を区別しない文字列リテラルで、「hex」、「utf-8」、「utf8」、または「base64」のいずれかになります。デフォルトでは、fmt が省略されている場合、変換のバイナリ形式は「hex」です。入力パラメータの少なくとも 1 つが NULL の場合、関数は NULL を返します。

関数	説明
to_char(numberExpr, formatExpr)	formatExpr に基づいて numberExpr を文字列に変換します。いずれかの条件が失敗すると、例外をスローします。形式は、大文字と小文字を区別しない次の文字で構成できます。'0' または '9': 0~9 の予想される数字を指定します。形式文字列の 0 または 9 の並びは、入力値の数字列と一致し、形式文字列の対応する列と同じ長さの結果文字列を生成します。0/9 の並びが、小数点以下の値の対応部分よりも桁数が多く、0 で始まり、小数点より前にある場合、結果文字列は 0 で左側を埋めます。それ以外の場合は、スペースで埋められます。'.' または 'D': 小数点の位置を指定します (オプション、1 回のみ許可)。',' または 'G': グループ化 (1,000) 区切り記号 (,) の位置を指定します。各グループ化区切り文字の左右には 0 または 9 が必要です。 '

関数	説明
to_number(expr, fmt)	文字列 "expr' を文字列形式 'fmt' に基づく数値に変換します。いずれかの条件が失敗すると、例外をスローします。形式は、大文字と小文字を区別しない次の文字で構成できます。'0' または '9': 0～9 の予想される数字を指定します。形式文字列の 0 または 9 のシーケンスは、入力文字列の数字のシーケンスと一致します。0/9 シーケンスが 0 で始まり、小数点より前である場合は、同じサイズの数字列にしか一致しません。それ以外の場合、シーケンスが 9 で始まるか、小数点の後にある場合、じ桁数またはそれ以下の桁数の数字列と一致する可能性があります。'.' または 'D': 小数点の位置を指定します (オプション、1 回のみ指定可能)。',' または 'G': グループ区切り (千) 記号の位置を指定します (,)。各グループ区切り記号の左右には、0 または 9 を配置する必要があります。'expr'は、数字のサイズに適したグループ区切り記号と一致している必要があります。 '

関数	説明
to_varchar(numberExpr, formatExpr)	formatExpr に基づいて numberExpr を文字列に変換します。いずれかの条件が失敗すると、例外をスローします。形式は、大文字と小文字を区別しない次の文字で構成できます。'0' または '9': 0 ~ 9 の予想される数字を指定します。形式文字列の 0 または 9 の並びは、入力値の数字列と一致し、形式文字列の対応する列と同じ長さの結果文字列を生成します。0/9 の並びが、小数点以下の値の対応部分よりも桁数が多く、0 で始まり、小数点より前にある場合、結果文字列は 0 で左側を埋めます。それ以外の場合は、スペースで埋められます。'.' または 'D': 小数点の位置を指定します (オプション、1 回のみ許可)。',' または 'G': グループ化 (1,000) 区切り記号 (,) の位置を指定します。各グループ化区切り文字の左右には 0 または 9 が必要です。 '
translate(input, from, to)	from 文字列に存在する文字を to 文字列内の対応する文字に置き換えることで input 文字列を翻訳します。
trim(str)	str から先頭と末尾のスペース文字を削除します。
trim(BOTH FROM str)	str から先頭と末尾のスペース文字を削除します。
trim(LEADING FROM str)	str から先頭のスペース文字を削除します。
trim(TRAILING FROM str)	str から末尾のスペース文字を削除します。
trim(trimStr FROM str)	str から先頭と末尾の trimStr 文字を削除します。

関数	説明
trim(BOTH trimStr FROM str)	str から先頭と末尾の trimStr 文字を削除します。
trim(LEADING trimStr FROM str)	str から先頭の trimStr 文字を削除します。
trim(TRAILING trimStr FROM str)	str から末尾の trimStr 文字を削除します。
try_to_binary(str[, fmt])	これは、同じ操作を実行する to_binary の特別なバージョンですが、変換を実行できない場合に エラーを発生させるのではなく NULL 値を返します。
try_to_number(expr, fmt)	文字列 'expr' を文字列形式 fmt に基づく数値に変換します。文字列 'expr' が予想される形式と一致しない場合、NULL を返します。形式は to_number 関数と同じセマンティクスに従います。
ucase(str)	すべての文字を大文字に変更して str を返します。
unbase64(str)	引数をベース 64 文字列 str からバイナリに変換します。
upper(str)	すべての文字を大文字に変更して str を返します。

例

```
-- ascii
SELECT ascii('222');
+----------+
|ascii(222)|
+----------+
|        50|
+----------+
SELECT ascii(2);
```

```
+--------+
|ascii(2)|
+--------+
|      50|
+--------+
-- base64
SELECT base64('Feathers');
+-----------------+
|base64(Feathers)|
+-----------------+
|     RmVhdGhlcnM=|
+-----------------+
SELECT base64(x'537061726b2053514c');
+-----------------------------+
|base64(X'537061726B2053514C')|
+-----------------------------+
|                 U3BhcmsgU1FM|
+-----------------------------+
-- bit_length
SELECT bit_length('Feathers');
+---------------------+
|bit_length(Feathers)|
+---------------------+
|                   64|
+---------------------+
SELECT bit_length(x'537061726b2053514c');
+---------------------------------+
|bit_length(X'537061726B2053514C')|
+---------------------------------+
|                               72|
+---------------------------------+
-- btrim
SELECT btrim('    Feathers   ');
+----------------------+
|btrim(    Feathers   )|
+----------------------+
|              Feathers|
+----------------------+
SELECT btrim(encode('    Feathers   ', 'utf-8'));
+-----------------------------------+
|btrim(encode(    Feathers   , utf-8))|
+-----------------------------------+
|                           Feathers|
+-----------------------------------+
```

```
SELECT btrim('Feathers', 'Fe');
+--------------------+
|btrim(Alphabet, Al)|
+--------------------+
|               athers|
+--------------------+
SELECT btrim(encode('Feathers', 'utf-8'), encode('Al', 'utf-8'));
+---------------------------------------------------+
|btrim(encode(Feathers, utf-8), encode(Al, utf-8))|
+---------------------------------------------------+
|                                             athers|
+---------------------------------------------------+
-- char
SELECT char(65);
+--------+
|char(65)|
+--------+
|       A|
+--------+
-- char_length
SELECT char_length('Feathers ');
+-----------------------+
|char_length(Feathers )|
+-----------------------+
|                      9 |
+-----------------------+
SELECT char_length(x'537061726b2053514c');
+----------------------------------+
|char_length(X'537061726B2053514C')|
+----------------------------------+
|                                 9|
+----------------------------------+
SELECT CHAR_LENGTH('Feathers ');
+-----------------------+
|char_length(Feathers )|
+-----------------------+
|                      9|
+-----------------------+
SELECT CHARACTER_LENGTH('Feathers ');
+----------------------------+
|character_length(Feathers )|
+----------------------------+
|                           9|
+----------------------------+
```

```
-- character_length
SELECT character_length('Feathers ');
+----------------------------+
|character_length(Feathers )|
+----------------------------+
|                           9|
+----------------------------+
SELECT character_length(x'537061726b2053514c');
+---------------------------------------+
|character_length(X'537061726B2053514C')|
+---------------------------------------+
|                                      9|
+---------------------------------------+
SELECT CHAR_LENGTH('Feathers ');
+-----------------------+
|char_length(Feathers )|
+-----------------------+
|                      9|
+-----------------------+
SELECT CHARACTER_LENGTH('Feathers ');
+----------------------------+
|character_length(Feathers )|
+----------------------------+
|                           9|
+----------------------------+
-- chr
SELECT chr(65);
+-------+
|chr(65)|
+-------+
|      A|
+-------+
-- concat_ws
SELECT concat_ws(' ', 'Fea', 'thers');
+------------------------+
|concat_ws( , Fea, thers)|
+------------------------+
|               Feathers|
+------------------------+
SELECT concat_ws('s');
+------------+
|concat_ws(s)|
+------------+
|            |
```

```
+------------+
SELECT concat_ws('/', 'foo', null, 'bar');
+---------------------------+
|concat_ws(/, foo, NULL, bar)|
+---------------------------+
|                     foo/bar|
+---------------------------+
SELECT concat_ws(null, 'Fea', 'thers');
+--------------------------+
|concat_ws(NULL, Fea, thers)|
+--------------------------+
|                      NULL|
+--------------------------+
-- contains
SELECT contains('Feathers', 'Fea');
+-------------------------+
|contains(Feathers, Fea)|
+-------------------------+
|                     true|
+-------------------------+
SELECT contains('Feathers', 'SQL');
+-------------------------+
|contains(Feathers, SQL)|
+-------------------------+
|                     false|
+-------------------------+
SELECT contains('Feathers', null);
+-------------------------+
|contains(Feathers, NULL)|
+-------------------------+
|                     NULL|
+-------------------------+
SELECT contains(x'537061726b2053514c', x'537061726b');
+-----------------------------------------------+
|contains(X'537061726B2053514C', X'537061726B')|
+-----------------------------------------------+
|                                           true|
+-----------------------------------------------+
-- decode
SELECT decode(encode('abc', 'utf-8'), 'utf-8');
+---------------------------------+
|decode(encode(abc, utf-8), utf-8)|
+---------------------------------+
|                              abc|
```

```
+---------------------------------+
SELECT decode(2, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle',
 'Non domestic');
+-------------------------------------------------------------------------------+
|decode(2, 1, Southlake, 2, San Francisco, 3, New Jersey, 4, Seattle, Non domestic)|
+-------------------------------------------------------------------------------+
|                                                                  San Francisco|
+-------------------------------------------------------------------------------+
SELECT decode(6, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle',
 'Non domestic');
+-------------------------------------------------------------------------------+
|decode(6, 1, Southlake, 2, San Francisco, 3, New Jersey, 4, Seattle, Non domestic)|
+-------------------------------------------------------------------------------+
|                                                                   Non domestic|
+-------------------------------------------------------------------------------+
SELECT decode(6, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle');
+-----------------------------------------------------------------+
|decode(6, 1, Southlake, 2, San Francisco, 3, New Jersey, 4, Seattle)|
+-----------------------------------------------------------------+
|                                                             NULL|
+-----------------------------------------------------------------+
SELECT decode(null, 6, 'Fea', NULL, 'thers', 4, 'rock');
+----------------------------------------+
|decode(NULL, 6, Fea, NULL, thers, 4, rock)|
+----------------------------------------+
|                                   thers|
+----------------------------------------+
-- elt
SELECT elt(1, 'scala', 'java');
+------------------+
|elt(1, scala, java)|
+------------------+
|             scala|
+------------------+
SELECT elt(2, 'a', 1);
+------------+
|elt(2, a, 1)|
+------------+
|           1|
+------------+
-- encode
SELECT encode('abc', 'utf-8');
+-----------------+
|encode(abc, utf-8)|
```

```
+------------------+
|        [61 62 63]|
+------------------+
-- endswith
SELECT endswith('Feathers', 'ers');
+------------------------+
|endswith(Feathers, ers)|
+------------------------+
|                    true|
+------------------------+
SELECT endswith('Feathers', 'SQL');
+--------------------------+
|endswith(Feathers, SQL)|
+--------------------------+
|                     false|
+--------------------------+
SELECT endswith('Feathers', null);
+--------------------------+
|endswith(Feathers, NULL)|
+--------------------------+
|                     NULL|
+--------------------------+
SELECT endswith(x'537061726b2053514c', x'537061726b');
+-----------------------------------------------+
|endswith(X'537061726B2053514C', X'537061726B')|
+-----------------------------------------------+
|                                          false|
+-----------------------------------------------+
SELECT endswith(x'537061726b2053514c', x'53514c');
+---------------------------------------+
|endswith(X'537061726B2053514C', X'53514C')|
+---------------------------------------+
|                                   true|
+---------------------------------------+
-- find_in_set
SELECT find_in_set('ab','abc,b,ab,c,def');
+-------------------------------+
|find_in_set(ab, abc,b,ab,c,def)|
+-------------------------------+
|                              3|
+-------------------------------+
-- format_number
SELECT format_number(12332.123456, 4);
+-----------------------------+
```

```
|format_number(12332.123456, 4)|
+------------------------------+
|                   12,332.1235|
+------------------------------+
SELECT format_number(12332.123456, '##################.###');
+--------------------------------------------------+
|format_number(12332.123456, ##################.###)|
+--------------------------------------------------+
|                                         12332.123|
+--------------------------------------------------+
-- format_string
SELECT format_string("Hello World %d %s", 100, "days");
+-------------------------------------------+
|format_string(Hello World %d %s, 100, days)|
+-------------------------------------------+
|                       Hello World 100 days|
+-------------------------------------------+
-- initcap
SELECT initcap('Feathers');
+------------------+
|initcap(Feathers)|
+------------------+
|         Feathers|
+------------------+
-- instr
SELECT instr('Feathers', 'ers');
+--------------------+
|instr(Feathers, ers)|
+--------------------+
|                   6|
+--------------------+
-- lcase
SELECT lcase('Feathers');
+---------------+
|lcase(Feathers)|
+---------------+
|       feathers|
+---------------+
-- left
SELECT left('Feathers', 3);
+------------------+
|left(Feathers, 3)|
+------------------+
|               Fea|
```

```
+------------------+
SELECT left(encode('Feathers', 'utf-8'), 3);
+---------------------------------+
|left(encode(Feathers, utf-8), 3)|
+---------------------------------+
|                       [RmVh]|
+---------------------------------+
-- len
SELECT len('Feathers ');
+---------------+
|len(Feathers )|
+---------------+
|             9|
+---------------+
SELECT len(x'537061726b2053514c');
+--------------------------+
|len(X'537061726B2053514C')|
+--------------------------+
|                         9|
+--------------------------+
SELECT CHAR_LENGTH('Feathers ');
+-----------------------+
|char_length(Feathers )|
+-----------------------+
|                     9|
+-----------------------+
SELECT CHARACTER_LENGTH('Feathers ');
+----------------------------+
|character_length(Feathers )|
+----------------------------+
|                          9|
+----------------------------+
-- length
SELECT length('Feathers ');
+------------------+
|length(Feathers )|
+------------------+
|                9|
+------------------+
SELECT length(x'537061726b2053514c');
+-----------------------------+
|length(X'537061726B2053514C')|
+-----------------------------+
|                           9|
```

```
+----------------------------+
SELECT CHAR_LENGTH('Feathers ');
+-----------------------+
|char_length(Feathers )|
+-----------------------+
|                     9|
+-----------------------+
SELECT CHARACTER_LENGTH('Feathers ');
+----------------------------+
|character_length(Feathers )|
+----------------------------+
|                           9|
+----------------------------+
-- levenshtein
SELECT levenshtein('kitten', 'sitting');
+----------------------------+
|levenshtein(kitten, sitting)|
+----------------------------+
|                           3|
+----------------------------+
SELECT levenshtein('kitten', 'sitting', 2);
+-------------------------------+
|levenshtein(kitten, sitting, 2)|
+-------------------------------+
|                             -1|
+-------------------------------+
-- locate
SELECT locate('bar', 'foobarbar');
+-------------------------+
|locate(bar, foobarbar, 1)|
+-------------------------+
|                        4|
+-------------------------+
SELECT locate('bar', 'foobarbar', 5);
+-------------------------+
|locate(bar, foobarbar, 5)|
+-------------------------+
|                        7|
+-------------------------+
SELECT POSITION('bar' IN 'foobarbar');
+-------------------------+
|locate(bar, foobarbar, 1)|
+-------------------------+
|                        4|
```

```
+-------------------------+
-- lower
SELECT lower('Feathers');
+---------------+
|lower(Feathers)|
+---------------+
|       feathers|
+---------------+
-- lpad
SELECT lpad('hi', 5, '??');
+---------------+
|lpad(hi, 5, ??)|
+---------------+
|          ???hi|
+---------------+
SELECT lpad('hi', 1, '??');
+---------------+
|lpad(hi, 1, ??)|
+---------------+
|              h|
+---------------+
SELECT lpad('hi', 5);
+--------------+
|lpad(hi, 5,  )|
+--------------+
|            hi|
+--------------+
SELECT hex(lpad(unhex('aabb'), 5));
+--------------------------------+
|hex(lpad(unhex(aabb), 5, X'00'))|
+--------------------------------+
|                      000000AABB|
+--------------------------------+
SELECT hex(lpad(unhex('aabb'), 5, unhex('1122')));
+-------------------------------------+
|hex(lpad(unhex(aabb), 5, unhex(1122)))|
+-------------------------------------+
|                           112211AABB|
+-------------------------------------+
-- ltrim
SELECT ltrim('    Feathers   ');
+----------------------+
|ltrim(    Feathers   )|
+----------------------+
```

```
|           Feathers   |
+----------------------+
-- luhn_check
SELECT luhn_check('8112189876');
+----------------------+
|luhn_check(8112189876)|
+----------------------+
|                  true|
+----------------------+
SELECT luhn_check('79927398713');
+----------------------+
|luhn_check(79927398713)|
+----------------------+
|                   true|
+----------------------+
SELECT luhn_check('79927398714');
+----------------------+
|luhn_check(79927398714)|
+----------------------+
|                  false|
+----------------------+
-- mask
SELECT mask('abcd-EFGH-8765-4321');
+---------------------------------------+
|mask(abcd-EFGH-8765-4321, X, x, n, NULL)|
+---------------------------------------+
|                    xxxx-XXXX-nnnn-nnnn|
+---------------------------------------+
SELECT mask('abcd-EFGH-8765-4321', 'Q');
+---------------------------------------+
|mask(abcd-EFGH-8765-4321, Q, x, n, NULL)|
+---------------------------------------+
|                    xxxx-QQQQ-nnnn-nnnn|
+---------------------------------------+
SELECT mask('AbCD123-@$#', 'Q', 'q');
+--------------------------------+
|mask(AbCD123-@$#, Q, q, n, NULL)|
+--------------------------------+
|                    QqQQnnn-@$#|
+--------------------------------+
SELECT mask('AbCD123-@$#');
+--------------------------------+
|mask(AbCD123-@$#, X, x, n, NULL)|
+--------------------------------+
```

```
|                     XxXXnnn-@$#|
+--------------------------------+
SELECT mask('AbCD123-@$#', 'Q');
+--------------------------------+
|mask(AbCD123-@$#, Q, x, n, NULL)|
+--------------------------------+
|                     QxQQnnn-@$#|
+--------------------------------+
SELECT mask('AbCD123-@$#', 'Q', 'q');
+--------------------------------+
|mask(AbCD123-@$#, Q, q, n, NULL)|
+--------------------------------+
|                     QqQQnnn-@$#|
+--------------------------------+
SELECT mask('AbCD123-@$#', 'Q', 'q', 'd');
+--------------------------------+
|mask(AbCD123-@$#, Q, q, d, NULL)|
+--------------------------------+
|                     QqQQddd-@$#|
+--------------------------------+
SELECT mask('AbCD123-@$#', 'Q', 'q', 'd', 'o');
+-----------------------------+
|mask(AbCD123-@$#, Q, q, d, o)|
+-----------------------------+
|                  QqQQdddoooo|
+-----------------------------+
SELECT mask('AbCD123-@$#', NULL, 'q', 'd', 'o');
+--------------------------------+
|mask(AbCD123-@$#, NULL, q, d, o)|
+--------------------------------+
|                     AqCDdddoooo|
+--------------------------------+
SELECT mask('AbCD123-@$#', NULL, NULL, 'd', 'o');
+-----------------------------------+
|mask(AbCD123-@$#, NULL, NULL, d, o)|
+-----------------------------------+
|                        AbCDdddoooo|
+-----------------------------------+
SELECT mask('AbCD123-@$#', NULL, NULL, NULL, 'o');
+--------------------------------------+
|mask(AbCD123-@$#, NULL, NULL, NULL, o)|
+--------------------------------------+
|                           AbCD123oooo|
+--------------------------------------+
```

```
SELECT mask(NULL, NULL, NULL, NULL, 'o');
+------------------------------+
|mask(NULL, NULL, NULL, NULL, o)|
+------------------------------+
|                          NULL|
+------------------------------+
SELECT mask(NULL);
+-------------------------+
|mask(NULL, X, x, n, NULL)|
+-------------------------+
|                     NULL|
+-------------------------+
SELECT mask('AbCD123-@$#', NULL, NULL, NULL, NULL);
+----------------------------------------+
|mask(AbCD123-@$#, NULL, NULL, NULL, NULL)|
+----------------------------------------+
|                             AbCD123-@$#|
+----------------------------------------+
-- octet_length
SELECT octet_length('Feathers');
+-----------------------+
|octet_length(Feathers)|
+-----------------------+
|                      8|
+-----------------------+
SELECT octet_length(x'537061726b2053514c');
+-----------------------------------+
|octet_length(X'537061726B2053514C')|
+-----------------------------------+
|                                  9|
+-----------------------------------+
-- overlay
SELECT overlay('Feathers' PLACING '_' FROM 6);
+---------------------------+
|overlay(Feathers, _, 6, -1)|
+---------------------------+
|                  Feathe_ers|
+---------------------------+
SELECT overlay('Feathers' PLACING 'ures' FROM 5);
+-------------------------------+
|overlay(Feathers, ures, 5, -1)|
+-------------------------------+
|                     Features  |
+-------------------------------+
```

```
-- position
SELECT position('bar', 'foobarbar');
+---------------------------+
|position(bar, foobarbar, 1)|
+---------------------------+
|                          4|
+---------------------------+
SELECT position('bar', 'foobarbar', 5);
+---------------------------+
|position(bar, foobarbar, 5)|
+---------------------------+
|                          7|
+---------------------------+
SELECT POSITION('bar' IN 'foobarbar');
+-------------------------+
|locate(bar, foobarbar, 1)|
+-------------------------+
|                        4|
+-------------------------+
-- printf
SELECT printf("Hello World %d %s", 100, "days");
+-----------------------------------+
|printf(Hello World %d %s, 100, days)|
+-----------------------------------+
|                Hello World 100 days|
+-----------------------------------+
-- regexp_count
SELECT regexp_count('Steven Jones and Stephen Smith are the best players', 'Ste(v|
ph)en');
+-------------------------------------------------------------------------------+
|regexp_count(Steven Jones and Stephen Smith are the best players, Ste(v|ph)en)|
+-------------------------------------------------------------------------------+
|                                                                              2|
+-------------------------------------------------------------------------------+
SELECT regexp_count('abcdefghijklmnopqrstuvwxyz', '[a-z]{3}');
+-----------------------------------------------+
|regexp_count(abcdefghijklmnopqrstuvwxyz, [a-z]{3})|
+-----------------------------------------------+
|                                              8|
+-----------------------------------------------+
-- regexp_extract
SELECT regexp_extract('100-200', '(\\d+)-(\\d+)', 1);
+-------------------------------------+
|regexp_extract(100-200, (\d+)-(\d+), 1)|
```

```
+--------------------------------------+
|                                100|
+--------------------------------------+
-- regexp_extract_all
SELECT regexp_extract_all('100-200, 300-400', '(\\d+)-(\\d+)', 1);
+-----------------------------------------------------+
|regexp_extract_all(100-200, 300-400, (\d+)-(\d+), 1)|
+-----------------------------------------------------+
|                                          [100, 300]|
+-----------------------------------------------------+
-- regexp_instr
SELECT regexp_instr('user@opensearch.org', '@[^.]*');
+-----------------------------------------------+
|regexp_instr(user@opensearch.org, @[^.]*, 0)|
+-----------------------------------------------+
|                                              5|
+-----------------------------------------------+
-- regexp_replace
SELECT regexp_replace('100-200', '(\\d+)', 'num');
+--------------------------------------+
|regexp_replace(100-200, (\d+), num, 1)|
+--------------------------------------+
|                             num-num|
+--------------------------------------+
-- regexp_substr
SELECT regexp_substr('Steven Jones and Stephen Smith are the best players', 'Ste(v|
ph)en');
+-------------------------------------------------------------------------------+
|regexp_substr(Steven Jones and Stephen Smith are the best players, Ste(v|ph)en)|
+-------------------------------------------------------------------------------+
|                                                                         Steven|
+-------------------------------------------------------------------------------+
SELECT regexp_substr('Steven Jones and Stephen Smith are the best players', 'Jeck');
+-----------------------------------------------------------------------+
|regexp_substr(Steven Jones and Stephen Smith are the best players, Jeck)|
+-----------------------------------------------------------------------+
|                                                                   NULL|
+-----------------------------------------------------------------------+
-- repeat
SELECT repeat('123', 2);
+--------------+
|repeat(123, 2)|
+--------------+
|        123123|
```

```
+--------------+
-- replace
SELECT replace('ABCabc', 'abc', 'DEF');
+-------------------------+
|replace(ABCabc, abc, DEF)|
+-------------------------+
|                   ABCDEF|
+-------------------------+
-- right
SELECT right('Feathers', 3);
+-------------------+
|right(Feathers, 3)|
+-------------------+
|                ers|
+-------------------+
-- rpad
SELECT rpad('hi', 5, '??');
+---------------+
|rpad(hi, 5, ??)|
+---------------+
|          hi???|
+---------------+
SELECT rpad('hi', 1, '??');
+---------------+
|rpad(hi, 1, ??)|
+---------------+
|              h|
+---------------+
SELECT rpad('hi', 5);
+--------------+
|rpad(hi, 5,  )|
+--------------+
|         hi   |
+--------------+
SELECT hex(rpad(unhex('aabb'), 5));
+---------------------------------+
|hex(rpad(unhex(aabb), 5, X'00'))|
+---------------------------------+
|                     AABB000000|
+---------------------------------+
SELECT hex(rpad(unhex('aabb'), 5, unhex('1122')));
+--------------------------------------+
|hex(rpad(unhex(aabb), 5, unhex(1122)))|
+--------------------------------------+
```

```
|                            AABB112211|
+--------------------------------------+
-- rtrim
SELECT rtrim('    Feathers   ');
+----------------------+
|rtrim(    Feathers   )|
+----------------------+
|              Feathers|
+----------------------+
-- sentences
SELECT sentences('Hi there! Good morning.');
+--------------------------------------+
|sentences(Hi there! Good morning., , )|
+--------------------------------------+
|                  [[Hi, there], [Go...|
+--------------------------------------+
-- soundex
SELECT soundex('Miller');
+---------------+
|soundex(Miller)|
+---------------+
|           M460|
+---------------+
-- space
SELECT concat(space(2), '1');
+-------------------+
|concat(space(2), 1)|
+-------------------+
|                  1|
+-------------------+
-- split
SELECT split('oneAtwoBthreeC', '[ABC]');
+--------------------------------+
|split(oneAtwoBthreeC, [ABC], -1)|
+--------------------------------+
|             [one, two, three, ]|
+--------------------------------+
SELECT split('oneAtwoBthreeC', '[ABC]', -1);
+--------------------------------+
|split(oneAtwoBthreeC, [ABC], -1)|
+--------------------------------+
|             [one, two, three, ]|
+--------------------------------+
SELECT split('oneAtwoBthreeC', '[ABC]', 2);
```

```
+------------------------------+
|split(oneAtwoBthreeC, [ABC], 2)|
+------------------------------+
|              [one, twoBthreeC]|
+------------------------------+
-- split_part
SELECT split_part('11.12.13', '.', 3);
+-------------------------+
|split_part(11.12.13, ., 3)|
+-------------------------+
|                       13|
+-------------------------+
-- startswith
SELECT startswith('Feathers', 'Fea');
+---------------------------+
|startswith(Feathers, Fea)|
+---------------------------+
|                       true|
+---------------------------+
SELECT startswith('Feathers', 'SQL');
+-------------------------+
|startswith(Feathers, SQL)|
+-------------------------+
|                      false|
+-------------------------+
SELECT startswith('Feathers', null);
+-------------------------+
|startswith(Feathers, NULL)|
+-------------------------+
|                       NULL|
+-------------------------+
SELECT startswith(x'537061726b2053514c', x'537061726b');
+-------------------------------------------------+
|startswith(X'537061726B2053514C', X'537061726B')|
+-------------------------------------------------+
|                                             true|
+-------------------------------------------------+
SELECT startswith(x'537061726b2053514c', x'53514c');
+-----------------------------------------+
|startswith(X'537061726B2053514C', X'53514C')|
+-----------------------------------------+
|                                    false|
+-----------------------------------------+
-- substr
```

```
SELECT substr('Feathers', 5);
+------------------------------+
|substr(Feathers, 5, 2147483647)|
+------------------------------+
|                          hers |
+------------------------------+
SELECT substr('Feathers', -3);
+-------------------------------+
|substr(Feathers, -3, 2147483647)|
+-------------------------------+
|                            ers|
+-------------------------------+
SELECT substr('Feathers', 5, 1);
+----------------------+
|substr(Feathers, 5, 1)|
+----------------------+
|                     h|
+----------------------+
SELECT substr('Feathers' FROM 5);
+---------------------------------+
|substring(Feathers, 5, 2147483647)|
+---------------------------------+
|                            hers |
+---------------------------------+
SELECT substr('Feathers' FROM -3);
+----------------------------------+
|substring(Feathers, -3, 2147483647)|
+----------------------------------+
|                               ers|
+----------------------------------+
SELECT substr('Feathers' FROM 5 FOR 1);
+-------------------------+
|substring(Feathers, 5, 1)|
+-------------------------+
|                        h|
+-------------------------+
-- substring
SELECT substring('Feathers', 5);
+---------------------------------+
|substring(Feathers, 5, 2147483647)|
+---------------------------------+
|                            hers |
+---------------------------------+
SELECT substring('Feathers', -3);
```

```
+-----------------------------------+
|substring(Feathers, -3, 2147483647)|
+-----------------------------------+
|                                ers|
+-----------------------------------+
SELECT substring('Feathers', 5, 1);
+--------------------------+
|substring(Feathers, 5, 1)|
+--------------------------+
|                         h|
+--------------------------+
SELECT substring('Feathers' FROM 5);
+----------------------------------+
|substring(Feathers, 5, 2147483647)|
+----------------------------------+
|                             hers |
+----------------------------------+
SELECT substring('Feathers' FROM -3);
+-----------------------------------+
|substring(Feathers, -3, 2147483647)|
+-----------------------------------+
|                                ers|
+-----------------------------------+
SELECT substring('Feathers' FROM 5 FOR 1);
+--------------------------+
|substring(Feathers, 5, 1)|
+--------------------------+
|                         h|
+--------------------------+
-- substring_index
SELECT substring_index('www.apache.org', '.', 2);
+-------------------------------------+
|substring_index(www.apache.org, ., 2)|
+-------------------------------------+
|                           www.apache|
+-------------------------------------+
-- to_binary
SELECT to_binary('abc', 'utf-8');
+---------------------+
|to_binary(abc, utf-8)|
+---------------------+
|           [61 62 63]|
+---------------------+
-- to_char
```

```
SELECT to_char(454, '999');
+-----------------+
|to_char(454, 999)|
+-----------------+
|              454|
+-----------------+
SELECT to_char(454.00, '000D00');
+------------------------+
|to_char(454.00, 000D00)|
+------------------------+
|                 454.00|
+------------------------+
SELECT to_char(12454, '99G999');
+----------------------+
|to_char(12454, 99G999)|
+----------------------+
|                12,454|
+----------------------+
SELECT to_char(78.12, '$99.99');
+----------------------+
|to_char(78.12, $99.99)|
+----------------------+
|                $78.12|
+----------------------+
SELECT to_char(-12454.8, '99G999D9S');
+-----------------------------+
|to_char(-12454.8, 99G999D9S)|
+-----------------------------+
|                   12,454.8-|
+-----------------------------+
-- to_number
SELECT to_number('454', '999');
+-------------------+
|to_number(454, 999)|
+-------------------+
|                454|
+-------------------+
SELECT to_number('454.00', '000.00');
+-------------------------+
|to_number(454.00, 000.00)|
+-------------------------+
|                   454.00|
+-------------------------+
SELECT to_number('12,454', '99,999');
```

```
+-------------------------+
|to_number(12,454, 99,999)|
+-------------------------+
|                    12454|
+-------------------------+
SELECT to_number('$78.12', '$99.99');
+-------------------------+
|to_number($78.12, $99.99)|
+-------------------------+
|                    78.12|
+-------------------------+
SELECT to_number('12,454.8-', '99,999.9S');
+-------------------------------+
|to_number(12,454.8-, 99,999.9S)|
+-------------------------------+
|                       -12454.8|
+-------------------------------+
-- to_varchar
SELECT to_varchar(454, '999');
+-----------------+
|to_char(454, 999)|
+-----------------+
|              454|
+-----------------+
SELECT to_varchar(454.00, '000D00');
+-----------------------+
|to_char(454.00, 000D00)|
+-----------------------+
|                 454.00|
+-----------------------+
SELECT to_varchar(12454, '99G999');
+----------------------+
|to_char(12454, 99G999)|
+----------------------+
|                12,454|
+----------------------+
SELECT to_varchar(78.12, '$99.99');
+----------------------+
|to_char(78.12, $99.99)|
+----------------------+
|                $78.12|
+----------------------+
SELECT to_varchar(-12454.8, '99G999D9S');
+---------------------------+
```

```
|to_char(-12454.8, 99G999D9S)|
+----------------------------+
|                  12,454.8-|
+----------------------------+
-- translate
SELECT translate('AaBbCc', 'abc', '123');
+----------------------------+
|translate(AaBbCc, abc, 123)|
+----------------------------+
|                      A1B2C3|
+----------------------------+
-- try_to_binary
SELECT try_to_binary('abc', 'utf-8');
+--------------------------+
|try_to_binary(abc, utf-8)|
+--------------------------+
|               [61 62 63]|
+--------------------------+
select try_to_binary('a!', 'base64');
+--------------------------+
|try_to_binary(a!, base64)|
+--------------------------+
|                      NULL|
+--------------------------+
select try_to_binary('abc', 'invalidFormat');
+---------------------------------+
|try_to_binary(abc, invalidFormat)|
+---------------------------------+
|                             NULL|
+---------------------------------+
-- try_to_number
SELECT try_to_number('454', '999');
+-----------------------+
|try_to_number(454, 999)|
+-----------------------+
|                    454|
+-----------------------+
SELECT try_to_number('454.00', '000.00');
+----------------------------+
|try_to_number(454.00, 000.00)|
+----------------------------+
|                      454.00|
+----------------------------+
SELECT try_to_number('12,454', '99,999');
```

```
+-----------------------------+
|try_to_number(12,454, 99,999)|
+-----------------------------+
|                        12454|
+-----------------------------+
SELECT try_to_number('$78.12', '$99.99');
+-----------------------------+
|try_to_number($78.12, $99.99)|
+-----------------------------+
|                        78.12|
+-----------------------------+
SELECT try_to_number('12,454.8-', '99,999.9S');
+----------------------------------+
|try_to_number(12,454.8-, 99,999.9S)|
+----------------------------------+
|                          -12454.8|
+----------------------------------+
-- ucase
SELECT ucase('Feathers');
+---------------+
|ucase(Feathers)|
+---------------+
|       FEATHERS|
+---------------+
-- unbase64
SELECT unbase64('U3BhcmsgU1FM');
+----------------------+
|unbase64(U3BhcmsgU1FM)|
+----------------------+
|  [53 70 61 72 6B 2...|
+----------------------+
-- upper
SELECT upper('Feathers');
+---------------+
|upper(Feathers)|
+---------------+
|       FEATHERS|
+---------------+
```

日付および時刻関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
add_months(start_date, num_months)	`start_date` より後の `num_months` の日付を返します。
convert_timezone([sourceTz,]targetTz, sourceTs)	タイムゾーン `sourceTs` のないタイムスタンプを `sourceTz` タイムゾーンから `targetTz` に変換します。
curdate()	クエリ評価の開始時点の現在の日付を返します。同じクエリ内のすべての curdate 呼び出しは、同じ値を返します。
current_date()	クエリ評価の開始時点の現在の日付を返します。同じクエリ内の current_date のすべての呼び出しは、同じ値を返します。
current_date	クエリ評価の開始時点の現在の日付を返します。
current_timestamp()	クエリ評価の開始時点の現在のタイムスタンプを返します。同じクエリ内の current_timestamp のすべての呼び出しは、同じ値を返します。
current_timestamp	クエリ評価の開始時点の現在のタイムスタンプを返します。
current_timezone()	現在のセッションのローカルタイムゾーンを返します。

関数	説明
date_add(start_date, num_days)	start_date より後の num_days の日付を返します。
date_diff(endDate, startDate)	startDate から endDate までの日数を返します。
date_format(timestamp, fmt)	日付形式 fmt で指定された形式で timestamp を文字列の値に変換します。
date_from_unix_date(days)	1970-01-01 からの日数から日付を作成します。
date_part(field, source)	日付/タイムスタンプまたは時間間隔のソースの一部を抽出します。
date_sub(start_date, num_days)	start_date より前の num_days の日付を返します。
date_trunc(fmt, ts)	タイムスタンプ ts を、形式モデル fmt で指定された単位に切り捨てて返します。
dateadd(start_date, num_days)	start_date より後の num_days の日付を返します。
datediff(endDate, startDate)	startDate から endDate までの日数を返します。
datepart(field, source)	日付/タイムスタンプまたは時間間隔のソースの一部を抽出します。
day(date)	日付/タイムスタンプの日付を返します。
dayofmonth(date)	日付/タイムスタンプの日付を返します。
dayofweek(date)	日付/タイムスタンプの曜日を返します (1 = 日曜日、2 = 月曜日、...、7 = 土曜日)。
dayofyear(date)	日付/タイムスタンプの年内日付を返します。

関数	説明
extract(field FROM source)	日付/タイムスタンプまたは時間間隔のソースの一部を抽出します。
from_unixtime(unix_time[, fmt])	指定された fmt で unix_time を返します。
from_utc_timestamp(timestamp, timezone)	'2017-07-14 02:40:00.0' のようなタイムスタンプがある場合、それを UTC の時間として解釈し、その時間を特定のタイムゾーンのタイムスタンプとしてレンダリングします。例えば、'GMT+1' は '2017-07-14 03:40:00.0' を生成します。
hour(timestamp)	文字列/タイムスタンプから時間の要素を返します。
last_day(date)	指定された日付が属する月の末日を返します。
localtimestamp()	クエリ評価の開始時にタイムゾーンなしで現在のタイムスタンプを返します。同じクエリ内の localtimestamp のすべての呼び出しは、同じ値を返します。
localtimestamp	クエリ評価の開始時にセッションタイムゾーンの現在のローカル日時を返します。
make_date(year, month, day)	年、月、日フィールドから日付を作成します。
make_dt_interval([days[, hours[, mins[, secs]]]])	DayTimeIntervalType の期間を日、時間、分、秒から作成します。
make_interval([years[, months[, weeks[, days[, hours[, mins[, secs]]]]]]])	年、月、週、日、時間、分、秒から時間間隔を作成します。

関数	説明
make_timestamp(year, month, day, hour, min, sec[, timezone])	年、月、日、時間、分、秒、タイムゾーンフィールドからタイムスタンプを作成します。
make_timestamp_ltz(year, month, day, hour, min, sec[, timezone])	現在のタイムスタンプを、年、月、日、時間、分、秒、タイムゾーンの各フィールドのローカルタイムゾーンで作成します。
make_timestamp_ntz(year, month, day, hour, min, sec)	年、月、日、時間、分、秒フィールドからローカル日時を作成します。
make_ym_interval([years[, months]])	年・月の間隔を年、月から設定します。
minute(タイムスタンプ)	文字列/タイムスタンプの分の要素を返します。
month(date)	日付/タイムスタンプの月の要素を返します。
months_between(timestamp1, timestamp2[, roundOff])	`timestamp1` が `timestamp2` より後の場合、結果は正になります。`timestamp1` と `timestamp2` が同じ月の日付である場合、または両方とも月末の日付である場合、時刻情報は無視されます。それ以外の場合、その差分は 1 か月を 31 日として計算され、roundOff=false でない限り小数点以下 8 桁に丸められます。
next_day(start_date, day_of_week)	指定された名前の `start_date` より後の最初の日付を返します。入力パラメータの少なくとも 1 つが NULL の場合、関数は NULL を返します。
now()	クエリ評価の開始時点の現在のタイムスタンプを返します。
quarter(date)	日付の四半期を 1～4 の範囲で返します。

関数	説明
second(timestamp)	文字列/タイムスタンプの秒の要素を返します。
session_window(time_column, gap_duration)	列とギャップの期間を指定するタイムスタンプを指定してセッションウィンドウを生成します。詳細な説明と例については、構造化ストリーミングガイドドキュメントの「時間枠のタイプ」を参照してください。
timestamp_micros(microseconds)	UTC エポックからのマイクロ秒数をもとにタイムスタンプを生成します。
timestamp_millis(milliseconds)	UTC エポックからのミリ秒数をもとにタイムスタンプを生成します。
timestamp_seconds(seconds)	UTC エポックからの秒数 (小数でも可) をもとにタイムスタンプを作成します。
to_date(date_str[, fmt])	`date_str` 式を `fmt` 式で解析し、日付型に変換します。無効な入力の場合は null を返します。デフォルトでは、`fmt` を省略すると、キャストルールに従って日付が設定されます。
to_timestamp(timestamp_str[, fmt])	`timestamp_str` 式を `fmt` 式で解析し、タイムスタンプに変換します。無効な入力の場合は null を返します。デフォルトでは、`fmt` を省略すると、キャストルールに従ってタイムスタンプになります。
to_timestamp_ltz(timestamp_str[, fmt])	`timestamp_str` 式を `fmt` 式で解析し、ローカルタイムゾーンのタイムスタンプに変換します。無効な入力の場合は null を返します。デフォルトでは、`fmt` を省略すると、キャストルールに従ってタイムスタンプになります。

関数	説明
to_timestamp_ntz(timestamp_str[, fmt])	`timestamp_str` 式を `fmt` 式で解析し、タイムゾーンなしのタイムスタンプに変換します。無効な入力の場合は null を返します。デフォルトでは、`fmt` を省略すると、キャストルールに従ってタイムスタンプになります。
to_unix_timestamp(timeExp[, fmt])	指定された時刻の UNIX タイムスタンプを返します。
to_utc_timestamp(timestamp, timezone)	'2017-07-14 02:40:00.0' のようなタイムスタンプがある場合、それを特定のタイムゾーンの時刻として解釈し、その時刻を UTC のタイムスタンプとしてレンダリングします。例えば、'GMT+1' は '2017-07-14 01:40:00.0' を生成します。
trunc(date, fmt)	形式モデル `fmt` で指定された単位に切り捨てられた日付の時刻部分で `date` を返します。
try_to_timestamp(timestamp_str[, fmt])	`timestamp_str` 式を `fmt` 式で解析し、タイムスタンプに変換します。
unix_date(date)	1970-01-01 からの日数を返します。
unix_micros(timestamp)	1970-01-01 00:00:00 UTC からのマイクロ秒数を返します。
unix_millis(timestamp)	1970-01-01 00:00:00 UTC からのミリ秒数を返します。より高いレベルの精度を切り捨てます。
unix_seconds(timestamp)	1970-01-01 00:00:00 UTC からの秒数を返します。より高いレベルの精度を切り捨てます。
unix_timestamp([timeExp[, fmt]])	現在または指定された時刻の UNIX タイムスタンプを返します。

関数	説明
weekday(date)	日付/タイムスタンプの曜日を返します (0 = 月曜日、1 = 火曜日、...、6 = 日曜日)。
weekofyear(date)	指定された日付の年の週を返します。1 週間は月曜日に開始されると見なされ、第 1 週は 3 日より多い日数の最初の週です。
window(time_column, window_duration[, slide_duration[, start_time]])	列を指定するタイムスタンプに基づき、行を 1 つ以上の時間枠にバケット化します。ウィンドウの開始は含まれますが、ウィンドウの終了は含まれません。例えば、12:05 はウィンドウ [12:05,12:10) に含まれますが、[12:00,12:05) には含まれません。ウィンドウはマイクロ秒の精度をサポートできます。月単位のウィンドウはサポートされていません。詳細な説明と例については、「構造化ストリーミングガイドドキュメント」の「イベント時刻のウィンドウ操作」を参照してください。
window_time(window_column)	ウィンドウのイベント時間値に使用できる時間/セッションウィンドウ列から時間値を抽出します。抽出時間は (window.end - 1) です。これは、集計ウィンドウが排他的な上限を持つことを反映しています - [start, end) 詳細な説明と例については、「構造化ストリーミングガイドドキュメント」の「イベント時間に対するウィンドウ操作」を参照してください。
year(date)	日付/タイムスタンプの年の要素を返します。

例

```
-- add_months
SELECT add_months('2016-08-31', 1);
+-------------------------+
|add_months(2016-08-31, 1)|
```

```
+-------------------------+
|               2016-09-30|
+-------------------------+
-- convert_timezone
SELECT convert_timezone('Europe/Brussels', 'America/Los_Angeles',
 timestamp_ntz'2021-12-06 00:00:00');
+-----------------------------------------------------------------------------------------------
+
|convert_timezone(Europe/Brussels, America/Los_Angeles, TIMESTAMP_NTZ '2021-12-06
 00:00:00')|
+-----------------------------------------------------------------------------------------------
+
|                                                                                 2021-12-05
 15:00:00|
+-----------------------------------------------------------------------------------------------
+
SELECT convert_timezone('Europe/Brussels', timestamp_ntz'2021-12-05 15:00:00');
+-----------------------------------------------------------------------------------------------
+
|convert_timezone(current_timezone(), Europe/Brussels, TIMESTAMP_NTZ '2021-12-05
 15:00:00')|
+-----------------------------------------------------------------------------------------------
+
|                                                                               2021-12-05
 07:00:00|
+-----------------------------------------------------------------------------------------------
+
-- curdate
SELECT curdate();
+--------------+
|current_date()|
+--------------+
|    2024-02-24|
+--------------+
-- current_date
SELECT current_date();
+--------------+
|current_date()|
+--------------+
|    2024-02-24|
+--------------+
SELECT current_date;
+--------------+
|current_date()|
```

```
+--------------+
|    2024-02-24|
+--------------+
-- current_timestamp
SELECT current_timestamp();
+--------------------+
| current_timestamp()|
+--------------------+
|2024-02-24 16:36:...|
+--------------------+
SELECT current_timestamp;
+--------------------+
| current_timestamp()|
+--------------------+
|2024-02-24 16:36:...|
+--------------------+
-- current_timezone
SELECT current_timezone();
+------------------+
|current_timezone()|
+------------------+
|        Asia/Seoul|
+------------------+
-- date_add
SELECT date_add('2016-07-30', 1);
+-----------------------+
|date_add(2016-07-30, 1)|
+-----------------------+
|             2016-07-31|
+-----------------------+
-- date_diff
SELECT date_diff('2009-07-31', '2009-07-30');
+---------------------------------+
|date_diff(2009-07-31, 2009-07-30)|
+---------------------------------+
|                                1|
+---------------------------------+
SELECT date_diff('2009-07-30', '2009-07-31');
+---------------------------------+
|date_diff(2009-07-30, 2009-07-31)|
+---------------------------------+
|                               -1|
+---------------------------------+
-- date_format
```

```
SELECT date_format('2016-04-08', 'y');
+-------------------------+
|date_format(2016-04-08, y)|
+-------------------------+
|                     2016|
+-------------------------+
-- date_from_unix_date
SELECT date_from_unix_date(1);
+---------------------+
|date_from_unix_date(1)|
+---------------------+
|           1970-01-02|
+---------------------+
-- date_part
SELECT date_part('YEAR', TIMESTAMP '2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|date_part(YEAR, TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                       2019|
+-----------------------------------------------------------+
SELECT date_part('week', timestamp'2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|date_part(week, TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                         33|
+-----------------------------------------------------------+
SELECT date_part('doy', DATE'2019-08-12');
+---------------------------------+
|date_part(doy, DATE '2019-08-12')|
+---------------------------------+
|                              224|
+---------------------------------+
SELECT date_part('SECONDS', timestamp'2019-10-01 00:00:01.000001');
+------------------------------------------------------------+
|date_part(SECONDS, TIMESTAMP '2019-10-01 00:00:01.000001')|
+------------------------------------------------------------+
|                                                    1.000001|
+------------------------------------------------------------+
SELECT date_part('days', interval 5 days 3 hours 7 minutes);
+-----------------------------------------------+
|date_part(days, INTERVAL '5 03:07' DAY TO MINUTE)|
+-----------------------------------------------+
|                                              5|
+-----------------------------------------------+
```

```
SELECT date_part('seconds', interval 5 hours 30 seconds 1 milliseconds 1 microseconds);
+-------------------------------------------------------------+
|date_part(seconds, INTERVAL '05:00:30.001001' HOUR TO SECOND)|
+-------------------------------------------------------------+
|                                                    30.001001|
+-------------------------------------------------------------+
SELECT date_part('MONTH', INTERVAL '2021-11' YEAR TO MONTH);
+---------------------------------------------------+
|date_part(MONTH, INTERVAL '2021-11' YEAR TO MONTH)|
+---------------------------------------------------+
|                                                 11|
+---------------------------------------------------+
SELECT date_part('MINUTE', INTERVAL '123 23:55:59.002001' DAY TO SECOND);
+-----------------------------------------------------------------+
|date_part(MINUTE, INTERVAL '123 23:55:59.002001' DAY TO SECOND)|
+-----------------------------------------------------------------+
|                                                               55|
+-----------------------------------------------------------------+
-- date_sub
SELECT date_sub('2016-07-30', 1);
+-----------------------+
|date_sub(2016-07-30, 1)|
+-----------------------+
|             2016-07-29|
+-----------------------+
-- date_trunc
SELECT date_trunc('YEAR', '2015-03-05T09:32:05.359');
+----------------------------------------+
|date_trunc(YEAR, 2015-03-05T09:32:05.359)|
+----------------------------------------+
|                     2015-01-01 00:00:00|
+----------------------------------------+
SELECT date_trunc('MM', '2015-03-05T09:32:05.359');
+---------------------------------------+
|date_trunc(MM, 2015-03-05T09:32:05.359)|
+---------------------------------------+
|                    2015-03-01 00:00:00|
+---------------------------------------+
SELECT date_trunc('DD', '2015-03-05T09:32:05.359');
+---------------------------------------+
|date_trunc(DD, 2015-03-05T09:32:05.359)|
+---------------------------------------+
|                    2015-03-05 00:00:00|
+---------------------------------------+
```

```
SELECT date_trunc('HOUR', '2015-03-05T09:32:05.359');
+----------------------------------------+
|date_trunc(HOUR, 2015-03-05T09:32:05.359)|
+----------------------------------------+
|                     2015-03-05 09:00:00|
+----------------------------------------+
SELECT date_trunc('MILLISECOND', '2015-03-05T09:32:05.123456');
+-----------------------------------------------------+
|date_trunc(MILLISECOND, 2015-03-05T09:32:05.123456)|
+-----------------------------------------------------+
|                              2015-03-05 09:32:...|
+-----------------------------------------------------+
-- dateadd
SELECT dateadd('2016-07-30', 1);
+-----------------------+
|date_add(2016-07-30, 1)|
+-----------------------+
|             2016-07-31|
+-----------------------+
-- datediff
SELECT datediff('2009-07-31', '2009-07-30');
+--------------------------------+
|datediff(2009-07-31, 2009-07-30)|
+--------------------------------+
|                               1|
+--------------------------------+
SELECT datediff('2009-07-30', '2009-07-31');
+--------------------------------+
|datediff(2009-07-30, 2009-07-31)|
+--------------------------------+
|                              -1|
+--------------------------------+
-- datepart
SELECT datepart('YEAR', TIMESTAMP '2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|datepart(YEAR FROM TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                       2019|
+-----------------------------------------------------------+
SELECT datepart('week', timestamp'2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|datepart(week FROM TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                         33|
```

```
+-----------------------------------------------------------+
SELECT datepart('doy', DATE'2019-08-12');
+-----------------------------------+
|datepart(doy FROM DATE '2019-08-12')|
+-----------------------------------+
|                                224|
+-----------------------------------+
SELECT datepart('SECONDS', timestamp'2019-10-01 00:00:01.000001');
+-------------------------------------------------------------+
|datepart(SECONDS FROM TIMESTAMP '2019-10-01 00:00:01.000001')|
+-------------------------------------------------------------+
|                                                     1.000001|
+-------------------------------------------------------------+
SELECT datepart('days', interval 5 days 3 hours 7 minutes);
+---------------------------------------------------+
|datepart(days FROM INTERVAL '5 03:07' DAY TO MINUTE)|
+---------------------------------------------------+
|                                                  5|
+---------------------------------------------------+
SELECT datepart('seconds', interval 5 hours 30 seconds 1 milliseconds 1 microseconds);
+-----------------------------------------------------------------+
|datepart(seconds FROM INTERVAL '05:00:30.001001' HOUR TO SECOND)|
+-----------------------------------------------------------------+
|                                                        30.001001|
+-----------------------------------------------------------------+
SELECT datepart('MONTH', INTERVAL '2021-11' YEAR TO MONTH);
+-----------------------------------------------------+
|datepart(MONTH FROM INTERVAL '2021-11' YEAR TO MONTH)|
+-----------------------------------------------------+
|                                                   11|
+-----------------------------------------------------+
SELECT datepart('MINUTE', INTERVAL '123 23:55:59.002001' DAY TO SECOND);
+-------------------------------------------------------------------+
|datepart(MINUTE FROM INTERVAL '123 23:55:59.002001' DAY TO SECOND)|
+-------------------------------------------------------------------+
|                                                                 55|
+-------------------------------------------------------------------+
-- day
SELECT day('2009-07-30');
+---------------+
|day(2009-07-30)|
+---------------+
|             30|
+---------------+
```

```
-- dayofmonth
SELECT dayofmonth('2009-07-30');
+----------------------+
|dayofmonth(2009-07-30)|
+----------------------+
|                    30|
+----------------------+
-- dayofweek
SELECT dayofweek('2009-07-30');
+---------------------+
|dayofweek(2009-07-30)|
+---------------------+
|                    5|
+---------------------+
-- dayofyear
SELECT dayofyear('2016-04-09');
+---------------------+
|dayofyear(2016-04-09)|
+---------------------+
|                  100|
+---------------------+
-- extract
SELECT extract(YEAR FROM TIMESTAMP '2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|extract(YEAR FROM TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                       2019|
+-----------------------------------------------------------+
SELECT extract(week FROM timestamp'2019-08-12 01:00:00.123456');
+-----------------------------------------------------------+
|extract(week FROM TIMESTAMP '2019-08-12 01:00:00.123456')|
+-----------------------------------------------------------+
|                                                         33|
+-----------------------------------------------------------+
SELECT extract(doy FROM DATE'2019-08-12');
+----------------------------------+
|extract(doy FROM DATE '2019-08-12')|
+----------------------------------+
|                               224|
+----------------------------------+
SELECT extract(SECONDS FROM timestamp'2019-10-01 00:00:01.000001');
+-------------------------------------------------------------+
|extract(SECONDS FROM TIMESTAMP '2019-10-01 00:00:01.000001')|
+-------------------------------------------------------------+
```

```
|                                                          1.000001|
+------------------------------------------------------------------+
SELECT extract(days FROM interval 5 days 3 hours 7 minutes);
+---------------------------------------------------+
|extract(days FROM INTERVAL '5 03:07' DAY TO MINUTE)|
+---------------------------------------------------+
|                                                  5|
+---------------------------------------------------+
SELECT extract(seconds FROM interval 5 hours 30 seconds 1 milliseconds 1 microseconds);
+-------------------------------------------------------------------+
|extract(seconds FROM INTERVAL '05:00:30.001001' HOUR TO SECOND)|
+-------------------------------------------------------------------+
|                                                         30.001001|
+-------------------------------------------------------------------+
SELECT extract(MONTH FROM INTERVAL '2021-11' YEAR TO MONTH);
+---------------------------------------------------+
|extract(MONTH FROM INTERVAL '2021-11' YEAR TO MONTH)|
+---------------------------------------------------+
|                                                 11|
+---------------------------------------------------+
SELECT extract(MINUTE FROM INTERVAL '123 23:55:59.002001' DAY TO SECOND);
+-----------------------------------------------------------------+
|extract(MINUTE FROM INTERVAL '123 23:55:59.002001' DAY TO SECOND)|
+-----------------------------------------------------------------+
|                                                               55|
+-----------------------------------------------------------------+
-- from_unixtime
SELECT from_unixtime(0, 'yyyy-MM-dd HH:mm:ss');
+--------------------------------------+
|from_unixtime(0, yyyy-MM-dd HH:mm:ss)|
+--------------------------------------+
|                  1970-01-01 09:00:00|
+--------------------------------------+
SELECT from_unixtime(0);
+--------------------------------------+
|from_unixtime(0, yyyy-MM-dd HH:mm:ss)|
+--------------------------------------+
|                  1970-01-01 09:00:00|
+--------------------------------------+
-- from_utc_timestamp
SELECT from_utc_timestamp('2016-08-31', 'Asia/Seoul');
+-----------------------------------------+
|from_utc_timestamp(2016-08-31, Asia/Seoul)|
+-----------------------------------------+
```

```
|                      2016-08-31 09:00:00|
+-----------------------------------------+
-- hour
SELECT hour('2009-07-30 12:58:59');
+-------------------------+
|hour(2009-07-30 12:58:59)|
+-------------------------+
|                       12|
+-------------------------+
-- last_day
SELECT last_day('2009-01-12');
+--------------------+
|last_day(2009-01-12)|
+--------------------+
|          2009-01-31|
+--------------------+
-- localtimestamp
SELECT localtimestamp();
+--------------------+
|    localtimestamp()|
+--------------------+
|2024-02-24 16:36:...|
+--------------------+
-- make_date
SELECT make_date(2013, 7, 15);
+----------------------+
|make_date(2013, 7, 15)|
+----------------------+
|            2013-07-15|
+----------------------+
SELECT make_date(2019, 7, NULL);
+------------------------+
|make_date(2019, 7, NULL)|
+------------------------+
|                    NULL|
+------------------------+
-- make_dt_interval
SELECT make_dt_interval(1, 12, 30, 01.001001);
+-------------------------------------+
|make_dt_interval(1, 12, 30, 1.001001)|
+-------------------------------------+
|                 INTERVAL '1 12:30...|
+-------------------------------------+
SELECT make_dt_interval(2);
```

```
+----------------------------------+
|make_dt_interval(2, 0, 0, 0.000000)|
+----------------------------------+
|               INTERVAL '2 00:00...|
+----------------------------------+
SELECT make_dt_interval(100, null, 3);
+---------------------------------------+
|make_dt_interval(100, NULL, 3, 0.000000)|
+---------------------------------------+
|                                   NULL|
+---------------------------------------+
-- make_interval
SELECT make_interval(100, 11, 1, 1, 12, 30, 01.001001);
+-----------------------------------------------+
|make_interval(100, 11, 1, 1, 12, 30, 1.001001)|
+-----------------------------------------------+
|                          100 years 11 mont...|
+-----------------------------------------------+
SELECT make_interval(100, null, 3);
+-----------------------------------------------+
|make_interval(100, NULL, 3, 0, 0, 0, 0.000000)|
+-----------------------------------------------+
|                                           NULL|
+-----------------------------------------------+
SELECT make_interval(0, 1, 0, 1, 0, 0, 100.000001);
+-------------------------------------------+
|make_interval(0, 1, 0, 1, 0, 0, 100.000001)|
+-------------------------------------------+
|                       1 months 1 days 1...|
+-------------------------------------------+
-- make_timestamp
SELECT make_timestamp(2014, 12, 28, 6, 30, 45.887);
+-----------------------------------------+
|make_timestamp(2014, 12, 28, 6, 30, 45.887)|
+-----------------------------------------+
|                      2014-12-28 06:30:...|
+-----------------------------------------+
SELECT make_timestamp(2014, 12, 28, 6, 30, 45.887, 'CET');
+-------------------------------------------------+
|make_timestamp(2014, 12, 28, 6, 30, 45.887, CET)|
+-------------------------------------------------+
|                           2014-12-28 14:30:...|
+-------------------------------------------------+
SELECT make_timestamp(2019, 6, 30, 23, 59, 60);
```

```
+--------------------------------------+
|make_timestamp(2019, 6, 30, 23, 59, 60)|
+--------------------------------------+
|                   2019-07-01 00:00:00|
+--------------------------------------+
SELECT make_timestamp(2019, 6, 30, 23, 59, 1);
+--------------------------------------+
|make_timestamp(2019, 6, 30, 23, 59, 1)|
+--------------------------------------+
|                  2019-06-30 23:59:01|
+--------------------------------------+
SELECT make_timestamp(null, 7, 22, 15, 30, 0);
+-------------------------------------+
|make_timestamp(NULL, 7, 22, 15, 30, 0)|
+-------------------------------------+
|                                 NULL|
+-------------------------------------+
-- make_timestamp_ltz
SELECT make_timestamp_ltz(2014, 12, 28, 6, 30, 45.887);
+-----------------------------------------------+
|make_timestamp_ltz(2014, 12, 28, 6, 30, 45.887)|
+-----------------------------------------------+
|                          2014-12-28 06:30:...|
+-----------------------------------------------+
SELECT make_timestamp_ltz(2014, 12, 28, 6, 30, 45.887, 'CET');
+-------------------------------------------------------+
|make_timestamp_ltz(2014, 12, 28, 6, 30, 45.887, CET)|
+-------------------------------------------------------+
|                               2014-12-28 14:30:...|
+-------------------------------------------------------+
SELECT make_timestamp_ltz(2019, 6, 30, 23, 59, 60);
+-----------------------------------------+
|make_timestamp_ltz(2019, 6, 30, 23, 59, 60)|
+-----------------------------------------+
|                      2019-07-01 00:00:00|
+-----------------------------------------+
SELECT make_timestamp_ltz(null, 7, 22, 15, 30, 0);
+----------------------------------------+
|make_timestamp_ltz(NULL, 7, 22, 15, 30, 0)|
+----------------------------------------+
|                                    NULL|
+----------------------------------------+
-- make_timestamp_ntz
SELECT make_timestamp_ntz(2014, 12, 28, 6, 30, 45.887);
```

```
+-----------------------------------------------+
|make_timestamp_ntz(2014, 12, 28, 6, 30, 45.887)|
+-----------------------------------------------+
|                            2014-12-28 06:30:...|
+-----------------------------------------------+
SELECT make_timestamp_ntz(2019, 6, 30, 23, 59, 60);
+-----------------------------------------+
|make_timestamp_ntz(2019, 6, 30, 23, 59, 60)|
+-----------------------------------------+
|                         2019-07-01 00:00:00|
+-----------------------------------------+
SELECT make_timestamp_ntz(null, 7, 22, 15, 30, 0);
+--------------------------------------+
|make_timestamp_ntz(NULL, 7, 22, 15, 30, 0)|
+--------------------------------------+
|                                  NULL|
+--------------------------------------+
-- make_ym_interval
SELECT make_ym_interval(1, 2);
+----------------------+
|make_ym_interval(1, 2)|
+----------------------+
|  INTERVAL '1-2' YE...|
+----------------------+
SELECT make_ym_interval(1, 0);
+----------------------+
|make_ym_interval(1, 0)|
+----------------------+
|  INTERVAL '1-0' YE...|
+----------------------+
SELECT make_ym_interval(-1, 1);
+----------------------+
|make_ym_interval(-1, 1)|
+----------------------+
|   INTERVAL '-0-11' ...|
+----------------------+
SELECT make_ym_interval(2);
+----------------------+
|make_ym_interval(2, 0)|
+----------------------+
|  INTERVAL '2-0' YE...|
+----------------------+
-- minute
SELECT minute('2009-07-30 12:58:59');
```

```
+--------------------------+
|minute(2009-07-30 12:58:59)|
+--------------------------+
|                        58|
+--------------------------+
-- month
SELECT month('2016-07-30');
+-----------------+
|month(2016-07-30)|
+-----------------+
|                7|
+-----------------+
-- months_between
SELECT months_between('1997-02-28 10:30:00', '1996-10-30');
+---------------------------------------------------+
|months_between(1997-02-28 10:30:00, 1996-10-30, true)|
+---------------------------------------------------+
|                                         3.94959677|
+---------------------------------------------------+
SELECT months_between('1997-02-28 10:30:00', '1996-10-30', false);
+----------------------------------------------------+
|months_between(1997-02-28 10:30:00, 1996-10-30, false)|
+----------------------------------------------------+
|                                  3.9495967741935485|
+----------------------------------------------------+
-- next_day
SELECT next_day('2015-01-14', 'TU');
+------------------------+
|next_day(2015-01-14, TU)|
+------------------------+
|              2015-01-20|
+------------------------+
-- now
SELECT now();
+--------------------+
|               now()|
+--------------------+
|2024-02-24 16:36:...|
+--------------------+
-- quarter
SELECT quarter('2016-08-31');
+-------------------+
|quarter(2016-08-31)|
+-------------------+
```

```
|                  3|
+-------------------+
-- second
SELECT second('2009-07-30 12:58:59');
+--------------------------+
|second(2009-07-30 12:58:59)|
+--------------------------+
|                        59|
+--------------------------+
-- session_window
SELECT a, session_window.start, session_window.end, count(*) as cnt FROM VALUES ('A1',
 '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:10:00'),
 ('A2', '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, session_window(b, '5 minutes')
 ORDER BY a, start;
+---+-------------------+-------------------+---+
|  a|              start|                end|cnt|
+---+-------------------+-------------------+---+
| A1|2021-01-01 00:00:00|2021-01-01 00:09:30|  2|
| A1|2021-01-01 00:10:00|2021-01-01 00:15:00|  1|
| A2|2021-01-01 00:01:00|2021-01-01 00:06:00|  1|
+---+-------------------+-------------------+---+
SELECT a, session_window.start, session_window.end, count(*) as cnt FROM VALUES ('A1',
 '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:10:00'),
 ('A2', '2021-01-01 00:01:00'), ('A2', '2021-01-01 00:04:30') AS tab(a, b) GROUP by a,
 session_window(b, CASE WHEN a = 'A1' THEN '5 minutes' WHEN a = 'A2' THEN '1 minute'
 ELSE '10 minutes' END) ORDER BY a, start;
+---+-------------------+-------------------+---+
|  a|              start|                end|cnt|
+---+-------------------+-------------------+---+
| A1|2021-01-01 00:00:00|2021-01-01 00:09:30|  2|
| A1|2021-01-01 00:10:00|2021-01-01 00:15:00|  1|
| A2|2021-01-01 00:01:00|2021-01-01 00:02:00|  1|
| A2|2021-01-01 00:04:30|2021-01-01 00:05:30|  1|
+---+-------------------+-------------------+---+
-- timestamp_micros
SELECT timestamp_micros(1230219000123123);
+---------------------------------+
|timestamp_micros(1230219000123123)|
+---------------------------------+
|              2008-12-26 00:30:...|
+---------------------------------+
-- timestamp_millis
SELECT timestamp_millis(1230219000123);
+-------------------------------+
```

```
|timestamp_millis(1230219000123)|
+-------------------------------+
|           2008-12-26 00:30:...|
+-------------------------------+
-- timestamp_seconds
SELECT timestamp_seconds(1230219000);
+-----------------------------+
|timestamp_seconds(1230219000)|
+-----------------------------+
|          2008-12-26 00:30:00|
+-----------------------------+
SELECT timestamp_seconds(1230219000.123);
+---------------------------------+
|timestamp_seconds(1230219000.123)|
+---------------------------------+
|             2008-12-26 00:30:...|
+---------------------------------+
-- to_date
SELECT to_date('2009-07-30 04:17:52');
+----------------------------+
|to_date(2009-07-30 04:17:52)|
+----------------------------+
|                  2009-07-30|
+----------------------------+
SELECT to_date('2016-12-31', 'yyyy-MM-dd');
+-------------------------------+
|to_date(2016-12-31, yyyy-MM-dd)|
+-------------------------------+
|                     2016-12-31|
+-------------------------------+
-- to_timestamp
SELECT to_timestamp('2016-12-31 00:12:00');
+---------------------------------+
|to_timestamp(2016-12-31 00:12:00)|
+---------------------------------+
|              2016-12-31 00:12:00|
+---------------------------------+
SELECT to_timestamp('2016-12-31', 'yyyy-MM-dd');
+-----------------------------------+
|to_timestamp(2016-12-31, yyyy-MM-dd)|
+-----------------------------------+
|                 2016-12-31 00:00:00|
+-----------------------------------+
-- to_timestamp_ltz
```

```
SELECT to_timestamp_ltz('2016-12-31 00:12:00');
+------------------------------------+
|to_timestamp_ltz(2016-12-31 00:12:00)|
+------------------------------------+
|                  2016-12-31 00:12:00|
+------------------------------------+
SELECT to_timestamp_ltz('2016-12-31', 'yyyy-MM-dd');
+----------------------------------------+
|to_timestamp_ltz(2016-12-31, yyyy-MM-dd)|
+----------------------------------------+
|                     2016-12-31 00:00:00|
+----------------------------------------+
-- to_timestamp_ntz
SELECT to_timestamp_ntz('2016-12-31 00:12:00');
+------------------------------------+
|to_timestamp_ntz(2016-12-31 00:12:00)|
+------------------------------------+
|                  2016-12-31 00:12:00|
+------------------------------------+
SELECT to_timestamp_ntz('2016-12-31', 'yyyy-MM-dd');
+----------------------------------------+
|to_timestamp_ntz(2016-12-31, yyyy-MM-dd)|
+----------------------------------------+
|                     2016-12-31 00:00:00|
+----------------------------------------+
-- to_unix_timestamp
SELECT to_unix_timestamp('2016-04-08', 'yyyy-MM-dd');
+----------------------------------------+
|to_unix_timestamp(2016-04-08, yyyy-MM-dd)|
+----------------------------------------+
|                                1460041200|
+----------------------------------------+
-- to_utc_timestamp
SELECT to_utc_timestamp('2016-08-31', 'Asia/Seoul');
+----------------------------------------+
|to_utc_timestamp(2016-08-31, Asia/Seoul)|
+----------------------------------------+
|                     2016-08-30 15:00:00|
+----------------------------------------+
-- trunc
SELECT trunc('2019-08-04', 'week');
+----------------------+
|trunc(2019-08-04, week)|
+----------------------+
```

```
|             2019-07-29|
+-----------------------+
SELECT trunc('2019-08-04', 'quarter');
+-------------------------+
|trunc(2019-08-04, quarter)|
+-------------------------+
|                2019-07-01|
+-------------------------+
SELECT trunc('2009-02-12', 'MM');
+---------------------+
|trunc(2009-02-12, MM)|
+---------------------+
|           2009-02-01|
+---------------------+
SELECT trunc('2015-10-27', 'YEAR');
+-----------------------+
|trunc(2015-10-27, YEAR)|
+-----------------------+
|             2015-01-01|
+-----------------------+
-- try_to_timestamp
SELECT try_to_timestamp('2016-12-31 00:12:00');
+-------------------------------------+
|try_to_timestamp(2016-12-31 00:12:00)|
+-------------------------------------+
|                  2016-12-31 00:12:00|
+-------------------------------------+
SELECT try_to_timestamp('2016-12-31', 'yyyy-MM-dd');
+---------------------------------------+
|try_to_timestamp(2016-12-31, yyyy-MM-dd)|
+---------------------------------------+
|                     2016-12-31 00:00:00|
+---------------------------------------+
SELECT try_to_timestamp('foo', 'yyyy-MM-dd');
+---------------------------------+
|try_to_timestamp(foo, yyyy-MM-dd)|
+---------------------------------+
|                             NULL|
+---------------------------------+
-- unix_date
SELECT unix_date(DATE("1970-01-02"));
+---------------------+
|unix_date(1970-01-02)|
+---------------------+
```

```
|                    1|
+---------------------+
-- unix_micros
SELECT unix_micros(TIMESTAMP('1970-01-01 00:00:01Z'));
+---------------------------------+
|unix_micros(1970-01-01 00:00:01Z)|
+---------------------------------+
|                          1000000|
+---------------------------------+
-- unix_millis
SELECT unix_millis(TIMESTAMP('1970-01-01 00:00:01Z'));
+---------------------------------+
|unix_millis(1970-01-01 00:00:01Z)|
+---------------------------------+
|                             1000|
+---------------------------------+
-- unix_seconds
SELECT unix_seconds(TIMESTAMP('1970-01-01 00:00:01Z'));
+---------------------------------+
|unix_seconds(1970-01-01 00:00:01Z)|
+---------------------------------+
|                                 1|
+---------------------------------+
-- unix_timestamp
SELECT unix_timestamp();
+-----------------------------------------------------------+
|unix_timestamp(current_timestamp(), yyyy-MM-dd HH:mm:ss)|
+-----------------------------------------------------------+
|                                                 1708760216|
+-----------------------------------------------------------+
SELECT unix_timestamp('2016-04-08', 'yyyy-MM-dd');
+-------------------------------------+
|unix_timestamp(2016-04-08, yyyy-MM-dd)|
+-------------------------------------+
|                           1460041200|
+-------------------------------------+
-- weekday
SELECT weekday('2009-07-30');
+------------------+
|weekday(2009-07-30)|
+------------------+
|                 3|
+------------------+
-- weekofyear
```

```
SELECT weekofyear('2008-02-20');
+----------------------+
|weekofyear(2008-02-20)|
+----------------------+
|                     8|
+----------------------+
-- window
SELECT a, window.start, window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01
 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:06:00'), ('A2',
 '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, window(b, '5 minutes') ORDER BY a,
 start;
+---+-------------------+-------------------+---+
|  a|              start|                end|cnt|
+---+-------------------+-------------------+---+
| A1|2021-01-01 00:00:00|2021-01-01 00:05:00|  2|
| A1|2021-01-01 00:05:00|2021-01-01 00:10:00|  1|
| A2|2021-01-01 00:00:00|2021-01-01 00:05:00|  1|
+---+-------------------+-------------------+---+
SELECT a, window.start, window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01
 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:06:00'), ('A2',
 '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, window(b, '10 minutes', '5 minutes')
 ORDER BY a, start;
+---+-------------------+-------------------+---+
|  a|              start|                end|cnt|
+---+-------------------+-------------------+---+
| A1|2020-12-31 23:55:00|2021-01-01 00:05:00|  2|
| A1|2021-01-01 00:00:00|2021-01-01 00:10:00|  3|
| A1|2021-01-01 00:05:00|2021-01-01 00:15:00|  1|
| A2|2020-12-31 23:55:00|2021-01-01 00:05:00|  1|
| A2|2021-01-01 00:00:00|2021-01-01 00:10:00|  1|
+---+-------------------+-------------------+---+
-- window_time
SELECT a, window.start as start, window.end as end, window_time(window), cnt FROM
 (SELECT a, window, count(*) as cnt FROM VALUES ('A1', '2021-01-01 00:00:00'), ('A1',
 '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:06:00'), ('A2', '2021-01-01 00:01:00')
 AS tab(a, b) GROUP by a, window(b, '5 minutes') ORDER BY a, window.start);
+---+-------------------+-------------------+--------------------+---+
|  a|              start|                end| window_time(window)|cnt|
+---+-------------------+-------------------+--------------------+---+
| A1|2021-01-01 00:00:00|2021-01-01 00:05:00|2021-01-01 00:04:...|  2|
| A1|2021-01-01 00:05:00|2021-01-01 00:10:00|2021-01-01 00:09:...|  1|
| A2|2021-01-01 00:00:00|2021-01-01 00:05:00|2021-01-01 00:04:...|  1|
+---+-------------------+-------------------+--------------------+---+
-- year
```

```
SELECT year('2016-07-30');
+----------------+
|year(2016-07-30)|
+----------------+
|            2016|
+----------------+
```

集計関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

集計関数は行全体の値に対して動作し、合計、平均、カウント、最小値/最大値、標準偏差、推定などの数学的計算と、一部の非数学的演算を実行します。

構文

```
aggregate_function(input1 [, input2, ...]) FILTER (WHERE boolean_expression)
```

パラメータ

- boolean_expression – 結果の型がブール値に評価される式を指定します。論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。

順序付きセット集計関数

これらの集計関数は、値を並び替える式 (通常は列名) を指定するため、ほかの集約関数とは異なる構文を使用します。

構文

```
{ PERCENTILE_CONT | PERCENTILE_DISC }(percentile) WITHIN GROUP (ORDER BY
 { order_by_expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [ , ... ] }) FILTER
 (WHERE boolean_expression)
```

パラメータ

- percentile – 検索する値のパーセンタイル。パーセンタイルは 0.0 ~ 1.0 の定数である必要があります。
- order_by_expression – 値を集計する前に値を並び替える式 (通常は列名)。
- boolean_expression – 結果の型がブール値に評価される式を指定します。論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。

例

```
CREATE OR REPLACE TEMPORARY VIEW basic_pays AS SELECT * FROM VALUES
('Jane Doe','Accounting',8435),
('Akua Mansa','Accounting',9998),
('John Doe','Accounting',8992),
('Juan Li','Accounting',8870),
('Carlos Salazar','Accounting',11472),
('Arnav Desai','Accounting',6627),
('Saanvi Sarkar','IT',8113),
('Shirley Rodriguez','IT',5186),
('Nikki Wolf','Sales',9181),
('Alejandro Rosalez','Sales',9441),
('Nikhil Jayashankar','Sales',6660),
('Richard Roe','Sales',10563),
('Pat Candella','SCM',10449),
('Gerard Hernandez','SCM',6949),
('Pamela Castillo','SCM',11303),
('Paulo Santos','SCM',11798),
('Jorge Souza','SCM',10586)
AS basic_pays(employee_name, department, salary);
SELECT * FROM basic_pays;
+------------------+----------+------+
|    employee_name  |department|salary|
+------------------+----------+------+
| Arnav Desai       |Accounting|  6627|
| Jorge Souza       |       SCM| 10586|
| Jane Doe          |Accounting|  8435|
| Nikhil Jayashankar|     Sales|  6660|
| Diego Vanauf      |     Sales| 10563|
| Carlos Salazar    |Accounting| 11472|
| Gerard Hernandez  |       SCM|  6949|
| John Doe          |Accounting|  8992|
| Nikki Wolf        |     Sales|  9181|
| Paulo Santos      |       SCM| 11798|
```

```
| Saanvi Sarkar     |        IT|  8113|
| Shirley Rodriguez |        IT|  5186|
| Pat Candella      |       SCM| 10449|
| Akua Mansa        |Accounting|  9998|
| Pamela Castillo   |       SCM| 11303|
| Alejandro Rosalez |     Sales|  9441|
| Juan Li           |Accounting|  8870|
+-------------------+----------+------+
SELECT
department,
percentile_cont(0.25) WITHIN GROUP (ORDER BY salary) AS pc1,
percentile_cont(0.25) WITHIN GROUP (ORDER BY salary) FILTER (WHERE employee_name LIKE
 '%Bo%') AS pc2,
percentile_cont(0.25) WITHIN GROUP (ORDER BY salary DESC) AS pc3,
percentile_cont(0.25) WITHIN GROUP (ORDER BY salary DESC) FILTER (WHERE employee_name
 LIKE '%Bo%') AS pc4,
percentile_disc(0.25) WITHIN GROUP (ORDER BY salary) AS pd1,
percentile_disc(0.25) WITHIN GROUP (ORDER BY salary) FILTER (WHERE employee_name LIKE
 '%Bo%') AS pd2,
percentile_disc(0.25) WITHIN GROUP (ORDER BY salary DESC) AS pd3,
percentile_disc(0.25) WITHIN GROUP (ORDER BY salary DESC) FILTER (WHERE employee_name
 LIKE '%Bo%') AS pd4
FROM basic_pays
GROUP BY department
ORDER BY department;
+----------+-------+--------+-------+--------+-----+-----+-----+-----+
|department|    pc1|     pc2|    pc3|     pc4|  pd1|  pd2|  pd3|  pd4|
+----------+-------+--------+-------+--------+-----+-----+-----+-----+
|Accounting|8543.75| 7838.25| 9746.5|10260.75| 8435| 6627| 9998|11472|
|        IT|5917.75|    NULL|7381.25|    NULL| 5186| NULL| 8113| NULL|
|     Sales|8550.75|    NULL| 9721.5|    NULL| 6660| NULL|10563| NULL|
|       SCM|10449.0|10786.25|11303.0|11460.75|10449|10449|11303|11798|
+----------+-------+--------+-------+--------+-----+-----+-----+-----+
```

条件関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
coalesce(expr1, expr2, ...)	存在する場合は、最初の NULL 以外の引数を返します。それ以外の場合は null です。
if(expr1, expr2, expr3)	expr1 が true に評価された場合、expr2 を返し、それ以外の場合は expr3 を返します。
ifnull(expr1, expr2)	expr1 が null の場合、expr2 を返します。それ以外の場合は expr1 を返します。
nanvl(expr1, expr2)	NaN でない場合は expr1 を返します。それ以外の場合は expr2 を返します。
nullif(expr1, expr2)	expr1 が expr2 に等しい場合は null を返し、それ以外の場合は expr1 を返します。
nvl(expr1、expr2)	expr1 が null の場合、expr2 を返します。それ以外の場合は expr1 を返します。
nvl2(expr1, expr2, expr3)	expr1 が null でない場合は expr2 を返します。それ以外の場合は expr3 を返します。
CASE WHEN expr1 THEN expr2 [WHEN expr3 THEN expr4]* [ELSE expr5] END	expr1 = true の場合は expr2 を返し、expr3 = true の場合は expr4 を返し、それ以外の場合は expr5 を返します。

例

```
-- coalesce
SELECT coalesce(NULL, 1, NULL);
+----------------------+
|coalesce(NULL, 1, NULL)|
+----------------------+
|                     1|
+----------------------+
-- if
SELECT if(1 < 2, 'a', 'b');
+------------------+
```

```
|(IF((1 < 2), a, b))|
+-------------------+
|                  a|
+-------------------+
-- ifnull
SELECT ifnull(NULL, array('2'));
+----------------------+
|ifnull(NULL, array(2))|
+----------------------+
|                   [2]|
+----------------------+
-- nanvl
SELECT nanvl(cast('NaN' as double), 123);
+-------------------------------+
|nanvl(CAST(NaN AS DOUBLE), 123)|
+-------------------------------+
|                          123.0|
+-------------------------------+
-- nullif
SELECT nullif(2, 2);
+------------+
|nullif(2, 2)|
+------------+
|        NULL|
+------------+
-- nvl
SELECT nvl(NULL, array('2'));
+-------------------+
|nvl(NULL, array(2))|
+-------------------+
|                [2]|
+-------------------+
-- nvl2
SELECT nvl2(NULL, 2, 1);
+----------------+
|nvl2(NULL, 2, 1)|
+----------------+
|               1|
+----------------+
-- when
SELECT CASE WHEN 1 > 0 THEN 1 WHEN 2 > 0 THEN 2.0 ELSE 1.2 END;
+-----------------------------------------------------------+
|CASE WHEN (1 > 0) THEN 1 WHEN (2 > 0) THEN 2.0 ELSE 1.2 END|
+-----------------------------------------------------------+
```

```
|                                                          1.0|
+-------------------------------------------------------------+
SELECT CASE WHEN 1 < 0 THEN 1 WHEN 2 > 0 THEN 2.0 ELSE 1.2 END;
+-------------------------------------------------------------+
|CASE WHEN (1 < 0) THEN 1 WHEN (2 > 0) THEN 2.0 ELSE 1.2 END|
+-------------------------------------------------------------+
|                                                          2.0|
+-------------------------------------------------------------+
SELECT CASE WHEN 1 < 0 THEN 1 WHEN 2 < 0 THEN 2.0 END;
+-------------------------------------------------+
|CASE WHEN (1 < 0) THEN 1 WHEN (2 < 0) THEN 2.0 END|
+-------------------------------------------------+
|                                             NULL|
+-------------------------------------------------+
```

JSON 関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
from_json (jsonStr, schema[, options])	指定された `jsonStr` と `schema` を持つ構造体値を返します。
get_json_ object(js on_txt, path)	`path` から JSON オブジェクトを抽出します。
json_arra y_length(jsonArray)	最も外側の JSON 配列に含まれる要素の数を返します。

関数	説明
json_object_keys(json_object)	最も外側の JSON オブジェクトのすべてのキーを配列として返します。
json_tuple(jsonStr, p1, p2, ..., pn)	get_json_object 関数のようなタプルを返しますが、それには複数の名前が付けられます。すべての入力パラメータと出力列タイプは文字列です。
schema_of_json(json[, options])	JSON 文字列の DDL 形式でスキーマを返します。
to_json(expr[, options])	指定された構造体値を持つ JSON 文字列を返します。

例

```
-- from_json
SELECT from_json('{"a":1, "b":0.8}', 'a INT, b DOUBLE');
+---------------------------+
| from_json({"a":1, "b":0.8}) |
+---------------------------+
| {1, 0.8}                  |
+---------------------------+

SELECT from_json('{"time":"26/08/2015"}', 'time Timestamp', map('timestampFormat', 'dd/
MM/yyyy'));
+--------------------------------+
| from_json({"time":"26/08/2015"}) |
+--------------------------------+
| {2015-08-26 00:00...           |
+--------------------------------+

SELECT from_json('{"teacher": "Alice", "student": [{"name": "Bob", "rank": 1}, {"name":
 "Charlie", "rank": 2}]}', 'STRUCT<teacher: STRING, student: ARRAY<STRUCT<name: STRING,
 rank: INT>>>');
+-------------------------------------------------------------------------------------------------------
+
```

```
| from_json({"teacher": "Alice", "student": [{"name": "Bob", "rank": 1}, {"name":
 "Charlie", "rank": 2}]}) |
+-------------------------------------------------------------------------------------------------------
+
| {Alice, [{Bob, 1}...
                  |
+-------------------------------------------------------------------------------------------------------
+

-- get_json_object
SELECT get_json_object('{"a":"b"}', '$.a');
+-------------------------------+
| get_json_object({"a":"b"}, $.a) |
+-------------------------------+
| b                             |
+-------------------------------+

-- json_array_length
SELECT json_array_length('[1,2,3,4]');
+----------------------------+
| json_array_length([1,2,3,4]) |
+----------------------------+
| 4                          |
+----------------------------+

SELECT json_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4]');
+-------------------------------------------------+
| json_array_length([1,2,3,{"f1":1,"f2":[5,6]},4]) |
+-------------------------------------------------+
| 5                                               |
+-------------------------------------------------+

SELECT json_array_length('[1,2');
+----------------------+
| json_array_length([1,2) |
+----------------------+
| NULL                 |
+----------------------+

-- json_object_keys
SELECT json_object_keys('{}');
+--------------------+
| json_object_keys({}) |
+--------------------+
```

```
| []                  |
+---------------------+

SELECT json_object_keys('{"key": "value"}');
+-----------------------------------+
| json_object_keys({"key": "value"}) |
+-----------------------------------+
| [key]                             |
+-----------------------------------+

SELECT json_object_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}}');
+-----------------------------------------------------------+
| json_object_keys({"f1":"abc","f2":{"f3":"a", "f4":"b"}}) |
+-----------------------------------------------------------+
| [f1, f2]                                                  |
+-----------------------------------------------------------+

-- json_tuple
SELECT json_tuple('{"a":1, "b":2}', 'a', 'b');
+---+---+
| c0| c1|
+---+---+
|  1|  2|
+---+---+

-- schema_of_json
SELECT schema_of_json('[{"col":0}]');
+----------------------------+
| schema_of_json([{"col":0}]) |
+----------------------------+
| ARRAY<STRUCT<col:...      |
+----------------------------+

SELECT schema_of_json('[{"col":01}]', map('allowNumericLeadingZeros', 'true'));
+----------------------------+
| schema_of_json([{"col":01}]) |
+----------------------------+
| ARRAY<STRUCT<col:...       |
+----------------------------+

-- to_json
SELECT to_json(named_struct('a', 1, 'b', 2));
+---------------------------------+
| to_json(named_struct(a, 1, b, 2)) |
```

```
+--------------------------------+
| {"a":1,"b":2}                  |
+--------------------------------+

SELECT to_json(named_struct('time', to_timestamp('2015-08-26', 'yyyy-MM-dd')),
 map('timestampFormat', 'dd/MM/yyyy'));
+------------------------------------------------------------------+
| to_json(named_struct(time, to_timestamp(2015-08-26, yyyy-MM-dd))) |
+------------------------------------------------------------------+
| {"time":"26/08/20...                                             |
+------------------------------------------------------------------+

SELECT to_json(array(named_struct('a', 1, 'b', 2)));
+---------------------------------------+
| to_json(array(named_struct(a, 1, b, 2))) |
+---------------------------------------+
| [{"a":1,"b":2}]                       |
+---------------------------------------+

SELECT to_json(map('a', named_struct('b', 1)));
+-----------------------------------+
| to_json(map(a, named_struct(b, 1))) |
+-----------------------------------+
| {"a":{"b":1}}                     |
+-----------------------------------+

SELECT to_json(map(named_struct('a', 1),named_struct('b', 2)));
+-----------------------------------------------------+
| to_json(map(named_struct(a, 1), named_struct(b, 2))) |
+-----------------------------------------------------+
| {"[1]":{"b":2}}                                     |
+-----------------------------------------------------+

SELECT to_json(map('a', 1));
+------------------+
| to_json(map(a, 1)) |
+------------------+
| {"a":1}          |
+------------------+

SELECT to_json(array(map('a', 1)));
+-------------------------+
| to_json(array(map(a, 1))) |
+-------------------------+
```

```
| [{"a":1}]               |
+-------------------------+
```

配列関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
array(expr, ...)	指定された要素を持つ配列を返します。
array_append(array, element)	最初の引数として渡された配列の末尾に要素を追加します。要素のタイプは、配列の要素のタイプと似ている必要があります。Null 要素も配列に追加されます。ただし、渡された配列が NULL の場合、出力も NULL になります
array_compact(array)	配列から null 値を削除します。
array_contains(array, value)	配列にその値が含まれている場合は true を返します。
array_distinct(array)	配列から重複した値を削除します。
array_except(array1, array2)	array1 に含まれ、array2 には含まれていない要素を重複なしで返します。
array_insert(x, pos, val)	配列 x のインデックス位置内に val を配置します。配列のインデックスは 1 から始まります。最大負のインデックスは -1 で、関数は現在の最後の要素の後に新しい要素を挿入します。インデックスが配列サイズを超える場合は配列の末尾に、インデックスが負の場合は配列の先頭に「null」要素を追加します。

関数	説明
array_intersect(array1, array2)	array1 と array2 の共通集合にある要素の配列を重複なしで返します。
array_join(array, delimiter[, nullReplacement])	区切り文字とオプションの文字列を使用して特定の配列の要素を連結し、null を置き換えます。nullReplacement に値が設定されていない場合、null 値はフィルタリングされます。
array_max(array)	配列の最大値を返します。NaN は、double/float 型について NaN 以外の要素よりも大きくなります。NULL 要素はスキップされます。
array_min(array)	配列内の最小値を返します。NaN は、double/float 型について NaN 以外の要素よりも大きくなります。NULL 要素はスキップされます。
array_position(array, element)	配列の最初に見つかった一致要素の (1 ベース) インデックスを、logn 型で返します。一致する要素が見つからない場合は 0 を返します。
array_prepend(array, element)	最初の引数として渡された配列の先頭に要素を追加します。要素のタイプは、配列の要素のタイプと同じである必要があります。Null 要素も配列の先頭に付け加えられます。ただし、渡された配列が NULL の場合、出力も NULL になります
array_remove(array, element)	配列から要素と等しいすべての要素を削除します。
array_repeat(element, count)	要素のカウント時間を含む配列を返します。
array_union(array1, array2)	array1 と array2 の和集合内の要素の配列を重複せずに返します。

関数	説明
arrays_overlap(a1, a2)	a1 に、a2 存在する NULL 以外の要素が少なくとも 1 つ含まれている場合は true を返します。配列に共通の要素がなく、両方とも空ではなく、どちらかに null 要素が含まれている場合は null が返され、それ以外の場合は false が返されます。
arrays_zip(a1, a2, ...)	入力配列のすべての N 番目の値を含む N 番目の構造体を要素とする構造体のマージされた配列を返します。
flatten(arrayOfArrays)	配列の配列を 1 つの配列に変換します。
get(array, index)	指定された (0 ベースの) インデックスで配列の要素を返します。インデックスが配列の境界外を指す場合、この関数は NULL を返します。
sequence(start, stop, step)	開始値から終了値まで (終了値を含む)、段階的に増分させながら要素の配列を生成します。戻り値の要素の型は、引数式の型と同じです。サポートされている型は、byte、short、integer、long、date、timestamp です。開始式と終了式は同じ型に解決する必要があります。開始式と終了式が 'date' 型または 'timestamp' 型に解決される場合、ステップ式は 'interval' 型、'year-month interval' 型、または 'day-time interval' 型に解決する必要があります。それ以外の場合は、開始式および終了式と同じ型に解決する必要があります。
shuffle(array)	指定された配列のランダム順列を返します。

関数	説明
slice(x, start, length)	指定された長さで、インデックスの開始から始まる配列 x の部分集合を抽出します (配列インデックスは 1 から始まり、開始が負の場合は末尾から始まります)。
sort_array(array[, ascendingOrder])	配列要素の自然な順序に従って、入力配列を昇順または降順にソートします。NaN は、double/float 型について NaN 以外の要素よりも大きくなります。Null 要素は、返された配列の先頭に昇順で配置されるか、または返された配列の末尾に降順で配置されます。

例

```
-- array
SELECT array(1, 2, 3);
+--------------+
|array(1, 2, 3)|
+--------------+
|     [1, 2, 3]|
+--------------+
-- array_append
SELECT array_append(array('b', 'd', 'c', 'a'), 'd');
+----------------------------------+
|array_append(array(b, d, c, a), d)|
+----------------------------------+
|                   [b, d, c, a, d]|
+----------------------------------+
SELECT array_append(array(1, 2, 3, null), null);
+---------------------------------------+
|array_append(array(1, 2, 3, NULL), NULL)|
+---------------------------------------+
|                    [1, 2, 3, NULL, N...|
+---------------------------------------+
SELECT array_append(CAST(null as Array<Int>), 2);
+--------------------+
|array_append(NULL, 2)|
+--------------------+
```

```
|                 NULL|
+---------------------+
-- array_compact
SELECT array_compact(array(1, 2, 3, null));
+-----------------------------------+
|array_compact(array(1, 2, 3, NULL))|
+-----------------------------------+
|                          [1, 2, 3]|
+-----------------------------------+
SELECT array_compact(array("a", "b", "c"));
+----------------------------+
|array_compact(array(a, b, c))|
+----------------------------+
|                    [a, b, c]|
+----------------------------+
-- array_contains
SELECT array_contains(array(1, 2, 3), 2);
+---------------------------------+
|array_contains(array(1, 2, 3), 2)|
+---------------------------------+
|                             true|
+---------------------------------+
-- array_distinct
SELECT array_distinct(array(1, 2, 3, null, 3));
+--------------------------------------+
|array_distinct(array(1, 2, 3, NULL, 3))|
+--------------------------------------+
|                        [1, 2, 3, NULL]|
+--------------------------------------+
-- array_except
SELECT array_except(array(1, 2, 3), array(1, 3, 5));
+-------------------------------------------+
|array_except(array(1, 2, 3), array(1, 3, 5))|
+-------------------------------------------+
|                                        [2]|
+-------------------------------------------+
-- array_insert
SELECT array_insert(array(1, 2, 3, 4), 5, 5);
+-------------------------------------+
|array_insert(array(1, 2, 3, 4), 5, 5)|
+-------------------------------------+
|                     [1, 2, 3, 4, 5]|
+-------------------------------------+
SELECT array_insert(array(5, 4, 3, 2), -1, 1);
```

```
+-------------------------------------+
|array_insert(array(5, 4, 3, 2), -1, 1)|
+-------------------------------------+
|                     [5, 4, 3, 2, 1]|
+-------------------------------------+
SELECT array_insert(array(5, 3, 2, 1), -4, 4);
+-------------------------------------+
|array_insert(array(5, 3, 2, 1), -4, 4)|
+-------------------------------------+
|                     [5, 4, 3, 2, 1]|
+-------------------------------------+
-- array_intersect
SELECT array_intersect(array(1, 2, 3), array(1, 3, 5));
+-----------------------------------------------+
|array_intersect(array(1, 2, 3), array(1, 3, 5))|
+-----------------------------------------------+
|                                         [1, 3]|
+-----------------------------------------------+
-- array_join
SELECT array_join(array('hello', 'world'), ' ');
+---------------------------------+
|array_join(array(hello, world),  )|
+---------------------------------+
|                     hello world|
+---------------------------------+
SELECT array_join(array('hello', null ,'world'), ' ');
+--------------------------------------+
|array_join(array(hello, NULL, world),  )|
+--------------------------------------+
|                           hello world|
+--------------------------------------+
SELECT array_join(array('hello', null ,'world'), ' ', ',');
+-----------------------------------------+
|array_join(array(hello, NULL, world),  , ,)|
+-----------------------------------------+
|                            hello , world|
+-----------------------------------------+
-- array_max
SELECT array_max(array(1, 20, null, 3));
+-------------------------------+
|array_max(array(1, 20, NULL, 3))|
+-------------------------------+
|                             20|
+-------------------------------+
```

```
-- array_min
SELECT array_min(array(1, 20, null, 3));
+------------------------------+
|array_min(array(1, 20, NULL, 3))|
+------------------------------+
|                             1|
+------------------------------+
-- array_position
SELECT array_position(array(312, 773, 708, 708), 708);
+-----------------------------------------------+
|array_position(array(312, 773, 708, 708), 708)|
+-----------------------------------------------+
|                                              3|
+-----------------------------------------------+
SELECT array_position(array(312, 773, 708, 708), 414);
+-----------------------------------------------+
|array_position(array(312, 773, 708, 708), 414)|
+-----------------------------------------------+
|                                              0|
+-----------------------------------------------+
-- array_prepend
SELECT array_prepend(array('b', 'd', 'c', 'a'), 'd');
+----------------------------------+
|array_prepend(array(b, d, c, a), d)|
+----------------------------------+
|                    [d, b, d, c, a]|
+----------------------------------+
SELECT array_prepend(array(1, 2, 3, null), null);
+---------------------------------------+
|array_prepend(array(1, 2, 3, NULL), NULL)|
+---------------------------------------+
|                     [NULL, 1, 2, 3, N...|
+---------------------------------------+
SELECT array_prepend(CAST(null as Array<Int>), 2);
+----------------------+
|array_prepend(NULL, 2)|
+----------------------+
|                  NULL|
+----------------------+
-- array_remove
SELECT array_remove(array(1, 2, 3, null, 3), 3);
+--------------------------------------+
|array_remove(array(1, 2, 3, NULL, 3), 3)|
+--------------------------------------+
```

```
|                            [1, 2, NULL]|
+----------------------------------------+
-- array_repeat
SELECT array_repeat('123', 2);
+--------------------+
|array_repeat(123, 2)|
+--------------------+
|          [123, 123]|
+--------------------+
-- array_union
SELECT array_union(array(1, 2, 3), array(1, 3, 5));
+-----------------------------------------+
|array_union(array(1, 2, 3), array(1, 3, 5))|
+-----------------------------------------+
|                               [1, 2, 3, 5]|
+-----------------------------------------+
-- arrays_overlap
SELECT arrays_overlap(array(1, 2, 3), array(3, 4, 5));
+------------------------------------------------+
|arrays_overlap(array(1, 2, 3), array(3, 4, 5))|
+------------------------------------------------+
|                                            true|
+------------------------------------------------+
-- arrays_zip
SELECT arrays_zip(array(1, 2, 3), array(2, 3, 4));
+----------------------------------------+
|arrays_zip(array(1, 2, 3), array(2, 3, 4))|
+----------------------------------------+
|                      [{1, 2}, {2, 3}, ...|
+----------------------------------------+
SELECT arrays_zip(array(1, 2), array(2, 3), array(3, 4));
+-------------------------------------------------+
|arrays_zip(array(1, 2), array(2, 3), array(3, 4))|
+-------------------------------------------------+
|                             [{1, 2, 3}, {2, 3...|
+-------------------------------------------------+
-- flatten
SELECT flatten(array(array(1, 2), array(3, 4)));
+---------------------------------------+
|flatten(array(array(1, 2), array(3, 4)))|
+---------------------------------------+
|                            [1, 2, 3, 4]|
+---------------------------------------+
-- get
```

```
SELECT get(array(1, 2, 3), 0);
+----------------------+
|get(array(1, 2, 3), 0)|
+----------------------+
|                     1|
+----------------------+
SELECT get(array(1, 2, 3), 3);
+----------------------+
|get(array(1, 2, 3), 3)|
+----------------------+
|                  NULL|
+----------------------+
SELECT get(array(1, 2, 3), -1);
+-----------------------+
|get(array(1, 2, 3), -1)|
+-----------------------+
|                   NULL|
+-----------------------+
-- sequence
SELECT sequence(1, 5);
+---------------+
| sequence(1, 5)|
+---------------+
|[1, 2, 3, 4, 5]|
+---------------+
SELECT sequence(5, 1);
+---------------+
| sequence(5, 1)|
+---------------+
|[5, 4, 3, 2, 1]|
+---------------+
SELECT sequence(to_date('2018-01-01'), to_date('2018-03-01'), interval 1 month);
+------------------------------------------------------------------------+
|sequence(to_date(2018-01-01), to_date(2018-03-01), INTERVAL '1' MONTH)|
+------------------------------------------------------------------------+
|                                                    [2018-01-01, 2018...|
+------------------------------------------------------------------------+
SELECT sequence(to_date('2018-01-01'), to_date('2018-03-01'), interval '0-1' year to
 month);
+-----------------------------------------------------------------------------------+
|sequence(to_date(2018-01-01), to_date(2018-03-01), INTERVAL '0-1' YEAR TO MONTH)|
+-----------------------------------------------------------------------------------+
|                                                                 [2018-01-01, 2018...|
+-----------------------------------------------------------------------------------+
```

```
-- shuffle
SELECT shuffle(array(1, 20, 3, 5));
+----------------------------+
|shuffle(array(1, 20, 3, 5))|
+----------------------------+
|              [5, 1, 20, 3]|
+----------------------------+
SELECT shuffle(array(1, 20, null, 3));
+-----------------------------+
|shuffle(array(1, 20, NULL, 3))|
+-----------------------------+
|              [1, NULL, 20, 3]|
+-----------------------------+
-- slice
SELECT slice(array(1, 2, 3, 4), 2, 2);
+-----------------------------+
|slice(array(1, 2, 3, 4), 2, 2)|
+-----------------------------+
|                        [2, 3]|
+-----------------------------+
SELECT slice(array(1, 2, 3, 4), -2, 2);
+------------------------------+
|slice(array(1, 2, 3, 4), -2, 2)|
+------------------------------+
|                         [3, 4]|
+------------------------------+
-- sort_array
SELECT sort_array(array('b', 'd', null, 'c', 'a'), true);
+-----------------------------------------+
|sort_array(array(b, d, NULL, c, a), true)|
+-----------------------------------------+
|                      [NULL, a, b, c, d]|
+-----------------------------------------+
```

Window 関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

ウィンドウ関数は、ウィンドウと呼ばれる行の集合に対して動作し、その集合に基づいて各行ごとに戻り値を計算します。ウィンドウ関数は、移動平均の計算、累積統計の計算、現在の行から相対的な位置にある行の値にアクセスするなどのタスク処理に有用です。

[Syntax] (構文)

```
window_function [ nulls_option ] OVER ( [ { PARTITION | DISTRIBUTE } BY
 partition_col_name = partition_col_val ( [ , ... ] ) ] { ORDER | SORT } BY expression
 [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [ , ... ] [ window_frame ] )
```

パラメータ

- ランク付け関数

  構文: `RANK | DENSE_RANK | PERCENT_RANK | NTILE | ROW_NUMBER`

  分析関数

  構文: `CUME_DIST | LAG | LEAD | NTH_VALUE | FIRST_VALUE | LAST_VALUE`

  集計関数

  構文: `MAX | MIN | COUNT | SUM | AVG | ...`
- `nulls_option` – ウィンドウ関数を評価するときに null 値をスキップするかどうかを指定します。RESPECT NULLS は null 値をスキップしないことを意味し、IGNORE NULLS はスキップすることを意味します。指定しない場合、デフォルトは RESPECT NULLS です。

  構文: `{ IGNORE | RESPECT } NULLS`

  注: `Only LAG | LEAD | NTH_VALUE | FIRST_VALUE | LAST_VALUE は IGNORE NULLS で使用できます。`
- `window_frame` – ウィンドウを開始する行とウィンドウを終了する行を指定します。

  構文: `{ RANGE | ROWS } { frame_start | BETWEEN frame_start AND frame_end }`

  frame_start と frame_end には次の構文があります。

  構文: `UNBOUNDED PRECEDING | offset PRECEDING | CURRENT ROW | offset FOLLOWING | UNBOUNDED FOLLOWING`

  offset: 現在の行の位置からのオフセットを指定します。

注 frame_end を省略すると、デフォルトで CURRENT ROW になります。

例

```
CREATE TABLE employees (name STRING, dept STRING, salary INT, age INT);
INSERT INTO employees VALUES ("Lisa", "Sales", 10000, 35);
INSERT INTO employees VALUES ("Evan", "Sales", 32000, 38);
INSERT INTO employees VALUES ("Fred", "Engineering", 21000, 28);
INSERT INTO employees VALUES ("Alex", "Sales", 30000, 33);
INSERT INTO employees VALUES ("Tom", "Engineering", 23000, 33);
INSERT INTO employees VALUES ("Jane", "Marketing", 29000, 28);
INSERT INTO employees VALUES ("Jeff", "Marketing", 35000, 38);
INSERT INTO employees VALUES ("Paul", "Engineering", 29000, 23);
INSERT INTO employees VALUES ("Chloe", "Engineering", 23000, 25);
SELECT * FROM employees;
+-----+-----------+------+-----+
| name|       dept|salary|  age|
+-----+-----------+------+-----+
|Chloe|Engineering| 23000|   25|
| Fred|Engineering| 21000|   28|
| Paul|Engineering| 29000|   23|
|Helen|  Marketing| 29000|   40|
|  Tom|Engineering| 23000|   33|
| Jane|  Marketing| 29000|   28|
| Jeff|  Marketing| 35000|   38|
| Evan|      Sales| 32000|   38|
| Lisa|      Sales| 10000|   35|
| Alex|      Sales| 30000|   33|
+-----+-----------+------+-----+
SELECT name, dept, salary, RANK() OVER (PARTITION BY dept ORDER BY salary) AS rank FROM
 employees;
+-----+-----------+------+----+
| name|       dept|salary|rank|
+-----+-----------+------+----+
| Lisa|      Sales| 10000|   1|
| Alex|      Sales| 30000|   2|
| Evan|      Sales| 32000|   3|
| Fred|Engineering| 21000|   1|
|  Tom|Engineering| 23000|   2|
|Chloe|Engineering| 23000|   2|
| Paul|Engineering| 29000|   4|
|Helen|  Marketing| 29000|   1|
```

```
| Jane|  Marketing| 29000|   1|
| Jeff|  Marketing| 35000|   3|
+-----+-----------+------+----+
SELECT name, dept, salary, DENSE_RANK() OVER (PARTITION BY dept ORDER BY salary ROWS
 BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS dense_rank FROM employees;
+-----+-----------+------+----------+
| name|       dept|salary|dense_rank|
+-----+-----------+------+----------+
| Lisa|      Sales| 10000|         1|
| Alex|      Sales| 30000|         2|
| Evan|      Sales| 32000|         3|
| Fred|Engineering| 21000|         1|
|  Tom|Engineering| 23000|         2|
|Chloe|Engineering| 23000|         2|
| Paul|Engineering| 29000|         3|
|Helen|  Marketing| 29000|         1|
| Jane|  Marketing| 29000|         1|
| Jeff|  Marketing| 35000|         2|
+-----+-----------+------+----------+
SELECT name, dept, age, CUME_DIST() OVER (PARTITION BY dept ORDER BY age
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cume_dist FROM employees;
+-----+-----------+------+------------------+
| name|       dept|age   |         cume_dist|
+-----+-----------+------+------------------+
| Alex|      Sales|    33|0.3333333333333333|
| Lisa|      Sales|    35|0.6666666666666666|
| Evan|      Sales|    38|               1.0|
| Paul|Engineering|    23|              0.25|
|Chloe|Engineering|    25|              0.75|
| Fred|Engineering|    28|              0.25|
|  Tom|Engineering|    33|               1.0|
| Jane|  Marketing|    28|0.3333333333333333|
| Jeff|  Marketing|    38|0.6666666666666666|
|Helen|  Marketing|    40|               1.0|
+-----+-----------+------+------------------+
SELECT name, dept, salary, MIN(salary) OVER (PARTITION BY dept ORDER BY salary) AS min
FROM employees;
+-----+-----------+------+-----+
| name|       dept|salary|  min|
+-----+-----------+------+-----+
| Lisa|      Sales| 10000|10000|
| Alex|      Sales| 30000|10000|
| Evan|      Sales| 32000|10000|
```

```
|Helen|  Marketing| 29000|29000|
| Jane|  Marketing| 29000|29000|
| Jeff|  Marketing| 35000|29000|
| Fred|Engineering| 21000|21000|
|  Tom|Engineering| 23000|21000|
|Chloe|Engineering| 23000|21000|
| Paul|Engineering| 29000|21000|
+-----+-----------+------+-----+
SELECT name, salary,
LAG(salary) OVER (PARTITION BY dept ORDER BY salary) AS lag,
LEAD(salary, 1, 0) OVER (PARTITION BY dept ORDER BY salary) AS lead
FROM employees;
+-----+-----------+------+-----+-----+
| name|       dept|salary|  lag| lead|
+-----+-----------+------+-----+-----+
| Lisa|      Sales| 10000|NULL |30000|
| Alex|      Sales| 30000|10000|32000|
| Evan|      Sales| 32000|30000|    0|
| Fred|Engineering| 21000| NULL|23000|
|Chloe|Engineering| 23000|21000|23000|
|  Tom|Engineering| 23000|23000|29000|
| Paul|Engineering| 29000|23000|    0|
|Helen|  Marketing| 29000| NULL|29000|
| Jane|  Marketing| 29000|29000|35000|
| Jeff|  Marketing| 35000|29000|    0|
+-----+-----------+------+-----+-----+
SELECT id, v,
LEAD(v, 0) IGNORE NULLS OVER w lead,
LAG(v, 0) IGNORE NULLS OVER w lag,
NTH_VALUE(v, 2) IGNORE NULLS OVER w nth_value,
FIRST_VALUE(v) IGNORE NULLS OVER w first_value,
LAST_VALUE(v) IGNORE NULLS OVER w last_value
FROM test_ignore_null
WINDOW w AS (ORDER BY id)
ORDER BY id;
+--+----+----+----+---------+-----------+----------+
|id|   v|lead| lag|nth_value|first_value|last_value|
+--+----+----+----+---------+-----------+----------+
| 0|NULL|NULL|NULL|     NULL|       NULL|      NULL|
| 1|   x|   x|   x|     NULL|          x|         x|
| 2|NULL|NULL|NULL|     NULL|          x|         x|
| 3|NULL|NULL|NULL|     NULL|          x|         x|
| 4|   y|   y|   y|        y|          x|         y|
| 5|NULL|NULL|NULL|        y|          x|         y|
```

```
| 6|   z|   z|   z|        y|          x|         z|
| 7|   v|   v|   v|        y|          x|         v|
| 8|NULL|NULL|NULL|        y|          x|         v|
+--+----+----+----+---------+-----------+----------+
```

変換関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
bigint(expr)	値 `expr` をターゲットデータ型 `bigint` にキャストします。
binary(expr)	値 `expr` をターゲットデータ型 `binary` にキャストします。
boolean(expr)	値 `expr` をターゲットデータ型 `boolean` にキャストします。
cast(expr AS type)	値 `expr` をターゲットデータ型 `type` にキャストします。
date(expr)	値 `expr` をターゲットデータ型 `date` にキャストします。
decimal(expr)	値 `expr` をターゲットデータ型 `decimal` にキャストします。
double(expr)	値 `expr` をターゲットデータ型 `double` にキャストします。
float(expr)	値 `expr` をターゲットデータ型 `float` にキャストします。
int(expr)	値 `expr` をターゲットデータ型 `int` にキャストします。
smallint(expr)	値 `expr` をターゲットデータ型 `smallint` にキャストします。
string(expr)	値 `expr` をターゲットデータ型 `string` にキャストします。
timestamp(expr)	値 `expr` をターゲットデータ型 `timestamp` にキャストします。
tinyint(expr)	値 `expr` をターゲットデータ型 `tinyint` にキャストします。

例

```
-- cast
SELECT cast(field as int);
+---------------+
|CAST(field AS INT)|
+---------------+
|             10|
+---------------+
```

述語関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
! expr	論理否定。
expr1 < expr2	`expr1` が `expr2` より小さい場合は true を返します。
expr1 <= expr2	`expr1` が `expr2` 以下の場合は true を返します。
expr1 <=> expr2	null 以外のオペランドの EQUAL(=) 演算子と同じ結果を返しますが、両方が null の場合は true、いずれかが null の場合は false を返します。
expr1 = expr2	`expr1` が `expr2` に等しい場合は true、それ以外の場合は false を返します。
expr1 == expr2	`expr1` が `expr2` に等しい場合は true、それ以外の場合は false を返します。
expr1 > expr2	`expr1` が `expr2` より大きい場合は true を返します。
expr1 >= expr2	`expr1` が `expr2` 以上の場合は true を返します。

関数	説明
expr1 and expr2	論理 AND
str ilike pattern[ESCAPE escape]	str が大文字と小文字を区別せずに `pattern` に `escape` で一致する場合は true、引数のいずれかが null の場合は null、それ以外の場合は false を返します。
expr1 in(expr2, expr3, ...)	`expr` が任意の valN に等しい場合は true を返します。
isnan(expr)	`expr` が NaN の場合は true、それ以外の場合は false を返します。
isnotnull(expr)	`expr` が null でない場合は true、それ以外の場合は false を返します。
isnull(expr)	`expr` が null の場合は true、それ以外の場合は false を返します。
str like pattern[ESCAPE escape]	str が `pattern` に `escape` で一致する場合は true、引数のいずれかが null の場合は null、それ以外の場合は false を返します。
not expr	論理否定。
expr1 or expr2	論理 OR
regexp(str, regexp)	`str` が `regexp` と一致する場合は true、それ以外の場合は false を返します。
regexp_like(str, regexp)	`str` が `regexp` と一致する場合は true、それ以外の場合は false を返します。
rlike(str, regexp)	`str` が `regexp` と一致する場合は true、それ以外の場合は false を返します。

例

```
-- !
SELECT ! true;
+-----------+
|(NOT true)|
+-----------+
```

```
|     false|
+----------+
SELECT ! false;
+-----------+
|(NOT false)|
+-----------+
|       true|
+-----------+
SELECT ! NULL;
+----------+
|(NOT NULL)|
+----------+
|      NULL|
+----------+
-- <
SELECT to_date('2009-07-30 04:17:52') < to_date('2009-07-30 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) < to_date(2009-07-30 04:17:52))|
+-------------------------------------------------------------+
|                                                        false|
+-------------------------------------------------------------+
SELECT to_date('2009-07-30 04:17:52') < to_date('2009-08-01 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) < to_date(2009-08-01 04:17:52))|
+-------------------------------------------------------------+
|                                                         true|
+-------------------------------------------------------------+
SELECT 1 < NULL;
+----------+
|(1 < NULL)|
+----------+
|      NULL|
+----------+
-- <=
SELECT 2 <= 2;
+--------+
|(2 <= 2)|
+--------+
|    true|
+--------+
SELECT 1.0 <= '1';
+----------+
|(1.0 <= 1)|
+----------+
```

```
|      true|
+----------+
SELECT to_date('2009-07-30 04:17:52') <= to_date('2009-07-30 04:17:52');
+------------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) <= to_date(2009-07-30 04:17:52))|
+------------------------------------------------------------------+
|                                                              true|
+------------------------------------------------------------------+
SELECT to_date('2009-07-30 04:17:52') <= to_date('2009-08-01 04:17:52');
+------------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) <= to_date(2009-08-01 04:17:52))|
+------------------------------------------------------------------+
|                                                              true|
+------------------------------------------------------------------+
SELECT 1 <= NULL;
+-----------+
|(1 <= NULL)|
+-----------+
|       NULL|
+-----------+
-- <=>
SELECT 2 <=> 2;
+---------+
|(2 <=> 2)|
+---------+
|     true|
+---------+
SELECT 1 <=> '1';
+---------+
|(1 <=> 1)|
+---------+
|     true|
+---------+
SELECT true <=> NULL;
+---------------+
|(true <=> NULL)|
+---------------+
|          false|
+---------------+
SELECT NULL <=> NULL;
+---------------+
|(NULL <=> NULL)|
+---------------+
|           true|
```

```
+---------------+
-- =
SELECT 2 = 2;
+-------+
|(2 = 2)|
+-------+
|   true|
+-------+
SELECT 1 = '1';
+-------+
|(1 = 1)|
+-------+
|   true|
+-------+
SELECT true = NULL;
+-------------+
|(true = NULL)|
+-------------+
|         NULL|
+-------------+
SELECT NULL = NULL;
+-------------+
|(NULL = NULL)|
+-------------+
|         NULL|
+-------------+
-- ==
SELECT 2 == 2;
+-------+
|(2 = 2)|
+-------+
|   true|
+-------+
SELECT 1 == '1';
+-------+
|(1 = 1)|
+-------+
|   true|
+-------+
SELECT true == NULL;
+-------------+
|(true = NULL)|
+-------------+
|         NULL|
```

```
+-------------+
SELECT NULL == NULL;
+-------------+
|(NULL = NULL)|
+-------------+
|         NULL|
+-------------+
-- >
SELECT 2 > 1;
+-------+
|(2 > 1)|
+-------+
|   true|
+-------+
SELECT 2 > 1.1;
+-------+
|(2 > 1)|
+-------+
|   true|
+-------+
SELECT to_date('2009-07-30 04:17:52') > to_date('2009-07-30 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) > to_date(2009-07-30 04:17:52))|
+-------------------------------------------------------------+
|                                                        false|
+-------------------------------------------------------------+
SELECT to_date('2009-07-30 04:17:52') > to_date('2009-08-01 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) > to_date(2009-08-01 04:17:52))|
+-------------------------------------------------------------+
|                                                        false|
+-------------------------------------------------------------+
SELECT 1 > NULL;
+----------+
|(1 > NULL)|
+----------+
|      NULL|
+----------+
-- >=
SELECT 2 >= 1;
+--------+
|(2 >= 1)|
+--------+
|    true|
```

```
+--------+
SELECT 2.0 >= '2.1';
+------------+
|(2.0 >= 2.1)|
+------------+
|       false|
+------------+
SELECT to_date('2009-07-30 04:17:52') >= to_date('2009-07-30 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) >= to_date(2009-07-30 04:17:52))|
+-------------------------------------------------------------+
|                                                         true|
+-------------------------------------------------------------+
SELECT to_date('2009-07-30 04:17:52') >= to_date('2009-08-01 04:17:52');
+-------------------------------------------------------------+
|(to_date(2009-07-30 04:17:52) >= to_date(2009-08-01 04:17:52))|
+-------------------------------------------------------------+
|                                                        false|
+-------------------------------------------------------------+
SELECT 1 >= NULL;
+-----------+
|(1 >= NULL)|
+-----------+
|       NULL|
+-----------+
-- and
SELECT true and true;
+---------------+
|(true AND true)|
+---------------+
|           true|
+---------------+
SELECT true and false;
+----------------+
|(true AND false)|
+----------------+
|           false|
+----------------+
SELECT true and NULL;
+---------------+
|(true AND NULL)|
+---------------+
|           NULL|
+---------------+
```

```
SELECT false and NULL;
+----------------+
|(false AND NULL)|
+----------------+
|           false|
+----------------+
-- ilike
SELECT ilike('Wagon', '_Agon');
+------------------+
|ilike(Wagon, _Agon)|
+------------------+
|              true|
+------------------+
SELECT '%SystemDrive%\Users\John' ilike '\%SystemDrive\%\\users%';
+-----------------------------------------------------------+
|ilike(%SystemDrive%\Users\John, \%SystemDrive\%\\users%)|
+-----------------------------------------------------------+
|                                                       true|
+-----------------------------------------------------------+
SELECT '%SystemDrive%\\USERS\\John' ilike '\%SystemDrive\%\\\\Users%';
+-----------------------------------------------------------+
|ilike(%SystemDrive%\USERS\John, \%SystemDrive\%\\Users%)|
+-----------------------------------------------------------+
|                                                       true|
+-----------------------------------------------------------+
SELECT '%SystemDrive%/Users/John' ilike '/%SYSTEMDrive/%//Users%' ESCAPE '/';
+-----------------------------------------------------------+
|ilike(%SystemDrive%/Users/John, /%SYSTEMDrive/%//Users%)|
+-----------------------------------------------------------+
|                                                       true|
+-----------------------------------------------------------+
-- in
SELECT 1 in(1, 2, 3);
+----------------+
|(1 IN (1, 2, 3))|
+----------------+
|            true|
+----------------+
SELECT 1 in(2, 3, 4);
+----------------+
|(1 IN (2, 3, 4))|
+----------------+
|           false|
+----------------+
```

```
SELECT named_struct('a', 1, 'b', 2) in(named_struct('a', 1, 'b', 1), named_struct('a',
 1, 'b', 3));
+-----------------------------------------------------------------------------------+
|(named_struct(a, 1, b, 2) IN (named_struct(a, 1, b, 1), named_struct(a, 1, b, 3)))|
+-----------------------------------------------------------------------------------+
|                                                                              false|
+-----------------------------------------------------------------------------------+
SELECT named_struct('a', 1, 'b', 2) in(named_struct('a', 1, 'b', 2), named_struct('a',
 1, 'b', 3));
+-----------------------------------------------------------------------------------+
|(named_struct(a, 1, b, 2) IN (named_struct(a, 1, b, 2), named_struct(a, 1, b, 3)))|
+-----------------------------------------------------------------------------------+
|                                                                               true|
+-----------------------------------------------------------------------------------+
-- isnan
SELECT isnan(cast('NaN' as double));
+--------------------------+
|isnan(CAST(NaN AS DOUBLE))|
+--------------------------+
|                      true|
+--------------------------+
-- isnotnull
SELECT isnotnull(1);
+---------------+
|(1 IS NOT NULL)|
+---------------+
|           true|
+---------------+
-- isnull
SELECT isnull(1);
+-----------+
|(1 IS NULL)|
+-----------+
|      false|
+-----------+
-- like
SELECT like('Wagon', '_Agon');
+----------------+
|Wagon LIKE _Agon|
+----------------+
|            true|
+----------------+
-- not
SELECT not true;
```

```
+----------+
|(NOT true)|
+----------+
|     false|
+----------+
SELECT not false;
+-----------+
|(NOT false)|
+-----------+
|       true|
+-----------+
SELECT not NULL;
+----------+
|(NOT NULL)|
+----------+
|      NULL|
+----------+
-- or
SELECT true or false;
+---------------+
|(true OR false)|
+---------------+
|           true|
+---------------+
SELECT false or false;
+----------------+
|(false OR false)|
+----------------+
|           false|
+----------------+
SELECT true or NULL;
+--------------+
|(true OR NULL)|
+--------------+
|          true|
+--------------+
SELECT false or NULL;
+---------------+
|(false OR NULL)|
+---------------+
|           NULL|
+---------------+
```

マップ関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
element_at(array, index)	指定された (1 ベースの) インデックスで配列の要素を返します。
element_at(map, key)	指定されたキーの値を返します。キーがマップに含まれていない場合、関数は NULL を返します。
map(key0, value0, key1, value1, ...)	指定されたキーと値のペアを持つマップを作成します。
map_concat(map, ...)	指定されたすべてのマップの和集合を返します。
map_contains_key(map, key)	マップ内にキーがある場合は true を返します。
map_entries(map)	指定されたマップ内のすべてのエントリの順序なし配列を返します。
map_from_arrays(keys, values)	指定されたキー/値の配列のペアを持つマップを作成します。キー内のすべての要素は NULL であってはなりません。
map_from_entries(arrayOfEntries)	指定されたエントリの配列から作成されたマップを返します。
map_keys(map)	マップのキーを含む順序なしの配列を返します。

関数	説明
map_values(map)	マップの値を含む順序なしの配列を返します。
str_to_map(text[, pairDelim[, keyValueDelim]])	区切り文字を使用してテキストをキーと値のペアに分割した後にマップを作成します。デフォルトの区切り文字は、`pairDelim` の場合は ','、`keyValueDelim` の場合は ':' です。`pairDelim` と `keyValueDelim` はどちらも正規表現として扱われます。
try_element_at(array, index)	指定された (1 ベースの) インデックスで配列の要素を返します。インデックスが 0 の場合、システムはエラーをスローします。インデックスが 0 より小さい場合、末尾から先頭に向かって要素にアクセスします。インデックスが配列の長さを超える場合、関数は常に NULL を返します。
try_element_at(map, key)	指定されたキーの値を返します。キーがマップに含まれていない場合、関数は必ず NULL を返します。

例

```
-- element_at
SELECT element_at(array(1, 2, 3), 2);
+----------------------------+
|element_at(array(1, 2, 3), 2)|
+----------------------------+
|                           2|
+----------------------------+
SELECT element_at(map(1, 'a', 2, 'b'), 2);
+-----------------------------+
|element_at(map(1, a, 2, b), 2)|
+-----------------------------+
|                            b|
+-----------------------------+
```

```
-- map
SELECT map(1.0, '2', 3.0, '4');
+--------------------+
| map(1.0, 2, 3.0, 4)|
+--------------------+
|{1.0 -> 2, 3.0 -> 4}|
+--------------------+
-- map_concat
SELECT map_concat(map(1, 'a', 2, 'b'), map(3, 'c'));
+--------------------------------------+
|map_concat(map(1, a, 2, b), map(3, c))|
+--------------------------------------+
|                  {1 -> a, 2 -> b, ...|
+--------------------------------------+
-- map_contains_key
SELECT map_contains_key(map(1, 'a', 2, 'b'), 1);
+-----------------------------------+
|map_contains_key(map(1, a, 2, b), 1)|
+-----------------------------------+
|                               true|
+-----------------------------------+
SELECT map_contains_key(map(1, 'a', 2, 'b'), 3);
+-----------------------------------+
|map_contains_key(map(1, a, 2, b), 3)|
+-----------------------------------+
|                              false|
+-----------------------------------+
-- map_entries
SELECT map_entries(map(1, 'a', 2, 'b'));
+----------------------------+
|map_entries(map(1, a, 2, b))|
+----------------------------+
|            [{1, a}, {2, b}]|
+----------------------------+
-- map_from_arrays
SELECT map_from_arrays(array(1.0, 3.0), array('2', '4'));
+-------------------------------------------+
|map_from_arrays(array(1.0, 3.0), array(2, 4))|
+-------------------------------------------+
|                         {1.0 -> 2, 3.0 -> 4}|
+-------------------------------------------+
-- map_from_entries
SELECT map_from_entries(array(struct(1, 'a'), struct(2, 'b')));
+------------------------------------------------+
```

```
|map_from_entries(array(struct(1, a), struct(2, b)))|
+---------------------------------------------------+
|                                {1 -> a, 2 -> b}|
+---------------------------------------------------+
-- map_keys
SELECT map_keys(map(1, 'a', 2, 'b'));
+-------------------------+
|map_keys(map(1, a, 2, b))|
+-------------------------+
|                   [1, 2]|
+-------------------------+
-- map_values
SELECT map_values(map(1, 'a', 2, 'b'));
+---------------------------+
|map_values(map(1, a, 2, b))|
+---------------------------+
|                     [a, b]|
+---------------------------+
-- str_to_map
SELECT str_to_map('a:1,b:2,c:3', ',', ':');
+-----------------------------+
|str_to_map(a:1,b:2,c:3, ,, :)|
+-----------------------------+
|         {a -> 1, b -> 2, ...|
+-----------------------------+
SELECT str_to_map('a');
+-------------------+
|str_to_map(a, ,, :)|
+-------------------+
|        {a -> NULL}|
+-------------------+
-- try_element_at
SELECT try_element_at(array(1, 2, 3), 2);
+---------------------------------+
|try_element_at(array(1, 2, 3), 2)|
+---------------------------------+
|                                2|
+---------------------------------+
SELECT try_element_at(map(1, 'a', 2, 'b'), 2);
+---------------------------------+
|try_element_at(map(1, a, 2, b), 2)|
+---------------------------------+
|                                 b|
```

```
+----------------------------------+
```

数学関数

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
expr1 % expr2	`expr1`/`expr2` の余りを返します。
expr1 * expr2	`expr1`*`expr2` を返します。
expr1 + expr2	`expr1`+`expr2` を返します。
expr1 - expr2	`expr1`-`expr2` を返します。
expr1 / expr2	`expr1`/`expr2` を返します。常に浮動小数点除算を実行します。
abs(expr)	数値または時間間隔の値の絶対値を返します。
acos(expr)	`java.lang.Math.acos` で計算された場合と同様に、`expr` の逆コサイン (a.k.a. arc cosine) を返します。
acosh(expr)	`expr` の逆双曲線コサインを返します。
asin(expr)	`java.lang.Math.asin` で計算された場合と同様に、`expr` の逆サイン (a.k.a. アークサイン) を返します。
asinh(expr)	`expr` の逆双曲線サインを返します。

関数	説明
atan(expr)	`java.lang.Math.atan` で計算された場合と同様に、`expr` の逆タンジェント (a.k.a. アークタンジェント) を返します。
atan2(exprY, exprX)	`java.lang.Math.atan2` で計算された場合と同様に、平面の正の x 軸と座標で指定されたポイント (`exprX`、`exprY`) の間の角度をラジアンで返します。
atanh(expr)	`expr` の逆双曲線タンジェントを返します。
bin(expr)	バイナリで表される長い値 `expr` の文字列表現を返します。
bround(expr, d)	HALF_EVEN 四捨五入モードを使用して、`expr` を `d` 小数位に四捨五入して返します。
cbrt(expr)	`expr` の立方根を返します。
ceil(expr[, scale])	`expr` 以上の切り上げ後の最小数を返します。オプションの `scale` パラメータを指定して、四捨五入挙動を制御できます。
ceiling(expr[, scale])	`expr` 以上の切り上げ後の最小数を返します。オプションの `scale` パラメータを指定して、四捨五入挙動を制御できます。
conv(num, from_base, to_base)	`num` を `from_base` から `to_base` に変換します。
cos(expr)	`java.lang.Math.cos` で計算された場合と同様に、`expr` のコサインを返します。
cosh(expr)	`java.lang.Math.cosh` で計算された場合と同様に、`expr` の双曲線コサインを返します。

関数	説明
cot(expr)	`1/java.lang.Math.tan` で計算された場合と同様に、`expr` のコタンジェントを返します。
csc(expr)	`1/java.lang.Math.sin` で計算された場合と同様に、`expr` のコセカントを返します。
degrees(expr)	ラジアンを度に変換します。
expr1 div expr2	`expr1` を `expr2` で割ります。オペランドが NULL の場合、または `expr2` が 0 の場合、NULL を返します。結果は long 型にキャストされます。
e()	オイラー数 e を返します。
exp(expr)	e を `expr` 乗した値を返します。
expm1(expr) – exp(`expr`) を返します	1
factorial(expr)	`expr` の因数を返します。`expr` は [0..20] です。それ以外の場合は null です。
floor(expr[, scale])	`expr` 以下の切り捨て後の最大数を返します。オプションの `scale` パラメータを指定して、四捨五入挙動を制御できます。
greatest(expr, ...)	null 値をスキップして、すべてのパラメータの最大値を返します。
hex(expr)	`expr` を 16 進数に変換します。
hypot(expr1, expr2)	sqrt(`expr1`**2 + `expr2`**2) を返します。
least(expr, ...)	null 値をスキップして、すべてのパラメータの最小値を返します。
ln(expr)	`expr` の自然対数 (底 e) を返します。

関数	説明
log(base, expr)	`expr` の対数を、`base` で計算して返します。
log10(expr)	`expr` の対数を、底 10 で計算して返します。
log1p(expr)	log(1 + `expr`) を返します。
log2(expr)	`expr` の対数を、底 2 で計算して返します。
expr1 mod expr2	`expr1`/`expr2` の余りを返します。
negative(expr)	`expr` の符号を反転させた値を返します。
pi()	pi を返します。
pmod(expr1, expr2)	`expr1` を `expr2` で割った余りの正の値を返します。
positive(expr)	`expr` の値を返します。
pow(expr1, expr2)	`expr1` を `expr2` 乗にします。
power(expr1, expr2)	`expr1` を `expr2` 乗にします。
radians(expr)	度をラジアンに変換します。
rand([seed])	[0, 1) の範囲で一様分布に従う、独立かつ同一分布 (i.i.d.)の乱数を返します。
randn([seed])	標準正規分布 (平均 0、標準偏差 1) に従う、独立かつ同一分布 (i.i.d.) の乱数を返す。
random([seed])	[0, 1) の範囲で一様分布に従う、独立かつ同一分布 (i.i.d.)の乱数を返します。
rint(expr)	引数に最も近い値で、数学的に整数となる倍精度浮動小数点値を返します。

関数	説明
round(expr, d)	HALF_UP 四捨五入モードを使用して、`expr` を `d` 小数位に四捨五入して返します。
sec(expr)	`1/java.lang.Math.cos` で計算された場合と同様に、`expr` の正割を返します。
shiftleft(base, expr)	ビット単位の左シフト。
sign(expr)	`expr` が負、0、または正の場合、-1.0、0.0、または 1.0 を返します。
signum(expr)	`expr` が負、0、または正の場合、-1.0、0.0、または 1.0 を返します。
sin(expr)	`java.lang.Math.sin` で計算された場合と同様に、`expr` のサインを返します。
sinh(expr)	`java.lang.Math.sinh` で計算された場合と同様に、`expr` の双曲線サインを返します。
sqrt(expr)	`expr` の平方根を返します。
tan(expr)	`java.lang.Math.tan` で計算された場合と同様に、`expr` のタンジェントを返します。
tanh(expr)	`java.lang.Math.tanh` で計算された場合と同様に、`expr` の双曲線タンジェントを返します。
try_add(expr1, expr2)	`expr1` と `expr2` の合計を返します。オーバーフロー時の結果は null です。許容される入力タイプは、`+` 演算子と同じです。
try_divide(dividend, divisor)	`dividend`/`divisor` を返します。常に浮動小数点除算を実行します。`expr2` が 0 の場合、結果は常に null になります。`dividend` は数値または間隔である必要があります。`divisor` は数値である必要があります。

関数	説明
try_multiply(expr1, expr2)	`expr1`*`expr2` を返し、オーバーフロー時に結果が null になります。許容される入力タイプは、`*` 演算子と同じです。
try_subtract(expr1, expr2)	`expr1`-`expr2` を返し、オーバーフロー時に結果が null になります。許容される入力タイプは、`-` 演算子と同じです。
unhex(expr)	16 進数の `expr` をバイナリに変換します。
width_bucket(value, min_value, max_value, num_bucket)	`min_value` から `max_value` の範囲の `num_bucket` バケットを持つ等角ヒストグラムで `value` が割り当てられるバケット番号を返します。

例

```
-- %
SELECT 2 % 1.8;
+---------+
|(2 % 1.8)|
+---------+
|      0.2|
+---------+
SELECT MOD(2, 1.8);
+-----------+
|mod(2, 1.8)|
+-----------+
|        0.2|
+-----------+
-- *
SELECT 2 * 3;
+-------+
|(2 * 3)|
+-------+
|      6|
+-------+
-- +
```

```
SELECT 1 + 2;
+-------+
|(1 + 2)|
+-------+
|      3|
+-------+
-- -
SELECT 2 - 1;
+-------+
|(2 - 1)|
+-------+
|      1|
+-------+
-- /
SELECT 3 / 2;
+-------+
|(3 / 2)|
+-------+
|    1.5|
+-------+
SELECT 2L / 2L;
+-------+
|(2 / 2)|
+-------+
|    1.0|
+-------+
-- abs
SELECT abs(-1);
+-------+
|abs(-1)|
+-------+
|      1|
+-------+
SELECT abs(INTERVAL -'1-1' YEAR TO MONTH);
+----------------------------------+
|abs(INTERVAL '-1-1' YEAR TO MONTH)|
+----------------------------------+
|              INTERVAL '1-1' YE...|
+----------------------------------+
-- acos
SELECT acos(1);
+-------+
|ACOS(1)|
+-------+
```

```
|    0.0|
+-------+
SELECT acos(2);
+-------+
|ACOS(2)|
+-------+
|    NaN|
+-------+
-- acosh
SELECT acosh(1);
+--------+
|ACOSH(1)|
+--------+
|     0.0|
+--------+
SELECT acosh(0);
+--------+
|ACOSH(0)|
+--------+
|     NaN|
+--------+
-- asin
SELECT asin(0);
+-------+
|ASIN(0)|
+-------+
|    0.0|
+-------+
SELECT asin(2);
+-------+
|ASIN(2)|
+-------+
|    NaN|
+-------+
-- asinh
SELECT asinh(0);
+--------+
|ASINH(0)|
+--------+
|     0.0|
+--------+
-- atan
SELECT atan(0);
+-------+
```

```
|ATAN(0)|
+-------+
|    0.0|
+-------+
-- atan2
SELECT atan2(0, 0);
+-----------+
|ATAN2(0, 0)|
+-----------+
|        0.0|
+-----------+
-- atanh
SELECT atanh(0);
+--------+
|ATANH(0)|
+--------+
|     0.0|
+--------+
SELECT atanh(2);
+--------+
|ATANH(2)|
+--------+
|     NaN|
+--------+
-- bin
SELECT bin(13);
+-------+
|bin(13)|
+-------+
|   1101|
+-------+
SELECT bin(-13);
+--------------------+
|            bin(-13)|
+--------------------+
|1111111111111111...|
+--------------------+
SELECT bin(13.3);
+---------+
|bin(13.3)|
+---------+
|     1101|
+---------+
-- bround
```

```
SELECT bround(2.5, 0);
+--------------+
|bround(2.5, 0)|
+--------------+
|             2|
+--------------+
SELECT bround(25, -1);
+--------------+
|bround(25, -1)|
+--------------+
|            20|
+--------------+
-- cbrt
SELECT cbrt(27.0);
+----------+
|CBRT(27.0)|
+----------+
|       3.0|
+----------+
-- ceil
SELECT ceil(-0.1);
+----------+
|CEIL(-0.1)|
+----------+
|         0|
+----------+
SELECT ceil(5);
+-------+
|CEIL(5)|
+-------+
|      5|
+-------+
SELECT ceil(3.1411, 3);
+---------------+
|ceil(3.1411, 3)|
+---------------+
|          3.142|
+---------------+
SELECT ceil(3.1411, -3);
+----------------+
|ceil(3.1411, -3)|
+----------------+
|            1000|
+----------------+
```

```
-- ceiling
SELECT ceiling(-0.1);
+-------------+
|ceiling(-0.1)|
+-------------+
|            0|
+-------------+
SELECT ceiling(5);
+----------+
|ceiling(5)|
+----------+
|         5|
+----------+
SELECT ceiling(3.1411, 3);
+------------------+
|ceiling(3.1411, 3)|
+------------------+
|             3.142|
+------------------+
SELECT ceiling(3.1411, -3);
+-------------------+
|ceiling(3.1411, -3)|
+-------------------+
|               1000|
+-------------------+
-- conv
SELECT conv('100', 2, 10);
+----------------+
|conv(100, 2, 10)|
+----------------+
|               4|
+----------------+
SELECT conv(-10, 16, -10);
+------------------+
|conv(-10, 16, -10)|
+------------------+
|               -16|
+------------------+
-- cos
SELECT cos(0);
+------+
|COS(0)|
+------+
|   1.0|
```

```
+------+
-- cosh
SELECT cosh(0);
+-------+
|COSH(0)|
+-------+
|    1.0|
+-------+
-- cot
SELECT cot(1);
+------------------+
|            COT(1)|
+------------------+
|0.6420926159343306|
+------------------+
-- csc
SELECT csc(1);
+------------------+
|            CSC(1)|
+------------------+
|1.1883951057781212|
+------------------+
-- degrees
SELECT degrees(3.141592653589793);
+--------------------------+
|DEGREES(3.141592653589793)|
+--------------------------+
|                     180.0|
+--------------------------+
-- div
SELECT 3 div 2;
+---------+
|(3 div 2)|
+---------+
|        1|
+---------+
SELECT INTERVAL '1-1' YEAR TO MONTH div INTERVAL '-1' MONTH;
+-----------------------------------------------------------+
|(INTERVAL '1-1' YEAR TO MONTH div INTERVAL '-1' MONTH)|
+-----------------------------------------------------------+
|                                                        -13|
+-----------------------------------------------------------+
-- e
SELECT e();
```

```
+-----------------+
|              E()|
+-----------------+
|2.718281828459045|
+-----------------+
-- exp
SELECT exp(0);
+------+
|EXP(0)|
+------+
|   1.0|
+------+
-- expm1
SELECT expm1(0);
+--------+
|EXPM1(0)|
+--------+
|     0.0|
+--------+
-- factorial
SELECT factorial(5);
+------------+
|factorial(5)|
+------------+
|         120|
+------------+
-- floor
SELECT floor(-0.1);
+-----------+
|FLOOR(-0.1)|
+-----------+
|         -1|
+-----------+
SELECT floor(5);
+--------+
|FLOOR(5)|
+--------+
|       5|
+--------+
SELECT floor(3.1411, 3);
+----------------+
|floor(3.1411, 3)|
+----------------+
|           3.141|
```

```
+----------------+
SELECT floor(3.1411, -3);
+-----------------+
|floor(3.1411, -3)|
+-----------------+
|                0|
+-----------------+
-- greatest
SELECT greatest(10, 9, 2, 4, 3);
+------------------------+
|greatest(10, 9, 2, 4, 3)|
+------------------------+
|                      10|
+------------------------+
-- hex
SELECT hex(17);
+-------+
|hex(17)|
+-------+
|     11|
+-------+
SELECT hex('SQL');
+------------------+
|    hex(SQL)|
+------------------+
|53514C|
+------------------+
-- hypot
SELECT hypot(3, 4);
+-----------+
|HYPOT(3, 4)|
+-----------+
|        5.0|
+-----------+
-- least
SELECT least(10, 9, 2, 4, 3);
+---------------------+
|least(10, 9, 2, 4, 3)|
+---------------------+
|                    2|
+---------------------+
-- ln
SELECT ln(1);
+-----+
```

```
|ln(1)|
+-----+
|  0.0|
+-----+
-- log
SELECT log(10, 100);
+------------+
|LOG(10, 100)|
+------------+
|         2.0|
+------------+
-- log10
SELECT log10(10);
+---------+
|LOG10(10)|
+---------+
|      1.0|
+---------+
-- log1p
SELECT log1p(0);
+--------+
|LOG1P(0)|
+--------+
|     0.0|
+--------+
-- log2
SELECT log2(2);
+-------+
|LOG2(2)|
+-------+
|    1.0|
+-------+
-- mod
SELECT 2 % 1.8;
+---------+
|(2 % 1.8)|
+---------+
|      0.2|
+---------+
SELECT MOD(2, 1.8);
+-----------+
|mod(2, 1.8)|
+-----------+
|        0.2|
```

```
+-----------+
-- negative
SELECT negative(1);
+-----------+
|negative(1)|
+-----------+
|         -1|
+-----------+
-- pi
SELECT pi();
+-----------------+
|             PI()|
+-----------------+
|3.141592653589793|
+-----------------+
-- pmod
SELECT pmod(10, 3);
+-----------+
|pmod(10, 3)|
+-----------+
|          1|
+-----------+
SELECT pmod(-10, 3);
+------------+
|pmod(-10, 3)|
+------------+
|           2|
+------------+
-- positive
SELECT positive(1);
+-----+
|(+ 1)|
+-----+
|    1|
+-----+
-- pow
SELECT pow(2, 3);
+---------+
|pow(2, 3)|
+---------+
|      8.0|
+---------+
-- power
SELECT power(2, 3);
```

```
+-----------+
|POWER(2, 3)|
+-----------+
|        8.0|
+-----------+
-- radians
SELECT radians(180);
+-----------------+
|     RADIANS(180)|
+-----------------+
|3.141592653589793|
+-----------------+
-- rand
SELECT rand();
+------------------+
|            rand()|
+------------------+
|0.7211420708112387|
+------------------+
SELECT rand(0);
+------------------+
|           rand(0)|
+------------------+
|0.7604953758285915|
+------------------+
SELECT rand(null);
+------------------+
|        rand(NULL)|
+------------------+
|0.7604953758285915|
+------------------+
-- randn
SELECT randn();
+-------------------+
|            randn()|
+-------------------+
|-0.8175603217732732|
+-------------------+
SELECT randn(0);
+------------------+
|          randn(0)|
+------------------+
|1.6034991609278433|
+------------------+
```

```
SELECT randn(null);
+------------------+
|       randn(NULL)|
+------------------+
|1.6034991609278433|
+------------------+
-- random
SELECT random();
+-----------------+
|           rand()|
+-----------------+
|0.394205008255365|
+-----------------+
SELECT random(0);
+------------------+
|           rand(0)|
+------------------+
|0.7604953758285915|
+------------------+
SELECT random(null);
+------------------+
|        rand(NULL)|
+------------------+
|0.7604953758285915|
+------------------+
-- rint
SELECT rint(12.3456);
+-------------+
|rint(12.3456)|
+-------------+
|         12.0|
+-------------+
-- round
SELECT round(2.5, 0);
+-------------+
|round(2.5, 0)|
+-------------+
|            3|
+-------------+
-- sec
SELECT sec(0);
+------+
|SEC(0)|
+------+
```

```
|   1.0|
+------+
-- shiftleft
SELECT shiftleft(2, 1);
+---------------+
|shiftleft(2, 1)|
+---------------+
|              4|
+---------------+
-- sign
SELECT sign(40);
+--------+
|sign(40)|
+--------+
|     1.0|
+--------+
SELECT sign(INTERVAL -'100' YEAR);
+--------------------------+
|sign(INTERVAL '-100' YEAR)|
+--------------------------+
|                      -1.0|
+--------------------------+
-- signum
SELECT signum(40);
+----------+
|SIGNUM(40)|
+----------+
|       1.0|
+----------+
SELECT signum(INTERVAL -'100' YEAR);
+----------------------------+
|SIGNUM(INTERVAL '-100' YEAR)|
+----------------------------+
|                        -1.0|
+----------------------------+
-- sin
SELECT sin(0);
+------+
|SIN(0)|
+------+
|   0.0|
+------+
-- sinh
SELECT sinh(0);
```

```
+-------+
|SINH(0)|
+-------+
|    0.0|
+-------+
-- sqrt
SELECT sqrt(4);
+-------+
|SQRT(4)|
+-------+
|    2.0|
+-------+
-- tan
SELECT tan(0);
+------+
|TAN(0)|
+------+
|   0.0|
+------+
-- tanh
SELECT tanh(0);
+-------+
|TANH(0)|
+-------+
|    0.0|
+-------+
-- try_add
SELECT try_add(1, 2);
+-------------+
|try_add(1, 2)|
+-------------+
|            3|
+-------------+
SELECT try_add(2147483647, 1);
+----------------------+
|try_add(2147483647, 1)|
+----------------------+
|                  NULL|
+----------------------+
SELECT try_add(date'2021-01-01', 1);
+-----------------------------+
|try_add(DATE '2021-01-01', 1)|
+-----------------------------+
|                   2021-01-02|
```

```
+----------------------------+
SELECT try_add(date'2021-01-01', interval 1 year);
+--------------------------------------------+
|try_add(DATE '2021-01-01', INTERVAL '1' YEAR)|
+--------------------------------------------+
|                                  2022-01-01|
+--------------------------------------------+
SELECT try_add(timestamp'2021-01-01 00:00:00', interval 1 day);
+------------------------------------------------------------+
|try_add(TIMESTAMP '2021-01-01 00:00:00', INTERVAL '1' DAY)|
+------------------------------------------------------------+
|                                     2021-01-02 00:00:00|
+------------------------------------------------------------+
SELECT try_add(interval 1 year, interval 2 year);
+-----------------------------------------------+
|try_add(INTERVAL '1' YEAR, INTERVAL '2' YEAR)|
+-----------------------------------------------+
|                           INTERVAL '3' YEAR|
+-----------------------------------------------+
-- try_divide
SELECT try_divide(3, 2);
+----------------+
|try_divide(3, 2)|
+----------------+
|             1.5|
+----------------+
SELECT try_divide(2L, 2L);
+----------------+
|try_divide(2, 2)|
+----------------+
|             1.0|
+----------------+
SELECT try_divide(1, 0);
+----------------+
|try_divide(1, 0)|
+----------------+
|            NULL|
+----------------+
SELECT try_divide(interval 2 month, 2);
+---------------------------------+
|try_divide(INTERVAL '2' MONTH, 2)|
+---------------------------------+
|             INTERVAL '0-1' YE...|
+---------------------------------+
```

```
SELECT try_divide(interval 2 month, 0);
+---------------------------------+
|try_divide(INTERVAL '2' MONTH, 0)|
+---------------------------------+
|                             NULL|
+---------------------------------+
-- try_multiply
SELECT try_multiply(2, 3);
+------------------+
|try_multiply(2, 3)|
+------------------+
|                 6|
+------------------+
SELECT try_multiply(-2147483648, 10);
+-----------------------------+
|try_multiply(-2147483648, 10)|
+-----------------------------+
|                         NULL|
+-----------------------------+
SELECT try_multiply(interval 2 year, 3);
+----------------------------------+
|try_multiply(INTERVAL '2' YEAR, 3)|
+----------------------------------+
|              INTERVAL '6-0' YE...|
+----------------------------------+
-- try_subtract
SELECT try_subtract(2, 1);
+------------------+
|try_subtract(2, 1)|
+------------------+
|                 1|
+------------------+
SELECT try_subtract(-2147483648, 1);
+----------------------------+
|try_subtract(-2147483648, 1)|
+----------------------------+
|                        NULL|
+----------------------------+
SELECT try_subtract(date'2021-01-02', 1);
+----------------------------------+
|try_subtract(DATE '2021-01-02', 1)|
+----------------------------------+
|                        2021-01-01|
+----------------------------------+
```

```
SELECT try_subtract(date'2021-01-01', interval 1 year);
+-------------------------------------------------+
|try_subtract(DATE '2021-01-01', INTERVAL '1' YEAR)|
+-------------------------------------------------+
|                                       2020-01-01|
+-------------------------------------------------+
SELECT try_subtract(timestamp'2021-01-02 00:00:00', interval 1 day);
+-------------------------------------------------------------------+
|try_subtract(TIMESTAMP '2021-01-02 00:00:00', INTERVAL '1' DAY)|
+-------------------------------------------------------------------+
|                                              2021-01-01 00:00:00|
+-------------------------------------------------------------------+
SELECT try_subtract(interval 2 year, interval 1 year);
+-------------------------------------------------+
|try_subtract(INTERVAL '2' YEAR, INTERVAL '1' YEAR)|
+-------------------------------------------------+
|                                 INTERVAL '1' YEAR|
+-------------------------------------------------+
-- unhex
SELECT decode(unhex('53514C'), 'UTF-8');
+---------------------------------------+
|decode(unhex(53514C), UTF-8)|
+---------------------------------------+
|                                 SQL|
+---------------------------------------+
-- width_bucket
SELECT width_bucket(5.3, 0.2, 10.6, 5);
+-------------------------------+
|width_bucket(5.3, 0.2, 10.6, 5)|
+-------------------------------+
|                              3|
+-------------------------------+
SELECT width_bucket(-2.1, 1.3, 3.4, 3);
+-------------------------------+
|width_bucket(-2.1, 1.3, 3.4, 3)|
+-------------------------------+
|                              0|
+-------------------------------+
SELECT width_bucket(8.1, 0.0, 5.7, 4);
+-------------------------------+
|width_bucket(8.1, 0.0, 5.7, 4)|
+-------------------------------+
|                             5|
+-------------------------------+
```

```
SELECT width_bucket(-0.9, 5.2, 0.5, 2);
+------------------------------+
|width_bucket(-0.9, 5.2, 0.5, 2)|
+------------------------------+
|                             3|
+------------------------------+
SELECT width_bucket(INTERVAL '0' YEAR, INTERVAL '0' YEAR, INTERVAL '10' YEAR, 10);
+-----------------------------------------------------------------------------+
|width_bucket(INTERVAL '0' YEAR, INTERVAL '0' YEAR, INTERVAL '10' YEAR, 10)|
+-----------------------------------------------------------------------------+
|                                                                            1|
+-----------------------------------------------------------------------------+
SELECT width_bucket(INTERVAL '1' YEAR, INTERVAL '0' YEAR, INTERVAL '10' YEAR, 10);
+-----------------------------------------------------------------------------+
|width_bucket(INTERVAL '1' YEAR, INTERVAL '0' YEAR, INTERVAL '10' YEAR, 10)|
+-----------------------------------------------------------------------------+
|                                                                            2|
+-----------------------------------------------------------------------------+
SELECT width_bucket(INTERVAL '0' DAY, INTERVAL '0' DAY, INTERVAL '10' DAY, 10);
+-------------------------------------------------------------------------+
|width_bucket(INTERVAL '0' DAY, INTERVAL '0' DAY, INTERVAL '10' DAY, 10)|
+-------------------------------------------------------------------------+
|                                                                        1|
+-------------------------------------------------------------------------+
SELECT width_bucket(INTERVAL '1' DAY, INTERVAL '0' DAY, INTERVAL '10' DAY, 10);
+-------------------------------------------------------------------------+
|width_bucket(INTERVAL '1' DAY, INTERVAL '0' DAY, INTERVAL '10' DAY, 10)|
+-------------------------------------------------------------------------+
|                                                                        2|
+-------------------------------------------------------------------------+
```

ジェネレータ関数

> **Note**
>
> これらの SQL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

関数	説明
explode(expr)	配列 `expr` の要素を複数の行に分割するか、マップ `expr` の要素を複数の行と列に分割します。特に指定がない限り、配列の要素にはデフォルトの列名 `col` を使用し、マップの要素には `key` と `value` を使用します。
explode_outer(expr)	配列 `expr` の要素を複数の行に分割するか、マップ `expr` の要素を複数の行と列に分割します。特に指定がない限り、配列の要素にはデフォルトの列名 `col` を使用し、マップの要素には `key` と `value` を使用します。
inline(expr)	構造体の配列をテーブルに展開します。特に指定がない限り、デフォルトで列名 col1、col2 などを使用します。
inline_outer(expr)	構造体の配列をテーブルに展開します。特に指定がない限り、デフォルトで列名 col1、col2 などを使用します。
posexplode(expr)	配列 `expr` の要素を位置情報付きで複数の行に分割するか、マップ「expr」の要素を位置情報付きで複数の行と列に分割します。特に指定がない限り、位置には列名 `pos` を使用し、配列の要素には `col` を使用し、マップの要素には `key` と `value` を使用します。
posexplode_outer(expr)	配列 `expr` の要素を位置情報付きで複数の行に分割するか、マップ「expr」の要素を位置情報付きで複数の行と列に分割します。特に指定がない限り、位置には列名 `pos` を使用し、配列の要素には `col` を使用し、マップの要素には `key` と `value` を使用します。
stack(n, expr1, ..., exprk)	`expr1`、...、`exprk` を `n` 行に区切ります。特に指定がない限り、デフォルトで列名 col0、col1 などを使用します。

例

```
-- explode
SELECT explode(array(10, 20));
+---+
|col|
+---+
```

```
| 10|
| 20|
+---+

SELECT explode(collection => array(10, 20));
+---+
|col|
+---+
| 10|
| 20|
+---+

SELECT * FROM explode(collection => array(10, 20));
+---+
|col|
+---+
| 10|
| 20|
+---+

-- explode_outer
SELECT explode_outer(array(10, 20));
+---+
|col|
+---+
| 10|
| 20|
+---+

SELECT explode_outer(collection => array(10, 20));
+---+
|col|
+---+
| 10|
| 20|
+---+

SELECT * FROM explode_outer(collection => array(10, 20));
+---+
|col|
+---+
| 10|
| 20|
+---+
```

```
-- inline
SELECT inline(array(struct(1, 'a'), struct(2, 'b')));
+----+----+
|col1|col2|
+----+----+
|   1|   a|
|   2|   b|
+----+----+

-- inline_outer
SELECT inline_outer(array(struct(1, 'a'), struct(2, 'b')));
+----+----+
|col1|col2|
+----+----+
|   1|   a|
|   2|   b|
+----+----+

-- posexplode
SELECT posexplode(array(10,20));
+---+---+
|pos|col|
+---+---+
|  0| 10|
|  1| 20|
+---+---+

SELECT * FROM posexplode(array(10,20));
+---+---+
|pos|col|
+---+---+
|  0| 10|
|  1| 20|
+---+---+

-- posexplode_outer
SELECT posexplode_outer(array(10,20));
+---+---+
|pos|col|
+---+---+
|  0| 10|
|  1| 20|
+---+---+
```

```
SELECT * FROM posexplode_outer(array(10,20));
+---+---+
|pos|col|
+---+---+
|  0| 10|
|  1| 20|
+---+---+

-- stack
SELECT stack(2, 1, 2, 3);
+----+----+
|col0|col1|
+----+----+
|   1|   2|
|   3|NULL|
+----+----+
```

SELECT 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

OpenSearch SQL は、1 つ以上のテーブルから結果セットを取得するために使用される SELECT ステートメントをサポートしています。次のセクションでは、次のセクションでは、クエリの全体的な構文と、クエリのさまざまな構成要素について説明します。

[Syntax] (構文)

```
select_statement
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select_statement, ... ]
[ ORDER BY
    { expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
    [ , ... ]
    }
]
[ SORT BY
    { expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
    [ , ... ]
```

```
    }
]
[ WINDOW { named_window [ , WINDOW named_window, ... ] } ]
[ LIMIT { ALL | expression } ]
```

select_statement は次のように定義されます。

```
SELECT [ ALL | DISTINCT ] { [ [ named_expression ] [ , ... ] ] }
FROM { from_item [ , ... ] }
[ PIVOT clause ]
[ UNPIVOT clause ]
[ LATERAL VIEW clause ] [ ... ]
[ WHERE boolean_expression ]
[ GROUP BY expression [ , ... ] ]
[ HAVING boolean_expression ]
```

パラメータ

- すべて

  リレーションから一致するすべての行を選択します。これはデフォルトで有効になっています。
- 個別

  結果の重複を削除した後、リレーションから一致するすべての行を選択します。
- named_expression

  名前が割り当てられた式。一般的に、列式を表します。

  構文: `expression [[AS] alias]`
- from_item

  テーブルのリレーション

  結合リレーション

  ピボットリレーション

  ピボット解除リレーション

  テーブル値関数

  インラインテーブル

```
[ LATERAL ] ( Subquery )
```

- PIVOT

  PIVOT 句はデータの観点から使用されます。特定の列値に基づいて集計値を取得できます。

- UNPIVOT

  UNPIVOT 句は列を行に変換します。これは、値の集約を除いて、PIVOT の逆です。

- LATERAL VIEW

  LATERAL VIEW 句は、1 つ以上の行を含む仮想テーブルを生成する EXPLODE などのジェネレータ関数と組み合わせて使用されます。

  LATERAL VIEW は、元の各出力行に行を適用します。

- WHERE

  指定された述語に基づいて FROM 句の結果をフィルタリングします。

- によるグループ化

  行のグループ化に使用される式を指定します。

  これは、集計関数 (MIN、MAX、COUNT、SUM、AVG など) と組み合わせて使用され、各グループのグループ化式と集計値に基づいて行をグループ化します。

  FILTER 句が集計関数にアタッチされると、一致する行のみがその関数に渡されます。

- を持つ

  GROUP BY によって生成された行をフィルタリングする述語を指定します。

  HAVING 句は、グループ化の実行後に行をフィルタリングするために使用されます。

  HAVING が GROUP BY なしで指定されている場合、式をグループ化しないで (グローバル集計) GROUP BY を示します。

- ORDER BY

  クエリの完全な結果セットの行の順序を指定します。

  出力行はパーティション間で順序付けられます。

このパラメータは SORT BY および DISTRIBUTE BY と相互に排他的であり、一緒に指定することはできません。

- SORT BY

  各パーティション内で行が並び替えられる順序を指定します。

  このパラメータは ORDER BY と相互に排他的であり、一緒に指定することはできません。

- LIMIT

  ステートメントまたはサブクエリによって返される行の最大数を指定します。

  この句は、決定的な結果を生成するために主に ORDER BY と組み合わせて使用されます。

- boolean_expression

  結果の型がブール値に評価される式を指定します。

  論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。

- expression

  1 つ以上の値、演算子、SQL 関数を組み合わせて構成され、値として評価される式を指定します。

- named_window

  1 つ以上のソースウィンドウ仕様のエイリアスを指定します。

  ソースウィンドウの仕様は、クエリのウィンドウ定義で参照できます。

WHERE 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

WHERE 句は、指定された条件に基づいてクエリまたはサブクエリの FROM 句の結果を制限するために使用されます。

[Syntax] (構文)

```
WHERE boolean_expression
```

パラメータ

- boolean_expression

  結果の型がブール値に評価される式を指定します。

  論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。

例

```
CREATE TABLE person (id INT, name STRING, age INT);
INSERT INTO person VALUES
(100, 'John', 30),
(200, 'Mary', NULL),
(300, 'Mike', 80),
(400, 'Dan',  50);

-- Comparison operator in `WHERE` clause.
SELECT * FROM person WHERE id > 200 ORDER BY id;
+---+----+---+
| id|name|age|
+---+----+---+
|300|Mike| 80|
|400| Dan| 50|
+---+----+---+

-- Comparison and logical operators in `WHERE` clause.
SELECT * FROM person WHERE id = 200 OR id = 300 ORDER BY id;
+---+----+----+
| id|name| age|
+---+----+----+
|200|Mary|null|
|300|Mike|  80|
+---+----+----+

-- IS NULL expression in `WHERE` clause.
SELECT * FROM person WHERE id > 300 OR age IS NULL ORDER BY id;
+---+----+----+
```

```
| id|name| age|
+---+----+----+
|200|Mary|null|
|400| Dan|  50|
+---+----+----+

-- Function expression in `WHERE` clause.
SELECT * FROM person WHERE length(name) > 3 ORDER BY id;
+---+----+----+
| id|name| age|
+---+----+----+
|100|John|  30|
|200|Mary|null|
|300|Mike|  80|
+---+----+----+

-- `BETWEEN` expression in `WHERE` clause.
SELECT * FROM person WHERE id BETWEEN 200 AND 300 ORDER BY id;
+---+----+----+
| id|name| age|
+---+----+----+
|200|Mary|null|
|300|Mike|  80|
+---+----+----+

-- Scalar Subquery in `WHERE` clause.
SELECT * FROM person WHERE age > (SELECT avg(age) FROM person);
+---+----+---+
| id|name|age|
+---+----+---+
|300|Mike| 80|
+---+----+---+

-- Correlated Subquery in `WHERE` clause.
SELECT id FROM person
WHERE exists (SELECT id FROM person where id = 200);
+---+----+----+
|id |name|age |
+---+----+----+
|200|Mary|null|
+---+----+----+
```

GROUP BY 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

GROUP BY 句は、指定されたグループ化式のセットに基づいて行をグループ化し、1 つ以上の指定された集計関数に基づいて行のグループで集計を計算するために使用されます。

システムは、GROUPING SETS、CUBE、ROLLUP 句を介して同じ入力レコードセットに対して複数の集計も実行します。グループ化式と高度な集計は、GROUP BY 句内で混在させることができ、また GROUPING SETS 句内にネストさせることができます。詳細については、「Mixed/Nested Grouping Analytics 」セクションを参照してください。

FILTER 句が集計関数にアタッチされると、一致する行のみがその関数に渡されます。

[Syntax] (構文)

```
GROUP BY group_expression [ , group_expression [ , ... ] ] [ WITH { ROLLUP | CUBE } ]
GROUP BY { group_expression | { ROLLUP | CUBE | GROUPING SETS } (grouping_set
 [ , ...]) } [ , ... ]
```

集計関数は次のように定義されます。

```
aggregate_name ( [ DISTINCT ] expression [ , ... ] ) [ FILTER ( WHERE
 boolean_expression ) ]
```

パラメータ

- group_expression

  行がグループ化される基準を指定します。行のグループ化は、グループ化式の結果値に基づいて実行されます。

  グループ化式は、GROUP BY a のような列名、GROUP BY 0 のような列位置、または GROUP BY a + b のような式になる場合があります。

- grouping_set

グループ化セットは、括弧内のゼロ個以上のカンマ区切り式で指定されます。グループ化セットに 1 つの要素しかない場合、括弧は省略できます。

例えば、GROUPING SETS ((a), (b)) は GROUPING SETS (a, b) と同じです。

構文: { ([expression [, ...]]) | expression }

- GROUPING SETS

  GROUPING SETS の後に指定された各グループ化セットの行をグループ化します。

  例えば、GROUP BY GROUPING SETS ((warehouse), (product)) は意味的に GROUP BY warehouse と GROUP BY product の結果の和集合と同等です。この句は、UNION ALL の簡略表現であり、UNION ALL 演算子の各分岐が GROUPING SETS 句で指定された各グルーピングセットに対して集計を実行します。

  同様に、GROUP BY GROUPING SETS ((warehouse, product), (product), ()) は意味的に GROUP BY warehouse, product, GROUP BY product とグローバル集計の結果の和集合と同等です。

- ROLLUP

  1 つのステートメントで複数のレベルの集計を指定します。この句は、複数のグループ化セットに基づいて集計を計算するために使用されます。ROLLUP は GROUPING SETS の簡略表現です。

  例えば、GROUP BY warehouse, product WITH ROLLUP or GROUP BY ROLLUP(warehouse, product) と GROUP BY GROUPING SETS((warehouse, product), (warehouse), ()) は同じです。

  GROUP BY ROLLUP(warehouse, product, (warehouse, location)) は GROUP BY GROUPING SETS((warehouse, product, location), (warehouse, product), (warehouse), ()) と同等です。

  ROLLUP 仕様の N 要素は N+1 GROUPING SETS になります。

- CUBE

  CUBE 句は、GROUP BY 句で指定されたグループ化列の組み合わせに基づいて集計を実行するために使用されます。CUBE は GROUPING SETS の簡略表現です。

例えば、GROUP BY warehouse, product WITH CUBE or GROUP BY CUBE(warehouse, product) と GROUP BY GROUPING SETS((warehouse, product), (warehouse), (product), ()) は同じです。

GROUP BY CUBE(warehouse, product, (warehouse, location)) は GROUP BY GROUPING SETS((warehouse, product, location), (warehouse, product), (warehouse, location), (product, warehouse, location), (warehouse), (product), (warehouse, product), ()) と同等です。CUBE 仕様の N 要素は 2^N GROUPING SETS になります。

- 混合/ネストされたグループ化分析

GROUP BY 句には、複数の group_expressions と複数の CUBE|ROLLUP|GROUPING SETS を含めることができます。GROUPING SETS には、GROUPING SETS(ROLLUP(warehouse, location)、CUBE(warehouse, location))、GROUPING SETS(warehouse, GROUPING SETS(location, GROUPING SETS(ROLLUP(warehouse, location),、CUBE(warehouse, location)))) などのネストされた CUBE|ROLLUP|GROUPING SETS 句を含めることもできます。

CUBE|ROLLUP は GROUPING SETS のシンタックスシュガーにすぎません。CUBE|ROLLUP を GROUPING SETS に変換する方法については、上記のセクションを参照してください。group_expression はこのコンテキストで単一グループ GROUPING SETS として扱うことができます。

GROUP BY 句内に複数の GROUPING SETS がある場合、元の GROUPING SETS のクロス積を実行して単一の GROUPING SETS を生成します。GROUPING SETS 句にネストされた GROUPING SETS の場合、グループ化セットを取得してネストを解除します。

例えば、GROUP BY warehouse, GROUPING SETS((product), ()), GROUPING SETS((location, size), (location), (size), ()) and GROUP BY warehouse, ROLLUP(product), CUBE(location, size) と GROUP BY GROUPING SETS((warehouse, product, location, size), (warehouse, product, location), (warehouse, product, size), (warehouse, product), (warehouse, location, size), (warehouse, location), (warehouse, size), (warehouse)) は同じです。

GROUP BY GROUPING SETS(GROUPING SETS(warehouse), GROUPING SETS((warehouse, product))) は GROUP BY GROUPING SETS((warehouse), (warehouse, product)) と同等です。

- aggregate_name

  集計関数名 (MIN、MAX、COUNT、SUM、AVG など) を指定します。

- 個別

  集計関数に渡される前に、入力行の重複を削除します。

- フィルター

  WHERE 句の boolean_expression が true と評価される入力行をフィルタリングして集計関数に渡します。他の行は破棄されます。

例

```
CREATE TABLE dealer (id INT, city STRING, car_model STRING, quantity INT);
INSERT INTO dealer VALUES
(100, 'Fremont', 'Honda Civic', 10),
(100, 'Fremont', 'Honda Accord', 15),
(100, 'Fremont', 'Honda CRV', 7),
(200, 'Dublin', 'Honda Civic', 20),
(200, 'Dublin', 'Honda Accord', 10),
(200, 'Dublin', 'Honda CRV', 3),
(300, 'San Jose', 'Honda Civic', 5),
(300, 'San Jose', 'Honda Accord', 8);

-- Sum of quantity per dealership. Group by `id`.
SELECT id, sum(quantity) FROM dealer GROUP BY id ORDER BY id;
+---+-------------+
| id|sum(quantity)|
+---+-------------+
|100|           32|
|200|           33|
|300|           13|
+---+-------------+

-- Use column position in GROUP by clause.
SELECT id, sum(quantity) FROM dealer GROUP BY 1 ORDER BY 1;
+---+-------------+
```

```
| id|sum(quantity)|
+---+-------------+
|100|           32|
|200|           33|
|300|           13|
+---+-------------+

-- Multiple aggregations.
-- 1. Sum of quantity per dealership.
-- 2. Max quantity per dealership.
SELECT id, sum(quantity) AS sum, max(quantity) AS max FROM dealer GROUP BY id ORDER BY
 id;
+---+---+---+
| id|sum|max|
+---+---+---+
|100| 32| 15|
|200| 33| 20|
|300| 13|  8|
+---+---+---+

-- Count the number of distinct dealer cities per car_model.
SELECT car_model, count(DISTINCT city) AS count FROM dealer GROUP BY car_model;
+------------+-----+
|   car_model|count|
+------------+-----+
| Honda Civic|    3|
|   Honda CRV|    2|
|Honda Accord|    3|
+------------+-----+

-- Sum of only 'Honda Civic' and 'Honda CRV' quantities per dealership.
SELECT id, sum(quantity) FILTER (
WHERE car_model IN ('Honda Civic', 'Honda CRV')
) AS `sum(quantity)` FROM dealer
GROUP BY id ORDER BY id;
+---+-------------+
| id|sum(quantity)|
+---+-------------+
|100|           17|
|200|           23|
|300|            5|
+---+-------------+

-- Aggregations using multiple sets of grouping columns in a single statement.
```

```
-- Following performs aggregations based on four sets of grouping columns.
-- 1. city, car_model
-- 2. city
-- 3. car_model
-- 4. Empty grouping set. Returns quantities for all city and car models.
SELECT city, car_model, sum(quantity) AS sum FROM dealer
GROUP BY GROUPING SETS ((city, car_model), (city), (car_model), ())
ORDER BY city;
+---------+------------+---+
|     city|   car_model|sum|
+---------+------------+---+
|     null|        null| 78|
|     null| HondaAccord| 33|
|     null|    HondaCRV| 10|
|     null|  HondaCivic| 35|
|   Dublin|        null| 33|
|   Dublin| HondaAccord| 10|
|   Dublin|    HondaCRV|  3|
|   Dublin|  HondaCivic| 20|
|  Fremont|        null| 32|
|  Fremont| HondaAccord| 15|
|  Fremont|    HondaCRV|  7|
|  Fremont|  HondaCivic| 10|
| San Jose|        null| 13|
| San Jose| HondaAccord|  8|
| San Jose|  HondaCivic|  5|
+---------+------------+---+

-- Group by processing with `ROLLUP` clause.
-- Equivalent GROUP BY GROUPING SETS ((city, car_model), (city), ())
SELECT city, car_model, sum(quantity) AS sum FROM dealer
GROUP BY city, car_model WITH ROLLUP
ORDER BY city, car_model;
+---------+------------+---+
|     city|   car_model|sum|
+---------+------------+---+
|     null|        null| 78|
|   Dublin|        null| 33|
|   Dublin| HondaAccord| 10|
|   Dublin|    HondaCRV|  3|
|   Dublin|  HondaCivic| 20|
|  Fremont|        null| 32|
|  Fremont| HondaAccord| 15|
|  Fremont|    HondaCRV|  7|
```

```
|  Fremont|  HondaCivic| 10|
| San Jose|        null| 13|
| San Jose| HondaAccord|  8|
| San Jose|  HondaCivic|  5|
+---------+------------+---+

-- Group by processing with `CUBE` clause.
-- Equivalent GROUP BY GROUPING SETS ((city, car_model), (city), (car_model), ())
SELECT city, car_model, sum(quantity) AS sum FROM dealer
GROUP BY city, car_model WITH CUBE
ORDER BY city, car_model;
+---------+------------+---+
|     city|   car_model|sum|
+---------+------------+---+
|     null|        null| 78|
|     null| HondaAccord| 33|
|     null|    HondaCRV| 10|
|     null|  HondaCivic| 35|
|   Dublin|        null| 33|
|   Dublin| HondaAccord| 10|
|   Dublin|    HondaCRV|  3|
|   Dublin|  HondaCivic| 20|
|  Fremont|        null| 32|
|  Fremont| HondaAccord| 15|
|  Fremont|    HondaCRV|  7|
|  Fremont|  HondaCivic| 10|
| San Jose|        null| 13|
| San Jose| HondaAccord|  8|
| San Jose|  HondaCivic|  5|
+---------+------------+---+

--Prepare data for ignore nulls example
CREATE TABLE person (id INT, name STRING, age INT);
INSERT INTO person VALUES
(100, 'Mary', NULL),
(200, 'John', 30),
(300, 'Mike', 80),
(400, 'Dan', 50);

--Select the first row in column age
SELECT FIRST(age) FROM person;
+--------------------+
| first(age, false)  |
+--------------------+
```

```
| NULL                |
+---------------------+

--Get the first row in column `age` ignore nulls,last row in column `id` and sum of
 column `id`.
SELECT FIRST(age IGNORE NULLS), LAST(id), SUM(id) FROM person;
+-------------------+------------------+----------+
| first(age, true)  | last(id, false)  | sum(id)  |
+-------------------+------------------+----------+
| 30                | 400              | 1000     |
+-------------------+------------------+----------+
```

HAVING 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

HAVING 句は、指定された条件に基づいて GROUP BY によって生成された結果をフィルタリングするために使用されます。多くの場合、GROUP BY 句と組み合わせて使用されます。

[Syntax] (構文)

```
HAVING boolean_expression
```

パラメータ

- boolean_expression

  結果の型がブール値に評価される式を指定します。論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。

  注 HAVING 句で指定された式は、以下のみを参照できます。

  1. 定数
  2. GROUP BY に表示される式
  3. 集計関数

例

```
CREATE TABLE dealer (id INT, city STRING, car_model STRING, quantity INT);
INSERT INTO dealer VALUES
(100, 'Fremont', 'Honda Civic', 10),
(100, 'Fremont', 'Honda Accord', 15),
(100, 'Fremont', 'Honda CRV', 7),
(200, 'Dublin', 'Honda Civic', 20),
(200, 'Dublin', 'Honda Accord', 10),
(200, 'Dublin', 'Honda CRV', 3),
(300, 'San Jose', 'Honda Civic', 5),
(300, 'San Jose', 'Honda Accord', 8);

-- `HAVING` clause referring to column in `GROUP BY`.
SELECT city, sum(quantity) AS sum FROM dealer GROUP BY city HAVING city = 'Fremont';
+-------+---+
|   city|sum|
+-------+---+
|Fremont| 32|
+-------+---+

-- `HAVING` clause referring to aggregate function.
SELECT city, sum(quantity) AS sum FROM dealer GROUP BY city HAVING sum(quantity) > 15;
+-------+---+
|   city|sum|
+-------+---+
| Dublin| 33|
|Fremont| 32|
+-------+---+

-- `HAVING` clause referring to aggregate function by its alias.
SELECT city, sum(quantity) AS sum FROM dealer GROUP BY city HAVING sum > 15;
+-------+---+
|   city|sum|
+-------+---+
| Dublin| 33|
|Fremont| 32|
+-------+---+

-- `HAVING` clause referring to a different aggregate function than what is present in
-- `SELECT` list.
SELECT city, sum(quantity) AS sum FROM dealer GROUP BY city HAVING max(quantity) > 15;
+------+---+
|  city|sum|
+------+---+
```

```
|Dublin| 33|
+------+---+

-- `HAVING` clause referring to constant expression.
SELECT city, sum(quantity) AS sum FROM dealer GROUP BY city HAVING 1 > 0 ORDER BY city;
+--------+---+
|    city|sum|
+--------+---+
|  Dublin| 33|
| Fremont| 32|
|San Jose| 13|
+--------+---+

-- `HAVING` clause without a `GROUP BY` clause.
SELECT sum(quantity) AS sum FROM dealer HAVING sum(quantity) > 10;
+---+
|sum|
+---+
| 78|
+---+
```

ORDER BY 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

ORDER BY 句は、ユーザーが指定した順序で結果行を並べ替えて返すために使用されます。SORT BY 句とは異なり、この句は出力の合計順序を保証します。

[Syntax] (構文)

```
ORDER BY { expression [ sort_direction | nulls_sort_order ] [ , ... ] }
```

パラメータ

- ORDER BY

  行の並び替えに使用される、カンマ区切りの式の一覧と、オプションのパラメータ sort_direction および nulls_sort_order を指定します。

- sort_direction

  オプションで、行を昇順にするか降順にするかを指定します。

  ソート方向の有効な値は、昇順の場合は ASC、降順の場合は DESC になります。

  ソート方向が明示的に指定されていない場合、デフォルトでは行が昇順にソートされます。

  構文: `[ ASC | DESC ]`

- nulls_sort_order

  必要に応じて、NULL 値が NULL 以外の値の前後に返されるかどうかを指定します。

  null_sort_order が指定されていない場合は、ソート順が ASC の場合は最初に NULLs ソートし、ソート順が DESC の場合は最後に NULLS ソートします。

  1. NULLS FIRST を指定すると、ソート順序に関係なく NULL 値が最初に返されます。

  2. NULLS LAST を指定すると、ソート順序に関係なく NULL 値が最後に返されます。

  構文: `[ NULLS { FIRST | LAST } ]`

例

```
CREATE TABLE person (id INT, name STRING, age INT);
INSERT INTO person VALUES
(100, 'John', 30),
(200, 'Mary', NULL),
(300, 'Mike', 80),
(400, 'Jerry', NULL),
(500, 'Dan',  50);

-- Sort rows by age. By default rows are sorted in ascending manner with NULL FIRST.
SELECT name, age FROM person ORDER BY age;
+-----+----+
| name| age|
+-----+----+
|Jerry|null|
| Mary|null|
| John|  30|
|  Dan|  50|
| Mike|  80|
```

```
+-----+----+

-- Sort rows in ascending manner keeping null values to be last.
SELECT name, age FROM person ORDER BY age NULLS LAST;
+-----+----+
| name| age|
+-----+----+
| John|  30|
|  Dan|  50|
| Mike|  80|
| Mary|null|
|Jerry|null|
+-----+----+

-- Sort rows by age in descending manner, which defaults to NULL LAST.
SELECT name, age FROM person ORDER BY age DESC;
+-----+----+
| name| age|
+-----+----+
| Mike|  80|
|  Dan|  50|
| John|  30|
|Jerry|null|
| Mary|null|
+-----+----+

-- Sort rows in ascending manner keeping null values to be first.
SELECT name, age FROM person ORDER BY age DESC NULLS FIRST;
+-----+----+
| name| age|
+-----+----+
|Jerry|null|
| Mary|null|
| Mike|  80|
|  Dan|  50|
| John|  30|
+-----+----+

-- Sort rows based on more than one column with each column having different
-- sort direction.
SELECT * FROM person ORDER BY name ASC, age DESC;
+---+-----+----+
| id| name| age|
+---+-----+----+
```

```
|500|  Dan|  50|
|400|Jerry|null|
|100| John|  30|
|200| Mary|null|
|300| Mike|  80|
+---+-----+----+
```

JOIN 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

SQL 結合は、結合条件に基づいて 2 つのリレーションの行を組み合わせるために使用されます。次のセクションでは、全体的な結合構文とさまざまなタイプの結合を例とともに説明します。

[Syntax] (構文)

```
relation INNER JOIN relation [ join_criteria ]
```

パラメータ

- リレーション

  結合するリレーションを指定します。
- join_type

  結合タイプを指定します。

  構文: `INNER | CROSS | LEFT OUTER`
- join_criteria

  あるリレーションの行を別のリレーションの行と組み合わせる方法を指定します。

  構文: `ON boolean_expression | USING ( column_name [ , ... ] )`
- boolean_expression

  戻り値の型がブール型の式を指定します。

結合タイプ

- Inner Join

  内部結合は明示的に指定する必要があります。両方のリレーションで一致する値を持つ行を選択します。

  構文: `relation INNER JOIN relation [ join_criteria ]`

- Left Join

  左結合は、左リレーションのすべての値と右リレーションの一致値を返します。一致しない場合は NULL を追加します。左外部結合とも呼ばれます。

  構文: `relation LEFT OUTER JOIN relation [ join_criteria ]`

- Cross Join

  クロス結合は、2 つのリレーションの直積集合を返します。

  構文: `relation CROSS JOIN relation [ join_criteria ]`

例

```
-- Use employee and department tables to demonstrate different type of joins.
SELECT * FROM employee;
+---+-----+------+
| id| name|deptno|
+---+-----+------+
|105|Chloe|     5|
|103| Paul|     3|
|101| John|     1|
|102| Lisa|     2|
|104| Evan|     4|
|106|  Amy|     6|
+---+-----+------+
SELECT * FROM department;
+------+-----------+
|deptno|   deptname|
+------+-----------+
|     3|Engineering|
|     2|      Sales|
|     1|  Marketing|
```

```
+------+-----------+

-- Use employee and department tables to demonstrate inner join.
SELECT id, name, employee.deptno, deptname
FROM employee INNER JOIN department ON employee.deptno = department.deptno;
+---+-----+------+-----------|
| id| name|deptno|   deptname|
+---+-----+------+-----------|
|103| Paul|     3|Engineering|
|101| John|     1|  Marketing|
|102| Lisa|     2|      Sales|
+---+-----+------+-----------|

-- Use employee and department tables to demonstrate left join.
SELECT id, name, employee.deptno, deptname
FROM employee LEFT JOIN department ON employee.deptno = department.deptno;
+---+-----+------+-----------|
| id| name|deptno|   deptname|
+---+-----+------+-----------|
|105|Chloe|     5|       NULL|
|103| Paul|     3|Engineering|
|101| John|     1|  Marketing|
|102| Lisa|     2|      Sales|
|104| Evan|     4|       NULL|
|106|  Amy|     6|       NULL|
+---+-----+------+-----------|

-- Use employee and department tables to demonstrate cross join.
SELECT id, name, employee.deptno, deptname FROM employee CROSS JOIN department;
+---+-----+------+-----------|
| id| name|deptno|   deptname|
+---+-----+------+-----------|
|105|Chloe|     5|Engineering|
|105|Chloe|     5|  Marketing|
|105|Chloe|     5|      Sales|
|103| Paul|     3|Engineering|
|103| Paul|     3|  Marketing|
|103| Paul|     3|      Sales|
|101| John|     1|Engineering|
|101| John|     1|  Marketing|
|101| John|     1|      Sales|
|102| Lisa|     2|Engineering|
|102| Lisa|     2|  Marketing|
|102| Lisa|     2|      Sales|
```

```
|104| Evan|     4|Engineering|
|104| Evan|     4|  Marketing|
|104| Evan|     4|      Sales|
|106|  Amy|     4|Engineering|
|106|  Amy|     4|  Marketing|
|106|  Amy|     4|      Sales|
+---+-----+------+-----------|
```

LIMIT 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

LIMIT 句は、SELECT テートメントによって返される行数を制限するために使用されます。一般的に、この句は ORDER BY と組み合わせて使用され、結果が決定的であることを確認します。

[Syntax] (構文)

```
LIMIT { ALL | integer_expression }
```

パラメータ

- すべて

  指定した場合、クエリはすべての行を返します。つまり、このオプションが指定されている場合、制限は適用されません。

- integer_expression

  整数を返す畳み込み式を指定します。

例

```
CREATE TABLE person (name STRING, age INT);
INSERT INTO person VALUES
('Jane Doe', 25),
('Pat C', 18),
('Nikki W', 16),
('John D', 25),
```

```
('Juan L', 18),
('Jorge S', 16);

-- Select the first two rows.
SELECT name, age FROM person ORDER BY name LIMIT 2;
+-------+---+
|   name|age|
+-------+---+
|  Pat C| 18|
|Jorge S| 16|
+------+---+

-- Specifying ALL option on LIMIT returns all the rows.
SELECT name, age FROM person ORDER BY name LIMIT ALL;
+--------+---+
|    name|age|
+--------+---+
|   Pat C| 18|
| Jorge S| 16|
|  Juan L| 18|
|  John D| 25|
| Nikki W| 16|
|Jane Doe| 25|
+--------+---+

-- A function expression as an input to LIMIT.
SELECT name, age FROM person ORDER BY name LIMIT length('OPENSEARCH');
+-------+---+
|   name|age|
+-------+---+
|  Pat C| 18|
|Jorge S| 16|
| Juan L| 18|
| John D| 25|
|Nikki W| 16|
+-------+---+
```

CASE 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

CASE 句は、ルールを使用して、他のプログラミング言語の if/else ステートメントと同様に、指定された条件に基づいて特定の結果を返します。

[Syntax] (構文)

```
CASE [ expression ] { WHEN boolean_expression THEN then_expression } [ ... ]
[ ELSE else_expression ]
END
```

パラメータ

- boolean_expression

  結果の型がブール値に評価される式を指定します。

  論理演算子 (AND、OR) を使用して、2 つ以上の式を組み合わせることができます。
- then_expression

  boolean_expression 条件に基づいて、次に式を指定します。

  then_expression と else_expression はすべて同じ型であるか、共通の型に強制的に変換可能である必要があります。
- else_expression

  デフォルトの式を指定します。

  then_expression と else_expression はすべて同じ型であるか、共通の型に強制的に変換可能である必要があります。

例

```
CREATE TABLE person (id INT, name STRING, age INT);
```

```
INSERT INTO person VALUES
(100, 'John', 30),
(200, 'Mary', NULL),
(300, 'Mike', 80),
(400, 'Dan', 50);
SELECT id, CASE WHEN id > 200 THEN 'bigger' ELSE 'small' END FROM person;
+------+-----------------------------------------------------+
|  id  | CASE WHEN (id > 200) THEN bigger ELSE small END  |
+------+-----------------------------------------------------+
| 100  | small                                               |
| 200  | small                                               |
| 300  | bigger                                              |
| 400  | bigger                                              |
+------+-----------------------------------------------------+
SELECT id, CASE id WHEN 100 then 'bigger' WHEN  id > 300 THEN '300' ELSE 'small' END
 FROM person;
+------
+-------------------------------------------------------------------------------------------------------------
+
|  id  | CASE WHEN (id = 100) THEN bigger WHEN (id = CAST((id > 300) AS INT)) THEN 300
 ELSE small END  |
+------
+-------------------------------------------------------------------------------------------------------------
+
| 100  | bigger
                |
| 200  | small
                |
| 300  | small
                |
| 400  | small
                |
+------
+-------------------------------------------------------------------------------------------------------------
+
```

共通テーブル式

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

共通テーブル式 (CTE) は、ユーザーが SQL ステートメントの範囲内で複数回参照できる一時的な結果セットを定義します。CTE は主に SELECT ステートメントで使用されます。

[Syntax] (構文)

```
WITH common_table_expression [ , ... ]
```

common_table_expression は次のように定義されます。

```
Syntexpression_name [ ( column_name [ , ... ] ) ] [ AS ] ( query )
```

パラメータ

- expression_name

  共通テーブル式の名前を指定します。

- query

  SELECT ステートメント。

例

```
-- CTE with multiple column aliases
WITH t(x, y) AS (SELECT 1, 2)
SELECT * FROM t WHERE x = 1 AND y = 2;
+---+---+
|  x|  y|
+---+---+
|  1|  2|
+---+---+

-- CTE in CTE definition
WITH t AS (
WITH t2 AS (SELECT 1)
SELECT * FROM t2
)
SELECT * FROM t;
+---+
|  1|
+---+
```

```
|  1|
+---+

-- CTE in subquery
SELECT max(c) FROM (
WITH t(c) AS (SELECT 1)
SELECT * FROM t
);
+------+
|max(c)|
+------+
|     1|
+------+

-- CTE in subquery expression
SELECT (
WITH t AS (SELECT 1)
SELECT * FROM t
);
+----------------+
|scalarsubquery()|
+----------------+
|               1|
+----------------+

-- CTE in CREATE VIEW statement
CREATE VIEW v AS
WITH t(a, b, c, d) AS (SELECT 1, 2, 3, 4)
SELECT * FROM t;
SELECT * FROM v;
+---+---+---+---+
|  a|  b|  c|  d|
+---+---+---+---+
|  1|  2|  3|  4|
+---+---+---+---+
```

EXPLAIN

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

EXPLAIN ステートメントは、入力ステートメントの論理/物理プランを提供するために使用されます。デフォルトでは、この句は物理プランに関する情報のみを提供します。

[Syntax] (構文)

```
EXPLAIN [ EXTENDED | CODEGEN | COST | FORMATTED ] statement
```

パラメータ

- EXTENDED

  解析された論理計画、分析された論理計画、最適化された論理計画、物理計画を生成します。

  解析された論理計画は、クエリから抽出された未解決の計画です。

  分析された論理プランは、unresolvedAttribute と unresolvedRelation を完全型オブジェクトに変換します。

  最適化論理プランは、一連の最適化ルールを通じて変換され、物理プランになります。
- CODEGEN

  および物理プランがある場合は、ステートメントのコードを生成します。
- COST

  計画ノード統計が利用可能な場合、論理計画と統計を生成します。
- FORMATTED

  物理プランの概要とノードの詳細の 2 つのセクションを生成します。
- ステートメント

  説明する SQL ステートメントを指定します。

例

```
-- Default Output
EXPLAIN select k, sum(v) from values (1, 2), (1, 3) t(k, v) group by k;
+----------------------------------------------------+
|                                                plan|
+----------------------------------------------------+
| == Physical Plan ==
```

```
*(2) HashAggregate(keys=[k#33], functions=[sum(cast(v#34 as bigint))])
+- Exchange hashpartitioning(k#33, 200), true, [id=#59]
+- *(1) HashAggregate(keys=[k#33], functions=[partial_sum(cast(v#34 as bigint))])
+- *(1) LocalTableScan [k#33, v#34]
|
+----------------------------------------------------

-- Using Extended
EXPLAIN EXTENDED select k, sum(v) from values (1, 2), (1, 3) t(k, v) group by k;
+----------------------------------------------------+
|                                                plan|
+----------------------------------------------------+
| == Parsed Logical Plan ==
'Aggregate ['k], ['k, unresolvedalias('sum('v), None)]
 +- 'SubqueryAlias `t`
+- 'UnresolvedInlineTable [k, v], [List(1, 2), List(1, 3)]

 == Analyzed Logical Plan ==
 k: int, sum(v): bigint
 Aggregate [k#47], [k#47, sum(cast(v#48 as bigint)) AS sum(v)#50L]
 +- SubqueryAlias `t`
    +- LocalRelation [k#47, v#48]

 == Optimized Logical Plan ==
 Aggregate [k#47], [k#47, sum(cast(v#48 as bigint)) AS sum(v)#50L]
 +- LocalRelation [k#47, v#48]

 == Physical Plan ==
 *(2) HashAggregate(keys=[k#47], functions=[sum(cast(v#48 as bigint))], output=[k#47,
 sum(v)#50L])
+- Exchange hashpartitioning(k#47, 200), true, [id=#79]
   +- *(1) HashAggregate(keys=[k#47], functions=[partial_sum(cast(v#48 as bigint))],
 output=[k#47, sum#52L])
    +- *(1) LocalTableScan [k#47, v#48]
|
+----------------------------------------------------+

-- Using Formatted
EXPLAIN FORMATTED select k, sum(v) from values (1, 2), (1, 3) t(k, v) group by k;
+----------------------------------------------------+
|                                                plan|
+----------------------------------------------------+
| == Physical Plan ==
 * HashAggregate (4)
```

```
 +- Exchange (3)
    +- * HashAggregate (2)
       +- * LocalTableScan (1)


 (1) LocalTableScan [codegen id : 1]
 Output: [k#19, v#20]

 (2) HashAggregate [codegen id : 1]
 Input: [k#19, v#20]

 (3) Exchange
 Input: [k#19, sum#24L]

 (4) HashAggregate [codegen id : 2]
 Input: [k#19, sum#24L]
|
+-----------------------------------------------------+
```

LATERAL SUBQUERY 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

LATERAL SUBQUERY は、キーワード LATERAL 先行するサブクエリです。先行する FROM 句の列を参照する方法を提供します。LATERAL キーワードがない場合、サブクエリは外部クエリの列のみを参照できますが、FROM 句では参照できません。LATERAL SUBQUERY は複雑なクエリをより簡単かつ効率的にします。

[Syntax] (構文)

```
[ LATERAL ] primary_relation [ join_relation ]
```

パラメータ

- primary_relation

  プライマリリレーションを指定します。次のいずれかを指定できます。

1. テーブルのリレーション
2. エイリアスクエリ

   構文: `( query ) [ [ AS ] alias ]`
3. エイリアスリレーション

   `Syntax: ( relation ) [ [ AS ] alias ]`

例

```
CREATE TABLE t1 (c1 INT, c2 INT);
INSERT INTO t1 VALUES (0, 1), (1, 2);
CREATE TABLE t2 (c1 INT, c2 INT);
INSERT INTO t2 VALUES (0, 2), (0, 3);
SELECT * FROM t1,
LATERAL (SELECT * FROM t2 WHERE t1.c1 = t2.c1);
+--------+-------+--------+-------+
|  t1.c1 | t1.c2 |  t2.c1 | t2.c2 |
+-------+--------+--------+-------+
|    0   |   1   |    0   |   3   |
|    0   |   1   |    0   |   2   |
+-------+--------+--------+-------+
SELECT a, b, c FROM t1,
LATERAL (SELECT c1 + c2 AS a),
LATERAL (SELECT c1 - c2 AS b),
LATERAL (SELECT a * b AS c);
+--------+-------+--------+
|    a   |   b   |    c   |
+-------+--------+--------+
|    3   |  -1   |   -3   |
|    1   |  -1   |   -1   |
+-------+--------+--------+
```

LATERAL VIEW 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

LATERAL VIEW 句は、1 つ以上の行を含む仮想テーブルを生成する EXPLODE などのジェネレータ関数と組み合わせて使用されます。LATERAL VIEW は、元の各出力行にその行を適用します。

[Syntax] (構文)

```
LATERAL VIEW [ OUTER ] generator_function ( expression [ , ... ] ) [ table_alias ] AS
 column_alias [ , ... ]
```

パラメータ

- 外部

  OUTER が指定されている場合、入力配列/マップが空または null の場合、null を返します。
- generator_function

  ジェネレーター関数 (EXPLODE、INLINE など) を指定します。
- table_alias

  generator_function のエイリアスで、オプションです。
- column_alias

  出力行で使用できる generator_function の列エイリアスを一覧表示します。

  generator_function に複数の出力列がある場合、複数のエイリアスを持つことができます。

例

```
CREATE TABLE person (id INT, name STRING, age INT, class INT, address STRING);
INSERT INTO person VALUES
(100, 'John', 30, 1, 'Street 1'),
(200, 'Mary', NULL, 1, 'Street 2'),
(300, 'Mike', 80, 3, 'Street 3'),
(400, 'Dan', 50, 4, 'Street 4');
SELECT * FROM person
LATERAL VIEW EXPLODE(ARRAY(30, 60)) tableName AS c_age
LATERAL VIEW EXPLODE(ARRAY(40, 80)) AS d_age;
+------+-------+-------+--------+-----------+--------+--------+
|  id  | name  |  age  | class  |  address  | c_age  | d_age  |
+------+-------+-------+--------+-----------+--------+--------+
| 100  | John  | 30    | 1      | Street 1  | 30     | 40     |
| 100  | John  | 30    | 1      | Street 1  | 30     | 80     |
```

```
| 100  | John  | 30    | 1      | Street 1  | 60     | 40     |
| 100  | John  | 30    | 1      | Street 1  | 60     | 80     |
| 200  | Mary  | NULL  | 1      | Street 2  | 30     | 40     |
| 200  | Mary  | NULL  | 1      | Street 2  | 30     | 80     |
| 200  | Mary  | NULL  | 1      | Street 2  | 60     | 40     |
| 200  | Mary  | NULL  | 1      | Street 2  | 60     | 80     |
| 300  | Mike  | 80    | 3      | Street 3  | 30     | 40     |
| 300  | Mike  | 80    | 3      | Street 3  | 30     | 80     |
| 300  | Mike  | 80    | 3      | Street 3  | 60     | 40     |
| 300  | Mike  | 80    | 3      | Street 3  | 60     | 80     |
| 400  | Dan   | 50    | 4      | Street 4  | 30     | 40     |
| 400  | Dan   | 50    | 4      | Street 4  | 30     | 80     |
| 400  | Dan   | 50    | 4      | Street 4  | 60     | 40     |
| 400  | Dan   | 50    | 4      | Street 4  | 60     | 80     |
+------+-------+-------+--------+-----------+--------+--------+
SELECT c_age, COUNT(1) FROM person
LATERAL VIEW EXPLODE(ARRAY(30, 60)) AS c_age
LATERAL VIEW EXPLODE(ARRAY(40, 80)) AS d_age
GROUP BY c_age;
+--------+-----------+
| c_age  | count(1)  |
+--------+-----------+
| 60     | 8         |
| 30     | 8         |
+--------+-----------+
SELECT * FROM person
LATERAL VIEW EXPLODE(ARRAY()) tableName AS c_age;
+-----+-------+------+--------+----------+--------+
| id  | name  | age  | class  | address  | c_age  |
+-----+-------+------+--------+----------+--------+
+-----+-------+------+--------+----------+--------+
SELECT * FROM person
LATERAL VIEW OUTER EXPLODE(ARRAY()) tableName AS c_age;
+------+-------+-------+--------+-----------+--------+
|  id  | name  |  age  | class  |  address  | c_age  |
+------+-------+-------+--------+-----------+--------+
| 100  | John  | 30    | 1      | Street 1  | NULL   |
| 200  | Mary  | NULL  | 1      | Street 2  | NULL   |
| 300  | Mike  | 80    | 3      | Street 3  | NULL   |
| 400  | Dan   | 50    | 4      | Street 4  | NULL   |
+------+-------+-------+--------+-----------+--------+
```

LIKE 述語

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

LIKE 述語は、特定のパターンを検索するために使用されます。この述語は、ANY、SOME、ALL などの定量化子を持つ複数のパターンもサポートしています。

[Syntax] (構文)

```
[ NOT ] { LIKE search_pattern [ ESCAPE esc_char ] | [ RLIKE | REGEXP ] regex_pattern }
[ NOT ] { LIKE quantifiers ( search_pattern [ , ... ]) }
```

パラメータ

- search_pattern

  LIKE 句で検索する文字列パターンを指定します。特別なパターンマッチング文字を含めることができます。

  - % は、0 個以上の文字に一致します。
  - _ は、正確に 1 個の文字に一致します。
- esc_char

  エスケープ文字を指定します。デフォルトのエスケープ文字は \ です。
- regex_pattern

  RLIKE または REGEXP 句で検索する正規表現検索パターンを指定します。
- 数量詞

  ANY、SOME、ALL などの数量詞を指定します。

  ANY または SOME は、いずれかのパターンが入力と一致する場合、true を返します。

  ALL は、すべてのパターンが入力と一致する場合、true を返します。

例

```
CREATE TABLE person (id INT, name STRING, age INT);
INSERT INTO person VALUES
(100, 'John', 30),
(200, 'Mary', NULL),
(300, 'Mike', 80),
(400, 'Dan',  50),
(500, 'Evan_w', 16);
SELECT * FROM person WHERE name LIKE 'M%';
+---+----+----+
| id|name| age|
+---+----+----+
|300|Mike|  80|
|200|Mary|null|
+---+----+----+
SELECT * FROM person WHERE name LIKE 'M_ry';
+---+----+----+
| id|name| age|
+---+----+----+
|200|Mary|null|
+---+----+----+
SELECT * FROM person WHERE name NOT LIKE 'M_ry';
+---+------+---+
| id|  name|age|
+---+------+---+
|500|Evan_W| 16|
|300|  Mike| 80|
|100|  John| 30|
|400|   Dan| 50|
+---+------+---+
SELECT * FROM person WHERE name RLIKE 'M+';
+---+----+----+
| id|name| age|
+---+----+----+
|300|Mike|  80|
|200|Mary|null|
+---+----+----+
SELECT * FROM person WHERE name REGEXP 'M+';
+---+----+----+
| id|name| age|
+---+----+----+
|300|Mike|  80|
|200|Mary|null|
+---+----+----+
```

```
SELECT * FROM person WHERE name LIKE '%\_%';
+---+------+---+
| id|  name|age|
+---+------+---+
|500|Evan_W| 16|
+---+------+---+
SELECT * FROM person WHERE name LIKE '%$_%' ESCAPE '$';
+---+------+---+
| id|  name|age|
+---+------+---+
|500|Evan_W| 16|
+---+------+---+
SELECT * FROM person WHERE name LIKE ALL ('%an%', '%an');
+---+----+----+
| id|name| age|
+---+----+----+
|400| Dan|  50|
+---+----+----+
SELECT * FROM person WHERE name LIKE ANY ('%an%', '%an');
+---+------+---+
| id|  name|age|
+---+------+---+
|400|   Dan| 50|
|500|Evan_W| 16|
+---+------+---+
SELECT * FROM person WHERE name LIKE SOME ('%an%', '%an');
+---+------+---+
| id|  name|age|
+---+------+---+
|400|   Dan| 50|
|500|Evan_W| 16|
+---+------+---+
SELECT * FROM person WHERE name NOT LIKE ALL ('%an%', '%an');
+---+----+----+
| id|name| age|
+---+----+----+
|100|John|  30|
|200|Mary|null|
|300|Mike|  80|
+---+----+----+
SELECT * FROM person WHERE name NOT LIKE ANY ('%an%', '%an');
+---+------+----+
| id|  name| age|
+---+------+----+
```

```
|100|  John|  30|
|200|  Mary|null|
|300|  Mike|  80|
|500|Evan_W|  16|
+---+------+----+
SELECT * FROM person WHERE name NOT LIKE SOME ('%an%', '%an');
+---+------+----+
| id|  name| age|
+---+------+----+
|100|  John|  30|
|200|  Mary|null|
|300|  Mike|  80|
|500|Evan_W|  16|
+---+------+----+
```

OFFSET

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

OFFSET 句は、SELECT ステートメントによって返される行を返す前に、スキップする行数を指定するために使用されます。一般的に、この句は ORDER BY と組み合わせて使用され、結果が決定的であることを確認します。

[Syntax] (構文)

```
OFFSET integer_expression
```

パラメータ

- integer_expression

  整数を返す畳み込み式を指定します。

例

```
CREATE TABLE person (name STRING, age INT);
INSERT INTO person VALUES
```

```
('Jane Doe', 25),
('Pat C', 18),
('Nikki W', 16),
('Juan L', 25),
('John D', 18),
('Jorge S', 16);

-- Skip the first two rows.
SELECT name, age FROM person ORDER BY name OFFSET 2;
+-------+---+
|   name|age|
+-------+---+
| John D| 18|
| Juan L| 25|
|Nikki W| 16|
|Jane Doe| 25|
+-------+---+

-- Skip the first two rows and returns the next three rows.
SELECT name, age FROM person ORDER BY name LIMIT 3 OFFSET 2;
+-------+---+
|   name|age|
+-------+---+
| John D| 18|
| Juan L| 25|
|Nikki W| 16|
+-------+---+

-- A function expression as an input to OFFSET.
SELECT name, age FROM person ORDER BY name OFFSET length('WAGON');
+-------+---+
|   name|age|
+-------+---+
|Jane Doe| 25|
+-------+---+
```

PIVOT 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

PIVOT 句はデータの観点から使用されます。特定の列値に基づいて集計値を取得できます。これは、SELECT 句で使用される複数の列に変換されます。PIVOT 句は、テーブル名またはサブクエリの後に指定できます。

[Syntax] (構文)

```
PIVOT ( { aggregate_expression [ AS aggregate_expression_alias ] } [ , ... ] FOR
 column_list IN ( expression_list ) )
```

パラメータ

- aggregate_expression

  集計式 (SUM(a)、COUNT(DISTINCT b) などを指定します)。
- aggregate_expression_alias

  集計式のエイリアスを指定します。
- column_list

  FROM 句に列が含まれ、新しい列に置き換える列を指定します。括弧を使用して、(c1, c2) のように、列を囲みます。
- expression_list

  集計条件として column_list 内の値と一致するために使用される新しい列を指定します。それらに対してエイリアスを追加することもできます。

例

```
CREATE TABLE person (id INT, name STRING, age INT, class INT, address STRING);
INSERT INTO person VALUES
(100, 'John', 30, 1, 'Street 1'),
(200, 'Mary', NULL, 1, 'Street 2'),
(300, 'Mike', 80, 3, 'Street 3'),
(400, 'Dan', 50, 4, 'Street 4');
SELECT * FROM person
PIVOT (
SUM(age) AS a, AVG(class) AS c
FOR name IN ('John' AS john, 'Mike' AS mike)
);
+------+-----------+---------+---------+---------+---------+
```

```
|  id  |  address  | john_a  | john_c  | mike_a  | mike_c  |
+------+-----------+---------+---------+---------+---------+
| 200  | Street 2  | NULL    | NULL    | NULL    | NULL    |
| 100  | Street 1  | 30      | 1.0     | NULL    | NULL    |
| 300  | Street 3  | NULL    | NULL    | 80      | 3.0     |
| 400  | Street 4  | NULL    | NULL    | NULL    | NULL    |
+------+-----------+---------+---------+---------+---------+
SELECT * FROM person
PIVOT (
SUM(age) AS a, AVG(class) AS c
FOR (name, age) IN (('John', 30) AS c1, ('Mike', 40) AS c2)
);
+------+-----------+-------+-------+-------+-------+
|  id  |  address  | c1_a  | c1_c  | c2_a  | c2_c  |
+------+-----------+-------+-------+-------+-------+
| 200  | Street 2  | NULL  | NULL  | NULL  | NULL  |
| 100  | Street 1  | 30    | 1.0   | NULL  | NULL  |
| 300  | Street 3  | NULL  | NULL  | NULL  | NULL  |
| 400  | Street 4  | NULL  | NULL  | NULL  | NULL  |
+------+-----------+-------+-------+-------+-------+
```

セット演算子

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

セット演算子は、2 つの入力リレーションを 1 つのリレーションに結合するために使用されます。OpenSearch SQL は、次の 3 種類のセット演算子をサポートしています。

- EXCEPT、または MINUS
- INTERSECT
- UNION

入力リレーションには、それぞれの列に対して同じ数の列と互換性のあるデータ型が必要です。

を除く

EXCEPT と EXCEPT ALL は、あるリレーションで見つかった行を返しますが、他のリレーションでは返されません。EXCEPT (または EXCEPT DISTINCT) は個別の行のみを取りますが、EXCEPT ALL は結果行から重複を削除しません。MINUS は EXCEPT のエイリアスであることに注意してください。

[Syntax] (構文)

```
 [ ( ] relation [ ) ] EXCEPT | MINUS [ ALL | DISTINCT ] [ ( ] relation [ ) ]
```

例

```
-- Use table1 and table2 tables to demonstrate set operators in this page.
SELECT * FROM table1;
+---+
|  c|
+---+
|  3|
|  1|
|  2|
|  2|
|  3|
|  4|
+---+
SELECT * FROM table2;
+---+
|  c|
+---+
|  5|
|  1|
|  2|
|  2|
+---+
SELECT c FROM table1 EXCEPT SELECT c FROM table2;
+---+
|  c|
+---+
|  3|
|  4|
+---+
SELECT c FROM table1 MINUS SELECT c FROM table2;
+---+
|  c|
```

```
+---+
|  3|
|  4|
+---+
SELECT c FROM table1 EXCEPT ALL (SELECT c FROM table2);
+---+
|  c|
+---+
|  3|
|  3|
|  4|
+---+
SELECT c FROM table1 MINUS ALL (SELECT c FROM table2);
+---+
|  c|
+---+
|  3|
|  3|
|  4|
+---+
```

交差する

INTERSECT と INTERSECT ALL は、両方のリレーションで見つかった行を返します。INTERSECT (または INTERSECT DISTINCT) は個別の行のみを取りますが、INTERSECT ALL は結果行から重複を削除しません。

[Syntax] (構文)

```
 [ ( ] relation [ ) ] INTERSECT [ ALL | DISTINCT ] [ ( ] relation [ ) ]
```

例

```
(SELECT c FROM table1) INTERSECT (SELECT c FROM table2);
+---+
|  c|
+---+
|  1|
|  2|
+---+
(SELECT c FROM table1) INTERSECT DISTINCT (SELECT c FROM table2);
+---+
```

```
|  c|
+---+
|  1|
|  2|
+---+
(SELECT c FROM table1) INTERSECT ALL (SELECT c FROM table2);
+---+
|  c|
+---+
|  1|
|  2|
|  2|
+---+
```

ユニオン

UNION と UNION ALL は、いずれかのリレーションで見つかった行を返します。UNION (または UNION DISTINCT) は個別の行のみを取りますが、UNION ALL は結果行から重複を削除しません。

[Syntax] (構文)

```
 [ ( ] relation [ ) ] UNION [ ALL | DISTINCT ] [ ( ] relation [ ) ]
```

例

```
(SELECT c FROM table1) UNION (SELECT c FROM table2);
+---+
|  c|
+---+
|  1|
|  3|
|  5|
|  4|
|  2|
+---+
(SELECT c FROM table1) UNION DISTINCT (SELECT c FROM table2);
+---+
|  c|
+---+
|  1|
|  3|
|  5|
```

```
|  4|
|  2|
+---+
SELECT c FROM table1 UNION ALL (SELECT c FROM table2);
+---+
|  c|
+---+
|  3|
|  1|
|  2|
|  2|
|  3|
|  4|
|  5|
|  1|
|  2|
|  2|
+---+
```

SORT BY 句

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

SORT BY 句を使うと、各パーティション内の行を指定した順序でソートして返すことができます。複数のパーティション SORT BY がある場合、部分的に並び替えられた結果が返されることがあります。これは、出力の合計順序を保証する ORDER BY 句とは異なります。

[Syntax] (構文)

```
SORT BY { expression [ sort_direction | nulls_sort_order ] [ , ... ] }
```

パラメータ

- SORT BY

  式のカンマ区切りリストと、各パーティション内の行のソートに使用されるオプションのパラメータ sort_direction と nulls_sort_order を指定します。

- sort_direction

  オプションで、行を昇順にするか降順にするかを指定します。

  ソート方向の有効な値は、昇順の場合は ASC、降順の場合は DESC になります。

  ソート方向が明示的に指定されていない場合、デフォルトでは行が昇順にソートされます。

  構文: `[ ASC | DESC ]`

- nulls_sort_order

  必要に応じて、NULL 以外の値の前後に NULL 値を返すかどうかを指定します。

  `null_sort_order` が指定されていない場合、ソート順が ASC の場合は NULLs ソートが最初に、ソート順が DESC の場合は NULLS ソートが最後になります。

  1. `NULLS FIRST` を指定すると、ソート順序に関係なく NULL 値が最初に返されます。

  2. `NULLS LAST` を指定すると、ソート順序に関係なく NULL 値が最後に返されます。

  構文: `[ NULLS { FIRST | LAST } ]`

例

```
CREATE TABLE person (zip_code INT, name STRING, age INT);
INSERT INTO person VALUES
(94588, 'Shirley Rodriguez', 50),
(94588, 'Juan Li', 18),
(94588, 'Anil K', 27),
(94588, 'John D', NULL),
(94511, 'David K', 42),
(94511, 'Aryan B.', 18),
(94511, 'Lalit B.', NULL);
-- Sort rows by `name` within each partition in ascending manner
SELECT name, age, zip_code FROM person SORT BY name;
+------------------+----+--------+
|              name| age|zip_code|
+------------------+----+--------+
|            Anil K|  27|   94588|
|           Juan Li|  18|   94588|
|            John D|null|   94588|
| Shirley Rodriguez|  50|   94588|
```

```
|          Aryan B.|  18|   94511|
|           David K|  42|   94511|
|          Lalit B.|null|   94511|
+------------------+----+--------+
-- Sort rows within each partition using column position.
SELECT name, age, zip_code FROM person SORT BY 1;
+----------------+----+----------+
|              name| age|zip_code|
+------------------+----+--------+
|            Anil K|  27|   94588|
|           Juan Li|  18|   94588|
|            John D|null|   94588|
| Shirley Rodriguez|  50|   94588|
|          Aryan B.|  18|   94511|
|           David K|  42|   94511|
|          Lalit B.|null|   94511|
+------------------+----+--------+

-- Sort rows within partition in ascending manner keeping null values to be last.
SELECT age, name, zip_code FROM person SORT BY age NULLS LAST;
+----+------------------+--------+
| age|              name|zip_code|
+----+------------------+--------+
|  18|           Juan Li|   94588|
|  27|            Anil K|   94588|
|  50| Shirley Rodriguez|   94588|
|null|            John D|   94588|
|  18|          Aryan B.|   94511|
|  42|           David K|   94511|
|null|          Lalit B.|   94511|
+--------------+--------+--------+

-- Sort rows by age within each partition in descending manner, which defaults to NULL
 LAST.
SELECT age, name, zip_code FROM person SORT BY age DESC;
+----+------------------+--------+
| age|              name|zip_code|
+----+------------------+--------+
|  50|          Shirley Rodriguez|   94588|
|  27|            Anil K|   94588|
|  18|           Juan Li|   94588|
|null|            John D|   94588|
|  42|           David K|   94511|
|  18|          Aryan B.|   94511|
```

```
|null|          Lalit B.|   94511|
+----+------------------+--------+

-- Sort rows by age within each partition in descending manner keeping null values to
 be first.
SELECT age, name, zip_code FROM person SORT BY age DESC NULLS FIRST;
+----+------------------+--------+
| age|              name|zip_code|
+----+------------------+--------+
|null|            John D|   94588|
|  50| Shirley Rodriguez|   94588|
|  27|             Anil K|   94588|
|  18|           Juan Li|   94588|
|null|          Lalit B.|   94511|
|  42|           David K|   94511|
|  18|          Aryan B.|   94511|
+----+--------------+--------+--------+

-- Sort rows within each partition based on more than one column with each column
 having
-- different sort direction.
SELECT name, age, zip_code FROM person
SORT BY name ASC, age DESC;
+------------------+----+--------+
|              name| age|zip_code|
+------------------+----+--------+
|             Anil K|  27|   94588|
|           Juan Li|  18|   94588|
|            John D|null|   94588|
| Shirley Rodriguez|  50|   94588|
|          Aryan B.|  18|   94511|
|           David K|  42|   94511|
|          Lalit B.|null|   94511|
+------------------+----+--------+
```

UNPIVOT

> **Note**
>
> この SQL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “サポートされている SQL コマンド”。

UNPIVOT 句は、複数の列を SELECT 句で使用される複数の行に変換します。UNPIVOT 句は、テーブル名またはサブクエリの後に指定できます。

[Syntax] (構文)

```
UNPIVOT [ { INCLUDE | EXCLUDE } NULLS ] (
    { single_value_column_unpivot | multi_value_column_unpivot }
) [[AS] alias]

single_value_column_unpivot:
    values_column
    FOR name_column
    IN (unpivot_column [[AS] alias] [, ...])

multi_value_column_unpivot:
    (values_column [, ...])
    FOR name_column
    IN ((unpivot_column [, ...]) [[AS] alias] [, ...])
```

パラメータ

- unpivot_column

  ピボットを解除する列を指定する FROM 句に列が含まれます。

- name_column

  ピボット解除された列の名前を格納する列の名前。

- values_column

  ピボット解除された列の値を格納する列の名前。

例

```
CREATE TABLE sales_quarterly (year INT, q1 INT, q2 INT, q3 INT, q4 INT);
INSERT INTO sales_quarterly VALUES
(2020, null, 1000, 2000, 2500),
(2021, 2250, 3200, 4200, 5900),
(2022, 4200, 3100, null, null);
-- column names are used as unpivot columns
SELECT * FROM sales_quarterly
UNPIVOT (
```

```
sales FOR quarter IN (q1, q2, q3, q4)
);
+------+---------+-------+
| year | quarter | sales |
+------+---------+-------+
| 2020 | q2      | 1000  |
| 2020 | q3      | 2000  |
| 2020 | q4      | 2500  |
| 2021 | q1      | 2250  |
| 2021 | q2      | 3200  |
| 2021 | q3      | 4200  |
| 2021 | q4      | 5900  |
| 2022 | q1      | 4200  |
| 2022 | q2      | 3100  |
+------+---------+-------+
-- NULL values are excluded by default, they can be included
-- unpivot columns can be alias
-- unpivot result can be referenced via its alias
SELECT up.* FROM sales_quarterly
UNPIVOT INCLUDE NULLS (
sales FOR quarter IN (q1 AS Q1, q2 AS Q2, q3 AS Q3, q4 AS Q4)
) AS up;
+------+---------+-------+
| year | quarter | sales |
+------+---------+-------+
| 2020 | Q1      | NULL  |
| 2020 | Q2      | 1000  |
| 2020 | Q3      | 2000  |
| 2020 | Q4      | 2500  |
| 2021 | Q1      | 2250  |
| 2021 | Q2      | 3200  |
| 2021 | Q3      | 4200  |
| 2021 | Q4      | 5900  |
| 2022 | Q1      | 4200  |
| 2022 | Q2      | 3100  |
| 2022 | Q3      | NULL  |
| 2022 | Q4      | NULL  |
+------+---------+-------+
-- multiple value columns can be unpivoted per row
SELECT * FROM sales_quarterly
UNPIVOT EXCLUDE NULLS (
(first_quarter, second_quarter)
FOR half_of_the_year IN (
(q1, q2) AS H1,
```

```
(q3, q4) AS H2
)
);
+------+------------------+---------------+----------------+
|  id  | half_of_the_year | first_quarter | second_quarter |
+------+------------------+---------------+----------------+
| 2020 | H1               | NULL          | 1000           |
| 2020 | H2               | 2000          | 2500           |
| 2021 | H1               | 2250          | 3200           |
| 2021 | H2               | 4200          | 5900           |
| 2022 | H1               | 4200          | 3100           |
+------+------------------+---------------+----------------+
```

サポートされている PPL コマンド

次の表は、OpenSearch Dashboards が CloudWatch Logs、Amazon S3、または Security Lake をクエリする際にサポートしている PPL コマンドと、CloudWatch Logs Insights がサポートしているコマンドを示しています。CloudWatch Logs Insights は、CloudWatch Logs のクエリ時に OpenSearch Dashboards と同じ CloudWatch PPL 構文を使用し、テーブルは両方とも CloudWatch Logs を参照します。

> **Note**
>
> OpenSearch Service 以外のデータを分析する場合、コマンドは OpenSearch インデックス上での動作と異なる可能性があります。

トピック

- コマンド
- 関数
- OpenSearch PPL を使用する CloudWatch Logs Insights ユーザー向けの追加情報

コマンド

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “fields”	投影が必要な一連のフィールドを表示します。	サポート対象	サポート対象	サポート対象	`fields field1, field2`
the section called “各一般項の意味は次のとおりです。”	指定した条件に基づいてデータをフィルタリングします。	サポート対象	サポート対象	サポート対象	`where field1="success" | where field2 != "i-023fe0a90929d8822" | fields field3, col4, col5, col6 | head 1000`
the section called “stats”	集計と計算を実行します。	サポート対象	サポート対象	サポート対象	`stats count(), count(`field1`), min(`field1`), max(`field1`), avg(`field1`) by field2`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
					`| head 1000`
the section called “parse”	文字列から正規表現 (regex) パターンを抽出し、抽出されたパターンを表示します。抽出されたパターンはその後、新しいフィールドの作成やデータのフィルタリングに使用できます。	サポート対象	サポート対象	サポート対象	`parse `field1` ".*/(?<field2>[^/]+$)" | where field2 = "requestId" | fields field2, `field2` | head 1000`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “パターン”	テキストフィールドからログパターンを抽出し、検索結果に結果を追加します。パターン別にログをグループ化すると、大量のログデータから統計情報を集計し、分析やトラブルシューティングを行うことが容易になります。	サポート対象外	サポート対象	サポート対象	`patterns new_field='no_numbers' pattern='[0-9]' message | fields message, no_numbers`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “sort”	表示された結果をフィールド名で並べ替えます。 sort -FieldName を使用して降順で並べ替えます。	サポート対象	サポート対象	サポート対象	`stats count(), count(`field1`), min(`field1`) as field1Alias, max(`field1`), avg(`field1`) by field2 | sort -field1Alias | head 1000`
the section called “評価”	フィールドの値を変更または処理し、別のフィールドに保存します。これは、列を数学的に変更したり、列に文字列関数を適用したり、列に日付関数を適用したりするのに有用です。	サポート対象	サポート対象	サポート対象	`eval field2 = `field1` * 2 | fields field1, field2 | head 20`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “rename”	検索結果の 1 つ以上のフィールドの名前を変更します。	サポート対象	サポート対象	サポート対象	`rename field2 as field1 | fields field1`
the section called “head”	表示されるクエリ結果を frst N 行に制限します。	サポート対象	サポート対象	サポート対象	`fields `@message` | head 20`
the section called “grok”	正規表現に基づいてテキストフィールドを grok パターンで解析し、検索結果に結果を追加します。	サポート対象	サポート対象	サポート対象	`grok email '.+@%{HOSTNAME:host}' | fields email`
the section called “top”	フィールド内で最も頻度の高い値を抽出します。	サポート対象	サポート対象	サポート対象	`top 2 Field1 by Field2`
the section called “dedup”	指定したフィールドに基づいて重複するエントリを削除します。	サポート対象	サポート対象	サポート対象	`dedup field1 | fields field1, field2, field3`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “結合”	2 つのデータセットを結合します。	サポート対象	サポート対象	サポート対象	`source=customer | join ON c_custkey = o_custkey orders | head 10`
the section called “lookup”	ルックアップインデックス (ディメンションテーブル) からデータを追加または置換することで、検索データを充実させます。インデックスのフィールドをディメンションテーブルの値で拡張し、ルックアップ条件が一致したときに値を追加または置換できます。	サポート対象外	サポート対象	サポート対象	`where orderType = 'Cancelled' | lookup account_list, mkt_id AS mkt_code replace amount, account_name as name | stats count(mkt_code), avg(amount) by name`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “subquery”	Piped Processing Language (PPL) ステートメント内で複雑なネストされたクエリを実行します。	サポート対象	サポート対象	サポート対象	`where id in [ subquery source=users | where user in [ subquery source=actions | where action="login" | fields user ] | fields uid ]`
the section called “rare”	フィールドリスト内のすべてのフィールドの最低頻度の値を検索します。	サポート対象	サポート対象	サポート対象	`rare Field1 by Field2`
the section called “trendline”	フィールドの移動平均を計算します。	サポート対象	サポート対象	サポート対象	`trendline sma(2, field1) as field1Alias`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “eventstats”	計算された要約統計量を使用してイベントデータを補強します。イベント内の指定されたフィールドを分析し、さまざまな統計量を計算して、その結果を新しいフィールドとして各イベントに追加します。	サポート対象 (を除く count())	サポート対象	サポート対象	`eventstats sum(field1) by field2`
the section called “flatten”	フィールドをフラット化します。対象のフィールドは、次の型である必要があります: `struct<?,?> or array<struct<?,?>>`	サポート対象	サポート対象	サポート対象	`source=table | flatten field1`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “fieldsummary”	各フィールドの基本統計 (カウント、個別カウント、最小、最大、平均、stddev、平均) を計算します。	サポート対象 (クエリごとに 1 つのフィールド)	サポート対象	サポート対象	`where field1 != 200 | fieldsummary includefields=field1 nulls=true`
the section called “fillnull”	null フィールドに指定した値を入力します。1 つ以上のフィールドで使用できます。	サポート対象	サポート対象	サポート対象	`fields field1 | eval field2=field1 | fillnull value=0 field1`
the section called “拡張”	複数の値を含むフィールドを別々の行に分割し、指定されたフィールドの値ごとに新しい行を作成します。	サポート対象	サポート対象	サポート対象	`expand employee | stats max(salary) as max by state, company`

PPL コマンド	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “説明”	テーブル、スキーマ、カタログの構造とメタデータに関する詳細情報を取得します。	サポート対象外	サポート対象	サポート対象	`describe schema.table`

関数

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “String” (CONCAT, CONCAT_WS , LENGTH, LOWER, LTRIM, POSITION, REVERSE, RIGHT, RTRIM, SUBSTRING , TRIM, UPPER)	PPL クエリ内の文字列およびテキストデータを操作および変換できる PPL の組み込み関数。例えば、大文字と小文字の変換、文字列の組み合わせ、パーツの抽出、テキストの	サポート対象	サポート対象	サポート対象	`eval col1Len = LENGTH(col1) | fields col1Len`

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
	クリーニングなどが挙げられます。				

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “日付および時間” (DAY, DAYOFMONTH , DAY_OF_MONTH ,DAYOFWEEK , DAY_OF_WEEK , DAYOFYEAR , DAY_OF_YEAR , DAYNAME, FROM_UNIXTIME , HOUR, HOUR_OF_DAY , LAST_DAY, LOCALTIME STAMP , LOCALTIME , MAKE_DATE , MINUTE, MINUTE_OF_HOUR , MONTH, MONTHNAME , MONTH_OF_YEAR , NOW, QUARTER, SECOND, SECOND_OF_MINUTE , SUBDATE, SYSDATE, TIMESTAMP , UNIX_TIMESTAMP , WEEK, WEEKDAY, WEEK_OF_YEAR , DATE_ADD,	PPL クエリで日付とタイムスタンプデータを処理および変換するための組み込み関数。例えば、date_add、datediff、curate などが挙げられます。	サポート対象	サポート対象	サポート対象	`eval newDate = ADDDATE(DATE('2020-08-26'), 1) | fields newDate`

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
DATE_SUB, TIMESTAMP ADD , TIMESTAMP DIFF , UTC_TIMESTAMP , CURRENT_TIMEZONE)					
the section called “Condition” (EXISTS, IF, IFNULL, ISNOTNULL , ISNULL, NULLIF)	複数の行で計算を実行して 1 つの要約値を生成する組み込み関数。例えば、sum、c などが挙げられます。	サポート対象	サポート対象	サポート対象	`eval field2 = isnull(col1) | fields field2, col1, field3`

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “数値演算” (ABS, ACOS, ASIN, ATAN, ATAN2, CEIL, CEILING, CONV, COS, COT, CRC32, DEGREES, E, EXP, FLOOR, LN, LOG, LOG2, LOG10, MOD, PI. POW, POWER, RADIANS, RAND, ROUND, SIGN, SIN, SQRT, CBRT)	PPL クエリで数学的計算と変換を実行するための組み込み関数。例: abs (絶対値)、round (四捨五入数値)、sqrt (平方根)、pow (電力計算)、ceil (最も近い整数に切り上げ)。	サポート対象	サポート対象	サポート対象	`eval field2 = ACOS(col1) | fields col1`
the section called “表現” (算術演算子 (+、-、*)、述語演算子 (>. <、IN))	式の組み込み関数、特に値式に対しては、スカラー値を返します。式にはさまざまな型と形式があります。	サポート対象	サポート対象	サポート対象	`where age > (25 + 5) | fields age`

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “IP アドレス” (CIDRMATCH)	CIDR などの IP アドレスを処理するための組み込み関数。	サポート対象	サポート対象	サポート対象	`where cidrmatch(ip, '********/24') | fields ip`
the section called “JSON” (ARRAY_LENGTH , ARRAY_LENGTH , JSON, JSON_ARRAY , JSON_EXTRACT , JSON_KEYS , JSON_OBJECT , JSON_VALID , TO_JSON_STRING)	配列、抽出、検証など、JSON を処理するための組み込み関数。	サポート対象	サポート対象	サポート対象	`eval` `json_extract('{"a":"b"}', '$.a')` = json_extract('{"a":"b"}', '$a')
the section called “Lambda” (EXISTS, FILTER, REDUCE, TRANSFORM)	配列、抽出、検証など、JSON を処理するための組み込み関数。	サポート対象外	サポート対象	サポート対象	`eval array = json_array(1, -1, 2), result = filter(array, x -> x > 0) | fields result`

PPL 関数	説明	CloudWatch Logs	Amazon S3	Security Lake	コマンドの例
the section called “暗号化” (MD5, SHA1, SHA2)	データの一意のフィンガープリントを生成できる組み込み関数。これは、検証、比較、またはより複雑なセキュリティプロトコルの一部として使用できます。	サポート対象	サポート対象	サポート対象	`eval `MD5('hello')` = MD5('hello') | fields `MD5('hello')``

OpenSearch PPL を使用する CloudWatch Logs Insights ユーザー向けの追加情報

CloudWatch Logs Insights はほとんどの OpenSearch PPL コマンドと関数をサポートしていますが、一部のコマンドと関数は現在サポートされていません。例えば、PPL では現在 Lookup コマンドはサポートされていません。2025 年 6 月 2 日現在、CloudWatch Logs Insights は PPL で JOIN、サブクエリ、Flatten、Fillnull、Expand、Cidrmatch、および JSON 関数をサポートするようになりました。サポートされているクエリコマンドと関数の完全なリストについては、上記の表の Amazon CloudWatch Logs 列を参照してください。

サンプルクエリとクォータ

以下は、CloudWatch Logs Insights ユーザーと CloudWatch データをクエリする OpenSearch ユーザーの両方に適用されます。

OpenSearch Service から CloudWatch Logs をクエリするときに適用される制限については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs quotas」を参照してください。制限には、クエリできる CloudWatch Log グループの数、同時実行できる最大クエリ数、最大クエリ実行時間、結果として返される最大行数が含まれます。CloudWatch Logs のクエリに使用する言語 (OpenSearch PPL、SQL、および Logs Insights QL) に関係なく、制限は同じです。

PPL コマンド

トピック

- コメント
- 相関コマンド
- 重複排除コマンド
- コマンドの説明
- 評価コマンド
- eventstats コマンド
- 拡張コマンド
- explain コマンド
- fillnull コマンド
- fields コマンド
- flatten コマンド
- grok コマンド
- head コマンド
- join コマンド
- lookup コマンド
- parse コマンド
- patterns コマンド
- rare コマンド
- rename コマンド
- search コマンド
- sort コマンド
- stats コマンド
- subquery コマンド

- top コマンド
- trendline コマンド
- where コマンド
- フィールドの概要
- 拡張コマンド
- PPL 関数

コメント

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

PPL は、行コメントとブロックコメントの両方をサポートしています。システムはコメントテキストを評価しません。

行コメント

行コメントは 2 つのスラッシュ // で始まり、新しい行で終わります。

例:

```
os> source=accounts | top gender // finds most common gender of all the accounts
fetched rows / total rows = 2/2
+-----------+
| gender    |
|-----------|
| M         |
| F         |
+-----------+
```

ブロックコメント

ブロックコメントはスラッシュの後にアスタリスク * で開始し、アスタリスクの後にスラッシュ */ で終了します。

例:

```
os> source=accounts | dedup 2 gender /* dedup the document with gender field keep 2
 duplication */ | fields account_number, gender
fetched rows / total rows = 3/3
+------------------+----------+
| account_number   | gender   |
|------------------+----------|
| 1                | M        |
| 6                | M        |
| 13               | F        |
+------------------+----------+
```

相関コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

一般的なディメンションと時間枠に従って、さまざまなデータソースを関連付けることができます。

この相関関係は、同じ期間を共有しているが正式に同期されていないさまざまな分野からの大量のデータを処理する場合に極めて重要です。

これらのさまざまなデータソースを時間枠や同様のディメンションに基づいて相関させることで、データを充実させ、貴重なインサイトを発見できます。

例

オブザーバビリティドメインには 3 つの異なるデータソースがあります。

- ログ
- メトリクス
- トレース

これらのデータソースは、共通のディメンションを共有する場合があります。あるデータソースから別のデータソースに移行するには、データソースを正しく関連付ける必要があります。セマンティック命名規則を使用すると、ログ、トレース、メトリクス間で共有要素を識別できます。

例:

```
{
  "@timestamp": "2018-07-02T22:23:00.186Z",
  "aws": {
    "elb": {
      "backend": {
        "http": {
          "response": {
            "status_code": 500
          }
        },
        "ip": "********",
        "port": "80"
      },
      ...
     "target_port": [
        "10.0.0.1:80"
      ],
      "target_status_code": [
        "500"
      ],
      "traceId": "Root=1-58337262-36d228ad5d99923122bbe354",
      "type": "http"
    }
  },
  "cloud": {
    "provider": "aws"
  },
  "http": {
    "request": {
    ...
  },
  "communication": {
    "source": {
      "address": "**************",
      "ip": "**************",
      "port": 2817
    }
  },
  "traceId": "Root=1-58337262-36d228ad5d99923122bbe354"
}
```


この例では、 AWS に存在するサービスから到着する ELB ログを示しています AWS。これは、ステータスコード 500 のバックエンド HTTP 応答を示しており、エラーを意味します。これにより、

アラートが発動されたり、通常のモニタリングプロセスの一部になったりする可能性があります。次のステップでは、このイベントに関する関連データを収集して徹底的な調査を行います。

時間枠に関連するすべてのデータをクエリしたくなるかもしれませんが、このアプローチは圧倒的な負荷を招く可能性があります。情報が多すぎて、根本原因を特定するよりも無関係なデータをフィルタリングする時間が長くなってしまう可能性があります。

代わりに、さまざまなソースからのデータを相関させることで、より的を絞ったアプローチを使用できます。相関関係には、次のディメンションを使用できます。

- IP – `"ip": "10.0.0.1" | "ip": "***************"`
- ポート – `"port": 2817 | "target_port": "10.0.0.1:80"`

追加のトレースおよびメトリクスインデックスにアクセスでき、スキーマ構造に精通している場合は、より精度の高い相関クエリを作成できます。

相関させたい HTTP 情報を含むトレースインデックスドキュメントの例を次に示します。

```
{
  "traceId": "c1d985bd02e1dbb85b444011f19a1ecc",
  "spanId": "55a698828fe06a42",
  "traceState": [],
  "parentSpanId": "",
  "name": "mysql",
  "kind": "CLIENT",
  "@timestamp": "2021-11-13T20:20:39+00:00",
  "events": [
    {
      "@timestamp": "2021-03-25T17:21:03+00:00",
       ...
    }
  ],
  "links": [
    {
      "traceId": "c1d985bd02e1dbb85b444011f19a1ecc",
      "spanId": "55a698828fe06a42w2",
      },
      "droppedAttributesCount": 0
    }
  ],
  "resource": {
    "service@name": "database",
```

```
    "telemetry@sdk@name": "opentelemetry",
    "host@hostname": "ip-172-31-10-8.us-west-2.compute.internal"
  },
  "status": {
    ...
  },
  "attributes": {
    "http": {
      "user_agent": {
        "original": "Mozilla/5.0"
      },
      "network": {
         ...
        }
      },
      "request": {
         ...
        }
      },
      "response": {
        "status_code": "200",
        "body": {
          "size": 500
        }
      },
      "client": {
        "server": {
          "socket": {
            "address": "***********",
            "domain": "example.com",
            "port": 80
          },
          "address": "***********",
          "port": 80
        },
        "resend_count": 0,
        "url": {
          "full": "http://example.com"
        }
      },
      "server": {
        "route": "/index",
        "address": "***********",
        "port": 8080,
```

```
        "socket": {
         ...
        },
        "client": {
         ...
         }
        },
        "url": {
         ...
        }
      }
    }
  }
}
```

このアプローチでは、システムの動作と状態をよりよく理解するために、elb ログと関連付けることができる traceId と http のクライアント/サーバー ip を確認できます。

新しい相関クエリコマンド

このタイプの調査を許可する新しいコマンドは次のとおりです。

```
source alb_logs, traces | where alb_logs.ip="10.0.0.1" AND
 alb_logs.cloud.provider="aws"|
correlate exact fields(traceId, ip) scope(@timestamp, 1D) mapping(alb_logs.ip =
 traces.attributes.http.server.address, alb_logs.traceId = traces.traceId )
```

コマンドの各部分の動作は次のとおりです。

1. source alb_logs, traces – 関連付けるデータソースを選択します。
2. where ip="10.0.0.1" AND cloud.provider="aws" – これにより、検索の範囲が絞り込まれます。
3. correlate exact fields(traceId, ip) – これにより、次のフィールドの完全一致に基づいてデータを相関させるようにシステムに指示します。
   - ip フィールドには明示的なフィルター条件があるため、すべてのデータソースの相関関係で使用されます。
   - traceId フィールドには明示的なフィルターがないため、すべてのデータソースで同じ traceIds に一致します。

フィールド名は、相関コマンド内の関数の論理的な意味を示します。実際の結合条件は、指定したマッピングステートメントによって異なります。

用語 exact は、相関ステートメントがクエリステートメントを満たすためにすべてのフィールドを一致させる必要があることを意味します。

この用語 approximate は、ベストケースシナリオで一致しようとし、部分一致の行を拒否しません。

さまざまなフィールドマッピングへの対応

データソース間で同じ論理フィールド (ip など) の名前が異なる場合は、パスフィールドの明示的なマッピングを指定する必要があります。これに対処するには、相関条件を拡張して、論理的に類似した意味を持つ異なるフィールド名と一致させることができます。その方法は以下の通りです:

```
alb_logs.ip = traces.attributes.http.server.address, alb_logs.traceId = traces.traceId
```

相関結合に参加する各フィールドに対して、この相関コマンドで結合するすべてのテーブルを含む関連するマッピングステートメントを提供する必要があります。

例

この例では、2 つのソースがあります: alb_logs, traces

2 つのフィールドがあります: traceId, ip

マッピングステートメントは 2 つあります: alb_logs.ip = traces.attributes.http.server.address, alb_logs.traceId = traces.traceId

相関期間の範囲

実行エンジン (ドライバー) による作業を簡素化するために、スコープステートメントを追加できます。これにより、この検索の範囲を指定する必要がある時間に結合クエリが明示的に指示されます。

scope(@timestamp, 1D)i

この例では、検索範囲は日単位で焦点を当てているため、同じ日に発生した相関関係はまとめてグループ化されます。このスコープメカニズムにより、結果の制御が簡単になり、ニーズに応じて増分検索解決が可能になります。

ドライバーのサポート

新しい相関コマンドは、実際には「非表示」結合コマンドです。したがって、次の PPL ドライバーのみがこのコマンドをサポートします。これらのドライバーでは、相関コマンドは適切な Catalyst Join 論理プランに直接変換されます。

例

```
source alb_logs, traces, metrics | where ip="10.0.0.1" AND
cloud.provider="aws"| correlate exact on (ip, port) scope(@timestamp,
2018-07-02T22:23:00, 1 D)
```

論理プラン:

```
'Project [*]
+- 'Join Inner, ('ip && 'port)
   :- 'Filter (('ip === "10.0.0.1" & 'cloud.provider === "aws") &
 inTimeScope('@timestamp, "2018-07-02T22:23:00", "1 D"))
      +- 'UnresolvedRelation [alb_logs]
   +- 'Join Inner, ('ip & 'port)
      :- 'Filter (('ip === "10.0.0.1" & 'cloud.provider === "aws") &
 inTimeScope('@timestamp, "2018-07-02T22:23:00", "1 D"))
         +- 'UnresolvedRelation [traces]
      +- 'Filter (('ip === "10.0.0.1" & 'cloud.provider === "aws") &
 inTimeScope('@timestamp, "2018-07-02T22:23:00", "1 D"))
         +- 'UnresolvedRelation [metrics]
```

カタリストエンジンは、最も効率的な結合順序に従ってこのクエリを最適化します。

重複排除コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

dedup コマンドを使用して、指定したフィールドに基づいて検索結果から同じドキュメントを削除します。

構文

次の構文を使用します。

```
dedup [int] <field-list> [keepempty=<bool>] [consecutive=<bool>]
```

int

- オプション。
- この dedup コマンドは、<int> を指定すると、組み合わせごとに複数のイベントを保持します。<int> の数値は 0 より大きくする必要があります。数値を指定しない場合、最初に発生したイベントのみが保持されます。他のすべての重複は結果から削除されます。
- デフォルト: 1

keepempty

- オプション。
- true の場合、フィールドリストのいずれかのフィールドの値が NULL または MISSING であるドキュメントを保持します。
- デフォルト: false

consecutive

- オプション。
- true の場合、値の重複した組み合わせが連続するイベントのみを削除します。
- デフォルト: false

field-list

- 必須。
- フィールドのカンマ区切りリスト。少なくとも 1 つのメタデータフィールドが必要です。

例 1: 1 つのフィールドに基づいて重複排除を行う

この例では、性別フィールドを使用してドキュメントを重複排除する方法を示します。

PPL クエリ:

```
os> source=accounts | dedup gender | fields account_number, gender;
fetched rows / total rows = 2/2
```

```
+------------------+----------+
| account_number   | gender   |
|------------------+----------|
| 1                | M        |
| 13               | F        |
+------------------+----------+
```

例 2: 2 つの重複ドキュメントを保持する

この例では、性別フィールドを使用してドキュメントを重複排除し、2 つの重複を保持する方法を示しています。

PPL クエリ:

```
os> source=accounts | dedup 2 gender | fields account_number, gender;
fetched rows / total rows = 3/3
+------------------+----------+
| account_number   | gender   |
|------------------+----------|
| 1                | M        |
| 6                | M        |
| 13               | F        |
+------------------+----------+
```

例 3: デフォルトで空のフィールドを保持または無視する

この例では、null 値フィールドを保持してドキュメントを重複排除する方法を示しています。

PPL クエリ:

```
os> source=accounts | dedup email keepempty=true | fields account_number, email;
fetched rows / total rows = 4/4
+------------------+-----------------------+
| account_number   | email                 |
+------------------+-----------------------+
| 1                | john_doe@example.com  |
| 6                | jane_doe@example.com  |
| 13               | null                  |
| 18               | juan_li@example.com   |
+------------------+-----------------------+
```

この例では、空の値フィールドを無視してドキュメントを重複排除する方法を示しています。

PPL クエリ:

```
os> source=accounts | dedup email | fields account_number, email;
fetched rows / total rows = 3/3
+------------------+-----------------------+
| account_number   | email                 |
+------------------+-----------------------+
| 1                | john_doe@example.com  |
| 6                | jane_doe@example.com  |
| 18               | juan_li@example.com   |
+------------------+-----------------------+
```

例 4: 連続ドキュメントの重複

この例では、連続するドキュメントで重複排除する方法を示しています。

PPL クエリ:

```
os> source=accounts | dedup gender consecutive=true | fields account_number, gender;
fetched rows / total rows = 3/3
+------------------+----------+
| account_number   | gender   |
+------------------+----------+
| 1                | M        |
| 13               | F        |
| 18               | M        |
+------------------+----------+
```

その他の例

- source = table | dedup a | fields a,b,c
- source = table | dedup a,b | fields a,b,c
- source = table | dedup a keepempty=true | fields a,b,c
- source = table | dedup a,b keepempty=true | fields a,b,c
- source = table | dedup 1 a | fields a,b,c
- source = table | dedup 1 a,b | fields a,b,c
- source = table | dedup 1 a keepempty=true | fields a,b,c
- source = table | dedup 1 a,b keepempty=true | fields a,b,c
- source = table | dedup 2 a | fields a,b,c

- source = table | dedup 2 a,b | fields a,b,c
- source = table | dedup 2 a keepempty=true | fields a,b,c
- source = table | dedup 2 a,b keepempty=true | fields a,b,c
- source = table | dedup 1 a consecutive=true| fields a,b,c (連続重複排除はサポートされていません)

制限

- | dedup 2 a, b keepempty=false 用

```
DataFrameDropColumns('_row_number_)
+- Filter ('_row_number_ <= 2) // allowed duplication = 2
   +- Window [row_number() windowspecdefinition('a, 'b, 'a ASC NULLS FIRST, 'b ASC
 NULLS FIRST, specifiedwindowframe(RowFrame, unboundedpreceding$(), currentrow$()))
 AS _row_number_], ['a, 'b], ['a ASC NULLS FIRST, 'b ASC NULLS FIRST]
      +- Filter (isnotnull('a) AND isnotnull('b)) // keepempty=false
         +- Project
            +- UnresolvedRelation
```

- | dedup 2 a, b keepempty=true 用

```
Union
:- DataFrameDropColumns('_row_number_)
:  +- Filter ('_row_number_ <= 2)
:     +- Window [row_number() windowspecdefinition('a, 'b, 'a ASC NULLS FIRST, 'b ASC
 NULLS FIRST, specifiedwindowframe(RowFrame, unboundedpreceding$(), currentrow$()))
 AS _row_number_], ['a, 'b], ['a ASC NULLS FIRST, 'b ASC NULLS FIRST]
:        +- Filter (isnotnull('a) AND isnotnull('b))
:           +- Project
:              +- UnresolvedRelation
+- Filter (isnull('a) OR isnull('b))
   +- Project
      +- UnresolvedRelation
```

コマンドの説明

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`describe` コマンドを使用して、テーブル、スキーマ、カタログの構造とメタデータに関する詳細情報を取得します。`describe` コマンドのさまざまな例とユースケースを次に示します。

説明

- `describe table` このコマンドは `DESCRIBE EXTENDED table` SQL コマンドと同等です
- `describe schema.table`
- `describe schema.`table``
- `describe catalog.schema.table`
- `describe catalog.schema.`table``
- `describe `catalog`.`schema`.`table``

評価コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`eval` コマンドは式を評価し、検索結果に結果を追加します。

構文

次の構文を使用します。

```
eval <field>=<expression> ["," <field>=<expression> ]...
```

- `field`: 必須。フィールド名が存在しない場合は、新しいフィールドが追加されます。このフィールド名がすでに存在している場合、上書きされます。

- expression: 必須。システムによってサポートされている任意の式。

例 1: 新しいフィールドを作成する

この例では、ドキュメントごとに新しい doubleAge フィールドを作成する方法を示します。新しい doubleAge は、年齢に 2 を掛けた評価結果です。

PPL クエリ:

```
os> source=accounts | eval doubleAge = age * 2 | fields age, doubleAge ;
fetched rows / total rows = 4/4
+-------+-------------+
| age   | doubleAge   |
|-------+-------------|
| 32    | 64          |
| 36    | 72          |
| 28    | 56          |
| 33    | 66          |
+-------+-------------+
```

例 2: 既存のフィールドを上書きする

この例では、既存の年齢フィールドを年齢プラス 1 で上書きする方法を示します。

PPL クエリ:

```
os> source=accounts | eval age = age + 1 | fields age ;
fetched rows / total rows = 4/4
+-------+
| age   |
|-------|
| 33    |
| 37    |
| 29    |
| 34    |
+-------+
```

例 3: 評価で定義されたフィールドを使用して新しいフィールドを作成する

この例では、評価コマンドで定義された ddAge フィールドを使用して新しいフィールドを作成する方法を示します。新しいフィールド ddAge は doubleAge の評価結果に 2 を掛けた値で、doubleAge は評価コマンドで定義されます。

PPL クエリ:

```
os> source=accounts | eval doubleAge = age * 2, ddAge = doubleAge * 2 | fields age,
 doubleAge, ddAge ;
fetched rows / total rows = 4/4
+-------+-------------+---------+
| age   | doubleAge   | ddAge   |
|-------+-------------+---------|
| 32    | 64          | 128     |
| 36    | 72          | 144     |
| 28    | 56          | 112     |
| 33    | 66          | 132     |
+-------+-------------+---------+
```

前提条件: a、b、c は table 内の既存のフィールドです

その他の例

- source = table | eval f = 1 | fields a,b,c,f
- source = table | eval f = 1 (出力 a、b、c、f フィールド)
- source = table | eval n = now() | eval t = unix_timestamp(a) | fields n,t
- source = table | eval f = a | where f > 1 | sort f | fields a,b,c | head 5
- source = table | eval f = a * 2 | eval h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = b | stats avg(f) by h
- source = table | eval f = ispresent(a)
- source = table | eval r = coalesce(a, b, c) | fields r
- source = table | eval e = isempty(a) | fields e
- source = table | eval e = isblank(a) | fields e
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one', a = 2, 'two', a = 3, 'three', a = 4, 'four', a = 5, 'five', a = 6, 'six', a = 7, 'se7en', a = 8, 'eight', a = 9, 'nine')
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else 'unknown')

- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else concat(a, ' is an incorrect binary digit'))
- source = table | eval f = a in ('foo', 'bar') | fields f
- source = table | eval f = a not in ('foo', 'bar') | fields f

ケース例を使用した評価:

```
source = table | eval e = eval status_category =
case(a >= 200 AND a < 300, 'Success',
a >= 300 AND a < 400, 'Redirection',
a >= 400 AND a < 500, 'Client Error',
a >= 500, 'Server Error'
else 'Unknown')
```

別のケースでの評価の例:

前提条件: a、b、c は table 内の既存のフィールドです

その他の例

- source = table | eval f = 1 | fields a,b,c,f
- source = table | eval f = 1 (出力 a、b、c、f フィールド)
- source = table | eval n = now() | eval t = unix_timestamp(a) | fields n,t
- source = table | eval f = a | where f > 1 | sort f | fields a,b,c | head 5
- source = table | eval f = a * 2 | eval h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = b | stats avg(f) by h
- source = table | eval f = ispresent(a)
- source = table | eval r = coalesce(a, b, c) | fields r
- source = table | eval e = isempty(a) | fields e
- source = table | eval e = isblank(a) | fields e
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one', a = 2, 'two', a = 3, 'three', a = 4, 'four', a = 5, 'five', a = 6, 'six', a = 7, 'se7en', a = 8, 'eight', a = 9, 'nine')

- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else 'unknown')
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else concat(a, ' is an incorrect binary digit'))
- source = table | eval f = a in ('foo', 'bar') | fields f
- source = table | eval f = a not in ('foo', 'bar') | fields f

ケース例を使用した評価:

```
source = table | eval e = eval status_category =
case(a >= 200 AND a < 300, 'Success',
a >= 300 AND a < 400, 'Redirection',
a >= 400 AND a < 500, 'Client Error',
a >= 500, 'Server Error'
else 'Unknown')
```

別のケースでの評価の例:

```
source = table |  where ispresent(a) |
eval status_category =
 case(a >= 200 AND a < 300, 'Success',
  a >= 300 AND a < 400, 'Redirection',
  a >= 400 AND a < 500, 'Client Error',
  a >= 500, 'Server Error'
  else 'Incorrect HTTP status code'
 )
 | stats count() by status_category
```

制限事項

- 既存のフィールドの上書きはサポートされていません。そのような操作を試みるクエリは、「参照 'a' が曖昧です」というメッセージで例外をスローします。

```
- `source = table | eval a = 10 | fields a,b,c`
- `source = table | eval a = a * 2 | stats avg(a)`
- `source = table | eval a = abs(a) | where a > 0`
- `source = table | eval a = signum(a) | where a < 0`
```

eventstats コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

eventstats コマンドを使用して、計算されたサマリー統計でイベントデータを強化します。イベント内の指定されたフィールドを分析し、さまざまな統統計的指標を計算し、これらの結果を各元のイベントに新しいフィールドとして追加することで動作します。

eventstats の主な側面

1. 結果セット全体または定義されたグループ内で計算を実行します。
2. 元のイベントはそのまま残り、統計結果を含む新しいフィールドが追加されます。
3. コマンドは、比較分析、外れ値の特定、個々のイベントへの追加のコンテキストの提供に特に役立ちます。

統計と eventstats の違い

stats および eventstats コマンドはどちらも統計の計算に使用されますが、動作方法と生成内容にはいくつかの重要な違いがあります。

出力形式

- stats: 計算された統計のみを含む概要テーブルを生成します。
- eventstats: 計算された統計を既存のイベントに新しいフィールドとして追加し、元のデータを保持します。

イベント保持

- stats: 結果セットを統計概要のみに減らし、個々のイベントを破棄します。
- eventstats: 元のイベントをすべて保持し、計算された統計を含む新しいフィールドを追加します。

ユースケース

- stats: 概要レポートまたはダッシュボードの作成に最適です。多くの場合、結果を要約するための最終コマンドとして使用されます。
- eventstats: さらなる分析やフィルタリングのために統計コンテキストでイベントを補強する必要がある場合に役立ちます。検索中に使用して、後続のコマンドで使用できる統計を追加できます。

構文

次の構文を使用します。

```
eventstats <aggregation>... [by-clause]
```

集計

- 必須。
- 集計関数。
- 集計の引数はフィールドである必要があります。

句ごと

- オプション。
- 構文: by [span-expression,] [field,]...
- by 句には、スカラー関数や集計関数などのフィールドと式を含めることができます。span 句を使用して、特定のフィールドを同じ間隔のバケットに分割することもできます。その後、eventstats コマンドは、これらのスパンバケットに基づいて集計を実行します。
- デフォルト: by 句を指定しない場合、eventstats コマンドは結果セット全体で集計されます。

span-expression

- オプション、最大 1 つまで。
- 構文: span(field_expr, interval_expr)
- 間隔式の単位は、デフォルトでは自然単位です。ただし、日付と時刻タイプのフィールドでは、日付/時刻の単位を使用するときに間隔式で単位を指定する必要があります。

例えば、age フィールドを 10 年ごとにバケットに分割するには、span(age, 10) を使用します。時間ベースのフィールドでは、span(timestamp, 1h) を使用して timestamp フィールドを時間単位の間隔に分割できます。

使用可能な時間単位

スパン間隔の単位
ミリ秒 (ms)
秒 (秒)
分 (m、大文字と小文字を区別する)
時間 (h)
日 (d)
週 (w)
月 (M、大文字と小文字を区別する)
四半期 (q)
年 (y)

集計関数

COUNT

COUNT は、SELECT ステートメントによって取得された行の expr の数をカウントして返します。

CloudWatch Logs では、COUNT はサポートされません。

例:

```
os> source=accounts | eventstats count();
fetched rows / total rows = 4/4
+----------------+---------+-----------+----------+-----+--------+-------------------
+------------+--------------------------+--------+-------+---------+
```

```
| account_number | balance  | firstname | lastname | age | gender | address
 | employer   | email                    | city   | state | count() |
+----------------+----------+-----------+----------+-----+--------+--------------------
+------------+--------------------------+--------+-------+---------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
 | AnyCorp    | janedoe@anycorp.com      | Brogan | IL    | 4       |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
 | AnyCompany | marymajor@anycompany.com | Dante  | TN    | 4       |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
 | AnyOrg     |                          | Nogal  | VA    | 4       |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
 |            | juanli@exampleorg.com    | Orick  | MD    | 4       |
+----------------+----------+-----------+----------+-----+--------+--------------------
+------------+--------------------------+--------+-------+---------+
```

SUM

SUM(expr) は expr の合計を返します。

例:

```
os> source=accounts | eventstats sum(age) by gender;
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+-----------------------+------------+--------------------------+--------+-------
+--------------------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer   | email                    | city   | state | sum(age) by gender |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+------------+--------------------------+--------+-------
+--------------------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | 880 Any Lane
   | AnyCorp    | janedoe@anycorp.com      | Brogan | IL    | 101                |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | AnyCompany | marymajor@anycompany.com | Dante  | TN    | 101                |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg     |                          | Nogal  | VA    | 28                 |
| 18             | 4180     | Juan      | Li       | 33  | M      | 467 Example Court
   |            | juanli@exampleorg.com    | Orick  | MD    | 101                |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+------------+--------------------------+--------+-------
+--------------------+
```

AVG

AVG(expr) は expr の平均値を返します。

例:

```
os> source=accounts | eventstats avg(age) by gender;
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+-----------------------+------------+---------------------------+--------+-------
+--------------------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                     | city   | state | avg(age) by gender |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+------------+---------------------------+--------+-------
+--------------------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | 880 Any Lane
   | AnyCorp     | janedoe@anycorp.com       | Brogan | IL    | 33.67              |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 33.67              |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg      |                           | Nogal  | VA    | 28.00              |
| 18             | 4180     | Juan      | Li       | 33  | M      | 467 Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 33.67              |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+---------------------------+--------+-------
+--------------------+
```

MAX

MAX(expr) は expr の最大値を返します。

例

```
os> source=accounts | eventstats max(age);
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+---------------------------+--------+-------
+-----------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                     | city   | state | max(age)  |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+---------------------------+--------+-------
+-----------+
```

```
| 1              | 39225    | Jane      | Doe      | 32  | M      | 880 Any Lane
   | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 36        |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 36        |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg      |                          | Nogal  | VA    | 36        |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 36        |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+
```

MIN

MIN(expr) は expr の最小値を返します。

例

```
os> source=accounts | eventstats min(age);
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                    | city   | state | min(age)  |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | 880 Any Lane
   | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 28        |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 28        |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | *** Any Street
   | AnyOrg      |                          | Nogal  | VA    | 28        |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 28        |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+
```

STDDEV_SAMP

STDDEV_SAMP(expr) は expr のサンプル標準偏差を返します。

例

```
os> source=accounts | eventstats stddev_samp(age);
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                     | city   | state | stddev_samp(age)       |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
   | AnyCorp     | janedoe@anycorp.com       | Brogan | IL    | 3.304037933599835      |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 3.304037933599835      |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg      |                           | Nogal  | VA    | 3.304037933599835      |
| 18             | 4180     | Juan      | Li       | 33  | M      | 467 Example Court
   |             | juanli@exampleorg.com     | Orick  | MD    | 3.304037933599835      |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
```

STDDEV_POP

STDDEV_POP(expr) は expr の母集団標準偏差を返します。

例

```
os> source=accounts | eventstats stddev_pop(age);
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                     | city   | state | stddev_pop(age)        |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | 880 Any Lane
   | AnyCorp     | janedoe@anycorp.com       | Brogan | IL    | 2.***************     |
```

```
| 6              | 5686     | Mary      | Major    | 36  | M      | *** Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 2.***************     |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | *** Any Street
   | AnyOrg      |                          | Nogal  | VA    | 2.***************     |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 2.***************     |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+------------------------+
```

PERCENTILE または PERCENTILE_APPROX

PERCENTILE(expr, percent) または PERCENTILE_APPROX(expr, percent) は expr の近似パーセンタイル値を指定されたパーセンテージで返します。

パーセント

- 数値は 0 ~ 100 の定数である必要があります。

例

```
os> source=accounts | eventstats percentile(age, 90) by gender;
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+--------------------------------+
| account_number | balance  | firstname | lastname | age | gender | address
    | employer    | email                    | city   | state | percentile(age, 90) by
 gender  |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+--------------------------------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
   | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 36
       |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 36
       |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg      |                          | Nogal  | VA    | 28
       |
```

```
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 36
       |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+--------------------------------+
```

例 1: フィールドの平均、合計、数をグループごとに計算する

この例では、性別ごとにグループ分けした全アカウントの平均年齢、年齢の合計、およびイベント数を算出します。

```
os> source=accounts | eventstats avg(age) as avg_age, sum(age) as sum_age, count() as
 count by gender;
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+-----------+-------+
| account_number | balance  | firstname | lastname | age | gender | address
    | employer    | email                    | city   | state | avg_age   | sum_age   |
 count |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+-----------+-------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
    | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 33.666667 | 101       |
 3     |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
    | Any Company | marymajor@anycompany.com | Dante  | TN    | 33.666667 | 101       |
 3     |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
    | AnyOrg      |                          | Nogal  | VA    | 28.000000 | 28        |
 1     |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
    |             | juanli@exampleorg.com    | Orick  | MD    | 33.666667 | 101       |
 3     |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+--------------------------+--------+-------
+-----------+-----------+-------+
```

例 2: スパンごとにカウントを計算する

この例では、年齢を 10 年間隔でカウントして集計します。

```
os> source=accounts | eventstats count(age) by span(age, 10) as age_span
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+--------------------------+--------+-------
+----------+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                    | city   | state | age_span |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+--------------------------+--------+-------
+----------+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
   | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 3        |
| 6              | 5686     | Mary      | Major    | 36  | M      | 671 Example Street
   | Any Company | marymajor@anycompany.com | Dante  | TN    | 3        |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | 789 Any Street
   | AnyOrg      |                          | Nogal  | VA    | 1        |
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com    | Orick  | MD    | 3        |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+--------------------------+--------+-------
+----------+
```

例 3: 性別とスパンごとにカウントを計算する

この例では、年齢のカウントを 5 年間隔で集計し、性別ごとにグループ化します。

```
os> source=accounts | eventstats count() as cnt by span(age, 5) as age_span, gender
fetched rows / total rows = 4/4
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+--------------------------+--------+-------
+-----+
| account_number | balance  | firstname | lastname | age | gender | address
   | employer    | email                    | city   | state | cnt |
+----------------+----------+-----------+----------+-----+--------
+-----------------------+-------------+--------------------------+--------+-------
+-----+
| 1              | 39225    | Jane      | Doe      | 32  | M      | *** Any Lane
   | AnyCorp     | janedoe@anycorp.com      | Brogan | IL    | 2   |
| 6              | 5686     | Mary      | Majo     | 36  | M      | 671 Example Street
   | Any Company | hattiebond@anycompany.com | Dante  | TN    | 1   |
| 13             | 32838    | Nikki     | Wolf     | 28  | F      | *** Any Street
   | AnyOrg      |                          | Nogal  | VA    | 1   |
```

```
| 18             | 4180     | Juan      | Li       | 33  | M      | *** Example Court
   |             | juanli@exampleorg.com     | Orick  | MD    | 2   |
+----------------+----------+-----------+----------+-----+--------
+----------------------+-------------+---------------------------+--------+-------
+-----+
```

使用方法

- source = table | eventstats avg(a)
- source = table | where a < 50 | eventstats avg(c)
- source = table | eventstats max(c) by b
- source = table | eventstats count(c) by b | head 5
- source = table | eventstats distinct_count(c)
- source = table | eventstats stddev_samp(c)
- source = table | eventstats stddev_pop(c)
- source = table | eventstats percentile(c, 90)
- source = table | eventstats percentile_approx(c, 99)

スパンによる集計

- source = table | eventstats count(a) by span(a, 10) as a_span
- source = table | eventstats sum(age) by span(age, 5) as age_span | head 2
- source = table | eventstats avg(age) by span(age, 20) as age_span, country | sort - age_span | head 2

時間枠スパンによる集計 (タンブルウィンドウ関数)

- source = table | eventstats sum(productsAmount) by span(transactionDate, 1d) as age_date | sort age_date
- source = table | eventstats sum(productsAmount) by span(transactionDate, 1w) as age_date, productId

複数のレベルによる集計グループ

- `source = table | eventstats avg(age) as avg_state_age by country, state | eventstats avg(avg_state_age) as avg_country_age by country`
- `source = table | eventstats avg(age) as avg_city_age by country, state, city | eval new_avg_city_age = avg_city_age - 1 | eventstats avg(new_avg_city_age) as avg_state_age by country, state | where avg_state_age > 18 | eventstats avg(avg_state_age) as avg_adult_country_age by country`

拡張コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

expand コマンドを使用して、次の型のフィールドをフラット化します:

- `Array<Any>`
- `Map<Any>`

構文

次の構文を使用します。

```
expand <field> [As alias]
```

field

- 拡張 (展開) するフィールド。サポートされているタイプである必要があります。

alias

- オプション。元のフィールド名の代わりに使用される名前。

使用方法

expand コマンドは、指定された配列またはマップフィールド内の各要素の行を生成します。ここでは、

- 配列要素は個々の行になります。
- マップキーと値のペアは別々の行に分割され、各キーと値は行として表されます。

- エイリアスを指定すると、展開された値は元のフィールド名ではなくエイリアスで表されます。
- これは、stats、eval、parse などの他のコマンドと組み合わせて使用して、拡張後のデータを操作または抽出できます。

例

- `source = table | expand employee | stats max(salary) as max by state, company`
- `source = table | expand employee as worker | stats max(salary) as max by state, company`
- `source = table | expand employee as worker | eval bonus = salary * 3 | fields worker, bonus`
- `source = table | expand employee | parse description '(?<email>.+@.+)' | fields employee, email`
- `source = table | eval array=json_array(1, 2, 3) | expand array as uid | fields name, occupation, uid`
- `source = table | expand multi_valueA as multiA | expand multi_valueB as multiB`

explain コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

explain コマンドを使用すると、クエリ実行プランを理解し、クエリを分析して最適化してパフォーマンスを向上させることができます。この概要では、explain コマンドの目的とクエリの最適化における重要性について簡潔に説明します。

Comment

- `source=accounts | top gender // finds most common gender of all the accounts` (行コメント)
- `source=accounts | dedup 2 gender /* dedup the document with gender field keep 2 duplication */ | fields account_number, gender` (ブロックコメント)

説明

- `describe table` このコマンドは `DESCRIBE EXTENDED table` SQL コマンドと同等です
- `describe schema.table`
- `describe schema.`table``
- `describe catalog.schema.table`
- `describe catalog.schema.`table``
- `describe `catalog`.`schema`.`table``

Explain (説明する)

- `explain simple | source = table | where a = 1 | fields a,b,c`
- `explain extended | source = table`
- `explain codegen | source = table | dedup a | fields a,b,c`
- `explain cost | source = table | sort a | fields a,b,c`
- `explain formatted | source = table | fields - a`
- `explain simple | describe table`

フィールド

- `source = table`
- `source = table | fields a,b,c`
- `source = table | fields + a,b,c`

- source = table | fields - b,c
- source = table | eval b1 = b | fields - b1,c

フィールドの概要

- source = t | fieldsummary includefields=status_code nulls=false
- source = t | fieldsummary includefields= id, status_code, request_path nulls=true
- source = t | where status_code != 200 | fieldsummary includefields= status_code nulls=true

ネスト化されたフィールド

- source = catalog.schema.table1, catalog.schema.table2 | fields A.nested1, B.nested1
- source = catalog.table | where struct_col2.field1.subfield > 'valueA' | sort int_col | fields int_col, struct_col.field1.subfield, struct_col2.field1.subfield
- source = catalog.schema.table | where struct_col2.field1.subfield > 'valueA' | sort int_col | fields int_col, struct_col.field1.subfield, struct_col2.field1.subfield

フィルター

- source = table | where a = 1 | fields a,b,c
- source = table | where a >= 1 | fields a,b,c
- source = table | where a < 1 | fields a,b,c
- source = table | where b != 'test' | fields a,b,c
- source = table | where c = 'test' | fields a,b,c | head 3
- source = table | where ispresent(b)
- source = table | where isnull(coalesce(a, b)) | fields a,b,c | head 3
- source = table | where isempty(a)
- source = table | where isblank(a)

- `source = table | where case(length(a) > 6, 'True' else 'False') = 'True'`
- `source = table | where a not in (1, 2, 3) | fields a,b,c`
- `source = table | where a between 1 and 4` – 注意: これは a >= 1 と a <= 4 を返します。つまり範囲は [1, 4] です。
- `source = table | where b not between '2024-09-10' and '2025-09-10'` – 注: これは b >= '**********' と b <= '2025-09-10' を返します。
- `source = table | where cidrmatch(ip, '***********/24')`
- `source = table | where cidrmatch(ipv6, '2003:db8::/32')`
- `source = table | trendline sma(2, temperature) as temp_trend`

IP 関連のクエリ

- `source = table | where cidrmatch(ip, '***************')`
- `source = table | where isV6 = false and isValid = true and cidrmatch(ipAddress, '***************')`
- `source = table | where isV6 = true | eval inRange = case(cidrmatch(ipAddress, '2003:***::/32'), 'in' else 'out') | fields ip, inRange`

複雑なフィルター

```
source = table | eval status_category =
case(a >= 200 AND a < 300, 'Success',
    a >= 300 AND a < 400, 'Redirection',
    a >= 400 AND a < 500, 'Client Error',
    a >= 500, 'Server Error'
else 'Incorrect HTTP status code')
| where case(a >= 200 AND a < 300, 'Success',
    a >= 300 AND a < 400, 'Redirection',
    a >= 400 AND a < 500, 'Client Error',
    a >= 500, 'Server Error'
else 'Incorrect HTTP status code'
) = 'Incorrect HTTP status code'
```

```
source = table
```

```
| eval factor = case(a > 15, a - 14, isnull(b), a - 7, a < 3, a + 1 else 1)
| where case(factor = 2, 'even', factor = 4, 'even', factor = 6, 'even', factor = 8,
 'even' else 'odd') = 'even'
| stats count() by factor
```

論理条件を使用したフィルター

- source = table | where c = 'test' AND a = 1 | fields a,b,c
- source = table | where c != 'test' OR a > 1 | fields a,b,c | head 1
- source = table | where c = 'test' NOT a > 1 | fields a,b,c

評価

前提条件: a、b、c は table 内の既存のフィールドです

- source = table | eval f = 1 | fields a,b,c,f
- source = table | eval f = 1 (出力 a、b、c、f フィールド)
- source = table | eval n = now() | eval t = unix_timestamp(a) | fields n,t
- source = table | eval f = a | where f > 1 | sort f | fields a,b,c | head 5
- source = table | eval f = a * 2 | eval h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = f * 2 | fields a,f,h
- source = table | eval f = a * 2, h = b | stats avg(f) by h
- source = table | eval f = ispresent(a)
- source = table | eval r = coalesce(a, b, c) | fields r
- source = table | eval e = isempty(a) | fields e
- source = table | eval e = isblank(a) | fields e
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one', a = 2, 'two', a = 3, 'three', a = 4, 'four', a = 5, 'five', a = 6, 'six', a = 7, 'se7en', a = 8, 'eight', a = 9, 'nine')
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else 'unknown')
- source = table | eval f = case(a = 0, 'zero', a = 1, 'one' else concat(a, ' is an incorrect binary digit'))

- `source = table | eval digest = md5(fieldName) | fields digest`
- `source = table | eval digest = sha1(fieldName) | fields digest`
- `source = table | eval digest = sha2(fieldName,256) | fields digest`
- `source = table | eval digest = sha2(fieldName,512) | fields digest`

fillnull コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

説明

`fillnull` コマンドを使用して、null 値を検索結果の 1 つ以上のフィールド内の指定した値に置き換えます。

構文

次の構文を使用します。

```
fillnull [with <null-replacement> in <nullable-field>["," <nullable-field>]] | [using
 <source-field> = <null-replacement> ["," <source-field> = <null-replacement>]]
```

- null-replacement: 必須。null 値を置き換えるために使用される値。
- nullable-field: 必須。フィールドリファレンス。このフィールドの null 値は、null 置換で指定された値に置き換えられます。

例 1: Fillnull 1 フィールド

この例では、単一のフィールドで fillnull を使用する方法を示しています。

```
os> source=logs | fields status_code | eval input=status_code | fillnull with 0 in
 status_code;
| input | status_code |
|-------|-------------|
| 403   | 403         |
```

```
| 403   | 403         |
| NULL  | 0           |
| NULL  | 0           |
| 200   | 200         |
| 404   | 404         |
| 500   | 500         |
| NULL  | 0           |
| 500   | 500         |
| 404   | 404         |
| 200   | 200         |
| 500   | 500         |
| NULL  | 0           |
| NULL  | 0           |
| 404   | 404         |
```

例 2: 複数のフィールドに適用される Fillnull

この例では、複数のフィールドに適用された fillnull を示しています。

```
os> source=logs | fields request_path, timestamp | eval
 input_request_path=request_path, input_timestamp = timestamp | fillnull with '???' in
 request_path, timestamp;
| input_request_path | input_timestamp       | request_path | timestamp             |
|--------------------------------------------------------------------------------------|
| /contact           | NULL                  | /contact     | ???                   |
| /home              | NULL                  | /home        | ???                   |
| /about             | 2023-10-01 10:30:00   | /about       | 2023-10-01 10:30:00   |
| /home              | 2023-10-01 10:15:00   | /home        | 2023-10-01 10:15:00   |
| NULL               | 2023-10-01 10:20:00   | ???          | 2023-10-01 10:20:00   |
| NULL               | 2023-10-01 11:05:00   | ???          | 2023-10-01 11:05:00   |
| /about             | NULL                  | /about       | ???                   |
| /home              | 2023-10-01 10:00:00   | /home        | 2023-10-01 10:00:00   |
| /contact           | NULL                  | /contact     | ???                   |
| NULL               | 2023-10-01 10:05:00   | ???          | 2023-10-01 10:05:00   |
| NULL               | 2023-10-01 10:50:00   | ???          | 2023-10-01 10:50:00   |
| /services          | NULL                  | /services    | ???                   |
| /home              | 2023-10-01 10:45:00   | /home        | 2023-10-01 10:45:00   |
| /services          | 2023-10-01 11:00:00   | /services    | 2023-10-01 11:00:00   |
| NULL               | 2023-10-01 10:35:00   | ???          | 2023-10-01 10:35:00   |
```

例 3: Fillnull は、さまざまな null 置換値を持つ複数のフィールドに適用されます。

この例では、null を置き換えるために使用されるさまざまな値を持つ fillnull を示しています。

- request_path フィールドの /error
- timestamp フィールドの 1970-01-01 00:00:00

```
os> source=logs | fields request_path, timestamp | eval
 input_request_path=request_path, input_timestamp = timestamp | fillnull using
 request_path = '/error', timestamp='1970-01-01 00:00:00';

| input_request_path | input_timestamp       | request_path | timestamp              |
|--------------------------------------------------------------------------------|
| /contact           | NULL                  | /contact     | 1970-01-01 00:00:00    |
| /home              | NULL                  | /home        | 1970-01-01 00:00:00    |
| /about             | 2023-10-01 10:30:00   | /about       | 2023-10-01 10:30:00    |
| /home              | 2023-10-01 10:15:00   | /home        | 2023-10-01 10:15:00    |
| NULL               | 2023-10-01 10:20:00   | /error       | 2023-10-01 10:20:00    |
| NULL               | 2023-10-01 11:05:00   | /error       | 2023-10-01 11:05:00    |
| /about             | NULL                  | /about       | 1970-01-01 00:00:00    |
| /home              | 2023-10-01 10:00:00   | /home        | 2023-10-01 10:00:00    |
| /contact           | NULL                  | /contact     | 1970-01-01 00:00:00    |
| NULL               | 2023-10-01 10:05:00   | /error       | 2023-10-01 10:05:00    |
| NULL               | 2023-10-01 10:50:00   | /error       | 2023-10-01 10:50:00    |
| /services          | NULL                  | /services    | 1970-01-01 00:00:00    |
| /home              | 2023-10-01 10:45:00   | /home        | 2023-10-01 10:45:00    |
| /services          | 2023-10-01 11:00:00   | /services    | 2023-10-01 11:00:00    |
| NULL               | 2023-10-01 10:35:00   | /error       | 2023-10-01 10:35:00    |
```

fields コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

fields コマンドを使用して、検索結果のフィールドを保持または削除します。

構文

次の構文を使用します。

```
field [+|-] <field-list>
```

- index: オプション。

  プラス (+) を使用すると、フィールドリストで指定されたフィールドのみが保持されます。

  マイナス (-) を使用すると、フィールドリストで指定されたすべてのフィールドが削除されます。

  デフォルト: +
- field list: 必須。保持または削除するフィールドのカンマ区切りリスト。

例 1: 結果から指定されたフィールドを選択する

この例では、検索結果から account_number、firstname、および lastname フィールドを取得する方法を示します。

PPL クエリ:

```
os> source=accounts | fields account_number, firstname, lastname;
fetched rows / total rows = 4/4
+------------------+-------------+------------+
| account_number   | firstname   | lastname   |
|------------------+-------------+------------|
| 1                | Jane        | Doe        |
| 6                | John        | Doe        |
| 13               | Jorge       | Souza      |
| 18               | Juan        | Li         |
+------------------+-------------+------------+
```

例 2: 指定されたフィールドを結果から削除する

この例では、検索結果から account_number フィールドを削除する方法を示します。

PPL クエリ:

```
os> source=accounts | fields account_number, firstname, lastname | fields -
 account_number ;
fetched rows / total rows = 4/4
+-------------+------------+
| firstname   | lastname   |
|-------------+------------|
| Jane        | Doe        |
| John        | Doe        |
| Jorge       | Souza      |
```

```
| Juan        | Li         |
+-------------+------------+
```

その他の例

- `source = table`
- `source = table | fields a,b,c`
- `source = table | fields + a,b,c`
- `source = table | fields - b,c`
- `source = table | eval b1 = b | fields - b1,c`

ネスト化フィールドの例:

```
`source = catalog.schema.table1, catalog.schema.table2 | fields A.nested1, B.nested1`
`source = catalog.table | where struct_col2.field1.subfield > 'valueA' | sort int_col |
 fields  int_col, struct_col.field1.subfield, struct_col2.field1.subfield`
`source = catalog.schema.table | where struct_col2.field1.subfield > 'valueA' | sort
 int_col | fields  int_col, struct_col.field1.subfield, struct_col2.field1.subfield`
```

flatten コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

flatten コマンドを使用して、次のタイプのフィールドを展開します。

- `struct<?,?>`
- `array<struct<?,?>>`

構文

次の構文を使用します。

```
flatten <field>
```

- field: フラット化するフィールド。フィールドはサポートされているタイプである必要があります。

Schema

col_name	data_type
_time	string
bridges	array<struct<length:bigint,name:string>>
city	string
coor	struct<alt:bigint,lat:double,long:double>
country	string

データ

_time	bridges	city	coor	country
2024-09-13T12:00:00	[{801、タワーブリッジ}, {928、ロンドン橋}]	ロンドン	{35, 51.5074, -0.1278}	英国
2024-09-13T12:00:00	[{232、ポンヌフ}, {160、アレクサンドル 3 世橋}]	Paris	{35, 48.8566, 2.3522}	フランス
2024-09-13T12:00:00	[{48、リアルト橋}, {11、ため息橋}]	ベネツィア	{2, 45.4408, 12.3155}	イタリア

_time	bridges	city	coor	country
2024-09-13T12:00:00	[{***、カレル橋}, {343、レギオン橋}]	プラハ	{200, 50.0755, 14.4378}	チェコ共和国
2024-09-13T12:00:00	[{375、セーチェー二鎖橋}, {333、自由橋}]	ブダペスト	{96, 47.4979, 19.0402}	ハンガリー
1990-09-13T12:00:00	NULL	ワルシャワ	NULL	ポーランド

例 1: 構造体をフラット化する

この例では、構造体フィールドをフラット化する方法を示します。

PPL クエリ:

```
source=table | flatten coor
```

_time	bridges	city	country	alt	lat	long
2024-09-13T12:00:00	[{801、タワーブリッジ}, {928、ロンドン橋}]	ロンドン	英国	35	51.5074	-0.1278
2024-09-13T12:00:00	[{232、ポンヌフ}, {160, ア	Paris	フランス	35	48.8566	2.3522

_time	bridges	city	country	alt	lat	long
	レクサンドル 3 世橋}]					
2024-09-13T12:00:00	[{48、リアルト橋}, {11、ため息橋}]	ベネツィア	イタリア	2	45.4408	12.3155
2024-09-13T12:00:00	[{516、カレル橋}、{343、レギオン橋}]	プラハ	チェコ共和国	200	50.0755	14.4378
2024-09-13T12:00:00	[{375、セーチェー二鎖橋}, {333、自由橋}]	ブダペスト	ハンガリー	96	47.4979	19.0402
1990-09-13T12:00:00	NULL	ワルシャワ	ポーランド	NULL	NULL	NULL

例 2: 配列をフラット化する

この例では、構造体フィールドの配列をフラット化する方法を示しています。

PPL クエリ:

```
source=table | flatten bridges
```

_time	city	coor	country	length	名前
2024-09-13T12:00:00	ロンドン	{35, 51.5074, -0.1278}	英国	801	タワーブリッジ
2024-09-13T12:00:00	ロンドン	{35, 51.5074, -0.1278}	英国	928	ロンドン橋
2024-09-13T12:00:00	Paris	{35, 48.8566, 2.3522}	フランス	232	ポンヌフ
2024-09-13T12:00:00	Paris	{35, 48.8566, 2.3522}	フランス	160	アレクサンドル 3 世橋
2024-09-13T12:00:00	ベネツィア	{2, 45.4408, 12.3155}	イタリア	48	リアルト橋
2024-09-13T12:00:00	ベネツィア	{2, 45.4408, 12.3155}	イタリア	11	ため息橋
2024-09-13T12:00:00	プラハ	{200, 50.0755, 14.4378}	チェコ共和国	516	カレル橋
2024-09-13T12:00:00	プラハ	{200, 50.0755, 14.4378}	チェコ共和国	343	レギオン橋
2024-09-13T12:00:00	ブダペスト	{96, 47.4979, 19.0402}	ハンガリー	375	セーチェー二鎖橋
2024-09-13T12:00:00	ブダペスト	{96, 47.4979, 19.0402}	ハンガリー	333	自由橋
1990-09-13T12:00:00	ワルシャワ	NULL	ポーランド	NULL	NULL

例 3: 配列と構造体をフラット化する

この例では、複数のフィールドをフラット化する方法を示します。

PPL クエリ:

```
source=table | flatten bridges | flatten coor
```

_time	city	country	length	名前	alt	lat	long
2024-09-13T12:00:00	ロンドン	英国	801	タワーブリッジ	35	51.5074	-0.1278
2024-09-13T12:00:00	ロンドン	英国	928	ロンドン橋	35	51.5074	-0.1278
2024-09-13T12:00:00	Paris	フランス	232	ポンヌフ	35	48.8566	2.3522
2024-09-13T12:00:00	Paris	フランス	160	アレクサンドル 3 世橋	35	48.8566	2.3522
2024-09-13T12:00:00	ベネツィア	イタリア	48	リアルト橋	2	45.4408	12.3155
2024-09-13T12:00:00	ベネツィア	イタリア	11	ため息橋	2	45.4408	12.3155
2024-09-13T12:00:00	プラハ	チェコ共和国	516	カレル橋	200	50.0755	14.4378
2024-09-13T12:00:00	プラハ	チェコ共和国	343	レギオン橋	200	50.0755	14.4378

_time	city	country	length	名前	alt	lat	long
2024-09-13T12:00:00	ブダペスト	ハンガリー	375	セーチェーニ鎖橋	96	47.4979	19.0402
2024-09-13T12:00:00	ブダペスト	ハンガリー	333	自由橋	96	47.4979	19.0402
1990-09-13T12:00:00	ワルシャワ	ポーランド	NULL	NULL	NULL	NULL	NULL

grok コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

grok コマンドは、テキストフィールドを grok パターンで解析し、検索結果に結果を追加します。

構文

次の構文を使用します。

```
grok <field> <pattern>
```

field

- 必須。
- フィールドはテキストフィールドである必要があります。

pattern

- 必須。

- 指定されたテキストフィールドから新しいフィールドを抽出するために使用される grok パターン。
- 新しいフィールド名が既に存在する場合、元のフィールドが置き換えられます。

Grok パターン

grok パターンは、各ドキュメントのテキストフィールドを照合して新しいフィールドを抽出するために使用されます。

例 1: 新しいフィールドを作成する

この例では、ドキュメントごとに新しいフィールド host を作成する方法を示します。host は email フィールドの @ の後に続くホスト名になります。null フィールドを解析すると、空の文字列が返されます。

```
os> source=accounts | grok email '.+@%{HOSTNAME:host}' | fields email, host ;
fetched rows / total rows = 4/4
+-------------------------+-------------+
| email                   | host        |
|-------------------------+-------------|
| jane_doe@example.com    | example.com |
| arnav_desai@example.net | example.net |
| null                    |             |
| juan_li@example.org     | example.org |
+-------------------------+-------------+
```

例 2: 既存のフィールドを上書きする

この例では、既存の address フィールドを番地を除去した状態で上書きする方法を示します。

```
os> source=accounts | grok address '%{NUMBER} %{GREEDYDATA:address}' | fields address ;
fetched rows / total rows = 4/4
+------------------+
| address          |
|------------------|
| Example Lane     |
| Any Street       |
| Main Street      |
| Example Court    |
+------------------+
```

例 3: grok を使用してログを解析する

この例では、grok を使用して raw ログを解析する方法を示します。

```
os> source=apache | grok message '%{COMMONAPACHELOG}' | fields COMMONAPACHELOG,
 timestamp, response, bytes ;
fetched rows / total rows = 4/4
+-----------------------------------------------------------------------------------------------------------
+----------------------------+------------+---------+
| COMMONAPACHELOG
                                          | timestamp                  | response   |
 bytes   |
|-----------------------------------------------------------------------------------------------------------
+----------------------------+------------+---------|
| 177.95.8.74 - upton5450 [28/Sep/2022:10:15:57 -0700] "HEAD /e-business/mindshare
 HTTP/1.0" 404 19927                        | 28/Sep/2022:10:15:57 -0700 | 404        |
 19927   |
| 127.45.152.6 - pouros8756 [28/Sep/2022:10:15:57 -0700] "GET /architectures/
convergence/niches/mindshare HTTP/1.0" 100 28722 | 28/Sep/2022:10:15:57 -0700 | 100
    | 28722   |
| *************** - - [28/Sep/2022:10:15:57 -0700] "PATCH /strategize/out-of-the-box
 HTTP/1.0" 401 27439                       | 28/Sep/2022:10:15:57 -0700 | 401        |
 27439   |
| ************** - - [28/Sep/2022:10:15:57 -0700] "POST /users HTTP/1.1" 301 9481
                                         | 28/Sep/2022:10:15:57 -0700 | 301        | 9481
    |
+-----------------------------------------------------------------------------------------------------------
+----------------------------+------------+---------+
```

制限事項

grok コマンドには、parse コマンドと同じ制限があります。

head コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

head コマンドを使用して、オプションのオフセットの以降の指定された結果の最初の N 数を検索順序で返します。

構文

次の構文を使用します。

```
head [<size>] [from <offset>]
```

<size>

- オプション整数
- 返される結果の数。
- デフォルト: 10

<offset>

- オプションの from の後の整数。
- スキップする結果の数。
- デフォルト: 0

例 1: 最初の 10 件の結果を取得する

この例では、アカウントインデックスから最大 10 件の結果を取得する方法を示します。

PPL クエリ:

```
os> source=accounts | fields firstname, age | head;
fetched rows / total rows = 4/4
+-------------+-------+
| firstname   | age   |
|-------------+-------|
| Jane        | 32    |
| John        | 36    |
| Jorge       | 28    |
| Juan        | 33    |
+-------------+-------+
```

例 2: 最初の N の結果を取得する

この例では、アカウントインデックスの最初の N の結果を示しています。

PPL クエリ:

```
os> source=accounts | fields firstname, age | head 3;
fetched rows / total rows = 3/3
+-------------+-------+
| firstname   | age   |
|-------------+-------|
| Jane        | 32    |
| John        | 36    |
| Jorge       | 28    |
+-------------+-------+
```

例 3: オフセット M の後に最初の N の結果を取得する

この例では、アカウントインデックスから M の結果をスキップした後に最初の N の結果を取得する方法を示します。

PPL クエリ:

```
os> source=accounts | fields firstname, age | head 3 from 1;
fetched rows / total rows = 3/3
+-------------+-------+
| firstname   | age   |
|-------------+-------|
| John        | 36    |
| Jorge       | 28    |
| Juan        | 33    |
+-------------+-------+
```

join コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

join コマンドを使用すると、共通フィールドに基づいて複数のソースからのデータを組み合わせることができ、複雑な分析を実行し、分散データセットからより深いインサイトを得ることができます。

Schema

otel-v1-apm-span-* (ラージ) と otel-v1-apm-service-map (スモール) の 2 つ以上のインデックスがあります。

インデックスの関連フィールド:

otel-v1-apm-span-*

- traceId – トレースの一意の識別子。同じトレースのすべてのスパンが同じ traceId を共有します。
- spanId – トレース内のスパンの一意の識別子。スパンの作成時に割り当てられます。
- parentSpanId – このスパンの親スパンの spanId。これがルートスパンである場合、このフィールドは空である必要があります。
- durationInNanos – startTime と endTime の差をナノ秒単位で表したもの (UI 上では latency として表示)。
- serviceName – スパンの送信元のリソース。
- traceGroup – トレースのルートスパンの名前。

otel-v1-apm-service-map

- serviceName – スパンを発行したサービスの名前。
- destination.domain – このクライアントによって呼び出されるサービスの serviceName。
- destination.resource – このクライアントによって呼び出されるスパン名 (API、オペレーションなど)。
- target.domain – クライアントによって呼び出されるサービスの serviceName。
- target.resource – クライアントによって呼び出されるスパン名 (API、オペレーションなど)。
- traceGroupName – リクエストチェーンを開始した最上位のスパン名。

要件

以下の計算を行うために、join をサポートします:

サービスごとに、サービスマップインデックスにスパンインデックスを結合して、さまざまなタイプのフィルターでメトリクスを計算します。

このサンプルクエリは、order サービスのトレースグループ client_cancel_order でフィルタリングされた場合のレイテンシーを計算します。

```
SELECT avg(durationInNanos)
```

```
FROM `otel-v1-apm-span-000001` t1
WHERE t1.serviceName = `order`
  AND ((t1.name in
          (SELECT target.resource
           FROM `otel-v1-apm-service-map`
           WHERE serviceName = `order`
             AND traceGroupName = `client_cancel_order`)
        AND t1.parentSpanId != NULL)
       OR (t1.parentSpanId = NULL
           AND t1.name = `client_cancel_order`))
  AND t1.traceId in
    (SELECT traceId
     FROM `otel-v1-apm-span-000001`
     WHERE serviceName = `order`)
```

PPL への移行

join コマンドの構文

```
SEARCH source=<left-table>
| <other piped command>
| [joinType] JOIN
    [leftAlias]
    ON joinCriteria
    <right-table>
| <other piped command>
```

書き換え

```
SEARCH source=otel-v1-apm-span-000001
| WHERE serviceName = 'order'
| JOIN left=t1 right=t2
    ON t1.traceId = t2.traceId AND t2.serviceName = 'order'
    otel-v1-apm-span-000001 -- self inner join
| EVAL s_name = t1.name -- rename to avoid ambiguous
| EVAL s_parentSpanId = t1.parentSpanId -- RENAME command would be better when it is
 supported
| EVAL s_durationInNanos = t1.durationInNanos
| FIELDS s_name, s_parentSpanId, s_durationInNanos -- reduce colunms in join
| LEFT JOIN left=s1 right=t3
    ON s_name = t3.target.resource AND t3.serviceName = 'order' AND t3.traceGroupName =
 'client_cancel_order'
```

```
    otel-v1-apm-service-map
| WHERE (s_parentSpanId IS NOT NULL OR (s_parentSpanId IS NULL AND s_name =
 'client_cancel_order'))
| STATS avg(s_durationInNanos) -- no need to add alias if there is no ambiguous
```

joinType

- 構文: INNER | LEFT OUTER | CROSS
- オプション
- 実行する結合のタイプ。指定されなかった場合、デフォルト値は INNER になります。

leftAlias

- 構文: left = <leftAlias>
- オプション
- あいまいな名前付けを避けるために、左結合で使用するサブクエリエイリアス。

joinCriteria

- 構文: <expression>
- 必須
- この構文は ON で始まります。任意の比較式にすることができます。通常、結合条件は <leftAlias>.<leftField>=<rightAlias>.<rightField> のようになります。

  例: l.id = r.id。結合条件に複数の条件が含まれている場合は、各比較式間で AND および OR 演算子を指定できます。例えば、l.id = r.id AND l.email = r.email AND (r.age > 65 OR r.age < 18)。

その他の の例

SQL クエリからの移行 (TPC-H Q13):

```
SELECT c_count, COUNT(*) AS custdist
FROM
  ( SELECT c_custkey, COUNT(o_orderkey) c_count
    FROM customer LEFT OUTER JOIN orders ON c_custkey = o_custkey
        AND o_comment NOT LIKE '%unusual%packages%'
```

```
    GROUP BY c_custkey
  ) AS c_orders
GROUP BY c_count
ORDER BY custdist DESC, c_count DESC;
```

PPL 結合クエリによって書き換えられました。

```
SEARCH source=customer
| FIELDS c_custkey
| LEFT OUTER JOIN
    ON c_custkey = o_custkey AND o_comment NOT LIKE '%unusual%packages%'
    orders
| STATS count(o_orderkey) AS c_count BY c_custkey
| STATS count() AS custdist BY c_count
| SORT - custdist, - c_count
```

制限: サブ検索は、右結合ではサポートされていません。

サブ検索がサポートされている場合は、上記の PPL クエリを次のように書き換えることができます。

```
SEARCH source=customer
| FIELDS c_custkey
| LEFT OUTER JOIN
   ON c_custkey = o_custkey
   [
      SEARCH source=orders
      | WHERE o_comment NOT LIKE '%unusual%packages%'
      | FIELDS o_orderkey, o_custkey
   ]
| STATS count(o_orderkey) AS c_count BY c_custkey
| STATS count() AS custdist BY c_count
| SORT - custdist, - c_count
```

lookup コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

lookup コマンドを使用して、ルックアップインデックス (ディメンションテーブル) からデータを追加または置換することで、検索データを充実させます。このコマンドを使用すると、ディメンションテーブルの値を使用してインデックスのフィールドを拡張できます。これを使用して、ルックアップ条件が満たされたときに値を追加または置き換えることもできます。lookup コマンドは、静的データセットを使用してソースデータを充実させる Join コマンドよりも適しています。

構文

次の構文を使用します。

```
SEARCH source=<sourceIndex>
| <other piped command>
| LOOKUP <lookupIndex> (<lookupMappingField> [AS <sourceMappingField>])...
    [(REPLACE | APPEND) (<inputField> [AS <outputField>])...]
| <other piped command>
```

lookupIndex

- 必須。
- ルックアップインデックスの名前 (ディメンションテーブル)。

lookupMappingField

- 必須。
- ルックアップインデックス内のマッピングキーで、右側のテーブルにおける結合キーに相当します。フィールドはカンマで区切って、複数のフィールドを指定できます。

sourceMappingField

- オプション。
- デフォルト: <lookupMappingField>。
- ソースクエリからのマッピングキーで、左側の結合キーに相当します。

inputField

- オプション。
- デフォルト: 一致した値が見つかったルックアップインデックスのすべてのフィールド。

- 一致した値が結果出力に適用されるルックアップインデックスのフィールド。フィールドはカンマで区切って、複数のフィールドを指定できます。

outputField

- オプション。
- デフォルト: <inputField>。
- 出力内のフィールド。複数の出力フィールドを指定できます。ソースクエリから既存のフィールド名を指定した場合、その値は inputField の一致する値に置き換えられるか、追加されます。新しいフィールド名を指定すると、結果に追加されます。

REPLACE | APPEND

- オプション。
- デフォルト: REPLACE
- 一致した値を処理する方法を指定します。REPLACE を指定すると、<lookupIndex> フィールドの一致した値が結果の値を上書きします。APPEND を指定した場合、<lookupIndex> フィールドの一致値は、結果の欠損値にのみ付加されます。

使用方法

- LOOKUP <lookupIndex> id AS cid REPLACE mail AS email
- LOOKUP <lookupIndex> name REPLACE mail AS email
- LOOKUP <lookupIndex> id AS cid, name APPEND address, mail AS email
- LOOKUP <lookupIndex> id

例

次の 例を参照してください。

```
SEARCH source=<sourceIndex>
| WHERE orderType = 'Cancelled'
| LOOKUP account_list, mkt_id AS mkt_code REPLACE amount, account_name AS name
| STATS count(mkt_code), avg(amount) BY name
```

```
SEARCH source=<sourceIndex>
```

```
| DEDUP market_id
| EVAL category=replace(category, "-", ".")
| EVAL category=ltrim(category, "dvp.")
| LOOKUP bounce_category category AS category APPEND classification
```

```
SEARCH source=<sourceIndex>
| LOOKUP bounce_category category
```

parse コマンド

parse コマンドは、テキストフィールドを正規表現で解析し、検索結果にその結果を追加します。

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

構文

次の構文を使用します。

```
parse <field> <pattern>
```

field

- 必須。
- フィールドはテキストフィールドである必要があります。

pattern

- 必須文字列。
- これは、指定されたテキストフィールドから新しいフィールドを抽出するために使用される正規表現パターンです。
- 新しいフィールド名が既に存在する場合、元のフィールドが置き換えられます。

正規表現

正規表現パターンは、各ドキュメントのテキストフィールド全体を Java 正規表現エンジンと一致させるために使用されます。式内の各名前付きキャプチャグループは、新しい STRING フィールドになります。

例 1: 新しいフィールドを作成する

例では、ドキュメントごとに新しいフィールド host を作成する方法を示します。host は email フィールドの @ の後に続くホスト名になります。null フィールドを解析すると、空の文字列が返されます。

PPL クエリ:

```
os> source=accounts | parse email '.+@(?<host>.+)' | fields email, host ;
fetched rows / total rows = 4/4
+-----------------------+-------------+
| email                 | host        |
|-----------------------+-------------|
| jane_doe@example.com  | example.com |
| john_doe@example.net  | example.net |
| null                  |             |
| juan_li@example.org   | example.org |
+-----------------------+-------------+
```

例 2: 既存のフィールドを上書きする

例では、既存の address フィールドを番地を除去した状態で上書きする方法を示します。

PPL クエリ:

```
os> source=accounts | parse address '\d+ (?<address>.+)' | fields address ;
fetched rows / total rows = 4/4
+------------------+
| address          |
|------------------|
| Example Lane     |
| Example Street   |
| Example Avenue   |
| Example Court    |
+------------------+
```

例 3: キャスト解析フィールドでフィルタリングおよびソートする

この例では、address フィールド内の 500 より大きい番地番号を並べ替える方法を示しています。

PPL クエリ:

```
os> source=accounts | parse address '(?<streetNumber>\d+) (?<street>.+)' | where
 cast(streetNumber as int) > 500 | sort num(streetNumber) | fields streetNumber,
 street ;
fetched rows / total rows = 3/3
+----------------+----------------+
| streetNumber   | street         |
|----------------+----------------|
| ***            | Example Street |
| ***            | Example Avenue |
| 880            | Example Lane   |
+----------------+----------------+
```

制限事項

parse コマンドにはいくつかの制限があります。

- 解析で定義されたフィールドは、再度解析することはできません。

  次のコマンドは機能しません。

  ```
  source=accounts | parse address '\d+ (?<street>.+)' | parse street '\w+ (?<road>\w+)'
  ```

- 解析で定義されたフィールドを他のコマンドで上書きすることはできません。

  street は上書きできないため、where はどのドキュメントにも一致しません。

  ```
  source=accounts | parse address '\d+ (?<street>.+)' | eval street='1' | where
   street='1' ;
  ```

- 解析で使用されるテキストフィールドは上書きできません。

  address は上書きされるため、street は正常に解析されません。

  ```
  source=accounts | parse address '\d+ (?<street>.+)' | eval address='1' ;
  ```

- 解析で定義されたフィールドは、stats コマンドで使用した後にフィルタリングまたはソートすることはできません。

  次のコマンドの where は機能しません:

```
source=accounts | parse email '.+@(?<host>.+)' | stats avg(age) by host | where
 host=pyrami.com ;
```

patterns コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

patterns コマンドは、テキストフィールドからログパターンを抽出し、検索結果に結果を追加します。パターン別にログをグループ化すると、大量のログデータから統計情報を集計し、分析やトラブルシューティングを行うことが容易になります。

構文

次の構文を使用します。

```
patterns [new_field=<new-field-name>] [pattern=<pattern>] <field>
```

new-field-name

- オプション文字列
- これは、抽出されたパターンの新しいフィールドの名前です。
- デフォルトは patterns_field です。
- 名前が既に存在する場合、元のフィールドが置き換えられます。

pattern

- オプション文字列
- これは、テキストフィールドから除外すべき文字列の正規表現パターンです。
- 存在しない場合、デフォルトのパターンは英数字 ([a-zA-Z\d]) です。

field

- 必須。
- フィールドはテキストフィールドである必要があります。

例 1: 新しいフィールドを作成する

この例では、ドキュメントごとに email で抽出句読点を使用する方法を示します。null フィールドを解析すると、空の文字列が返されます。

PPL クエリ:

```
os> source=accounts | patterns email | fields email, patterns_field ;
fetched rows / total rows = 4/4
+----------------------+------------------+
| email                | patterns_field   |
|----------------------+------------------|
| jane_doe@example.com | @.               |
| john_doe@example.net | @.               |
| null                 |                  |
| juan_li@example.org  | @.               |
+----------------------+------------------+
```

例 2: ログパターンを抽出する

この例では、デフォルトのパターンを使用して raw ログフィールドから句読点を抽出する方法を示します。

PPL クエリ:

```
os> source=apache | patterns message | fields message, patterns_field ;
fetched rows / total rows = 4/4
+-------------------------------------------------------------------------------------------------------
+---------------------------------+
| message
                                       | patterns_field                 |
|-------------------------------------------------------------------------------------------------------
+---------------------------------|
| 177.95.8.74 - upton5450 [28/Sep/2022:10:15:57 -0700] "HEAD /e-business/mindshare
 HTTP/1.0" 404 19927                          | ... -  [//::: -] " /-/ /."      |
| ************ - pouros8756 [28/Sep/2022:10:15:57 -0700] "GET /architectures/
convergence/niches/mindshare HTTP/1.0" 100 28722 | ... -  [//::: -] " //// /."     |
```

```
| *************** - - [28/Sep/2022:10:15:57 -0700] "PATCH /strategize/out-of-the-box
 HTTP/1.0" 401 27439                      | ... - - [//::: -] " //--- /."   |
| ************** - - [28/Sep/2022:10:15:57 -0700] "POST /users HTTP/1.1" 301 9481
                                        | ... - - [//::: -] " / /."       |
+-----------------------------------------------------------------------------------------
+--------------------------------+
```

例 3: カスタム正規表現パターンを使用してログパターンを抽出する

この例では、ユーザー定義パターンを使用して raw ログフィールドから句読点を抽出する方法を示します。

PPL クエリ:

```
os> source=apache | patterns new_field='no_numbers' pattern='[0-9]' message | fields
 message, no_numbers ;
fetched rows / total rows = 4/4
+-----------------------------------------------------------------------------------------
+-------------------------------------------------------------------------------
+
| message
                                        | no_numbers
                                        |
|-----------------------------------------------------------------------------------------
+-------------------------------------------------------------------------------|
| 177.95.8.74 - upton5450 [28/Sep/2022:10:15:57 -0700] "HEAD /e-business/mindshare
 HTTP/1.0" 404 19927                         | ... - upton [/Sep/::: -] "HEAD /e-
business/mindshare HTTP/."                           |
| 127.45.152.6 - pouros8756 [28/Sep/2022:10:15:57 -0700] "GET /architectures/
convergence/niches/mindshare HTTP/1.0" 100 28722 | ... - pouros [/Sep/::: -] "GET /
architectures/convergence/niches/mindshare HTTP/."   |
| *************** - - [28/Sep/2022:10:15:57 -0700] "PATCH /strategize/out-of-the-box
 HTTP/1.0" 401 27439                      | ... - - [/Sep/::: -] "PATCH /strategize/
out-of-the-box HTTP/."                          |
| ************** - - [28/Sep/2022:10:15:57 -0700] "POST /users HTTP/1.1" 301 9481
                                        | ... - - [/Sep/::: -] "POST /users HTTP/."
                                        |
+-----------------------------------------------------------------------------------------
+-------------------------------------------------------------------------------
+
```

制限

patterns コマンドには、parse コマンドと同じ制限があります。

rare コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`rare` コマンドを使用して、フィールドリスト内のすべてのフィールドの値の最も一般的なタプルを見つけます。

> **Note**
>
> グループ別フィールドの値の個別のタプルごとに最大 10 個の結果が返されます。

構文

次の構文を使用します。

```
rare [N] <field-list> [by-clause] rare_approx [N] <field-list> [by-clause]
```

フィールドリスト

- 必須。
- フィールド名のカンマ区切りリスト。

句ごと

- オプション。
- 結果をグループ化する 1 つ以上のフィールド。

いいえ

- 返される結果の数。

- デフォルト: 10

rare_approx

- HyperLogLog++ アルゴリズムによる推定基数を使用した、希少な (n) フィールドの概算数。

例 1: フィールド内で最も出現頻度の低い値を検索する

この例では、すべてのアカウントの最も出現頻度の低い性別を検索します。

PPL クエリ:

```
os> source=accounts | rare gender;
os> source=accounts | rare_approx 10 gender;
os> source=accounts | rare_approx gender;
fetched rows / total rows = 2/2
+----------+
| gender   |
|----------|
| F        |
| M        |
+----------+
```

例 2: 性別ごとに整理された最も頻度の低い値を検索する

この例では、性別ごとにすべてのアカウントグループの最も出現頻度の低い年齢を抽出します。

PPL クエリ:

```
os> source=accounts | rare 5 age by gender;
os> source=accounts | rare_approx 5 age by gender;
fetched rows / total rows = 4/4
+----------+-------+
| gender   | age   |
|----------+-------|
| F        | 28    |
| M        | 32    |
| M        | 33    |
| M        | 36    |
+----------+-------+
```

rename コマンド

`rename` コマンドを使用して、検索結果の 1 つ以上のフィールドの名前を変更します。

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

構文

次の構文を使用します。

```
rename <source-field> AS <target-field>["," <source-field> AS <target-field>]...
```

source-field

- 必須。
- これは、名前を変更するフィールドの名前です。

target-field

- 必須。
- これは、変更後の名前です。

例 1: 1 つのフィールドの名前を変更する

この例では、1 つのフィールドの名前を変更する方法を示します。

PPL クエリ:

```
os> source=accounts | rename account_number as an | fields an;
fetched rows / total rows = 4/4
+------+
| an   |
|------|
| 1    |
| 6    |
```

```
| 13   |
| 18   |
+------+
```

例 2: 複数のフィールドの名前を変更する

この例では、複数のフィールドの名前を変更する方法を示します。

PPL クエリ:

```
os> source=accounts | rename account_number as an, employer as emp | fields an, emp;
fetched rows / total rows = 4/4
+------+---------+
| an   | emp     |
|------+---------|
| 1    | Pyrami  |
| 6    | Netagy  |
| 13   | Quility |
| 18   | null    |
+------+---------+
```

制限事項

- 既存のフィールドの上書きはサポートされていません:

```
source=accounts | grok address '%{NUMBER} %{GREEDYDATA:address}' | fields address
```

search コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

search コマンドを使用して、インデックスからドキュメントを取得します。search コマンドは、PPL クエリの最初のコマンドとしてのみ使用できます。

構文

次の構文を使用します。

```
search source=[<remote-cluster>:]<index> [boolean-expression]
```

検索

- オプション。
- 検索キーワード (省略可能)

index

- 必須。
- 検索コマンドは、クエリ元のインデックスを指定する必要があります。
- インデックス名には、クラスター間検索用に <cluster name>: をプレフィックスとして付けることができます。

ブール式

- オプション。
- ブール値に評価される任意の式

例 1: すべてのデータを取得する

この例では、アカウントインデックスからすべてのドキュメントを取得します。

PPL クエリ:

```
os> source=accounts;
+------------------+-------------+----------------------+-----------+----------
+--------+----------------+---------+-------+-----------------------+------------+
| account_number   | firstname   | address              | balance   | gender   | city
 | employer       | state   | age   | email                 | lastname   |
|------------------+-------------+----------------------+-----------+----------
+--------+----------------+---------+-------+-----------------------+------------|
| 1                | Jorge       | *** Any Lane         | 39225     | M        | Brogan
 | ExampleCorp    | IL      | 32    | jane_doe@example.com  | Souza      |
| 6                | John        | *** Example Street   | 5686      | M        | Dante
 | AnyCorp        | TN      | 36    | john_doe@example.com  | Doe        |
```

```
| 13                | Jane         | *** Any Street       | *****      | F         | Nogal
 | ExampleCompany | VA       | 28     | null                   | Doe         |
| 18                | Juan         | *** Example Court    | 4180       | M         | Orick
 | null           | MD       | 33     | juan_li@example.org    | Li          |
+-------------------+--------------+----------------------+------------+-----------
+--------+----------------+---------+-------+-----------------------+------------+
```

例 2: 条件付きでデータを取得する

この例では、条件付きでアカウントインデックスからすべてのドキュメントを取得する方法を示しています。

PPL クエリ:

```
os> SEARCH source=accounts account_number=1 or gender="F";
+-------------------+--------------+--------------------+-----------+-----------+--------
+----------------+---------+-------+-------------------------+------------+
| account_number    | firstname    | address            | balance    | gender    | city    |
 employer        | state    | age    | email                 -  | lastname    |
|-------------------+--------------+--------------------+-----------+-----------+--------
+----------------+---------+-------+-------------------------+------------|
| 1                 | Jorge        | *** Any Lane       | *****      | M         | Brogan |
 ExampleCorp     | IL       | 32     | jorge_souza@example.com | Souza       |
| 13                | Jane         | *** Any Street     | *****      | F         | Nogal  |
 ExampleCompany | VA       | 28     | null                    | Doe         |
+-------------------+--------------+--------------------+-----------+-----------+--------
+------------------+---------+-------+-------------------------+------------+
```

sort コマンド

`sort` コマンドを使用して、指定したフィールドで検索結果をソートします。

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

構文

次の構文を使用します。

```
sort <[+|-] sort-field>...
```

[+|-]

- オプション。
- プラス [+] は昇順を示し、NULL/MISSING 値が先頭に並びます。
- マイナス [-] は降順を示し、NULL/MISSING 値が末尾に並びます。
- デフォルト: 昇順で、NULL/MISSING 値を先頭に配置されます。

ソートフィールド

- 必須。
- ソートに使用されるフィールド。

例 1: 1 つのフィールドでソートする

この例では、年齢フィールドを使用してドキュメントを昇順でソートする方法を示しています。

PPL クエリ:

```
os> source=accounts | sort age | fields account_number, age;
fetched rows / total rows = 4/4
+------------------+-------+
| account_number   | age   |
|------------------+-------|
| 13               | 28    |
| 1                | 32    |
| 18               | 33    |
| 6                | 36    |
+------------------+-------+
```

例 2: 1 つのフィールドでソートし、すべての結果を返します

この例では、年齢フィールドを使用してドキュメントを昇順でソートする方法を示しています。

PPL クエリ:

```
os> source=accounts | sort age | fields account_number, age;
```

```
fetched rows / total rows = 4/4
+------------------+-------+
| account_number   | age   |
|------------------+-------|
| 13               | 28    |
| 1                | 32    |
| 18               | 33    |
| 6                | 36    |
+------------------+-------+
```

例 3: 1 つのフィールドで降順でソートする

この例では、年齢フィールドを使用してドキュメントを降順でソートする方法を示しています。

PPL クエリ:

```
os> source=accounts | sort - age | fields account_number, age;
fetched rows / total rows = 4/4
+------------------+-------+
| account_number   | age   |
|------------------+-------|
| 6                | 36    |
| 18               | 33    |
| 1                | 32    |
| 13               | 28    |
+------------------+-------+
```

例 4: 複数のフィールドでソートする

この例では、性別フィールドを昇順に、年齢フィールドを降順にしてドキュメントをソートする方法を示しています。

PPL クエリ:

```
os> source=accounts | sort + gender, - age | fields account_number, gender, age;
fetched rows / total rows = 4/4
+------------------+----------+-------+
| account_number   | gender   | age   |
|------------------+----------+-------|
| 13               | F        | 28    |
| 6                | M        | 36    |
| 18               | M        | 33    |
```

```
| 1                 | M          | 32     |
+-------------------+------------+--------+
```

例 5: null を含むフィールドでソートする

この例では、デフォルトのオプション (昇順かつ NULL 値を最初に) で雇用主フィールドを並べ替える方法を示しています。結果は、null 値が最初の行にあることを示しています。

PPL クエリ:

```
os> source=accounts | sort employer | fields employer;
fetched rows / total rows = 4/4
+------------+
| employer   |
|------------|
| null       |
| AnyCompany |
| AnyCorp    |
| AnyOrgty   |
+------------+
```

stats コマンド

`stats` コマンドを使用して、検索結果から集計を計算します。

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

NULL/MISSING 値の処理

NULL/MISSING 値の処理

関数	NULL	MISSING
COUNT	カウントされません	カウントされません
SUM	Ignore (無視)	Ignore (無視)

関数	NULL	MISSING
AVG	Ignore (無視)	Ignore (無視)
MAX	Ignore (無視)	Ignore (無視)
MIN	Ignore (無視)	Ignore (無視)

構文

次の構文を使用します。

```
stats <aggregation>... [by-clause]
```

集計

- 必須。
- フィールドに適用される集計関数。

句ごと

- オプション。
- 構文: by [span-expression,] [field,]...
- 集計結果をグループ化するためのフィールドと式を指定します。by-clause を使用すると、フィールドと式を使用して集計結果をグループ化できます。スカラー関数、集計関数、スパン式を使用して、特定のフィールドを同じ間隔のバケットに分割することもできます。
- デフォルト: <by-clause> が指定されていない場合、stats コマンドは結果セット全体の集計を表す単一の行を返します。

span-expression

- オプション、最大 1 つまで。
- 構文: span(field_expr, interval_expr)
- 間隔式の単位は、デフォルトでは自然単位です。フィールドが日付と時刻タイプのフィールドで、間隔が日付/時刻の単位である場合は、間隔式の中で単位を指定します。

- 例えば、age フィールドを 10 年でバケットに分割すると、span(age, 10) のようになります。タイムスタンプフィールドを時間単位の間隔に分割するには、span(timestamp, 1h) を使用します。

使用可能な時間単位

スパン間隔の単位
ミリ秒 (ms)
秒 (秒)
分 (m、大文字と小文字を区別する)
時間 (h)
日 (d)
週 (w)
月 (M、大文字と小文字を区別する)
四半期 (q)
年 (y)

集計関数

COUNT

SELECT ステートメントによって取得された行の expr の数をカウントして返します。

例:

```
os> source=accounts | stats count();
fetched rows / total rows = 1/1
+-----------+
| count()   |
|-----------|
| 4         |
```

```
+-----------+
```

SUM

SUM(expr) を使用して expr の合計を返します。

例

```
os> source=accounts | stats sum(age) by gender;
fetched rows / total rows = 2/2
+------------+----------+
| sum(age)   | gender   |
|------------+----------|
| 28         | F        |
| 101        | M        |
+------------+----------+
```

AVG

AVG(expr) を使用して expr の平均値を返します。

例

```
os> source=accounts | stats avg(age) by gender;
fetched rows / total rows = 2/2
+--------------------+----------+
| avg(age)           | gender   |
|--------------------+----------|
| 28.0               | F        |
| 33.666666666666664 | M        |
+--------------------+----------+
```

MAX

MAX(expr) を使用して expr の最大値を返します。

例

```
os> source=accounts | stats max(age);
fetched rows / total rows = 1/1
+------------+
| max(age)   |
|------------|
```

```
| 36         |
+------------+
```

MIN

MIN(expr) を使用して expr の最小値を返します。

例

```
os> source=accounts | stats min(age);
fetched rows / total rows = 1/1
+------------+
| min(age)   |
|------------|
| 28         |
+------------+
```

STDDEV_SAMP

STDDEV_SAMP(expr) を使用して expr のサンプル標準偏差を返します。

例:

```
os> source=accounts | stats stddev_samp(age);
fetched rows / total rows = 1/1
+--------------------+
| stddev_samp(age)   |
|--------------------|
| 3.304037933599835  |
+--------------------+
```

STDDEV_POP

STDDEV_POP(expr) を使用して expr の母集団標準偏差を返します。

例:

```
os> source=accounts | stats stddev_pop(age);
fetched rows / total rows = 1/1
+--------------------+
| stddev_pop(age)    |
|--------------------|
| 2.**************** |
```

```
+--------------------+
```

TAKE

TAKE(field [, size]) を使用してフィールドの元の値を返します。値の順序は保証されません。

field

- 必須。
- フィールドはテキストフィールドである必要があります。

サイズ

- オプション整数
- 値の数を返す必要があります。
- デフォルトは 10 です。

例

```
os> source=accounts | stats take(firstname);
fetched rows / total rows = 1/1
+-----------------------------+
| take(firstname)             |
|-----------------------------|
| [Jane, Mary, Nikki, Juan    |
+-----------------------------+
```

PERCENTILE または PERCENTILE_APPROX

PERCENTILE(expr, percent) または PERCENTILE_APPROX(expr, percent)を使用して、expr の近似パーセンタイル値を指定されたパーセンテージで返します。

パーセント

- 数値は 0 ~ 100 の定数である必要があります。

例

```
os> source=accounts | stats percentile(age, 90) by gender;
fetched rows / total rows = 2/2
+----------------------+----------+
| percentile(age, 90)   | gender   |
|----------------------+----------|
| 28                    | F        |
| 36                    | M        |
+----------------------+----------+
```

例 1: イベントの数を計算する

この例では、アカウントのイベント数を計算する方法を示します。

```
os> source=accounts | stats count();
fetched rows / total rows = 1/1
+-----------+
| count()   |
|-----------|
| 4         |
+-----------+
```

例 2: フィールドの平均を計算する

この例では、すべてのアカウントの平均年齢を計算する方法を示しています。

```
os> source=accounts | stats avg(age);
fetched rows / total rows = 1/1
+------------+
| avg(age)   |
|------------|
| 32.25      |
+------------+
```

例 3: フィールドの平均をグループごとに計算する

この例では、性別ごとにグループ化された、すべてのアカウントの平均年齢を計算する方法を示しています。

```
os> source=accounts | stats avg(age) by gender;
fetched rows / total rows = 2/2
+--------------------+----------+
| avg(age)           | gender   |
```

```
|--------------------+----------|
| 28.0               | F        |
| 33.666666666666664 | M        |
+--------------------+----------+
```

例 4: フィールドの平均、合計、数をグループごとに計算する

この例では、すべてのアカウントの平均年齢、合計年齢、イベント数を性別ごとにグループ化して計算する方法を示しています。

```
os> source=accounts | stats avg(age), sum(age), count() by gender;
fetched rows / total rows = 2/2
+--------------------+------------+-----------+----------+
| avg(age)           | sum(age)   | count()   | gender   |
|--------------------+------------+-----------+----------|
| 28.0               | 28         | 1         | F        |
| 33.666666666666664 | 101        | 3         | M        |
+--------------------+------------+-----------+----------+
```

例 5: フィールドの最大数を計算する

この例では、すべてのアカウントの最大有効期間を計算します。

```
os> source=accounts | stats max(age);
fetched rows / total rows = 1/1
+------------+
| max(age)   |
|------------|
| 36         |
+------------+
```

例 6: フィールドの最大数と最小数をグループごとに計算する

この例では、すべてのアカウントの年齢の最大値と最小値を性別ごとにグループ化して計算します。

```
os> source=accounts | stats max(age), min(age) by gender;
fetched rows / total rows = 2/2
+------------+------------+----------+
| max(age)   | min(age)   | gender   |
|------------+------------+----------|
| 28         | 28         | F        |
| 36         | 32         | M        |
```

```
+------------+-----------+----------+
```

例 7: フィールドの個別の数を計算する

フィールドの個別の値の数を取得するには、COUNT の代わりに DISTINCT_COUNT (または DC) 関数を使用できます。この例では、すべてのアカウントの性別フィールドの数と個別の数の両方を計算します。

```
os> source=accounts | stats count(gender), distinct_count(gender);
fetched rows / total rows = 1/1
+-----------------+--------------------------+
| count(gender)   | distinct_count(gender)   |
|-----------------+--------------------------|
| 4               | 2                        |
+-----------------+--------------------------+
```

例 8: スパンごとにカウントを計算する

この例では、年齢を 10 年間隔でカウントして集計します。

```
os> source=accounts | stats count(age) by span(age, 10) as age_span
fetched rows / total rows = 2/2
+--------------+------------+
| count(age)   | age_span   |
|--------------+------------|
| 1            | 20         |
| 3            | 30         |
+--------------+------------+
```

例 9: 性別とスパンごとにカウントを計算する

この例では、性別と 5 年の年齢スパン別にグループ化されたレコードをカウントします。

```
os> source=accounts | stats count() as cnt by span(age, 5) as age_span, gender
fetched rows / total rows = 3/3
+-------+------------+----------+
| cnt   | age_span   | gender   |
|-------+------------+----------|
| 1     | 25         | F        |
| 2     | 30         | M        |
| 1     | 35         | M        |
+-------+------------+----------+
```

コマンドで指定された順序に関係なく、スパン式は常に最初のグループ化キーとして表示されます。

```
os> source=accounts | stats count() as cnt by gender, span(age, 5) as age_span
fetched rows / total rows = 3/3
+-------+------------+----------+
| cnt   | age_span   | gender   |
|-------+------------+----------|
| 1     | 25         | F        |
| 2     | 30         | M        |
| 1     | 35         | M        |
+-------+------------+----------+
```

例 10: 性別とスパンでカウントを計算し、E メールリストを取得する

この例では、年齢の数を 10 年間隔で取得し、性別ごとにグループ化します。さらに、行ごとに最大 5 通の E メールのリストを取得します。

```
os> source=accounts | stats count() as cnt, take(email, 5) by span(age, 5) as age_span,
 gender
fetched rows / total rows = 3/3
+-------+-----------------------------------------------------+------------+----------+
| cnt   | take(email, 5)                                      | age_span   | gender   |
|-------+-----------------------------------------------------+------------+----------|
| 1     | []                                                  | 25         | F        |
| 2     | [janedoe@anycompany.com,juanli@examplecompany.org] | 30         | M        |
| 1     | [marymajor@examplecorp.com]                         | 35         | M        |
+-------+-----------------------------------------------------+------------+----------+
```

例 11: フィールドのパーセンタイルを計算する

この例では、すべてのアカウントの 90 番目の年齢のパーセンタイルを計算する方法を示しています。

```
os> source=accounts | stats percentile(age, 90);
fetched rows / total rows = 1/1
+-----------------------+
| percentile(age, 90)   |
|-----------------------|
| 36                    |
+-----------------------+
```

例 12: フィールドのパーセンタイルをグループ別に計算する

この例では、すべてのアカウントグループのパーセンタイル 90 番目の年齢を性別ごとに計算する方法を示しています。

```
os> source=accounts | stats percentile(age, 90) by gender;
fetched rows / total rows = 2/2
+-----------------------+----------+
| percentile(age, 90)   | gender   |
|-----------------------+----------|
| 28                    | F        |
| 36                    | M        |
+-----------------------+----------+
```

例 13: 性別とスパンごとにパーセンタイルを計算する

この例では、この例では、年齢を 10 年ごとの間隔で区切り、性別ごとにパーセンタイル 90 番目の年齢を取得します。

```
os> source=accounts | stats percentile(age, 90) as p90 by span(age, 10) as age_span,
 gender
fetched rows / total rows = 2/2
+-------+------------+----------+
| p90   | age_span   | gender   |
|-------+------------+----------|
| 28    | 20         | F        |
| 36    | 30         | M        |
+-------+------------+----------+
```

```
- `source = table | stats avg(a) `
- `source = table | where a < 50 | stats avg(c) `
- `source = table | stats max(c) by b`
- `source = table | stats count(c) by b | head 5`
- `source = table | stats distinct_count(c)`
- `source = table | stats stddev_samp(c)`
- `source = table | stats stddev_pop(c)`
- `source = table | stats percentile(c, 90)`
- `source = table | stats percentile_approx(c, 99)`
```

スパンによる集計

```
- `source = table  | stats count(a) by span(a, 10) as a_span`
- `source = table  | stats sum(age) by span(age, 5) as age_span | head 2`
- `source = table  | stats avg(age) by span(age, 20) as age_span, country  | sort -
 age_span |  head 2`
```

タイムウィンドウスパンの集計 (タンブルウィンドウ関数)

```
- `source = table | stats sum(productsAmount) by span(transactionDate, 1d) as age_date
 | sort age_date`
- `source = table | stats sum(productsAmount) by span(transactionDate, 1w) as age_date,
 productId`
```

複数のレベルによる集計グループ

```
- `source = table | stats avg(age) as avg_state_age by country, state | stats
 avg(avg_state_age) as avg_country_age by country`
- `source = table | stats avg(age) as avg_city_age by country, state, city | eval
 new_avg_city_age = avg_city_age - 1 | stats avg(new_avg_city_age) as avg_state_age
 by country, state | where avg_state_age > 18 | stats avg(avg_state_age) as
 avg_adult_country_age by country`
```

subquery コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

subquery コマンドを使用して、Piped Processing Language (PPL) ステートメント内で複雑なネストされたクエリを実行します。

```
source=logs | where field in [ subquery source=events | where condition | fields
 field ]
```

この例では、プライマリ検索 (source=logs) はサブクエリ (source=events) の結果によってフィルタリングされます。

サブクエリコマンドは、複雑なデータ分析のために複数レベルのネストをサポートします。

ネスト化したサブクエリの例

```
source=logs | where id in [ subquery source=users | where user in [ subquery
 source=actions | where action="login" | fields user] | fields uid ]
```

InSubquery の使用

- `source = outer | where a in [ source = inner | fields b ]`
- `source = outer | where (a) in [ source = inner | fields b ]`
- `source = outer | where (a,b,c) in [ source = inner | fields d,e,f ]`
- `source = outer | where a not in [ source = inner | fields b ]`
- `source = outer | where (a) not in [ source = inner | fields b ]`
- `source = outer | where (a,b,c) not in [ source = inner | fields d,e,f ]`
- `source = outer a in [ source = inner | fields b ]` (サブクエリによる検索フィルタリング)
- `source = outer a not in [ source = inner | fields b ]` (サブクエリによる検索フィルタリング)
- `source = outer | where a in [ source = inner1 | where b not in [ source = inner2 | fields c ] | fields b ]` (ネスト化済み)
- `source = table1 | inner join left = l right = r on l.a = r.a AND r.a in [ source = inner | fields d ] | fields l.a, r.a, b, c` (結合フィルターとして)

IN-Subquery PPL を使用した SQL 移行の例

TPC-H Q4 (集計を含むサブクエリ内)

```
select
  o_orderpriority,
  count(*) as order_count
from
  orders
where
  o_orderdate >= date '1993-07-01'
  and o_orderdate < date '1993-07-01' + interval '3' month
```

```
  and o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    where l_commitdate < l_receiptdate
  )
group by
  o_orderpriority
order by
  o_orderpriority
```

PPL InSubquery クエリによって書き換えられました:

```
source = orders
| where o_orderdate >= "1993-07-01" and o_orderdate < "1993-10-01" and o_orderkey IN
  [ source = lineitem
    | where l_commitdate < l_receiptdate
    | fields l_orderkey
  ]
| stats count(1) as order_count by o_orderpriority
| sort o_orderpriority
| fields o_orderpriority, order_count
```

TPC-H Q20 (サブクエリでネスト化)

```
select
  s_name,
  s_address
from
  supplier,
  nation
where
  s_suppkey in (
    select
      ps_suppkey
    from
      partsupp
    where
      ps_partkey in (
        select
          p_partkey
        from
```

```
          part
        where
          p_name like 'forest%'
      )
  )
  and s_nationkey = n_nationkey
  and n_name = 'CANADA'
order by
  s_name
```

PPL InSubquery クエリによって書き換えられました:

```
source = supplier
| where s_suppkey IN [
    source = partsupp
    | where ps_partkey IN [
        source = part
        | where like(p_name, "forest%")
        | fields p_partkey
      ]
    | fields ps_suppkey
  ]
| inner join left=l right=r on s_nationkey = n_nationkey and n_name = 'CANADA'
  nation
| sort s_name
```

ExistsSubquery の使用

前提: a、b はテーブル outer のフィールド、c、d はテーブル inner のフィールド、e、f はテーブル inner2 のフィールドです。

- `source = outer | where exists [ source = inner | where a = c ]`
- `source = outer | where not exists [ source = inner | where a = c ]`
- `source = outer | where exists [ source = inner | where a = c and b = d ]`
- `source = outer | where not exists [ source = inner | where a = c and b = d ]`
- `source = outer exists [ source = inner | where a = c ]` (サブクエリによる検索フィルタリング)
- `source = outer not exists [ source = inner | where a = c ]` (サブクエリによる検索フィルタリング)

- `source = table as t1 exists [ source = table as t2 | where t1.a = t2.a ]` (テーブルエイリアスは、存在するサブクエリで有用です)
- `source = outer | where exists [ source = inner1 | where a = c and exists [ source = inner2 | where c = e ] ]` (ネスト化済み)
- `source = outer | where exists [ source = inner1 | where a = c | where exists [ source = inner2 | where c = e ] ]` (ネスト化済み)
- `source = outer | where exists [ source = inner | where c > 10 ]` (非相関が存在する)
- `source = outer | where not exists [ source = inner | where c > 10 ]` (非相関が存在する)
- `source = outer | where exists [ source = inner ] | eval l = "nonEmpty" | fields l` (特別な非相関が存在する)

ScalarSubquery の使用

前提: a、b はテーブル outer のフィールド、c、d はテーブル inner のフィールド、e、f はテーブル inner2 のフィールドです。

非相関スカラーサブクエリ

Select 内:

- `source = outer | eval m = [ source = inner | stats max(c) ] | fields m, a`
- `source = outer | eval m = [ source = inner | stats max(c) ] + b | fields m, a`

Where 内:

- `source = outer | where a > [ source = inner | stats min(c) ] | fields a`

検索フィルター内:

- `source = outer a > [ source = inner | stats min(c) ] | fields a`

相関スカラーサブクエリ

Select 内:

- `source = outer | eval m = [ source = inner | where outer.b = inner.d | stats max(c) ] | fields m, a`
- `source = outer | eval m = [ source = inner | where b = d | stats max(c) ] | fields m, a`
- `source = outer | eval m = [ source = inner | where outer.b > inner.d | stats max(c) ] | fields m, a`

Where 内:

- `source = outer | where a = [ source = inner | where outer.b = inner.d | stats max(c) ]`
- `source = outer | where a = [ source = inner | where b = d | stats max(c) ]`
- `source = outer | where [ source = inner | where outer.b = inner.d OR inner.d = 1 | stats count() ] > 0 | fields a`

検索フィルター内:

- `source = outer a = [ source = inner | where b = d | stats max(c) ]`
- `source = outer [ source = inner | where outer.b = inner.d OR inner.d = 1 | stats count() ] > 0 | fields a`

ネスト化スカラーサブクエリ

- `source = outer | where a = [ source = inner | stats max(c) | sort c ] OR b = [ source = inner | where c = 1 | stats min(d) | sort d ]`
- `source = outer | where a = [ source = inner | where c = [ source = nested | stats max(e) by f | sort f ] | stats max(d) by c | sort c | head 1 ]`

(関係) サブクエリ

InSubquery、ExistsSubquery、ScalarSubquery はすべてサブクエリ式です。ただし、RelationSubquery はサブクエリ式ではなく、Join 句または From 句でよく使用されるサブクエリプランです。

- `source = table1 | join left = l right = r [ source = table2 | where d > 10 | head 5 ]` (右結合側のサブクエリ)
- `source = [ source = table1 | join left = l right = r [ source = table2 | where d > 10 | head 5 ] | stats count(a) by b ] as outer | head 1`

追加のコンテキスト

InSubquery、ExistsSubquery、ScalarSubquery は、where 句と検索フィルターで一般的に使用されるサブクエリ式です。

Where コマンド:

```
| where <boolean expression> | ...
```

検索フィルター:

```
search source=* <boolean expression> | ...
```

サブクエリ式は、ブール式で使用できます。

```
| where orders.order_id in [ source=returns | where return_reason="damaged" | field
 order_id ]
```

`orders.order_id in [ source=... ]` は `<boolean expression>` です。

一般的に、この種のサブクエリ句を InSubquery 式と呼びます。これは `<boolean expression>` です。

異なる結合タイプのサブクエリ

ScalarSubquery を使用した例:

```
source=employees
| join source=sales on employees.employee_id = sales.employee_id
| where sales.sale_amount > [ source=targets | where target_met="true" | fields
 target_value ]
```

InSubquery、ExistsSubquery、ScalarSubquery とは異なり、RelationSubquery はサブクエリ式ではありません。代わりに、これはサブクエリプランです。

```
SEARCH source=customer
| FIELDS c_custkey
| LEFT OUTER JOIN left = c, right = o ON c.c_custkey = o.o_custkey
   [
      SEARCH source=orders
      | WHERE o_comment NOT LIKE '%unusual%packages%'
      | FIELDS o_orderkey, o_custkey
   ]
| STATS ...
```

top コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

top コマンドを使用して、フィールドリスト内のすべてのフィールドの最も一般的な値のタプルを見つけます。

構文

次の構文を使用します。

```
top [N] <field-list> [by-clause] top_approx [N] <field-list> [by-clause]
```

いいえ

- 返される結果の数。
- デフォルト: 10

フィールドリスト

- 必須。
- フィールド名のカンマ区切りリスト。

句ごと

- オプション。
- 結果をグループ化する 1 つ以上のフィールド。

top_approx

- HyperLogLog++ アルゴリズムによる推定基数を使用した (n) 上位フィールドの概算数。

例 1: フィールドで最も出現頻度の高い値を検索する

この例では、すべてのアカウントで最も出現頻度の高い性別を検索します。

PPL クエリ:

```
os> source=accounts | top gender;
os> source=accounts | top_approx gender;
fetched rows / total rows = 2/2
+----------+
| gender   |
|----------|
| M        |
| F        |
+----------+
```

例 2: フィールドで最も出現頻度の高い値を検索する (1 に制限)

この例では、すべてのアカウントで最も出現頻度の高い性別を 1 つ検索します。

PPL クエリ:

```
os> source=accounts | top_approx 1 gender;
fetched rows / total rows = 1/1
+----------+
| gender   |
|----------|
| M        |
+----------+
```

例 3: 性別ごとにグループ化された最も頻度の高い値を検索する

この例では、性別ごとにグループ化された、すべてのアカウントの最も頻度の高い年齢を検索します。

PPL クエリ:

```
os> source=accounts | top 1 age by gender;
os> source=accounts | top_approx 1 age by gender;
fetched rows / total rows = 2/2
+----------+-------+
| gender   | age   |
|----------+-------|
| F        | 28    |
| M        | 32    |
+----------+-------+
```

trendline コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`trendline` コマンドを使用して、フィールドの移動平均を計算します。

構文

次の構文を使用します

```
TRENDLINE [sort <[+|-] sort-field>] SMA(number-of-datapoints, field) [AS alias]
 [SMA(number-of-datapoints, field) [AS alias]]...
```

[+|-]

- オプション。
- プラス [+] は昇順を示し、NULL/MISSING 値が先頭に並びます。
- マイナス [-] は降順を示し、NULL/MISSING 値が末尾に並びます。
- デフォルト: 昇順で、NULL/MISSING 値を先頭に配置されます。

ソートフィールド

- ソートを使用する場合は必須です。
- ソートに使用されるフィールド。

number-of-datapoints

- 必須。
- 移動平均を計算するデータポイントの数。
- ゼロより大きい値を指定する必要があります。

field

- 必須。
- 移動平均を計算するフィールドの名前。

alias

- オプション。
- 移動平均を含む結果の列の名前。

単純移動平均 (SMA) タイプのみがサポートされています。これは次のように計算されます:

```
f[i]: The value of field 'f' in the i-th data-point
n: The number of data-points in the moving window (period)
t: The current time index

SMA(t) = (1/n) * Σ(f[i]), where i = t-n+1 to t
```

例 1: 気温の時系列データに対する単純移動平均を計算する

この例では、2 つのデータポイントを使用して、気温に対する単純移動平均を計算します。

PPL クエリ:

```
os> source=t | trendline sma(2, temperature) as temp_trend;
fetched rows / total rows = 5/5
+-----------+---------+--------------------+----------+
```

```
|temperature|device-id|           timestamp|temp_trend|
+-----------+---------+--------------------+----------+
|         12|     1492|2023-04-06 17:07:...|      NULL|
|         12|     1492|2023-04-06 17:07:...|      12.0|
|         13|      256|2023-04-06 17:07:...|      12.5|
|         14|      257|2023-04-06 17:07:...|      13.5|
|         15|      258|2023-04-06 17:07:...|      14.5|
+-----------+---------+--------------------+----------+
```

例 2: 温度の時系列データに対してソートを施した単純移動平均を計算する

この例では、device-id で降順でソートされた 2 つと 3 つのデータポイントを使用して、温度に対する 2 つの単純移動平均を計算します。

PPL クエリ:

```
os> source=t | trendline sort - device-id sma(2, temperature) as temp_trend_2 sma(3,
 temperature) as temp_trend_3;
fetched rows / total rows = 5/5
+-----------+---------+--------------------+------------+------------------+
|temperature|device-id|           timestamp|temp_trend_2|      temp_trend_3|
+-----------+---------+--------------------+------------+------------------+
|         15|      258|2023-04-06 17:07:...|        NULL|              NULL|
|         14|      257|2023-04-06 17:07:...|        14.5|              NULL|
|         13|      256|2023-04-06 17:07:...|        13.5|              14.0|
|         12|     1492|2023-04-06 17:07:...|        12.5|              13.0|
|         12|     1492|2023-04-06 17:07:...|        12.0|12.333333333333334|
+-----------+---------+--------------------+------------+------------------+
```

where コマンド

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`where` コマンドはブール式を使用して検索結果をフィルタリングします。ブール式が true と評価された場合にのみ結果を返します。

構文

次の構文を使用します。

```
where <boolean-expression>
```

ブール式

- オプション。
- ブール値に評価できる任意の式。

例 1: 条件付きで結果セットをフィルタリングする

この例では、特定の条件を満たすアカウントインデックスからドキュメントを取得する方法を示します。

PPL クエリ:

```
os> source=accounts | where account_number=1 or gender="F" | fields account_number,
 gender;
fetched rows / total rows = 2/2
+------------------+----------+
| account_number   | gender   |
|------------------+----------|
| 1                | M        |
| 13               | F        |
+------------------+----------+
```

その他の例

論理条件を使用したフィルター

- `source = table | where c = 'test' AND a = 1 | fields a,b,c`
- `source = table | where c != 'test' OR a > 1 | fields a,b,c | head 1`
- `source = table | where c = 'test' NOT a > 1 | fields a,b,c`
- `source = table | where a = 1 | fields a,b,c`
- `source = table | where a >= 1 | fields a,b,c`
- `source = table | where a < 1 | fields a,b,c`

- source = table | where b != 'test' | fields a,b,c
- source = table | where c = 'test' | fields a,b,c | head 3
- source = table | where ispresent(b)
- source = table | where isnull(coalesce(a, b)) | fields a,b,c | head 3
- source = table | where isempty(a)
- source = table | where isblank(a)
- source = table | where case(length(a) > 6, 'True' else 'False') = 'True'
- source = table | where a between 1 and 4 – 注意: これは a >= 1 と a <= 4 を返します。つまり範囲は [1, 4] です。
- source = table | where b not between '2024-09-10' and '2025-09-10' – 注: これは b >= '**********' と b <= '2025-09-10' を返します。
- source = table | where cidrmatch(ip, '***********/24')
- source = table | where cidrmatch(ipv6, '2003:db8::/32')

```
source = table | eval status_category =
    case(a >= 200 AND a < 300, 'Success',
    a >= 300 AND a < 400, 'Redirection',
    a >= 400 AND a < 500, 'Client Error',
    a >= 500, 'Server Error'
    else 'Incorrect HTTP status code')
    | where case(a >= 200 AND a < 300, 'Success',
    a >= 300 AND a < 400, 'Redirection',
    a >= 400 AND a < 500, 'Client Error',
    a >= 500, 'Server Error'
    else 'Incorrect HTTP status code'
    ) = 'Incorrect HTTP status code'
```

```
source = table
    | eval factor = case(a > 15, a - 14, isnull(b), a - 7, a < 3, a + 1 else 1)
    | where case(factor = 2, 'even', factor = 4, 'even', factor = 6, 'even', factor =
 8, 'even' else 'odd') = 'even'
    |  stats count() by factor
```

フィールドの概要

> **Note**
>
> この PPL コマンドをサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “コマンド”。

`fieldsummary` コマンドを使用して、各フィールドの基本統計 (カウント、個別カウント、最小、最大、平均、標準偏差、平均) を計算し、各フィールドのデータ型を決定します。このコマンドは、その直前に置かれた任意のパイプと併用でき、それらを考慮に入れます。

構文

以下の構文を使用します。CloudWatch Logs のユースケースでは、クエリ内の 1 つのフィールドのみがサポートされています。

```
... | fieldsummary <field-list> (nulls=true/false)
```

includefields

- 統計を使用して収集されるすべての列のリストを統一された結果セットにまとめます。

Null

- オプション。
- true に設定する場合は、集計計算に null 値を含めます (数値の場合は null を 0 に置き換えます)。

例 1

PPL クエリ:

```
os> source = t | where status_code != 200 | fieldsummary includefields= status_code
 nulls=true
+------------------+-------------+------------+------------+------------+------------
+------------+------------+----------------|
| Fields           | COUNT       | COUNT_DISTINCT    |  MIN  |  MAX   |  AVG   |  MEAN
  |        STDDEV       | NUlls | TYPEOF |
```

```
|-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
| "status_code"    |      2      |         2         | 301   |   403   |  352.0 |  352.0
  |  72.12489168102785  |  0    | "int"  |
+-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
```

例 2

PPL クエリ:

```
os> source = t | fieldsummary includefields= id, status_code, request_path nulls=true
+-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
| Fields           | COUNT       | COUNT_DISTINCT    |  MIN  |  MAX   |  AVG   |  MEAN
  |        STDDEV       | NUlls | TYPEOF |
|-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
|       "id"       |      6      |         6         | 1     |   6    |  3.5   |   3.5
 |  1.8708286933869707  |  0    | "int"  |
+-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
| "status_code"    |      4      |         3         | 200   |   403   |  184.0 |  184.0
  |  161.16699413961905 |  2    | "int"  |
+-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
| "request_path"   |      2      |         2         | /about| /home   |  0.0    |  0.0
    |      0            |  2    |"string"|
+-----------------+-------------+-----------+-----------+-----------+-----------
+-----------+-----------+----------------|
```

拡張コマンド

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

expand コマンドを使用して Array<Any> または Map<Any> タイプのフィールドをフラット化し、要素またはキーと値のペアごとに個別の行を生成します。

構文

次の構文を使用します。

```
expand <field> [As alias]
```

field

- 拡張 (展開) するフィールド。
- フィールドは、サポート対象のタイプである必要があります。

alias

- オプション。
- 元のフィールド名の代わりに使用される名前。

使用上のガイドライン

拡張コマンドは、指定された配列またはマップフィールド内の各要素の行を生成します。ここでは、

- 配列要素は個々の行になります。
- マップキーと値のペアは別々の行に分割され、各キーと値は行として表されます。
- エイリアスを指定すると、展開された値は元のフィールド名ではなくエイリアスで表されます。

このコマンドを統計、評価、解析などの他のコマンドと組み合わせて使用して、拡張後のデータを操作または抽出できます。

例

- `source = table | expand employee | stats max(salary) as max by state, company`
- `source = table | expand employee as worker | stats max(salary) as max by state, company`
- `source = table | expand employee as worker | eval bonus = salary * 3 | fields worker, bonus`
- `source = table | expand employee | parse description '(?<email>.+@.+)' | fields employee, email`

- `source = table | eval array=json_array(1, 2, 3) | expand array as uid | fields name, occupation, uid`
- `source = table | expand multi_valueA as multiA | expand multi_valueB as multiB`

expand コマンドは、評価、統計などの他のコマンドと組み合わせて使用できます。複数の expand コマンドを使用すると、各複合配列またはマップ内のすべての内部要素の直積集合が作成されます。

有効な SQL プッシュダウンクエリ

expand コマンドは、LATERAL VIEW の爆発を使用して同等の SQL オペレーションに変換されるため、SQL クエリレベルで配列またはマップを効率的に展開させることができます。

```
SELECT customer exploded_productId
FROM table
LATERAL VIEW explode(productId) AS exploded_productId
```

explode コマンドは、次の機能を提供します:

- これは、新しい列を返す列操作です。
- 展開した列のすべての要素に新しい行が作成されます。
- 内部 null は、展開したフィールドの一部として無視されます (null の場合は作成/展開される行はありません)。

PPL 関数

トピック

- PPL 条件関数
- PPL 暗号化ハッシュ関数
- PPL 日付および時刻関数
- PPL 式
- PPL IP アドレス関数
- PPL JSON 関数
- PPL Lambda 関数
- PPL 数学関数

- PPL 文字列関数
- PPL 型変換関数

PPL 条件関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

ISNULL

説明: フィールドが null の場合、`isnull(field)` は true を返します。

引数の型:

- サポートされているすべてのデータ型:

戻り値の型:

- BOOLEAN

例:

```
os> source=accounts | eval result = isnull(employer) | fields result, employer,
 firstname
fetched rows / total rows = 4/4
+----------+-------------+-------------+
| result   | employer    | firstname   |
|----------+-------------+-------------|
| False    | AnyCompany  | Mary        |
| False    | ExampleCorp | Jane        |
| False    | ExampleOrg  | Nikki       |
| True     | null        | Juan        |
+----------+-------------+-------------+
```

ISNOTNULL

説明: フィールドが null でない場合、`isnotnull(field)` は true を返します。

引数の型:

- サポートされているすべてのデータ型:

戻り値の型:

- BOOLEAN

例:

```
os> source=accounts | where not isnotnull(employer) | fields account_number, employer
fetched rows / total rows = 1/1
+------------------+------------+
| account_number   | employer   |
|------------------+------------|
| 18               | null       |
+------------------+------------+
```

EXISTS

例:

```
os> source=accounts | where exists(email) | fields account_number, email
fetched rows / total rows = 1/1
```

IFNULL

説明: `field1` が null `field2` の場合、`ifnull(field1, field2)` はを を返します。

引数の型:

- サポートされているすべてのデータ型:
- 2 つのパラメータのタイプが異なる場合、関数はセマンティックチェックに失敗します。

戻り値の型:

- いずれか

例:

```
os> source=accounts | eval result = ifnull(employer, 'default') | fields result,
 employer, firstname
fetched rows / total rows = 4/4
+------------+------------+-------------+
| result     | employer   | firstname   |
|------------+------------+-------------|
| AnyCompany | AnyCompany | Mary        |
| ExampleCorp| ExampleCorp| Jane        |
| ExampleOrg | ExampleOrg | Nikki       |
| default    | null       | Juan        |
+------------+------------+-------------+
```

NULLIF

説明: `nullif(field1, field2)` は 2 つのパラメータが同じ場合は null を返し、それ以外の場合は field1 を返します。

引数の型:

- サポートされているすべてのデータ型:
- 2 つのパラメータのタイプが異なる場合、関数はセマンティックチェックに失敗します。

戻り値の型:

- いずれか

例:

```
os> source=accounts | eval result = nullif(employer, 'AnyCompany') | fields result,
 employer, firstname
fetched rows / total rows = 4/4
+----------------+----------------+-------------+
| result         | employer       | firstname   |
|----------------+----------------+-------------|
| null           | AnyCompany     | Mary        |
| ExampleCorp    | ExampleCorp    | Jane        |
| ExampleOrg     | ExampleOrg     | Nikki       |
| null           | null           | Juan        |
+----------------+----------------+-------------+
```

IF

説明: 条件が true の場合、`if(condition, expr1, expr2)` は `expr1` を返し、それ以外の場合は `expr2` を返します。

引数の型:

- サポートされているすべてのデータ型:
- 2 つのパラメータのタイプが異なる場合、関数はセマンティックチェックに失敗します。

戻り値の型:

- いずれか

例:

```
os> source=accounts | eval result = if(true, firstname, lastname) | fields result,
 firstname, lastname
fetched rows / total rows = 4/4
+----------+-------------+----------+
| result   | firstname | lastname   |
|----------+-------------+----------|
| Jane     | Jane      | Doe        |
| Mary     | Mary      | Major      |
| Pat      | Pat       | Candella   |
| Dale     | Jorge     | Souza      |
+----------+-----------+------------+

os> source=accounts | eval result = if(false, firstname, lastname) | fields result,
 firstname, lastname
fetched rows / total rows = 4/4
+----------+-------------+------------+
| result   | firstname   | lastname   |
|----------+-------------+------------|
| Doe      | Jane        | Doe        |
| Major    | Mary        | Major      |
| Candella | Pat         | Candella   |
| Souza    | Jorge       | Souza      |
+----------+-------------+------------+

os> source=accounts | eval is_vip = if(age > 30 AND isnotnull(employer), true, false) |
 fields is_vip, firstname, lastname
```

```
fetched rows / total rows = 4/4
+----------+-------------+------------+
| is_vip   | firstname   | lastname   |
|----------+-------------+------------|
| True     | Jane        | Doe        |
| True     | Mary        | Major      |
| False    | Pat         | Candella   |
| False    | Jorge       | Souza      |
+----------+-------------+------------+
```

PPL 暗号化ハッシュ関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

MD5

MD5 は MD5 ダイジェストを計算し、32 文字の 16 進文字列として値を返します。

使用状況: `md5('hello')`

引数の型:

- STRING

戻り値の型:

- STRING

例:

```
os> source=people | eval `MD5('hello')` = MD5('hello') | fields `MD5('hello')`
fetched rows / total rows = 1/1
+----------------------------------+
| MD5('hello')                     |
|----------------------------------|
| <32 character hex string>        |
```

```
+----------------------------------+
```

SHA1

SHA1 は SHA-1 の 16 進文字列の結果を返します。

使用状況: `sha1('hello')`

引数の型:

- STRING

戻り値の型:

- STRING

例:

```
os> source=people | eval `SHA1('hello')` = SHA1('hello') | fields `SHA1('hello')`
fetched rows / total rows = 1/1
+------------------------------------------+
| SHA1('hello')                            |
|------------------------------------------|
| <40-character SHA-1 hash result>         |
+------------------------------------------+
```

SHA2

SHA2 は、ハッシュ関数の SHA-2 ファミリー (SHA-224、SHA-256, SHA-384、SHA-512) の 16 進文字列結果を返します。numBits は結果の希望するビット長を指定するもので、224、256、384、512 のいずれかである必要があります。

使用方法:

- `sha2('hello',256)`
- `sha2('hello',512)`

引数の型:

- STRING、INTEGER

戻り値の型:

- STRING

例:

```
os> source=people | eval `SHA2('hello',256)` = SHA2('hello',256) | fields
 `SHA2('hello',256)`
fetched rows / total rows = 1/1
+------------------------------------------------------------------+
| SHA2('hello',256)                                                |
|------------------------------------------------------------------|
| <64-character SHA-256 hash result>                               |
+------------------------------------------------------------------+

os> source=people | eval `SHA2('hello',512)` = SHA2('hello',512) | fields
 `SHA2('hello',512)`
fetched rows / total rows = 1/1
+------------------------------------------------------------------+
| SHA2('hello',512)                                                |
                                          |
|------------------------------------------------------------------|
| <128-character SHA-512 hash result>                              |
+------------------------------------------------------------------+
```

PPL 日付および時刻関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

DAY

使用法: DAY(date) は 1～31 の範囲で、日付から月の日数を抽出します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAYOFMONTH、DAY_OF_MONTH

例:

```
os> source=people | eval `DAY(DATE('2020-08-26'))` = DAY(DATE('2020-08-26')) | fields
 `DAY(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+---------------------------+
| DAY(DATE('2020-08-26'))   |
|---------------------------|
| 26                        |
+---------------------------+
```

DAYOFMONTH

使用法: DAYOFMONTH(date) は 1～31 の範囲で、日付から月の日数を抽出します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAY、DAY_OF_MONTH

例:

```
os> source=people | eval `DAYOFMONTH(DATE('2020-08-26'))` =
 DAYOFMONTH(DATE('2020-08-26')) | fields `DAYOFMONTH(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+---------------------------------+
| DAYOFMONTH(DATE('2020-08-26'))   |
|---------------------------------|
| 26                              |
+---------------------------------+
```

DAY_OF_MONTH

使用法: DAY_OF_MONTH(DATE) は 1～31 の範囲で、日付から月の日数を抽出します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAY、DAYOFMONTH

例:

```
os> source=people | eval `DAY_OF_MONTH(DATE('2020-08-26'))` =
 DAY_OF_MONTH(DATE('2020-08-26')) | fields `DAY_OF_MONTH(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| DAY_OF_MONTH(DATE('2020-08-26'))   |
|-----------------------------------|
| 26                                |
+-----------------------------------+
```

DAYOFWEEK

使用法: DAYOFWEEK(DATE) は日付の曜日インデックスを返します (1 = 日曜日、2 = 月曜日、...、7 = 土曜日)。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAY_OF_WEEK

例:

```
os> source=people | eval `DAYOFWEEK(DATE('2020-08-26'))` =
 DAYOFWEEK(DATE('2020-08-26')) | fields `DAYOFWEEK(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+---------------------------------+
| DAYOFWEEK(DATE('2020-08-26'))   |
|---------------------------------|
| 4                               |
+---------------------------------+
```

DAY_OF_WEEK

使用法: DAY_OF_WEEK(DATE) は日付の曜日インデックスを返します (1 = 日曜日、2 = 月曜日、...、7 = 土曜日)。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAYOFWEEK

例:

```
os> source=people | eval `DAY_OF_WEEK(DATE('2020-08-26'))` =
 DAY_OF_WEEK(DATE('2020-08-26')) | fields `DAY_OF_WEEK(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| DAY_OF_WEEK(DATE('2020-08-26'))   |
|-----------------------------------|
| 4                                 |
+-----------------------------------+
```

DAYOFYEAR

使用法: DAYOFYEAR(DATE) は、1 ~ 366 の範囲で、日付から年内の日を返します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAY_OF_YEAR

例:

```
os> source=people | eval `DAYOFYEAR(DATE('2020-08-26'))` =
 DAYOFYEAR(DATE('2020-08-26')) | fields `DAYOFYEAR(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+---------------------------------+
| DAYOFYEAR(DATE('2020-08-26'))   |
|---------------------------------|
| 239                             |
+---------------------------------+
```

DAY_OF_YEAR

使用法: DAY_OF_YEAR(DATE) は、1 ~ 366 の範囲で、日付から年内の日を返します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: DAYOFYEAR

例:

```
os> source=people | eval `DAY_OF_YEAR(DATE('2020-08-26'))` =
 DAY_OF_YEAR(DATE('2020-08-26')) | fields `DAY_OF_YEAR(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| DAY_OF_YEAR(DATE('2020-08-26'))   |
|-----------------------------------|
| 239                               |
+-----------------------------------+
```

DAYNAME

使用法: DAYNAME(DATE) は、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日、日曜日など、日付から曜日の名前を返します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: STRING

例:

```
os> source=people | eval `DAYNAME(DATE('2020-08-26'))` = DAYNAME(DATE('2020-08-26')) |
 fields `DAYNAME(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+-------------------------------+
| DAYNAME(DATE('2020-08-26'))   |
|-------------------------------|
| Wednesday                     |
+-------------------------------+
```

FROM_UNIXTIME

使用法: FROM_UNIXTIME はタイムスタンプまたは文字列値として指定された引数の表現を返します。この関数は、UNIX_TIMESTAMP 関数の逆変換を実行します。

2 番目の引数を指定すると、FROM_UNIXTIME はそれを使用して DATE_FORMAT 関数と同様の結果をフォーマットします。

タイムスタンプが 1970-01-01 00:00:00 ~ 3001-01-18 23:59:59.999999 (0 ~ 32536771199.999999 エポック時間) の範囲外の場合、関数は NULL を返します。

引数タイプ: DOUBLE、STRING

戻り値の型マップ:

DOUBLE -> TIMESTAMP

DOUBLE、STRING -> STRING

例:

```
os> source=people | eval `FROM_UNIXTIME(1220249547)` = FROM_UNIXTIME(1220249547) |
 fields `FROM_UNIXTIME(1220249547)`
fetched rows / total rows = 1/1
+-----------------------------+
| FROM_UNIXTIME(1220249547)   |
|-----------------------------|
| 2008-09-01 06:12:27         |
+-----------------------------+

os> source=people | eval `FROM_UNIXTIME(1220249547, 'HH:mm:ss')` =
 FROM_UNIXTIME(1220249547, 'HH:mm:ss') | fields `FROM_UNIXTIME(1220249547, 'HH:mm:ss')`
fetched rows / total rows = 1/1
+----------------------------------------+
| FROM_UNIXTIME(1220249547, 'HH:mm:ss')   |
|----------------------------------------|
| 06:12:27                               |
+----------------------------------------+
```

HOUR

使用法: HOUR(TIME) は時間の時の値を抽出します。

標準時刻とは異なり、この関数の時間値は 23 を超える範囲を持つことができます。その結果、HOUR(TIME) の戻り値は 23 を超える可能性があります。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: HOUR_OF_DAY

例:

```
os> source=people | eval `HOUR(TIME('01:02:03'))` = HOUR(TIME('01:02:03')) | fields
 `HOUR(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+-------------------------+
| HOUR(TIME('01:02:03'))   |
```

```
|--------------------------|
| 1                        |
+--------------------------+
```

HOUR_OF_DAY

使用法: HOUR_OF_DAY(TIME) は指定された時間から時間値を抽出します。

標準時刻とは異なり、この関数の時間値は 23 を超える範囲を持つことができます。その結果、HOUR_OF_DAY(TIME) の戻り値は 23 を超える可能性があります。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: HOUR

例:

```
os> source=people | eval `HOUR_OF_DAY(TIME('01:02:03'))` =
 HOUR_OF_DAY(TIME('01:02:03')) | fields `HOUR_OF_DAY(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+---------------------------------+
| HOUR_OF_DAY(TIME('01:02:03'))   |
|---------------------------------|
| 1                               |
+---------------------------------+
```

LAST_DAY

使用法: LAST_DAY は、指定された日付引数の DATE 値として月の最終日を返します。

引数タイプ: DATE/STRING/TIMESTAMP/TIME

戻り値の型: DATE

例:

```
os> source=people | eval `last_day('2023-02-06')` = last_day('2023-02-06') | fields
 `last_day('2023-02-06')`
fetched rows / total rows = 1/1
+-------------------------+
| last_day('2023-02-06')   |
```

```
|--------------------------|
| 2023-02-28               |
+--------------------------+
```

LOCALTIMESTAMP

使用法: LOCALTIMESTAMP( ) は NOW( ) のシノニムです。

例:

```
> source=people | eval `LOCALTIMESTAMP()` = LOCALTIMESTAMP() | fields
 `LOCALTIMESTAMP()`
fetched rows / total rows = 1/1
+---------------------+
| LOCALTIMESTAMP()    |
|---------------------|
| 2022-08-02 15:54:19 |
+---------------------+
```

LOCALTIME

使用法: LOCALTIME( ) は NOW( ) のシノニムです。

例:

```
> source=people | eval `LOCALTIME()` = LOCALTIME() | fields `LOCALTIME()`
fetched rows / total rows = 1/1
+---------------------+
| LOCALTIME()         |
|---------------------|
| 2022-08-02 15:54:19 |
+---------------------+
```

MAKE_DATE

使用法: MAKE_DATE は指定された年、月、日の値に基づいて日付値を返します。すべての引数は整数に丸められます。

仕様: 1。MAKE_DATE(INTEGER, INTEGER, INTEGER) -> DATE

引数タイプ: INTEGER、INTEGER、INTEGER

戻り値の型: DATE

例:

```
os> source=people | eval `MAKE_DATE(1945, 5, 9)` = MAKEDATE(1945, 5, 9) | fields
 `MAKEDATE(1945, 5, 9)`
fetched rows / total rows = 1/1
+------------------------+
| MAKEDATE(1945, 5, 9)   |
|------------------------|
| 1945-05-09             |
+------------------------+
```

MINUTE

使用法: MINUTE(TIME) は、指定された時間の分コンポーネントを 0～59 の範囲の整数として返します。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: MINUTE_OF_HOUR

例:

```
os> source=people | eval `MINUTE(TIME('01:02:03'))` =  MINUTE(TIME('01:02:03')) |
 fields `MINUTE(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+----------------------------+
| MINUTE(TIME('01:02:03'))   |
|----------------------------|
| 2                          |
+----------------------------+
```

MINUTE_OF_HOUR

使用法: MINUTE_OF_HOUR(TIME) は、指定された時間の分コンポーネントを 0～59 の範囲の整数として返します。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: MINUTE

例:

```
os> source=people | eval `MINUTE_OF_HOUR(TIME('01:02:03'))` =
 MINUTE_OF_HOUR(TIME('01:02:03')) | fields `MINUTE_OF_HOUR(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| MINUTE_OF_HOUR(TIME('01:02:03'))  |
|-----------------------------------|
| 2                                 |
+-----------------------------------+
```

MONTH

使用法: MONTH(DATE) は、指定された日付の月を整数で返します。範囲は 1～12 です (1 は 1 月、12 は 12 月を表します)。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: MONTH_OF_YEAR

例:

```
os> source=people | eval `MONTH(DATE('2020-08-26'))` =  MONTH(DATE('2020-08-26')) |
 fields `MONTH(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+----------------------------+
| MONTH(DATE('2020-08-26'))  |
|----------------------------|
| 8                          |
+----------------------------+
```

MONTHNAME

使用法: MONTHNAME(DATE) は、指定された日付の月を整数で返します。範囲は 1～12 です (1 は 1 月、12 は 12 月を表します)。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: MONTH_OF_YEAR

例:

```
os> source=people | eval `MONTHNAME(DATE('2020-08-26'))` =
 MONTHNAME(DATE('2020-08-26')) | fields `MONTHNAME(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+---------------------------------+
| MONTHNAME(DATE('2020-08-26'))   |
|---------------------------------|
| August                          |
+---------------------------------+
```

MONTH_OF_YEAR

使用法: MONTH_OF_YEAR(DATE) は、指定された日付の月を整数で返します。範囲は 1 ~ 12 です (1 は 1 月、12 は 12 月を表します)。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

シノニム: MONTH

例:

```
os> source=people | eval `MONTH_OF_YEAR(DATE('2020-08-26'))` =
 MONTH_OF_YEAR(DATE('2020-08-26')) | fields `MONTH_OF_YEAR(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| MONTH_OF_YEAR(DATE('2020-08-26'))   |
|-----------------------------------|
| 8                                 |
+-----------------------------------+
```

NOW

使用法: NOW は現在の日付と時刻を「YYYY-MM-DD hh:mm:ss」形式の TIMESTAMP 値として返します。値はクラスターのタイムゾーンで表されます。

> **Note**
>
> NOW( ) は、ステートメントの実行が開始された時刻を示す一定の時間を返します。これは、SYSDATE( ) とは異なり、実行の正確な時刻を返します。

戻り値の型: TIMESTAMP

仕様: NOW() -> TIMESTAMP

例:

```
os> source=people | eval `value_1` = NOW(), `value_2` = NOW() | fields `value_1`,
 `value_2`
fetched rows / total rows = 1/1
+---------------------+---------------------+
| value_1             | value_2             |
|---------------------+---------------------|
| 2022-08-02 15:39:05 | 2022-08-02 15:39:05 |
+---------------------+---------------------+
```

QUARTER

使用法: QUARTER(DATE) は、指定された日付の四半期を 1～4 の範囲の整数として返します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

例:

```
os> source=people | eval `QUARTER(DATE('2020-08-26'))` = QUARTER(DATE('2020-08-26')) |
 fields `QUARTER(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+------------------------------+
| QUARTER(DATE('2020-08-26'))  |
|------------------------------|
| 3                            |
+------------------------------+
```

SECOND

使用法: SECOND(TIME) は、指定された時間の秒の要素を整数として 0～59 の範囲で返します。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: SECOND_OF_MINUTE

例:

```
os> source=people | eval `SECOND(TIME('01:02:03'))` = SECOND(TIME('01:02:03')) | fields
 `SECOND(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+----------------------------+
| SECOND(TIME('01:02:03'))   |
|----------------------------|
| 3                          |
+----------------------------+
```

SECOND_OF_MINUTE

使用法: SECOND_OF_MINUTE(TIME) は、指定された時間の秒の要素を整数として 0～59 の範囲で返します。

引数タイプ: STRING/TIME/TIMESTAMP

戻り値の型: INTEGER

シノニム: SECOND

例:

```
os> source=people | eval `SECOND_OF_MINUTE(TIME('01:02:03'))` =
 SECOND_OF_MINUTE(TIME('01:02:03')) | fields `SECOND_OF_MINUTE(TIME('01:02:03'))`
fetched rows / total rows = 1/1
+-------------------------------------+
| SECOND_OF_MINUTE(TIME('01:02:03'))   |
|-------------------------------------|
| 3                                   |
+-------------------------------------+
```

SUBDATE

使用法: SUBDATE(DATE, DAYS) は指定された日付から 2 番目の引数 (DATE や DAYS など) を減算します。

引数タイプ: DATE/TIMESTAMP、LONG

戻り値の型マップ: (DATE、LONG) -> DATE

アントニム: ADDDATE

例:

```
os> source=people | eval `'2008-01-02' - 31d` = SUBDATE(DATE('2008-01-02'), 31),
 `'2020-08-26' - 1` = SUBDATE(DATE('2020-08-26'), 1), `ts '2020-08-26 01:01:01' -
 1` = SUBDATE(TIMESTAMP('2020-08-26 01:01:01'), 1) | fields `'2008-01-02' - 31d`,
 `'2020-08-26' - 1`, `ts '2020-08-26 01:01:01' - 1`
fetched rows / total rows = 1/1
+---------------------+--------------------+--------------------------------+
| '2008-01-02' - 31d  | '2020-08-26' - 1   | ts '2020-08-26 01:01:01' - 1   |
|---------------------+--------------------+--------------------------------|
| 2007-12-02 00:00:00 | 2020-08-25         | 2020-08-25 01:01:01            |
+---------------------+--------------------+--------------------------------+
```

SYSDATE

使用法: SYSDATE( ) は現在の日付と時刻を「YYYY-MM-DD hh:mm:ss.nnnnnn」形式の TIMESTAMP 値として返します。

SYSDATE( ) は、実行される正確な時刻を返します。これは NOW() とは異なり、ステートメントの実行が開始された時刻を示す一定の時間を返します。

オプションの引数タイプ: INTEGER (0 ~ 6) – 戻り値の小数秒の桁数を指定します。

戻り値の型: TIMESTAMP

例:

```
os> source=people | eval `SYSDATE()` = SYSDATE() | fields `SYSDATE()`
fetched rows / total rows = 1/1
+----------------------------+
| SYSDATE()                  |
|----------------------------|
| 2022-08-02 15:39:05.123456 |
+----------------------------+
```

TIMESTAMP

使用法: TIMESTAMP(EXPR) は、入力文字列 expr をタイムスタンプとしてタイムスタンプ g 型を構築します。

単一の引数を使用して、TIMESTAMP(expr) は入力からタイムスタンプを作成します。expr が文字列の場合、それはタイムスタンプとして解釈されます。文字列以外の引数の場合、関数は UTC タイムゾーンを使用してタイムスタンプに expr をキャストします。expr が TIME 値の場合、関数はキャスト前に今日の日付を適用します。

2 つの引数で使用すると、TIMESTAMP(expr1, expr2) は日付式またはタイムスタンプ式 (expr1) に時間式 (expr2) を追加し、その結果をタイムスタンプ値として返します。

引数タイプ: STRING/DATE/TIME/TIMESTAMP

戻り値の型マップ:

(STRING/DATE/TIME/TIMESTAMP) -> TIMESTAMP

(STRING/DATE/TIME/TIMESTAMP, STRING/DATE/TIME/TIMESTAMP) -> TIMESTAMP

例:

```
os> source=people | eval `TIMESTAMP('2020-08-26 13:49:00')` = TIMESTAMP('2020-08-26
 13:49:00'), `TIMESTAMP('2020-08-26 13:49:00', TIME('12:15:42'))` =
 TIMESTAMP('2020-08-26 13:49:00', TIME('12:15:42')) | fields `TIMESTAMP('2020-08-26
 13:49:00')`, `TIMESTAMP('2020-08-26 13:49:00', TIME('12:15:42'))`
fetched rows / total rows = 1/1
+-----------------------------------
+-----------------------------------------------------+
| TIMESTAMP('2020-08-26 13:49:00')   | TIMESTAMP('2020-08-26 13:49:00',
 TIME('12:15:42'))   |
|-----------------------------------
+-----------------------------------------------------|
| 2020-08-26 13:49:00               | 2020-08-27 02:04:42
     |
+-----------------------------------
+-----------------------------------------------------+
```

UNIX_TIMESTAMP

使用法: UNIX_TIMESTAMP は指定された日付引数を Unix 時間 (1970 年の初めに開始されたエポックからの秒数) に変換します。引数を指定しない場合、現在の Unix 時間を返します。

date 引数は、DATE、TIMESTAMP 文字列、または YYMMDD、YYMMDDhhmmss、YYYYMMDD、のいずれかの形式の数値にすることができますYYYYMMDDhhmmss。引数に時間の要素が含まれている場合、オプションで小数秒が含まれることがあります。

引数が無効な形式であるか、1970-01-01 00:00:00 から 3001-01-18 23:59:59.999999 (エポック時間では 0 から 32536771199.999999) の範囲外である場合、関数は NULL を返します。

この関数は、DATE、TIMESTAMP、または を引数タイプ DOUBLE として受け入れるか、引数なしを受け入れます。常に Unix タイムスタンプを表す DOUBLE 値を返します。

逆変換には、FROM_UNIXTIME 関数を使用できます。

引数タイプ: <NONE>/DOUBLE/DATE/TIMESTAMP

戻り値の型: DOUBLE

例:

```
os> source=people | eval `UNIX_TIMESTAMP(double)` = UNIX_TIMESTAMP(20771122143845),
 `UNIX_TIMESTAMP(timestamp)` = UNIX_TIMESTAMP(TIMESTAMP('1996-11-15 17:05:42')) |
 fields `UNIX_TIMESTAMP(double)`, `UNIX_TIMESTAMP(timestamp)`
fetched rows / total rows = 1/1
+-------------------------+----------------------------+
| UNIX_TIMESTAMP(double)  | UNIX_TIMESTAMP(timestamp)  |
|-------------------------+----------------------------|
| 3404817525.0            | 848077542.0                |
+-------------------------+----------------------------+
```

WEEK

使用法: WEEK(DATE) は特定の日付の週番号を返します。

引数タイプ: DATE/TIMESTAMP/STRING

戻り値の型: INTEGER

シノニム: WEEK_OF_YEAR

例:

```
os> source=people | eval `WEEK(DATE('2008-02-20'))` = WEEK(DATE('2008-02-20')) | fields
 `WEEK(DATE('2008-02-20'))`
fetched rows / total rows = 1/1
+---------------------------+
| WEEK(DATE('2008-02-20'))  |
|---------------------------|
| 8                         |
```

```
+----------------------------+
```

WEEKDAY

使用法: WEEKDAY(DATE) は日付の曜日インデックスを返します (0 = 月曜日、1 = 火曜日、...、6 = 日曜日)。

これは dayofweek 関数に似ていますが、日ごとに異なるインデックスを返します。

引数タイプ: STRING/DATE/TIME/TIMESTAMP

戻り値の型: INTEGER

例:

```
os> source=people | eval `weekday(DATE('2020-08-26'))` = weekday(DATE('2020-08-26'))
 | eval `weekday(DATE('2020-08-27'))` = weekday(DATE('2020-08-27')) | fields
 `weekday(DATE('2020-08-26'))`, `weekday(DATE('2020-08-27'))`
fetched rows / total rows = 1/1
+-------------------------------+-------------------------------+
| weekday(DATE('2020-08-26'))   | weekday(DATE('2020-08-27'))   |
|-------------------------------+-------------------------------|
| 2                             | 3                             |
+-------------------------------+-------------------------------+
```

WEEK_OF_YEAR

使用法: 指定された日付の週番号 WEEK_OF_YEAR(DATE) を返します。

引数タイプ: DATE/TIMESTAMP/STRING

戻り値の型: INTEGER

シノニム: WEEK

例:

```
os> source=people | eval `WEEK_OF_YEAR(DATE('2008-02-20'))` = WEEK(DATE('2008-02-20'))|
 fields `WEEK_OF_YEAR(DATE('2008-02-20'))`
fetched rows / total rows = 1/1
+-----------------------------------+
| WEEK_OF_YEAR(DATE('2008-02-20'))  |
|-----------------------------------|
| 8                                 |
```

```
+-----------------------------------+
```

YEAR

使用法: YEAR(DATE) は日付の年を返します (範囲: 1000～9999)。「ゼロ」日付の場合は 0 を返します。

引数タイプ: STRING/DATE/TIMESTAMP

戻り値の型: INTEGER

例:

```
os> source=people | eval `YEAR(DATE('2020-08-26'))` = YEAR(DATE('2020-08-26')) | fields
 `YEAR(DATE('2020-08-26'))`
fetched rows / total rows = 1/1
+----------------------------+
| YEAR(DATE('2020-08-26'))   |
|----------------------------|
| 2020                       |
+----------------------------+
```

DATE_ADD

使用法: DATE_ADD(date, INTERVAL expr unit) は、指定された間隔を指定された日付に追加します。

引数タイプ: DATE、INTERVAL

戻り値の型: DATE

アントニム: DATE_SUB

例:

```
os> source=people | eval `'2020-08-26' + 1d` = DATE_ADD(DATE('2020-08-26'), INTERVAL 1
 DAY) | fields `'2020-08-26' + 1d`
fetched rows / total rows = 1/1
+---------------------+
| '2020-08-26' + 1d   |
|---------------------|
| 2020-08-27          |
+---------------------+
```

DATE_SUB

使用法: DATE_SUB(date, INTERVAL expr unit) は日付から間隔 expr を減算します。

引数タイプ: DATE、INTERVAL

戻り値の型: DATE

アントニム: DATE_ADD

例:

```
os> source=people | eval `'2008-01-02' - 31d` = DATE_SUB(DATE('2008-01-02'), INTERVAL
 31 DAY) | fields `'2008-01-02' - 31d`
fetched rows / total rows = 1/1
+---------------------+
| '2008-01-02' - 31d  |
|---------------------|
| 2007-12-02          |
+---------------------+
```

TIMESTAMPADD

使用法: 指定した時間間隔を特定の日付に追加した後、TIMESTAMP 値を返します。

引数:

- 時間間隔: INTERVAL
  (SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER、YEAR)
- 整数: INTEGER
- 日付: DATE、TIMESTAMP、または STRING

STRING を日付引数として指定する場合は、有効な TIMESTAMP としてフォーマットします。関数は DATE 引数を自動的に TIMESTAMP に変換します。

例:

```
os> source=people | eval `TIMESTAMPADD(DAY, 17, '2000-01-01 00:00:00')` =
 TIMESTAMPADD(DAY, 17, '2000-01-01 00:00:00') | eval `TIMESTAMPADD(QUARTER, -1,
 '2000-01-01 00:00:00')` = TIMESTAMPADD(QUARTER, -1, '2000-01-01 00:00:00') | fields
```

```
 `TIMESTAMPADD(DAY, 17, '2000-01-01 00:00:00')`, `TIMESTAMPADD(QUARTER, -1, '2000-01-01
 00:00:00')`
fetched rows / total rows = 1/1
+-----------------------------------------------
+-------------------------------------------------+
| TIMESTAMPADD(DAY, 17, '2000-01-01 00:00:00') | TIMESTAMPADD(QUARTER, -1, '2000-01-01
 00:00:00') |
|-----------------------------------------------
+-------------------------------------------------|
| 2000-01-18 00:00:00                           | 1999-10-01 00:00:00
           |
+-----------------------------------------------
+-------------------------------------------------+
```

TIMESTAMPDIFF

使用法: TIMESTAMPDIFF(interval, start, end) は指定した間隔単位で開始日時と終了日時の差を返します。

引数:

- 時間間隔: INTERVAL (SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER、YEAR)
- 開始: DATE、TIMESTAMP、または STRING
- 終了: DATE、TIMESTAMP、または STRING

関数は、必要に応じて引数を自動的に TIMESTAMP に変換します。STRING 引数を有効な TIMESTAMP としてフォーマットします。

例:

```
os> source=people | eval `TIMESTAMPDIFF(YEAR, '1997-01-01 00:00:00', '2001-03-06
 00:00:00')` = TIMESTAMPDIFF(YEAR, '1997-01-01 00:00:00', '2001-03-06 00:00:00') |
 eval `TIMESTAMPDIFF(SECOND, timestamp('1997-01-01 00:00:23'), timestamp('1997-01-01
 00:00:00'))` = TIMESTAMPDIFF(SECOND, timestamp('1997-01-01 00:00:23'),
 timestamp('1997-01-01 00:00:00')) | fields `TIMESTAMPDIFF(YEAR, '1997-01-01 00:00:00',
 '2001-03-06 00:00:00')`, `TIMESTAMPDIFF(SECOND, timestamp('1997-01-01 00:00:23'),
 timestamp('1997-01-01 00:00:00'))`
fetched rows / total rows = 1/1
+-------------------------------------------------------------------
+-------------------------------------------------------------------------------------------
+
```

```
| TIMESTAMPDIFF(YEAR, '1997-01-01 00:00:00', '2001-03-06 00:00:00') |
 TIMESTAMPDIFF(SECOND, timestamp('1997-01-01 00:00:23'), timestamp('1997-01-01
 00:00:00')) |
|-------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------|
| 4                                                                  | -23
                                                                        |
+-------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------
+
```

UTC_TIMESTAMP

使用法: UTC_TIMESTAMP は 'YYYY-MM-DD hh:mm:ss' の値として現在の UTC タイムスタンプを返します。

戻り値の型: TIMESTAMP

仕様: UTC_TIMESTAMP() -> TIMESTAMP

例:

```
> source=people | eval `UTC_TIMESTAMP()` = UTC_TIMESTAMP() | fields `UTC_TIMESTAMP()`
fetched rows / total rows = 1/1
+---------------------+
| UTC_TIMESTAMP()     |
|---------------------|
| 2022-10-03 17:54:28 |
+---------------------+
```

CURRENT_TIMEZONE

使用法: CURRENT_TIMEZONE は現在のローカルタイムゾーンを返します。

戻り値の型: STRING

例:

```
> source=people | eval `CURRENT_TIMEZONE()` = CURRENT_TIMEZONE() | fields
 `CURRENT_TIMEZONE()`
fetched rows / total rows = 1/1
+------------------------+
```

```
| CURRENT_TIMEZONE()     |
|------------------------|
| America/Chicago        |
+------------------------+
```

PPL 式

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

式、特に値式はスカラー値を返します。式にはさまざまな型と形式があります。例えば、リテラル値はアトム式として扱われ、それらを基盤として算術式、述語式、関数式が構築されます。式は、Filter および Stats コマンドで算術式を使用するなど、さまざまな句で使用できます。

演算子

算術式は、次のように数値リテラルとバイナリ算術演算子によって形成された式です。

1. +: 加算。
2. -: 減算。
3. *: 乗算。
4. /: 除算 (整数の場合、結果は小数部が破棄された整数です)
5. %: 剰余 (整数でのみ使用、結果は除算の余り)

優先順位

括弧を使用して、算術演算子の優先順位を制御します。それ以外の場合は、優先順位の高い演算子が最初に実行されます。

型変換

演算子のシグネチャを照合する際、暗黙の型変換が適用されます。例えば、整数 + の実数がシグネチャ +(double,double) と一致すると、結果として実数が得られます。このルールは関数呼び出しにも適用されます。

さまざまなタイプの算術式の例:

```
os> source=accounts | where age > (25 + 5) | fields age ;
fetched rows / total rows = 3/3
+-------+
| age   |
|-------|
| 32    |
| 36    |
| 33    |
+-------+
```

述語演算子

述語演算子は、true と評価される式です。MISSING と NULL 値の比較は、次のルールに従います。

- MISSING 値は MISSING 値とのみ等しく、他の値よりも小さくなります。
- NULL 値は NULL 値と等しく、MISSING 値よりも大きく、他のすべての値よりも小さくなります。

演算子

述語演算子

名前	説明
>	大なり演算子
>=	以上演算子
<	未満演算子
!=	不等号演算子
<=	以下演算子
=	等号演算子
LIKE	単純パターンマッチング
IN	NULL 値テスト
AND	AND 演算子

名前	説明
OR	OR 演算子
XOR	XOR 演算子
NOT	NOT NULL 値テスト

日付時刻を比較できます。異なる日付時刻タイプ (DATE や TIME など) を比較する場合、どちらも DATETIME に変換されます。以下のルールが変換に適用されます:

- TIME は今日の日付に適用されます。
- DATE は午前 0 時に解釈されます。

基本的な述語演算子

比較演算子の例:

```
os> source=accounts | where age > 33 | fields age ;
fetched rows / total rows = 1/1
+-------+
| age   |
|-------|
| 36    |
+-------+
```

IN

値リストの IN 演算子テストフィールドの例:

```
os> source=accounts | where age in (32, 33) | fields age ;
fetched rows / total rows = 2/2
+-------+
| age   |
|-------|
| 32    |
| 33    |
+-------+
```

OR

OR 演算子の例:

```
os> source=accounts | where age = 32 OR age = 33 | fields age ;
fetched rows / total rows = 2/2
+-------+
| age   |
|-------|
| 32    |
| 33    |
+-------+
```

NOT

NOT 演算子の例:

```
os> source=accounts | where age not in (32, 33) | fields age ;
fetched rows / total rows = 2/2
+-------+
| age   |
|-------|
| 36    |
| 28    |
+-------+
```

PPL IP アドレス関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

CIDRMATCH

使用法: `CIDRMATCH(ip, cidr)` は指定された IP アドレスが指定された cidr 範囲内にあるかどうかを確認します。

引数の型:

- STRING、STRING
- 戻り値の型: BOOLEAN

例:

```
os> source=ips | where cidrmatch(ip, '***********/24') | fields ip
fetched rows / total rows = 1/1
+--------------+
| ip           |
|--------------|
| ***********  |
+--------------+

os> source=ipsv6 | where cidrmatch(ip, '2003:db8::/32') | fields ip
fetched rows / total rows = 1/1
+----------------------------------------+
| ip                                     |
|----------------------------------------|
| 2003:0db8:****:****:****:****:****:0000 |
+----------------------------------------+
```

> **Note**
>
> - `ip` は IPv4 アドレスまたは IPv6 アドレスにすることができます。
> - `cidr` は IPv4 または IPv6 ブロックにすることができます。
> - `ip` と `cidr` は、両方の IPv4 または両方の IPv6 である必要があります。
> - `ip` と `cidr` は両方とも有効で、空でない/null でない必要があります。

PPL JSON 関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

`JSON`

使用法: `json(value)` は文字列を JSON 形式で解析できるかどうかを評価します。この関数は、有効な JSON の場合は元の文字列を返し、無効な場合は null を返します。

引数タイプ: STRING

戻り値の型: STRING/NULL。有効な JSON オブジェクト形式の STRING 式。

例:

```
os> source=people | eval `valid_json()` = json('[1,2,3,{"f1":1,"f2":[5,6]},4]') |
 fields valid_json
fetched rows / total rows = 1/1
+-----------------------------------+
| valid_json                        |
+-----------------------------------+
| [1,2,3,{"f1":1,"f2":[5,6]},4]   |
+-----------------------------------+

os> source=people | eval `invalid_json()` = json('{"invalid": "json"') | fields
 invalid_json
fetched rows / total rows = 1/1
+----------------+
| invalid_json   |
+----------------+
| null           |
+----------------+
```

JSON_OBJECT

使用法: `json_object(<key>, <value>[, <key>, <value>]...)` はキーと値のペアのメンバーから JSON オブジェクトを返します。

引数の型:

- <key> は STRING である必要があります。
- <value> は任意のデータ型にすることができます。

戻り値の型: JSON_OBJECT。有効な JSON オブジェクトの StructType 式。

例:

```
os> source=people | eval result = json_object('key', 123.45) | fields result
fetched rows / total rows = 1/1
+------------------+
| result           |
+------------------+
```

```
| {"key":123.45}   |
+------------------+

os> source=people | eval result = json_object('outer', json_object('inner', 123.45)) |
 fields result
fetched rows / total rows = 1/1
+-----------------------------+
| result                      |
+-----------------------------+
| {"outer":{"inner":123.45}}  |
+-----------------------------+
```

JSON_ARRAY

使用法: json_array(<value>...) は、値のリストを使用して JSON ARRAY を作成します。

引数タイプ: <value> は、文字列、数値、ブール値など、任意の種類の値にすることができます。

戻り値の型: ARRAY。有効な JSON 配列でサポートされている任意のデータ型の配列。

例:

```
os> source=people | eval `json_array` = json_array(1, 2, 0, -1, 1.1, -0.11)
fetched rows / total rows = 1/1
+-----------------------------+
| json_array                  |
+-----------------------------+
| [1.0,2.0,0.0,-1.0,1.1,-0.11] |
+-----------------------------+

os> source=people | eval `json_array_object` = json_object("array", json_array(1, 2, 0,
 -1, 1.1, -0.11))
fetched rows / total rows = 1/1
+---------------------------------------+
| json_array_object                     |
+---------------------------------------+
| {"array":[1.0,2.0,0.0,-1.0,1.1,-0.11]} |
+---------------------------------------+
```

TO_JSON_STRING

使用法: to_json_string(jsonObject) は指定された JSON オブジェクト値を持つ JSON 文字列を返します。

引数タイプ: JSON_OBJECT

戻り値の型: STRING

例:

```
os> source=people | eval `json_string` = to_json_string(json_array(1, 2, 0, -1, 1.1,
 -0.11)) | fields json_string
fetched rows / total rows = 1/1
+--------------------------------+
| json_string                    |
+--------------------------------+
| [1.0,2.0,0.0,-1.0,1.1,-0.11]   |
+--------------------------------+

os> source=people | eval `json_string` = to_json_string(json_object('key', 123.45)) |
 fields json_string
fetched rows / total rows = 1/1
+-----------------+
| json_string     |
+-----------------+
| {'key', 123.45} |
+-----------------+
```

ARRAY_LENGTH

使用法: `array_length(jsonArray)` は最も外側の配列に含まれる要素の数を返します。

引数タイプ: ARRAY。ARRAY または JSON_ARRAY オブジェクト。

戻り値の型: INTEGER

例:

```
os> source=people | eval `json_array` = json_array_length(json_array(1,2,3,4)),
 `empty_array` = json_array_length(json_array())
fetched rows / total rows = 1/1
+--------------+---------------+
| json_array   | empty_array   |
+--------------+---------------+
| 4            | 0             |
+--------------+---------------+
```

JSON_EXTRACT

使用法: json_extract(jsonStr, path) は指定された JSON パスに基づいて JSON 文字列から JSON オブジェクトを抽出します。入力 JSON 文字列が無効の場合、関数は null を返します。

引数タイプ: STRING、STRING

戻り値の型: STRING

- 有効な JSON オブジェクト形式の STRING 式。
- 無効な JSON の場合、NULL が返されます。

例:

```
os> source=people | eval `json_extract('{"a":"b"}', '$.a')` = json_extract('{"a":"b"}',
 '$a')
fetched rows / total rows = 1/1
+----------------------------------+
| json_extract('{"a":"b"}', 'a')   |
+----------------------------------+
| b                                |
+----------------------------------+

os> source=people | eval `json_extract('{"a":[{"b":1},{"b":2}]}', '$.a[1].b')` =
 json_extract('{"a":[{"b":1},{"b":2}]}', '$.a[1].b')
fetched rows / total rows = 1/1
+------------------------------------------------------------+
| json_extract('{"a":[{"b":1.0},{"b":2.0}]}', '$.a[1].b')   |
+------------------------------------------------------------+
| 2.0                                                        |
+------------------------------------------------------------+

os> source=people | eval `json_extract('{"a":[{"b":1},{"b":2}]}', '$.a[*].b')` =
 json_extract('{"a":[{"b":1},{"b":2}]}', '$.a[*].b')
fetched rows / total rows = 1/1
+------------------------------------------------------------+
| json_extract('{"a":[{"b":1.0},{"b":2.0}]}', '$.a[*].b')   |
+------------------------------------------------------------+
| [1.0,2.0]                                                  |
+------------------------------------------------------------+

os> source=people | eval `invalid_json` = json_extract('{"invalid": "json"')
```

```
fetched rows / total rows = 1/1
+----------------+
| invalid_json   |
+----------------+
| null           |
+----------------+
```

JSON_KEYS

使用法: json_keys(jsonStr) は最も外側の JSON オブジェクトのすべてのキーを配列として返します。

引数タイプ: STRING。有効な JSON オブジェクト形式の STRING 式。

戻り値の型: ARRAY[STRING]。この関数は、他の有効な JSON 文字列、空の文字列、または無効な JSON に対して NULL を返します。

例:

```
os> source=people | eval `keys` = json_keys('{"f1":"abc","f2":{"f3":"a","f4":"b"}}')
fetched rows / total rows = 1/1
+------------+
| keus       |
+------------+
| [f1, f2]   |
+------------+

os> source=people | eval `keys` = json_keys('[1,2,3,{"f1":1,"f2":[5,6]},4]')
fetched rows / total rows = 1/1
+--------+
| keys   |
+--------+
| null   |
+--------+
```

JSON_VALID

使用法: json_valid(jsonStr) は JSON 文字列が有効な JSON 構文を使用しているかどうかを評価し、TRUE または FALSE を返します。

引数タイプ: STRING

戻り値の型: BOOLEAN

例:

```
os> source=people | eval `valid_json` = json_valid('[1,2,3,4]'), `invalid_json` =
 json_valid('{"invalid": "json"') | feilds `valid_json`, `invalid_json`
fetched rows / total rows = 1/1
+--------------+----------------+
| valid_json   | invalid_json   |
+--------------+----------------+
| True         | False          |
+--------------+----------------+

os> source=accounts | where json_valid('[1,2,3,4]') and isnull(email) | fields
 account_number, email
fetched rows / total rows = 1/1
+------------------+---------+
| account_number   | email   |
|------------------+---------|
| 13               | null    |
+------------------+---------+
```

PPL Lambda 関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

EXISTS

使用法: `exists(array, lambda)` は Lambda 述語が配列内の 1 つ以上の要素を保持するかどうかを評価します。

引数タイプ: ARRAY、LAMBDA

戻り値の型: BOOLEAN。配列内の少なくとも 1 つの要素が Lambda 述語を満たす場合は TRUE を返し、それ以外の場合は FALSE を返します。

例:

```
os> source=people | eval array = json_array(1, -1, 2), result = exists(array, x -> x >
0) | fields result
```

```
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| true      |
+-----------+

os> source=people | eval array = json_array(-1, -3, -2), result = exists(array, x -> x
 > 0) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| false     |
+-----------+
```

FILTER

使用法: `filter(array, lambda)` は、指定された Lambda 関数を使用して入力配列をフィルタリングします。

引数タイプ: ARRAY、LAMBDA

戻り値の型: ARRAY。Lambda 述語を満たす入力配列内のすべての要素を含む ARRAY。

例:

```
 os> source=people | eval array = json_array(1, -1, 2), result = filter(array, x -> x >
 0) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| [1, 2]    |
+-----------+

os> source=people | eval array = json_array(-1, -3, -2), result = filter(array, x -> x
 > 0) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| []        |
```

```
+-----------+
```

TRANSFORM

使用法: transform(array, lambda) は Lambda 変換関数を使用して配列内の要素を変換します。2 番目の引数は、バイナリ Lambda 関数を使用する場合の要素のインデックスを意味します。これは、機能プログラミングの map に類似しています。

引数タイプ: ARRAY、LAMBDA

戻り値の型: ARRAY。Lambda 変換関数を入力配列の各要素に適用した結果を含む ARRAY。

例:

```
os> source=people | eval array = json_array(1, 2, 3), result = transform(array, x -> x
 + 1) | fields result
fetched rows / total rows = 1/1
+--------------+
| result       |
+--------------+
| [2, 3, 4]    |
+--------------+

os> source=people | eval array = json_array(1, 2, 3), result = transform(array, (x, i)
 -> x + i) | fields result
fetched rows / total rows = 1/1
+--------------+
| result       |
+--------------+
| [1, 3, 5]    |
+--------------+
```

REDUCE

使用法: reduce(array, start, merge_lambda, finish_lambda) は Lambda 関数を適用することで、配列を 1 つの値に集約します。この関数は、開始値とすべての配列要素に merge_lambda を適用し、結果に finish_lambda を適用します。

引数タイプ: ARRAY、ANY、LAMBDA、LAMBDA

戻り値の型: ANY。Lambda 関数を開始値と入力配列に適用した最終結果。

例:

```
 os> source=people | eval array = json_array(1, 2, 3), result = reduce(array, 0, (acc,
 x) -> acc + x) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| 6         |
+-----------+

os> source=people | eval array = json_array(1, 2, 3), result = reduce(array, 10, (acc,
 x) -> acc + x) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| 16        |
+-----------+

os> source=people | eval array = json_array(1, 2, 3), result = reduce(array, 0, (acc,
 x) -> acc + x, acc -> acc * 10) | fields result
fetched rows / total rows = 1/1
+-----------+
| result    |
+-----------+
| 60        |
+-----------+
```

PPL 数学関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

ABS

使用法: ABS(x) は x の絶対値を計算します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: INTEGER/LONG/FLOAT/DOUBLE

例:

```
os> source=people | eval `ABS(-1)` = ABS(-1) | fields `ABS(-1)`
fetched rows / total rows = 1/1
+-----------+
| ABS(-1)   |
|-----------|
| 1         |
+-----------+
```

ACOS

使用法: ACOS(x) は x のアークコサインを計算します。x が -1 ~ 1 の範囲にない場合、NULL を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `ACOS(0)` = ACOS(0) | fields `ACOS(0)`
fetched rows / total rows = 1/1
+--------------------+
| ACOS(0)            |
|--------------------|
| 1.5707963267948966 |
+--------------------+
```

ASIN

使用法: asin(x) は x のアークサインを計算します。x が -1 ~ 1 の範囲にない場合、NULL を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `ASIN(0)` = ASIN(0) | fields `ASIN(0)`
fetched rows / total rows = 1/1
```

```
+-----------+
| ASIN(0)   |
|-----------|
| 0.0       |
+-----------+
```

ATAN

使用法: ATAN(x) は x のアークタンジェントを計算します。atan(y, x) は y/x のアークタンジェントを計算します。ただし、両引数の符号によって結果の象限が決まります。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `ATAN(2)` = ATAN(2), `ATAN(2, 3)` = ATAN(2, 3) | fields
 `ATAN(2)`, `ATAN(2, 3)`
fetched rows / total rows = 1/1
+--------------------+--------------------+
| ATAN(2)            | ATAN(2, 3)         |
|--------------------+--------------------|
| 1.1071487177940904 | 0.5880026035475675 |
+--------------------+--------------------+
```

ATAN2

使用法: ATAN2(y, x) は y/x のアークタンジェントを計算しますが、両引数の符号によって結果の象限が決まります。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `ATAN2(2, 3)` = ATAN2(2, 3) | fields `ATAN2(2, 3)`
fetched rows / total rows = 1/1
+--------------------+
| ATAN2(2, 3)        |
|--------------------|
| 0.5880026035475675 |
```

```
+--------------------+
```

CBRT

使用法: CBRT は数値の立方根を計算します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE:

INTEGER/LONG/FLOAT/DOUBLE -> DOUBLE

例:

```
opensearchsql> source=location | eval `CBRT(8)` = CBRT(8), `CBRT(9.261)` = CBRT(9.261),
 `CBRT(-27)` = CBRT(-27) | fields `CBRT(8)`, `CBRT(9.261)`, `CBRT(-27)`;
fetched rows / total rows = 2/2
+-----------+---------------+-------------+
| CBRT(8)   | CBRT(9.261)   | CBRT(-27)   |
|-----------+---------------+-------------|
| 2.0       | 2.1           | -3.0        |
| 2.0       | 2.1           | -3.0        |
+-----------+---------------+-------------+
```

CEIL

使用法: CEILING 関数の工イリアス。CEILING(T) は値 T の上限を取ります。

制限: CEILING は IEEE 754 ダブルタイプが保存時に 10 進数を表示する場合にのみ、予想どおりに動作します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: LONG

例:

```
os> source=people | eval `CEILING(0)` = CEILING(0), `CEILING(50.00005)` =
 CEILING(50.00005), `CEILING(-50.00005)` = CEILING(-50.00005) | fields `CEILING(0)`,
 `CEILING(50.00005)`, `CEILING(-50.00005)`
fetched rows / total rows = 1/1
+--------------+---------------------+----------------------+
| CEILING(0)   | CEILING(50.00005)   | CEILING(-50.00005)   |
```

```
|--------------+---------------------+----------------------|
| 0            | 51                  | -50                  |
+--------------+---------------------+----------------------+

os> source=people | eval `CEILING(3147483647.12345)` = CEILING(3147483647.12345),
 `CEILING(113147483647.12345)` = CEILING(113147483647.12345),
 `CEILING(3147483647.00001)` = CEILING(3147483647.00001) | fields
 `CEILING(3147483647.12345)`, `CEILING(113147483647.12345)`,
 `CEILING(3147483647.00001)`
fetched rows / total rows = 1/1
+-----------------------------+------------------------------
+-----------------------------+
| CEILING(3147483647.12345)   | CEILING(113147483647.12345)   |
 CEILING(3147483647.00001)   |
|-----------------------------+------------------------------
+-----------------------------|
| 3147483648                  | 113147483648                  | 3147483648
     |
+-----------------------------+------------------------------
+-----------------------------+
```

CONV

使用法: CONV(x, a, b) は数値 x を a ベースから b ベースに変換します。

引数タイプ: x: STRING、a: INTEGER、b: INTEGER

戻り値の型: STRING

例:

```
os> source=people | eval `CONV('12', 10, 16)` = CONV('12', 10, 16), `CONV('2C', 16,
 10)` = CONV('2C', 16, 10), `CONV(12, 10, 2)` = CONV(12, 10, 2), `CONV(1111, 2, 10)` =
 CONV(1111, 2, 10) | fields `CONV('12', 10, 16)`, `CONV('2C', 16, 10)`, `CONV(12, 10,
 2)`, `CONV(1111, 2, 10)`
fetched rows / total rows = 1/1
+----------------------+----------------------+------------------
+---------------------+
| CONV('12', 10, 16)   | CONV('2C', 16, 10)   | CONV(12, 10, 2)   | CONV(1111, 2, 10)
 |
|----------------------+----------------------+------------------
+---------------------|
| c                    | 44                   | 1100              | 15
 |
```

```
+----------------------+----------------------+------------------
+----------------------+
```

COS

使用法: COS(x) は x のコサインを計算します。x はラジアンで表されます。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `COS(0)` = COS(0) | fields `COS(0)`
fetched rows / total rows = 1/1
+----------+
| COS(0)   |
|----------|
| 1.0      |
+----------+
```

COT

使用法: COT(x) は x のコタンジェントを計算します。x が 0 に等しい場合、範囲外エラーを返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `COT(1)` = COT(1) | fields `COT(1)`
fetched rows / total rows = 1/1
+--------------------+
| COT(1)             |
|--------------------|
| 0.6420926159343306 |
+--------------------+
```

CRC32

使用法: CRC32 は巡回冗長検査値を計算し、32 ビットの符号なし値を返します。

引数タイプ: STRING

戻り値の型: LONG

例:

```
os> source=people | eval `CRC32('MySQL')` = CRC32('MySQL') | fields `CRC32('MySQL')`
fetched rows / total rows = 1/1
+------------------+
| CRC32('MySQL')   |
|------------------|
| 3259397556       |
+------------------+
```

DEGREES

使用法: DEGREES(x) は x をラジアンから度に変換します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `DEGREES(1.57)` = DEGREES(1.57) | fields `DEGREES(1.57)`
fetched rows / total rows  = 1/1
+-------------------+
| DEGREES(1.57)     |
|-------------------|
| 89.95437383553924 |
+-------------------+
```

E

使用法: E() はオイラー数を返します。

戻り値の型: DOUBLE

例:

```
os> source=people | eval `E()` = E() | fields `E()`
fetched rows / total rows = 1/1
```

```
+-------------------+
| E()               |
|-------------------|
| 2.718281828459045 |
+-------------------+
```

EXP

使用法: EXP(x) は e の x 乗を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `EXP(2)` = EXP(2) | fields `EXP(2)`
fetched rows / total rows = 1/1
+------------------+
| EXP(2)           |
|------------------|
| 7.38905609893065 |
+------------------+
```

FLOOR

使用法: FLOOR(T) は値 T の下限を取ります。

制限: FLOOR は IEEE 754 ダブルタイプが保存時に 10 進数を表示する場合にのみ、予想どおりに動作します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: LONG

例:

```
os> source=people | eval `FLOOR(0)` = FLOOR(0), `FLOOR(50.00005)` = FLOOR(50.00005),
 `FLOOR(-50.00005)` = FLOOR(-50.00005) | fields `FLOOR(0)`, `FLOOR(50.00005)`,
 `FLOOR(-50.00005)`
fetched rows / total rows = 1/1
+------------+-------------------+--------------------+
| FLOOR(0)   | FLOOR(50.00005)   | FLOOR(-50.00005)   |
```

```
|------------+------------------+--------------------|
| 0          | 50               | -51                |
+------------+------------------+--------------------+

os> source=people | eval `FLOOR(3147483647.12345)` = FLOOR(3147483647.12345),
 `FLOOR(113147483647.12345)` = FLOOR(113147483647.12345), `FLOOR(3147483647.00001)`
 = FLOOR(3147483647.00001) | fields `FLOOR(3147483647.12345)`,
 `FLOOR(113147483647.12345)`, `FLOOR(3147483647.00001)`
fetched rows / total rows = 1/1
+--------------------------+----------------------------+--------------------------+
| FLOOR(3147483647.12345)  | FLOOR(113147483647.12345)  | FLOOR(3147483647.00001)  |
|--------------------------+----------------------------+--------------------------|
| 3147483647               | 113147483647               | 3147483647               |
+--------------------------+----------------------------+--------------------------+

os> source=people | eval `FLOOR(282474973688888.022)` = FLOOR(282474973688888.022),
 `FLOOR(9223372036854775807.022)` = FLOOR(9223372036854775807.022),
 `FLOOR(9223372036854775807.0000001)` = FLOOR(9223372036854775807.0000001)
 | fields `FLOOR(282474973688888.022)`, `FLOOR(9223372036854775807.022)`,
 `FLOOR(9223372036854775807.0000001)`
fetched rows / total rows = 1/1
+-----------------------------+-----------------------------------
+-------------------------------------+
| FLOOR(282474973688888.022)  | FLOOR(9223372036854775807.022)  |
 FLOOR(9223372036854775807.0000001)  |
|-----------------------------+-----------------------------------
+-------------------------------------|
| 282474973688888             | 9223372036854775807             | 9223372036854775807
                 |
+-----------------------------+-----------------------------------
+-------------------------------------+
```

LN

使用法: LN(x) は x の自然対数を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `LN(2)` = LN(2) | fields `LN(2)`
fetched rows / total rows = 1/1
```

```
+--------------------+
| LN(2)              |
|--------------------|
| 0.6931471805599453 |
+--------------------+
```

LOG

使用法: LOG(x) は x の自然対数 (底が e の対数) を返します。log(B, x) は log(x)/log(B) と同等です。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `LOG(2)` = LOG(2), `LOG(2, 8)` = LOG(2, 8) | fields `LOG(2)`,
 `LOG(2, 8)`
fetched rows / total rows = 1/1
+--------------------+-------------+
| LOG(2)             | LOG(2, 8)   |
|--------------------+-------------|
| 0.6931471805599453 | 3.0         |
+--------------------+-------------+
```

LOG2

使用法: LOG2(x) は log(x)/log(2) に相当します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `LOG2(8)` = LOG2(8) | fields `LOG2(8)`
fetched rows / total rows = 1/1
+-----------+
| LOG2(8)   |
|-----------|
| 3.0       |
+-----------+
```

LOG10

使用法: LOG10(x) は log(x)/log(10) に相当します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `LOG10(100)` = LOG10(100) | fields `LOG10(100)`
fetched rows / total rows = 1/1
+--------------+
| LOG10(100)   |
|--------------|
| 2.0          |
+--------------+
```

MOD

使用法: MOD(n, m) は数値 n の残りを m で割って計算します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: m がゼロ以外の値の場合、n 型と m 型の間でより広い型。m が 0 に等しい場合、NULL を返します。

例:

```
os> source=people | eval `MOD(3, 2)` = MOD(3, 2), `MOD(3.1, 2)` = MOD(3.1, 2) | fields
 `MOD(3, 2)`, `MOD(3.1, 2)`
fetched rows / total rows = 1/1
+-------------+---------------+
| MOD(3, 2)   | MOD(3.1, 2)   |
|-------------+---------------|
| 1           | 1.1           |
+-------------+---------------+
```

PI

使用法: PI() は定数 pi を返します。

戻り値の型: DOUBLE

例:

```
os> source=people | eval `PI()` = PI() | fields `PI()`
fetched rows / total rows = 1/1
+-------------------+
| PI()              |
|-------------------|
| 3.141592653589793 |
+-------------------+
```

POW

使用法: POW(x, y) は x の y 乗の値を計算します。不正な入力は NULL 結果を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

シノニム: POWER(_, _)

例:

```
os> source=people | eval `POW(3, 2)` = POW(3, 2), `POW(-3, 2)` = POW(-3, 2), `POW(3,
 -2)` = POW(3, -2) | fields `POW(3, 2)`, `POW(-3, 2)`, `POW(3, -2)`
fetched rows / total rows = 1/1
+-------------+--------------+--------------------+
| POW(3, 2)   | POW(-3, 2)   | POW(3, -2)         |
|-------------+--------------+--------------------|
| 9.0         | 9.0          | 0.1111111111111111 |
+-------------+--------------+--------------------+
```

POWER

使用法: POWER(x, y) は x の y 乗の値を計算します。不正な入力は NULL 結果を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

シノニム: POW(_, _)

例:

```
os> source=people | eval `POWER(3, 2)` = POWER(3, 2), `POWER(-3, 2)` = POWER(-3, 2),
 `POWER(3, -2)` = POWER(3, -2) | fields `POWER(3, 2)`, `POWER(-3, 2)`, `POWER(3, -2)`
fetched rows / total rows = 1/1
+---------------+----------------+--------------------+
| POWER(3, 2)   | POWER(-3, 2)   | POWER(3, -2)       |
|---------------+----------------+--------------------|
| 9.0           | 9.0            | 0.1111111111111111 |
+---------------+----------------+--------------------+
```

RADIANS

使用法: RADIANS(x) は x を度からラジアンに変換します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `RADIANS(90)` = RADIANS(90) | fields `RADIANS(90)`
fetched rows / total rows  = 1/1
+--------------------+
| RADIANS(90)        |
|--------------------|
| 1.5707963267948966 |
+--------------------+
```

RAND

使用法: RAND( )/RAND(N) は、範囲 0 <= 値 < 1.0 のランダム浮動小数点値を返します。整数 N を指定すると、関数は実行前にシードを初期化します。この挙動の一つの意味は、同一の引数 N に対して rand(N) は毎回同じ値を返し、列値の繰り返し可能なシーケンスを生成するということです。

引数タイプ: INTEGER

戻り値の型: FLOAT

例:

```
os> source=people | eval `RAND(3)` = RAND(3) | fields `RAND(3)`
fetched rows / total rows = 1/1
+------------+
```

```
| RAND(3)    |
|------------|
| 0.73105735 |
+------------+
```

ROUND

使用法: ROUND(x, d) は引数 x を小数点以下 d 桁まで丸めます。d を指定しない場合、デフォルトは 0 です。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型マップ:

- (INTEGER/LONG [,INTEGER]) -> LONG
- (FLOAT/DOUBLE [,INTEGER]) -> LONG

例:

```
os> source=people | eval `ROUND(12.34)` = ROUND(12.34), `ROUND(12.34, 1)` =
 ROUND(12.34, 1), `ROUND(12.34, -1)` = ROUND(12.34, -1), `ROUND(12, 1)` = ROUND(12, 1)
 | fields `ROUND(12.34)`, `ROUND(12.34, 1)`, `ROUND(12.34, -1)`, `ROUND(12, 1)`
fetched rows / total rows = 1/1
+----------------+------------------+--------------------+----------------+
| ROUND(12.34)   | ROUND(12.34, 1)   | ROUND(12.34, -1)   | ROUND(12, 1)   |
|----------------+------------------+--------------------+----------------|
| 12.0           | 12.3              | 10.0               | 12             |
+----------------+------------------+--------------------+----------------+
```

SIGN

使用法: SIGN は、引数が負の場合は -1、ゼロの場合は 0、正の場合は 1 を返します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: INTEGER

例:

```
os> source=people | eval `SIGN(1)` = SIGN(1), `SIGN(0)` = SIGN(0), `SIGN(-1.1)` =
 SIGN(-1.1) | fields `SIGN(1)`, `SIGN(0)`, `SIGN(-1.1)`
fetched rows / total rows = 1/1
```

```
+-----------+-----------+--------------+
| SIGN(1)   | SIGN(0)   | SIGN(-1.1)   |
|-----------+-----------+--------------|
| 1         | 0         | -1           |
+-----------+-----------+--------------+
```

SIN

使用法: sin(x) は x のサインを計算します。x はラジアンで表されます。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型: DOUBLE

例:

```
os> source=people | eval `SIN(0)` = SIN(0) | fields `SIN(0)`
fetched rows / total rows = 1/1
+----------+
| SIN(0)   |
|----------|
| 0.0      |
+----------+
```

SQRT

使用法: SQRT は負でない数値の平方根を計算します。

引数タイプ: INTEGER/LONG/FLOAT/DOUBLE

戻り値の型マップ:

- (負以外) INTEGER/LONG/FLOAT/DOUBLE -> DOUBLE
- (負) INTEGER/LONG/FLOAT/DOUBLE -> NULL

例:

```
os> source=people | eval `SQRT(4)` = SQRT(4), `SQRT(4.41)` = SQRT(4.41) | fields
 `SQRT(4)`, `SQRT(4.41)`
fetched rows / total rows = 1/1
+-----------+--------------+
| SQRT(4)   | SQRT(4.41)   |
```

```
|-----------+--------------|
| 2.0       | 2.1          |
+-----------+--------------+
```

PPL 文字列関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

CONCAT

使用法: `CONCAT(str1, str2, ...., str_9)` は最大 9 つの文字列を合計します。

引数の型:

- STRING、STRING、....、STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `CONCAT('hello', 'world')` = CONCAT('hello', 'world'),
 `CONCAT('hello ', 'whole ', 'world', '!')` = CONCAT('hello ', 'whole ', 'world', '!')
 | fields `CONCAT('hello', 'world')`, `CONCAT('hello ', 'whole ', 'world', '!')`
fetched rows / total rows = 1/1
+----------------------------+---------------------------------------------+
| CONCAT('hello', 'world')   | CONCAT('hello ', 'whole ', 'world', '!')    |
|----------------------------+---------------------------------------------|
| helloworld                 | hello whole world!                          |
+----------------------------+---------------------------------------------+
```

CONCAT_WS

使用法: `CONCAT_WS(sep, str1, str2)` は指定された区切り文字を使用して 2 つ以上の文字列を連結します。

引数の型:

- STRING、STRING、....、STRING

- 戻り値の型: STRING

例:

```
os> source=people | eval `CONCAT_WS(',', 'hello', 'world')` = CONCAT_WS(',', 'hello',
 'world') | fields `CONCAT_WS(',', 'hello', 'world')`
fetched rows / total rows = 1/1
+------------------------------------+
| CONCAT_WS(',', 'hello', 'world')   |
|------------------------------------|
| hello,world                        |
+------------------------------------+
```

LENGTH

使用法: `length(str)` はバイト単位で測定された入力文字列の長さを返します。

引数の型:

- STRING
- 戻り値の型: INTEGER

例:

```
os> source=people | eval `LENGTH('helloworld')` = LENGTH('helloworld') | fields
 `LENGTH('helloworld')`
fetched rows / total rows = 1/1
+------------------------+
| LENGTH('helloworld')   |
|------------------------|
| 10                     |
+------------------------+
```

LOWER

使用法: `lower(string)` は入力文字列を小文字に変換します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `LOWER('helloworld')` = LOWER('helloworld'),
 `LOWER('HELLOWORLD')` = LOWER('HELLOWORLD') | fields `LOWER('helloworld')`,
 `LOWER('HELLOWORLD')`
fetched rows / total rows = 1/1
+----------------------+----------------------+
| LOWER('helloworld')   | LOWER('HELLOWORLD')   |
|----------------------+----------------------|
| helloworld            | helloworld            |
+----------------------+----------------------+
```

LTRIM

使用法: ltrim(str) は入力文字列から先頭のスペース文字を削除します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `LTRIM('   hello')` = LTRIM('   hello'), `LTRIM('hello   ')` =
 LTRIM('hello   ') | fields `LTRIM('   hello')`, `LTRIM('hello   ')`
fetched rows / total rows = 1/1
+---------------------+---------------------+
| LTRIM('   hello')   | LTRIM('hello   ')   |
|---------------------+---------------------|
| hello               | hello               |
+---------------------+---------------------+
```

POSITION

使用法: POSITION(substr IN str) は文字列内の部分文字列の最初の出現位置を返します。サブ文字列が文字列に含まれていない場合、0 を返します。引数のいずれかが NULL の場合、NULL を返します。

引数の型:

- STRING、STRING

- 戻り値の型 INTEGER

例:

```
os> source=people | eval `POSITION('world' IN 'helloworld')` = POSITION('world'
 IN 'helloworld'), `POSITION('invalid' IN 'helloworld')`= POSITION('invalid' IN
 'helloworld')  | fields `POSITION('world' IN 'helloworld')`, `POSITION('invalid' IN
 'helloworld')`
fetched rows / total rows = 1/1
+-------------------------------------+---------------------------------------+
| POSITION('world' IN 'helloworld')   | POSITION('invalid' IN 'helloworld')   |
|-------------------------------------+---------------------------------------|
| 6                                   | 0                                     |
+-------------------------------------+---------------------------------------+
```

REVERSE

使用法: REVERSE(str) は入力文字列の逆引き文字列を返します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `REVERSE('abcde')` = REVERSE('abcde') | fields
 `REVERSE('abcde')`
fetched rows / total rows = 1/1
+--------------------+
| REVERSE('abcde')   |
|--------------------|
| edcba              |
+--------------------+
```

RIGHT

使用法: right(str, len) は入力文字列から右端の文字を返します。サブ文字列が文字列に含まれていない場合、0 を返します。引数のいずれかが NULL の場合、NULL を返します。

引数の型:

- STRING、INTEGER
- 戻り値の型: STRING

例:

```
os> source=people | eval `RIGHT('helloworld', 5)` = RIGHT('helloworld', 5),
 `RIGHT('HELLOWORLD', 0)` = RIGHT('HELLOWORLD', 0) | fields `RIGHT('helloworld', 5)`,
 `RIGHT('HELLOWORLD', 0)`
fetched rows / total rows = 1/1
+--------------------------+--------------------------+
| RIGHT('helloworld', 5)   | RIGHT('HELLOWORLD', 0)   |
|--------------------------+--------------------------|
| world                    |                          |
+--------------------------+--------------------------+
```

RTRIM

使用法: `rtrim(str)` は入力文字列から末尾のスペース文字を削除します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `RTRIM('   hello')` = RTRIM('   hello'), `RTRIM('hello   ')` =
 RTRIM('hello   ') | fields `RTRIM('   hello')`, `RTRIM('hello   ')`
fetched rows / total rows = 1/1
+---------------------+---------------------+
| RTRIM('   hello')   | RTRIM('hello   ')   |
|---------------------+---------------------|
|    hello            | hello               |
+---------------------+---------------------+
```

SUBSTRING

使用法: `substring(str, start)` または `substring(str, start, length)` は、入力文字列の部分文字列を返します。長さを指定しない場合、開始位置から文字列全体を返します。

引数の型:

- STRING、INTEGER、INTEGER
- 戻り値の型: STRING

例:

```
os> source=people | eval `SUBSTRING('helloworld', 5)` = SUBSTRING('helloworld',
 5), `SUBSTRING('helloworld', 5, 3)` = SUBSTRING('helloworld', 5, 3) | fields
 `SUBSTRING('helloworld', 5)`, `SUBSTRING('helloworld', 5, 3)`
fetched rows / total rows = 1/1
+------------------------------+---------------------------------+
| SUBSTRING('helloworld', 5)   | SUBSTRING('helloworld', 5, 3)   |
|------------------------------+---------------------------------|
| oworld                       | owo                             |
+------------------------------+---------------------------------+
```

TRIM

使用法: `trim(string)` は入力文字列から先頭と末尾の空白を削除します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `TRIM('   hello')` = TRIM('   hello'), `TRIM('hello   ')` =
 TRIM('hello   ') | fields `TRIM('   hello')`, `TRIM('hello   ')`
fetched rows / total rows = 1/1
+--------------------+--------------------+
| TRIM('   hello')   | TRIM('hello   ')   |
|--------------------+--------------------|
| hello              | hello              |
+--------------------+--------------------+
```

UPPER

使用法: `upper(string)` は入力文字列を大文字に変換します。

引数の型:

- STRING
- 戻り値の型: STRING

例:

```
os> source=people | eval `UPPER('helloworld')` = UPPER('helloworld'),
 `UPPER('HELLOWORLD')` = UPPER('HELLOWORLD') | fields `UPPER('helloworld')`,
 `UPPER('HELLOWORLD')`
fetched rows / total rows = 1/1
+----------------------+----------------------+
| UPPER('helloworld')   | UPPER('HELLOWORLD')   |
|----------------------+----------------------|
| HELLOWORLD            | HELLOWORLD            |
+----------------------+----------------------+
```

PPL 型変換関数

> **Note**
>
> この PPL 関数をサポートする AWS データソース統合を確認するには、「」を参照してくださいthe section called “関数”。

`TRIM`

使用法: `cast(expr as dateType)` は `expr` を `dataType` にキャストし、`dataType` の値を返します。

次の変換ルールが適用されます。

型変換ルール

Src/Target	STRING	NUMBER	BOOLEAN	TIMESTAMP	DATE	TIME
STRING		Note1	Note1	TIMESTAMP()	DATE()	TIME()
NUMBER	Note1		v!=0	該当なし	該当なし	該当なし

Src/Target	STRING	NUMBER	BOOLEAN	TIMESTAMP	DATE	TIME
BOOLEAN	Note1	v?1:0		該当なし	該当なし	該当なし
TIMESTAMP	Note1	該当なし	該当なし		DATE()	TIME()
DATE	Note1	該当なし	該当なし	該当なし		該当なし
TIME	Note1	該当なし	該当なし	該当なし	該当なし	

文字列へのキャストの例:

```
os> source=people | eval `cbool` = CAST(true as string), `cint` = CAST(1 as string),
 `cdate` = CAST(CAST('2012-08-07' as date) as string) | fields `cbool`, `cint`, `cdate`
fetched rows / total rows = 1/1
+---------+--------+------------+
| cbool   | cint   | cdate      |
|---------+--------+------------|
| true    | 1      | 2012-08-07 |
+---------+--------+------------+
```

数字へのキャストの例:

```
os> source=people | eval `cbool` = CAST(true as int), `cstring` = CAST('1' as int) |
 fields `cbool`, `cstring`
fetched rows / total rows = 1/1
+---------+-----------+
| cbool   | cstring   |
|---------+-----------|
| 1       | 1         |
+---------+-----------+
```

日付へのキャストの例:

```
os> source=people | eval `cdate` = CAST('2012-08-07' as date), `ctime` =
 CAST('01:01:01' as time), `ctimestamp` = CAST('2012-08-07 01:01:01' as timestamp) |
 fields `cdate`, `ctime`, `ctimestamp`
fetched rows / total rows = 1/1
+------------+----------+---------------------+
| cdate      | ctime    | ctimestamp          |
|------------+----------+---------------------|
```

```
| 2012-08-07 | 01:01:01 | 2012-08-07 01:01:01 |
+------------+----------+---------------------+
```

連鎖キャストの例:

```
os> source=people | eval `cbool` = CAST(CAST(true as string) as boolean) | fields
 `cbool`
fetched rows / total rows = 1/1
+---------+
| cbool   |
|---------|
| True    |
+---------+
```

Amazon OpenSearch Service ドメインのモニタリング

モニタリングは、Amazon OpenSearch Service およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS には、OpenSearch Service リソースをモニタリングし、問題を報告し、必要に応じて自動アクションを実行するための以下のツールが用意されています。

Amazon CloudWatch

Amazon CloudWatch は OpenSearch Service リソースをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、およびメトリクスが特定のしきい値に達したときに通知したりアクションを実行したりするアラームの設定を行うことができます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

Amazon CloudWatch Logs

Amazon CloudWatch Logs を使用すると、OpenSearch のログファイルをモニタリング、保存、およびアクセスすることができます。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知できます。詳細については、Amazon CloudWatch Logs ユーザーガイドを参照してください。

Amazon EventBridge

Amazon EventBridge は、OpenSearch Service ドメインの変更を記述したシステムイベントのストリームをほぼリアルタイムに配信します。特定のイベントを監視するルールを作成し、これらのイベントが発生したときに他の AWS サービスで自動アクションをトリガーできます。詳細については、Amazon EventBridge ユーザーガイドを参照してください。

AWS CloudTrail

AWS CloudTrail は、OpenSearch Service に対して行われた設定 API コールをイベントとしてキャプチャします。指定した Amazon S3 バケットにこれらのイベントが渡されます。この情報を使用して、リクエストを行ったユーザーとアカウント、リクエスト元のソース IP アドレス、およびリクエストの発生日時を特定できます。詳細については、AWS CloudTrail ユーザーガイドを参照してください。

トピック

- Amazon CloudWatch を用いた OpenSearch クラスターメトリクスのモニタリング
- Amazon CloudWatch Logs を用いて OpenSearch ログをモニタリングする

- Amazon OpenSearch Service での監査ログのモニタリング
- Amazon EventBridge を用いた OpenSearch Service イベントのモニタリング
- を使用した Amazon OpenSearch Service API コールのモニタリング AWS CloudTrail

Amazon CloudWatch を用いた OpenSearch クラスターメトリクスのモニタリング

Amazon OpenSearch Service は、ドメインから Amazon CloudWatch にデータを公開します。CloudWatch では、それらのデータポイントについての統計 (メトリクスと呼ばれる) を、順序付けられた時系列データのセットとして取得できます。OpenSearch Service は、ほとんどのメトリクスを 60 秒間隔で CloudWatch に送信します。汎用または磁気 EBS ボリュームを使用する場合、EBS ボリュームのメトリクスは 5 分ごとのみに更新されます。すべての累積メトリクス (例: `ThreadpoolWriteRejected`、`ThreadpoolSearchRejected`) はメモリ内であり、状態が失われます。メトリクスは、ノードドロップ、ノードバウンス、ノード交換、ブルー/グリーンデプロイ中にリセットされます。Amazon CloudWatch の詳細については、Amazon CloudWatch ユーザーガイドを参照してください。

OpenSearch Service コンソールには、CloudWatch の raw データに基づいて一連のグラフが表示されます。必要に応じて、コンソールでのグラフではなく Amazon CloudWatch にクラスターデータを表示することもできます。サービスは、メトリクスを 2 週間アーカイブし、その後破棄します。このメトリクスに追加料金はかかりませんが、CloudWatch には引き続きダッシュボードとアラームの作成に対して課金されます。詳細については、「Amazon CloudWatch の料金」を参照してください。

OpenSearch Service は CloudWatch にメトリクスを発行します。

- the section called “クラスターメトリクス”
- the section called “専用マスターノードメトリクス”
- the section called “EBS ボリュームメトリクス”
- the section called “インスタンスメトリクス”
- the section called “ウォームメトリクス”
- the section called “専用コーディネーターノードメトリクス”
- the section called “コールドストレージのメトリクス”
- the section called “アラートメトリクス”
- the section called “異常検出のメトリクス”

- the section called “非同期検索メトリクス”
- the section called “SQL メトリクス”
- the section called “k-NN メトリクス”
- the section called “クラスター間検索のメトリクス”
- the section called “クラスター間レプリケーションメトリクス”
- the section called “Learning to Rank のメトリクス”
- the section called “Piped Processing Language のメトリクス”

CloudWatch でのメトリクスの表示

CloudWatch メトリクスはまずサービスの名前空間ごとにグループ化され、次に各名前空間内のさまざまなディメンションの組み合わせごとにグループ化されます。

CloudWatch コンソールを使用してメトリクスを表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左のナビゲーションペインで、[Metrics] (メトリクス) を見つけ、[All metrics] (すべてのメトリクス) を選択します。ES 名前空間を選択します。
3. ディメンションを選択して、対応するメトリクスを表示します。個別のノードのメトリクスは、`ClientId, DomainName, NodeId` ディメンションにあります。クラスターメトリクスは、`Per-Domain, Per-Client Metrics` ディメンションにあります。一部のノードメトリクスは、クラスターレベルで集計されるため、両方のディメンションに含まれます。シャードメトリクスは、`ClientId, DomainName, NodeId, ShardRole` ディメンションにあります。

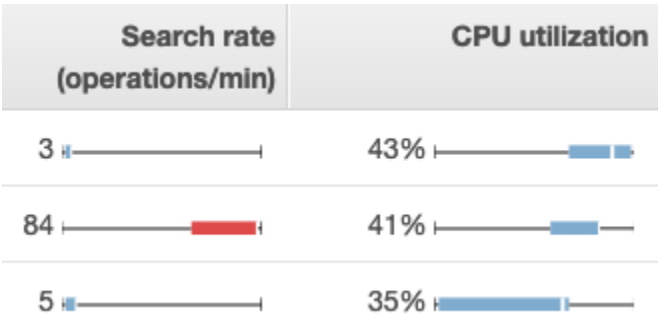
を使用してメトリクスのリストを表示するには AWS CLI

次のコマンドを実行します。

```
aws cloudwatch list-metrics --namespace "AWS/ES"
```

OpenSearch Service でのヘルスチャートの解釈

OpenSearch Service でメトリクスを表示するには、[クラスターヘルス] および [インスタンスヘルス] タブを使用します [インスタンスヘルス] タブでは、ボックスチャートを使用して、各 OpenSearch ノードのヘルスを一目で確認できます。



- それぞれの色付きの箱は、指定した期間におけるノードの値範囲を示しています。
- 青色の箱は他のノードと同じ値を示します。赤色の箱は異常値を示します。
- 各箱の白線は、ノードの現在値を示します。
- 各拍の両脇にある「ひげ」は、期間におけるすべてのノードの最大値と最小値を示します。

ドメインの設定を変更すると、[クラスターヘルス] タブと [インスタンスヘルス] タブの各インスタンスのリストが正しい数に戻る前に、しばらくの間 2 倍のサイズになる場合があります。この動作の説明については、「the section called “設定変更”」を参照してください。

クラスターメトリクス

Amazon OpenSearch Service は、以下のクラスターのメトリクスを提供します。

メトリクス	説明
`ClusterStatus.green`	値 1 は、すべてのインデックスシャードがクラスターのノードに割り当てられることを示します。 関連する統計: Maximum
`ClusterStatus.yellow`	値 1 は、すべてのインデックスのプライマリシャードがクラスターのノードに割り当てられていることを示しますが、1 つ以上のインデックスのレプリカシャードが割り当てられていません。詳細については、「the section called “黄色のクラスター状態”」を参照してください。 関連する統計: Maximum

メトリクス	説明
ClusterStatus.red	値 1 は、少なくとも 1 つのインデックスのプライマリとレプリカの両方のシャードが、クラスターのノードに割り当てられないことを示します。詳細については、「the section called “赤のクラスター状態”」を参照してください。 関連する統計: Maximum
Shards.active	アクティブなプライマリとレプリカの両方のシャードの合計数。 関連する統計: Maximum、Sum
Shards.unassigned	クラスターのノードに割り当てられていないシャードの数。 関連する統計: Maximum、Sum
Shards.delayedUnassigned	タイムアウト設定によってノード割り当てが遅れたシャードの数。 関連する統計: Maximum、Sum
Shards.activePrimary	アクティブなプライマリシャードの数。 関連する統計: Maximum、Sum
Shards.initializing	初期化中のシャードの数。 関連する統計情報: Sum
Shards.relocating	再配置中のシャードの数。 関連する統計情報: Sum
Nodes	専用マスターノードとウォームノードを含む、OpenSearch Service クラスター内のノードの数。詳細については、「the section called “設定変更”」を参照してください。 関連する統計: Maximum

メトリクス	説明
SearchableDocuments	クラスター内のすべてのデータノードで検索可能なドキュメントの合計数。 関連する統計情報: Minimum、Maximum、Average
DeletedDocuments	クラスター内のすべてのデータノードで削除対象としてマークされたドキュメントの合計数。これらのドキュメントは検索結果に表示されなくなっていますが、セグメントマージ中にディスクから削除されたドキュメントのみが OpenSearch によって削除されます。このメトリクスは、削除リクエスト後に増加し、セグメントマージ後に減少します。 関連する統計情報: Minimum、Maximum、Average
CPUUtilization	クラスター内のデータノードの CPU 使用率の割合。Maximum は、CPU 使用率が最も高いノードを示します。Average は、クラスター内のすべてのノードを表します。このメトリクスは、個別のノードでも利用できます。 関連する統計: Maximum、Average

メトリクス	説明
FreeStorageSpace	クラスター内のデータノードの空き領域。Sum はクラスターの合計空き容量を示しますが、正確な値を得るには期間を 1 分にする必要があります。Minimum と Maximum は、それぞれ空き領域が最も少ないノードと最も多いノードを示します。このメトリクスは、個別のノードでも利用できます。OpenSearch Service は、このメトリクスが 0 に達したときに ClusterBlockException をスローします。復旧するには、インデックスを削除する、より大きなインスタンスを追加する、既存のインスタンスに EBS ベースのストレージを追加する、のいずれかを実行する必要があります。詳細についてはthe section called “使用可能なストレージ領域の不足”を参照してください。 OpenSearch Service コンソールには、この値が GiB 単位で表示されます。Amazon CloudWatch コンソールには、この値が MiB 単位で表示されます。 **Note** FreeStorageSpace は、OpenSearch の _cluster/stats および _cat/allocation API が提供する値より常に小さくなります。OpenSearch Service は、内部オペレーションの各インスタンスでストレージ領域の割合を予約します。詳細については、「ストレージ要件の計算」を参照してください。 関連する統計: Minimum、Maximum、Average、Sum
ClusterUsedSpace	クラスターの合計使用領域。正確な値を取得するには、期間を 1 分のままにしておく必要があります。 OpenSearch Service コンソールには、この値が GiB 単位で表示されます。Amazon CloudWatch コンソールには、この値が MiB 単位で表示されます。 関連する統計: Minimum、Maximum

メトリクス	説明
ClusterIndexWritesBlocked	クラスターで、着信する書き込みリクエストを受け入れるか、ブロックするかを指定します。値 0 では、クラスターでリクエストを受け入れます。値 1 ではリクエストをブロックします。 代表的なものとしては、FreeStorageSpace が少なすぎる、JVMMemoryPressure が高すぎるなどがあります。この問題を軽減するには、ディスク容量の追加やクラスターのスケーリングを検討します。 関連する統計: Maximum
JVMMemoryPressure	クラスター内のすべてのデータノードで使用する Java ヒープの最大パーセンテージ。OpenSearch Service は、インスタンスの RAM の半分を Java ヒープ (32 GiB のヒープサイズまで) に使用します。インスタンスは最大 64 GiB の RAM まで垂直スケーリングでき、それ以上はインスタンスを追加することで水平方向にスケーリングできます。「the section called “推奨される CloudWatch アラーム”」を参照してください。 関連する統計: Maximum **Note** このメトリクスのロジックは、サービスソフトウェア R20220323 で変更されました。詳細については、「release notes」を参照してください。
OldGenJVMMemoryPressure	クラスター内のすべてのデータノードにおける「旧世代」で使用する Java ヒープの最大パーセンテージ。このメトリクスは、ノードレベルでも使用できます。 関連する統計: Maximum

メトリクス	説明
AutomatedSnapshotFailure	クラスターの失敗した自動スナップショットの数。1 の値は、自動スナップショットが過去 36 時間、ドメイン用に取られなかったことを示します。 関連する統計: Minimum、Maximum
CPUCreditBalance	クラスター内の、データノードに使用できる残りの CPU クレジット。CPU クレジットは、フル CPU パフォーマンスを 1 分間実現します。詳細については、Amazon EC2 デベロッパーガイドの「CPU クレジット」を参照してください。このメトリクスは、T2 のインスタンスタイプでのみ使用できます。 関連する統計: Minimum
OpenSearchDashboardsHealthyNodes	OpenSearch Dashboards のヘルスチェック。最小、最大、および平均がすべて 1 に等しい場合、Dashboards は正常に動作しています。最大が 1、最小が 0、平均が 0.7 の 10 個のノードがある場合、これは 7 個のノード (70%) が正常であり、3 個のノード (30%) が正常でないことを意味します。 関連する統計情報: Minimum、Maximum、Average
OpensearchDashboardsReportingFailedRequestSysErrCount	サーバーの問題または機能の制限により失敗した OpenSearch Dashboards レポートを生成するリクエストの数。 関連する統計情報: Sum
OpensearchDashboardsReportingFailedRequestUserErrCount	クライアントの問題により失敗した OpenSearch Dashboards レポートを生成するリクエストの数。 関連する統計情報: Sum
OpensearchDashboardsReportingRequestCount	OpenSearch Dashboards レポートを生成するリクエストの合計数。 関連する統計情報: Sum

メトリクス	説明
OpensearchDashboardsReportingSuccessCount	OpenSearch Dashboards レポートを生成するリクエストに成功した数。 関連する統計情報: Sum
KMSKeyError	値 1 は、保管中のデータの暗号化に使用される AWS KMS キーが無効になっていることを示します。通常のオペレーションにドメインを復元するには、キーを再度有効にします。コンソールでは、保管時のデータを暗号化するドメインに対してのみこのメトリクスが表示されます。 関連する統計: Minimum、Maximum
KMSKeyInaccessible	値 1 は、保管中のデータの暗号化に使用される AWS KMS キーが削除されたか、OpenSearch Service への許可が取り消されたことを示します。この状態にあるドメインを復元することはできません。ただし、手動のスナップショットがある場合は、それを使用してドメインのデータを新しいドメインに移行できます。コンソールでは、保管時のデータを暗号化するドメインに対してのみこのメトリクスが表示されます。 関連する統計: Minimum、Maximum
InvalidHostHeaderRequests	OpenSearch クラスターに対する HTTP リクエストのうち、ホストヘッダーが無効な (または欠落している) ものの数。有効なリクエストには、ホストヘッダー値としてドメインホスト名が含まれます。OpenSearch Service は、制限付きアクセスポリシーを持たないパブリックアクセスドメインの不正なリクエストを拒否します。すべてのドメインに制限付きアクセスポリシーを適用することをお勧めします。 このメトリクスで大きな値が表示される場合は、OpenSearch クライアントのリクエストにドメインホスト名が含まれていることを確認します (ない場合は、IP アドレスなど)。 関連する統計情報: Sum

メトリクス	説明
OpenSearchRequests (previously ElasticsearchRequests)	OpenSearch クラスターに対するリクエストの数。 関連する統計情報: Sum
TLSNegotiationError	クライアントとドメインエンドポイントの間で失敗した TLS ハンドシェイク試行の数。このメトリクスは、クライアントがサポートされていない TLS バージョンまたは暗号スイートを使用して接続しようとすると増加します。 関連する統計情報: Sum
2xx, 3xx, 4xx, 5xx	特定の HTTP レスポンスコード (2xx、3xx、4xx、5xx) の発生につながった、ドメインへのリクエストの数。 関連する統計情報: Sum
ThroughputThrottle	ディスクがスロットリングされたかどうかが示されます。スロットリングは、ReadThroughputMicroBursting と WriteThroughputMicroBursting の合計スループットが最大スループット MaxProvisionedThroughput よりも高い場合に発生します。MaxProvisionedThroughput は、インスタンススループットまたはプロビジョニングされたボリュームスループットのうち、低い方の値です。値 1 は、ディスクがスロットリングされていることを示しています。値 0 は正常な動作を示します。 インスタンスのスループットについては、「Amazon EBS 最適化インスタンスを使用する」を参照してください。ボリュームスループットの詳細については、「Amazon EBS ボリュームの種類」を参照してください。 関連する統計: Minimum、Maximum

メトリクス	説明
IopsThrottle	ドメインの 1 秒あたりの入出力オペレーション (IOPS) の数がスロットリングされているかどうかを示します。スロットリングは、データノードの IOPS が、EBS ボリュームまたはデータノードの EC2 インスタンスの最大許容制限を超えた場合に発生します。 インスタンスの IOPS については、「Amazon EBS–optimized instances」を参照してください。ボリュームの IOPS については、「Amazon EBS volume types」を参照してください。 関連する統計: Minimum、Maximum
HighSwapUsage	1 の値は、ページ障害によるスワップが、特定の期間中に基盤となるディスク使用量の急増を引き起こす可能性があることを示します。 関連する統計: Maximum

専用マスターノードメトリクス

Amazon OpenSearch Service は、以下の専用マスターノードのメトリクスを提供します。

メトリクス	説明
MasterCPUUtilization	専用マスターノードが使用する CPU リソースの最大パーセンテージ。このメトリクスが 60 パーセントに達する場合、インスタンスタイプのサイズを増やすことをお勧めします。 関連する統計: Maximum
MasterFreeStorageSpace	このメトリクスは関係ないため無視できます。このサービスはデータノードとしてマスターノードを使用しません。
MasterJVMMemoryPressure	クラスター内のすべての専用マスターノードで使用する Java ヒープの最大パーセンテージ。このメトリクスが 85 パーセントに達す

メトリクス	説明
	る場合、より大規模なインスタンスタイプに移行することをお勧めします。 関連する統計: Maximum Note このメトリクスのロジックは、サービスソフトウェア R20220323 で変更されました。詳細については、「release notes」を参照してください。
MasterOldGenJVMMemoryPressure	マスターノードごとの「旧世代」で使用される Java ヒープの最大パーセンテージ。 関連する統計: Maximum
MasterCPUCreditBalance	クラスター内の専用マスターノードで使用できる、残りの CPU クレジット。CPU クレジットは、フル CPU パフォーマンスを 1 分間実現します。詳細については、Amazon EC2 デベロッパーガイドの「CPU クレジット」を参照してください。このメトリクスは、T2 のインスタンスタイプでのみ使用できます。 関連する統計: Minimum
MasterReachableFromNode	MasterNotDiscovered 例外のヘルスチェック。値 1 は正常な動作を示します。値 0 は、/_cluster/health/ の動作が正常ではないことを示します。 障害が発生すると、ソースノードからマスターノードにアクセスすることができなくなります。通常、ネットワーク接続の問題や AWS 依存関係の問題が原因です。 関連する統計: Maximum
MasterSysMemoryUtilization	使用中のマスターノードのメモリの割合。 関連する統計: Maximum

専用コーディネーターノードメトリクス

Amazon OpenSearch Service は、以下の専用コーディネーターノードのメトリクスを提供します。

メトリクス	説明
CoordinatorCPUUtilization	専用コーディネーターノードが使用する CPU リソースの最大パーセンテージ。このメトリクスが 80 パーセントに達する場合、インスタンスタイプのサイズを増やすことをお勧めします。 関連する統計: Maximum
CoordinatorJVMMemoryPressure	クラスター内のすべての専用コーディネーターノードで使用する Java ヒープの最大パーセンテージ。このメトリクスが 85 パーセントに達する場合、より大規模なインスタンスタイプに移行することをお勧めします。 関連する統計: Maximum
CoordinatorOldGenJVMMemoryPressure	マスターノードごとの「旧世代」で使用される Java ヒープの最大パーセンテージ。 関連する統計: Maximum
CoordinatorSysMemoryUtilization	使用中のコーディネーターノードのメモリの割合。 関連する統計: Maximum
CoordinatorFreeStorageSpace	このメトリクスは、サービスがコーディネーターノードをデータノードとして使用していないことを示します。

EBS ボリュームメトリクス

Amazon OpenSearch Service は、以下の EBS ボリュームのメトリクスを提供します。

メトリクス	説明
ReadLatency	EBS ボリュームでの読み取り操作のレイテンシー (秒単位)。このメトリクスは、個別のノードでも利用できます。

メトリクス	説明
	関連する統計情報: Minimum、Maximum、Average
WriteLatency	EBS ボリュームでの書き込み操作のレイテンシー (秒単位)。このメトリクスは、個別のノードでも利用できます。 関連する統計情報: Minimum、Maximum、Average
ReadThroughput	EBS ボリュームでの読み取り操作のスループット (バイト/秒単位)。このメトリクスは、個別のノードでも利用できます。 関連する統計情報: Minimum、Maximum、Average
ReadThroughputMicroBursting	マイクロバーストを考慮に入れたときの、EBS ボリュームにおける読み取りオペレーションのスループット (バイト/秒)。このメトリクスは、個別のノードでも利用できます。マイクロバーストは、EBS ボリュームがきわめて短い時間 (1 分未満) に高い IOPS またはスループットをバーストするときに発生します。 関連する統計情報: Minimum、Maximum、Average
WriteThroughput	EBS ボリュームでの書き込み操作のスループット (バイト/秒単位)。このメトリクスは、個別のノードでも利用できます。 関連する統計情報: Minimum、Maximum、Average
WriteThroughputMicroBursting	マイクロバーストを考慮に入れたときの、EBS ボリュームにおける書き込みオペレーションのスループット (バイト/秒)。このメトリクスは、個別のノードでも利用できます。マイクロバーストは、EBS ボリュームがきわめて短い時間 (1 分未満) に高い IOPS またはスループットをバーストするときに発生します。 関連する統計情報: Minimum、Maximum、Average
DiskQueueDepth	EBS ボリュームに対する保留中の入出力 (I/O) リクエストの数。 関連する統計情報: Minimum、Maximum、Average

メトリクス	説明
ReadIOPS	EBS ボリュームでの読み取り操作の入出力 (I/O) 操作数 (1 秒あたり)。このメトリクスは、個別のノードでも利用できます。 関連する統計情報: Minimum、Maximum、Average
ReadIOPSMicroBursting	マイクロバーストを考慮に入れたときの、EBS ボリュームでの読み取り操作の入出力 (I/O) 操作数 (1 秒あたり)。このメトリクスは、個別のノードでも利用できます。マイクロバーストは、EBS ボリュームがきわめて短い時間 (1 分未満) に高い IOPS またはスループットをバーストするときに発生します。 関連する統計情報: Minimum、Maximum、Average
WriteIOPS	EBS ボリュームでの書き込み操作の入出力 (I/O) 操作数 (1 秒あたり)。このメトリクスは、個別のノードでも利用できます。 関連する統計情報: Minimum、Maximum、Average
WriteIOPSMicroBursting	マイクロバーストを考慮に入れたときの、EBS ボリュームでの書き込み操作の入出力 (I/O) 操作数 (1 秒あたり)。このメトリクスは、個別のノードでも利用できます。マイクロバーストは、EBS ボリュームがきわめて短い時間 (1 分未満) に高い IOPS またはスループットをバーストするときに発生します。 関連する統計情報: Minimum、Maximum、Average
BurstBalance	EBS ボリュームの、バーストバケットに残っている入出力 (I/O) クレジットの割合。値 100 は、ボリュームが最大クレジット数を累積したことを意味します。このパーセンテージが 70% を下回る場合は、「the section called “低 EBS バーストバランス”」を参照してください。gp3 ボリュームタイプを使用するドメインと、ボリュームサイズが 1,000 GiB を超える gp2 ボリュームを使用するドメインのバーストバランスは 0 のままになります。 関連する統計情報: Minimum、Maximum、Average

メトリクス	説明
VolumeStalledIOcheck	障害が発生した時期を判断するための EBS ボリュームのステータス。このメトリクスは、EBS ボリュームが入出力操作を完了できるかどうかに基づいて、0 (合格) または 1 (失敗) のステータスを返すバイナリ値です。VolumeStalledIOcheck は個々のノードでも使用できます。 関連する統計情報: Minimum、Maximum、Average

インスタンスメトリクス

Amazon OpenSearch Service は、以下のドメイン内の各インスタンスのメトリクスを提供します。また、OpenSearch Service では、これらのインスタンスメトリクスを集約することで、クラスター全体のヘルスへのインサイトを提供します。この動作を確認するには、コンソールで [サンプル数] 統計を使用します。以下のテーブルの各メトリクスには、ノードとクラスターに関連する統計を含みます。

> **Important**
>
> Elasticsearch のバージョンが異なる場合、_index API への呼び出しの処理にも異なるスレッドプールが使用されます。Elasticsearch 1.5 および 2.3 は、インデックス作成スレッドプールを使用します。Elasticsearch 5.x、6.0、および 6.2 はバルクスレッドプールを使用します。OpenSearch および Elasticsearch 6.3 以降では書き込みスレッドプールを使用します。現在、OpenSearch Service コンソールにはバルクスレッドプールのグラフは含まれていません。
> GET _cluster/settings?include_defaults=true を使用して、クラスターのスレッドプールとキューサイズを確認します。

メトリクス	説明
FetchLatency	ノード内のすべてのシャードフェッチ操作にかかった合計時間 (ミリ秒) の、N 分と (N - 1) 分の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum

メトリクス	説明
FetchRate	データノード上のすべてのシャードに対する 1 分あたりのシャードフェッチ操作の合計数。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum
ScrollTotal	データノード上のすべてのシャードに対する 1 分あたりのシャードスクロール操作の合計数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum
ScrollCurrent	現在実行中のシャードスクロールオペレーションの数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum
OpenContexts	オープン検索コンテキストの数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum
ThreadCount	OpenSearch プロセスで現在使用されているスレッドの合計数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum
ShardReactivateCount	すべてのシャードがアイドル状態からアクティブ化された合計回数。 関連するノードの統計: Sum、Maximum 関連するクラスターの統計: Sum、Maximum

メトリクス	説明
ConcurrentSearchRate	データノードのすべてのシャードに対する 1 分あたりの同時セグメント検索を使用した検索リクエストの総数。_search API への 1 回の呼び出しに対して、さまざまなシャードから結果が返される可能性があります。これらのシャードのうちの 5 つが 1 つのノードにある場合、クライアントが 1 つのリクエストしか行っていない場合でも、ノードはこのメトリクスについて 5 を報告します。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum
ConcurrentSearchLatency	ノード内で同時セグメント検索を使用したすべての検索にかかった合計時間 (ミリ秒) の、N 分と (N-1) 分の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum
IndexingLatency	ノード内のすべてのインデックス作成オペレーションにかかった合計時間 (ミリ秒) の、N 分と (N-1) 分の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum

メトリクス	説明
IndexingRate	1 分あたりのインデックス作成オペレーションの数。2 つのドキュメントを追加し、2 つのカウントを 4 つのオペレーションとして更新する _bulk API への 1 回の呼び出し。これは 1 つ以上のノードに分散する可能性があります。そのインデックスに 1 つ以上のレプリカがあり、最適化インスタンスが使用されない OpenSearch ドメインにある場合、クラスター内の他のノードにも合計 4 つのインデックス作成オペレーションが記録されます。インスタンスが最適化された OpenSearch ドメインの場合、レプリカを持つ他のノードはオペレーションを記録しません。ドキュメントの削除はこのメトリクスに対してカウントされません。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum
SearchLatency	ノード内のすべての検索にかかった合計時間 (ミリ秒) の、N 分と (N-1) 分の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum
SearchRate	データノードのすべてのシャードに対する 1 分あたりの検索リクエストの総数。_search API への 1 回の呼び出しに対して、さまざまなシャードから結果が返される可能性があります。これらのシャードのうちの 5 つが 1 つのノードにある場合、クライアントが 1 つのリクエストしか行っていない場合でも、ノードはこのメトリクスについて 5 を報告します。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum

メトリクス	説明
SegmentCount	データノードでのセグメントの数。セグメントの数が多いほど、各検索にかかる時間が長くなります。OpenSearch では、小さなセグメントを大きなセグメントにマージすることがあります。 関連するノードの統計: Maximum、Average 関連するクラスターの統計: Sum、Maximum、Average
SysMemoryUtilization	インスタンスが使用中のメモリの割合。このメトリクスの高い値は正常であり、通常はクラスターに問題はありません。潜在的なパフォーマンスおよび安定性の問題の指標については、「JVMMemoryPressure メトリクス」を参照してください。 関連するノードの統計: Minimum、Maximum、Average 関連するクラスターの統計: Minimum、Maximum、Average
JVMGCYoungCollectionCount	「新世代」ガベージコレクションが実行された回数。実行数が大量になり、かつ増え続けることは、通常のクラスター操作の一部です。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
JVMGCYoungCollectionTime	クラスターで「新世代」ガベージコレクションの実行にかかった時間 (ミリ秒)。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
JVMGCOldCollectionCount	「旧世代」ガベージコレクションが実行された回数。十分なリソースがあるクラスターでは、この回数は少ないままですが、まれに増加します。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average

メトリクス	説明
JVMGCOldCollectionTime	クラスターで「旧世代」ガベージコレクションの実行にかかった時間 (ミリ秒)。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
OpenSearchDashboardsConcurrentConnections	OpenSearch Dashboards へのアクティブな同時接続の数。この数が一貫して増加する場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
OpenSearchDashboardsHealthyNode	個別の OpenSearch Dashboards ノードのヘルスチェック。値 1 は正常な動作を示します。値 0 は Dashboards がアクセス不可であることを示します。 関連するノードの統計: Minimum 関連するクラスターの統計: Minimum、Maximum、Average
OpenSearchDashboardsHeapTotal	OpenSearch Dashboards に割り当てられたヒープメモリの容量 (MiB)。EC2 インスタンスタイプが異なると、正確なメモリ割り当てに影響する可能性があります。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
OpenSearchDashboardsHeapUsed	OpenSearch Dashboards で使用されるヒープメモリの絶対容量 (MiB)。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average

メトリクス	説明
OpenSearchDashboardsHeapUtilization	OpenSearch Dashboards で使用される使用可能なヒープメモリの最大パーセンテージ。この値が 80% を超えて増加する場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Minimum、Maximum、Average
OpenSearchDashboardsOS1MinuteLoad	OpenSearch Dashboards の 1 分の CPU ロードの平均。CPU ロードは、理想的には 1.00 未満にとどまるはずです。一時的なスパイクは問題ありませんが、このメトリクスが一貫して 1.00 を超える場合は、インスタンスタイプのサイズを増やすことをお勧めします。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum
OpenSearchDashboardsRequestTotal	OpenSearch Dashboards に対して行われた HTTP リクエストの合計数。システムの速度が遅い、または Dashboards リクエストの数が多い場合は、インスタンスタイプのサイズを増やすことを検討してください。 関連するノードの統計: Sum 関連するクラスターの統計: Sum
OpenSearchDashboardsResponseTimesMaxInMillis	OpenSearch Dashboards がリクエストに応答するのに必要な最大時間 (ミリ秒)。リクエストで結果が返ってくるために一貫して時間がかかる場合は、インスタンスタイプのサイズを増やすことを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Maximum、Average

メトリクス	説明
SearchTaskCancelled	コーディネーターノードのキャンセル数。 関連するノードの統計: Sum 関連するクラスターの統計: Sum
SearchShardTaskCancelled	データノードのキャンセル数。 関連するノードの統計: Sum 関連するクラスターの統計: Sum、
ThreadpoolForce_mergeQueue	強制マージスレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
ThreadpoolForce_mergeRejected	強制マージスレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
ThreadpoolForce_mergeThreads	強制マージスレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum

メトリクス	説明
ThreadpoolIndexQueue	インデックス作成スレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。インデックスキューの最大サイズは 200 です。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
ThreadpoolIndexRejected	インデックス作成スレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
ThreadpoolIndexThreads	インデックス作成スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
ThreadpoolSearchQueue	検索スレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。検索キューの最大サイズは 1,000 です。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
ThreadpoolSearchRejected	検索スレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum

メトリクス	説明
ThreadpoolSearchThreads	検索スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
Threadpoolsql-workerQueue	SQL 検索スレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
Threadpoolsql-workerRejected	SQL 検索スレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
Threadpoolsql-workerThreads	SQL 検索スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
ThreadpoolBulkQueue	バルクスレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average

メトリクス	説明
ThreadpoolBulkRejected	バルクスレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
ThreadpoolBulkThreads	バルクスレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
ThreadpoolIndexSearcherQueue	インデックス検索スレッドプールでキューに入っているタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
ThreadpoolIndexSearcherRejected	インデックス検索スレッドプールで拒否されたタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
ThreadpoolIndexSearcherThreads	インデックス検索スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
ThreadpoolWriteThreads	書き込みスレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum

メトリクス	説明
ThreadpoolWriteQueue	書き込みスレッドプールでキューに入っているタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
ThreadpoolWriteRejected	書き込みスレッドプールで拒否されたタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum **Note** バージョン 7.1 では、デフォルトの書き込みキューのサイズが 200 から 10000 に増加したため、このメトリクスは OpenSearch Service からの拒否の唯一の指標ではなくなりました。CoordinatingWriteRejected 、PrimaryWriteRejected 、および ReplicaWriteRejected メトリクスを使用して、バージョン 7.1 以降での拒否をモニタリングします。
CoordinatingWriteRejected	最後の OpenSearch Service プロセスのスタートアップ以降のインデックス作成のプレッシャーにより調整ノードで発生した拒否の総数。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum このメトリクスは、バージョン 7.1 以降で使用できます。

メトリクス	説明
PrimaryWriteRejected	最後の OpenSearch Service プロセスのスタートアップ以降のインデックス作成のプレッシャーによりプライマリシャードで発生した拒否の総数。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum このメトリクスは、バージョン 7.1 以降で使用できます。
ReplicaWriteRejected	最後の OpenSearch Service プロセスのスタートアップ以降のインデックス作成のプレッシャーによりレプリカシャードで発生した拒否の合計数。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum このメトリクスは、バージョン 7.1 以降で使用できます。
WorkloadManagementEnabled	ワークロード管理機能が有効になっているかどうかを示します。値 1 は有効、値 0 は `monitor_only` または無効を意味します。 関連するノードの統計: Maximum、Minimum 関連するクラスターの統計: Average、Sum このメトリクスは、バージョン 7.1 以降で使用できます。
SoftQueryGroupCount	ドメインでソフトモードになっているクエリグループの数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum このメトリクスは、バージョン 7.1 以降で使用できます。

メトリクス	説明
EnforcedQueryGroupCount	ドメインで強制モードになっているクエリグループの数。 関連するノードの統計: Average、Maximum 関連するクラスターの統計: Average、Maximum、Sum このメトリクスは、バージョン 7.1 以降で使用できます。

ウォームメトリクス

Amazon OpenSearch Service は、多層ストレージアーキテクチャと UltraWarm に次のメトリクスを提供します。

> **Note**
> ウォームインデックス作成関連のメトリクスは、多層ストレージアーキテクチャにのみ適用されます。

メトリクス	説明
WarmIndexingLatency	N 分から 1 分 (N-1) の間のウォームノード内のすべてのインデックス作成オペレーションによって取得された合計時間のミリ秒単位の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum
WarmIndexingRate	1 分あたりのウォームに対するインデックス作成オペレーションの数。2 つのドキュメントを追加し、2 つのカウントを 4 つのオペレーションとして更新する _bulk API への 1 回の呼び出し。これは 1 つ以上のノードに分散する可能性があります。そのインデックスに 1 つ以上のレプリカがあり、最適化インスタンスが使用されない OpenSearch ドメインにある場合、クラスター内の他のノードにも合計 4 つのインデックス作成オペレーションが記録されます。インスタンスが最適化された OpenSearch ドメインの場合、レプリカを持つ他のノードはオペレー

メトリクス	説明
	ションを記録しません。ドキュメントの削除はこのメトリクスに対してカウントされません。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum
WarmThreadpoolIndexingQueue	インデックス作成スレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。インデックスキューの最大サイズは 200 です。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Maximum、Sum
WarmThreadpoolIndexingRejected	インデックス作成スレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
WarmThreadpoolIndexingThreads	インデックス作成スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
WarmCPUUtilization	クラスター内のウォームノードの CPU 使用率。Maximum は、CPU 使用率が最も高いノードを示します。Average は、クラスター内のすべてのウォームノードを表します。このメトリクスは、個々のウォームノードでも使用できます。 関連する統計: Maximum、Average

メトリクス	説明
WarmFreeStorageSpace	ウォームストレージの空き容量 (MiB)。Warm はアタッチされたディスクではなく Amazon S3 を使用するため、関連する統計Sumは のみです。正確な値を取得するには、期間を 1 分のままにしておく必要があります。 関連する統計情報: Sum
WarmSearchableDocuments	クラスター内のすべてのウォームインデックスで検索可能なドキュメントの合計数。正確な値を取得するには、期間を 1 分のままにしておく必要があります。 関連する統計情報: Sum
WarmSearchLatency	ウォームでの N 分から 1 分 (N-1) までのすべての検索で取得された合計時間のミリ秒単位の差。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum
WarmSearchRate	Warm ノード上のすべてのシャードに対する 1 分あたりの検索リクエストの合計数。_search API への 1 回の呼び出しに対して、さまざまなシャードから結果が返される可能性があります。これらのシャードのうちの 5 つが 1 つのノードにある場合、クライアントが 1 つのリクエストしか行っていない場合でも、ノードはこのメトリクスについて 5 を報告します。 関連するノードの統計: Average 関連するクラスターの統計: Average、Maximum、Sum
WarmStorageSpaceUtilization	クラスターで使用中のウォームストレージスペースの合計容量 (MiB)。 関連する統計: Maximum
HotStorageSpaceUtilization	クラスターで使用しているホットストレージの合計容量。 関連する統計: Maximum

メトリクス	説明
WarmSysMemoryUtilization	使用中のウォームノードのメモリの割合。 関連する統計: Maximum
HotToWarmMigrationQueueSize	現在、ホットストレージからウォームトストレージへの移行を待機しているインデックスの数。 関連する統計: Maximum
WarmToHotMigrationQueueSize	現在、ウォームストレージからホットストレージへの移行を待機しているインデックスの数。 関連する統計: Maximum
HotToWarmMigrationFailureCount	失敗したホットからウォームへの移行の合計数。このメトリクスは UltraWarm ノードでのみ使用できます。 関連する統計情報: Sum
HotToWarmMigrationForceMergeLatency	移行プロセスの強制マージステージの平均レイテンシー。このステージに一貫して時間がかかりすぎる場合は、 を増やすことを検討してください。このメトリクスは UltraWarm ノードでのみ使用できます。index.ultrawarm.migration.force_merge.max_num_segments 関連する統計: Average
HotToWarmMigrationSnapshotLatency	移行プロセスのスナップショットステージの平均レイテンシー。この段階が一貫して時間がかかりすぎる場合は、シャードが適切にサイズ設定され、クラスター全体に分散されていることを確認します。このメトリクスは UltraWarm ノードでのみ使用できます。 関連する統計: Average

メトリクス	説明
HotToWarmMigrationProcessingLatency	ホットからウォームへの移行が成功した場合の平均レイテンシーで、キューに費やした時間を含まない。この値は、移行プロセスの強制マージ、スナップショット、およびシャード再配置ステージを完了するのにかかる時間の合計です。このメトリクスは UltraWarm ノードでのみ使用できます。 関連する統計: Average
HotToWarmMigrationSuccessCount	ホットからウォームへの移行に成功した合計数。 関連する統計情報: Sum
HotToWarmMigrationSuccessLatency	ホットからウォームへの移行が成功した場合の平均レイテンシーで、キューに費やされた時間を含む。 関連する統計: Average
WarmThreadpoolSearchThreads	ウォーム検索スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Average、Sum
WarmThreadpoolSearchRejected	ウォーム検索スレッドプールで拒否されたタスクの数。この数が継続的に増加する場合は、ウォームノードを追加することを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
WarmThreadpoolSearchQueue	ウォーム検索スレッドプール内のキューに入れられたタスクの数。キューサイズが一貫して大きい場合は、ウォームノードを追加することを検討してください。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average

メトリクス	説明
WarmJVMMemoryPressure	Warm ノードに使用される Java ヒープの最大パーセンテージ。 関連する統計: Maximum Note このメトリクスのロジックは、サービスソフトウェア R20220323 で変更されました。詳細については、「release notes」を参照してください。
WarmOldGenJVMMemoryPressure	ウォームノードあたりの「旧世代」に使用される Java ヒープの最大パーセンテージ。 関連する統計: Maximum
WarmJVMGCYoungCollectionCount	「低世代」ガベージコレクションがウォームノードで実行された回数。実行数が大量になり、かつ増え続けることは、通常のクラスター操作の一部です。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
WarmJVMGCYoungCollectionTime	クラスターがウォームノードで「低生成」ガベージコレクションの実行に費やしたミリ秒単位の時間。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
WarmJVMGCOldCollectionCount	「旧世代」ガベージコレクションがウォームノードで実行された回数。十分なリソースがあるクラスターでは、この回数は少ないままですが、まれに増加します。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average

メトリクス	説明
WarmConcurrentSearchRate	Warm ノード上のすべてのシャードに対して 1 分あたりの同時セグメント検索を使用した検索リクエストの合計数。_search API への 1 回の呼び出しに対して、さまざまなシャードから結果が返される可能性があります。これらのシャードのうちの 5 つが 1 つのノードにある場合、クライアントが 1 つのリクエストしか行っていない場合でも、ノードはこのメトリクスについて 5 を報告します。 関連するノードの統計: Average 関連するクラスターの統計: Sum、Maximum、Average
WarmConcurrentSearchLatency	ウォームノードでの同時セグメント検索を使用したすべての検索で、分 N から分 (N-1) の合計時間の差をミリ秒単位で表します。 関連するノードの統計: Average 関連するクラスターの統計: Maximum、Average
WarmThreadpoolIndexSearcherQueue	ウォームインデックス検索スレッドプール内のキューに入れられたタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Maximum、Average
WarmThreadpoolIndexSearcherRejected	ウォームインデックス検索スレッドプールで拒否されたタスクの数。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum
WarmThreadpoolIndexSearcherThreads	ウォームインデックス検索スレッドプールのサイズ。 関連するノードの統計: Maximum 関連するクラスターの統計: Sum、Average

コールドストレージのメトリクス

Amazon OpenSearch Service は、以下のコールドストレージのメトリクスを提供します。

メトリクス	説明
ColdStorageSpaceUtilization	クラスターで使用しているコールドストレージの合計容量 (MiB)。 関連する統計情報: Max
ColdToWarmMigrationFailureCount	コールドからウォームへの移行に失敗した移行の合計数。 関連する統計情報: Sum
ColdToWarmMigrationLatency	コールドからウォームへの移行が正常に完了するまでの時間。 関連する統計: Average
ColdToWarmMigrationQueueSize	現在、コールドストレージからウォームストレージへの移行を待機しているインデックスの数。 関連する統計: Maximum
ColdToWarmMigrationSuccessCount	コールドからウォームへの移行に成功した合計数。 関連する統計情報: Sum
WarmToColdMigrationFailureCount	ウォームからコールドへの移行に失敗した合計数。 関連する統計情報: Sum
WarmToColdMigrationLatency	ウォームからコールドへの移行が正常に完了するまでの時間。 関連する統計: Average
WarmToColdMigrationQueueSize	現在、ウォームストレージからコールドストレージへの移行を待機しているインデックスの数。 関連する統計: Maximum

メトリクス	説明
WarmToColdMigrationSuccessCount	ウォームからコールドへの移行に成功した合計数。 関連する統計情報: Sum

OpenSearch Optimized インスタンス (OR1) メトリクス

Amazon OpenSearch Service は、OR1 インスタンスについて次のメトリクスを提供します。

メトリクス	説明
RemoteStorageUsedSpace	クラスターで使用中の Amazon S3 スペースの合計容量 (MiB)。 関連する統計情報: Sum
RemoteStorageWriteRejected	リモートストレージとレプリケーションの負荷により、プライマリシャードで拒否されたリクエストの総数。これは、OpenSearch Service のプロセスの最後の起動を起点として計算されます。 関連する統計情報: Sum
ReplicationLagMaxTime	レプリカシャードがプライマリシャードの背後にある時間 (ミリ秒) 関連する統計: Maximum

アラートメトリクス

Amazon OpenSearch Service は、以下のアラートのメトリクスを提供します。

メトリクス	説明
AlertingDegraded	値 1 は、アラートインデックスが赤であるか、1 つ以上のノードがスケジュールどおりでないことを意味します。値 0 は正常な動作を示します。

メトリクス	説明
	関連する統計: Maximum
AlertingIndexExists	値 1 は、.opensearch-alerting-config インデックスが存在することを意味します。値 0 は、そのインデックスが存在しないことを意味します。アラート機能を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum
AlertingIndexStatus.green	インデックスのヘルス。値 1 は、緑を意味します。値 0 は、インデックスが存在しないか、緑ではないことを意味します。 関連する統計: Maximum
AlertingIndexStatus.red	インデックスのヘルス。値 1 は、赤を意味します。値 0 は、インデックスが存在しないか、赤でないことを意味します。 関連する統計: Maximum
AlertingIndexStatus.yellow	インデックスのヘルス。値 1 は、黄色を意味します。値 0 は、インデックスが存在しないか、黄色でないことを意味します。 関連する統計: Maximum
AlertingNodesNotOnSchedule	値 1 は、一部のジョブがスケジュールどおりに実行されていないことを意味します。値 0 は、すべてのアラートジョブがスケジュールどおりに実行されていることを意味します (またはアラートジョブが存在しないことを意味します)。OpenSearch Service コンソールを確認するか、_nodes/stats リクエストを行って、リソースの使用率が高いノードがあるかどうかを確認します。 関連する統計: Maximum

メトリクス	説明
AlertingNodesOnSchedule	値 1 は、すべてのアラートジョブがスケジュールどおりに実行されていることを意味します (またはアラートジョブが存在しないことを意味します)。値 0 は、一部のジョブがスケジュールどおりに実行されていないことを意味します。 関連する統計: Maximum
AlertingScheduledJobEnabled	値 1 は、opensearch.scheduled_jobs.enabled クラスター設定が true であることを意味します。値 0 は、その設定が false であり、スケジュールされたジョブが無効であることを意味します。 関連する統計: Maximum

異常検出のメトリクス

Amazon OpenSearch Service は、以下の異常検出のメトリクスを提供します。

メトリクス	説明
ADPluginUnhealthy	値 1 は、異常検出プラグインが正しく動作していないことを意味します。これは、障害の数が多いか、使用しているインデックスの 1 つが赤の状態であるためです。値 0 は、プラグインが想定どおりに動作していることを示します。 関連する統計: Maximum
ADExecuteRequestCount	異常検出のリクエストの数。 関連する統計情報: Sum
ADExecuteFailureCount	異常検出に失敗したリクエストの数。 関連する統計情報: Sum
ADHCExecuteFailureCount	高基数ディテクターの異常検出に失敗したリクエストの数。 関連する統計情報: Sum

メトリクス	説明
ADHCExecuteRequestCount	高基数ディテクターの異常検出のリクエストの数。 関連する統計情報: Sum
ADAnomalyResultsIndexStatusIndexExists	値 1 は、.opensearch-anomaly-results エイリアスが指すインデックスが存在することを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum
ADAnomalyResultsIndexStatus.red	値 1 は、 .opensearch-anomaly-results エイリアスが指すインデックスが赤の状態であることを意味します。値 0 は、そうでないことを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum
ADAnomalyDetectorsIndexStatusIndexExists	値 1 は、.opensearch-anomaly-detectors インデックスが存在することを意味します。値 0 は、そのインデックスが存在しないことを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum
ADAnomalyDetectorsIndexStatus.red	値 1 は、.opensearch-anomaly-detectors インデックスが赤の状態であることを意味します。値 0 は、そうでないことを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum
ADModelsCheckpointIndexStatusIndexExists	値 1 は、.opensearch-anomaly-checkpoints インデックスが存在することを意味します。値 0 は、そのインデックスが存在しないことを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum

メトリクス	説明
ADModelsCheckpointIndexStatus.red	値 1 は、.opensearch-anomaly-checkpoints インデックスが赤の状態であることを意味します。値 0 は、そうでないことを意味します。異常検出を初めて使用するまで、この値は 0 のままです。 関連する統計: Maximum

非同期検索メトリクス

Amazon OpenSearch Service は、以下の非同期検索のメトリクスを提供します。

非同期検索コーディネーターノードの統計 (コーディネーターノードあたり)

メトリクス	説明
AsynchronousSearchSubmissionRate	過去 1 分間に送信された非同期検索の数。
AsynchronousSearchInitializedRate	過去 1 分間に初期化された非同期検索の数。
AsynchronousSearchRunningCurrent	現在実行中の非同期検索の数。
AsynchronousSearchCompletionRate	過去 1 分間に正常に完了した非同期検索の数。
AsynchronousSearchFailureRate	過去 1 分間に完了し、失敗した非同期検索の数。

メトリクス	説明
AsynchronousSearchPersistRate	過去 1 分間に持続した非同期検索の数。
AsynchronousSearchPersistFailedRate	過去 1 分間に持続できなかった非同期検索の数。
AsynchronousSearchRejected	ノードの稼働時間以降に拒否された非同期検索の合計数。
AsynchronousSearchCancelled	ノードの稼働時間以降にキャンセルされた非同期検索の合計数。
AsynchronousSearchMaxRunningTime	過去 1 分間にノードで実行されている非同期検索の最長時間。

非同期検索クラスター統計

メトリクス	説明
AsynchronousSearchStoreHealth	過去 1 分間に持続したインデックス (赤/赤以外) 内のストアのヘルス。
AsynchronousSearchStoreSize	過去 1 分間のすべてのシャードのシステムインデックスのサイズ。
AsynchronousSearch	過去 1 分間にシステムインデックスに保存されたレスポンスの数。

メトリクス	説明
StoredResponseCount	

Auto-Tune メトリクス

Amazon OpenSearch Service は、Auto-Tune に以下のメトリクスを提供します。

メトリクス	説明
AutoTuneChangesHistoryHeapSize	ヒープサイズチューニング値の MiB 単位の変更履歴。
AutoTuneChangesHistoryJVMYoungGenArgs	JVM YongGen 引数の変更履歴。
AutoTuneFailed	Auto-Tune の変更が失敗したかどうかを示すブール値。
AutoTuneSucceeded	Auto-Tune の変更が成功したかどうかを示すブール値。
AutoTuneValue	中断を伴わない変更のキュー変更履歴 (カウント) とキャッシュチューニング変更履歴 (MiB 単位)。

Multi-AZ with Standby メトリクス

Amazon OpenSearch Service は、Multi-AZ with Standby で次のメトリクスを提供します。

アクティブなアベイラビリティーゾーンにおけるデータノードのノードレベルメトリクス

メトリクス	説明
CPUUtilization	クラスター内のデータノードの CPU 使用率の割合。Maximum は、CPU 使用率が最も高いノードを示します。Average は、クラスター内のすべ

メトリクス	説明
	てのノードを表します。このメトリクスは、個別のノードでも利用できます。
FreeStorageSpace	クラスター内のデータノードの空き領域。Sum はクラスターの合計空き容量を示しますが、正確な値を得るには期間を 1 分にする必要があります。Minimum と Maximum は、それぞれ空き領域が最も少ないノードと最も多いノードを示します。このメトリクスは、個別のノードでも利用できます。OpenSearch Service は、このメトリクスが 0 に達したときに ClusterBlockException をスローします。復旧するには、インデックスを削除する、より大きなインスタンスを追加する、既存のインスタンスに EBS ベースのストレージを追加する、のいずれかを実行する必要があります。詳細についてはthe section called “使用可能なストレージ領域の不足”を参照してください。 OpenSearch Service コンソールには、この値が GiB 単位で表示されます。Amazon CloudWatch コンソールには、この値が MiB 単位で表示されます。
JVMMemoryPressure	クラスター内のすべてのデータノードで使用する Java ヒープの最大パーセンテージ。OpenSearch Service は、インスタンスの RAM の半分を Java ヒープ (32 GiB のヒープサイズまで) に使用します。インスタンスは最大 64 GiB の RAM まで垂直スケーリングでき、それ以上はインスタンスを追加することで水平方向にスケーリングできます。「the section called “推奨される CloudWatch アラーム”」を参照してください。
SysMemoryUtilization	インスタンスが使用中のメモリの割合。このメトリクスの高い値は正常であり、通常はクラスターに問題はありません。潜在的なパフォーマンスおよび安定性の問題の指標については、「JVMMemoryPressure メトリクス」を参照してください。
IndexingLatency	ノード内のすべてのインデックス作成オペレーションにかかった合計時間 (ミリ秒) の、N 分と (N-1) 分の差。
IndexingRate	1 分あたりのインデックス作成オペレーションの数。

メトリクス	説明
SearchLatency	ノード内のすべての検索にかかった合計時間 (ミリ秒) の、N 分と (N-1) 分の差。
SearchRate	データノードのすべてのシャードに対する 1 分あたりの検索リクエストの総数。
ThreadpoolSearchQueue	検索スレッドプールでキューに入っているタスクの数。キューのサイズが一貫して大きい場合は、クラスターのスケーリングを検討してください。検索キューの最大サイズは 1,000 です。
ThreadpoolWriteQueue	書き込みスレッドプールでキューに入っているタスクの数。
ThreadpoolSearchRejected	検索スレッドプールで拒否されたタスクの数。この数が増え続ける場合は、クラスターのスケーリングを検討してください。
ThreadpoolWriteRejected	書き込みスレッドプールで拒否されたタスクの数。

アクティブなアベイラビリティーゾーンにおけるクラスターのクラスターレベルメトリクス

メトリクス	説明
DataNodes	アクティブシャードとスタンバイシャードの合計数。
DataNodesShards.active	アクティブなプライマリとレプリカの両方のシャードの合計数。
DataNodesShards.unassigned	クラスターのノードに割り当てられていないシャードの数。
DataNodesShards.initializing	初期化中のシャードの数。

メトリクス	説明
DataNodes Shards.relocating	再配置中のシャードの数。

アベイラビリティーゾーンのローテーションメトリクス

ActiveReads.*Availability-Zone* = 1 の場合、ゾーンはアクティブです。ActiveReads.*Availability-Zone* = 0 の場合、ゾーンはスタンバイ状態です。

ポイントインタイムメトリクス

Amazon OpenSearch Service は、ポイントインタイム (PIT) 検索で次のメトリクスを提供します。

PIT コーディネーターノードの統計 (コーディネーターノードあたり)

メトリクス	説明
CurrentPointInTime	ノード内のアクティブな PIT 検索コンテキストの数。
TotalPointInTime	ノードのアップタイム以降の、期限切れの PIT 検索コンテキストの数。
AvgPointInTimeAliveTime	ノードのアップタイム以降の、PIT 検索コンテキストの平均キープアライブ数。
HasActivePointInTime	値が 1 の場合、アクティブな PIT コンテキストがノードのアップタイム以降に存在していることを示します。値が 0 の場合は存在していません。
HasUsedPointInTime	値が 1 の場合、期限切れの PIT コンテキストがノードのアップタイム以降に存在していることを示します。値が 0 の場合は存在していません。

SQL メトリクス

Amazon OpenSearch Service は、以下の SQL サポートのメトリクスを提供します。

メトリクス	説明
SQLFailedRequestCountByCusErr	クライアントの問題により失敗した _sql API へのリクエストの数。例えば、IndexNotFoundException により、リクエストが HTTP ステータスコード 400 を返す場合があります。 関連する統計情報: Sum
SQLFailedRequestCountBySysErr	サーバーの問題または機能の制限により失敗した _sql API へのリクエストの数。例えば、VerificationException により、リクエストが HTTP ステータスコード 503 を返す場合があります。 関連する統計情報: Sum
SQLRequestCount	_sql API へのリクエストの数。 関連する統計情報: Sum
SQLDefaultCursorRequestCount	SQLRequestCount に似ていますが、測定対象はページネーションのリクエストのみです。 関連する統計情報: Sum
SQLUnhealthy	値 1 は、特定のリクエストへのレスポンスで、SQL プラグインが 5xx レスポンスコードを返すか、無効なクエリ DSL を OpenSearch に渡すことを示します。他のリクエストは引き続き成功します。値 0 は、最近の障害がないことを示します。値 1 が持続して表示される場合、クライアントがプラグインに対して行っているリクエストのトラブルシューティングを行います。 関連する統計: Maximum

k-NN メトリクス

Amazon OpenSearch Service は、以下の k 最近傍 (k-NN) プラグインのメトリクスを提供します。

メトリクス	説明
KNNCacheCapacityReached	キャッシュ容量に達したかどうかのノード単位のメトリクス。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計: Maximum
KNNCircuitBreakerTriggered	サーキットブレーカーがトリガーされるかどうかのクラスタ単位のメトリクス。KNNCacheCapacityReached についていずれかのノードが 1 の値を返す場合、この値も 1 を返します。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計: Maximum
KNNEvictionCount	メモリ制約またはアイドル時間のためにキャッシュから削除されたグラフ数のノード単位のメトリクス。インデックスの削除のために発生した明示的な削除はカウントされません。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
KNNGraphIndexErrors	ドキュメントの knn_vector フィールドを、エラーを生成したグラフに追加するリクエストの数のノードごとのメトリクス。 関連する統計情報: Sum
KNNGraphIndexRequests	ドキュメントの knn_vector フィールドを、グラフに追加するリクエストの数のノードごとのメトリクス。 関連する統計情報: Sum
KNNGraphMemoryUsage	現在のキャッシュサイズ (メモリー内のすべてのグラフの合計サイズ) のノードごとのメトリクス (KB)。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計: Average
KNNGraphQueryErrors	エラーを生成したグラフクエリの数のノード単位のメトリクス。

メトリクス	説明
	関連する統計情報: Sum
KNNGraphQueryRequests	グラフクエリの数のノード単位のメトリクス。 関連する統計情報: Sum
KNNHitCount	キャッシュヒットの数のノード単位のメトリクス。ユーザーがすでにメモリにロードされているグラフのクエリを行ったときに、キャッシュヒットが発生します。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
KNNLoadExceptionCount	グラフをキャッシュにロードしようとしたときに例外が発生した回数のノード単位のメトリクス。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
KNNLoadSuccessCount	プラグインがグラフをキャッシュに正常にロードした回数のノード単位のメトリクス。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
KNNMissCount	キャッシュミスの数のノード単位のメトリクス。ユーザーがまだメモリにロードされていないグラフのクエリを行ったときに、キャッシュミスが発生します。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
KNNQueryRequests	k-NN プラグインが受信したクエリリクエストの数のノード単位のメトリクス。 関連する統計情報: Sum

メトリクス	説明
KNNRemoteBuildEnabled	この機能を有効にするかどうかを指定するバイナリ値。 関連する統計: バイナリ
KNNRemoteIndexBuildFailureCount	ビルド失敗の合計数。 関連する統計情報: Sum
KNNRemoteIndexBuildSuccessCount	成功したビルドの合計数。 関連する統計情報: Sum
KNNScriptCompilationErrors	スクリプトのコンパイル中のエラーの数のノード単位のメトリクス。この統計は、k-NN スコアスクリプト検索にのみ関係します。 関連する統計情報: Sum
KNNScriptCompilations	k-NN スクリプトがコンパイルされた回数のノードごとのメトリクス。通常、この値は 1 または 0 であるはずですが、コンパイルされたスクリプトを含むキャッシュがいっぱいになると、k-NN スクリプトが再コンパイルされる可能性があります。この統計は、k-NN スコアスクリプト検索にのみ関係します。 関連する統計情報: Sum
KNNScriptQueryErrors	スクリプトクエリ中のエラーの数のノード単位のメトリクス。この統計は、k-NN スコアスクリプト検索にのみ関係します。 関連する統計情報: Sum
KNNScriptQueryRequests	スクリプトクエリの合計数のノード単位のメトリクス。この統計は、k-NN スコアスクリプト検索にのみ関係します。 関連する統計情報: Sum

メトリクス	説明
KNNTotalLoadTime	k-NN がグラフをキャッシュにロードするのにかかった時間 (ナノ秒)。このメトリクスは、おおよその k-NN 検索にのみ関係します。 関連する統計情報: Sum
VectorIndexBuildAccelerationOCU	ベクトルインデックス作成の高速化に使用される OpenSearch コンピューティングユニット (OCUs) の数。 関連する統計情報: Sum

クラスター間検索のメトリクス

Amazon OpenSearch Service は、以下のクラスター間検索のメトリクスを提供します。

ソースドメインのメトリクス

メトリクス	ディメンション	説明
CrossClusterOutboundConnections	ConnectionId	接続されたノードの数。スキップされたドメインが 1 つ以上レスポンスに含まれている場合は、このメトリクスを使用して異常な接続を追跡します。この数が 0 になった場合、その接続は正常ではありません。
CrossClusterOutboundRequests	ConnectionId	ターゲットドメインに送信された検索リクエストの数。ドメインでクラスター間検索リクエストが過負荷になっているかどうかを確認し、このメトリクスのスパイクと JVM/CPU スパイクの関連性を探るために使用します。

ターゲットドメインのメトリクス

メトリクス	ディメンション	説明
CrossClusterInboundRequests	ConnectionId	ソースドメインから受信した着信接続リクエストの数。

予期せず接続が切断された場合に CloudWatch アラームを追加します。アラームを作成する手順については、「静的しきい値に基づいて CloudWatch アラームを作成する」を参照してください。

クラスター間レプリケーションメトリクス

Amazon OpenSearch Service は、以下のクラスター間レプリケーションのメトリクスを提供します。

メトリクス	説明
ReplicationRate	1 秒あたりのレプリケーションオペレーションの平均率。このメトリクスは IndexingRate メトリクスに似ています。
LeaderCheckPoint	特定の接続については、すべてのレプリケートインデックスにおけるリーダーチェックポイントの合計値。このメトリクスを使用して、レプリケーションのレイテンシーを測定できます。
FollowerCheckPoint	特定の接続については、すべてのレプリケートインデックスにおけるフォロワーチェックポイントの合計値。このメトリクスを使用して、レプリケーションのレイテンシーを測定できます。
ReplicationNumSyncingIndices	SYNCING のレプリケーションステータスを持つインデックスの数。
ReplicationNumBootstrappingIndices	BOOTSTRAPPING のレプリケーションステータスを持つインデックスの数。

メトリクス	説明
ReplicationNumPausedIndices	PAUSED のレプリケーションステータスを持つインデックスの数。
ReplicationNumFailedIndices	FAILED のレプリケーションステータスを持つインデックスの数。
CrossClusterOutboundReplicationRequests	フォロワードメイン上のレプリケーショントランスポートリクエストの数。トランスポートリクエストは内部的なものであり、レプリケーション API オペレーションが呼び出されるたびに発生します。これらは、フォロワードメインがリーダードメインからの変更をポーリングする際にも発生します。
CrossClusterInboundReplicationRequests	リーダードメイン上のレプリケーショントランスポートリクエストの数。トランスポートリクエストは内部的なものであり、レプリケーション API オペレーションが呼び出されるたびに発生します。
AutoFollowNumSuccessStartReplication	特定の接続のレプリケーションルールによって正常に作成されたフォロワーインデックスの数。
AutoFollowNumFailedStartReplication	一致するパターンがあったときにレプリケーションルールによって作成されなかったフォロワーインデックスの数。この問題は、リモートクラスターのネットワーク上の問題、またはセキュリティ上の問題 (すなわち、関連付けられているロールにレプリケーションを開始する許可がない) が原因となって発生する可能性があります。
AutoFollowLeaderCallFailure	新しいデータをプルするための、フォロワーインデックスからリーダーインデックスへのクエリが失敗したかどうか。1 の値は、直前の 1 分間に 1 つ以上の失敗した呼び出しがあったことを意味します。

Learning to Rank のメトリクス

Amazon OpenSearch Service は、以下の Learning to Rank のメトリクスを提供します。

メトリクス	説明
LTRRequestTotalCount	ランク付けリクエストの合計数。
LTRRequestErrorCount	失敗したリクエストの合計数。
LTRStatus.red	プラグインの実行に必要なインデックスの 1 つが赤であるかどうかを追跡します。
LTRMemoryUsage	プラグインで使用されるメモリの合計。
LTRFeatureMemoryUsageInBytes	Learning to Rank 機能フィールドで使用されるメモリの量 (バイト単位)。
LTRFeaturesetMemoryUsageInBytes	すべての Learning to Rank 機能セットで使用されるメモリの量 (バイト単位)。
LTRModelMemoryUsageInBytes	すべての Learning to Rank モデルで使用されるメモリの量 (バイト単位)。

Piped Processing Language のメトリクス

Amazon OpenSearch Service は、以下の Piped Processing Language のメトリクスを提供します。

メトリクス	説明
PPLFailedRequestCountByCusErr	クライアントの問題により失敗した _ppl API へのリクエストの数。例えば、IndexNotFoundException により、リクエストが HTTP ステータスコード 400 を返す場合があります。

メトリクス	説明
PPLFailedRequestCountBySysErr	サーバーの問題または機能の制限により失敗した _ppl API へのリクエストの数。例えば、VerificationException により、リクエストが HTTP ステータスコード 503 を返す場合があります。
PPLRequestCount	_ppl API へのリクエストの数。

Amazon CloudWatch Logs を用いて OpenSearch ログをモニタリングする

Amazon OpenSearch Service は、Amazon CloudWatch Logs を通じて以下の OpenSearch ログを公開します。

- エラーログ
- 検索リクエストのスローログ
- シャードスローログ
- 監査ログ

検索シャードスローログ、インデックス作成シャードスローログ、およびエラーログは、パフォーマンスと安定性の問題のトラブルシューティングに役立ちます。監査ログは、コンプライアンスの目的でユーザーのアクティビティを追跡します。すべてのログはデフォルトで無効になっています。有効にすると、標準の CloudWatch 料金が適用されます。

> **Note**
>
> エラーログは、OpenSearch および Elasticsearch バージョン 5.1 以降のでのみ利用できます。スローログは、すべてのバージョンの OpenSearch および Elasticsearch で利用できます。

ログについては、OpenSearch は、Apache Log4j 2 とその組み込みログレベル (昇順) の TRACE、DEBUG、INFO、WARN、ERROR、および FATAL を使用します。

エラーログを有効にすると、OpenSearch Service によって、WARN、ERROR、および FATAL のログ行が CloudWatch に発行されます。OpenSearch サービスは、DEBUG レベルからのいくつかの例外も公開し、それには以下が含まれます。

- `org.opensearch.index.mapper.MapperParsingException`
- `org.opensearch.index.query.QueryShardException`
- `org.opensearch.action.search.SearchPhaseExecutionException`
- `org.opensearch.common.util.concurrent.OpenSearchRejectedExecutionException`
- `java.lang.IllegalArgumentException`

以下のように、エラーログは多くの状況でトラブルシューティングに役立ちます。

- Painless スクリプトのコンパイルの問題
- 無効なクエリ
- インデックス作成の問題
- スナップショットの失敗
- Index State Management の移行の失敗

> **Note**
>
> すべてのエラーがエラーログで報告されるわけではありません。

> **Note**
>
> OpenSearch Service は、発生したすべてのエラーをログに記録しません。

トピック

- ログ発行を有効にする (コンソール)
- ログ発行の有効化 (AWS CLI)
- ログ発行 (AWS SDKs) の有効化
- ログの発行を有効にする (CloudFormation)
- 検索リクエストスのスローログのしきい値設定

- シャードスローログのしきい値設定
- スローログのテスト
- ログの表示

ログ発行を有効にする (コンソール)

CloudWatch へのログの発行を有効にするには、OpenSearch Service コンソールを使用する方法が最も簡単です。

CloudWatch へのログの発行を有効にするには (コンソール)

1. aws.amazon.com に移動し、[サインイン] を選択して認証情報を指定します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. 更新するドメインを選択します。
4. [ログ] タブで、ログタイプを選択し、[有効化] を選択します。
5. 新しい CloudWatch ロググループを作成するか、既存のロググループを選択します。

   > **Note**
   >
   > 複数のログを有効にする場合は、各ログを個別にロググループに発行することをお勧めします。分離することで、ログをスキャンしやすくなります。

6. 適切なアクセス権限を含むアクセスポリシーを選択するか、コンソールに用意された JSON を使用してポリシーを作成します。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream"
```

```
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:log-
group:cw_log_group_name:*"
    }
  ]
}
```

Confused Deputy Problem (混乱した代理の問題) から自分を守るため
に、aws:SourceAccount および aws:SourceArn の条件キーをポリシーに追加することをお勧めします。ソースアカウントはドメインの所有者であり、ソース ARN はドメインの ARN です。これらの条件キーを追加するには、ドメインがサービスソフトウェア R20211203 以降にある必要があります。

例えば、次の条件ブロックをポリシーに追加できます。

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account-id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"
    }
}
```

> **Important**
>
> CloudWatch Logs では、リージョンごとに 10 個のリソースポリシーがサポートされます。複数の OpenSearch Service ドメインでログを有効にする場合は、この制限に達しないように、複数のロググループを含む広範囲のポリシーを作成して再利用します。ポリシーを更新する手順については、「the section called “ログ発行の有効化 (AWS CLI)”」を参照してください。

7. [有効] を選択します。

ドメインのステータスが [アクティブ] から [処理中] に変わります。ステータスは、ログの発行が有効になる前に [アクティブ] に戻る必要があります。この変更には通常 30 分かかりますが、ドメイン設定によってはさらに時間がかかる場合があります。

シャードスローログのいずれかを有効にしている場合は、「the section called “シャードスローログのしきい値設定”」を参照してください。監査ログを有効にした場合は、「the section called “ステップ 2: OpenSearch Dashboards で、監査ログを有効にする”」を参照してください。エラーログのみ有効にしている場合は、追加のステップを行う必要はありません。

ログ発行の有効化 (AWS CLI)

ログの発行を有効にする前に、CloudWatch ロググループが必要です。まだロググループがない場合は、次のコマンドを使用して作成できます。

```
aws logs create-log-group --log-group-name my-log-group
```

次のコマンドを入力してロググループの ARN を検索し、それを書き留めます。

```
aws logs describe-log-groups --log-group-name my-log-group
```

これで、このロググループに書き込むための許可を OpenSearch Service に付与できます。ロググループの末尾に近い位置にロググループの ARN を指定する必要があります。

```
aws logs put-resource-policy \
  --policy-name my-policy \
  --policy-document '{ "Version": "2012-10-17",        "Statement": [{ "Sid":
 "", "Effect": "Allow", "Principal": { "Service": "es.amazonaws.com"}, "Action":
[ "logs:PutLogEvents","logs:CreateLogStream"],"Resource": "cw_log_group_arn:*"}]}'
```

> **Important**
>
> CloudWatch Logs では、リージョンごとに 10 個のリソースポリシーがサポートされます。複数の OpenSearch Service ドメインでシャードスローログを有効にする場合は、複数のロググループが含まれている適用範囲のより広いポリシーを作成して再利用し、この制限に達しないようにします。

後でこのポリシーを確認する必要がある場合は、`aws logs describe-resource-policies` コマンドを使用します。ポリシーを更新するには、新しいポリシードキュメントで同じ `aws logs put-resource-policy` コマンドを実行します。

最後に、--log-publishing-options オプションを使用して発行を有効化できます。オプションの構文は、create-domain コマンドと update-domain-config コマンドのどちらでも同じです。

パラメータ	有効な値
--log-publishing-options	SEARCH_SLOW_LOGS={CloudWatchLogsLogGroupArn= *cw_log_group_arn* ,Enabled=true|false}
	INDEX_SLOW_LOGS={CloudWatchLogsLogGroupArn= *cw_log_group_arn* ,Enabled=true|false}
	ES_APPLICATION_LOGS={CloudWatchLogsLogGroupArn= *cw_log_group_arn* ,Enabled=true|false}
	AUDIT_LOGS={CloudWatchLogsLogGroupArn= *cw_log_group_arn* ,Enabled=true|false}

> **Note**
>
> 複数のログを有効にする場合は、各ログを個別にロググループに発行することをお勧めします。分離することで、ログをスキャンしやすくなります。

例

次の例では、指定されたドメインに対して検索およびインデックス作成のシャードスローログの発行を有効にします。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --log-publishing-options
 "SEARCH_SLOW_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-east-1:123456789012:log-group:my-log-
group,Enabled=true},INDEX_SLOW_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-east-1:123456789012:log-group:my-other-log-group,Enabled=true}"
```

CloudWatch への発行を無効にするには、Enabled=false を使用して同じコマンドを実行します。

シャードスローログのいずれかを有効にしている場合は、「the section called “シャードスローログのしきい値設定”」を参照してください。監査ログを有効にした場合は、「the section called “ステップ 2: OpenSearch Dashboards で、監査ログを有効にする”」を参照してください。エラーログのみ有効にしている場合は、追加のステップを行う必要はありません。

ログ発行 (AWS SDKs) の有効化

ログの発行を有効にする前に、CloudWatch ロググループを作成してその ARN を取得し、そのロググループに書き込むための許可を OpenSearch Service に付与する必要があります。関連するオペレーションは Amazon CloudWatch Logs API リファレンスに記載されています。

- `CreateLogGroup`
- `DescribeLogGroup`
- `PutResourcePolicy`

これらのオペレーションには、AWS SDK を使用してアクセスできます。

AWS SDKs (Android および iOS SDKs を除く) は、 `CreateDomain`および `--log-publishing-options`のオプションを含め、Amazon OpenSearch Service API リファレンスで定義されているすべてのオペレーションをサポートします`UpdateDomainConfig`。

シャードスローログのいずれかを有効にしている場合は、「the section called “シャードスローログのしきい値設定”」を参照してください。エラーログのみ有効にしている場合は、追加のステップを行う必要はありません。

ログの発行を有効にする (CloudFormation)

この例では、CloudFormation を使用して `opensearch-logs` というロググループを作成し、適切な許可を割り当ててから、アプリケーションログ、検索スローログ、およびインデックス作成シャードスローログのためにログの発行が有効化されたドメインを作成します。

ログの公開を有効化する前に、CloudWatch ロググループを作成する必要があります。

```
Resources:
  OpenSearchLogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: opensearch-logs
Outputs:
```

```
  Arn:
    Value:
      'Fn::GetAtt':
        - OpenSearchLogGroup
        - Arn
```

テンプレートは、ロググループの ARN を出力します。この場合、ARNは `arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs` です。

ARN を使用して、OpenSearch Service にロググループに書き込むための許可を付与するリソースポリシーを作成します。

```
Resources:
 OpenSearchLogPolicy:
   Type: AWS::Logs::ResourcePolicy
   Properties:
     PolicyName: my-policy
     PolicyDocument: "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Sid\": \"\",
 \"Effect\": \"Allow\", \"Principal\": { \"Service\": \"es.amazonaws.com\"}, \"Action
\":[ \"logs:PutLogEvents\",\"logs:CreateLogStream\"],\"Resource\": \"arn:aws:logs:us-
east-1:123456789012:log-group:opensearch-logs:*\"}]}"
```

最後に、以下の CloudFormation スタックが作成されます。これは、ログの発行が有効化された OpenSearch Service ドメインを生成します。アクセスポリシーは、 のユーザーがドメイン AWS アカウント へのすべての HTTP リクエストを行うことを許可します。

```
Resources:
  OpenSearchServiceDomain:
    Type: "AWS::OpenSearchService::Domain"
    Properties:
      DomainName: my-domain
      EngineVersion: "OpenSearch_1.0"
      ClusterConfig:
        InstanceCount: 2
        InstanceType: "r6g.xlarge.search"
        DedicatedMasterEnabled: true
        DedicatedMasterCount: 3
        DedicatedMasterType: "r6g.xlarge.search"
      EBSOptions:
        EBSEnabled: true
        VolumeSize: 10
        VolumeType: "gp2"
```

```
      AccessPolicies:
        Version: "2012-10-17"
        Statement:
            Effect: "Allow"
            Principal:
                AWS: "arn:aws:iam::123456789012:user/es-user"
            Action: "es:*"
            Resource: "arn:aws:es:us-east-1:123456789012:domain/my-domain/*"
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-
group:opensearch-logs"
          Enabled: true
        SEARCH_SLOW_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-
group:opensearch-logs"
          Enabled: true
        INDEX_SLOW_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-
group:opensearch-logs"
          Enabled: true
```

構文の詳細については、CloudFormation ユーザーガイドの「ログの発行オプション」を参照してください。

検索リクエストのスローログのしきい値設定

検索リクエストのスローログは、バージョン 2.13 以降で実行されている OpenSearch Service ドメインで検索できます。検索リクエストのスローログのしきい値は、リクエストの合計所要時間に対して設定されます。これは、個々のシャードの所要時間に対して設定されたシャードリクエストのスローログとは異なります。

クラスター設定を使用して、検索リクエストのスローログを指定できます。これは、インデックス設定で有効にするシャードスローログとは異なります。例えば、次の設定は、OpenSearch REST API を使用して指定できます。

```
PUT domain-endpoint/_cluster/settings
{
  "transient": {
    "cluster.search.request.slowlog.threshold.warn": "5s",
    "cluster.search.request.slowlog.threshold.info": "2s"
  }
```

```
}
```

> **Note**
>
> PUT /_cluster/settings エンドポイントは Amazon OpenSearch Service で制限されています。このリクエストを送信するとエラーが返されます。シャードスローログのしきい値`Your request: '/_cluster/settings' payload is not allowed.`を設定するには、代わりにインデックスレベルの設定を使用します。例えば、次のようになります。
>
> ```
> PUT domain-endpoint/index/_settings
> {
>   "index.search.slowlog.threshold.query.warn": "5s",
>   "index.search.slowlog.threshold.query.info": "2s"
> }
> ```

シャードスローログのしきい値設定

OpenSearch では、デフォルトでシャードスローログが無効になっています。CloudWatch へのシャードスローログの発行を有効にしたら、さらに OpenSearch インデックスごとにログしきい値を指定する必要があります。これらのしきい値は、ログに記録する内容とログレベルを正確に定義します。

例えば、次の設定は、OpenSearch REST API を使用して指定できます。

```
PUT domain-endpoint/index/_settings
{
  "index.search.slowlog.threshold.query.warn": "5s",
  "index.search.slowlog.threshold.query.info": "2s"
}
```

スローログのテスト

検索リクエストのスローログとシャードスローログの正常な発行をテストするには、極度に低い値から開始してログが CloudWatch に表示されることを確認し、その後、しきい値をより有用なレベルに高めます。

ログが表示されない場合は、以下を確認してください。

- CloudWatch ロググループが存在しているかどうか? CloudWatch コンソールを確認します。
- ロググループに書き込むための許可が OpenSearch Service にあるかどうか? OpenSearch Service コンソールを確認します。
- OpenSearch Service ドメインはロググループに発行するように設定されているかどうか? OpenSearch Service コンソールを確認するか、 AWS CLI `describe-domain-config`オプションを使用するか、いずれかの SDK `DescribeDomainConfig`を使用して を呼び出します。 SDKs
- OpenSearch ログしきい値が低くてリクエストが超過していないかどうか?

  ドメインの検索リクエストのスローログのしきい値を確認するには、次のコマンドを使用します。

  ```
  GET domain-endpoint/_cluster/settings?flat_settings
  ```

  インデックスのシャードスローログのしきい値を確認するには、次のコマンドを使用します。

  ```
  GET domain-endpoint/index/_settings?pretty
  ```

インデックスのスローログを無効にする場合は、変更したしきい値をデフォルト値の -1 に戻します。

OpenSearch Service コンソールを使用して CloudWatch への発行を無効にすると、OpenSearch によるログの生成は停止 AWS CLI されず、それらのログの発行のみが停止します。シャードスローログが不要になった場合はインデックス設定、検索リクエストのスローログが不要になった場合はドメイン設定を確認してください。

ログの表示

CloudWatchでアプリケーションとスローログを表示する方法は、その他の CloudWatch ログを表示する方法と同じです。詳細については、Amazon CloudWatch Logs ユーザーガイドの「ログデータの表示」を参照してください。

ログを表示する際の考慮事項は以下のとおりです。

- OpenSearch Service は、各行の最初の 255,000 文字のみを CloudWatch に発行します。残りのコンテンツは切り捨てられます。監査ログの場合、それはメッセージあたり 10,000 文字です。
- CloudWatch では、ログの内容を識別しやすいようにログストリーム名のサフィックスとして `-index-slow-logs`、`-search-slow-logs`、`-application-logs`、および `-audit-logs` が使用されます。

Amazon OpenSearch Service での監査ログのモニタリング

Amazon OpenSearch Service ドメインできめ細かなアクセスコントロールを使用する場合、データの監査ログを有効にすることができます。監査ログは高度にカスタマイズ可能で、認証の成功と失敗、OpenSearch へのリクエスト、インデックスの変更、受信検索クエリなど、OpenSearch クラスターでのユーザーアクティビティを追跡できます。デフォルトの設定では、一般的な一連のユーザーアクションが追跡されますが、正確なニーズに合わせて設定を調整することをお勧めします。

OpenSearch アプリケーションログおよびスローログのように、OpenSearch Service は CloudWatch Logs に監査ログを発行します。有効にすると、標準の CloudWatch 料金が適用されます。

> **Note**
>
> 監査ログを有効にするには、ユーザーロールが `security_manager` ロールにマッピングされる必要があり、これにより、OpenSearch `plugins/_security` REST APIへのアクセス権が提供されます。詳細については the section called “マスターユーザーの変更”を参照してください。

トピック

- 制限事項
- 監査ログ記録の有効化
- を使用して監査ログ記録を有効にする AWS CLI
- 設定 API を使用して監査ログを有効にする
- 監査ログのレイヤーとカテゴリ
- 監査ログの設定
- 監査ログの例
- REST API を使用した監査ログの設定

制限事項

監査ログには以下の制限事項があります。

- 監査ログには、送信先のドメインアクセスポリシーによって拒否されたクロスクラスター検索リクエストは含まれません。

- 各監査ログメッセージの最大サイズは 10,000 文字です。この制限を超えると、監査ログメッセージが切り捨てられます。

監査ログ記録の有効化

監査ログの有効化は、2 段階のプロセスです。まず、CloudWatch Logs に CloudWatch Logs に監査ログを発行するようにドメインを設定します。次に、OpenSearch Dashboards で監査ログを有効にして、ニーズを満たすように監査ログを設定します。

> **Important**
>
> これらの手順の実行中にエラーが発生した場合、トラブルシューティング情報については、the section called “監査ログを有効にできない” を参照してください。

ステップ 1: 監査ログを有効にし、アクセスポリシーを設定する

次の手順では、コンソールを使用して監査ログを有効にする方法について説明します。または OpenSearch Service API を使用して有効に AWS CLIすることもできます。

OpenSearch Service ドメインの監査ログを有効にするには (コンソール)

1. ドメインを選択して設定を開き、[ログ] タブに移動します。
2. [監査ログ] を選択してから、[有効化] を選択します。
3. CloudWatch ロググループを作成するか、既存のロググループを選択します。
4. 適切なアクセス権限を含むアクセスポリシーを選択するか、コンソールに用意された JSON を使用してポリシーを作成します。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
```

```
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
opensearch/domains/domain-name/*"
    }
  ]
}
```

Confused Deputy Problem (混乱した代理の問題) から自分を守るために、aws:SourceAccount および aws:SourceArn の条件キーをポリシーに追加することをお勧めします。ソースアカウントはドメインの所有者であり、ソース ARN はドメインの ARN です。これらの条件キーを追加するには、ドメインがサービスソフトウェア R20211203 以降にある必要があります。

例えば、次の条件ブロックをポリシーに追加できます。

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account-id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"
    }
}
```

5. [有効化] を選択します。

ステップ 2: OpenSearch Dashboards で、監査ログを有効にする

OpenSearch Service コンソールで監査ログを有効にした後、OpenSearch Dashboards でも有効にして、ニーズを満たすように設定する必要があります。

1. OpenSearch Dashboards を開き、左側のサイドメニューから [セキュリティ] を選択します。
2. [監査ログ] を選択します。
3. [監査ログ作成を有効にする] を選択します。

Dashboards UI には、[全般設定] および [コンプライアンス設定] の下で、監査ログの設定の完全なコントロールが用意されています。すべての設定オプションの説明については、「監査ログの設定」を参照してください。

を使用して監査ログ記録を有効にする AWS CLI

次の AWS CLI コマンドは、既存のドメインの監査ログを有効にします。

```
aws opensearch update-domain-config --domain-name my-domain --log-publishing-options
 "AUDIT_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-east-1:123456789012:log-
group:my-log-group,Enabled=true}"
```

ドメインを作成するときに、監査ログを有効にすることもできます。詳細については、「AWS CLI コマンドのリファレンス」を参照してください。

設定 API を使用して監査ログを有効にする

次の設定 API へのリクエストにより、既存のドメインで監査ログが有効になります。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
  "LogPublishingOptions": {
    "AUDIT_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:123456789012:log-
group1:sample-domain",
      "Enabled":true
    }
  }
}
```

詳細については、「Amazon OpenSearch Service API reference」(Amazon OpenSearch Service API リファレンス) を参照してください。

監査ログのレイヤーとカテゴリ

クラスターコミュニケーションは、2 つの別々のレイヤーを介して行われます: RESTレイヤーとトランスポートレイヤー。

- REST レイヤーは、curl、Logstash、OpenSearch Dashboards、Java 高レベル REST クライアント、Python Requests ライブラリなどの HTTP クライアントとの通信 (クラスターに到達する HTTP リクエスト) を対象としています。

- トランスポートレイヤーは、ノード間のコミュニケーションを対象としています。例えば、検索リクエストが（REST レイヤーを介して）クラスターに到着した後、リクエストを処理する調整ノードは、クエリを他のノードに送信し、そのレスポンスを受け取り、必要なドキュメントを収集し、それらを最終的なレスポンスと照合します。シャード割り当てや再分散などのオペレーションは、トランスポートレイヤーでも実行されます。

レイヤー全体の監査ログと、レイヤーの個々の監査カテゴリを有効または無効にできます。次の表に、監査カテゴリの概要と、監査カテゴリが使用可能なレイヤーを示します。

Category	説明	REST で利用可能	トランスポートで利用可能
FAILED_LOGIN	リクエストに無効な資格情報が含まれ、認証に失敗しました。	はい	はい
MISSING_PRIVILEGES	ユーザーには、リクエストを行う権限がありませんでした。	はい	はい
GRANTED_PRIVILEGES	ユーザーには、リクエストを行う権限がありました。	はい	はい
OPENSEARCH_SECURITY_INDEX_ATTEMPT	リクエストが `.opendistro_security` インデックスを修正しようとしました。	いいえ	はい
AUTHENTICATED	リクエストに有効な認証情報が含まれ、認証に成功しました。	はい	はい
INDEX_EVENT	リクエストは、インデックスの作成、エイリアスの設定、強制マージの実行など、インデック	いいえ	はい

Category	説明	REST で利用可能	トランスポートで利用可能
	スの管理オペレーションを実行しました。このカテゴリに含まれる `indices:admin/` アクションの完全なリストは、OpenSearch ドキュメントで入手可能です。		

これらの標準カテゴリに加えて、きめ細かなアクセスコントロールには、データコンプライアンス要件を満たすように設計されたいくつかの追加カテゴリが用意されています。

Category	説明
COMPLIANCE_DOC_READ	リクエストは、インデックス内のドキュメントに対して読み取りイベントを実行しました。
COMPLIANCE_DOC_WRITE	リクエストは、インデックス内のドキュメントに対して書き込みイベントを実行しました。
COMPLIANCE_INTERNAL_CONFIG_READ	リクエストは、`.opendistro_security` インデックスで読み取りイベントを実行しました。
COMPLIANCE_INTERNAL_CONFIG_WRITE	リクエストは、`.opendistro_security` インデックスで書き込みイベントを実行しました。

カテゴリおよびメッセージ属性の任意の組み合わせを指定できます。例えば、REST リクエストを送信してドキュメントのインデックスを作成すると、監査ログに次の行が表示されることがあります。

- REST レイヤーで AUTHENTICATED (認証)
- トランスポートレイヤーで GRANTED_PREVIEWE (認可)
- COMPLIANCE_DOC_WRITE (インデックスに書き込まれたドキュメント)

監査ログの設定

監査ログには、多数の設定オプションがあります。

全般設定

全般設定では、個々のカテゴリまたはレイヤー全体を有効または無効にできます。除外カテゴリとして GRANTED_Privileges および Authenticated を残すことを強くお勧めします。それ以外の場合、これらのカテゴリは、クラスターへの有効なリクエストごとにログされます。

名前	バックエンドの設定	説明
REST レイヤー	enable_rest	REST レイヤーで発生するイベントを有効または無効にします。
REST 無効なカテゴリ	disabled_rest_categories	REST レイヤーで無視する監査カテゴリを指定します。これらのカテゴリを変更すると、監査ログのサイズが大幅に増加する可能性があります。
トランスポートレイヤー	enable_transport	トランスポートレイヤーで発生するイベントを有効または無効にします。
トランスポート無効なカテゴリ	disabled_transport_categories	トランスポートレイヤーで無視する必要がある監査カテゴリを指定します。これらのカテゴリを変更すると、監査ログのサイズが大幅に増加する可能性があります。

属性設定では、各ログ行の詳細量をカスタマイズできます。

名前	バックエンドの設定	説明
一括リクエスト	resolve_bulk_requests	この設定を有効にすると、一括要求でドキュメントごとにログが生成されます。これにより、監査ログのサイズが大幅に増加する可能性があります。
リクエストボディ	log_request_body	リクエストのリクエストボディを含めます。

名前	バックエンドの設定	説明
インデックスの解決	resolve_indices	インデックスに対してエイリアスを解決します。

無視設定を使用して、一連のユーザーまたは API パスを除外します。

名前	バックエンドの設定	説明
無視されたユーザー	ignore_users	除外するユーザーを指定します。
無視されたリクエスト	ignore_requests	除外するリクエストパターンを指定します。

コンプライアンス設定

コンプライアンス設定では、インデックス、ドキュメント、またはフィールドレベルのアクセスを調整できます。

名前	バックエンドの設定	説明
コンプライアンスのロギング	enable_compliance	コンプライアンスのロギングを有効または無効にします。

読み取りおよび書き込みイベントのロギングには以下の設定を指定できます。

名前	バックエンドの設定	説明
内部設定のロギング	internal_config	`.opendistro_security` インデックスのイベントのロギングを有効または無効にします。

読み取りイベントには以下の設定を指定できます。

名前	バックエンドの設定	説明
メタデータの読み取り	read_metadata_only	読み取りイベントのメタデータのみを含めます。ドキュメントフィールドを含めないでください。
無視されたユーザー	read_ignore_users	読み取りイベントには特定のユーザーを含めないでください。
監視するフィールド	read_watched_fields	読み取りイベントを監視するインデックスとフィールドを指定します。監視するフィールドを追加すると、ドキュメントアクセスごとに 1 つのログが生成されます。これにより、監査ログのサイズが大幅に増加する可能性があります。監視するフィールドは、インデックスパターンとフィールドパターンをサポートします。 `{ "index-name-pattern": [ "field-name-pattern" ], "logs*": [ "message" ], "twitter": [ "id", "user*" ] }`

書き込みイベントには以下の設定を指定できます。

名前	バックエンドの設定	説明
メタデータの書き込み	write_metadata_only	書き込みイベントのメタデータのみを含めます。ドキュメントフィールドを含めないでください。

名前	バックエンドの設定	説明
差分のログ	write_log_diffs	write_metadata_only が false の場合は、書き込みイベント間の違いのみを含めます。
無視されたユーザー	write_ignore_users	書き込みイベントには特定のユーザーを含めないでください。
インデックスの監視	write_watched_indices	書き込みイベントを監視するインデックスまたはインデックスパターンを指定します。監視するフィールドを追加すると、ドキュメントアクセスごとに 1 つのログが生成されます。これにより、監査ログのサイズが大幅に増加する可能性があります。

監査ログの例

このセクションには、設定例、検索リクエスト、およびインデックスのすべての読み取りおよび書き込みイベントの結果となる監査ログが含まれます。

ステップ 1: 監査ログを設定する

CloudWatch Logs グループへの監査ログの公開を有効にした後、OpenSearch Dashboards の監査ロギングページに移動し、[監査ロギングを有効にする] を選択します。

1. [全般設定] で、[設定] を選択し、[REST レイヤー] が有効であることを確認します。
2. [コンプライアンス設定] で、[設定] を選択します。
3. [書き込み] の下で、[監視フィールド] に、追加このインデックスへのすべての書き込みイベントのために `accounts` を追加します。
4. [読み取り] で、[監視フィールド] に、`accounts` インデックスの `ssn` および `id-` フィールドを追加します。

```
{
  "accounts-": [
    "ssn",
    "id-"
  ]
}
```

ステップ 2: 読み取りイベントと書き込みイベントを実行する

1. OpenSearch Dashboards に移動し、[デベロッパーツール] を選択し、サンプルドキュメントのインデックスを作成します。

```
PUT accounts/_doc/0
{
  "ssn": "123",
  "id-": "456"
}
```

2. 読み取りイベントをテストするには、次のリクエストを送信します。

```
GET accounts/_search
{
  "query": {
    "match_all": {}
  }
}
```

ステップ 3: ログを確認する

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. 監査ログを有効にするときに指定したロググループを選択します。ロググループ内で、OpenSearch Service はドメイン内の各ノードのログストリームを作成します。
4. [ログストリーム] で、[すべて検索] を選択します。
5. 読み取りおよび書き込みイベントについては、対応するログを参照してください。ログが表示される前に、5 秒の遅延が予想されます。

   監査ログの書き込みの例

```
{
  "audit_compliance_operation": "CREATE",
  "audit_cluster_name": "824471164578:audit-test",
  "audit_node_name": "be217225a0b77c2bd76147d3ed3ff83c",
  "audit_category": "COMPLIANCE_DOC_WRITE",
  "audit_request_origin": "REST",
  "audit_compliance_doc_version": 1,
```

```
  "audit_node_id": "3xNJhm4XS_yTzEgDWcGRjA",
  "@timestamp": "2020-08-23T05:28:02.285+00:00",
  "audit_format_version": 4,
  "audit_request_remote_address": "3.236.145.227",
  "audit_trace_doc_id": "lxnJGXQBqZSlDB91r_uZ",
  "audit_request_effective_user": "admin",
  "audit_trace_shard_id": 8,
  "audit_trace_indices": [
    "accounts"
  ],
  "audit_trace_resolved_indices": [
    "accounts"
  ]
}
```

監査ログの読み取りの例

```
{
  "audit_cluster_name": "824471164578:audit-docs",
  "audit_node_name": "806f6050cb45437e2401b07534a1452f",
  "audit_category": "COMPLIANCE_DOC_READ",
  "audit_request_origin": "REST",
  "audit_node_id": "saSevm9ASte0-pjAtYi2UA",
  "@timestamp": "2020-08-31T17:57:05.015+00:00",
  "audit_format_version": 4,
  "audit_request_remote_address": "54.240.197.228",
  "audit_trace_doc_id": "config:7.7.0",
  "audit_request_effective_user": "admin",
  "audit_trace_shard_id": 0,
  "audit_trace_indices": [
    "accounts"
  ],
  "audit_trace_resolved_indices": [
    "accounts"
  ]
}
```

リクエストボディを含めるには、OpenSearch Dashboards の [コンプライアンス設定] に戻り、[メタデータの書き込み] を無効にします。特定のユーザーによるイベントを除外するには、そのユーザーを [無視されたユーザー] に追加します。

各監査ログフィールドの説明については、「監査ログフィールドのリファレンス]を参照してください。監査ログデータの検索および分析の詳細については、Amazon CloudWatch Logs ユーザーガイドの「CloudWatch Logs Insights を用いたログデータの分析」を参照してください。

REST API を使用した監査ログの設定

OpenSearch Dashboards を使用して監査ログを設定することをお勧めしますが、きめ細かなアクセスコントロールの REST API を使用することもできます。このセクションには、リクエストの例が含まれています。REST API に関する完全なドキュメントは、OpenSearch ドキュメントで入手可能です。

```
PUT _opendistro/_security/api/audit/config
{
  "enabled": true,
  "audit": {
    "enable_rest": true,
    "disabled_rest_categories": [
      "GRANTED_PRIVILEGES",
      "AUTHENTICATED"
    ],
    "enable_transport": true,
    "disabled_transport_categories": [
      "GRANTED_PRIVILEGES",
      "AUTHENTICATED"
    ],
    "resolve_bulk_requests": true,
    "log_request_body": true,
    "resolve_indices": true,
    "exclude_sensitive_headers": true,
    "ignore_users": [
      "kibanaserver"
    ],
    "ignore_requests": [
      "SearchRequest",
      "indices:data/read/*",
      "/_cluster/health"
    ]
  },
  "compliance": {
    "enabled": true,
    "internal_config": true,
    "external_config": false,
```

```
    "read_metadata_only": true,
    "read_watched_fields": {
      "read-index-1": [
        "field-1",
        "field-2"
      ],
      "read-index-2": [
        "field-3"
      ]
    },
    "read_ignore_users": [
      "read-ignore-1"
    ],
    "write_metadata_only": true,
    "write_log_diffs": false,
    "write_watched_indices": [
      "write-index-1",
      "write-index-2",
      "log-*",
      "*"
    ],
    "write_ignore_users": [
      "write-ignore-1"
    ]
  }
}
```

Amazon EventBridge を用いた OpenSearch Service イベントのモニタリング

Amazon OpenSearch Service は Amazon EventBridge と統合され、ドメインに影響する特定のイベントを通知します。 AWS サービスからのイベントは、ほぼリアルタイムで EventBridge に配信されます。同じイベントが Amazon EventBridge の先行である Amazon CloudWatch Events にも送信されます。簡単なルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。自動的にトリガーできるオペレーションには、以下が含まれます。

- AWS Lambda 関数の呼び出し
- Amazon EC2 Run Command の呼び出し
- Amazon Kinesis Data Streams へのイベントの中継

- AWS Step Functions ステートマシンのアクティブ化
- Amazon SNS トピックまたは Amazon SQS キューの通知

詳細については、Amazon EventBridge ユーザーガイドの「Amazon EventBridge の開始方法」を参照してください。

トピック

- サービスソフトウェア更新イベント
- Auto-Tune イベント
- クラスターヘルスイベント
- VPC エンドポイントイベント
- ノードの廃止イベント
- デグレードされたノードの廃止イベント
- ドメインエラーイベント
- チュートリアル: Amazon OpenSearch Service EventBridge イベントをリッスンする
- チュートリアル: 利用可能なソフトウェア更新に関する Amazon SNS アラートの送信

サービスソフトウェア更新イベント

OpenSearch Service は、次のサービスソフトウェア更新イベントのいずれかが発生したときに、EventBridge にイベントを送信します。

サービスソフトウェア更新が利用可能

OpenSearch Service は、サービスソフトウェア更新が利用可能になったときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
```

```
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Available",
    "severity": "Informational",
    "description": "Service software update R20220928 available. Service Software
 Deployment Mechanism:
                   Blue/Green. For more information on deployment configuration,
 please
                   see: https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/managedomains-configuration-changes.html"
  }
}
```

サービスソフトウェア更新をスケジュールしました

OpenSearch Service は、サービスソフトウェア更新がスケジュールされたときに、このイベントを送信します。オプションの更新の場合、予定日に通知が届きます。再スケジュールはいつでも可能です。必要な更新の場合、予定日の 3 日前に通知が届きます。再スケジュールは、決められた期間内に行えます。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Scheduled",
    "severity": "High",
    "description": "A new service software update [R20200330-p1] has been scheduled at
 [21st May 2023 12:40 GMT].
```

```
                    Please see documentation for more information on scheduling
 software updates:
                    https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/service-software.html."
  }
}
```

サービスソフトウェア更新を再スケジュールしました

OpenSearch Service は、サービスソフトウェア更新が再スケジュールされたときに、このイベントを送信します。詳細については、「the section called “オプションの更新と必須の更新”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Rescheduled",
    "severity": "High",
    "description": "The service software update [R20200330-p1], which was originally
 scheduled for
                    [21st May 2023 12:40 GMT], has been rescheduled to [23rd May 2023
 12:40 GMT].
                    Please see documentation for more information on scheduling
 software updates:
                    https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/service-software.html."
  }
}
```

サービスソフトウェア更新が開始しました

OpenSearch Service は、サービスソフトウェア更新が開始したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Started",
    "severity": "Informational",
    "description": "Service software update [R20200330-p1] started.
  }
}
```

サービスソフトウェアの更新が完了しました

OpenSearch Service は、サービスソフトウェアの更新が完了したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
```

```
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Completed",
    "severity": "Informational",
    "description": "Service software update [R20200330-p1] completed."
  }
}
```

サービスソフトウェア更新をキャンセルしました

OpenSearch Service は、サービスソフトウェア更新がキャンセルされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Cancelled",
    "severity": "Informational",
    "description": "The scheduled service software update [R20200330-p1] has been
 cancelled as a
                    newer update is available. Please schedule the latest update."
  }
}
```

スケジュールされたサービスソフトウェア更新をキャンセルしました

OpenSearch Service は、ドメインに対して以前に予定されていたサービスソフトウェア更新がキャンセルされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Cancelled",
    "severity": "Informational",
    "description": "The scheduled service software update [R20200330-p1] has been
 cancelled."
  }
}
```

サービスソフトウェア更新が実行されませんでした

OpenSearch Service は、サービスソフトウェア更新が開始できないときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
```

```
    "status": "Unexecuted",
    "severity": "Informational",
    "description": "The scheduled service software update [R20200330-p1] cannot be
 started. Reason: [reason]"
  }
}
```

サービスソフトウェアの更新に失敗しました

OpenSearch Service は、サービスソフトウェア更新が失敗したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Failed",
    "severity": "High",
    "description": "Installation of service software update [R20200330-p1] failed.
 [reason].
  }
}
```

サービスソフトウェア更新が必要

OpenSearch Service は、サービスソフトウェア更新が必要なときに、このイベントを送信します。詳細については、「the section called “オプションの更新と必須の更新”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Required",
    "severity": "High",
    "description": "Service software update [R20200330-p1] available. Update
                    will be automatically installed after [21st May 2023] if no
                    action is taken. Service Software Deployment Mechanism: Blue/Green.
                    For more information on deployment configuration, please see:
                    https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/managedomains-configuration-changes.html"
  }
}
```

Auto-Tune イベント

OpenSearch Service は、次の Auto-Tune イベントのいずれかが発生したときに、EventBridge にイベントを送信します。

Auto-Tune 保留中

OpenSearch Service は、Auto-Tune がクラスターのパフォーマンスと可用性を改善するためのチューニングの推奨事項を特定したときに、このイベントを送信します。このイベントは、Auto-Tune が無効になっているドメインに対してのみ表示されます。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
```

```
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Pending",
    "description": "Auto-Tune recommends the following new settings for your
 domain: { JVM Heap size : 60%}. Enable Auto-Tune to improve cluster stability and
 performance.",
    "scheduleTime": "{iso8601-timestamp}"
  }
}
```

Auto-Tune が開始されました

OpenSearch Service は、Auto-Tune がドメインに新しい設定を適用し始めたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Started",
    "scheduleTime": "{iso8601-timestamp}",
    "startTime": "{iso8601-timestamp}",
    "description" : "Auto-Tune is applying the following settings to your domain: { JVM
 Heap size : 60%}."
  }
```

```
}
```

Auto-Tune には、スケジュールされた Blue/Green のデプロイが必要です。

OpenSearch Service は、スケジュールされた Blue/Green のデプロイを必要とするチューニングの推奨事項を Auto-Tune が特定したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Low",
    "status": "Pending",
    "startTime": "{iso8601-timestamp}",
    "description": "Auto-Tune has identified the following settings for your domain
 that require a blue/green deployment: { JVM Heap size : 60%}.
                   You can schedule the deployment for your preferred time."
  }
}
```

Auto-Tune がキャンセルされました

OpenSearch Service は、保留中のチューニングの推奨事項がないために Auto-Tune スケジュールがキャンセルされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
```

```
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Low",
    "status": "Cancelled",
    "scheduleTime": "{iso8601-timestamp}",
    "description": "Auto-Tune has cancelled the upcoming blue/green deployment."
  }
}
```

自動チューニングが完了しました

OpenSearch Service は、Auto-Tune が Blue/Green のデプロイを完了し、クラスターが新しい JVM 設定を用いてオペレーション可能になったときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Completed",
    "completionTime": "{iso8601-timestamp}",
    "description": "Auto-Tune has completed the blue/green deployment and successfully
 applied the following settings: { JVM Heap size : 60%}."
  }
```

```
}
```

Auto-Tune が無効になり、変更が元に戻されました

OpenSearch Service は、Auto-Tune が無効になっており、適用された変更がロールバックされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": [ "arn:aws:es:us-east-1:123456789012:domain/test-domain" ],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Completed",
    "description": "Auto-Tune is now disabled. All settings have been reverted. Auto-
Tune will continue to evaluate
                    cluster performance and provide recommendations.",
    "completionTime": "{iso8601-timestamp}"
  }
}
```

Auto-Tune が無効になり、変更が保持されました

OpenSearch Service は、Auto-Tune が無効になっており、適用された変更が保持されたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
```

```
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Completed",
    "description": "Auto-Tune is now disabled. The most-recent settings by Auto-Tune
 have been retained.
                   Auto-Tune will continue to evaluate cluster performance and provide
 recommendations.",
    "completionTime": "{iso8601-timestamp}"
  }
}
```

クラスターヘルスイベント

OpenSearch サービスは、クラスターのヘルスが侵害されると、特定のイベントを EventBridge に送信します。

赤いクラスターの復旧が開始しました

OpenSearch Service は、クラスターのステータスが赤になってから連続して 1 時間以上が経過した後に、このイベントを送信します。クラスタステータスを修正するために、スナップショットから赤いインデックスの自動復元が試行されます。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
```

```
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"Automatic Snapshot Restore for Red Indices",
      "status":"Started",
      "severity":"High",
      "description":"Your cluster status is red. We have started automatic snapshot
 restore for the red indices.
                    No action is needed from your side. Red indices [red-index-0, red-
index-1]"
   }
}
```

赤いクラスタの復旧が部分的に完了しました

OpenSearch Service は、赤いクラスターステータスを修正する際、スナップショットから赤いインデックスのサブセットのみを復元できたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"Automatic Snapshot Restore for Red Indices",
      "status":"Partially Restored",
      "severity":"High",
      "description":"Your cluster status is red. We were able to restore the following
 Red indices from
                    snapshot: [red-index-0]. Indices not restored: [red-index-1].
 Please refer https://docs.aws.amazon.com/opensearch-service/latest/developerguide/
handling-errors.html#handling-errors-red-cluster-status for troubleshooting steps."
   }
```

```
}
```

赤いクラスターの復旧に失敗しました

OpenSearch Service は、赤いクラスターステータスを修正する際、スナップショットから赤いインデックスのサブセットのみを復元できたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"Automatic Snapshot Restore for Red Indices",
      "status":"Failed",
      "severity":"High",
      "description":"Your cluster status is red. We were unable to restore the Red
 indices automatically.
                    Indices not restored: [red-index-0, red-index-1]. Please refer
 https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-
errors.html#handling-errors-red-cluster-status for troubleshooting steps."
   }
}
```

削除するシャード

OpenSearch Service は、14 日間連続して赤いままのクラスターステータスの自動修正を試み、1 つ以上のインデックスが赤のままのときに、このイベントを送信します。さらに 7 日後 (赤のステータスになってから連続して 21 日後)、OpenSearch Service は、すべての赤いインデックスの割り当てられていないシャードを削除します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2022-04-09T10:36:48Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "severity":"Medium",
      "description":"Your cluster status is red. Please fix the red indices as soon as
 possible.
                    If not fixed by 2022-04-12 01:51:47+00:00, we will delete all
 unassigned shards,
                    the unit of storage and compute, for these red indices to recover
 your domain and make it green.
                    Please refer to https://docs.aws.amazon.com/opensearch-service/
latest/developerguide/handling-errors.html#handling-errors-red-cluster-status for
 troubleshooting steps.
                    test_data, test_data1",
      "event":"Automatic Snapshot Restore for Red Indices",
      "status":"Shard(s) to be deleted"
   }
}
```

シャードが削除されました

OpenSearch Service は、クラスターステータスが赤になってから連続して 21 日以上が経過した後に、このイベントを送信します。すべての赤いインデックスの未割り当てシャード (ストレージとコンピューティング) が削除されます。詳細については、「the section called “赤いクラスターの自動修復”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
```

```
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2022-04-09T10:54:48Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "severity":"High",
      "description":"We have deleted unassinged shards, the unit of storage and
 compute, in
                    red indices: index-1, index-2 because these indices were red for
 more than
                    21 days and could not be restored with the automated restore
 process.
                    Please refer to https://docs.aws.amazon.com/opensearch-service/
latest/developerguide/handling-errors.html#handling-errors-red-cluster-status for
 troubleshooting steps.",
      "event":"Automatic Snapshot Restore for Red Indices",
      "status":"Shard(s) deleted"
   }
}
```

シャードカウントが高い警告

OpenSearch Service は、ホットデータノードの平均シャード数が、推奨されるデフォルト制限の 1,000 の 90% を超えた場合に、このイベントを送信します。Elasticsearch および OpenSearch の以降のバージョンでは、ノードあたりのシャード数の制限を設定できますが、ノードあたり 1,000 シャード以下です。シャード数の選択を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2016-11-01T13:12:22Z",
```

```
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"High Shard Count",
     "status":"Warning",
     "severity":"Low",
     "description":"One or more data nodes have close to 1000 shards. To ensure optimum
 performance and stability of your
                    cluster, please refer to the best practice guidelines - https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/sizing-domains.html#bp-
sharding."
  }
}
```

シャード数の制限を超えました

OpenSearch Service は、ホットデータノードの平均シャード数が、推奨されるデフォルト制限の 1,000 を超えた場合に、このイベントを送信します。Elasticsearch および OpenSearch の以降のバージョンでは、ノードあたりのシャード数の制限を設定できますが、ノードあたり 1,000 シャード以下です。シャード数の選択を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2016-11-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"High Shard Count",
     "status":"Warning",
     "severity":"Medium",
     "description":"One or more data nodes have more than 1000 shards. To ensure
 optimum performance and stability of your
                    cluster, please refer to the best practice guidelines - https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/sizing-domains.html#bp-
sharding."
```

```
  }
}
```

ディスク容量の不足

OpenSearch Service は、クラスター内の 1 つ以上のノードが使用可能なストレージ領域の 25% 未満、または 25 GB 未満の場合にこのイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Low Disk Space",
     "status":"Warning",
     "severity":"Medium",
     "description":"One or more data nodes in your cluster has less than 25% of storage
 space or less than 25GB.
                  Your cluster will be blocked for writes at 20% or 20GB. Please refer
 to the documentation for more information - https://docs.aws.amazon.com/opensearch-
service/latest/developerguide/handling-errors.html#troubleshooting-cluster-block"
  }
}
```

低ディスクウォーターマーク超過

OpenSearch Service は、クラスター内のすべてのノードで利用可能なストレージ領域が 10% 未満になった、または 10 GB 未満になったときにこのイベントを送信します。すべてのノードが低ディスクウォーターマークを超過すると、新しいインデックスは黄色のクラスターになり、すべてのノードが高ディスクウォーターマークを下回ると、赤色のクラスターになります。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Low Disk Watermark Breach",
     "status":"Warning",
     "severity":"Medium",
     "description":"Low Disk Watermark threshold is about to be breached. Once the
 threshold is breached, new index creation will be blocked on all
                    nodes to prevent the cluster status from turning red. Please
 increase disk size to suit your storage needs. For more information,
                    see https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/handling-errors.html#troubleshooting-cluster-block".
  }
}
```

EBS バーストバランスが 70% 未満

OpenSearch Service は、1 つ以上のデータノードで EBS バーストバランスが 70% を下回ったときにこのイベントを送信します。EBS バーストバランスが枯渇すると、クラスターが広範囲にわたって使用できなくなり、I/O リクエストがスロットリングされ、インデックス作成や検索リクエストのレイテンシーが高くなり、タイムアウトが発生する可能性があります。この問題を修正するステップについては、「the section called “低 EBS バーストバランス”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
```

```
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"EBS Burst Balance",
     "status":"Warning",
     "severity":"Medium",
     "description":"EBS burst balance on one or more data nodes is below 70%.
                    Follow https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/handling-errors.html#handling-errors-low-ebs-burst
                    to fix this issue."
  }
}
```

EBS バーストバランスが 20% 未満

OpenSearch Service は、1 つ以上のデータノードで EBS バーストバランスが 20% を下回ったときにこのイベントを送信します。EBS バーストバランスが枯渇すると、クラスターが広範囲にわたって使用できなくなり、I/O リクエストがスロットリングされ、インデックス作成や検索リクエストのレイテンシーが高くなり、タイムアウトが発生する可能性があります。この問題を修正するステップについては、「the section called “低 EBS バーストバランス”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"EBS Burst Balance",
     "status":"Warning",
     "severity":"High",
     "description":"EBS burst balance on one or more data nodes is below 20%.
                    Follow https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/handling-errors.html#handling-errors-low-ebs-burst
                    to fix this issue.
```

```
  }
}
```

ディスクスループットのスロットリング

OpenSearch Service は、EBS ボリュームまたは EC2 インスタンスのスループット制限を理由として、ドメインへの読み取りおよび書き込みリクエストがスロットリングされているときに、このイベントを送信します。この通知を受け取った場合は、 AWS 推奨されるベストプラクティスに従ってボリュームまたはインスタンスをスケールアップすることを検討してください。ボリュームタイプが gp2 の場合は、ボリュームサイズを大きくします。ボリュームタイプが gp3 の場合は、より多くのスループットをプロビジョニングします。また、インスタンスベースと最大 EBS スループットがプロビジョニングされたボリュームスループット以上であることを確認し、それに応じてスケールアップすることもできます。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Disk Throughput Throttle",
     "status":"Warning",
     "severity":"Medium",
     "description":"Your domain is experiencing throttling due to instance or volume
 throughput limitations.
                    Please consider scaling your domain to suit your throughput needs.
 In July 2023, we improved
                    the accuracy of throughput throttle calculation by replacing ‘Max
 volume throughput’ with
                    ‘Provisioned volume throughput’. Please refer to the documentation
 for more information."
   }
}
```

大きなシャードサイズ

OpenSearch Service は、クラスター内の 1 つ以上のシャードが 50 GiB または 65 GiB を超えたときにこのイベントを送信します。クラスターのパフォーマンスと安定性を最適化するには、シャードサイズを減らします。

詳細については、「sharding best practices」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Large Shard Size",
     "status":"Warning",
     "severity":"Medium",
     "description":"One or more shards are larger than 65GiB. To ensure optimum cluster
 performance and stability, reduce shard sizes.
                    For more information, see https://docs.aws.amazon.com/opensearch-
service/latest/developerguide/monitoring-events.html#monitoring-events-large-shard-
size."
  }
}
```

高い JVM 使用率

OpenSearch Service は、ドメインの JVMMemoryPressure メトリクスが 80% を超えたときにこのイベントを送信します。30 分間で 92% を超えると、クラスターに対するすべての書き込みオペレーションがブロックされます。クラスターの最適な安定性を確保するには、クラスターへのトラフィックを減らすか、またはドメインをスケールしてワークロードに十分なメモリを提供します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"High JVM Usage",
     "status":"Warning",
     "severity":"High",
     "description":"JVM memory pressure has exceeded 80%. If it exceeds 92% for 30
 minutes, all write operations to your cluster
                    will be blocked. To ensure optimum cluster stability, reduce
 traffic to the cluster or use larger instance types.
                    For more information, see https://docs.aws.amazon.com/opensearch-
service/latest/developerguide/monitoring-events.html#monitoring-events-high-jvm."
  }
}
```

GC が不足している

OpenSearch Service は、最大 JVM が 70% を超え、最大値と最小値の差が 30% 未満の場合にこのイベントを送信します。これは、JVM がワークロード用のガベージコレクションサイクル中に十分なメモリを再利用できないことを示唆している可能性があります。これにより、レスポンスがますます遅くなり、レイテンシーが高くなる可能性があります。また場合によっては、ヘルスチェックのタイムアウトによりノードがドロップすることさえあります。クラスターの最適な安定性を確保するには、クラスターへのトラフィックを減らすか、またはドメインをスケールしてワークロードに十分なメモリを提供します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
```

```
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Insufficient GC",
     "status":"Warning",
     "severity":"Medium",
     "description":"Maximum JVM is above 70% and JVM range is less than 30%. This may
 indicate insufficient garbage collection for your workload.
                   For more information, see https://docs.aws.amazon.com/opensearch-
service/latest/developerguide/monitoring-events.html#monitoring-events-insufficient-
gc."
  }
}
```

カスタムインデックスルーティングの警告

OpenSearch Service は、ドメインが処理状態にあり、ブルー/グリーンデプロイが停止する原因となり得る index.routing.allocation のカスタム設定を持つインデックスが含まれているときにこのイベントを送信します。設定が適切に適用されていることを確認します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Custom Index Routing Warning",
     "status":"Warning",
     "severity":"Medium",
     "description":"Your domain is in processing state and contains indice(s) with
 custom index.routing.allocation
```

```
                    settings which can cause blue-green deployments to get stuck.
 Verify settings are applied properly.
                    For more information, see https://docs.aws.amazon.com/opensearch-
service/latest/developerguide/monitoring-events.html#monitoring-events-index-routing."
  }
}
```

シャードロックの失敗

OpenSearch Service は、[ShardLockObtainFailedException] を含むシャードが割り当てられていないことを理由としてドメインが異常な状態である場合にこのイベントを送信します。詳細については、「Amazon OpenSearch Service のインメモリシャードロックの例外を解決するにはどうすればよいですか?」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2017-12-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Failed Shard Lock",
     "status":"Warning",
     "severity":"Medium",
     "description":"Your domain is unhealthy due to unassigned shards with
 [ShardLockObtainFailedException]. For more information,
                    see https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/monitoring-events.html#monitoring-events-failed-shard-lock."
}
```

VPC エンドポイントイベント

OpenSearch Service では、AWS PrivateLink インターフェイスエンドポイントに関連する EventBridge に特定のイベントを送信します。

VPC エンドポイントが作成できない

OpenSearch Service は、リクエストされた VPC エンドポイントを作成できない場合、このイベントを送信します。このエラーは、1 つのリージョンで許可される VPC エンドポイントの数の制限に到達したために発生することがあります。このエラーは、指定されたサブネットまたはセキュリティグループが存在しない場合にも表示されます。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service VPC Endpoint Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"VPC Endpoint Create Validation",
      "status":"Failed",
      "severity":"High",
      "description":"Unable to create VPC endpoint aos-0d4c74c0342343 for domain
                     arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the
 following validation failures: You've reached the limit on the
                     number of VPC endpoints that you can create in the AWS Region."
   }
}
```

VPC エンドポイントが更新できない

OpenSearch Service は、リクエストされた VPC エンドポイントを削除できない場合、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service VPC Endpoint Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"VPC Endpoint Update Validation",
      "status":"Failed",
      "severity":"High",
      "description":"Unable to update VPC endpoint aos-0d4c74c0342343 for domain
                     arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the
 following validation failures: <failure message>."
    }
}
```

VPC エンドポイントが削除できない

OpenSearch Service は、リクエストされた VPC エンドポイントを削除できない場合、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service VPC Endpoint Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
      "event":"VPC Endpoint Delete Validation",
```

```
      "status":"Failed",
      "severity":"High",
      "description":"Unable to delete VPC endpoint aos-0d4c74c0342343 for domain
                    arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the
 following validation failures: Specified subnet doesn't exist."
   }
}
```

ノードの廃止イベント

OpenSearch Service は、以下にあるノードの廃止イベントのいずれかが発生したときに、EventBridge にイベントを送信します。

ノードの廃止をスケジュールしました

OpenSearch Service は、ノードの廃止がスケジュールされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2023-04-07T10:07:33Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Node Retirement Notification",
    "status": "Scheduled",
    "severity": "Medium",
    "description": "An automated action to retire and replace a node has been scheduled
 on your domain.
                    The node will be replaced in the next off-peak window. For more
 information, see
                    https://docs.aws.amazon.com/opensearch-service/latest/
developerguide/monitoring-events.html."
  }
}
```

ノードの廃止が完了しました

OpenSearch Service は、ノードの廃止が完了したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。


```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2023-04-07T10:07:33Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Node Retirement Notification",
    "status": "Completed",
    "severity": "Medium",
    "description": "The node has been retired and replaced with a new node."
  }
}
```


ノードの廃止に失敗しました

OpenSearch Service は、ノードの廃止が失敗したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。


```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2023-04-07T10:07:33Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
```

```
    "event": "Node Retirement Notification",
    "status": "Failed",
    "severity": "Medium",
    "description": "Node retirement failed. No actions are required from your end. We
 will automatically
                   retry replacing the node."
  }
}
```

デグレードされたノードの廃止イベント

OpenSearch Service は、ノード上のハードウェアのデグレードによりノードの交換が必要な場合に、これらのイベントを送信します。

デグレードされたノードの廃止通知

OpenSearch Service は、デグレードされたノードを廃止して置き換える自動アクションがドメインにスケジュールされたときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"db233454-aad1-7676-3b15-10a84b052baa",
   "detail-type":"Amazon OpenSearch Service Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2024-01-11T08:16:06Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-node-replacement"
   ],
   "detail":{
      "severity":"Medium",
      "description":"An automated action to retire and replace a node has
 been scheduled on your domain. For more information, please see https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/monitoring-events.html.",
      "event":"Degraded Node Retirement Notification",
      "status":"Scheduled"
   }
```

```
}
```

デグレードされたノードの廃止を完了しました

OpenSearch Service は、デグレードされたノードが廃止され、新しいノードに置き換えられたときにこのイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"7444215c-90f9-a52d-bcda-e85973a9a762",
   "detail-type":"Amazon OpenSearch Service Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2024-01-11T10:20:30Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-node-replacement"
   ],
   "detail":{
      "severity":"Medium",
      "description":"The node has been retired and replaced with a new node.",
      "event":"Degraded Node Retirement Notification",
      "status":"Completed"
   }
}
```

デグレードされたノードの廃止に失敗しました

OpenSearch Service は、デグレードされたノードの廃止が失敗したときに、このイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"c328e9bb-93b9-c0b2-b17a-df527fdf96b6",
   "detail-type":"Amazon OpenSearch Service Notification",
```

```
   "source":"aws.es",
   "account":"123456789012",
   "time":"2024-01-11T08:31:38Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-node-replacement"
   ],
   "detail":{
      "severity":"Medium",
      "description":"Node retirement failed. No actions are required from your end. We
 will automatically re-try replacing the node.",
      "event":"Degraded Node Retirement Notification",
      "status":"Failed"
   }
}
```

ドメインエラーイベント

OpenSearch Service は、次のドメインエラーのいずれかが発生したときに、EventBridge にイベントを送信します。

ドメイン更新の検証エラー

OpenSearch Service は、ドメインで更新または設定変更の実行を試行した際に 1 つ以上の検証エラーが発生した場合にこのイベントを送信します。これらの障害を解決するためのステップについては、「the section called “検証エラーのトラブルシューティング”」を参照してください。

例

このタイプのイベントの例を以下に示します。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-012345678901",
   "detail-type":"Amazon OpenSearch Service Domain Update Notification",
   "source":"aws.es",
   "account":"123456789012",
   "time":"2016-11-01T13:12:22Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:es:us-east-1:123456789012:domain/test-domain"
   ],
   "detail":{
```

```
      "event":"Domain Update Validation",
      "status":"Failed",
      "severity":"High",
      "description":"Unable to perform updates to your domain due to the following
 validation failures: <failures>
                    Please see the documentation for more information https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/managedomains-
configuration-changes.html#validation"
   }
}
```

KMS キーにアクセスできない

OpenSearch Service は、AWS KMS キーにアクセスできないときにこのイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Domain Error Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2016-11-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"KMS Key Inaccessible",
     "status":"Error",
     "severity":"High",
     "description":"The KMS key associated with this domain is inaccessible. You are at
 risk of losing access to your domain.
                    For more information, please refer to https://docs.aws.amazon.com/
opensearch-service/latest/developerguide/encryption-at-rest.html#disabled-key."
  }
}
```

ドメインの分離

OpenSearch Service は、ドメインが分離されてネットワークからアクセスできないためにリクエストの受信、読み取り、または書き込みができなくなったときにこのイベントを送信します。

例

このタイプのイベントの例を以下に示します。

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2023-11-01T13:12:22Z",
  "region":"us-east-1",
  "resources":["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail":{
     "event":"Domain Isolation Notification",
     "status":"Error",
     "severity":"High",
     "description":"Your OpenSearch Service domain has been isolated. An isolated
 domain is unreachable by network and cannot receive, read, or write requests. For more
 information and assistance, please contact AWS Support at https://docs.aws.amazon.com/
opensearch-service/latest/developerguide/encryption-at-rest.html#disabled-key."
  }
}
```

チュートリアル: Amazon OpenSearch Service EventBridge イベントをリッスンする

このチュートリアルでは、Amazon OpenSearch Service イベントをリッスンし、CloudWatch Logs ログストリームに書き込むシンプルな AWS Lambda 関数を設定します。

前提条件

このチュートリアルは、既存の OpenSearch Service ドメインがあることを前提としています。ドメインをまだ作成していない場合は、1 つ作成するために「ドメインの作成と管理」の手順に従います。

ステップ 1: Lambda 関数を作成する

この手順では、OpenSearch Service イベントメッセージのターゲットとなるシンプルな Lambda 関数を作成します。

ターゲットの Lambda 関数を作成するには

1. https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. [Lambda 関数の作成]、[一から作成] の順に選択します。
3. [関数名] では、event-handler と入力します。
4. [ランタイム] では、[Python 3.8] を選択します。
5. [Create function] を選択します。
6. [関数コード] セクションで、以下の例に一致するようにサンプルコードを編集します。

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.es":
        raise ValueError("Function only supports input from events with a source
 type of: aws.es")

    print(json.dumps(event))
```

これは、OpenSearch Service から送信されたイベントを印刷するシンプルな Python 3.8 関数です。すべてが正しく設定されると、このチュートリアルの最後に、この Lambda 関数に関連付けられている CloudWatch Logs ログストリームにイベントの詳細が表示されます。

7. [デプロイ] をクリックします。

ステップ 2: イベントルールを登録する

このステップでは、OpenSearch Service ドメインからイベントをキャプチャする EventBridge ルールを作成します。このルールでは、それが定義されているアカウント内のすべてのイベントがキャプチャされます。イベントメッセージ自体に、イベントソースに関する情報 (イベントソースの送信元ドメインの情報など) が含まれています。この情報を使用して、プログラムでイベントをフィルタリングおよびソートできます。

EventBridge ルールを作成するには

1. EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. [ルールの作成] を選択します。
3. ルールに event-rule と名前を付けます。
4. [次へ] を選択します。

5. イベントパターンの場合、[AWS のサービス]、[Amazon OpenSearch Service] 、[すべてのイベント] の順にクリックします。このパターンは、すべての OpenSearch Service ドメインおよびすべての OpenSearch Service イベントに適用されます。または、より具体的なパターンを作成し、一部の結果をフィルターで除外することもできます。
6. [次へ] を選択します。
7. ターゲットに [Lambda 関数] を選択します。機能ドロップダウンで [イベントハンドラ] を選択します。
8. [次へ] を選択します。
9. タグをスキップして [次へ] をクリックします。
10. 設定を確認して、[ルールの作成] を選択します。

ステップ 3: 設定をテストする

次回 OpenSearch Service コンソールの通知セクションで通知を受け取ったときに、すべてが適切に設定されている場合、Lambda 関数がトリガーされ、関数の CloudWatch Logs ログストリームにイベントデータが書き込まれます。

設定をテストするには

1. CloudWatch コンソール https://console.aws.amazon.com/cloudwatch/ を開きます。
2. ナビゲーションペインで、[ログ] を選択して Lambda 関数 (/aws/lambda/event-handler など) のロググループを選択します。
3. イベントデータを表示するログストリームを選択します。

チュートリアル: 利用可能なソフトウェア更新に関する Amazon SNS アラートの送信

このチュートリアルでは、Amazon OpenSearch Service で利用可能なサービスソフトウェア更新の通知をキャプチャし、 Amazon Simple Notification Service (Amazon SNS) を介して E メール通知を送信する Amazon EventBridge イベントルールを設定します。

前提条件

このチュートリアルは、既存の OpenSearch Service ドメインがあることを前提としています。ドメインをまだ作成していない場合は、1 つ作成するために「ドメインの作成と管理」の手順に従います。

ステップ 1: Amazon SNS トピックを作成してサブスクライブする

新しいイベントルールのイベントターゲットとして使用する Amazon SNS トピックを設定します。

Amazon SNS ターゲットを作成するには

1. Amazon SNS コンソールのhttps://console.aws.amazon.com/sns/v3/homeを開いてください。
2. [トピック] および [トピックの作成] を選択します。
3. ジョブタイプでは、[標準] を選択し、ジョブに software-update と名前を付けます。
4. [トピックを作成] を選択します。
5. トピックが作成されたら、[サブスクリプションの作成] を選択します。
6. [プロトコル] で [E メール] を選択します。[Endpoint (エンドポイント)] では、現在アクセスできるメールアドレスを入力し、[Create subscription (サブスクリプションの作成)] を選択します。
7. メールアカウントを確認し、サブスクリプションの確認メールメッセージが届くのを待ちます。確認メールが届いたら、[サブスクリプションの確認] を選択します。

ステップ 2: イベントルールを登録する

次に、サービスソフトウェア更新イベントのみをキャプチャするイベントルールを登録します。

イベントルールを作成するには

1. EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. [ルールの作成] を選択します。
3. ルールに softwareupdate-rule と名前を付けます。
4. [次へ] を選択します。
5. イベントパターンの場合、[AWS のサービス]、[Amazon OpenSearch Service (マネージドクラスター)]、[Amazon OpenSearch Service ソフトウェアの更新通知] の順に選択します。このパターンは、OpenSearch Service からのサービスソフトウェアアップデートイベントに一致します。イベントパターンの詳細については、Amazon EventBridge ユーザーガイドの「Amazon EventBridge イベントパターン」を参照してください。
6. 特定の重大度のみをフィルターできます (オプション)。各イベントの重大度については、「the section called “サービスソフトウェア更新イベント”」を参照してください。
7. [次へ] を選択します。
8. ターゲットに [SNS トピック] を選択し、[software-update] (ソフトウェア更新) を選択します。

9. [次へ] を選択します。
10. タグをスキップして [次へ] を選択します。
11. ルールの設定を確認して、[ルールの作成] を選択します。

次回 OpenSearch Service から利用可能なサービスソフトウェアの更新に関する通知を受け取ったときに、すべてが適切に設定されている場合、Amazon SNS は更新に関するアラートを E メールで送信します。

を使用した Amazon OpenSearch Service API コールのモニタリング AWS CloudTrail

Amazon OpenSearch Service は AWS CloudTrail、OpenSearch Service のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されます。CloudTrail は、OpenSearch Service のすべての設定 API 呼び出しをイベントとしてキャプチャします。

> **Note**
>
> CloudTrail は、`CreateDomain` や `GetUpgradeStatus` などの、Configuration API への呼び出しのみをキャプチャします。CloudTrail は、`_search` や `_bulk` などの、OpenSearch APIへの呼び出しをキャプチャしません。これらの呼び出しについては、「the section called "監査ログのモニタリング"」を参照してください。

キャプチャされた呼び出しには、OpenSearch Service コンソール AWS CLI、または AWS SDK からの呼び出しが含まれます。証跡を作成する場合は、OpenSearch Service のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます 証跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、OpenSearch Service に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail での Amazon OpenSearch Service 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。OpenSearch Service でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントと

ともに CloudTrail イベントに記録されます。最近のイベントは、 AWS アカウント アカウントで表示、検索、ダウンロードできます。詳細については、「CloudTrail Event 履歴でのイベントの表示」を参照してください。

OpenSearch Service のイベントなど、 AWS アカウント アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については次を参照してください:

- の証跡の作成 AWS アカウント
- AWS CloudTrail Logs との サービス統合
- 「CloudTrail の Amazon SNS 通知の設定」
- CloudTrail ログファイルを複数のリージョンから受け取る、複数のアカウントから CloudTrail ログファイルを受け取る

すべての OpenSearch Service 設定 API アクションは CloudTrail によってログに記録され、「Amazon OpenSearch Service API リファレンス」に記載されています。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか
- リクエストの送信に使用された一時的なセキュリティ認証情報に、ロールとフェデレーテッドユーザーのどちらが使用されたか
- リクエストが別の AWS サービスによって行われたかどうか

詳細については、「CloudTrail userIdentity 要素」を参照してください。

Amazon OpenSearch Service のログファイルエントリを理解する

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエ

ストパラメータなどに関する情報が含まれます。CloudTrailログファイルは、パブリックAPIコールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`CreateDomain`オペレーションを示すCloudTrailログエントリの例は、次のとおりです。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/test-user",
    "accountId": "123456789012",
    "accessKeyId": "access-key",
    "userName": "test-user",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-08-21T21:59:11Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2018-08-21T22:00:05Z",
  "eventSource": "es.amazonaws.com",
  "eventName": "CreateDomain",
  "awsRegion": "us-west-1",
  "sourceIPAddress": "123.123.123.123",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "engineVersion": "OpenSearch_1.0",
    "clusterConfig": {
      "instanceType": "m4.large.search",
      "instanceCount": 1
    },
    "snapshotOptions": {
      "automatedSnapshotStartHour": 0
    },
    "domainName": "test-domain",
    "encryptionAtRestOptions": {},
    "eBSOptions": {
      "eBSEnabled": true,
      "volumeSize": 10,
      "volumeType": "gp2"
    },
```

```
    "accessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"123456789012\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:us-west-1:123456789012:domain/test-domain/*\"}]}",
    "advancedOptions": {
      "rest.action.multi.allow_explicit_index": "true"
    }
  },
  "responseElements": {
    "domainStatus": {
      "created": true,
      "clusterConfig": {
        "zoneAwarenessEnabled": false,
        "instanceType": "m4.large.search",
        "dedicatedMasterEnabled": false,
        "instanceCount": 1
      },
      "cognitoOptions": {
        "enabled": false
      },
      "encryptionAtRestOptions": {
        "enabled": false
      },
      "advancedOptions": {
        "rest.action.multi.allow_explicit_index": "true"
      },
      "upgradeProcessing": false,
      "snapshotOptions": {
        "automatedSnapshotStartHour": 0
      },
      "eBSOptions": {
        "eBSEnabled": true,
        "volumeSize": 10,
        "volumeType": "gp2"
      },
      "engineVersion": "OpenSearch_1.0",
      "processing": true,
      "aRN": "arn:aws:es:us-west-1:123456789012:domain/test-domain",
      "domainId": "123456789012/test-domain",
      "deleted": false,
      "domainName": "test-domain",
      "accessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":\"arn:aws:iam::123456789012:root\"},\"Action\":\"es:*\",
\"Resource\":\"arn:aws:es:us-west-1:123456789012:domain/test-domain/*\"}]}"
    }
```

```
  },
  "requestID": "12345678-1234-1234-1234-987654321098",
  "eventID": "87654321-4321-4321-4321-987654321098",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Cluster Insights による統合運用モニタリング

Amazon OpenSearch Service に Cluster Insights が追加されました。これは、単一のダッシュボードを通じてクラスターの包括的な運用上の可視性を提供するモニタリングソリューションです。これにより、クラスターの可用性やパフォーマンスに対する潜在的なリスクを特定するために、さまざまなログやメトリクスを分析して関連付ける必要がなくなります。このソリューションは、ノード、インデックス、シャード間の重要な運用データの統合を自動化し、複雑なトラブルシューティングを合理化されたプロセスに変換します。大きなシャードや低ディスクウォーターマークなどの問題を検出し、ノード、インデックス、シャードレベルで詳細なメトリクスを表示し、セキュリティと回復性のベストプラクティスにアクセスできます。

> **Note**
>
> Cluster Insights は、以下のバージョンサポートで追加料金なしで利用できます。
>
> - Amazon OpenSearch Service コンソール (クラスターヘルスタブ): Elasticsearch 6.8 以降、およびすべての OpenSearch バージョン (1.0 以降) で使用できます。
> - OpenSearch Dashboards UI (OpenSearch UI アプリケーション経由): OpenSearch 2.17 以降が必要です。以前のバージョンを実行しているドメインはデータソースとして関連付けることができますが、UI のクラスターインサイトの概要には表示されません。OpenSearch バージョン 2.17 および 2.19 のドメインも、最新のサービスソフトウェアバージョン更新に含まれている必要があります。
> - クエリビュータブ (OpenSearch Dashboards UI の場合): OpenSearch 2.19 以降が必要です。
>
> 注: OpenSearch バージョン 2.17 より前のバージョンを実行しているドメインは、OpenSearch UI アプリケーションでデータソースとして関連付けることができますが、クラスターインサイトの概要には表示されません。OpenSearch 2.17 以降を実行しているドメインのみが、Dashboards UI の Cluster Insights で完全にサポートされています。

利点

- プロアクティブモニタリング - 個々のノードやインデックスからシャードや検索クエリまで、すべてのコンポーネントにわたって詳細なパフォーマンスメトリクスを使用してクラスターの状態をプロアクティブにモニタリングします。

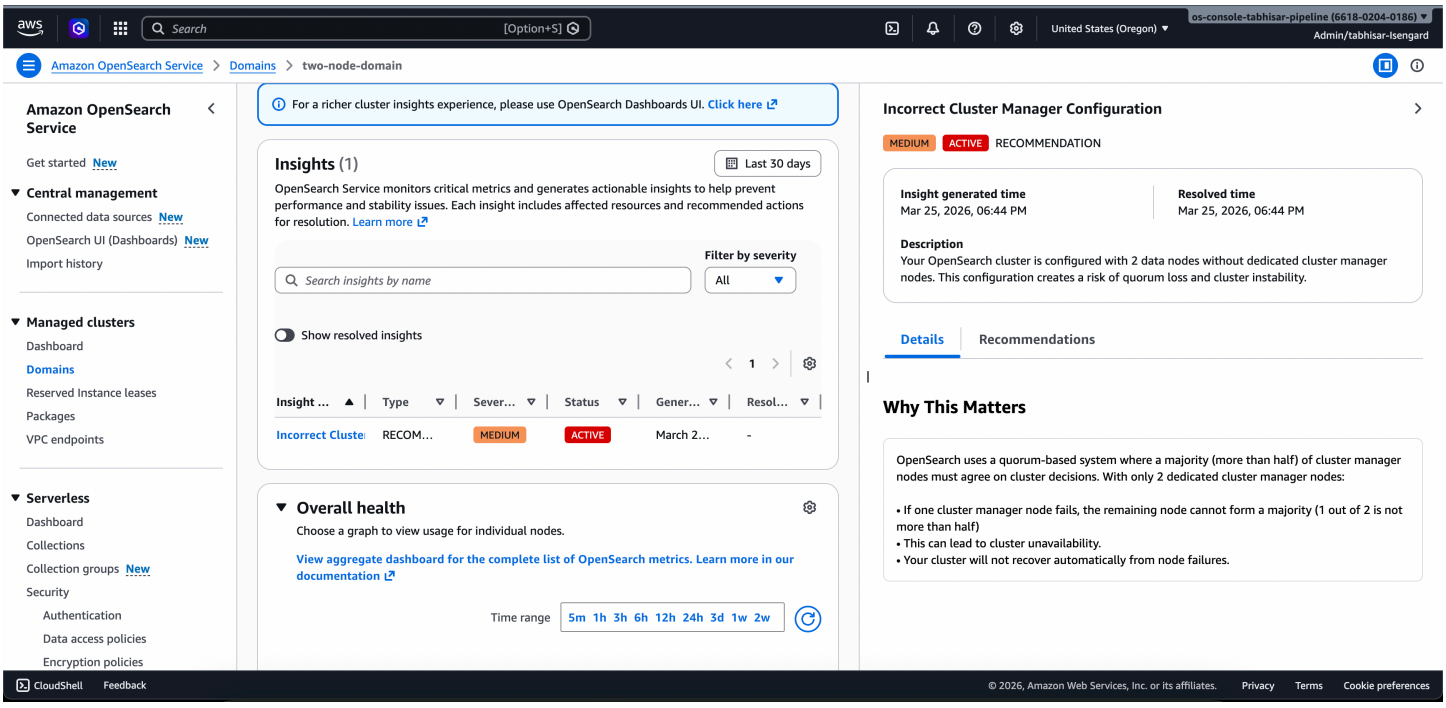
- 統一された可視性 - モニタリングデータを 1 つのダッシュボードに統合
- 実用的な推奨事項 - 問題解決のためのstep-by-stepガイダンスを取得する
- 包括的なカバレッジ - OpenSearch クラスター全体のセキュリティ、安定性、回復性をモニタリングする
- クエリの最適化 - リソースを大量に消費するクエリを特定し、パフォーマンスを最適化する

Cluster Insights を使用すると、最適なクラスターパフォーマンスを維持し、運用オーバーヘッドを減らし、OpenSearch クラスター全体で一貫したベストプラクティスを確保できます。

コンソールからクラスターインサイトにアクセスする

パフォーマンスと回復力に関する推奨事項を確認し、必要な設定変更をすべて同じコンソール内で行います。コンソールのクラスターヘルスタブで、アクティブなすべてのインサイトを一覧表示するクラスターインサイトにアクセスできます。任意のインサイトをクリックして、レコメンデーションを表示します。

Screen-1: Cluster Health タブの下の Cluster Insights



OpenSearch UI を使用して Cluster Insights と詳細なメトリクスにアクセスする

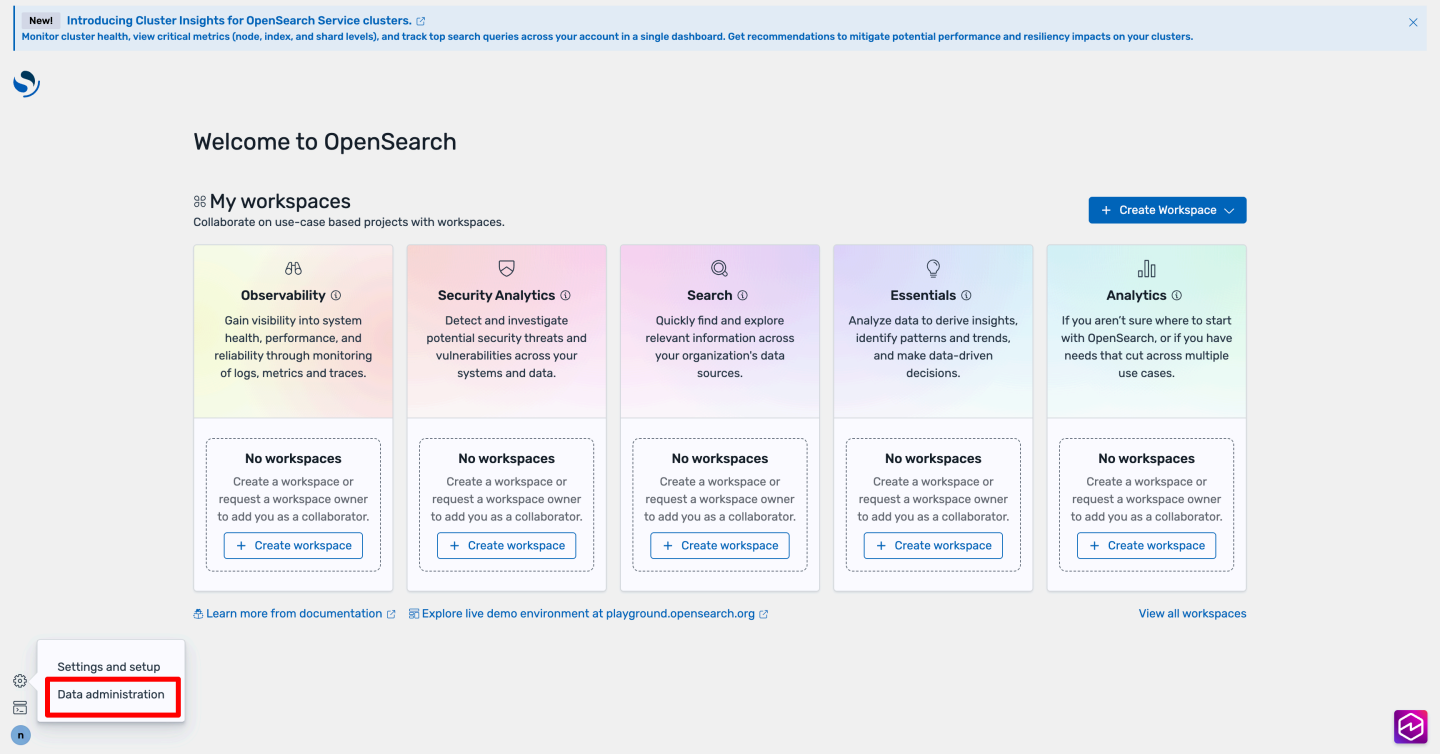
OpenSearch UI (ダッシュボード) を使用して、特定の OpenSearch Service クラスターのインサイトを表示できます。OpenSearch UI では、アプリケーションは単にフォルダのような組織構造です。各アプリケーションは、複数の OpenSearch Service クラスターに接続してインサイトを表示できます。Cluster Insights にアクセスするには、OpenSearch UI アプリケーションで管理ロールが必要です。

> **Note**
>
> Cluster Insights にアクセスするには、OpenSearch UI アプリケーションで管理ロールが必要です。

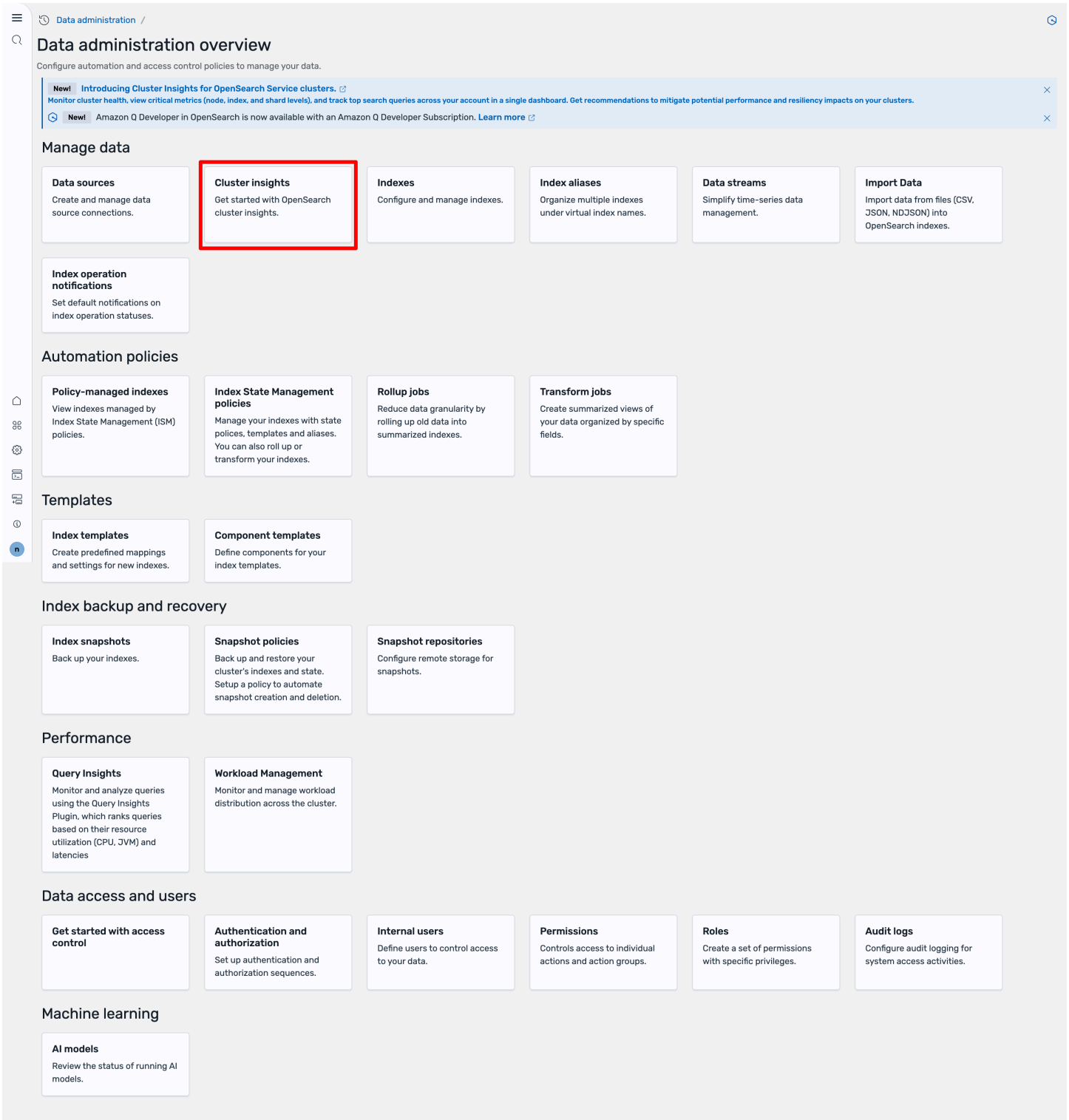
クラスターインサイトを表示するようにアプリケーションを作成して設定する

1. で OpenSearch Service コンソールを開きます。 https://console.aws.amazon.com/aos/home
2. 左側のナビゲーションから OpenSearch UI (ダッシュボード) を選択する
3. アプリケーションを作成して設定するには、次の手順を実行します。
    1. OpenSearch Service アプリケーションを作成する
    2. データソースの関連付け
4. これら 2 つのステップを完了すると、OpenSearch UI ダッシュボードで Cluster Insights を表示できます。設定アイコンを選択し、データ管理者 > クラスターインサイトを選択します。

Screen-2: OpenSearch UI からデータ管理者にアクセスする



Screen-3: データの管理セクションの下のクラスターインサイト



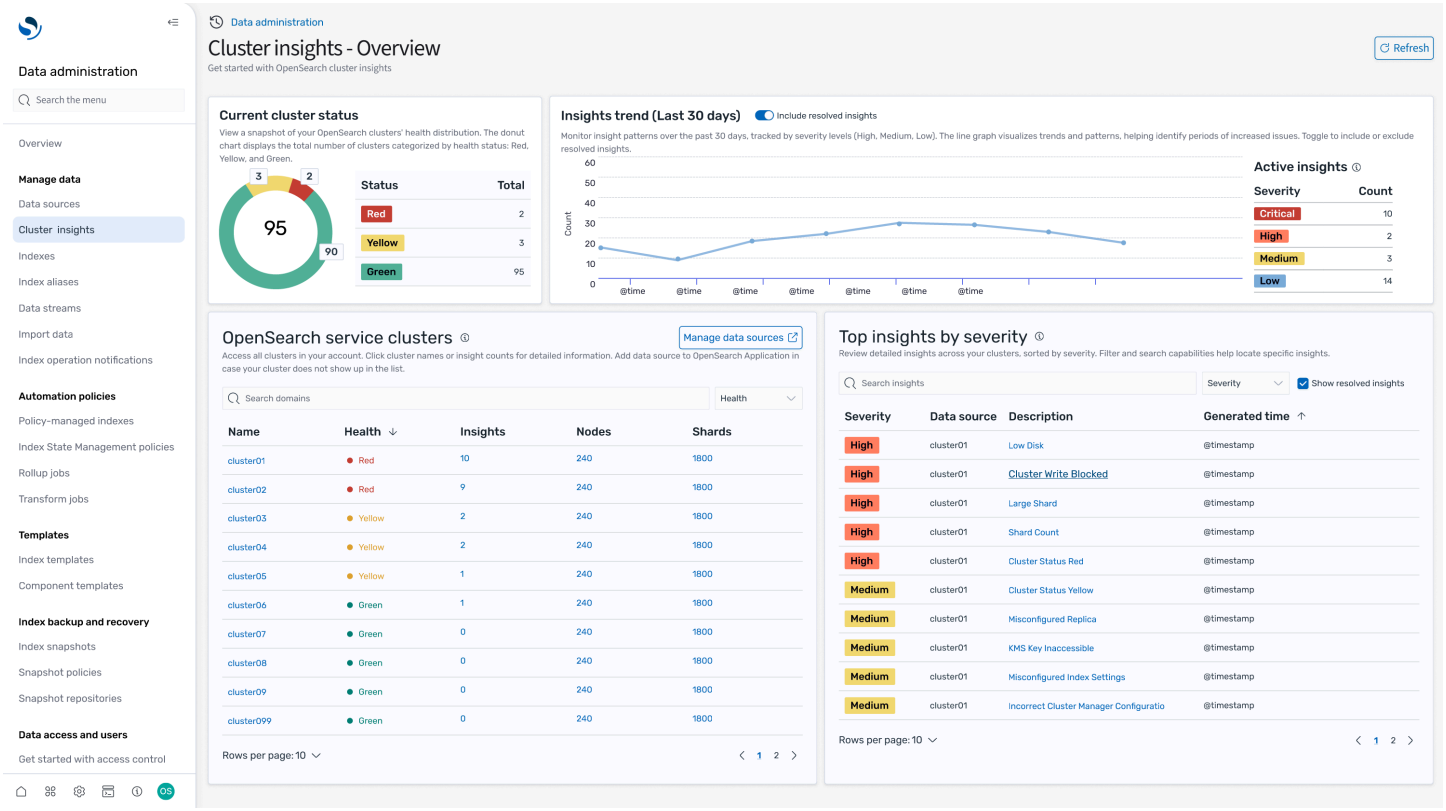
クラスターインサイトについて

このセクションでは、Cluster Insights で使用できるさまざまなインサイトについて説明します。

概要ダッシュボード

クラスターインサイトの概要ページは、次のスクリーンショットに示すように、アプリケーションレベルでクラスターの状態の概要を示し、以下のセクションで構成されています。

Screen-4: OpenSearch UI アプリケーションのクラスターインサイトランディングページ。



現在のクラスターステータス

ドーナツグラフには、クラスターのヘルスステータスが表示されます。

- 緑 - すべてのプライマリシャードとレプリカがノードに割り当てられています
- 黄 - すべてのプライマリシャードが割り当てられますが、一部のレプリカは割り当てられません
- 赤 - 少なくとも 1 つのプライマリシャードがノードに割り当てられていない

インサイトの傾向

トレンドグラフは、過去 30 日間の問題パターンを追跡し、新たな問題を特定し、解決の進行状況をモニタリングするのに役立ちます。

現在のオープンインサイト

過去 30 日間のオープンインサイトの重要度別に整理された数。

OpenSearch Service クラスター

このセクションでは、ノード数、シャード数、アクティブなクエリなどの主要な統計を含むすべての OpenSearch クラスターを一覧表示します。

重要度別の上位のインサイト

アプリケーション内のすべてのドメインでインサイトを確認できます。このセクションでは、即時対応が必要な問題 (重大度が高い、重要度が高い) を優先します。各インサイトには説明と特定のレコメンデーションが含まれており、重要な問題に最初に集中するのに役立ちます。

インサイトの詳細

重要度別の上位インサイトセクションの各インサイトはインタラクティブで、詳細な分析を提供します。たとえば、ラージシャードサイズのインサイトを選択する場合:

1. しきい値を超えるシャードの数と影響を受けるインデックスが表示されます。
2. リソースマップは、インデックス、ID、現在のサイズでオーバーサイズの各シャードを識別します。
3. レコメンデーションタブにはstep-by-step修復ガイダンスが表示されます。
4. 履歴タブには、リソース修復アクションのタイムラインが表示されます。

クラスターの詳細

OpenSearch Service クラスターセクションで特定のクラスターを選択すると、OpenSearch はクラスターの状態、ノードビュー、インデックスビュー、シャードビュー、クエリビューのタブにそのクラスターに関するインサイトを表示します。クラスターのヘルスタブには、次の情報が表示されます。

概要

キー情報には、クラスターの状態、シャード数、ノード数、インデックス数、ドキュメント統計が含まれます。

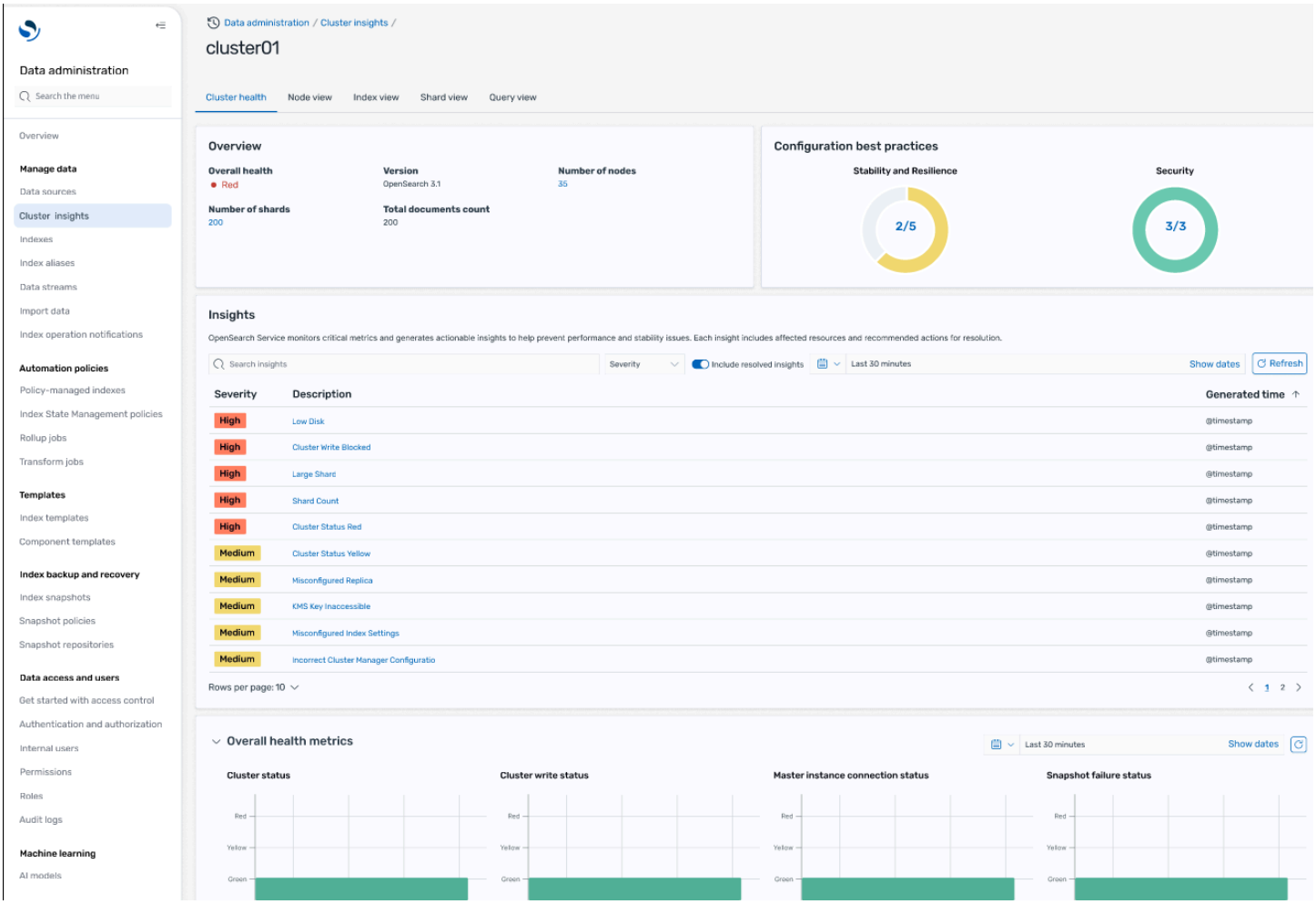
設定のベストプラクティス

ドーナツグラフは、回復力とセキュリティに関する推奨設定への準拠を示しています。

Insights

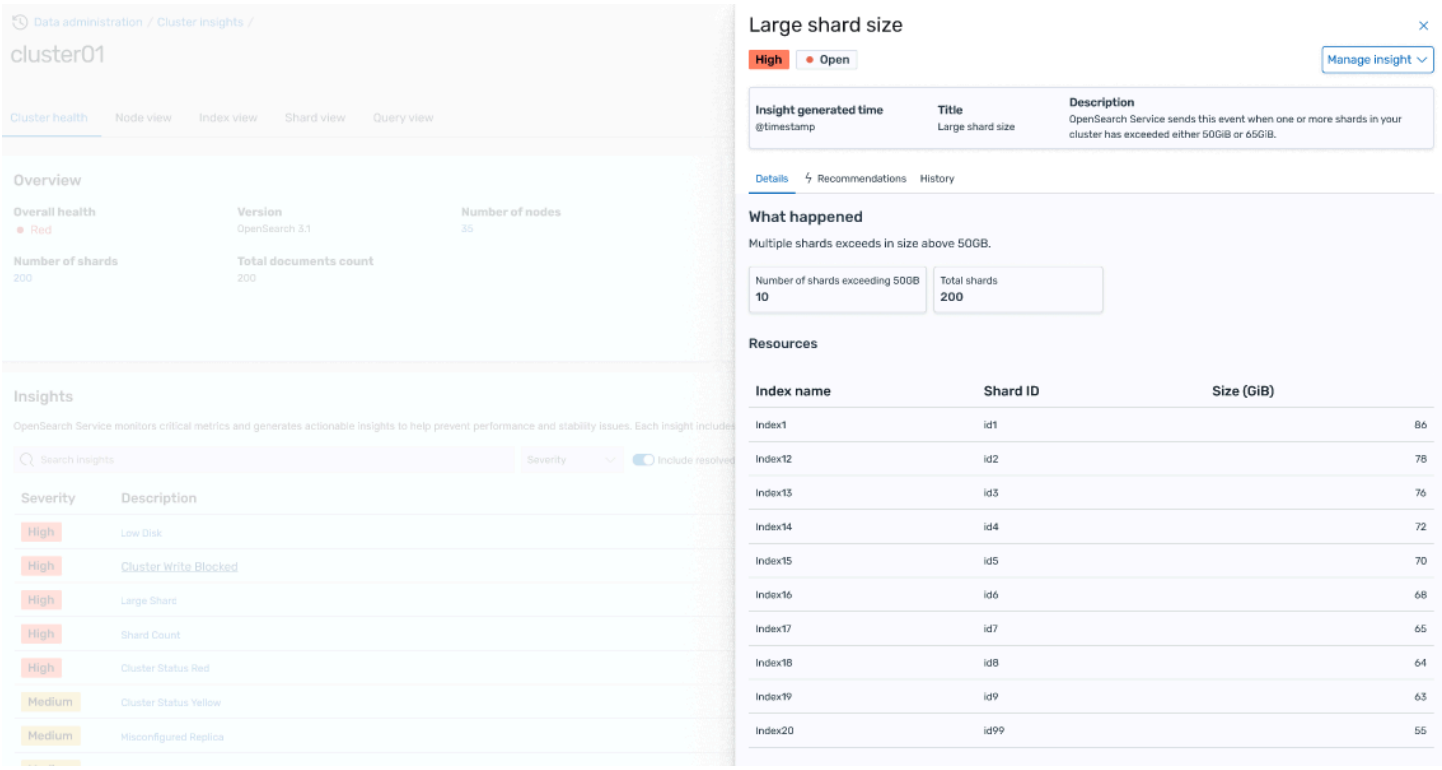
表には、クラスター用に生成された最近のインサイトと、概要ページから利用できる同じ詳細な内訳と修復ガイダンスが一覧表示されています。

Screen-5: クラスターヘルスの概要には、主要なメトリクス、ベストプラクティス、インサイトが記載されています。



インサイトをクリックすると、詳細と影響を受けるリソース、推奨事項を確認できます。さらに、固定リソースの履歴を表示することもできます。

Screen-6: インサイトの詳細。詳細、推奨事項、履歴タイムラインを提供します。



メトリクスセクション

このセクションのインタラクティブグラフには、次のクラスターメトリクスが表示されます。

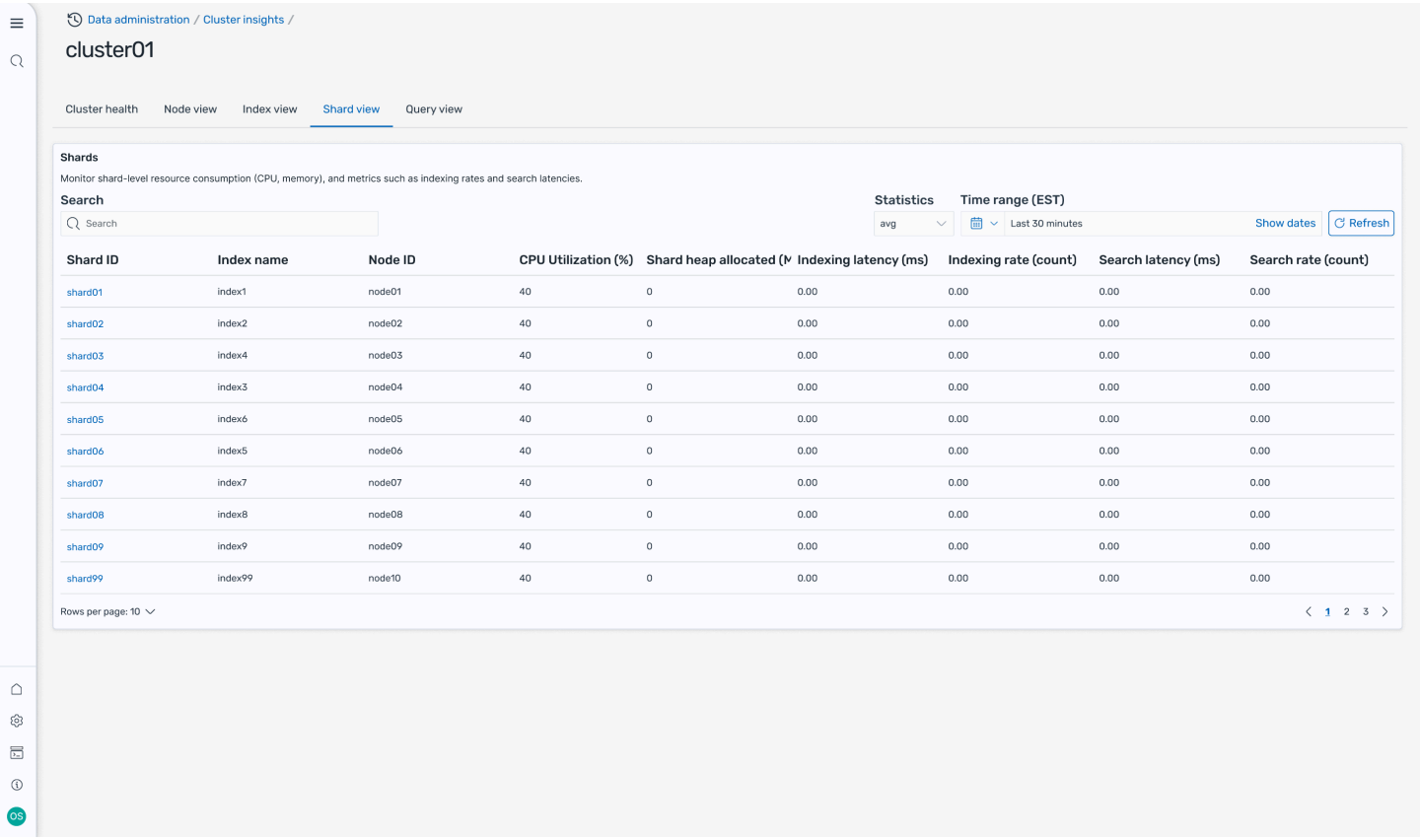
- クラスターステータス、書き込みステータス、検索可能なドキュメントなどの全体的なクラスターヘルスメトリクス
- インデックス作成や検索のレートやレイテンシーなどの KPIs (主要業績評価指標)
- JVM や CPU 使用率などのリソース使用率メトリクス

ノード、インデックス、シャードビュー

ノード、インデックス、シャードビューは OpenSearch 統計を使用して、クラスターオペレーションを詳細に可視化します。 次を表示できます。

- CPU 使用率や JVM メモリ負荷などのリアルタイムメトリクス
- パフォーマンスデータの検索とインデックス作成
- クラスターコンポーネント間のリソースホットスポット
- 詳細なノードレベルの診断
- 上位シャードヒープが割り当てられました

Screen-7: ノード、インデックス、シャードレベルのメトリクス



クエリビュー

> **Note**
>
> クエリビュー機能は、OpenSearch バージョン 2.19 以降でサポートされています。

クエリビューページは、以下を使用してリソースを大量に消費するクエリをモニタリングするのに役立ちます。

アクセス設定

Top N クエリを表示するには、きめ細かなアクセスコントロールのアクセス許可が必要です。以下のことを確認してください。

- きめ細かなアクセスコントロールがドメインで有効になっています。
- IAM ロール (または内部ユーザー) は、クエリインサイトに必要なクラスターアクセス許可を持つ OpenSearch ロールにマッピングされます。

- 完全な管理者アクセスのために、IAM ロール ARN をバックエンドロールとして all_access ロールと security_manager ロールの両方にマッピングします。これは、OpenSearch Dashboards の Security > Roles > Select the role > Mapped users > Manage mapping、または Security API (PUT _plugins/_security/api/rolesmapping/all_access) を使用して行うことができます。

適切なロールマッピングがないと、ユーザーはクエリインサイトデータにアクセスしようとすると 403 Forbidden エラーを受け取る可能性があります。詳細については、「きめ細かなアクセスコントロール」を参照してください。

ライブダッシュボード

すべてのクエリの実行統計、CPU とメモリの使用状況、完了の進行状況を表示します。

上位 N 件のクエリ

ランク付けされたテーブルには、次のような詳細を含む最も重要なクエリが表示されます。

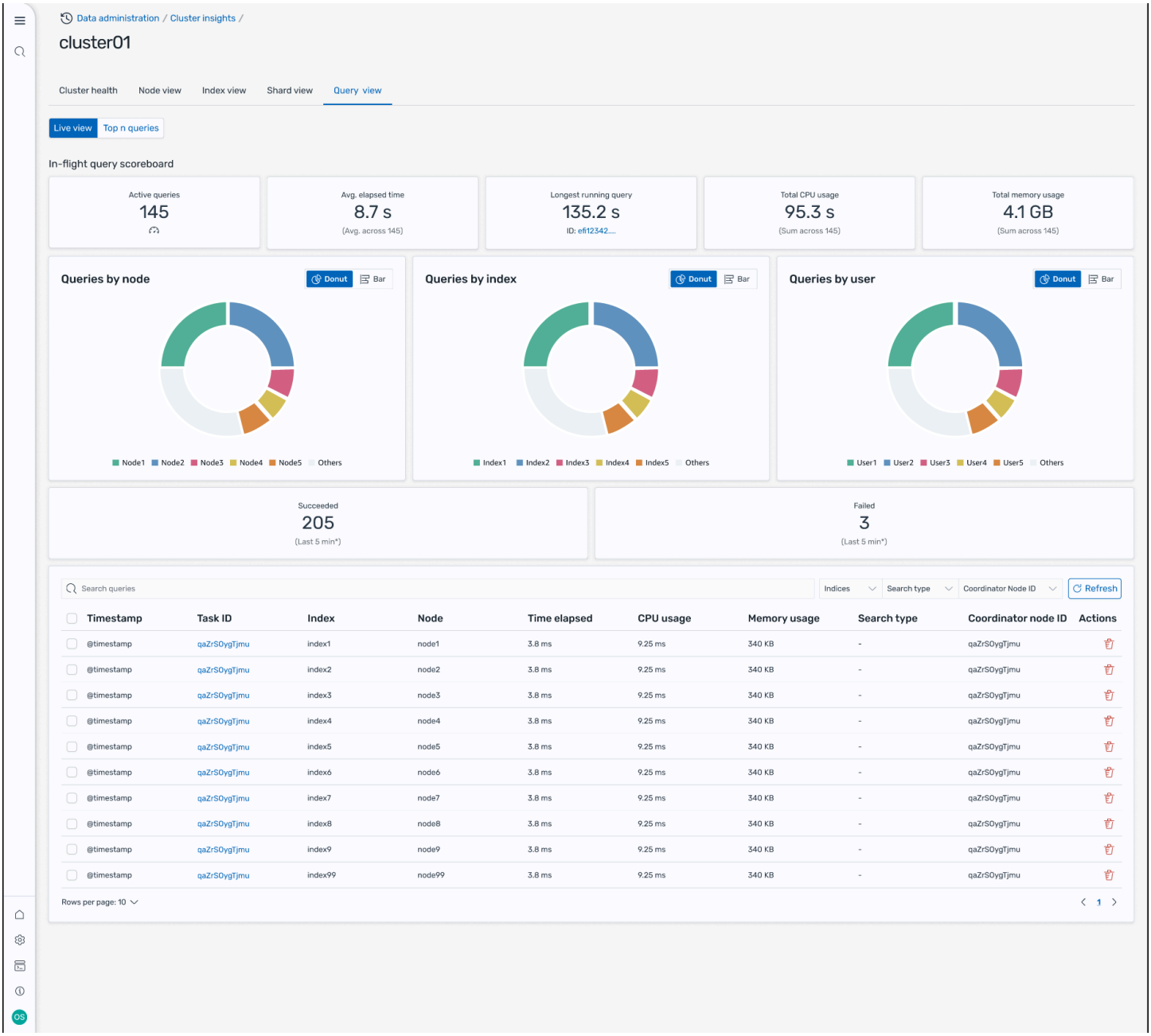
- クエリ数
- レイテンシー、CPU、メモリ使用量
- 検索タイプとコーディネーターノード
- ターゲットインデックスとシャード数

クエリの詳細

クエリをダブルクリックすると、以下が表示されます。

- 正確なクエリペイロードと実行ステップ
- 各フェーズのレイテンシーの内訳 (展開、クエリ、フェッチ)
- 最適化に関する推奨事項

Screen-8: インフライトライブビュー。Top-N クエリを表示することもできます。



Amazon EventBridge イベントを通じて Insights にアクセスする

Amazon EventBridge イベントを通じてインサイトをモニタリングできます。詳細については、通知を確認してください。

Insights Catalog

Cluster Insights は、影響を受けるクラスターに関する以下の実用的なインサイトを提供します。

Insights	説明
低ディスク	クラスターの FreeStorage が総容量の 25% または 25GB のいずれかのしきい値を最初に下回ると、重要度の高いインサイトが生成されます。
クラスター書き込みがブロックされました	クラスターの FreeStorage が総容量の 20% または 20GB のいずれかのしきい値を最初に下回ると、重大度に関するインサイトが生成されます。
大きなシャード	シャードサイズが 50GB ~ 100GB の場合、重要度が低いインサイトが生成されます。重大度は、シャードサイズが 100GB から 200GB0 GB の間になると中に上がり、シャードサイズが 200GB を超えるとさらに高に上がります。
シャード数	重要度の高いインサイトは、クラスターが設定されたヒープの GB あたり 25 シャードを超えた場合、またはノードあたりのシャード数が 900 を超えた場合に生成されます。OpenSearch バージョン 2.17 以降では、このインサイトは、このドキュメントで説明されているクラスターマネージャーの設定に基づいて生成されます。
クラスターステータス 赤	OpenSearch クラスターでパフォーマンスが低下したノードを置き換えると、重大度に関するインサイトが生成され、クラスターのステータスが赤になります。これには、クラスターをグリーン状態に復元するための即時のアクションが必要です。
クラスターステータス黄	OpenSearch クラスターでデグレードしたノードを置き換えると、重要度が中程度のインサイトが生成され、クラスターのステータスが黄になります。
設定ミスのレプリカ	中重要度インサイトは、クラスターにレプリカが設定されていない場合、またはレプリカの数がデータノードの総数を超えた場合に生成されます。
KMS キーにアクセスできません	OpenSearch クラスターが保管時の暗号化に使用される KMS キーにアクセスできなくなると、重大度に関するインサイトが生成されます。これにより、クラスターが暗号化されたデータにアクセ

Insights	説明
	スできなくなり、クラスターオペレーションに影響を与え、アクセスを復元するための即時のアクションが必要になります。
誤った設定のインデックス設定	中重要度インサイトは、インデックス設定「index.routing.allocation.require/include/exclude.<attribute>」が OpenSearch クラスター内の任意のインデックスに存在する場合に生成されます。この設定により、クラスターの更新中のブルー/グリーンデプロイの成功を防ぐことができます。
クラスターマネージャーの設定が正しくない	OpenSearch クラスターに専用クラスターマネージャーノードのないデータノードが 2 つ、または専用クラスターマネージャーノードが 2 つある場合、重大度に関するインサイトが生成されます。これらの設定により、クォーラムが失われ、クラスターが不安定になるリスクが生じます。
最適ではないシャーディング戦略	(a) 1 つ以上のインデックスのシャード数がデータノードの 80% 未満であることを示す中程度の重要度のインサイトが生成されます。これにより、クラスター全体のワークロード分散が最適ではない可能性があります。 (b) ノードあたりの平均シャードサイズよりも 20% 多くのデータを持つシャードが 1 つ以上のインデックスにあることを示す中程度の重要度のインサイトが生成されます。これにより、クラスター全体のワークロード分散が最適ではない可能性があります。
クラスターの過負荷	(a) クラスターの過負荷に関する重要度の高いインサイトは、クラスター内のデータノードの 70% 以上が CPU、メモリ、ディスク I/O、ディスクスループット、またはディスク使用率で使用率が高いことを示しています。これは、クラスターが容量制限に近づいている可能性があることを示します。 (b) 重大度インサイトは、データノードの 70% 以上が CPU、メモリ、ディスク I/O、ディスクスループット、ディスク使用率でリソース使用率が高く、リクエストのスロットリングまたは拒否が観察されていることを示しています。これには、クラスターをスケーリングするための即時のアクションが必要です。

Insights	説明
未使用のインデックス	このインサイトは、OpenSearch クラスター内の 1 つ以上のインデックスで過去 30 日間に検索アクティビティとインデックス作成アクティビティがゼロになったときにトリガーされます。未使用のインデックスはストレージを消費し、シャード数を増やし、不要なクラスターオーバーヘッドを追加します。インサイトは、影響を受けるインデックス名を表示し、ドメイン全体のインデックス数に対する未使用のインデックスの合計数を表示し、それらの合計ストレージフットプリントを表示します。お客様がリソースを再利用してコストを削減できるように、実用的な推奨事項が用意されています。
EBS gp3 過剰プロビジョニング	このインサイトは、ドメインの gp3 EBS ボリュームが過去 30 日間にわたって一貫して十分に活用されておらず、平均 IOPS とスループットの両方がプロビジョニングされた値の 50% 未満であり、その期間中にスロットリングイベントが観察されていない場合に生成されます。このレコメンデーションには、現在のプロビジョンド IOPS 値とスループット値に加えて、実際の使用量に合った低コストの設定目標が含まれており、パフォーマンスに影響を与えずに EBS コストを削減できます。

Amazon OpenSearch Service の可観測性

オブザーバビリティとは、複雑なシステムの内部状態とパフォーマンスを出力を調べて理解する方法です。従来のモニタリングでは、システムがダウンしていることがわかります。オブザーバビリティは、テレメトリデータに関する新しい質問をさせて、その理由を理解するのに役立ちます。

Amazon OpenSearch Service が提供するもの

Amazon OpenSearch Service は、3 種類のテレメトリデータを収集、相関、視覚化することで、統一されたオブザーバビリティソリューションを提供します。

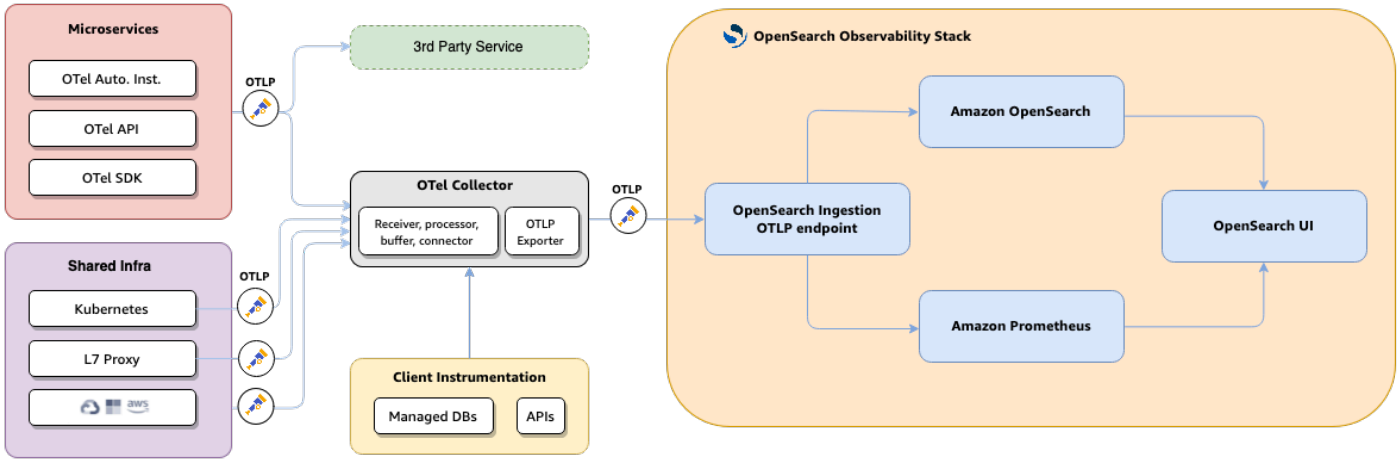
- ログ – イベントのタイムスタンプ付きレコード。
- トレース – End-to-endジャーニー。
- メトリクス – Amazon Managed Service for Prometheus の直接クエリ統合を通じて、システムの状態を表す時系列データ。

これらを 1 つのインターフェイスに統合することで、Amazon OpenSearch Service は運用チーム、SREs、開発者が問題を迅速に検出、診断、解決するのに役立ちます。

オブザーバビリティに対する Amazon OpenSearch Service のアプローチ

Amazon OpenSearch Service は、3 つの主要な方法で差別化します。

- OpenTelemetry-native with OpenSearch Ingestion as the last mile – 計測と収集のために OTel を標準化します。Amazon OpenSearch Ingestion は、インデックス作成前にテレメトリデータをフィルタリング、強化、変換、ルーティングするフルマネージドパイプラインとして機能します。
- OpenSearch UI の統合ログ、トレース、メトリクス – 1 つのオブザーバビリティワークスペースから 3 つのシグナルタイプをすべて分析します。スロートレースをアプリケーションログに関連付けるか、サービスダッシュボードに Prometheus メトリクスをオーバーレイします。
- 目的主導型のクエリ言語 – ログとトレースには Piped Processing Language (PPL)、メトリクスには PromQL を使用します。各言語はシグナルタイプに合わせて最適化されているため、表現的クエリを妥協することなく実行できます。



> **Note**
>
> このセクションで説明するオブザーバビリティ機能は、OpenSearch UI でのみ使用できます。OpenSearch Dashboards では使用できません。新しいオブザーバビリティワークロードの場合は、オブザーバビリティワークスペースOpenSearch UIで を設定することをお勧めします。

はじめに

オブザーバビリティスタックを で実行 AWS し、数分でテレメトリデータの送信を開始します。

クイックスタート

end-to-endのオブザーバビリティスタックを にデプロイする最も簡単な方法は、 CLI インストーラ AWS です。次のリソースを作成します。

- Amazon OpenSearch Service ドメイン
- Amazon Managed Service for Prometheus ワークスペース
- Amazon OpenSearch Ingestion パイプライン
- オブザーバビリティワークスペースを備えた OpenSearch UI アプリケーション

オプションで、インストーラはサンプルのテレメトリ用に OpenTelemetry Demo を使用して EC2 インスタンスを起動します。

次のコマンドを実行してインストールを開始します。

```
bash -c "$(curl -fsSL https://raw.githubusercontent.com/opensearch-project/
observability-stack/main/install.sh)" -- --deployment-target=aws
```

インストールには約 15 分かかります。

CDK デプロイ

infrastructure-as-code には、 AWS CDK を使用します。CDK デプロイでは、次の 2 つのスタックが作成されます。

スタック	作成内容	デプロイ所要時間
ObsInfra	OpenSearch ドメイン、Amazon Managed Service for Prometheus ワークスペース、直接クエリデータソース、パイプライン IAM ロール	~17 分
ObservabilityStack	きめ細かなアクセスコントロールマッピング、OpenSearch Ingestion パイプライン、OpenSearch UI アプリケーション、ダッシュボードの初期化、デモワークロード (オプション)	~6 分

次のコマンドを実行してデプロイします。

```
cd aws/cdk
npm install
cdk deploy --all
```

詳細については、GitHub の CDK デプロイ README を参照してください。

テレメトリの送信

どちらのデプロイ方法でも、OTLP データを受け入れる OpenSearch Ingestion エンドポイントが作成されます。SigV4 認証を使用してエクスポートするように OTel Collector を設定します。

```
extensions:
  sigv4auth:
    region: us-west-2
    service: osis
```

```
exporters:
  otlphttp/logs:
    logs_endpoint: ${OSIS_ENDPOINT}/v1/logs
    auth: { authenticator: sigv4auth }
    compression: none
  otlphttp/traces:
    traces_endpoint: ${OSIS_ENDPOINT}/v1/traces
    auth: { authenticator: sigv4auth }
    compression: none
  otlphttp/metrics:
    metrics_endpoint: ${OSIS_ENDPOINT}/v1/metrics
    auth: { authenticator: sigv4auth }
    compression: none
```

> **Note**
>
> データを送信する IAM プリンシパルには、パイプライン ARN に対する `osis:Ingest`と アクセス`aps:RemoteWrite`許可が必要です。

詳細情報

テレメトリデータの送信の詳細については、次のリソースを参照してください。

- OpenTelemetry 計測ガイド (言語ごと)
- インフラストラクチャモニタリング (AWS、Docker、Kubernetes、Prometheus)
- OTel Collector の設定
- データパイプラインとバッチ処理
- このガイドの「Amazon OpenSearch Ingestion」

MCP アプリによるエージェントのオブザーバビリティ

OpenSearch は、確認可能な MCP アプリレンダリングビジュアライゼーションを使用して、オブザーバビリティワークフローをエージェント IDE に提供する MCP アプリのエージェントオブザーバビリティエクスペリエンスを提供します。このエクスペリエンスにより、OpenSearch UI でのウェブページベースのオブザーバビリティエクスペリエンス (ログ、トレース、メトリクス、アラート) が強化されます。AI エージェントは、OpenSearch ドメインとコレクションに保存されているデー

タを使用してインシデントを調査できます。MCP アプリでは、各 MCP ツール呼び出しは、エージェントが推論するテキスト概要と、同じ会話スレッドで検証するインタラクティブな視覚化の両方を返します。

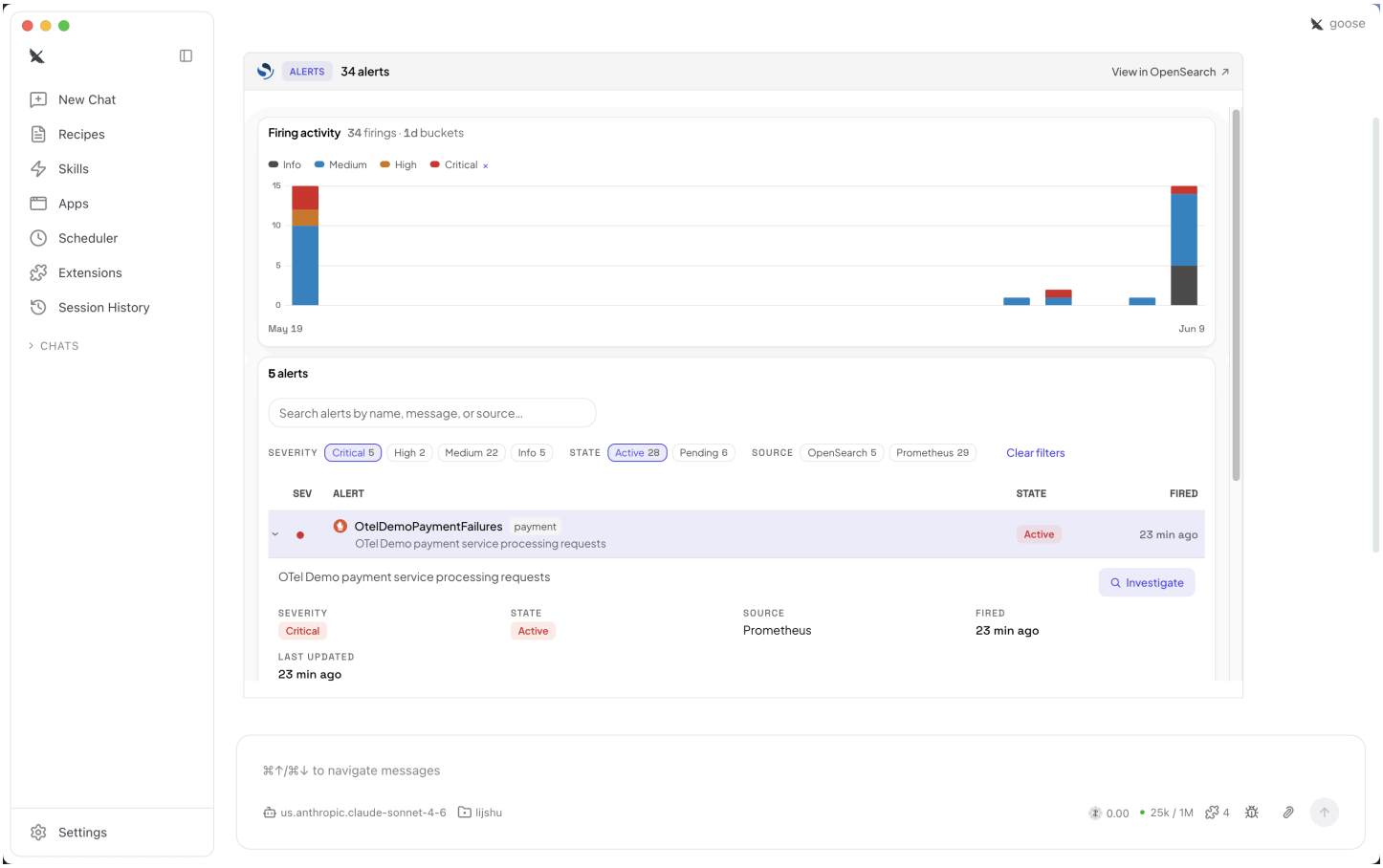
トピック

- the section called “MCP アプリの例”
- the section called “主要なコンセプト”
- the section called “前提条件”
- the section called “MCP サーバーのセットアップ”
- the section called “エージェントオブザーバビリティに利用可能な MCP アプリ”
- the section called “mcpb ファイルの検証”

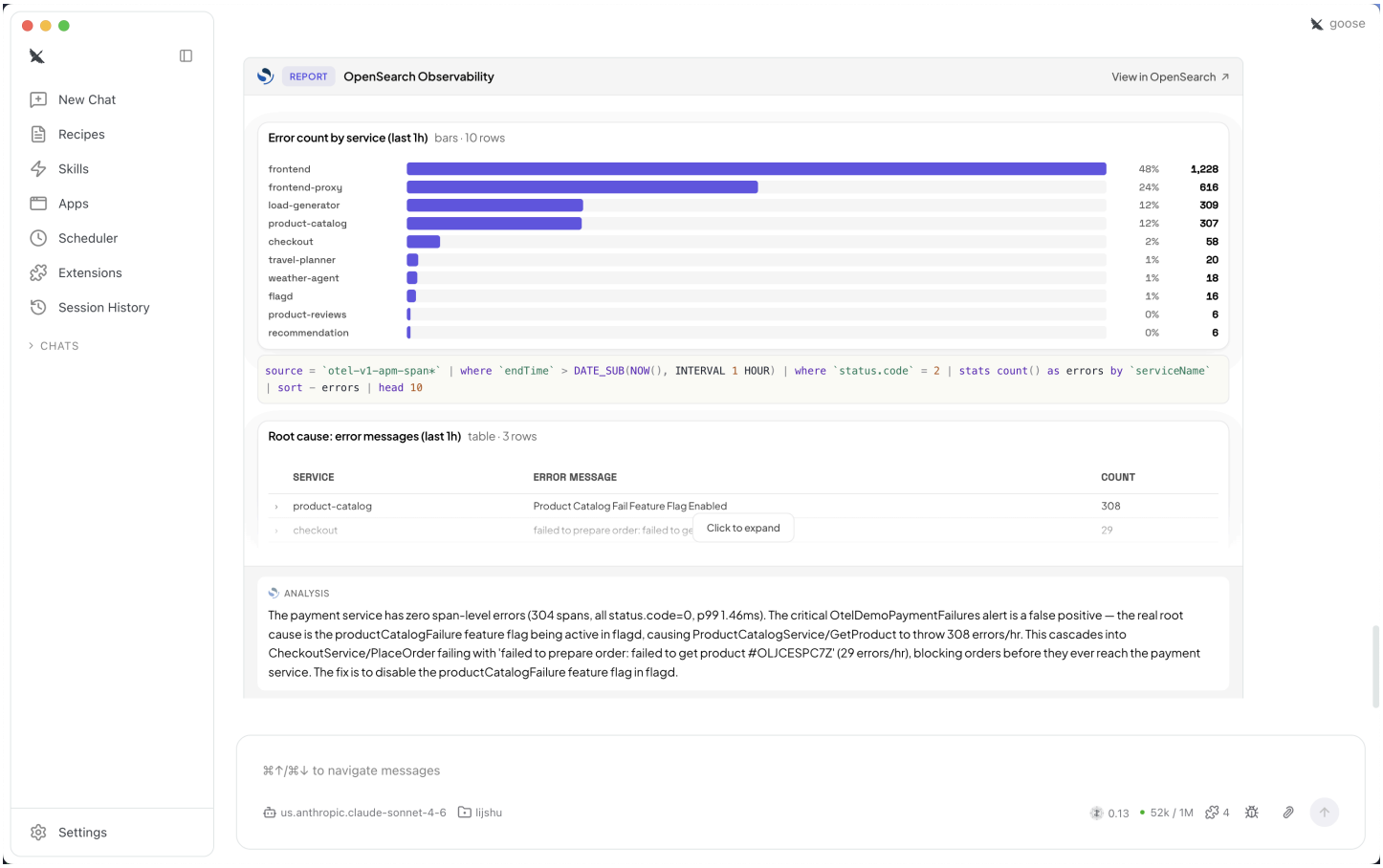
MCP アプリの例

次の例は、インシデントの調査中に MCP Apps が互換性のあるエージェント IDE 内でインタラクティブビジュアライゼーションをレンダリングする方法を示しています。

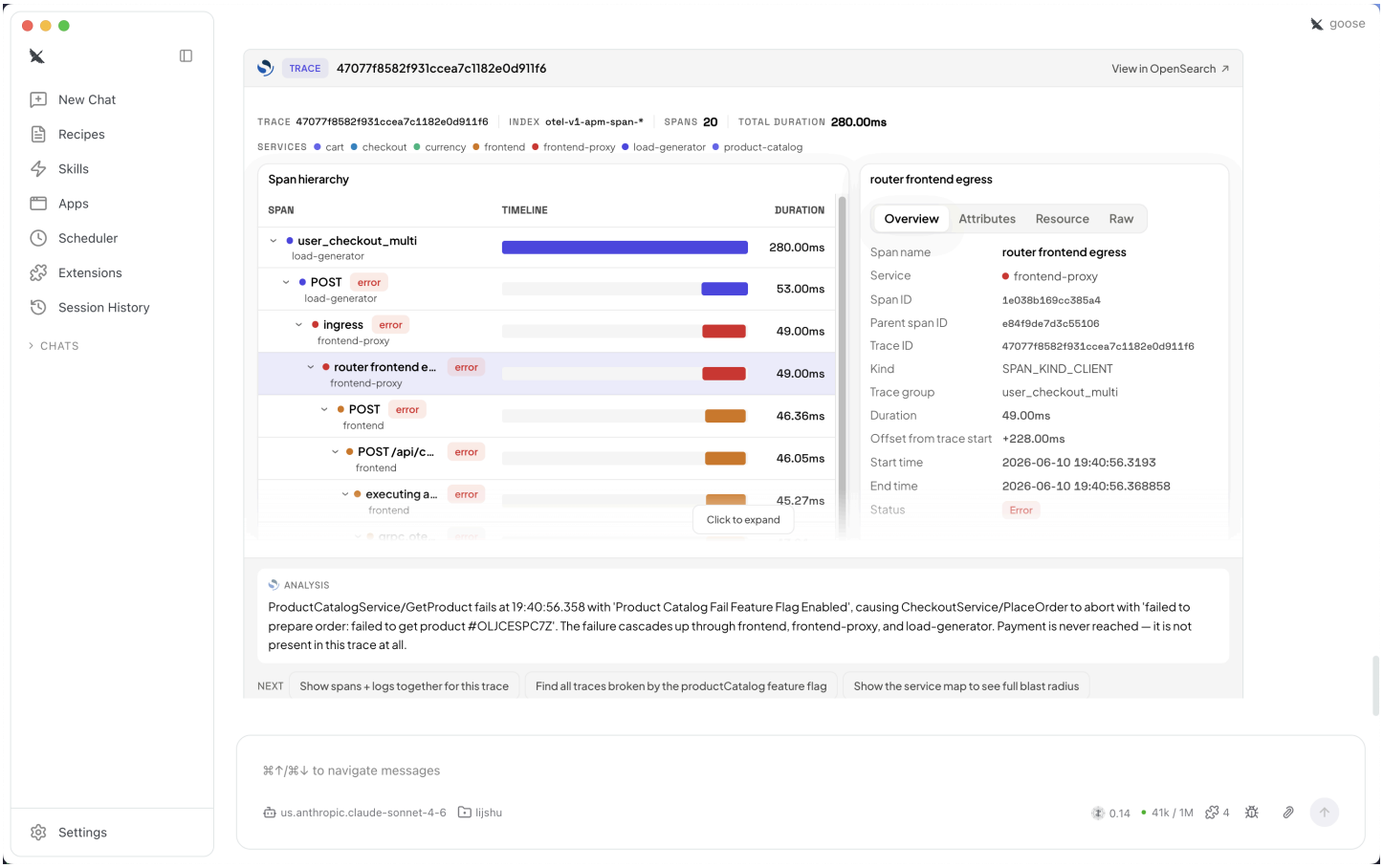
アラート – データソース全体で発生したアラートを表示してトリアージします。アラートビューには、時間の経過に伴う発射アクティビティ、重要度の内訳、アラートごとの詳細が表示され、調査を開始するオプションがあります。次のスクリーンショットは、発射アクティビティを含むアラートビューを示しています。



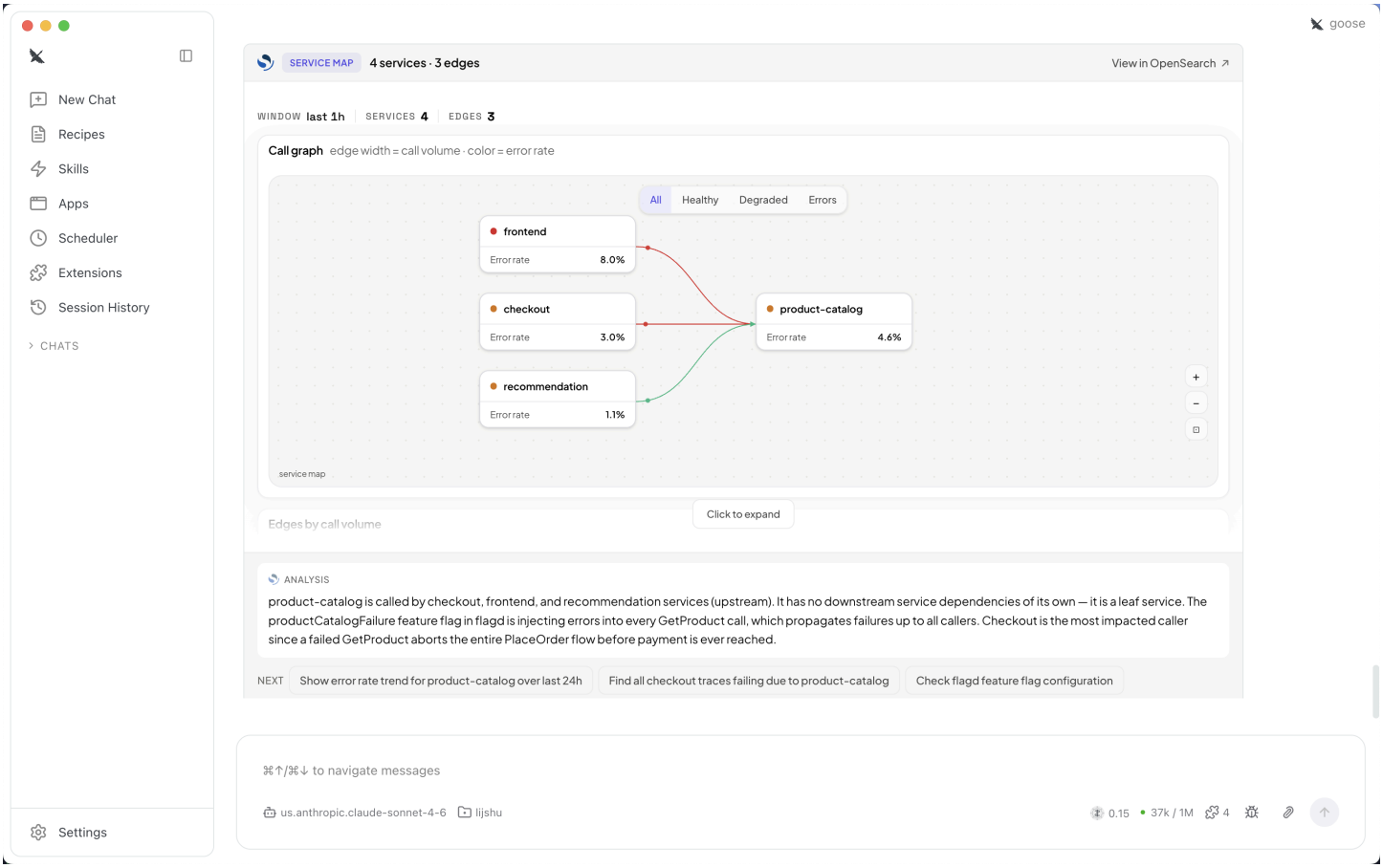
オブザーバビリティレポート – 基盤となるクエリや AI 生成の根本原因分析とともに、サービスごとのエラー数などの集計結果を確認します。次のスクリーンショットは、オブザーバビリティレポートを示しています。



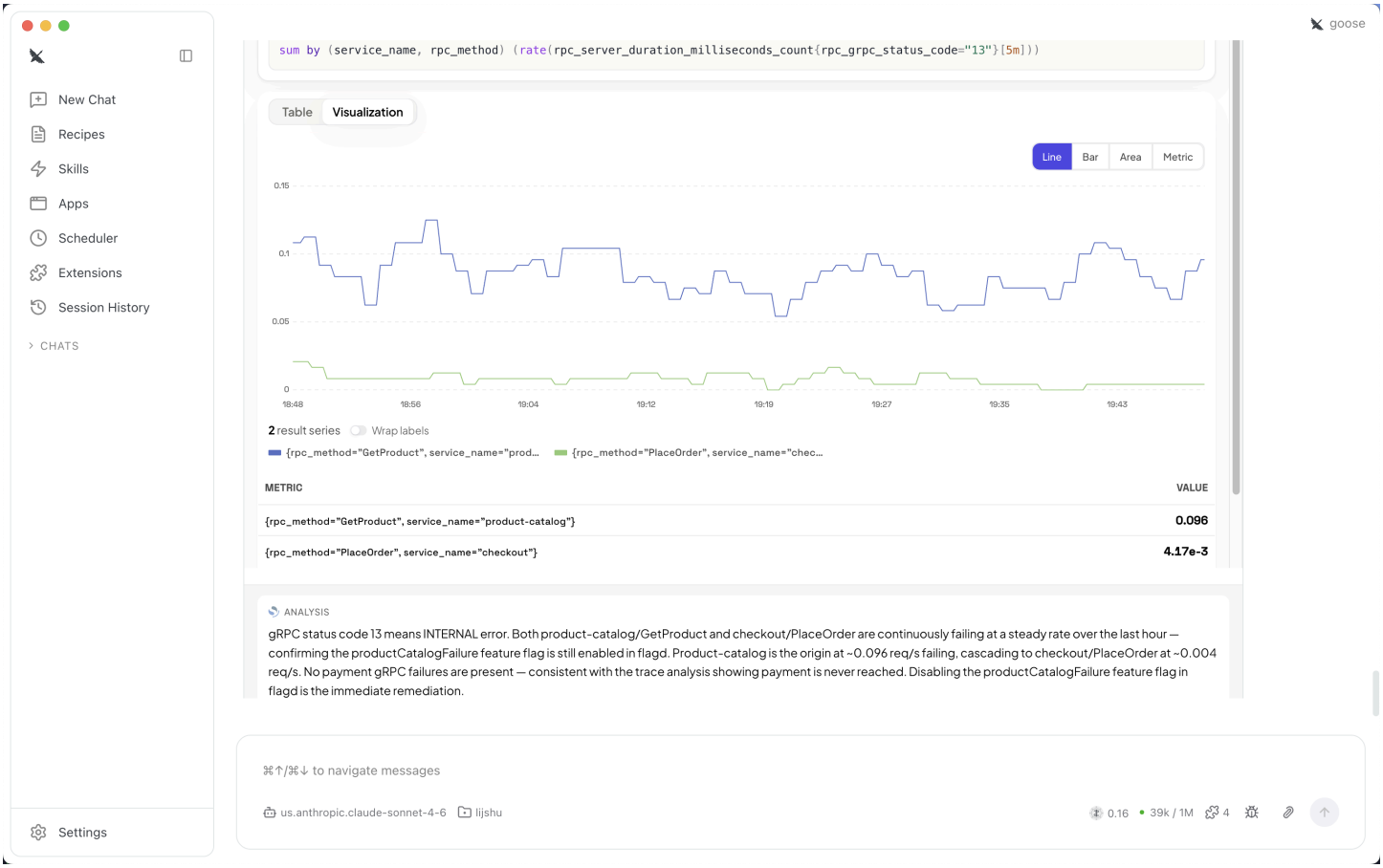
トレース調査 – 分散トレースのスパン階層とタイムラインを調べ、個々のスパンの詳細を検査し、障害が発生した場所の分析を読み取ります。次のスクリーンショットは、トレース調査ビューを示しています。



サービスマップ – サービスの依存関係を呼び出しグラフとして視覚化します。エッジ幅は呼び出し量を反映し、色はエラー率を反映して、インシデントの爆発半径を理解します。次の図は、サービスマップを示しています。



メトリクスの視覚化 – PromQL クエリ結果をインタラクティブな折れ線、棒、面積、またはメトリクスグラフとしてレンダリングし、結果シリーズとメトリクスの動作の分析を行います。次のスクリーンショットは、メトリクスの視覚化を示しています。



主要なコンセプト

OpenSearch UI

OpenSearch UI は、Amazon OpenSearch Service のマネージド型のクラウドベースのダッシュボードおよび運用分析エクスペリエンスです。統合された単一エンドポイントインターフェイスを提供し、複数のデータソースのデータを調査、クエリ、視覚化します。MCP アプリを介して生成された視覚化は、OpenSearch UI アプリケーションから取得されます。

ローカル MCP サーバー

ローカル MCP サーバーは、エージェント IDE と OpenSearch UI アプリケーション間の安全な双方向ブリッジとして機能するコンピュータ上で直接実行されるプログラムです。MCP サーバーには、互換性のある IDE 内で UI をレンダリングする MCP アプリツールなど、AI エージェントが使用できる複数のツールが含まれています。以下のセクションの手順に従って、MCP アプリ機能を備えたローカル MCP サーバーをインストールします。

MCP アプリ

Claude Desktop などの MCP ホスト内でレンダリングするインタラクティブ UI アプリケーション。OpenSearch エージェントオブザーバビリティエクスペリエンスでは、MCP アプリは、SLO またはトレースに関する視覚化を表示するなど、単一のオブザーバビリティジョブを実行するコンポーザブルユニットです。

デュアルレスポンス

MCP Apps では、互換性のある IDE で 1 回のツール呼び出しで 2 つの出力を同時に返すようになりました。オブザーバビリティエージェントに期待されるコンパクトテキストの概要と、同じ会話でレンダリングされたインタラクティブな視覚化により、結果を簡単に確認および検証できます。

前提条件

- オブザーバビリティワークスペースが作成され、少なくとも 1 OpenSearch つのデータソース (OpenSearch ドメイン、コレクション、または Amazon Managed Prometheus) に接続されている OpenSearch UI アプリケーション OpenSearch
- MCP アプリをサポートするエージェント IDE (Claude Desktop、VS Code GitHub Copilot、Gose、ChatGPT、Cursor)
- オブザーバビリティの概念 (ログ、トレース、メトリクス) と IDE の使用に関する基本的な知識
- Node.js 22 以降がローカルマシンにインストールされている
- AWS OpenSearch UI アプリケーションにアクセスするためのアクセス許可で設定された 認証情報。認証情報では、 `es:ESHttpGet`および `es:ESHttpPost`アクションを許可する必要があります。
- (オプション) 本番稼働用データなしでテストする場合は、サンプルデータでデプロイされた OpenTelemetry デモアプリケーション

MCP サーバーのセットアップ

mcpb ファイルを使用して MCP サーバーを設定するには

1. OpenSearch オブザーバビリティ MCP サーバーファイルをダウンロードします。オプションで、検証のために署名ファイルをダウンロードすることもできます。
2. ダウンロードした ファイルを解凍します。

3. mcpb ファイルを開き、互換性のある IDE をトリガーして設定フローを開始します。ページの指示に従ってください。

または、mcpb ファイルを開いても機能しない場合は、server.js ファイルをリファレンスとして使用し、IDE の拡張機能で MCP サーバーを手動で設定できます。

MCP サーバーを手動で設定するには

1. 選択した IDE を開き (MCP アプリと互換性があることを確認してください)、拡張機能または MCP セクションに移動します。新しい設定の追加 を選択します。
2. 設定ページで、次の値を更新します。
   - "args" 値を実際の server.js ファイルパスに置き換えます。
   - OS_UI_ENDPOINT を OpenSearch UI アプリケーションの URL に置き換えます。
   - リージョンを OpenSearch UI アプリケーションのリージョンに置き換えます。

次の例は、MCP サーバーの設定を示しています。

```
{
  "mcpServers": {
    "opensearch-observability-stack-mcp": {
      "command": "node",
      "args": ["/path/to/opensearch-observability-stack-mcp/server/server.js"],
      "env": {
        "OS_UI_ENDPOINT": "application-foo-bar.us-west-2.opensearch.amazonaws.com",
        "AWS_REGION": "us-west-2",
        "AWS_PROFILE": "my-profile"
      }
    }
  }
}
```

- `/path/to/opensearch-observability-stack-mcp/server/server.js` – を server.js ファイルへのパスに置き換えます。
- `OS_UI_ENDPOINT` – を OpenSearch UI アプリケーション URL に置き換えます。
- `AWS_REGION` – を自分の AWS リージョンに置き換えます。
- `AWS_PROFILE` – を AWS プロファイル名に置き換えます。プロファイルがまだ設定されていない場合は、この手順に従ってください。

MCP サーバーを設定したら、「使用可能なオブザーバビリティデータソースを一覧表示する」などの質問を IDE に入力して接続を確認します。エラーメッセージが表示された場合は、手順に従って設定を修正します。

エージェントオブザーバビリティに利用可能な MCP アプリ

- トリアージと対応 — アラートの相関関係、インシデントのタイムライン
- ログ調査 – パターン検索、ログクラスタリング
- トレース調査 – トレースファインダー、スパンの詳細、レイテンシーの内訳
- メトリクス調査 – PromQL 探索、しきい値分析
- サービスパフォーマンス – RED メトリクス、サービスレベルビュー
- トポロジ – サービスマップ、依存関係グラフ
- 視覚化 – 動的視覚化
- データセットと相関 — クロスシグナル結合、データ概要
- AI とエージェントのオブザーバビリティ – LLM コールトレース、エージェントトレースマップと概要
- スタックヘルス – クラスターステータス、シャード割り当て
- 計測スコア – テレメトリを使用して計測の落とし穴を検出する

mcpb ファイルの検証

(オプション) ダウンロードしたアーティファクトの署名を確認するには、パブリック署名キーをインポートし、次のコマンドを実行します。

```
# 1. Import the public signing key (one-time)
curl -s https://d373kuglijqwic.cloudfront.net/opensearch-mcp-signing-public.asc | gpg
 --import

# 2. Download the artifact and its signature
curl -O https://d373kuglijqwic.cloudfront.net/opensearch-observability-stack-mcp.zip
curl -O https://d373kuglijqwic.cloudfront.net/opensearch-observability-stack-
mcp.zip.asc

# 3. Verify
gpg --verify opensearch-observability-stack-mcp.zip.asc opensearch-observability-stack-
mcp.zip
```

検証が成功すると、出力には が含まれますGood signature from "OpenSearch MCP <aos-observability-mcp-releases@amazon.com>"。

アプリケーションのテレメトリの取り込み

Amazon OpenSearch Service でオブザーバビリティ機能を使用するには、アプリケーショントレース、ログ、メトリクスを取り込む必要があります。このページでは、テレメトリデータを処理して OpenSearch および Amazon Managed Service for Prometheus にルーティングするように OpenTelemetry Collector および OpenSearch Ingestion パイプラインを設定する方法について説明します。

OpenTelemetry Collector の設定

OpenTelemetry (OTel) Collector は、すべてのアプリケーションテレメトリのエントリポイントです。OTLP を介してデータを受信し、メトリクスを Prometheus に送信しながら、トレースとログを OpenSearch Ingestion にルーティングします。

OTel Collector は、次のいずれかの方法を使用して設定できます。

OTel Collector は、SigV4 認証を使用して OpenSearch Ingestion エンドポイントにトレースとログをエクスポートし、Prometheus リモート書き込みエクスポーターを使用してメトリクスを Amazon Managed Service for Prometheus にエクスポートします。OpenSearch Ingestion は、処理、エンリッチメント、および OpenSearch へのルーティングを処理します。

OpenSearch Ingestion を使用した OTel Collector の設定

次の設定例では、SigV4 認証を使用して、トレースとログを OpenSearch Ingestion エンドポイントにエクスポートし、メトリクスを Prometheus にエクスポートします。

```
extensions:
  sigv4auth:
    region: us-west-2
    service: osis

exporters:
  otlphttp/osis-traces:
    traces_endpoint: ${OSIS_ENDPOINT}/v1/traces
    auth: { authenticator: sigv4auth }
    compression: none
  otlphttp/osis-logs:
    logs_endpoint: ${OSIS_ENDPOINT}/v1/logs
```

```
    auth: { authenticator: sigv4auth }
    compression: none
  # Amazon Managed Service for Prometheus via Prometheus Remote Write with SigV4 auth
  prometheusremotewrite/amp:
    endpoint: "https://aps-workspaces.region.amazonaws.com/workspaces/workspace-id/api/
v1/remote_write"
    auth:
      authenticator: sigv4auth

service:
  extensions: [sigv4auth]
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlphttp/osis-traces]
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlphttp/osis-logs]
    metrics:
      receivers: [otlp]
      processors: [batch]
      exporters: [prometheusremotewrite/amp]
```

> **Note**
>
> データを送信する IAM プリンシパルには、パイプライン ARN に対する `osis:Ingest`と アクセス`aps:RemoteWrite`許可が必要です。

OpenSearch Ingestion パイプラインの設定

OpenSearch Ingestion (またはセルフマネージド Data Prepper) は、OTel Collector からテレメトリを受け取り、アプリケーションパフォーマンスモニタリング (APM) のために処理します。

パイプラインアーキテクチャ

パイプラインは、テレメトリデータを次の段階で処理します。

1. エントリパイプラインはすべてのテレメトリを受信し、ログとトレースを別々のサブパイプラインにルーティングします。

2. ログパイプラインは、log-analytics-plainインデックスタイプを使用して OpenSearch にログデータを書き込みます。
3. トレースパイプラインは、raw ストレージパイプラインとサービスマップパイプラインにスパンを分散します。
4. raw トレースパイプラインプロセスはotel_tracesプロセッサにまたがり、trace-analytics-plain-rawインデックスタイプに保存されます。
5. サービスマップパイプラインは、otel_apm_service_mapプロセッサを使用してトポロジと RED (レート、エラー、期間) メトリクスを生成します。リモート書き込みを通じて OpenSearch と Prometheus に書き込みます。

パイプラインの設定

次の例は、ログ、トレース、メトリクスなど、すべてのオブザーバビリティシグナルタイプをカバーする OpenSearch Ingestion の完全なパイプライン設定を示しています。すべてのパイプラインを含めることも、ユースケースに関連するパイプラインのみを含めることもできます。*#########*を、ユーザー自身の情報に置き換えます。

```
version: '2'

# Main OTLP pipeline - receives all telemetry and routes by signal type
otlp-pipeline:
  source:
    otlp:
      logs_path: '/pipeline-name/v1/logs'
      traces_path: '/pipeline-name/v1/traces'
      metrics_path: '/pipeline-name/v1/metrics'
  route:
    - logs: 'getEventType() == "LOG"'
    - traces: 'getEventType() == "TRACE"'
    - metrics: 'getEventType() == "METRIC"'
  processor: []
  sink:
    - pipeline:
        name: otel-logs-pipeline
        routes:
          - logs
    - pipeline:
        name: otel-traces-pipeline
        routes:
          - traces
```

```
    - pipeline:
        name: otel-metrics-pipeline
        routes:
          - metrics

# Log processing pipeline
otel-logs-pipeline:
  source:
    pipeline:
      name: otlp-pipeline
  processor:
    - copy_values:
        entries:
          - from_key: "time"
            to_key: "@timestamp"
  sink:
    - opensearch:
        hosts:
          - 'https://opensearch-endpoint'
        index_type: log-analytics-plain
        aws:
          serverless: false
          region: 'region'
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"

# Trace fan-out pipeline
otel-traces-pipeline:
  source:
    pipeline:
      name: otlp-pipeline
  processor: []
  sink:
    - pipeline:
        name: traces-raw-pipeline
        routes: []
    - pipeline:
        name: service-map-pipeline
        routes: []

# Raw trace storage pipeline
traces-raw-pipeline:
  source:
    pipeline:
      name: otel-traces-pipeline
```

```
  processor:
    - otel_traces:
  sink:
    - opensearch:
        hosts:
          - 'https://opensearch-endpoint'
        index_type: trace-analytics-plain-raw
        aws:
          serverless: false
          region: 'region'
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"

# Service map generation pipeline (APM)
service-map-pipeline:
  source:
    pipeline:
      name: otel-traces-pipeline
  processor:
    - otel_apm_service_map:
        group_by_attributes:
          - telemetry.sdk.language # Add any resource attribute to group by
        window_duration: 30s
  route:
    - otel_apm_service_map_route: 'getEventType() == "SERVICE_MAP"'
    - service_processed_metrics: 'getEventType() == "METRIC"'
  sink:
    - opensearch:
        hosts:
          - 'https://opensearch-endpoint'
        aws:
          serverless: false
          region: 'region'
          sts_role_arn: "arn:aws:iam::account-id:role/pipeline-role"
        routes:
          - otel_apm_service_map_route
        index_type: otel-v2-apm-service-map
    - prometheus:
        url: 'https://aps-workspaces.region.amazonaws.com/workspaces/workspace-id/api/
v1/remote_write'
        aws:
          region: 'region'
        routes:
          - service_processed_metrics
```

```
# Metrics processing pipeline
otel-metrics-pipeline:
  source:
    pipeline:
      name: otlp-pipeline
  processor:
    - otel_metrics:
  sink:
    - prometheus:
        url: 'https://aps-workspaces.region.amazonaws.com/workspaces/workspace-id/api/
v1/remote_write'
        aws:
          region: 'region'
```

取り込みの検証

OTel Collector とパイプラインを設定したら、テレメトリデータが正しく流れていることを確認します。

- OpenSearch インデックスの検証 – ドメインに次のインデックスが存在することを確認します: `otel-v1-apm-span-*`、`otel-v2-apm-service-map`、`logs-otel-v1-*`。
- Prometheus ターゲットの検証 – Prometheus リモート書き込みターゲットがサービスマップパイプラインからメトリクスを受信していることを確認します。
- OpenSearch UI で検証 – オブザーバビリティに移動し、Application Monitoring に移動してサービスが表示されることを確認します。

次の手順

テレメトリデータが取り込まれたことを確認したら、次のトピックを参照してください。

- the section called “アプリケーションのモニタリング” – サービスマップと RED メトリクスを使用してアプリケーションのヘルスをモニタリングします。
- トレースの検出 – 分散トレースを検出して分析します。
- ログの検出 – アプリケーションログを検出してクエリします。
- メトリクスの検出 – PromQL を使用して Prometheus メトリクスを検出してクエリします。

データセット

データセットは、オブザーバビリティデータの論理グループを表すインデックスのコレクションです。データセットを使用してログとトレースデータを整理し、Discover エクスペリエンスで関連するインデックスを一緒にクエリおよび分析できるようにします。各データセットは OpenSearch Service ドメインの 1 つ以上のインデックスにマッピングされ、検出ページのデータ型、時間フィールド、クエリ言語を定義します。

データセットタイプ

次の表に、作成できるデータセットタイプを示します。

型	説明	クエリ言語
ログ	検出ログページでクエリと視覚化のために 1 つ以上のログインデックスをグループ化します。	PPL
トレース	トレースの検出ページでクエリと視覚化のためにトレーススパンインデックスをグループ化します。	PPL

> **Note**
>
> メトリクスデータは OpenSearch に保存されないため、メトリクスにはデータセットは必要ありません。メトリクスは、PromQL を使用して Amazon Managed Service for Prometheus から直接クエリされます。

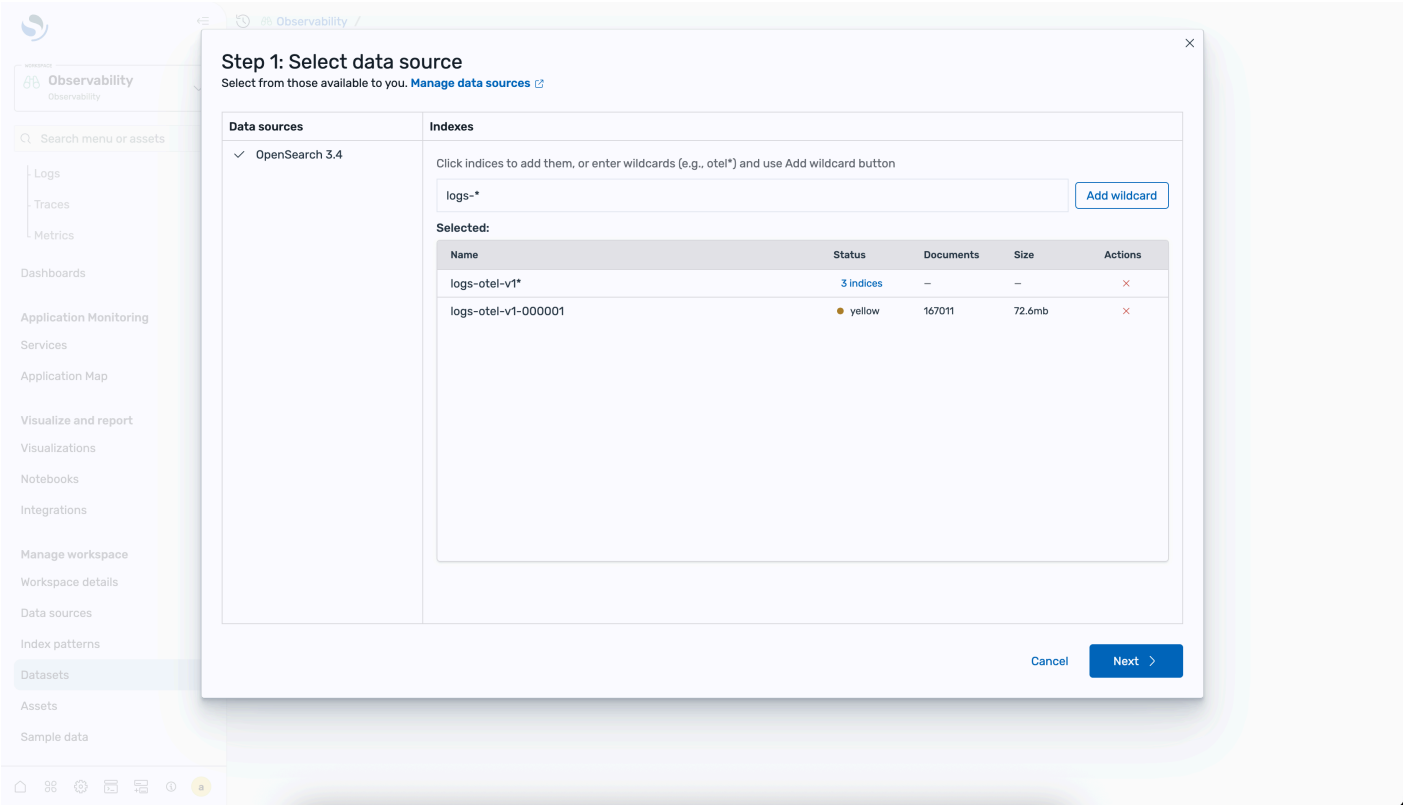
ログデータセットを作成するには

OpenSearch UI でログデータセットを作成するには、次の手順を実行します。

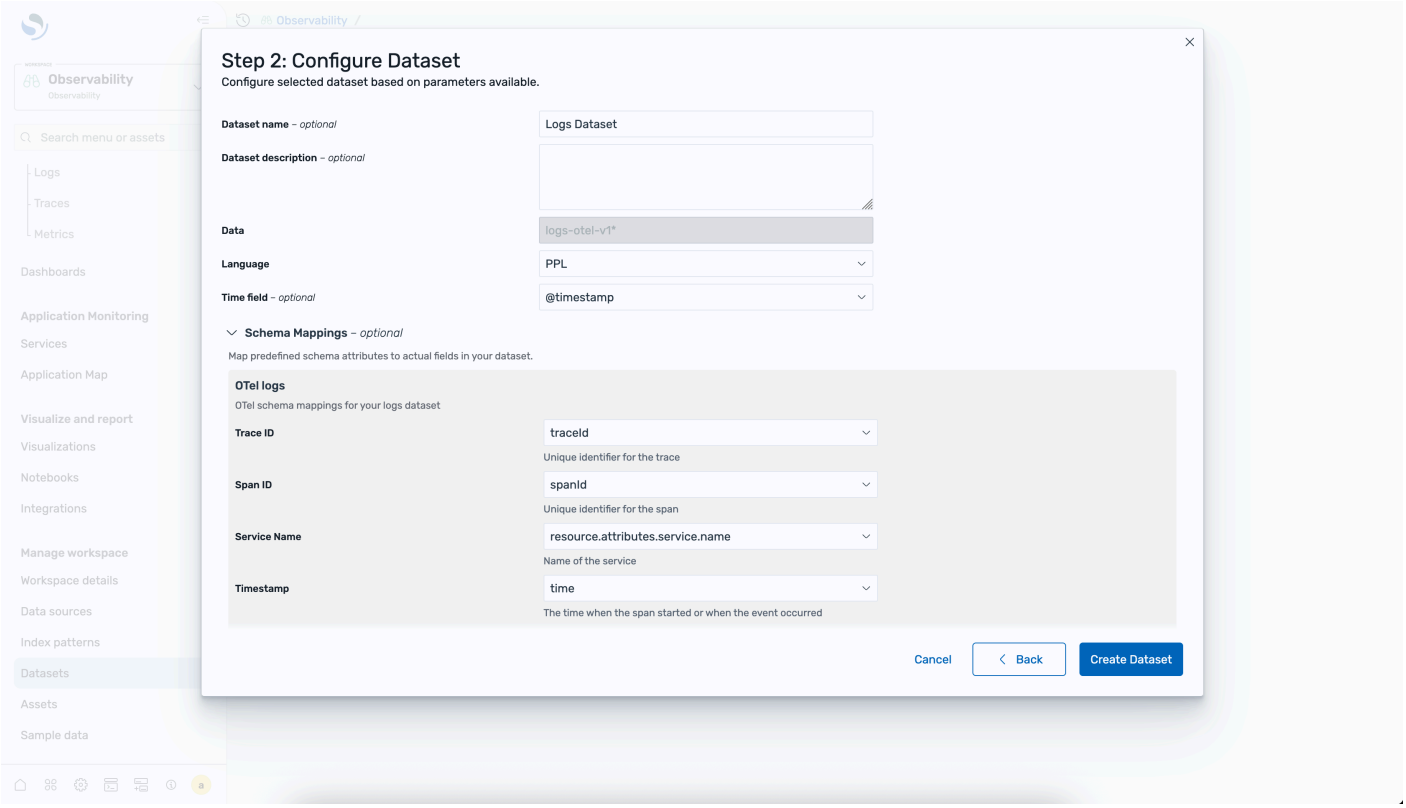
1. オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、ログを選択します。

2. [データセットを作成] を選択します。

3. 使用可能な OpenSearch Service 接続のリストからデータソースを選択します。



4. 名前を入力し、インデックスを選択し、タイムスタンプフィールドを指定してデータセットを設定します。

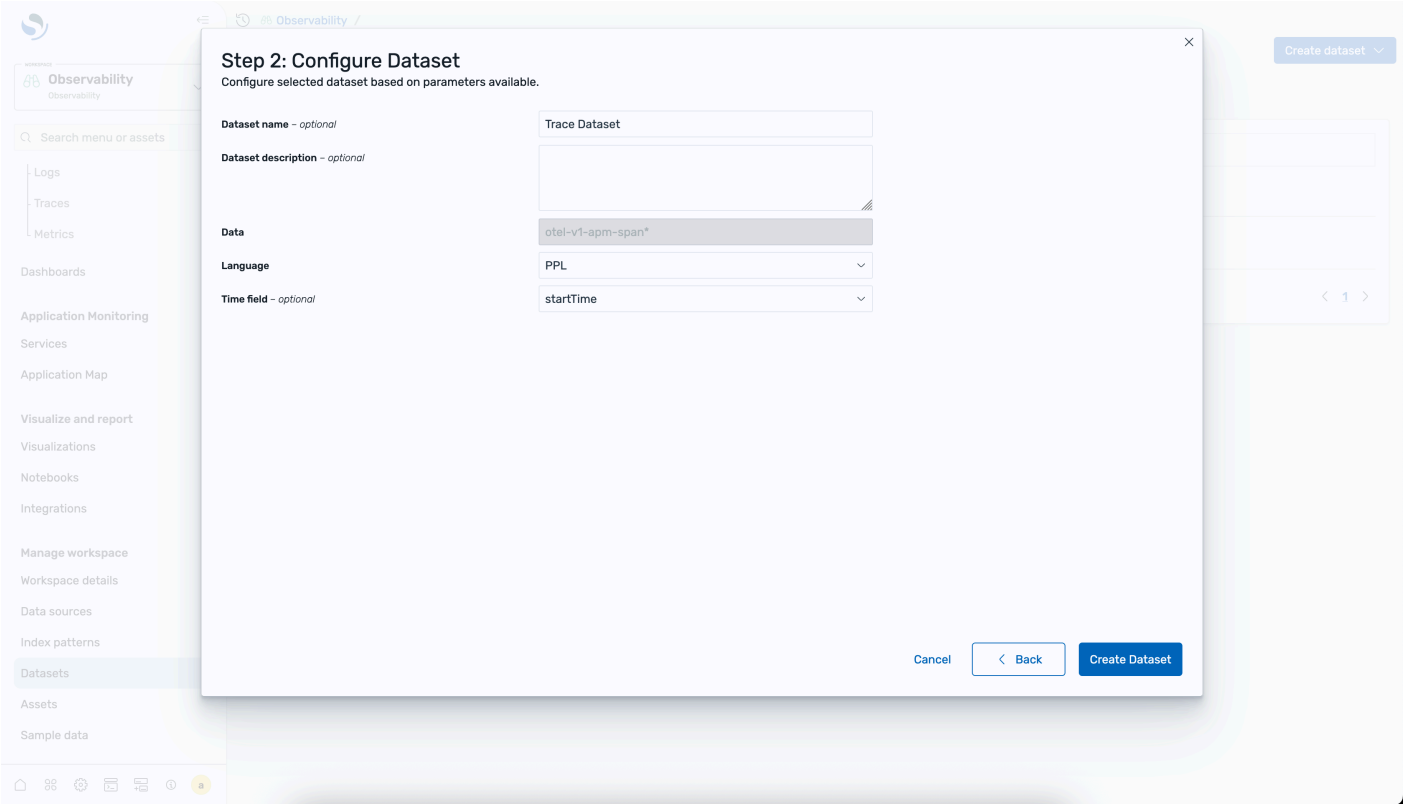


5. データセットの作成を選択して設定を保存します。

トレースデータセットを作成するには

OpenSearch UI でトレースデータセットを作成するには、次の手順を実行します。

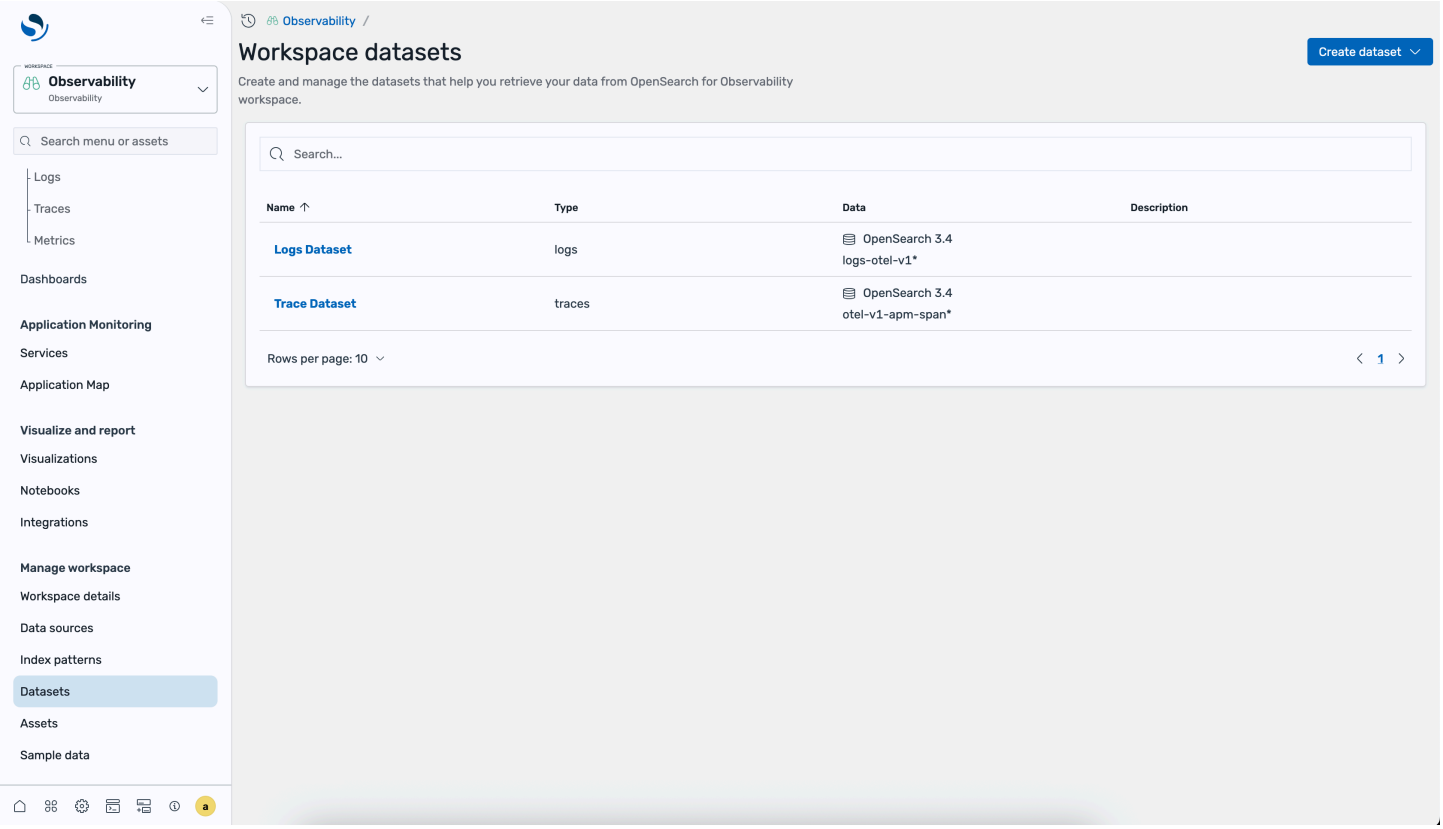
1. オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、トレースを選択します。
2. [データセットを作成] を選択します。
3. 使用可能な OpenSearch Service 接続のリストからデータソースを選択します。
4. データセットを設定するには、名前を入力し、スパンインデックスを選択し、タイムスタンプフィールドを指定します。



5. データセットの作成を選択して設定を保存します。

データセットを表示するには

設定されたすべてのデータセットは、Discover Logs または Discover Traces ページのデータセットセレクターから表示できます。データセットリストには、各データセットの名前、タイプ、データソース、タイムスタンプフィールドが表示されます。



Discover でのデータセットの分析

データセットを作成したら、対応する検出ページで分析できます。

ログ

検出ログページのデータセットセレクターからログデータセットを選択して、PPL を使用してログデータをクエリおよび視覚化します。詳細については、「the section called “ログの検出”」を参照してください。

トレース

トレースの検出ページのデータセットセレクターからトレースデータセットを選択して、トレーススパンの探索、RED メトリクスの表示、個々のトレースのドリルダウンを行います。詳細については、「the section called “トレースの検出”」を参照してください。

ログの検出

ログの検出ページには、OpenSearch Service オブザーバビリティワークスペースのログデータを探索および分析するための専用インターフェイスが用意されています。PPL クエリを記述して、ログ

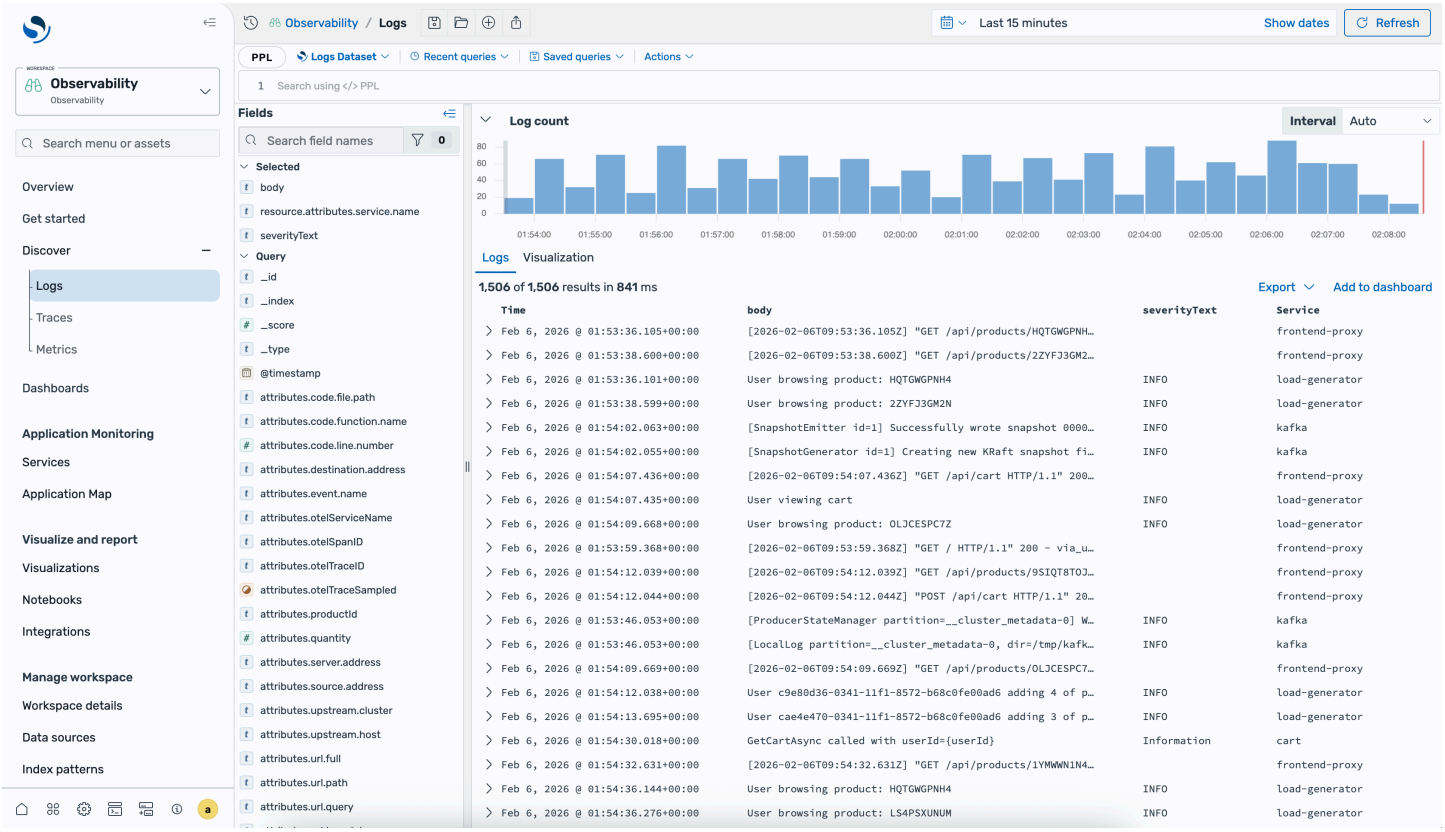
データをフィルタリングおよび集計し、クエリ結果から直接ビジュアライゼーションを作成し、それらのビジュアライゼーションをダッシュボードに追加できます。このページでは、OpenSearch AI アシスタントによる自然言語クエリのサポートも提供しています。

ログページにアクセスするには

オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、ログを選択します。

ログデータの探索

Discover Logs インターフェイスには、ログデータを探索するための以下のコンポーネントが用意されています。



- データセットセレクタ – クエリするログデータセットを選択します。各データセットは、OpenSearch Service ドメイン内の 1 つ以上のインデックスにマッピングされます。
- クエリエディタ – PPL クエリを記述して、ログデータをフィルタリング、集計、変換します。エディタには、オートコンプリートの提案と構文のハイライトが表示されます。
- 時間フィルター – クエリ結果の時間範囲を指定します。相対範囲を選択するか、絶対開始時刻と終了時刻を指定できます。

- 結果パネル – クエリ結果をログイベントのテーブルとして表示します。個々のイベントを展開して、すべてのフィールドを表示できます。
- ヒストグラム – ログイベントの経時的な分布を表示します。ヒストグラムは、クエリと時間フィルターに基づいて自動的に更新されます。
- フィールドパネル – データセット内の使用可能なフィールドを参照し、結果テーブルに列として追加します。

PPL を使用したログのクエリ

パイプ処理言語 (PPL) は、コマンドの連鎖にパイプベースの (|) 構文を使用するクエリ言語です。PPL を使用して、ログデータをフィルタリング、集約、変換できます。

基本的なクエリ

データセットからすべてのログイベントを取得するには、 `source` コマンドを使用します。

```
source = my-logs-dataset
```

結果の数を制限するには、 `head` コマンドを使用します。

```
source = my-logs-dataset | head 20
```

WHERE を使用したフィルタリング

`where` 句を使用して、フィールド値に基づいてログイベントをフィルタリングします。

```
source = my-logs-dataset | where severity_text = 'ERROR'
```

複数の条件を組み合わせることができます。

```
source = my-logs-dataset |
    where severity_text = 'ERROR' and service_name = 'payment-service'
```

クエリの管理

頻繁に使用するクエリを保存して再利用できます。クエリを保存するには、クエリエディタツールバーで保存を選択し、クエリの名前を入力します。保存されたクエリをロードするには、開く を選択し、リストからクエリを選択します。

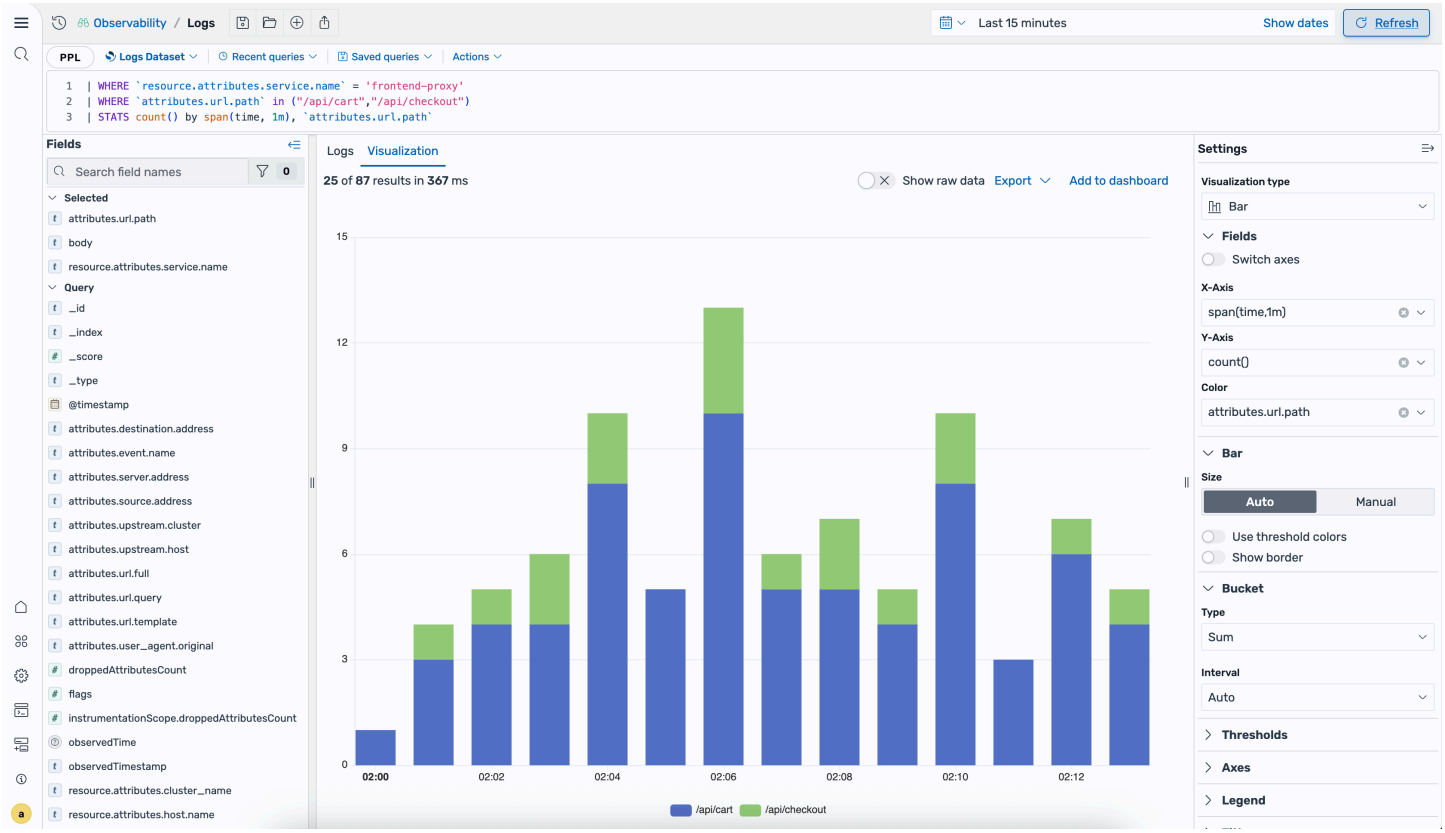
PPL コマンドと関数の完全なリストについては、「Piped Processing Language リファレンス」を参照してください。

ログからの視覚化の作成

PPL クエリ結果から直接視覚化を作成できます。stats コマンドを使用して、視覚化のためにデータを集約します。

```
source = my-logs-dataset |
    stats count() as error_count by service_name, span(timestamp, 1h)
```

stats クエリを実行したら、視覚化タブを選択して、結果をグラフとして表示します。



視覚化タイプ

次の表は、使用できる視覚化タイプを示しています。

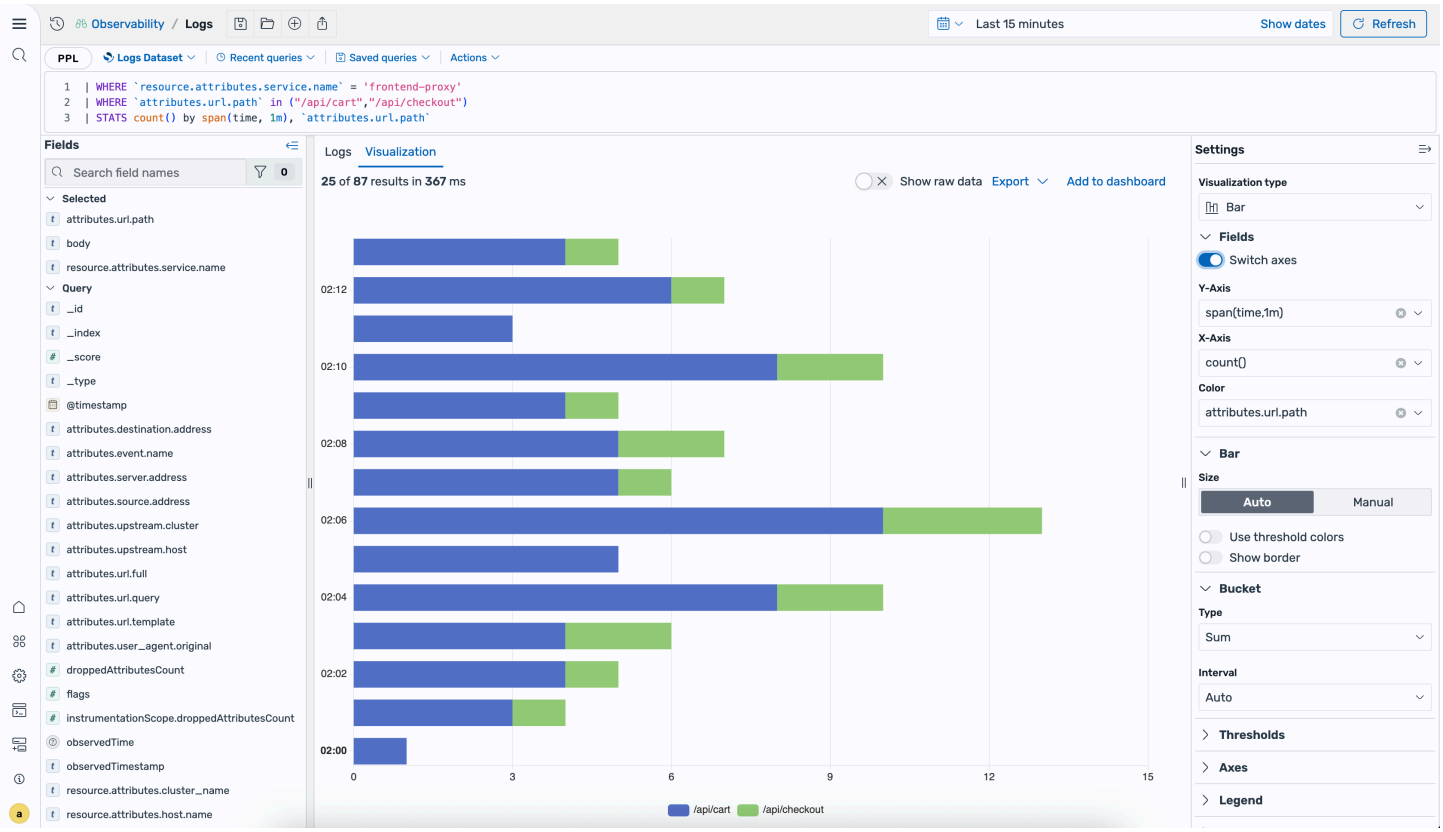
型	説明
線グラフ	線で接続されたデータポイントを表示します。時間の経過に伴う傾向を表示するのに役立ちます。
[面積]	折れ線の下の領域が埋められた折れ線グラフと同様に、時間の経過とともにボリュームを表示するのに役立ちます。
Bar	データを縦棒または横棒として表示し、カテゴリ間で値を比較するのに役立ちます。
メトリクス	1 つの数値を表示します。主要なパフォーマンス指標を表示するのに役立ちます。
状態タイムライン	時間の経過に伴う状態の変化を色付きの帯として表示します。ステータス遷移のモニタリングに役立ちます。
ヒートマップ	データを色付きのセルのマトリックスとして表示し、密度とパターンを表示するのに役立ちます。
棒ゲージ	1 つの値を範囲内の塗りつぶしバーとして表示します。しきい値の進行状況を表示するのに役立ちます。
円グラフ	データを円の比例スライスとして表示します。コンポジションを表示するのに役立ちます。



視覚化設定

視覚化タブがアクティブになると、設定パネルが画面の右側に表示されます。このパネルを使用して、グラフタイプの設定、軸へのフィールドのマッピング、色や凡例などのビジュアルスタイルをカスタマイズします。

視覚化の軸を切り替えるには、設定パネルで軸設定を使用します。



ダッシュボードへの視覚化の追加

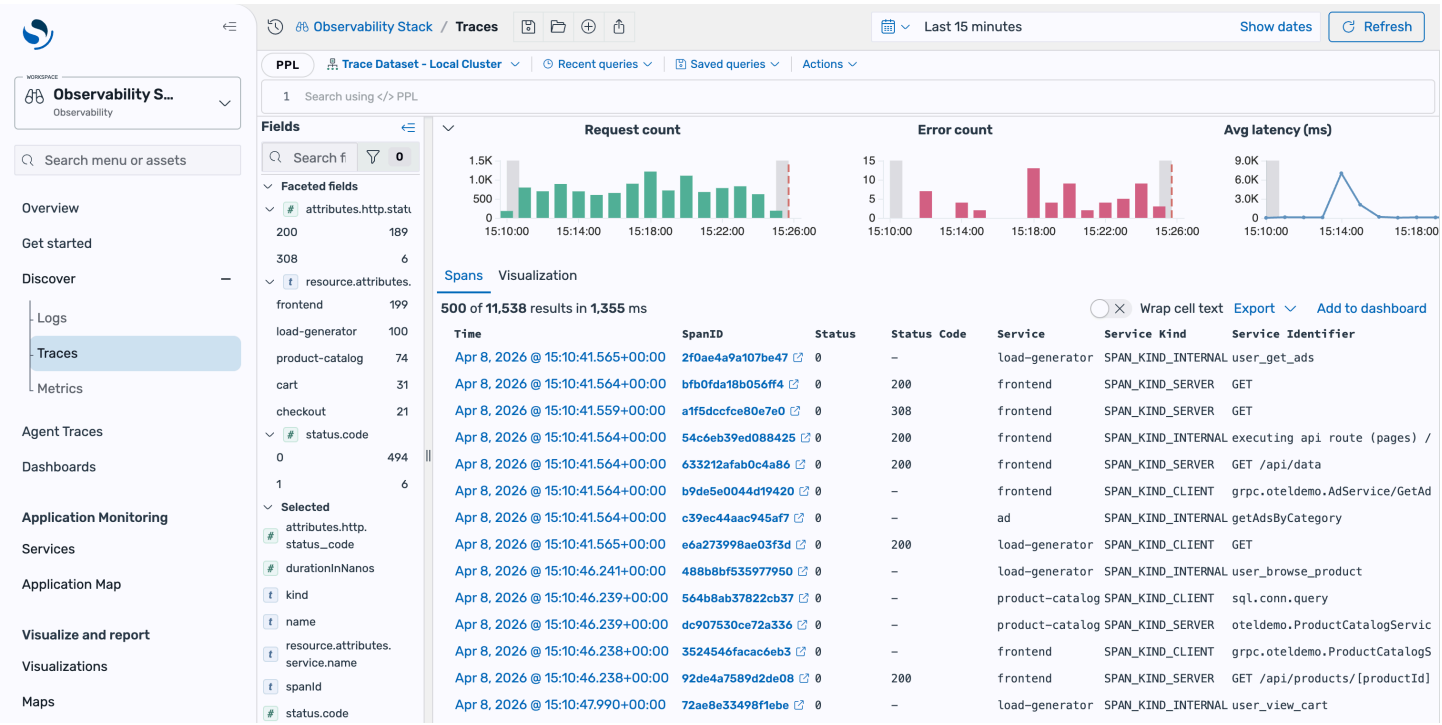
ビジュアライゼーションを作成したら、ダッシュボードに追加して継続的なモニタリングを行うことができます。視覚化ツールバーでダッシュボードに保存を選択し、既存のダッシュボードを選択するか、新しいダッシュボードを作成します。視覚化は基盤となる PPL クエリとともに保存されるため、ダッシュボードを開くと自動的に更新されます。

トレースの検出

トレースの検出ページには、OpenSearch Service オブザーバビリティワークスペースで分散トレースデータを探索するための専用インターフェイスが用意されています。サービスの RED メトリクス (レート、エラー率、期間) を表示したり、ファセットフィルタリングを使用してトレーススパンを参照したり、個々のスパンとトレースをドリルダウンしてパフォーマンスの問題を診断したりできます。このページでは、トレースと関連するログデータの関連付けもサポートしています。

トレースページにアクセスするには

オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、トレースを選択します。

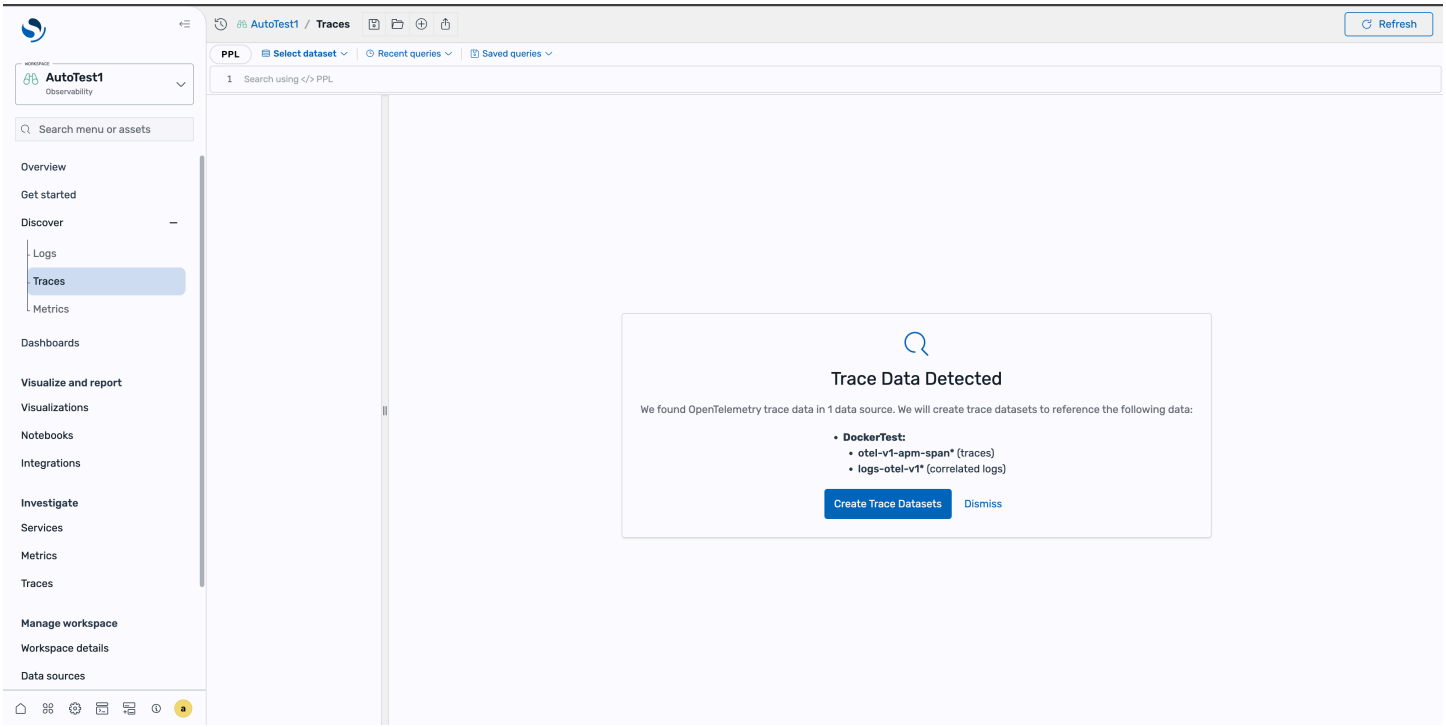


トレースデータセットの設定

トレースデータを調べる前に、トレースデータセットを設定する必要があります。データセットは、自動または手動で作成できます。

データセットの自動作成

トレースの検出ページに初めて移動し、トレースデータがドメインに存在する場合、ページにはデータセットを自動的に作成するように求められます。データセットの作成を選択して、デフォルト設定を受け入れます。



手動データセット作成

トレースデータセットを手動で作成するには、「」の手順に従いますthe section called “トレースデータセットを作成するには”。手動作成では、インデックスパターン、タイムスタンプフィールド、データセット名を制御できます。

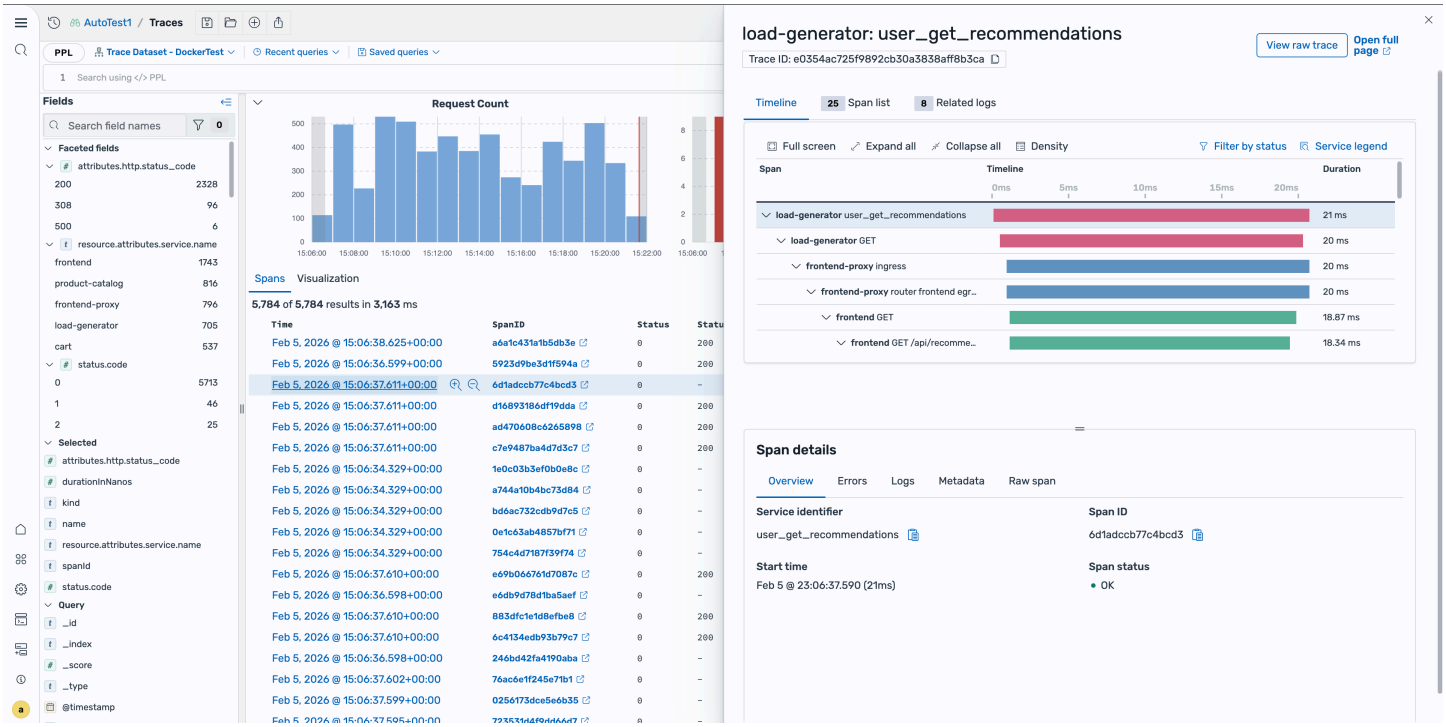
トレースデータの探索

トレースの検出ページには、トレースデータを探索するための以下のコンポーネントが表示されます。

- RED メトリクス – 選択したデータセットのレート (1 秒あたりのリクエスト数) 、エラーレート (失敗したリクエストの割合) 、期間 (レイテンシーパーセンタイル数) を表示します。これらのメトリクスは、時間フィルターに基づいて更新されます。
- ファセットフィールド – サービス名、オペレーション、ステータスコード、およびその他のスパン属性でトレーススパンをフィルタリングします。ファセットフィールドパネルの値を選択して、結果を絞り込みます。
- スパンテーブル – トレース ID、スパン ID、サービス名、オペレーション、期間、ステータスの列を含む個々のスパンを参照します。任意の列でソートし、行を展開してスパンの詳細を表示できます。

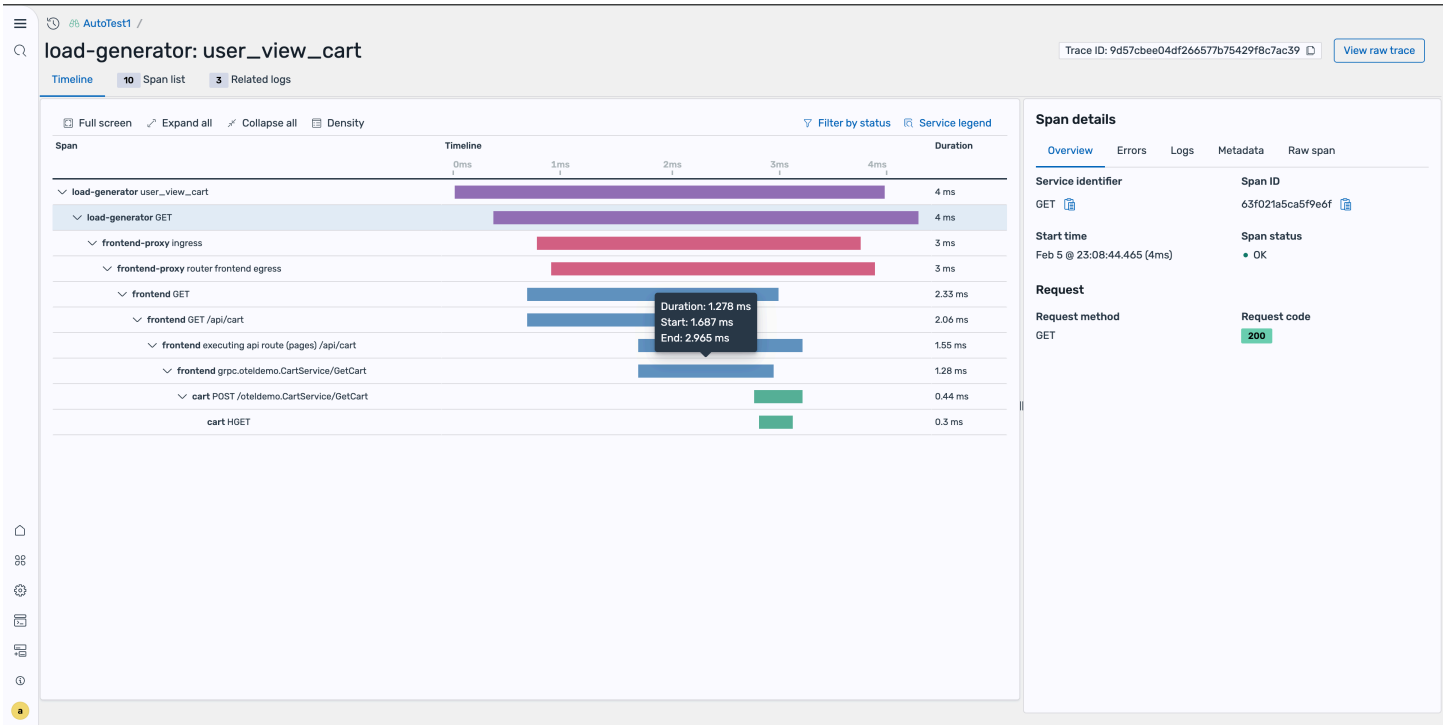
特定のスパンの表示

特定のスパンの詳細を表示するには、スパンテーブルのスパン行を選択します。フライアウトパネルが開き、スパン属性、リソース属性、イベント情報が表示されます。



トレースの詳細ページ

トレース全体を表示するには、スパンテーブルまたはフライアウトパネルでトレース ID リンクを選択します。トレースの詳細ページには、トレース内のすべてのスパン、そのタイミング関係、全体的なトレース期間を示すウォーターフォールグラフが表示されます。個々のスパンを展開して属性を表示し、ボトルネックを特定できます。

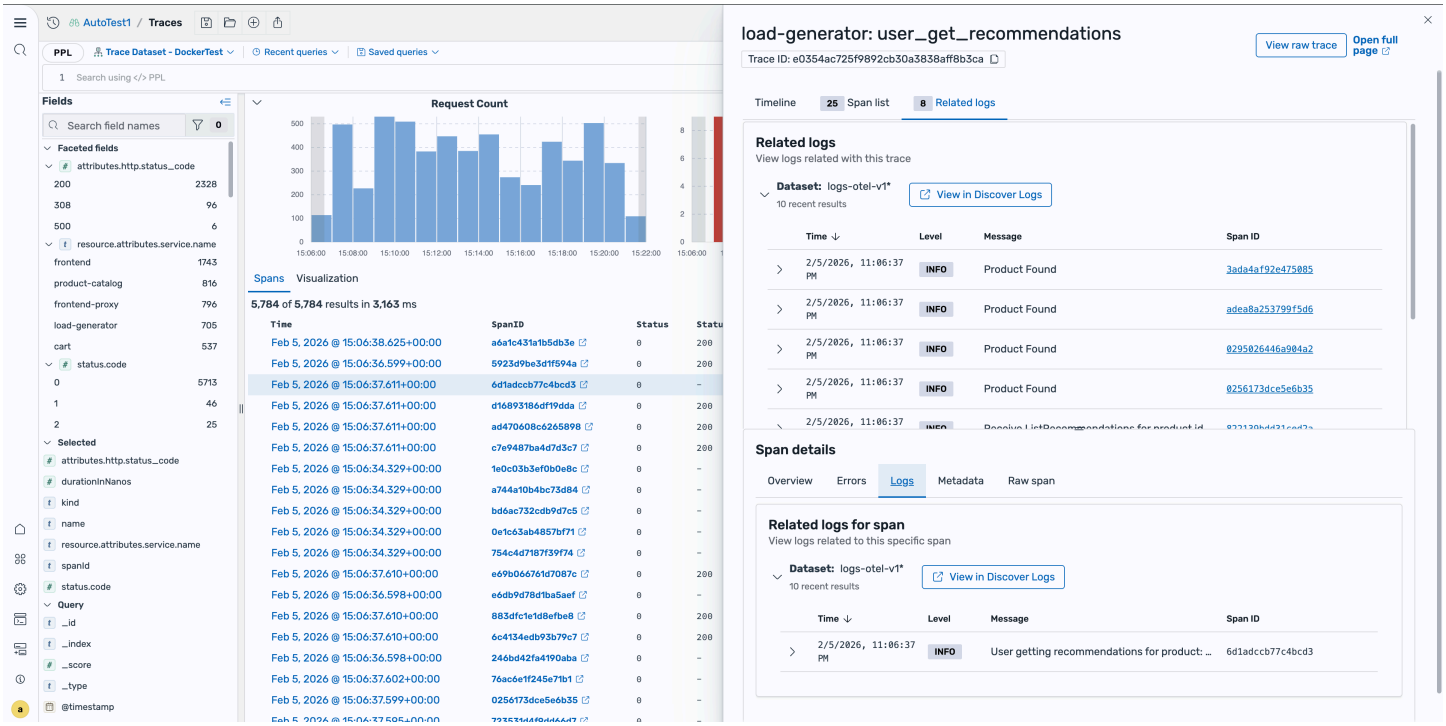


トレースとログの関連付け

トレースデータセットとログデータセットの相関を設定すると、トレースの検出ページから直接関連するログエントリを表示できます。相関関係の作成については、「」を参照してくださいthe section called “相関関係”。

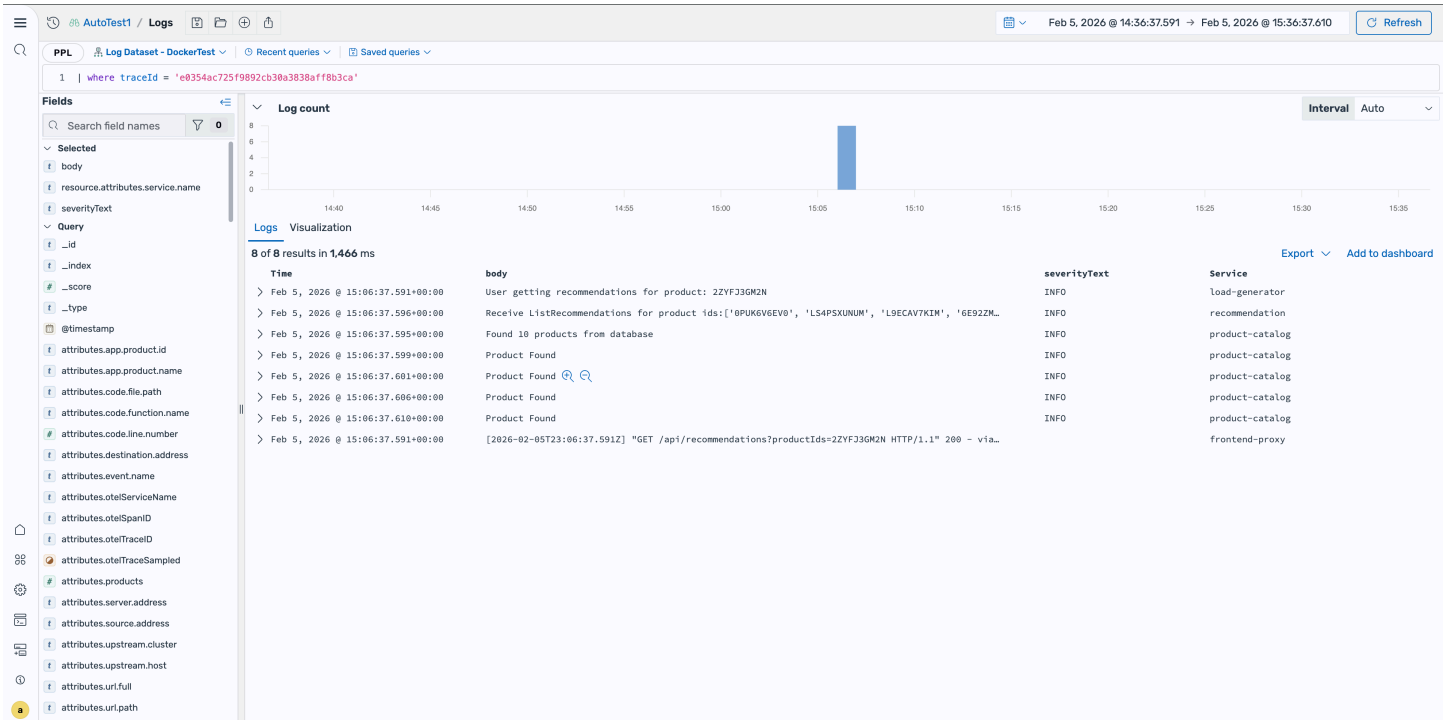
関連ログの表示

スパンの詳細フライアウトまたはトレースの詳細ページで、関連ログタブを選択して、スパンのトレース ID、サービス名、時間範囲に一致するログエントリを表示します。この相関関係は、スパンの実行中にアプリケーションで何が起こったかを理解するのに役立ちます。



コンテキストによるログリダイレクト

トレーススパンから、関連するコンテキストが保持された Discover Logs ページに直接移動できます。関連するログパネルからログで表示を選択し、スパンのトレース ID と時間範囲でフィルタリングするクエリが事前入力された検出ログページを開きます。



PPL を使用したトレースのクエリ

PPL を使用してトレースデータを直接クエリできます。PPL は、パイプ文字を使用してスパンデータをフィルタリング、変換、集計するコマンドを連鎖します。

次の例では、10 個の最も遅いトレースを見つけます。

```
source = otel-v1-apm-span-*
| where durationInNanos > 5000000000
| fields traceId, serviceName, name, durationInNanos
| sort - durationInNanos
| head 10
```

次の例では、サービスごとにエラーをカウントします。

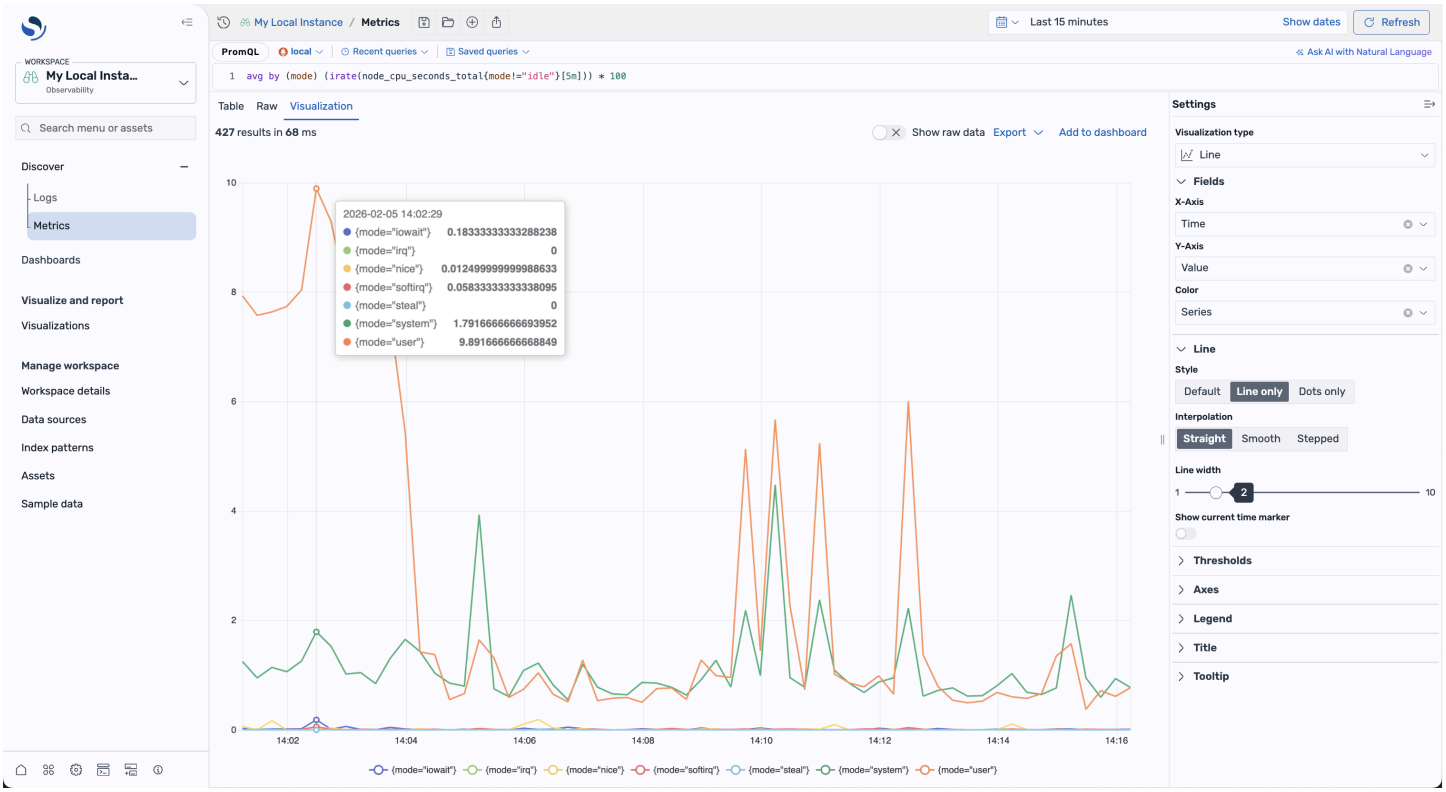
```
source = otel-v1-apm-span-*
| where status.code = 2
| stats count() as errorCount by serviceName
| sort - errorCount
```

次の の例では、特定のサービスのトレースを検索します。

```
source = otel-v1-apm-span-*
| where serviceName = 'checkout-service'
| where parentSpanId = ''
| sort - startTime
| head 20
```

メトリクス

OpenSearch UI のメトリクスの検出ページには、時系列メトリクスデータを検出、クエリ、視覚化するための専用インターフェイスが用意されています。このページは、PromQL を使用して Prometheus メトリクスを操作するために最適化されています。



メトリクスの検出ページは、オブザーバビリティワークスペースで利用できます。これにアクセスするには、オブザーバビリティワークスペースに移動し、左側のナビゲーションで検出を展開して、メトリクスを選択します。

Prometheus データソースの設定

開始する前に、次のいずれかの方法を使用して Prometheus データソースを設定します。

- the section called “Managed Prometheus データソースの作成” AWS マネジメントコンソールの
- AWS CLI

クエリパネル

メトリクス検出ページの上部にあるクエリパネルで、メトリクスクエリを記述して実行できます。クエリエディタは、PromQL のオートコンプリート提案と構文ハイライトを提供します。

クエリの記述

PromQL 構文を使用してクエリを書き込みます。例えば、次のようになります。

```
up{job="prometheus"}
```

クエリの実行

クエリを実行するには、クエリエディタにクエリを入力し、更新を選択します。

複数の PromQL クエリをセミコロン (;) で区切ることで、それらを一緒に実行できます。

```
up{job="prometheus"};
node_cpu_seconds_total{mode="idle"};
```

各クエリは個別に実行され、結果は出力にまとめられます。

時間フィルター

時間フィルターを使用して、メトリクスデータの時間範囲を指定します。

- クイック選択 – 相対時間範囲 (過去 15 分または過去 1 時間など) を選択します。
- よく使用される – 事前定義された時間範囲から選択します。
- カスタム – 絶対開始時刻と終了時刻を指定します。
- 自動更新 – 自動更新間隔を設定します。

結果の表示

クエリを実行すると、結果はタブ付きインターフェイスに表示されます。

- メトリクス – 各系列の最新のデータポイントをテーブル形式で表示します。
- Raw – 各系列の最新のデータポイントを、データソースから返された raw JSON として表示します。
- 視覚化 – メトリクスデータのインタラクティブなグラフを提供します。

ビジュアライゼーションの設定

視覚化タブを選択すると、設定パネルが画面の右側に表示されます。このパネルを使用して、次の操作を行います。

- グラフタイプを選択する – 折れ線、棒、円、ゲージ、またはテーブルの視覚化から選択します。
- 軸のマッピング – X 軸と Y 軸にフィールドを割り当てます。

- スタイルのカスタマイズ – 色、凡例、グリッド線、その他のビジュアルオプションを調整します。

設定を変更すると、視覚化は自動的に更新されます。

相関関係

相関関係はトレースデータセットをログデータセットにリンクするため、トレーススパンを調査するときに関連するログエントリを表示できます。相関を定義することで、スパンの実行中に発生したログが検出トレースページに表示され、ページを切り替えることなく問題を迅速に診断できます。

相関関係の要件

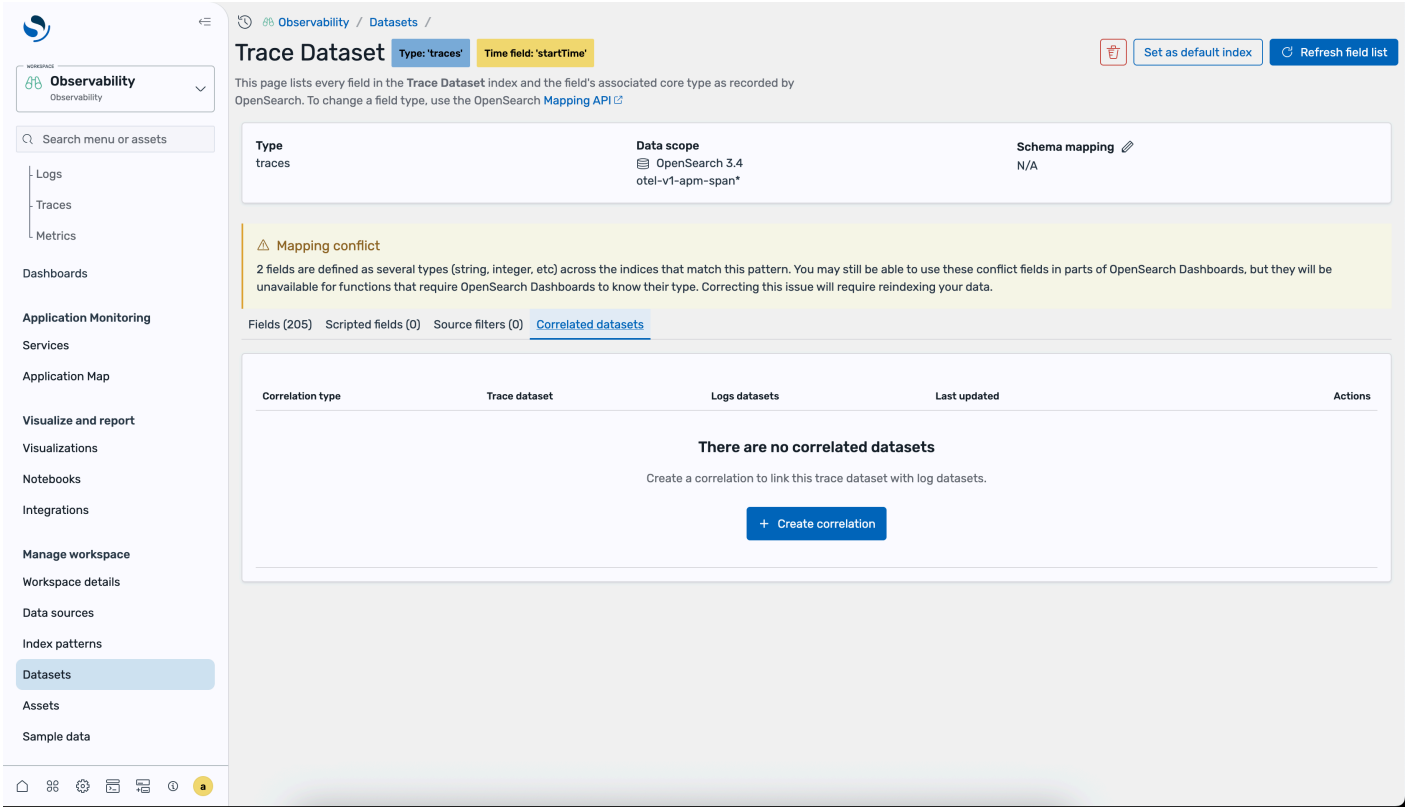
相関関係を作成するには、ログデータとトレースデータに一致するフィールドが含まれている必要があります。次の表は、相関関係がトレースデータとログデータを結合するために使用するフィールドを示しています。

フィールド	説明	必須
トレース ID	トレースの一意の識別子。トレーススパンインデックスとログインデックスの両方に存在する必要があります。	はい
スパン ID	スパンの一意の識別子。トレース内の特定のスパンにログを一致させるために使用されます。	いいえ
サービス名	テレメトリを生成したサービスの名前。サービス別に関連ログをフィルタリングするために使用されます。	いいえ
タイムスタンプ	スパンの時間範囲に関連するログの範囲を設定するために使用される時間フィールド。	はい

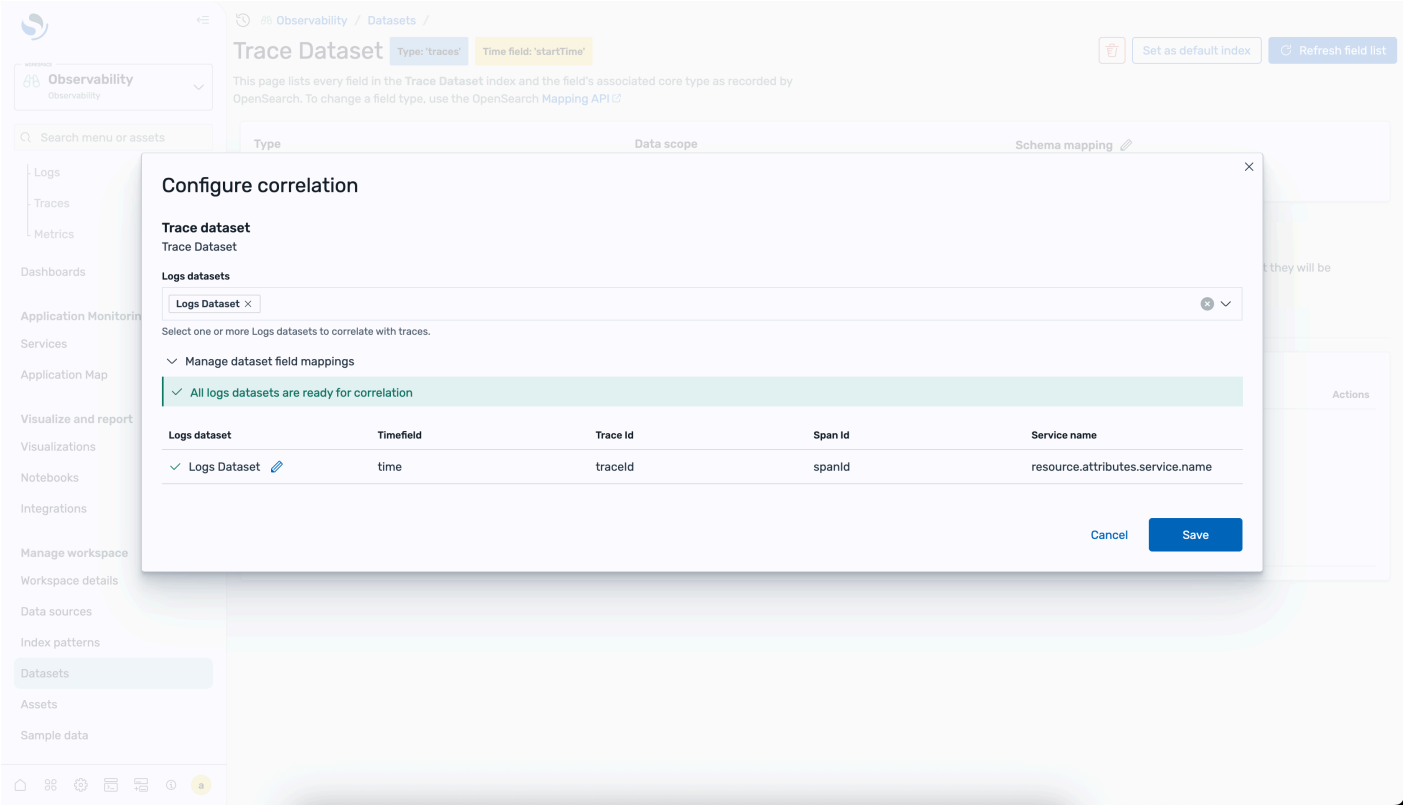
ログtrace-to-logs相関を作成するには

トレースデータセットとログデータセットの間に相関関係を作成するには、次のステップを実行します。

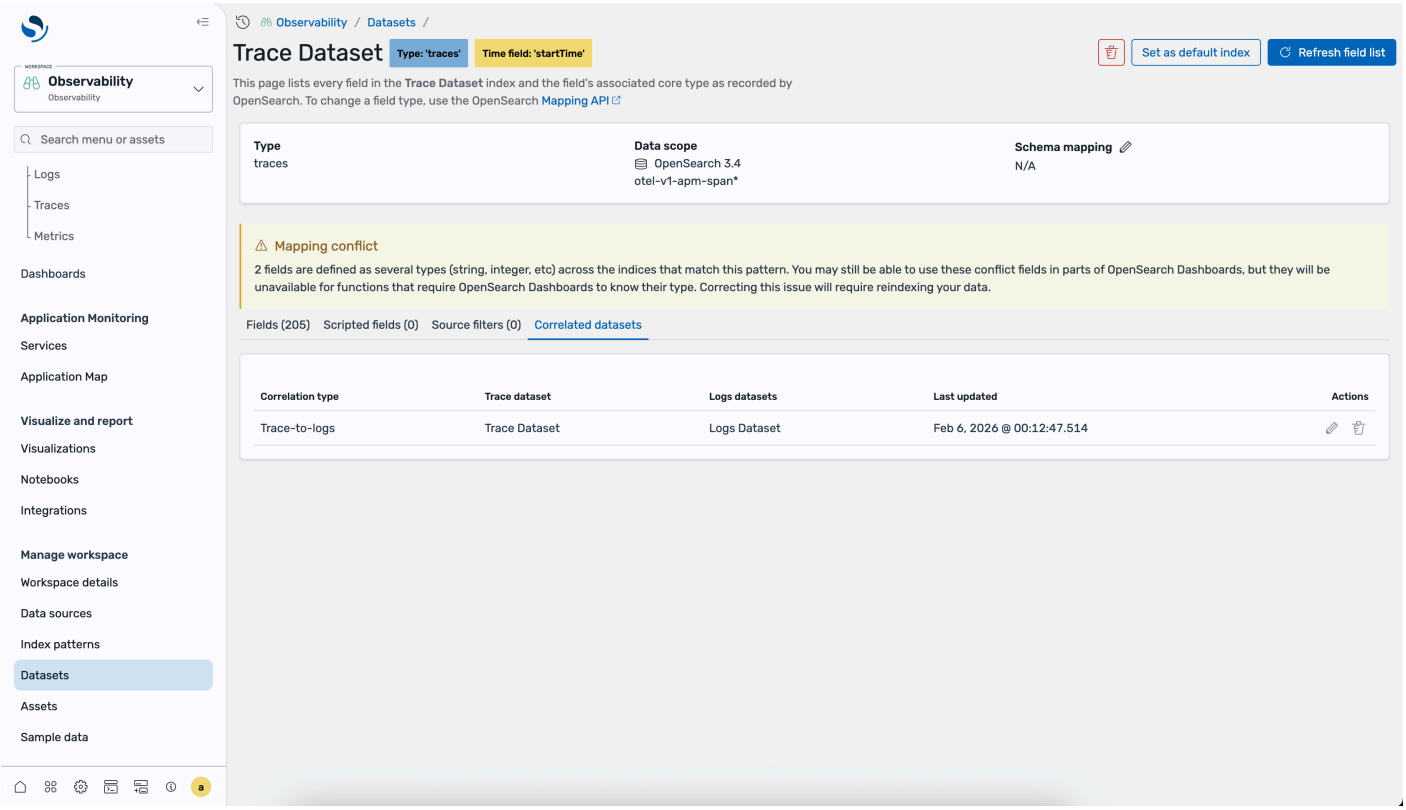
1. オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、トレースを選択します。
2. 関連付けるトレースデータセットを選択します。
3. データセット設定パネルの相関タブを選択します。



4. 相関関係の作成 を選択します。
5. 設定ダイアログで、ターゲットログデータセットを選択し、必要な相関フィールド (トレース ID とタイムスタンプ) をマッピングします。必要に応じて、より正確なマッチングのためにスパン ID とサービス名をマッピングします。

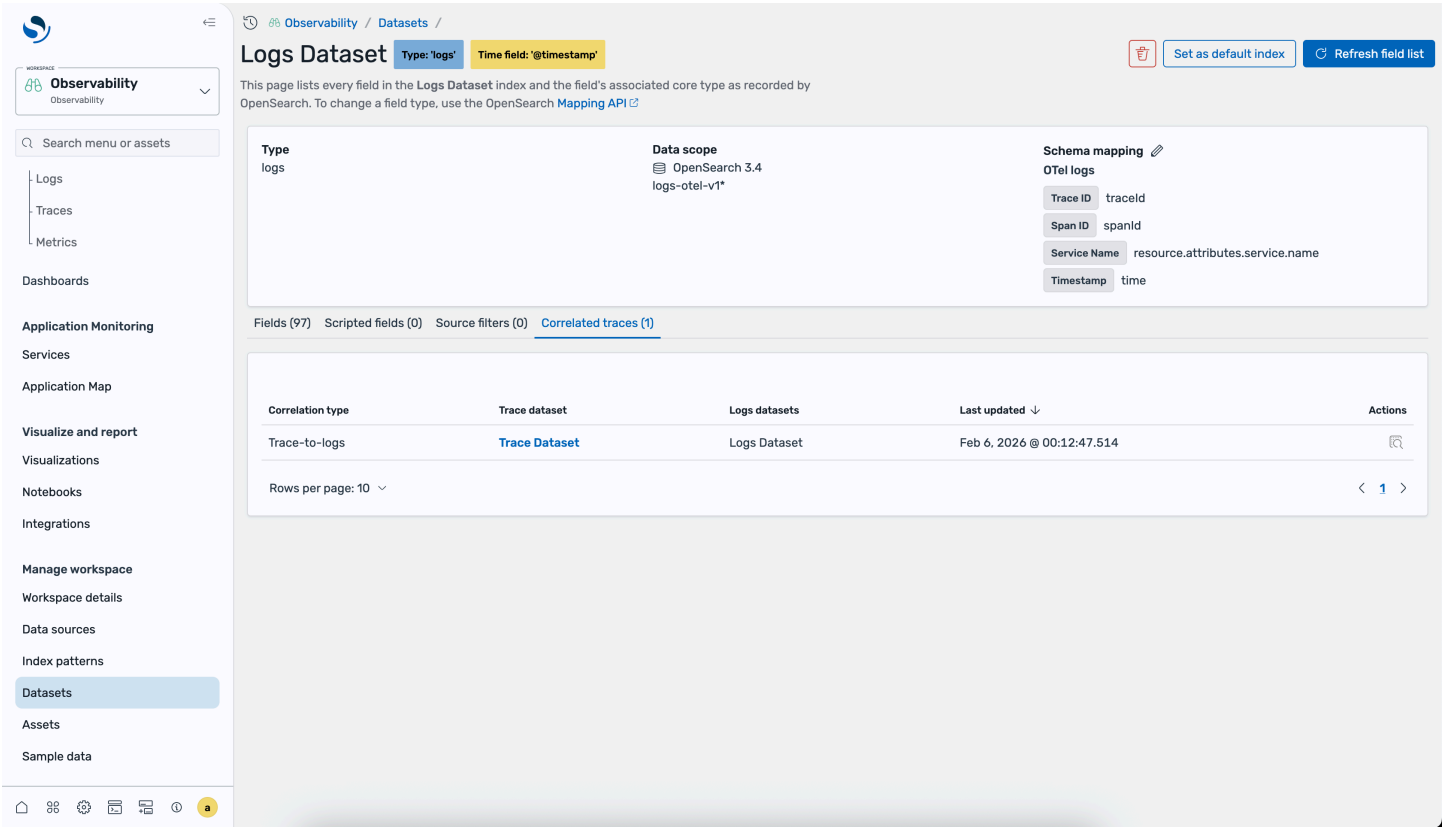


6. Create を選択して相関関係を保存します。
7. 相関関係が相関関係テーブルに表示されていることを確認します。



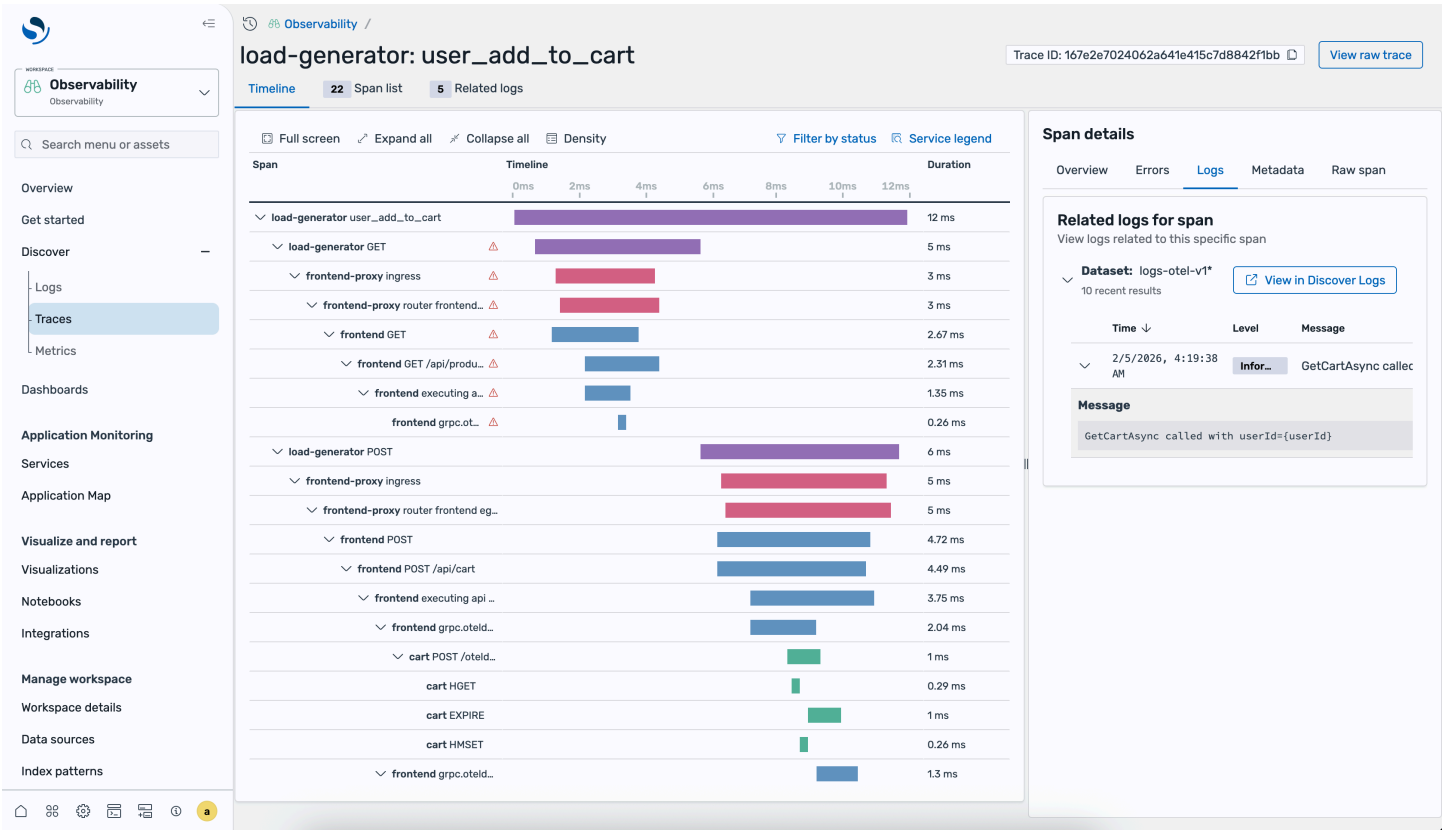
ログデータセットでの相関関係の表示

相関を作成したら、ログデータセット側から表示することもできます。ログの検出ページに移動し、相関ログデータセットを選択し、相関タブを選択してリンクされたトレースデータセットを表示します。



トレースページでの相関関係の使用

相関関係が存在する場合、トレースの検出ページには、関連するログがスパンの詳細ビューに表示されます。スパンテーブルでスパンを選択して詳細フライアウトを開き、関連ログタブを選択して相関ログエントリを表示します。



相関関係の管理

トレースまたはログデータセットの相関タブから相関を編集または削除できます。

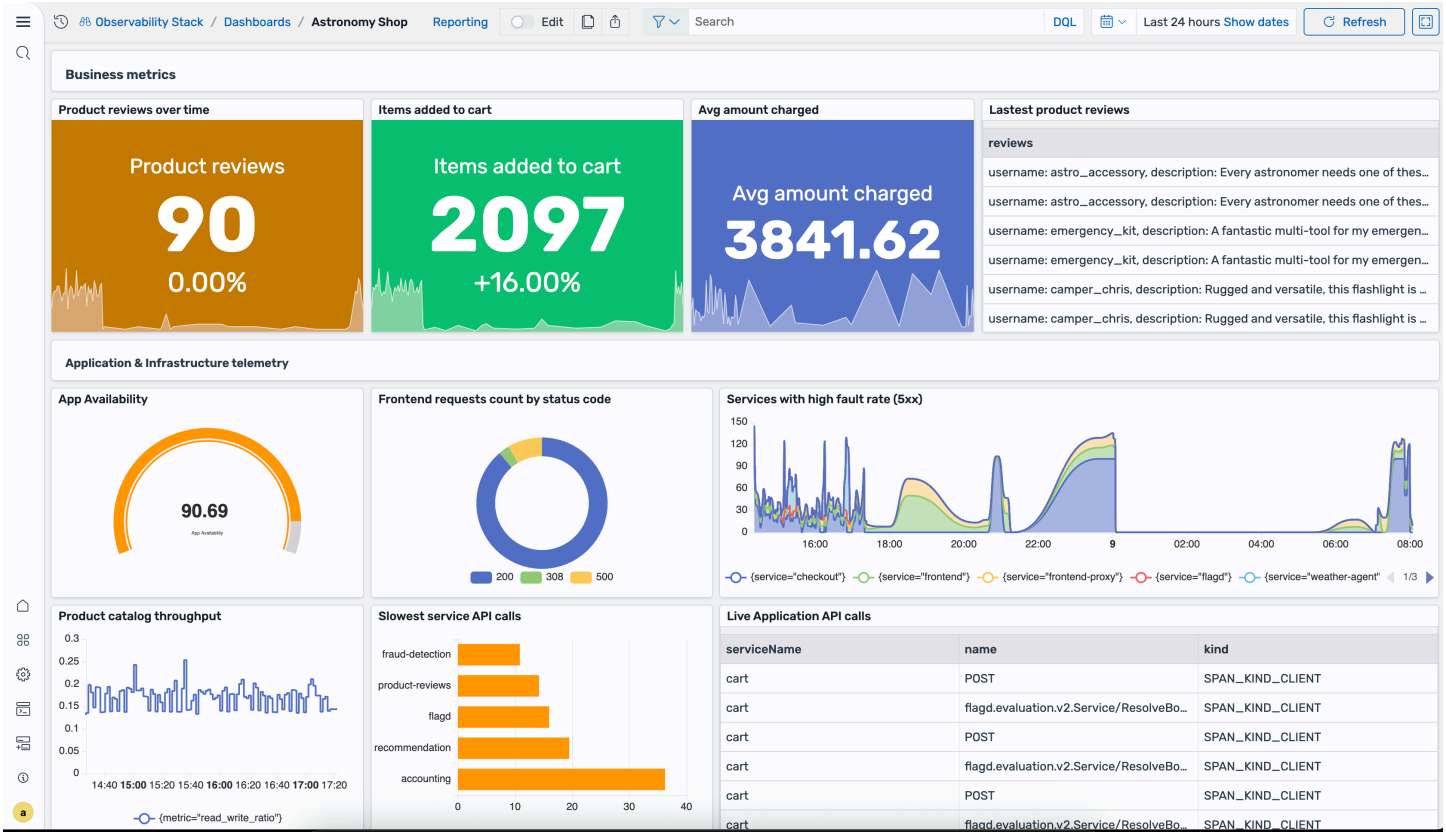
- 編集 — テーブル内の相関関係を選択し、編集を選択してフィールドマッピングまたはターゲットデータセットを更新します。
- 削除 — テーブル内の相関関係を選択し、削除を選択して相関関係を削除します。相関を削除しても、データは削除されません。

ダッシュボード

ダッシュボードは、ログ、トレース、メトリクスからの視覚化を 1 つのビューにまとめます。ダッシュボードを使用して、運用状態のモニタリング、インシデントへの対応、分散システム全体のリソース使用率の追跡を行うことができます。

次の表に、ダッシュボードの一般的なユースケースを示します。

ユースケース	例
運用モニタリング	サービスのヘルス、スループット、エラー率をリアルタイムで追跡します。
インシデントへの対応	アクティブなインシデント中にログ、トレース、メトリクスを関連付けます。
容量プランニング	リソース使用率の傾向をモニタリングしてスケーリングを計画します。
可用性の追跡	サービスレベルの目標に照らして稼働時間と可用性を測定します。
インシデント後レビュー	履歴データを分析して、過去のインシデントの根本原因を理解します。



ダッシュボード構造

ダッシュボードは、グリッドに配置されたパネルのコレクションです。各パネルは以下のコンポーネントで構成されます。

- データソース – パネルがクエリする OpenSearch インデックスまたは Amazon Managed Service for Prometheus データソース。
- クエリ – パネルのデータを取得する PPL または PromQL クエリ。
- 視覚化タイプ – 線、棒、メトリクス値など、クエリ結果のレンダリングに使用されるグラフタイプ。
- オプションの設定 – 軸、凡例、しきい値、書式設定オプション。

ダッシュボードの上部にある時間範囲ピッカーは、すべてのパネルに適用されます。必要に応じて、個々のパネルの時間範囲を上書きできます。

Discover からのダッシュボードの構築

ダッシュボードを構築するための推奨ワークフローは、Discover から始まります。このワークフローは、ログ、トレース、メトリクス間で一貫しています。

1. 検出でデータをクエリする – 検出ログ、検出トレース、または検出メトリクスに移動し、PPL (ログとトレースの場合) または PromQL (メトリクスの場合) を使用してクエリを書き込みます。
2. ビジュアライゼーションの構築 – クエリが結果を返したら、ビジュアライゼーションタブを使用してグラフタイプを選択し、表示を設定します。ログクエリとトレースクエリの場合、 などの集約コマンド`stats`は視覚化ビューに自動的に切り替わります。
3. ダッシュボードに保存する – ダッシュボードに追加を選択して、視覚化を新規または既存のダッシュボードに保存します。パネルはライブのままで、新しいデータが到着すると更新されます。
4. 反復 – ダッシュボードで回答する質問ごとに繰り返します。ダッシュボードで問題が発生したら、パネルを選択して、Discover で基盤となるクエリを開き、さらに調査します。

> **Important**
>
> OpenSearch UI のビジュアライゼーションページを通じて作成されたビジュアライゼーションは、現時点ではパイプ処理言語 (PPL) をサポートしていない DQL (ダッシュボードクエリ

言語) と DSL (ドメイン固有の言語) を使用します。PPL ベースの視覚化を作成するには、上記の Discover ワークフローを使用します。

ダッシュボードフィルター

フィルターを使用すると、個々のクエリを編集せずに、ダッシュボード上のすべてのパネルに表示されるデータを絞り込むことができます。

フィルターを追加するには

1. フィルタリングするダッシュボードを開きます。
2. フィルターバーでフィルターの追加を選択します。
3. ドロップダウンリストからフィールド名を選択します。
4. 演算子を選択し、値を入力します。
5. [保存] を選択します。

次の表に、一般的なフィルターのユースケースを示します。

シナリオ	フィールド	オペレーター	値
単一の環境を表示する	`environment`	`is`	`production`
エラーを分離する	`status_code`	`is greater than or equal to`	`400`
特定のサービスに焦点を当てる	`service.name`	`is`	`order-service`
ヘルスチェックを除外する	`http.url`	`is not`	`/health`

ピン留めされたフィルターとピン留めされていないフィルターの比較 – ダッシュボード間を移動しても、ピン留めされたフィルターは保持されます。ピン留めされていないフィルターは、現在のダッシュボードにのみ適用されます。フィルターを固定するには、フィルターバッジの横にあるピンアイコンを選択します。

ダッシュボードの構築

視覚化タイプ

次の表は、ダッシュボードパネルで使用できる視覚化タイプを示しています。

タイプ	ユースケース
線グラフ	リクエストレートやレイテンシーなどの経時的な傾向
[面積]	積み上げ内訳による経時的なボリューム
Bar	カテゴリ間の値の比較
水平バー	エラー数による上位サービスなどのランク比較
データテーブル	ソートとページ分割を含む表形式データ
メトリクス値	リクエストの合計など、単一の主要業績評価指標
ゲージ	CPU 使用率などのしきい値への進行状況
円グラフ	リージョン別のトラフィックなどの構成と比率
ヒートマップ	2 つのディメンションにわたる密度と分布パターン
クラウドにタグを付ける	一般的なエラーメッセージなどの用語の相対頻度

パネルの設定

各パネルには、PPL または PromQL クエリを記述するクエリエディタがあります。次の例は、一般的なパネルクエリを示しています。

サービス別のエラー数 (PPL):

```
source = logs-dataset |
```

```
    where severity_text = 'ERROR' |
    stats count() as error_count by service_name, span(timestamp, 5m)
```

CPU 使用率 (PromQL):

```
rate(container_cpu_usage_seconds_total{namespace="production"}[5m])
```

次のパネルオプションを設定することもできます。

- 軸 – 軸ラベル、スケール (線形または対数) 、値の範囲を設定します。
- 凡例 – 凡例の位置と表示する系列を制御します。
- しきい値 – 水平しきい値線を追加して、警告レベルまたはクリティカルレベルを強調表示します。

レイアウトのヒント

次のヒントを使用して、ダッシュボードパネルを効果的に整理します。

- ダッシュボードの上部に概要パネル (メトリクス値、ゲージ) を配置します。
- 1 つのサービスのすべてのパネルなど、関連するパネルをグループ化します。
- 同じ行のパネルには一貫した幅を使用します。
- パネルエッジをドラッグしてサイズを変更し、パネルヘッダーをドラッグして位置を変更します。

推奨レイアウト

次の表は、一般的なダッシュボードタイプの推奨パネルレイアウトを示しています。

サービスヘルスダッシュボード

パネル	ビジュアライゼーションのタイプ
リクエスト料金	線グラフ
エラー率	線グラフ
P99 レイテンシー	線グラフ

パネル	ビジュアライゼーションのタイプ
アクティブなアラート	メトリクス値
サービス別の上位エラー	水平バー

インシデント対応ダッシュボード

パネル	ビジュアライゼーションのタイプ
エラーログ	データテーブル
経時的なエラー数	[面積]
影響を受けるサービス	円グラフ
レイテンシースパイク	線グラフ

リソース使用率ダッシュボード

パネル	ビジュアライゼーションのタイプ
CPU 使用率	ゲージ
時間の経過に伴うメモリ使用量	[面積]
ディスク I/O	線グラフ
ネットワークスループット	線グラフ

時間範囲コントロール

ダッシュボードの上部にある時間範囲ピッカーは、すべてのパネルの時間枠を制御します。プリセット範囲 (過去 15 分または過去 24 時間など) を選択するか、カスタム絶対範囲を指定できます。

自動更新を有効にするには、時間範囲ピッカーの横にある更新間隔ドロップダウンを選択し、間隔を選択します。自動更新は、ダッシュボードに最新のデータが表示されるように、指定された間隔ですべてのパネルクエリを再実行します。

ダッシュボードの共有

URLs、スナップショット、エクスポートを使用して、組織内の他のユーザーとダッシュボードを共有できます。

URL 経由で共有する

ブラウザのアドレスバーからダッシュボード URL をコピーし、直接共有します。URL は、現在の時間範囲とフィルターを保持します。ダッシュボードリンクは、ブックマーク、ランブック、またはインシデント対応ドキュメントに含めることができます。

スナップショット

スナップショットは、特定の時点におけるすべてのパネルデータを含むダッシュボードの現在の状態をキャプチャします。スナップショットは読み取り専用であり、基盤となるデータが変更されても更新されません。スナップショットを使用して、インシデントまたはレビュー中のダッシュボードの状態の記録を保持します。

定義のインポートとエクスポート

ダッシュボード定義を JSON としてエクスポートし、別のワークスペースまたは環境にインポートできます。このアプローチは、ダッシュボードを開発から本番環境に昇格させたり、チーム間で標準レイアウトを共有したりするのに役立ちます。

共有のベストプラクティス

- 対象者 – オンコールエンジニアやリーダーシップなど、特定の対象者向けのダッシュボードを設計します。
- フォーカス – 各ダッシュボードを 1 つの目的またはワークフローに制限します。
- 規則 – 組織全体のダッシュボードとパネルに一貫した命名規則を使用します。
- バージョン管理 – ダッシュボードの JSON 定義をエクスポートしてバージョン管理に保存し、経時的な変更を追跡します。

ダッシュボードのトラブルシューティング

このセクションでは、一般的なダッシュボードの問題とその解決方法について説明します。

パネルにデータがない

パネルにデータが表示されない場合は、次の一般的な原因を確認してください。

原因	チェック	Fix
時間範囲が狭すぎる	ダッシュボードの時間範囲が、データが取り込まれた期間を力バーしていることを確認します。	時間範囲を展開するか、過去 24 時間を選択します。
データを除くアクティブなフィルター	一致するすべてのドキュメントを除外する可能性のあるフィルターについては、フィルターバーを確認してください。	フィルターを削除または調整し、データが表示されることを確認します。
インデックスパターンが正しくない	パネルデータソースがデータを含むインデックスを指していることを確認します。	パネルエディタでデータソースを正しいインデックスパターンに更新します。
クエリ構文エラー	パネルヘッダーまたはクエリエディタでエラーメッセージを探します。	PPL または PromQL 構文を修正し、クエリを再実行します。

パネルのデータが正しくない

パネルに予期しない結果が表示された場合は、次の手順を試してください。

- 最初に Discover で実行して、クエリが期待されるフィールドを返すことを確認します。
- 視覚化タイプがデータシェイプと一致することを確認します (時系列データに折れ線グラフを使用するなど)。
- パネルエディタで正しいデータソースが選択されていることを確認します。

古いデータ

ダッシュボードパネルに古い情報が表示された場合は、次の手順を試してください。

- ツールバーの更新アイコンを選択して、すべてのパネルを手動で更新します。
- 自動更新が有効で、適切な間隔に設定されていることを確認します。

- 取り込みパイプラインが、設定されたインデックスにデータをアクティブに送信していることを確認します。

パフォーマンスの問題

以下のヒントは、一般的なパフォーマンスの問題を解決するのに役立ちます。

- スローダッシュボード – パネルの数を減らすか、時間範囲を絞り込みます。多数のパネルを持つダッシュボードは複数のクエリを同時に実行するため、ロード時間が長くなる可能性があります。
- スローパネル – パネルクエリを簡素化します。PPL `where`句でワイルドカードパターンを使用することは避け、集約バケットの数を制限します。
- ブラウザの遅延 — 視覚化のデータ密度を減らします。たとえば、`stats`コマンドの時間間隔を長くして、生成されるデータポイントを減らします。

問題をフィルタリングする

フィルターが正常に動作しない場合は、次の手順を試してください。

- フィルターのフィールド名がインデックスマッピングのフィールド名と一致することを確認します。
- 別のダッシュボードのピン留めされたフィルターが結果に影響を与えているかどうかを確認します。
- すべてのフィルターを削除し、一度に 1 つずつ追加して、問題を分離します。

パネルの検査

パネルインスペクターは、データのデバッグと問題のクエリに役立ちます。インスペクターを開くには、パネルメニュー (3 つのドット) を選択し、Inspect を選択します。インスペクターには以下のタブがあります。

- データ – クエリによって返される raw データを表形式で表示します。
- リクエスト – 完全な PPL または PromQL ステートメントを含む、データソースに送信されたクエリを表示します。
- レスポンス – タイミングやステータス情報など、データソースからの raw レスポンスを表示します。

ブラウザ開発者ツール

高度なトラブルシューティングを行うには、ブラウザ開発者ツールを使用してネットワークリクエストを検査します。ネットワークタブを開き、API コールをフィルタリングして、失敗したリクエストや遅いレスポンスを探します。根本原因の特定に役立つエラーメッセージがないか、レスポンス本文を確認します。

アプリケーションのモニタリング

アプリケーションモニタリングは、サービスのパフォーマンスをリアルタイムで表示します。OpenSearch に保存されているトポロジデータと Amazon Managed Service for Prometheus の時系列 RED メトリクス (レート、エラー、期間) を組み合わせて、分散システム全体のヘルス、レイテンシー、スループット、エラー情報を表示します。

アプリケーションモニタリングにアクセスするには、OpenSearch UI でオブザーバビリティ > アプリケーションモニタリングに移動します。サイドバーには 2 つのビューが表示されます。

- アプリケーションマップ – サービス依存関係のインタラクティブトポロジグラフ
- サービス – フィルタリング、詳細ビュー、相関リンクを含むすべての計測済みサービスのカタログ

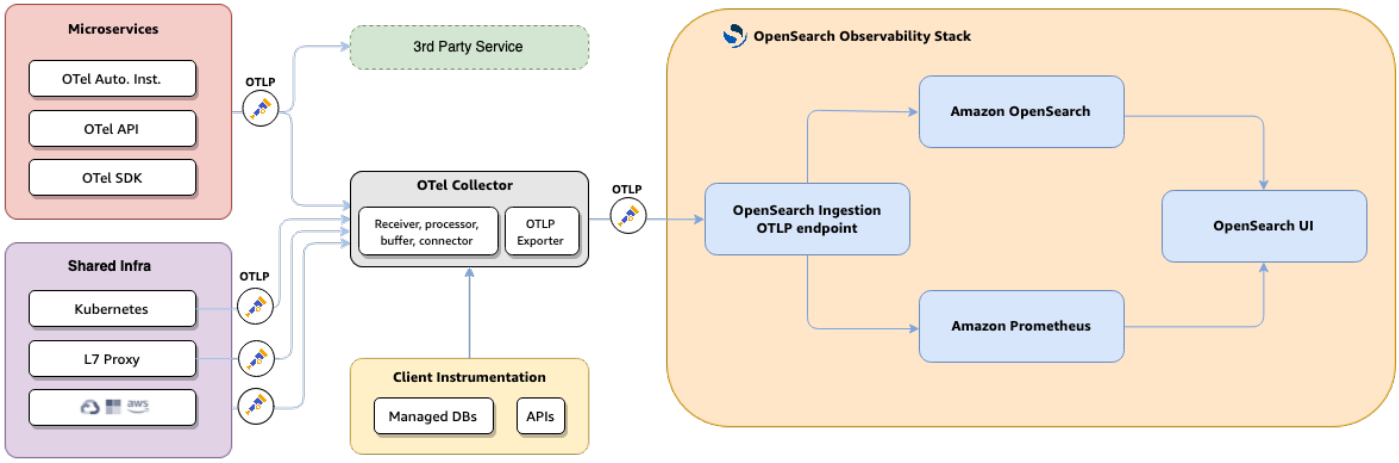
前提条件

アプリケーションモニタリングを使用する前に、次のリソースを設定する必要があります。

- OTel Collectors から OpenSearch Ingestion に流れる OTLP トレースデータ OpenSearch (メトリクスとログはオプション)
- OpenSearch Ingestion からリモート書き込みを受信するように設定された Amazon Managed Service for Prometheus
- オブザーバビリティが有効になっている OpenSearch UI ワークスペース

仕組み

次の図は、アプリケーションモニタリングのend-to-endアーキテクチャを示しています。



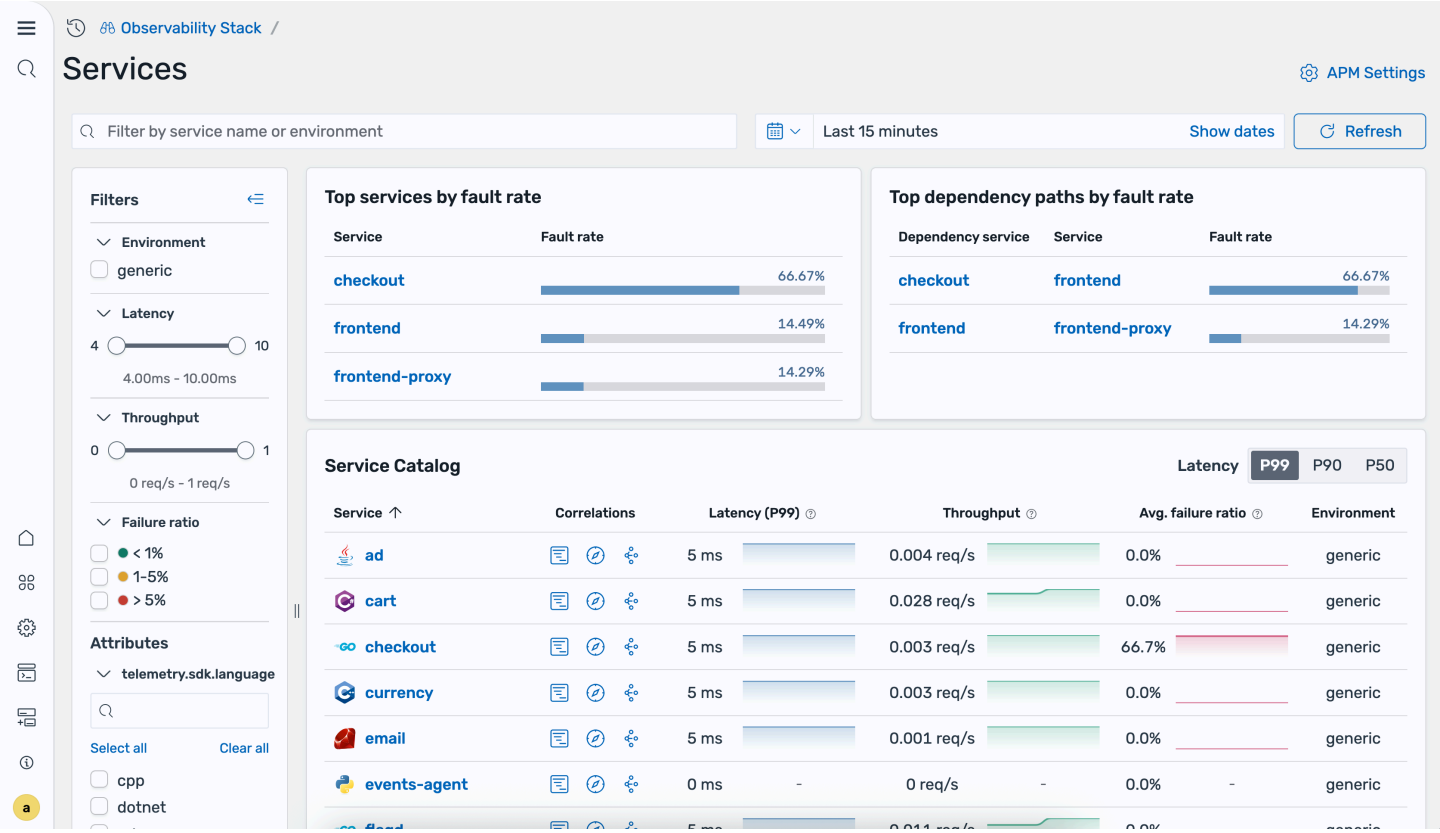
1. アプリケーションとインフラストラクチャは、OpenTelemetry SDKs、自動計測、または OTel API を介してテレメトリを OTel Collector に出力します。
2. OTel Collector はトレースデータを OTLP 経由で OpenSearch Ingestion に転送します。
3. OpenSearch Ingestion `otel_apm_service_map`プロセッサは、service-to-service関係を抽出し、RED メトリクスを計算します。
4. トポロジと raw トレースデータは OpenSearch にインデックス化されます。RED メトリクスは、リモート書き込みを通じて Amazon Managed Service for Prometheus にエクスポートされます。
5. OpenSearch UI は、アプリケーションマップ、サービスカタログ、サービス詳細ビューをレンダリングするために、両方の ストアにクエリを実行します。

サービス

サービスビューには、計測されたすべてのサービスの一元化されたカタログが表示され、RED メトリクス (レート、エラー、期間) が一目でわかります。このビューを使用すると、異常なサービスをすばやく特定し、詳細ビューをドリルダウンしてより詳細な分析を行うことができます。

サービスビューにアクセスするには、OpenSearch UI のオブザーバビリティワークスペースに移動し、APM > サービスを選択します。

サービスのホームページには、すべての計測済みサービスのテーブルと概要パネルが表示されます。次の図は、 サービスのホームページを示しています。



次の表は、 サービステーブルの列を示しています。

列	[Description] (説明)
サービス名	計測されたサービスの名前。
P99 レイテンシー	サービスの 99 パーセンタイルレイテンシー。
P90 レイテンシー	サービスの 90 パーセンタイルレイテンシー。
P50 レイテンシー	サービスの 50 パーセンタイル (中央値) レイテンシー。
Total requests	選択した時間範囲で処理されたリクエストの合計数。
失敗率	失敗したリクエストと合計リクエストの比率。
環境	`production` や など、サービスのデプロイ環境`staging`。

ホームページには、次の概要パネルも含まれています。

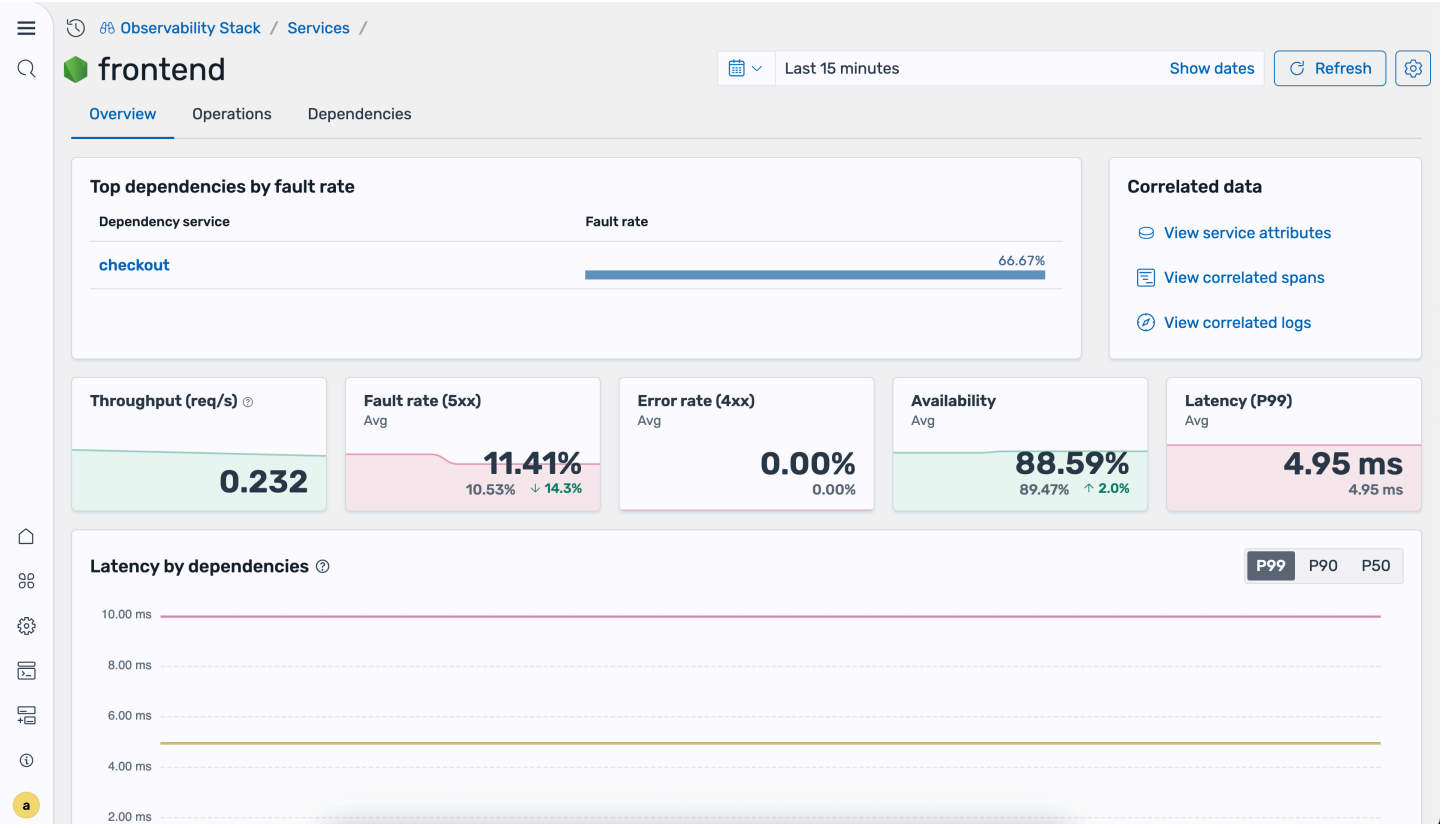
- 障害率別の上位サービス — 5xx レスポンスの割合が最も高いサービス。
- 障害率別の上位の依存関係パス — 障害率が最も高かったService-to-service依存関係パス。

次のフィルターを使用して、サービステーブルをフィルタリングできます。

- 環境 – デプロイ環境でフィルタリングします。
- レイテンシー – レイテンシー範囲でフィルタリングします。
- スループット – リクエストスループット範囲でフィルタリングします。
- 失敗率 – 失敗率の範囲でフィルタリングします。

サービスの概要

サービス詳細ビューを開くには、サービステーブルでサービス名を選択します。概要タブには、選択したサービスのメトリクスタイルと時系列グラフが表示されます。

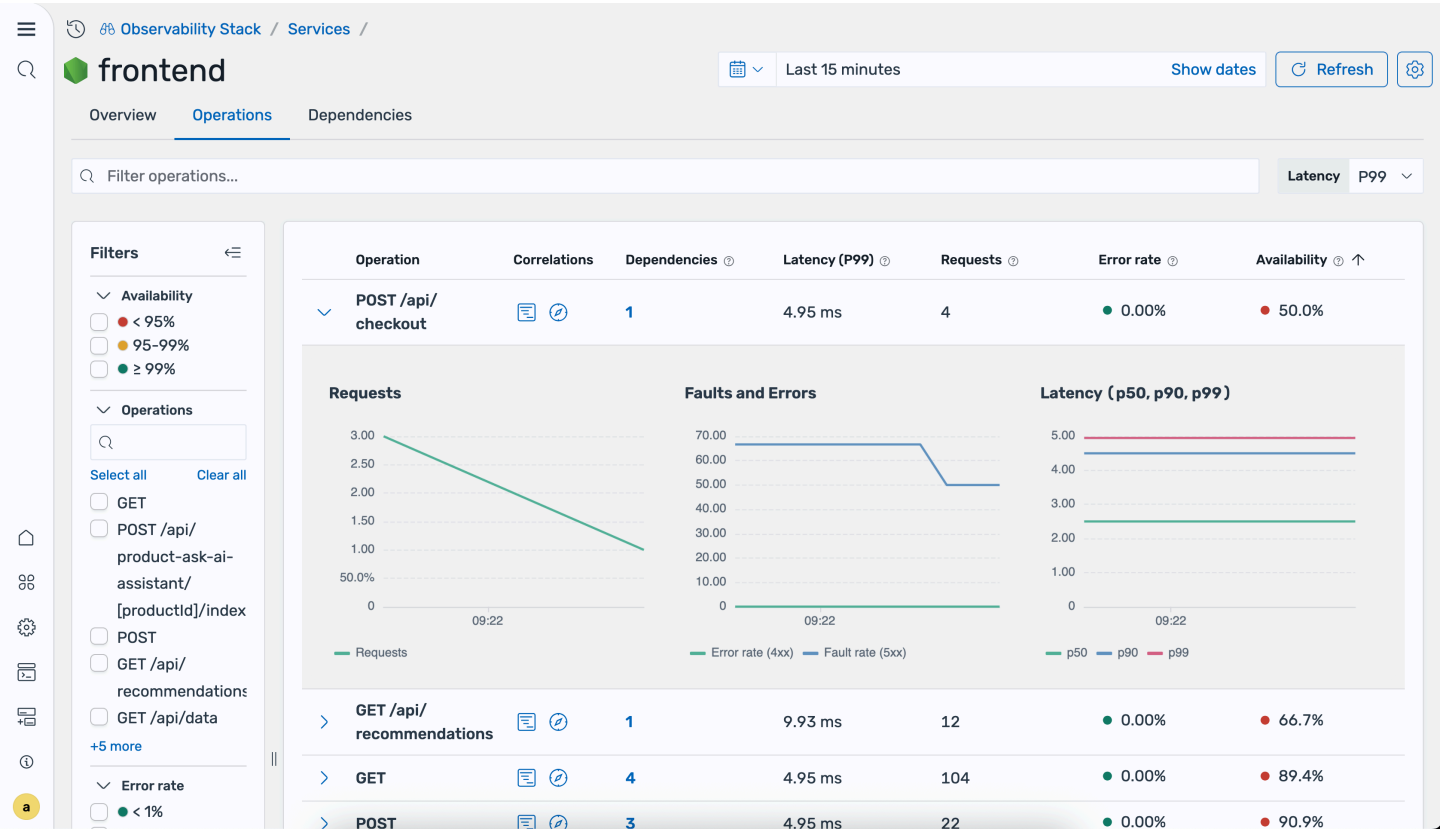


概要タブには、次の時系列グラフが含まれます。

- サービス依存関係別のレイテンシー – ダウンストリーム依存関係別に分類された P50、P90、P99 のレイテンシー。
- オペレーション別のリクエスト – サービスの各オペレーションのリクエストボリューム。
- オペレーション別の可用性 – 各オペレーションの成功レスポンスの割合。
- オペレーション別の障害率とエラー率 — オペレーションごとの 5xx および 4xx レスポンスの割合。

オペレーション

オペレーションタブには、選択したサービスのオペレーションごとの内訳が表示されます。任意の列でテーブルをソートして、問題のあるオペレーションを特定できます。



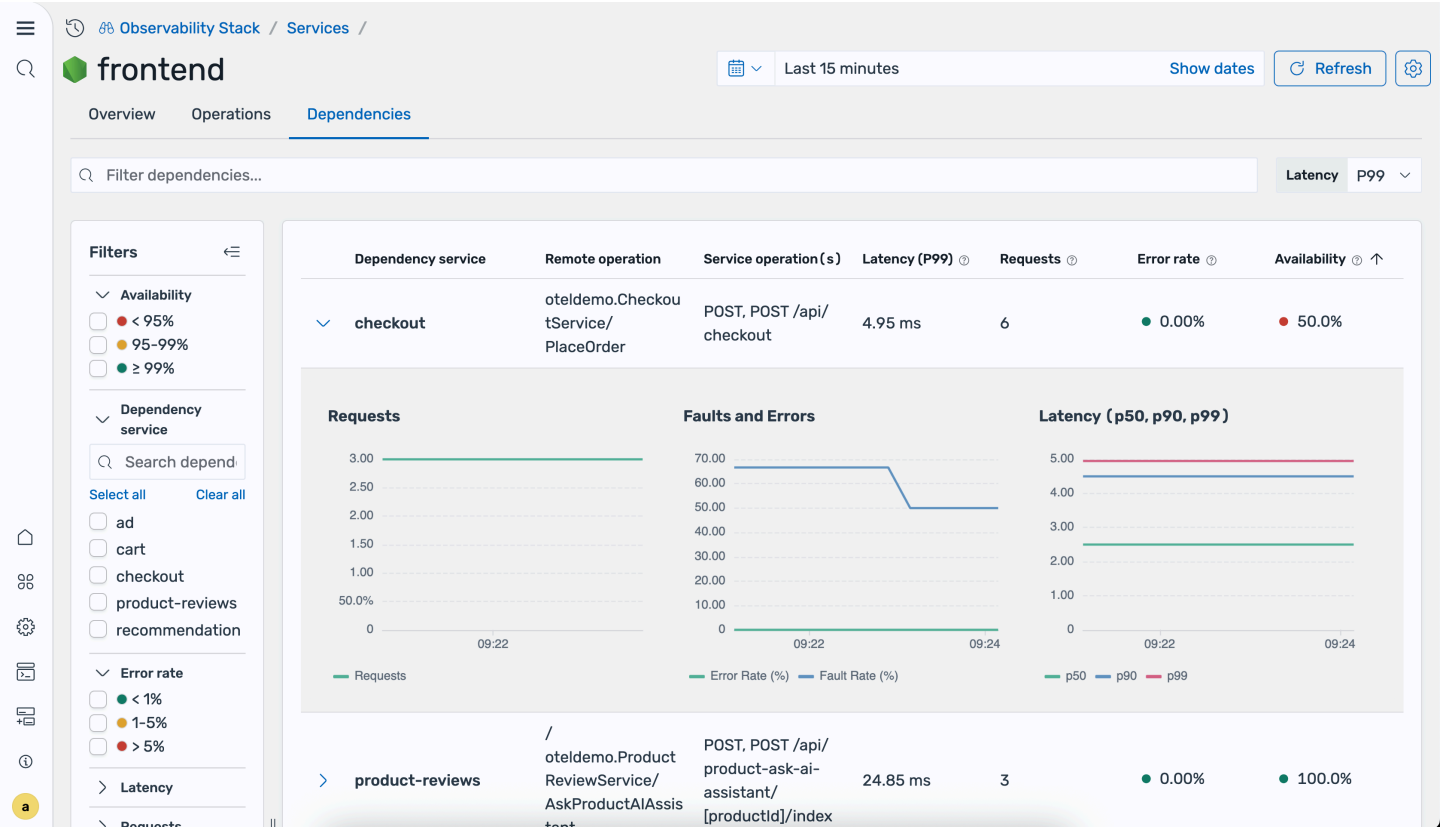
次の表は、オペレーションテーブルの列を示しています。

列	[Description] (説明)
オペレーション名	オペレーションの名前。

列	[Description] (説明)
P50/P90/P99 レイテンシー	オペレーションの 50 パーセンタイル、90 パーセンタイル、99 パーセンタイルのレイテンシー。
Total requests	選択した時間範囲におけるオペレーションのリクエストの合計数。
エラー率	エラーを返したリクエストの割合。
可用性	オペレーションの成功したレスポンスの割合。

依存関係

依存関係タブには、選択したサービスが呼び出すダウンストリームサービスが表示されます。

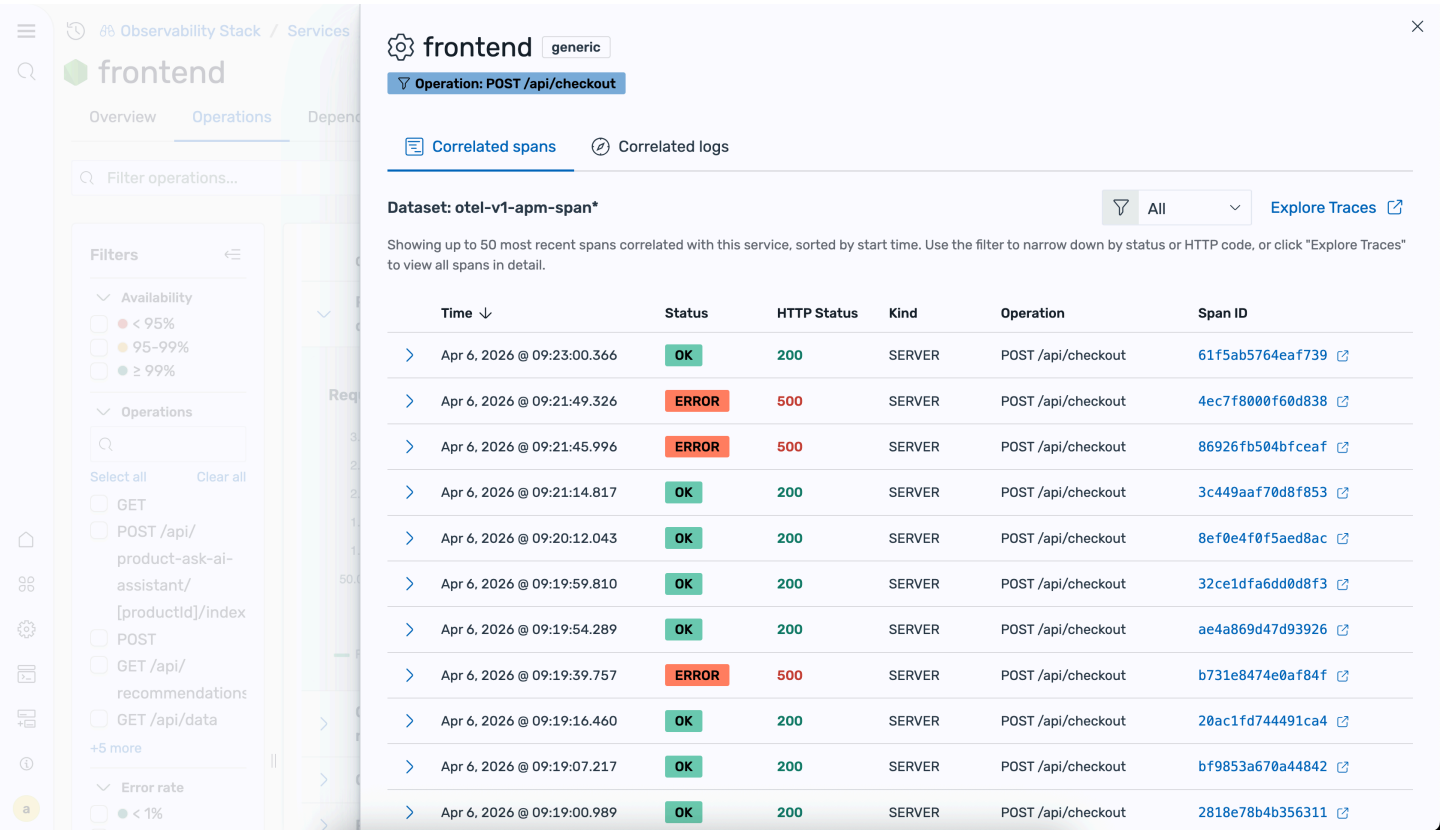


次の表は、依存関係テーブルの列を示しています。

列	[Description] (説明)
依存関係サービス	ダウンストリームサービスの名前。
リモートオペレーション	ダウンストリームサービスで呼び出された オペレーション。
サービスオペレーション	この依存関係を呼び出す現在のサービスのオペレーション。
P99/P90/P50 レイテンシー	依存関係パスの 99 パーセンタイル、90 パーセンタイル、50 パーセンタイルのレイテンシー。
Total requests	選択した時間範囲内の依存関係へのリクエストの合計数。
エラー率	エラーを返した依存関係へのリクエストの割合。
可用性	依存関係からの正常なレスポンスの割合。

相関関係

サービス詳細ビューには、サービスメトリクスから関連するトレースやログに直接移動できるコンテキスト内相関が表示されます。相関関係を使用して、レイテンシーの急増やエラー率の増加の根本原因を調査できます。



次の相関オプションを使用できます。

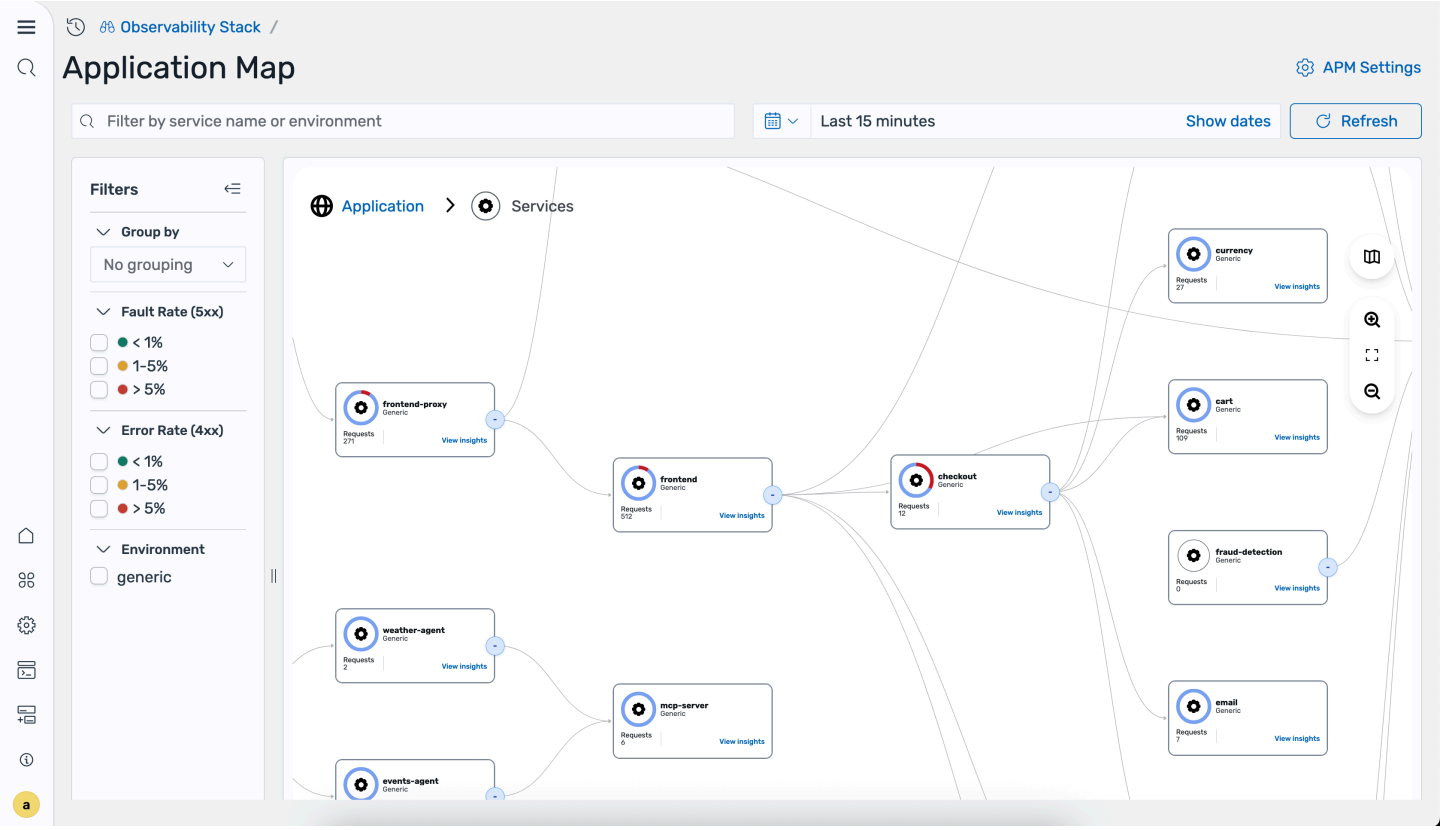
- 関連するトレースの表示 – 選択したサービスまたはオペレーションのフィルタリングされたトレースビューを開きます。
- 関連ログの表示 – 選択したサービスまたはオペレーションのフィルタリングされたログビューを開きます。
- 属性でフィルタリング – 特定のスパン属性で相関結果を絞り込みます。

アプリケーションマップ

アプリケーションマップは、OpenSearch Ingestion が`otel_apm_service_map`プロセッサを使用してトレースデータから自動生成するインタラクティブなトポロジ視覚化です。マップには、REDメトリクス (レート、エラー、期間) でオーバーレイされた通信パターンを示す指向性エッジを持つノードとしてサービスが表示されます。

アプリケーションマップにアクセスするには、OpenSearch UI のオブザーバビリティワークスペースに移動し、APM > アプリケーションマップを選択します。

次の図は、アプリケーションマップを示しています。



マップには、サービスごとに次の RED メトリクスが表示されます。

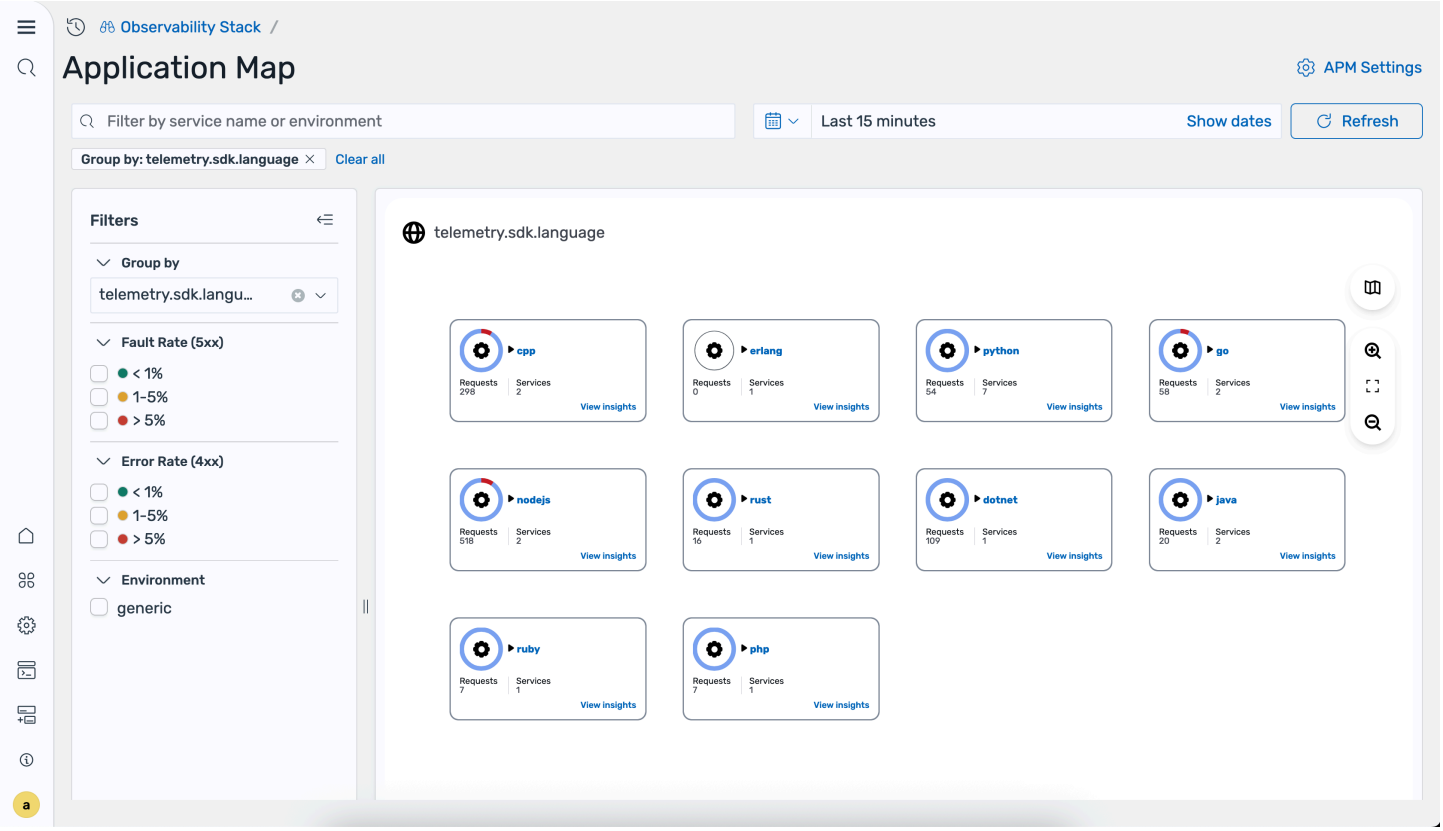
- レート – サービスによって処理された 1 秒あたりのリクエスト数。
- エラー – 4xx および 5xx レスポンスの割合。
- 期間 – サービスの P50 および P99 レイテンシー。

`otel_apm_service_map` プロセッサはこれらのメトリクスを生成し、リモート書き込みを通じて Amazon Managed Service for Prometheus に保存します。

トポロジの視覚化は、サービスをノードとして、通信方向をエッジとして表します。カラーコーディングは、各サービスのヘルスステータスを示します。OpenSearch Ingestion が新しいトレースデータを取り込むと、マップは自動的に更新されます。

サービスのグループ化

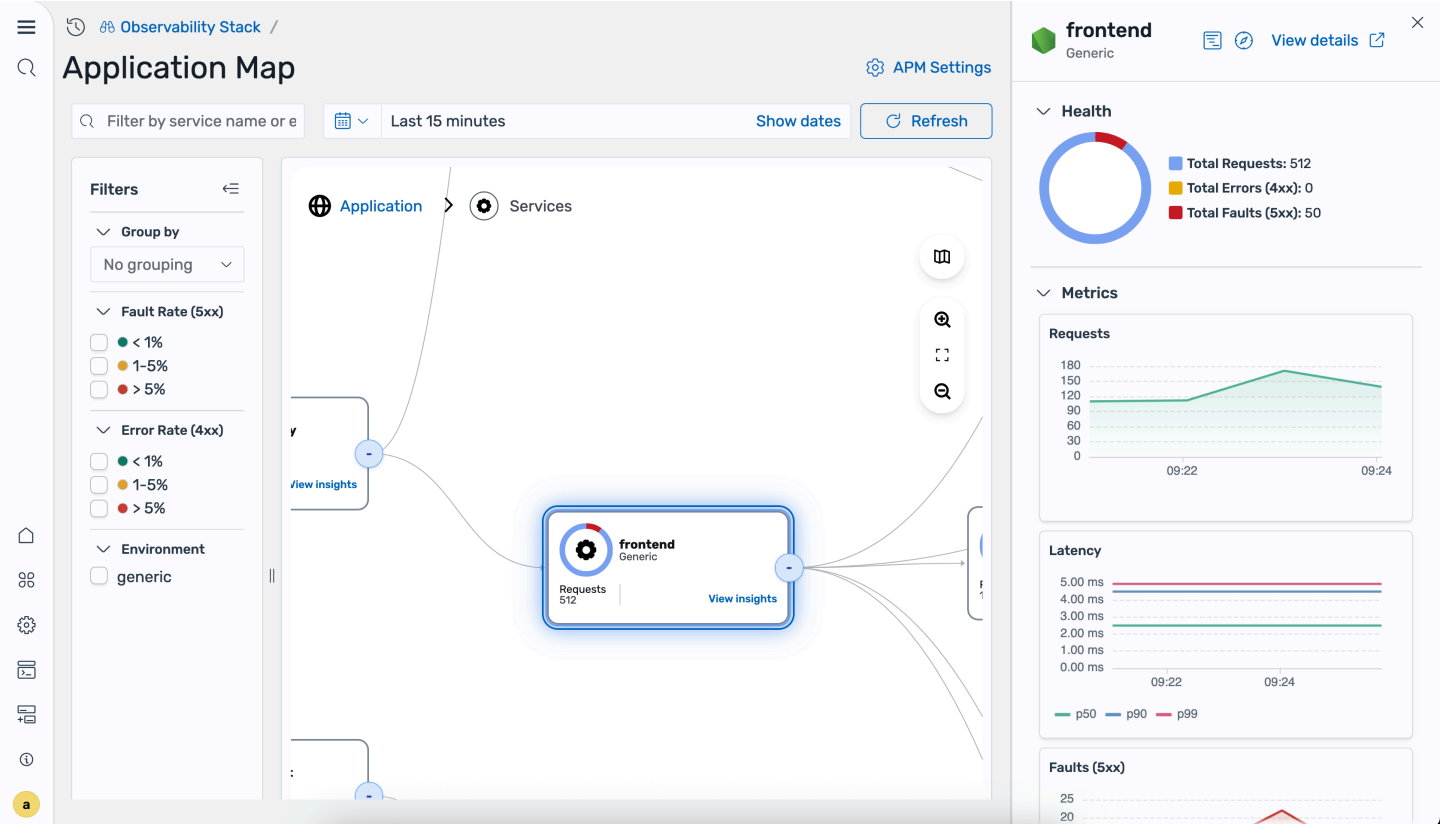
サービスは、プログラミング言語、チーム、環境などの属性別にグループ化できます。グループごとの属性を選択すると、マップはトポロジグラフからカードグリッドビューに切り替わります。各カードは、同じ属性値を共有するサービスのグループを表します。



使用可能なグループごとの属性は、OpenSearch Ingestion の`otel_apm_service_map`プロセッサ設定の `group_by_attributes`設定によって決まります。

ノードの詳細の表示

サービスの詳細を表示するには、マップでノードを選択します。詳細パネルが開き、以下のセクションが表示されます。



Health セクションには、次の概要メトリクスが表示されます。

- Total Requests
- エラー合計 4xx
- 合計障害数 5xx

メトリクスセクションには、次の時系列グラフが表示されます。

- リクエスト
- レイテンシー P50/P90/P99
- 障害 5xx
- エラー 4xx

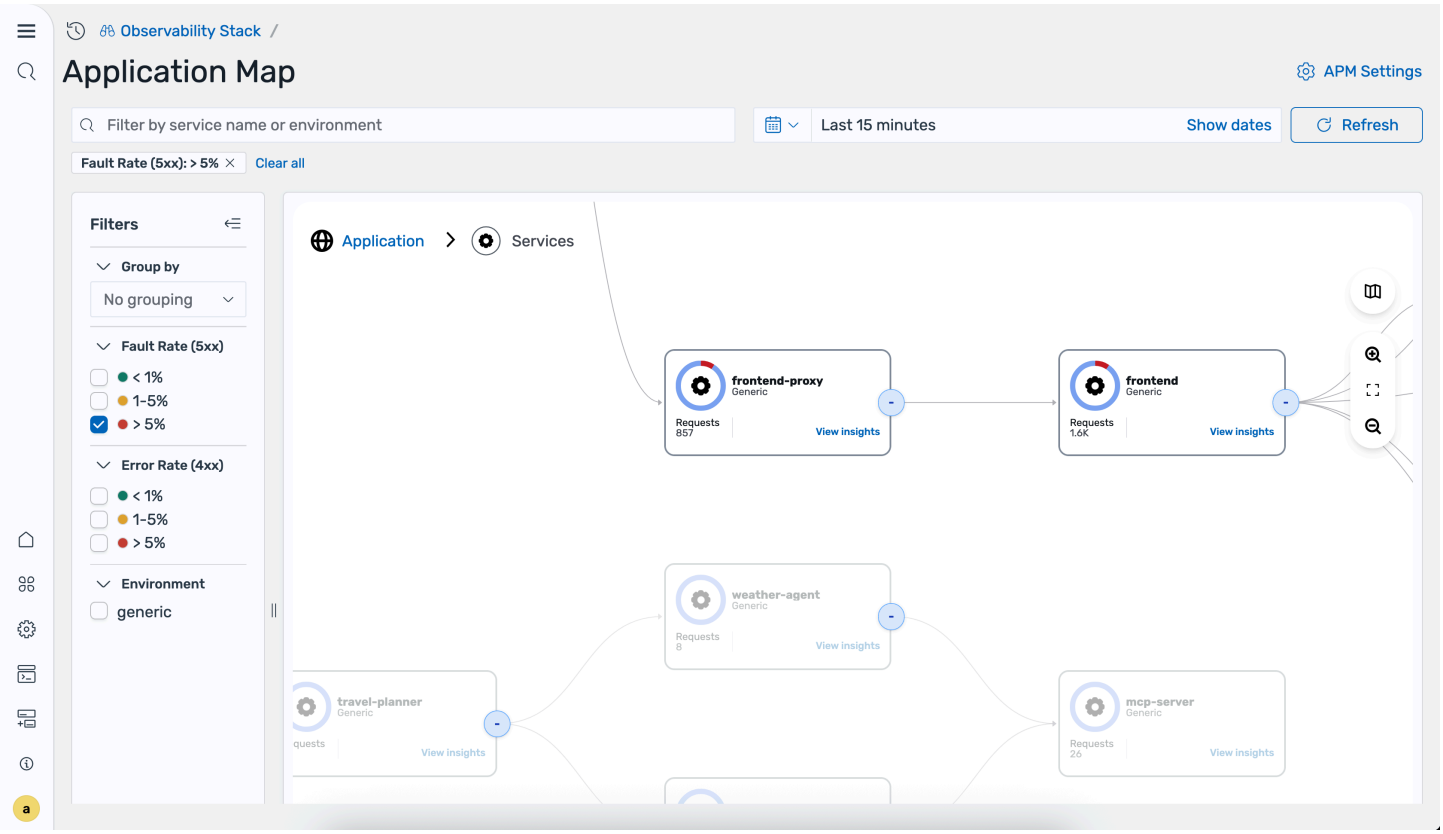
詳細の表示を選択して、選択したサービスのサービス詳細ビューに移動します。

マップのフィルタリング

アプリケーションマップは、次のフィルターを使用してフィルタリングできます。

- 障害率 – サーバー側の障害率 (5xx) でサービスをフィルタリングします。
- エラー率 – クライアント側のエラー率 (4xx) でサービスをフィルタリングします。
- 環境 – デプロイ環境によってサービスをフィルタリングします。

次の図は、エラー率でフィルタリングされたマップを示しています。



コンテキスト内相関

トポロジビューから関連するトレースとログに直接移動できます。任意のサービスノードから、次の相関オプションを使用できます。

- 関連するトレースの表示 – 選択したサービスのフィルタリングされたトレースビューを開きます。
- 関連ログの表示 – 選択したサービスのフィルタリングされたログビューを開きます。

AI オブザーバビリティ

OpenSearch の AI オブザーバビリティは、AI エージェントと大規模言語モデル (LLM) ワークフローをモニタリング、デバッグ、最適化するためのend-to-endのツールを提供します。GenAI セマン

ティック規則に基づいて構築され、OpenTelemetry (OTel) とネイティブに統合されているため、AI アプリケーションが本番環境でどのように動作するかを完全に可視化できます。

AI オブザーバビリティには、次の機能が含まれます。

- エージェントトレース – エージェントオーケストレーションステップ、LLM 呼び出し、ツール呼び出し、取得オペレーション全体の階層実行トレースをキャプチャします。
- GenAI セマンティック規則 – `gen_ai.system`、、 などの標準化された OTel 属性`gen_ai.usage.input_tokens`を使用して`gen_ai.request.model`、AI 固有のテレメトリを記述します。
- 自動計測 – OpenAI、Anthropic、Amazon Bedrock、LangChain、20 を超える追加ライブラリなど、一般的な AI フレームワークとプロバイダーからのトレースを自動的にキャプチャします。
- PPL クエリ – OpenSearch UI から直接 Piped Processing Language (PPL) を使用してトレースデータをクエリおよび集計します。

開始方法

このセクションでは、AI エージェントの計測、Amazon OpenSearch Service へのトレースの送信、OpenSearch UI でのトレースの表示について説明します。

SDK をインストールするには

OpenTelemetry GenAI 計測パッケージをインストールします。

```
pip install opentelemetry-instrumentation-openai-v2 opentelemetry-sdk opentelemetry-
exporter-otlp
```

エージェントコードを計測するには

次の例は、OTel SDK を登録し、`@observe`デコレータでエージェント関数に注釈を付け、GenAI 属性でスパンを強化する方法を示しています。

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import BatchSpanProcessor
from opentelemetry.exporter.otlp.proto.grpc.trace_exporter import OTLPSpanExporter

# Register the tracer provider
provider = TracerProvider()
```

```
processor = BatchSpanProcessor(
    OTLPSpanExporter(endpoint="<your-osis-endpoint>")  # Use your OpenSearch Ingestion
 endpoint with SigV4
)
provider.add_span_processor(processor)
trace.set_tracer_provider(provider)

tracer = trace.get_tracer(__name__)

# Decorate your agent function
@observe
def run_agent(prompt: str):
    with tracer.start_as_current_span("invoke_agent") as span:
        span.set_attribute("gen_ai.operation.name", "invoke_agent")
        span.set_attribute("gen_ai.system", "openai")
        span.set_attribute("gen_ai.request.model", "gpt-4")

        # Enrich with token usage after the LLM call
        response = call_llm(prompt)
        span.set_attribute("gen_ai.usage.input_tokens", response.input_tokens)
        span.set_attribute("gen_ai.usage.output_tokens", response.output_tokens)
        return response
```

> **Note**
>
> Amazon OpenSearch Service にトレースを送信するときは、localhost エンドポイントではなくSigV4 認証で OpenSearch Ingestion パイプラインエンドポイントを使用します。OpenSearch Ingestion パイプラインの設定の詳細については、「」を参照してくださいAmazon OpenSearch Ingestion。

OpenSearch UI でトレースを表示するには

計測されたアプリケーションがトレースデータを送信したら、OpenSearch UI で調べることができます。オブザーバビリティワークスペースで、左側のナビゲーションで検出を展開し、エージェントトレースを選択します。

エージェントトレース UI

OpenSearch UI の エージェントトレースページには、LLM エージェント実行トレースを探索、デバッグ、モニタリングするための専用インターフェイスが用意されています。これにより、デベロッ

パーやプラットフォームオペレーターは、階層トレースビュー、詳細フライアウト、フローの視覚化、メトリクスの集計など、エージェント AI アプリケーションを完全に観察できます。

アーキテクチャ

次の図は、計測されたアプリケーションから エージェントトレース UI へのデータフローを示しています。

```
LLM Application (with OTel SDK + GenAI instrumentation)
    |
    |  OTLP (gRPC/HTTP)
    v
OTel Collector (batch, transform)
    |
    +---- OTLP ----> OpenSearch Ingestion --> OpenSearch (otel-v1-apm-span-*)
    |
    +---- Prometheus Remote Write --> Prometheus (metrics)
                                            |
                                            v
                                OpenSearch UI
                                +-- Agent Traces Plugin
```

前提条件

エージェントトレースを使用する前に、以下があることを確認してください。

- `otel-v1-apm-span-*` インデックスでインデックス化されたトレースデータを持つ OpenSearch クラスター。
- LLM アプリケーションで GenAI セマンティック規則が有効になっている OpenTelemetry 計測。
- OpenSearch にスパンを取り込むように`otel_trace_raw`プロセッサで設定された OpenSearch Ingestion。
- OpenSearch UI で有効になっている PPL クエリのサポート。

必要なスパン属性

エージェントトレースでは、トレースデータを正しくレンダリングするために特定のスパン属性が必要です。次の表は、コアフィールドと GenAI 固有の属性を示しています。

コアスパンフィールド

各スパンには、次のコアフィールドを含める必要があります。

フィールド	タイプ	説明
traceId	文字列	トレース全体の一意の識別子。
spanId	String	このスパンの一意の識別子。
parentSpanId	String	親スパンの識別子。ルートスパンの場合は空です。
startTime	タイムスタンプ	スパンが開始された時刻。
endTime	タイムスタンプ	スパンが終了した時刻。
durationInNanos	Long	ナノ秒単位のスパンの期間。
status.code	整数	スパンステータスコード (0 = 未設定、1 = OK、2 = エラー) 。

GenAI 属性

次のgen_ai.*属性は、エージェントトレース UI で AI 固有の機能を有効にします。

属性	値の例	説明
gen_ai.operation.name	chat	GenAI オペレーションのタイプ。スパンカテゴリを決定します。
gen_ai.system	openai	AI システムまたはプロバイダー。
gen_ai.request.model	gpt-4	リクエストに使用されるモデル。
gen_ai.usage.input_tokens	150	消費された入力トークンの数。

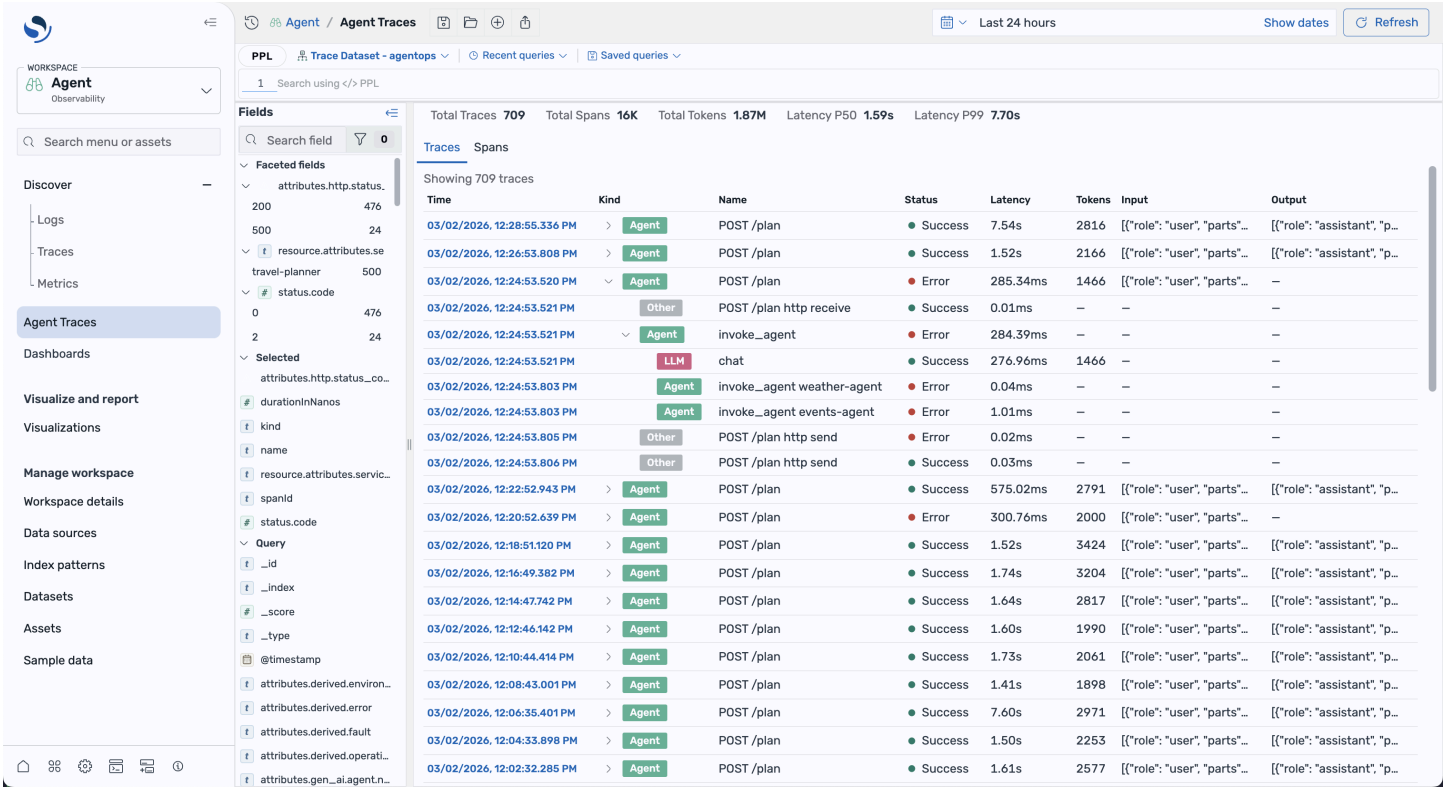
属性	値の例	説明
gen_ai.usage.output_tokens	85	生成された出力トークンの数。
gen_ai.response.finish_reasons	["stop"]	モデルが生成を停止した理由。

ページレイアウトとメトリクスバー

エージェントトレースページの上部には、表示されているすべてのトレースの主要な統計をまとめたメトリクスバーが表示されます。メトリクスには、合計トレース数、平均期間、エラー率、トークン使用量が含まれます。これらの値は、時間フィルターとクエリに基づいて動的に更新されます。

トレースタブ

トレースタブには、現在のクエリと時間範囲に一致するすべてのルートエージェントトレースが一覧表示されます。各行は 1 つのエージェント呼び出しを表します。



次の表は、トレーステーブルの列を示しています。

列	[Description] (説明)
トレース ID	トレースの一意の識別子。リンクを選択して、トレースの詳細フライアウトを開きます。
エージェント名	トレースを開始したエージェントの名前。
ステータス	全体的なトレースステータス (OK または エラー) 。
時間	トレースの最初のスパンから最後のスパンまでの合計時間。
スパン	トレース内のスパンの合計数。
入力トークン	トレース内のすべての LLM 呼び出しで消費された入力トークンの合計。
出力トークン	トレース内のすべての LLM 呼び出しで生成された出力トークンの合計。
開始時間	トレースが開始されたときのタイムスタンプ。

スパンカテゴリ

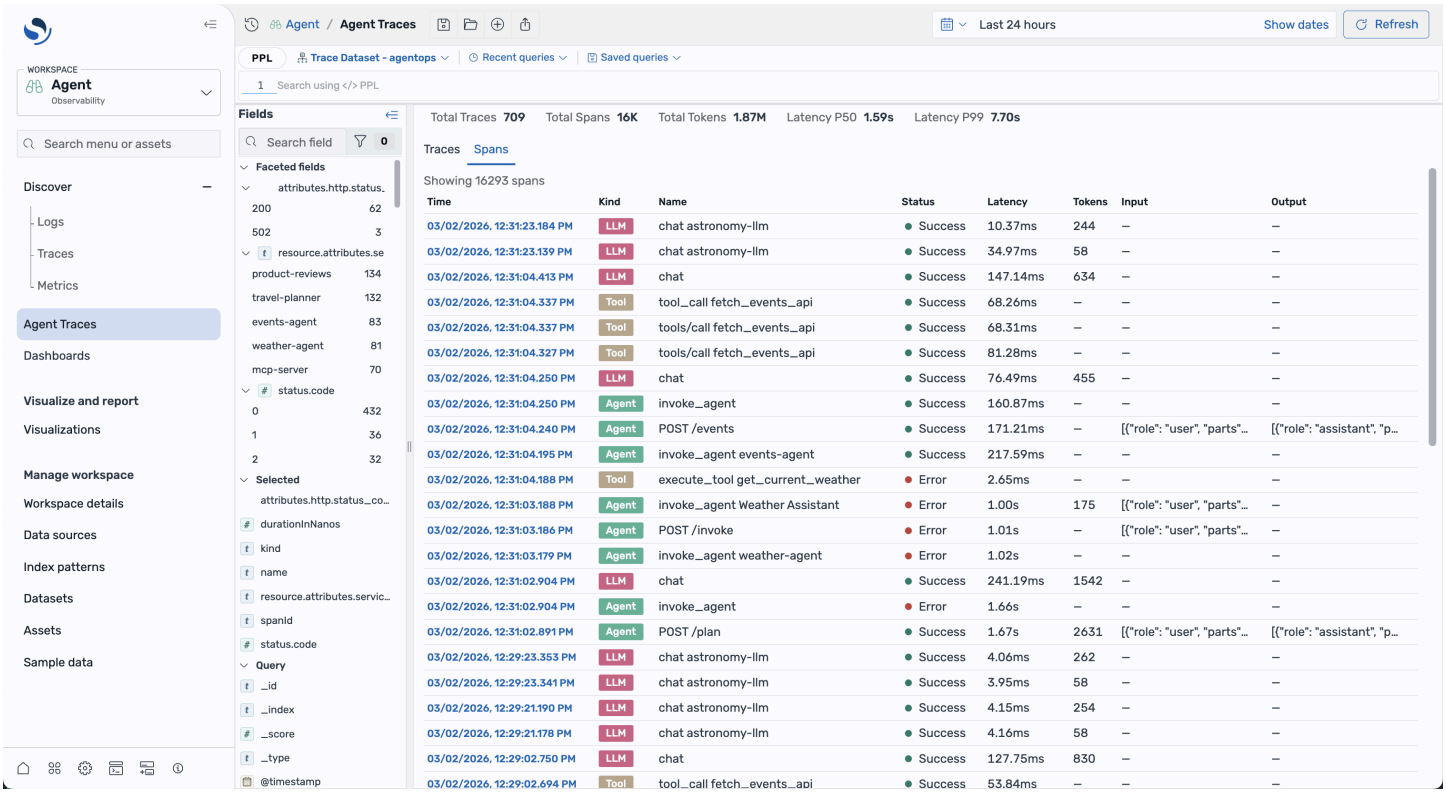
スパンは `gen_ai.operation.name` 属性に基づいて分類されます。各カテゴリは、UI に一意の色とアイコンで表示されます。

オペレーション名	Category	説明
`invoke_agent` , `create_agent`	[エージェント]	エージェントオーケストレーションステップ。
`chat`	LLM	LLM チャット完了呼び出し。
`text_completion` , `generate_content`	コンテンツ	テキスト生成オペレーション。

オペレーション名	Category	説明
execute_tool	ツール	ツール呼び出し。
embeddings	埋め込み	埋め込み生成。
retrieval	取得	データ取り出しオペレーション。

スパンタブ

スパンタブには、すべてのトレースにまたがる個々のスパンが表示されます。スパンをフィルタリングおよびソートして、特定のオペレーションを検索できます。



トレースの詳細フライアウト

トレース ID を選択すると、フライアウトパネルが開き、次の 2 つの主要エリアが表示されます。

- 左パネル – トレースツリーを表示します。トレースツリーには、スパン間の階層的な親子関係が表示されます。また、エージェントの実行パスを視覚化するフロー DAG (有向非巡回グラフ) も含まれています。

- 右側のパネル – 2 つのタブがあります。詳細タブには、スパン属性、リソース属性、および GenAI 固有のメタデータが表示されます。タイムラインタブには、スパン期間とそのタイミング関係のウォーターフォールグラフが表示されます。

トレースのクエリ

エージェントトレースは、すべてのデータフェッチに PPL (パイプ処理言語) を使用します。ページ上部のクエリパネルにクエリを記述できます。

ルートトレースを一覧表示するには

次のクエリは、100 個の最新のルートエージェントトレースを返します。

```
source = otel-v1-apm-span-*
| where parentSpanId = "" AND isnotnull(`attributes.gen_ai.operation.name`)
| sort - startTime
| head 100
```

トレースのすべてのスパンを取得するには

次のクエリは、特定のトレースの完全なスパンツリーを返します。

```
source = otel-v1-apm-span-*
| where traceId = "trace-id"
| head 1000
```

集計メトリクスを計算するには

次のクエリは、モデル別にグループ化された平均所要時間とトークン使用量の合計を計算します。

```
source = otel-v1-apm-span-*
| where isnotnull(`attributes.gen_ai.request.model`)
| stats avg(durationInNanos) as avg_duration,
        sum(`attributes.gen_ai.usage.input_tokens`) as total_input_tokens,
        sum(`attributes.gen_ai.usage.output_tokens`) as total_output_tokens
    by `attributes.gen_ai.request.model`
```

Amazon OpenSearch Service のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – クラウドで AWS AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。「AWS 」 コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon OpenSearch Service に適用されるコンプライアンスプログラムについては、コンプライアンスプログラムによるAWS のサービスを参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、OpenSearch Service を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます 以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するようにいずれかの OpenSearch Service を設定する方法について説明します。また、OpenSearch Service リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon OpenSearch Service でのデータ保護
- Amazon OpenSearch Service での Identity and Access Management
- サービス間の混乱した代理の防止
- Amazon OpenSearch Service のきめ細かなアクセスコントロール
- Amazon OpenSearch Service のコンプライアンス検証
- Amazon OpenSearch Service の耐障害性
- Amazon OpenSearch Service n の JWT 認証と承認
- Amazon OpenSearch Service のインフラストラクチャセキュリティ
- OpenSearch Dashboards の SAML 認証

- IAM アイデンティティセンターによる OpenSearch に対する信頼できる ID の伝達のサポート
- OpenSearch Dashboards の Amazon Cognito 認証の設定
- Amazon OpenSearch Service 用のサービスにリンクされたロールの使用

Amazon OpenSearch Service でのデータ保護

責任 AWS 共有モデル、Amazon OpenSearch Service でのデータ保護に適用されます。 https://aws.amazon.com/compliance/shared-responsibility-model/このモデルで説明されているように、AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して OpenSearch Service AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Amazon OpenSearch Service の保管中のデータの暗号化

OpenSearch Service ドメインでは、保管中のデータの暗号化という、データへの不正アクセスを防止するのに役立つセキュリティ機能が提供されます。この機能では、暗号化キーの保存と管理に AWS Key Management Service (AWS KMS) を使用し、暗号化を実行するために 256 ビットキー (AES-256) を使用する Advanced Encryption Standard アルゴリズムを使用します。有効にすると、この機能によりドメインの次の要素が暗号化されます。

- すべてのインデックス (UltraWarm ストレージのものを含む)
- OpenSearch ログ
- スワップファイル
- アプリケーションディレクトリのその他すべてのデータ
- 自動スナップショット

以下は、保管中のデータの暗号化を有効にするときに暗号化されませんが、追加のステップを行って保護することができます。

- 手動スナップショット: 現在、 AWS KMS キーを使用して手動スナップショットを暗号化することはできません。ただし、S3 で管理されたキーまたは KMS キーによるサーバー側の暗号化を使用してスナップショットリポジトリとして使用しているバケットを暗号化できます。手順については、「the section called “手動スナップショットレポジトリの登録”」を参照してください。
- スローログとエラーログ: ログを発行して暗号化する場合は、OpenSearch Service ドメインと同じ AWS KMS キーを使用して CloudWatch Logs ロググループを暗号化できます。詳しくは、Amazon CloudWatch Logs ユーザーガイドのAWS Key Management Serviceを使用して CloudWatch Logs のログデータを暗号化するを参照してください。

> **Note**
>
> UltraWarm またはコールドストレージがドメインで有効になっている場合は、既存のドメインで保管中の暗号化を有効にすることはできません。まず UltraWarm またはコールドストレージを無効にし、保管中の暗号化を有効にしてから、UltraWarm またはコールドストレージを再度有効にする必要があります。UltraWarm またはコールドストレージにインデックスを保持したい場合は、UltraWarm またはコールドストレージを無効にする前に、それらのインデックスをホットストレージに移動する必要があります。

OpenSearch Service は、対称暗号化 KMS キーのみをサポートしており、非対称暗号化 KMS キーはサポートしていません。対称キーを作成する方法については、「AWS Key Management Service デベロッパーガイド」の「Create a KMS key」を参照してください。

保存時の暗号化が有効になっているかどうかにかかわらず、すべてのドメインは AES-256 または OpenSearch Service 管理キーを使用してカスタムパッケージを自動的に暗号します。

アクセス許可

OpenSearch Service コンソールを使用して保管中のデータの暗号化を設定するには、次のアイデンティティベースのポリシーなど AWS KMS、 への読み取りアクセス許可が必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:List*",
        "kms:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 所有キー以外のキーを使用する場合は、キーの許可を作成するアクセス許可も必要です。通常、これらの許可は、キーの作成時に指定するリソースベースのポリシーの形式になります。

キーを OpenSearch Service 固有にする場合、キーポリシーに kms:ViaService 条件を追加できます。

```
"Condition": {
  "StringEquals": {
    "kms:ViaService": "es.us-west-1.amazonaws.com"
  },
  "Bool": {
    "kms:GrantIsForAWSResource": "true"
  }
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMSでのキーポリシー」を参照してください。

保管中のデータの暗号化の有効化

新しいドメインでの保管中のデータの暗号化には OpenSearch または Elasticsearch 5.1 以降が必要です。既存のドメインでそれを有効にするには、OpenSearch または Elasticsearch 6.7 以降が必要です。

保管中のデータの暗号化を有効にするには (コンソール)

1. AWS コンソールでドメインを開き、アクションとセキュリティ設定の編集を選択します。
2. 暗号化した状態で、保管中のデータの暗号化を有効にするを選択します。
3. 使用する AWS KMS キーを選択し、変更の保存を選択します。

設定 API を使って、暗号化を有効にすることもできます。次のリクエストは、既存ドメインで保存中のデータの暗号化を有効にします。

```
{
   "ClusterConfig":{
      "EncryptionAtRestOptions":{
         "Enabled": true,
         "KmsKeyId":"arn:aws:kms:us-east-1:123456789012:alias/my-key"
      }
   }
```

```
}
```

KMS キーを無効化または削除

ドメインの暗号化に使用したキーを無効化または削除すると、そのドメインにアクセスできなくなります。OpenSearch サービスが、KMS キーにアクセスできないことを通知します。すぐにキーを再度有効にして、ドメインにアクセスします。

キーが削除された場合、OpenSearch Service チームはデータの復旧をサポートできません。 は、少なくとも 7 日間の待機期間後にのみキー AWS KMS を削除します。キーの削除が保留中の場合は、削除をキャンセルするか、手動スナップショットを取って、ドメインのデータの損失を防ぎます。

保管中のデータの暗号化の無効化

保管中のデータを暗号化するようにドメインを設定した後、設定を無効にすることはできません。代わりに、既存のドメインの手動スナップショットを作成し、別のドメインを作成してからデータを移行して、古いドメインを削除することができます。

保管中のデータを暗号化するドメインのモニタリング

保管中のデータを暗号化するドメインには、`KMSKeyError` と `KMSKeyInaccessible` の 2 つの追加のメトリクスがあります。これらのメトリクスは、ドメインの暗号化キーに問題が発生した場合にのみ表示されます。これらのメトリクスの詳細については、「the section called “クラスターメトリクス”」を参照してください。OpenSearch Service コンソールまたは Amazon CloudWatch コンソールのいずれかを使用して表示できます。

> **Tip**
>
> 各メトリクスは、ドメインの重大な問題を表しているため、両方の CloudWatch アラームを作成することをお勧めします。詳細については、「the section called “推奨される CloudWatch アラーム”」を参照してください。

その他の考慮事項

- 自動キーローテーションは AWS KMS キーのプロパティを保持するため、ローテーションは OpenSearch データにアクセスする機能には影響しません。暗号化された OpenSearch Service ドメインは、新しいキーの作成と古いキーのすべてのレファレンスの更新を含むマスターキーローテーションをサポートしません。詳細については、 AWS Key Management Service デベロッパーガイドのAWS KMS 「キーのローテーション」を参照してください。

- 特定のインスタンスタイプは、保管中のデータの暗号化をサポートしていません。詳細については、「the section called “サポートされるインスタンスタイプ”」を参照してください。
- 保管中のデータを暗号化するドメインでは、自動スナップショット用に別のリポジトリ名を使用します。詳細については、「the section called “スナップショットからのデータの復元”」を参照してください。
- 静止時に暗号化を有効にすることを強くお勧めしますが、CPU オーバーヘッドが増加し、数ミリ秒の遅延が発生する可能性があります。ただし、ほとんどのユースケースはこれらの違いに敏感ではなく、影響の大きさは、クラスター、クライアント、および使用プロファイルの構成によって異なります。

Amazon OpenSearch Service のノード間の暗号化

ノード間の暗号化では、Amazon OpenSearch Service のデフォルトの機能の上に追加のセキュリティレイヤーを提供します。

各 OpenSearch Service ドメインは、ドメインが VPC アクセスを使用しているかどうかにかかわらず、専用の VPC 内に存在します。このアーキテクチャにより、潜在的な攻撃者から OpenSearch ノード間のトラフィックを傍受し、安全なクラスターが維持されます。ただし、デフォルトでは、VPC 内のトラフィックは暗号化されません。ノード間の暗号化により、VPC 内のすべての通信で TLS 1.2 暗号化が有効になります。

HTTPS 経由でデータを OpenSearch Service に送信する場合、ノード間の暗号化を使用すると、OpenSearch がクラスター全体に分散 (および再分散) する際、データは暗号化された状態で維持されます。HTTP 経由で暗号化されていないデータが到着するあ場合、OpenSearch Service はそれがクラスターに到達した後で暗号化します。ドメインへのすべてのトラフィックは、 コンソール AWS CLI、または設定 API を使用して HTTPS 経由で到着するように要求できます。

きめ細かなアクセスコントロールを有効にする場合は、ノード間の暗号化が必須です。

ノード間の暗号化を有効にする

新しいドメインでのノード間の暗号化には、任意のバージョンの OpenSearch、または Elasticsearch 6.0 以降が必要です。既存のドメインでノード間の暗号化を有効にするには、任意のバージョンの OpenSearch、または Elasticsearch 6.7 以降が必要です。 AWS コンソールで既存のドメインを選択し、[アクション] から [セキュリティ設定の編集] を選択します。

または、 AWS CLI または 設定 API を使用することもできます。詳細については、「AWS CLI Command Reference」と「OpenSearch Service API reference」(OpenSearch Service API リファレンス) を参照してください。

ノード間の暗号化を無効にする

ノード間の暗号化を使用するようにドメインを設定した後、設定を無効にすることはできません。代わりに、暗号化されたドメインの手動スナップショットを作成し、別のドメインを作成してからデータを移行して、古いドメインを削除することができます。

Amazon OpenSearch Service での Identity and Access Management

Amazon OpenSearch Service には、ドメインへのアクセスを制御するいくつかの方法が用意されています。このトピックでは、さまざまなポリシータイプ、それぞれがやり取りする方法、および独自のカスタムポリシーを作成する方法を説明します。

> **Important**
>
> VPC は OpenSearch Service のアクセス制御へのいくつかの追加考慮事項の導入をサポートしています。詳細については、「the section called “VPC ドメインのアクセスポリシーについて”」を参照してください。

ポリシーのタイプ

OpenSearch Service はアクセスポリシーの 3 つのタイプをサポートしています。

- the section called “リソースベースのポリシー”
- the section called “アイデンティティベースのポリシー”
- the section called “IP ベースのポリシー”

リソースベースのポリシー

ドメインを作成するときに、ドメインアクセスポリシーと呼ばれる場合があるリソースベースのポリシーを追加します。これらのポリシーは、ドメインのサブリソースでプリンシパルが実行するアクションを指定します (cross-cluster 検索を除く)。サブリソースには、OpenSearch インデックスおよび API が含まれます。IAM の Principal JSON ポリシー要素は、アクセスを許可するアカウント、ユーザー、またはロール (プリンシパル) を指定します。Resource JSON ポリシー要素は、これらのプリンシパルがアクセスできるサブリソースを指定します。

例えば、次のリソースベースのポリシーでは、test-domain のサブリソースへのフルアクセス (es:*) が test-user に付与されます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:*"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    }
  ]
}
```

このポリシーには、2 つの重要な考慮事項が適用されます。

- これらの権限はこのドメインのみに適用されます。他のドメインで同様のポリシーを作成しない限り、test-user は test-domain にしかアクセスできません。
- Resource 要素の末尾の /* は重要であり、リソースベースのポリシーがドメイン自体ではなく、ドメインのサブリソースにのみ適用されることを示します。リソースベースのポリシーでは、es:* アクションは es:ESHttp* と同等です。

  例えば、test-user はインデックス (GET https://search-test-domain.us-west-1.es.amazonaws.com/test-index) に対してリクエストを行うことができますが、ドメインの設定を更新できません (POST https://es.us-west-1.amazonaws.com/2021-01-01/opensearch/domain/test-domain/config)。2 つのエンドポイント間の違いに注目してください。設定 API にアクセスするには ID ベースのポリシーが必要です。

ワイルドカードを追加することで、インデックス名の一部を指定できます。次の例では、commerce で始まるすべてのインデックスを特定しています。

```
arn:aws:es:us-west-1:987654321098:domain/test-domain/commerce*
```

この場合、ワイルドカードの意味は、commerce で始まる名前の test-domain のインデックスに test-user がリクエストを送信できるということです。

test-user をさらに制限するには、次のポリシーを適用することができます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:ESHttpGet"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/commerce-
data/_search"
    }
  ]
}
```

これにより、test-user は commerce-data インデックスに対する検索で、1 つのオペレーションのみを実行できます。ドメインのその他すべてのインデックスはアクセス不可となり、、test-user は、es:ESHttpPut または es:ESHttpPost アクションを使用する許可なしにドキュメントを追加あるいは変更できなくなります。

次に、パワーユーザーのロールを設定することができます。このポリシーは、power-user-role にインデックスのすべての URI に対する HTTP GET メソッドおよび PUT メソッドへのアクセスを付与します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/power-user-role"
        ]
      },
      "Action": [
        "es:ESHttpGet",
        "es:ESHttpPut"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/commerce-
data/*"
    }
  ]
}
```

ドメインが VPC 内にあるか、きめ細かなアクセスコントロールを使用している場合は、オープンドメインアクセスポリシーを使用できます。それ以外の場合、ドメインアクセスポリシーには、プリンシパルまたは IP アドレスによる制限が含まれている必要があります。

使用できるすべてのアクションの詳細については、「the section called “ポリシーエレメントのリファレンス”」を参照してください。データをより細かく制御するには、きめ細かなアクセスコントロールとともにオープンドメインアクセスポリシーを使用します。

アイデンティティベースのポリシー

各 OpenSearch Service ドメインの一部であるリソースベースのポリシーとは異なり、 AWS Identity and Access Management (IAM) サービスを使用してアイデンティティベースのポリシーをユーザーまたはロールにアタッチします。リソースベースのポリシーと同様に、アイデンティティベースのポリシーは、サービスに誰がアクセスできるか、どのアクションキーを実行できるか、そして該当する場合には、これらのアクションを実行できるリソースを指定します。

これらを実行する実際の義務はないため、アイデンティティベースのポリシーはより一般的になる傾向があります。これにより、多くの場合、ユーザーが実行できる設定 API アクションのみ管理さ

れます。これらのポリシーの配置が完了したら、OpenSearch Service でリソースベースのポリシー (またはきめ細かなアクセスコントロール) を使用して、OpenSearch インデックスおよび API へのアクセスをユーザーに提供できます。

> **Note**
>
> AWS 管理AmazonOpenSearchServiceReadOnlyAccessポリシーを持つユーザーは、 コンソールでクラスターのヘルスステータスを表示できません。これらのユーザーがヘルスステータス (および他の OpenSearch データ) を確認できるようにするには、アクセスポリシーに `es:ESHttpGet` アクションを追加し、これをユーザーのアカウントまたはロールにアタッチします。

アイデンティティベースのポリシーはユーザーあるいはロール (プリンシパル) にアタッチされるため、JSON はプリンシパルを指定しません。次のポリシーは、`Describe` および `List` で始まるアクションへのアクセスを付与します。このアクションの組み合わせはドメインの設定への読み取り専用のアクセスを提供しますが、ドメイン自体に保存されたデータへのアクセスは提供しません。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "es:Describe*",
        "es:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

管理者には、OpenSearch Service およびすべてのドメインに保存された全データへのフルアクセス権がある場合があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "es:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

ID ベースのポリシーでは、タグを使用して設定 API へのアクセスを制御できます。例えば、次のポリシーでは、ドメインに team:devops タグがある場合、アタッチされたプリンシパルがドメインの設定を表示および更新できるようにします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Action": [
      "es:UpdateDomainConfig",
      "es:DescribeDomain",
      "es:DescribeDomainConfig"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "aws:ResourceTag/team": [
          "devops"
        ]
      }
    }
  }]
}
```

タグを使用して、OpenSearch API へのアクセスを制御することもできます。OpenSearch API のタグベースのポリシーは、HTTP メソッドにのみ適用されます。例えば、次のポリシーでは、ドメインに `environment:production` タグがある場合、接続されたプリンシパルが GET および PUT リクエストを OpenSearch API に送信できます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Action": [
      "es:ESHttpGet",
      "es:ESHttpPut"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "aws:ResourceTag/environment": [
          "production"
        ]
      }
    }
  }]
}
```

OpenSearch API をよりきめ細かく制御するには、きめ細かいアクセス制御 の使用を検討してください。

> **Note**
>
> 1 つ以上の OpenSearch API をタグベースのポリシーに追加した後、変更をドメインで有効にするには、単一の タグオペレーション (タグの追加、削除、変更など) を実行する必要があります。タグベースのポリシーに OpenSearch API オペレーションを含めるには、サービスソフトウェア R20211203 以降を使用している必要があります。

OpenSearch サービスは、OpenSearch API ではなく、設定 API の `RequestTag` および `TagKeys` グローバル条件キーをサポートします。これらの条件は、リクエスト内にタグを含む API 呼び出し

(CreateDomain、AddTags、および RemoveTags など) にのみ適用されます。次のポリシーでは、アタッチされたプリンシパルがドメインを作成できるようにしますが、それは、team:it タグをリクエストに含めた場合のみです。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "es:CreateDomain",
      "es:AddTags"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/team": [
          "it"
        ]
      }
    }
  }
}
```

アクセスコントロールにタグを使用する方法と、リソースベースのポリシーとアイデンティティベースのポリシーの違いの詳細については、「IAM ユーザーガイド」の「ABAC 認可で属性に基づいてアクセス許可を定義する」を参照してください。

IP ベースのポリシー

IP ベースのポリシーは、1 つ以上の IP アドレスあるいは CIDR ブロックにドメインへのアクセスを制限します。技術的には、IP ベースのポリシーは異なるタイプのポリシーではありません。これらのポリシーは、匿名のプリンシパルを指定し、特別な Condition を含む、リソースベースのポリシーです。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素 Condition」を参照してください。

IP ベースのポリシーの最大の特徴は、OpenSearch Service ドメインへの署名なしのリクエストを許可することです。これにより、curl や OpenSearch Dashboards などのクライアントを使用したり、プロキシサーバーを介してドメインにアクセスしたりすることができます。詳細については the

section called “プロキシを使用して Dashboards から OpenSearch Service にアクセスする”を参照してください。

> **Note**
>
> ドメインで VPC アクセスを有効にすると、IP ベースのポリシーを設定することはできません。代わりに、どの IP アドレスがドメインにアクセスできるかを制御する `security groups` を使用できます。詳細については、以下の各トピックを参照してください。
>
> - the section called “VPC ドメインのアクセスポリシーについて”
> - 「Amazon VPC ユーザーガイド」の「セキュリティグループを使用して AWS リソースへのトラフィックを制御する」

以下のポリシーは、指定された IP 範囲からのすべての HTTP リクエストが `test-domain` にアクセスする許可を付与します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24"
          ]
        }
      },
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    }
  ]
```

```
}
```

ドメインにパブリックエンドポイントがあり、きめ細かなアクセスコントロールを使用しない場合は、IAM プリンシパルと IP アドレスを組み合わせることをお勧めします。このポリシーでは、指定した IP 範囲からのリクエストの場合にのみ、`test-user` HTTP アクセスを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::987654321098:user/test-user"
      ]
    },
    "Action": [
      "es:ESHttp*"
    ],
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24"
        ]
      }
    },
    "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
  }]
}
```

OpenSearch Service リクエストの作成と署名

完全にオープンなリソースベースのポリシーを設定する場合、OpenSearch Service 設定 API へのすべてのリクエストは署名付きである必要があります。ポリシーで IAM ロールまたはユーザーが指定されている場合、OpenSearch APIs へのリクエストも AWS 署名バージョン 4 を使用して署名する必要があります。署名メソッドは API によって異なります。

- OpenSearch Service 設定 API 呼び出しを行うには、AWS SDK のいずれかを使用することが推奨されます。SDK を使用するほうが、独自のリクエストを作成し署名するよりも、プロセスが簡素化し、大幅な時間の節約ができます。API エンドポイント設定には次の形式を使用します。

```
es.region.amazonaws.com/2021-01-01/
```

例えば、次のリクエストは、movies ドメインの設定を変更しますが、自分で署名する必要があります (お勧めしません)。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/movies/config
{
  "ClusterConfig": {
    "InstanceType": "c5.xlarge.search"
  }
}
```

Boto 3 などのいずれかの SDK を使用する場合、SDK はリクエスト署名を自動的に処理します。

```
import boto3

client = boto3.client(es)
response = client.update_domain_config(
  DomainName='movies',
  ClusterConfig={
    'InstanceType': 'c5.xlarge.search'
  }
)
```

Java コードの例については、「the section called “ AWS SDKsの使用”」を参照してください。

- OpenSearch API への呼び出しを行うには、独自リクエストに署名する必要があります。OpenSearch API は、次の形式を使用します。

```
domain-id.region.es.amazonaws.com
```

例えば、次のリクエストは、thor の movies インデックスを検索します。

```
GET https://my-domain.us-east-1.es.amazonaws.com/movies/_search?q=thor
```

> **Note**
>
> このサービスは、署名バージョン 4 で署名された HTTP POST リクエストの URL で渡されるパラメータを無視します。

複数のポリシーが衝突する場合

複数のポリシーが同意しない、あるいはユーザーを明示的に指定しない場合、困難が生じます。「IAM ユーザーガイド」の「IAM の仕組み」では、ポリシーの評価ロジックの簡略な概要を説明しています。

- デフォルトでは、すべてのリクエストが拒否されます。
- 明示的な許可はこのデフォルトに優先します。
- 明示的な拒否はすべての許可に上書きされます。

例えば、リソースベースのポリシーがドメインサブリソース (OpenSearch インデックスまたは API) にアクセスを許可しているが、アイデンティティベースのポリシーはアクセスを拒否するため、アクセスは拒否されます。アイデンティティベースのポリシーとリソースベースのポリシーによってユーザーがアクセス許可を持つべきかを指定いない場合、アクセスは許可されます。ドメインサブリソースのすべての結果の概要における交差するポリシーの図を以下で参照してください。

	リソースベースのポリシーで許可	リソースベースのポリシーで拒否	リソースベースのポリシーで許可あるいは拒否されない
Allowed in identity-based policy	許可	Deny	Allow
Denied in identity-based policy	Deny	Deny	Deny
Neither allowed nor denied in identity-based policy	Allow	Deny	Deny

ポリシーエレメントのリファレンス

OpenSearch Service では、「IAM ポリシーエレメントのリファレンス」のほとんどのポリシー要素がサポートされています (NotPrincipal は除く)。次の表は、最も一般的なエレメントを示しています。

JSON ポリシーエレメント	概要
Version	ポリシー言語の最新バージョンは 2012-10-17 です。すべてのアクセスポリシーでこの値を指定する必要があります。
Effect	このエレメントは、指定されたアクションへのアクセスをステートメントが許可するか拒否するかを指定します。有効な値は Allow または Deny です。
Principal	この要素は、リソースへのアクセスを許可または拒否し、いくつかの形式を取ることができる AWS アカウント または IAM ロールまたはユーザーを指定します。 • AWS アカウント: "Principal":{"AWS": ["123456789012"]} または "Principal":{"AWS": ["arn:aws:iam::123456789012:root"]} • IAM ユーザー: "Principal":{"AWS": ["arn:aws:iam::123456789012:user/test-user"]} • IAM ロール: "Principal":{"AWS": ["arn:aws:iam::123456789012:role/test-role"]} **Important** * ワイルドカードを指定すると、ドメインへの匿名アクセスが有効になります。これは IP ベースの条件を追加する、VPC サポートを使用する、またはきめ細かなアクセスコントロールを有効にしない限り、お勧めしません。さらに、次のポリシーを注意深く調べて、広範なアクセスが許可されていないことを確かめてください。

JSON ポリシーエレメント	概要
	• 関連する AWS プリンシパル (IAM ロールなど) にアタッチされているアイデンティティベースのポリシー • 関連付けられた AWS リソースにアタッチされたリソースベースのポリシー (AWS Key Management Service KMS キーなど)

JSON ポリシーエレメント	概要
`Action`	OpenSearch Service は、OpenSearch HTTP メソッドに対して `ESHttp*` アクションを使用します。残りのアクションは設定 API に適用されます。 特定の `es:` アクションは、リソースレベルの許可をサポートします。例えば、すべてのドメインを削除する許可を付与せずに、1 つの特定のドメインのみ削除する許可をユーザーに付与することができます。その他のアクションはサービス自体に適用されます。例えば、`es:ListDomainNames` は単一ドメインのコンテキストでは意味を成しませんが、それでもワイルドカードを必要とします。 使用可能なすべてのアクションのリストと、それらがドメインのサブリソース (`test-domain/*`) に適用されるのか、ドメイン設定 (`test-domain`) に適用されるのか、サービス (*) のみに適用されるのかについては、「サービス認証リファレンス」の「Actions, resources, and condition keys for Amazon OpenSearch Service」を参照してください リソースベースのポリシーは、リソースレベルのアクセス権限とは異なっています。リソースベースのポリシー は、ドメインにアタッチする完全な JSON ポリシーです。リソースレベルのアクセス許可は、特定のドメインあるいはサブリソースにアクションを制限します。実際には、リソースレベルのアクセス許可はリソース、あるいはアイデンティティベースのポリシーのオプションの一部として捉えることができます。 `es:CreateDomain` へのリソースレベルのアクセス権限は直観的ではないように見えることがあります。(つまり、すでに存在するドメインを作成するアクセス権限をなぜユーザーに付与するのでしょうか？) ワイルドカードの使用でドメインに簡単な命名スキームを適用できます (`"Resource": "arn:aws:es:us-west-1:987654321098:domain/my-team-name-*"` など)。 もちろん、次のように、制限がより緩和されたリソース要素のアクションを含めることもできます。

JSON ポリシーエレメント	概要
	JSON アクションとリソースのペアリングについての詳細は、テーブルの Resource 要素を参照してください。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttpGet",
        "es:DescribeDomain"
      ],
      "Resource": "*"
    }
  ]
}
```

JSON ポリシーエレメント	概要
`Condition`	OpenSearch Service は、IAM ユーザーガイドの「AWS グローバル条件コンテキストキー」に記載されているほとんどの条件をサポートします。注目すべき例外は、OpenSearch Service がサポートしていない `aws:PrincipalTag` キーがあることです。 IP ベースのポリシーを設定する場合、次に示すように、IP アドレスまたは CIDR ブロックを条件として指定します。 `"Condition": { "IpAddress": { "aws:SourceIp": [ "192.0.2.0/32" ] } }` the section called “アイデンティティベースのポリシー” に記載されているように、`aws:ResourceTag` 、`aws:RequestTag` 、および `aws:TagKeys` 条件キーは、設定 API と OpenSearch API に適用されます。

JSON ポリシーエレメント	概要

`Resource`	OpenSearch サービスは、3 つの基本的な方法で `Resource` エレメントを使用します。
	
	• `es:ListDomainNames` やフルアクセスを許可するといった、OpenSearch Service 自体に適用されるアクションには、次の構文を使用します。
	
	`"Resource": "*"`
	
	• `es:DescribeDomain` のように、ドメインの設定に関連するアクションには、次の構文を使用します。
	
	`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name "`
	
	• `es:ESHttpGet` のように、ドメインのサブリソースを適用するアクションには、次の構文を使用します。
	
	`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name /*"`
	
	ワイルドカードを使用する必要はありません。OpenSearch Service は、各 OpenSearch インデックスまたは API に異なるアクセスポリシーを定義します。例えば、`test-index` インデックスへのユーザーのアクセス許可を制限できます。
	
	`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name /test-index"`
	
	`test-index` へのフルアクセスの代わりに、検索 API のみにポリシーを制限することもできます。
	
	`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name /test-index/_search"`

JSON ポリシーエレメント	概要
	個々のドキュメントへのアクセスを制御することもできます。 `"Resource": "arn:aws:es:  region:aws-account-id:domain/domain-name /test-index/test-type/1"` 基本的に、OpenSearch によってサブリソースが URI として表現される場合、アクセスポリシーを使用してサブリソースへのアクセスをコントロールできます。どのリソースにユーザーがアクセスできるかをさらに細かく制御するには、「the section called “きめ細かなアクセスコントロール”」を参照してください。 どのアクションがリソースレベルのアクセス権限をサポートするかの詳細については、このテーブルの Action 要素を参照してください。

詳細オプションと API に関する考慮事項

OpenSearch Service にはいくつかの詳細オプションがあり、その 1 つにはアクセスコントロールが関連します (`rest.action.multi.allow_explicit_index`)。デフォルト設定の true では、特定の状況下でユーザーがサブリソースへのアクセス権限を回避することが許可されます。

例えば、次のリソースベースのポリシーを考えます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
```

```
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-west-1:987654321098:domain/test-domain/test-index/*",
        "arn:aws:es:us-west-1:987654321098:domain/test-domain/_bulk"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:ESHttpGet"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/
restricted-index/*"
    }
  ]
}
```

このポリシーは、test-user に test-index および OpenSearch バルク API へのフルアクセスを付与します。また、GET への restricted-index リクエストを許可します。

次のインデックスリクエストは、見てわかるように、アクセス権限エラーによって失敗します。

```
PUT https://search-test-domain.us-west-1.es.amazonaws.com/restricted-index/movie/1
{
  "title": "Your Name",
  "director": "Makoto Shinkai",
  "year": "2016"
}
```

インデックス API とは異なり、バルク API では、単一の呼び出しで多くのドキュメントの作成、更新、削除を実行できます。多くの場合、これらのオペレーションはリクエスト URL ではなく、リクエストの本文で指定します。OpenSearch Service が URL を使用してドメインサブリソースへのアクセスを制御するため、test-user は、結果的にバルク API を使用して restricted-index に変

更を加えることができます。ユーザーにはインデックスで POST アクセス権限が欠如していますが、次のリクエストは成功します。

```
POST https://search-test-domain.us-west-1.es.amazonaws.com/_bulk
{ "index" : { "_index": "restricted-index", "_type" : "movie", "_id" : "1" } }
{ "title": "Your Name", "director": "Makoto Shinkai", "year": "2016" }
```

このような状況では、アクセスポリシーはその目的達成に失敗します。ユーザーがこれらの制限を回避することを防ぐためには、rest.action.multi.allow_explicit_index を false に変更できます。この値が false の場合、リクエストボディでインデックス名を指定するバルク、mget、および msearch API へのすべての呼び出し動作は停止します。つまり、_bulk への呼び出しは機能しなくなりますが、test-index/_bulk への呼び出しは動作します。この 2 番目のエンドポイントにはインデックス名が含まれるため、リクエストボディにそれを指定する必要はありません。

OpenSearch Dashboards は mget および msearch に大きく依存しています。そのため、この変更後には正常に動作しないことが予想されます。部分的な解決策としては、rest.action.multi.allow_explicit_index を true のまま残して、特定のユーザーが 1 つ以上の API にアクセスすることを拒否します。

この設定の変更については、「the section called “高度なクラスター設定”」を参照してください。

同様に、以下のリソースベースのポリシーには 2 つの小さな問題があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test-user"
      },
      "Action": "es:ESHttp*",
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    },
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test-user"
      },
```

```
      "Action": "es:ESHttp*",
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/
restricted-index/*"
    }
  ]
}
```

- 明示的な拒否にも関わらず、test-user が GET https://search-test-domain.us-west-1.es.amazonaws.com/_all/_search や GET https://search-test-domain.us-west-1.es.amazonaws.com/*/_search などの呼び出しを実行して、restricted-index のドキュメントにアクセスできることです。
- Resource エレメントレファレンス restricted-index/* により、test-user はインデックスのドキュメントに直接アクセスする権限がありません。しかし、ユーザーにはインデックス全体を削除する権限があります。アクセスと削除を防ぐには、このポリシーで restricted-index* を指定する必要があります。

広範囲な許可と一部の拒否を混合するよりは、最小限の権限の原則を守り、タスクを実行するために必要なアクセス権限のみを付与することが最も安全なアプローチです。個々のインデックスまたは OpenSearch オペレーションに対するアクセスコントロールの詳細については、the section called “きめ細かなアクセスコントロール” を参照してください。

> **Important**
>
> * ワイルドカードを指定すると、ドメインへの匿名アクセスが有効になります。ワイルドカードを使用することはお勧めしません。さらに、以下のポリシーを注意深く調べて、広範なアクセスが許可されていないことを確かめてください。
>
> - 関連付けられた AWS プリンシパルにアタッチされたアイデンティティベースのポリシー (IAM ロールなど)
> - 関連付けられたリソースにアタッチされた AWS リソースベースのポリシー (AWS Key Management Service KMS キーなど)

アクセスポリシーの設定

- リソースの作成あるいは変更、そして OpenSearch Service の IP ベースのポリシーに関する説明は、「the section called “アクセスポリシーの設定”」を参照してください。

- IAM でアイデンティティベースのポリシーを作成または変更する方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

追加のサンプルポリシー

この章には多くのサンプルポリシーが含まれていますが、 AWS アクセスコントロールは複雑なテーマであり、例を通じて最もよく理解できます。詳細については、IAM ユーザーガイドの「IAM アイデンティティベースのポリシーの例」を参照してください。

Amazon OpenSearch Service API のアクセス許可リファレンス

アクセスコントロールをセットアップするときに、IAM ID (アイデンティティベースのポリシー) にアタッチできるアクセス許可ポリシーを記述します。詳細なリファレンスについては「サービス認可リファレンスガイド」で以下のトピックを参照してください。

- OpenSearch Service のアクション、リソース、条件キー
- OpenSearch Ingestion のアクション、リソース、条件キー

このリファレンスには、IAM ポリシーで使用できる API オペレーションに関する情報が含まれています。また、アクセス許可を付与できる AWS リソースと、きめ細かなアクセスコントロールのために含めることができる条件キーも含まれます。

ポリシーの `Action` フィールドにアクションを、ポリシーの `Resource` フィールドにリソース値を、ポリシーの `Condition` フィールドに条件を指定します。OpenSearch Service のアクションを指定するには、API オペレーション名 (`es:CreateDomain` など) の前に `es:` プレフィックスを使用します。OpenSearch Ingestion のアクションを指定するには、API オペレーション (`osis:CreatePipeline` など) の前に `osis:` プレフィックスを使用します。

AWS Amazon OpenSearch Service の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS マネージドポリシーを更新する可能性が高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AmazonOpenSearchDirectQueryGlueCreateAccess

Amazon OpenSearch Service Direct Query Service
に、`CreateDatabase`、`CreatePartitionCreateTable`、および へのアクセスを許可しますBatchCreatePartition AWS Glue API。

AmazonOpenSearchDirectQueryGlueCreateAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchServiceFullAccess

AWS アカウントのために OpenSearch Service 設定 API オペレーションとリソースへのフルアクセス権を付与します。

AmazonOpenSearchServiceFullAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchServiceReadOnlyAccess

のすべての OpenSearch Service リソースへの読み取り専用アクセスを許可します AWS アカウント。

AmazonOpenSearchServiceReadOnlyAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchServiceRolePolicy

IAM エンティティに `AmazonOpenSearchServiceRolePolicy` をアタッチすることはできません。このポリシーは、OpenSearch Service がアカウントリソースにアクセスすることを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「the section called “アクセス許可”」を参照してください。

AmazonOpenSearchServiceRolePolicy ポリシーは、IAM コンソールにあります。

AmazonOpenSearchServiceCognitoAccess

Cognito 認証を有効にするために必要な最小限の Amazon Cognito の許可を提供します。

AmazonOpenSearchServiceCognitoAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchIngestionServiceRolePolicy

IAM エンティティに AmazonOpenSearchIngestionServiceRolePolicy をアタッチすることはできません。このポリシーは、OpenSearch Ingestion が、インジェストパイプラインへの VPC アクセスを有効にし、タグを作成し、インジェスト関連の CloudWatch メトリクスをアカウントに公開することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「the section called “サービスにリンクされたロールの使用”」を参照してください。

AmazonOpenSearchServerlessServiceRolePolicy ポリシーは、IAM コンソールにあります。

OpenSearchIngestionSelfManagedVpcePolicy

IAM エンティティに OpenSearchIngestionSelfManagedVpcePolicy をアタッチすることはできません。このポリシーは、OpenSearch Ingestion が、インジェストパイプラインへの セルフマネージド VPC アクセスを有効にし、タグを作成し、インジェスト関連の CloudWatch メトリクスをアカウントに公開することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「the section called “サービスにリンクされたロールの使用”」を参照してください。

OpenSearchIngestionSelfManagedVpcePolicy ポリシーは IAM コンソールにあります。

AmazonOpenSearchIngestionFullAccess

AWS アカウントのために、OpenSearch Ingestion API オペレーションとリソースへの完全なアクセスを許可します。

AmazonOpenSearchIngestionFullAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchIngestionReadOnlyAccess

AWS アカウントのためにすべての OpenSearch Ingestion リソースへの読み取り専用アクセスを許可します。

AmazonOpenSearchIngestionReadOnlyAccess ポリシーは、IAM コンソールにあります。

AmazonOpenSearchServerlessServiceRolePolicy

OpenSearch Serverless メトリクスデータを CloudWatch に送信するために必要な最小限の Amazon CloudWatch アクセス許可を提供します。

AmazonOpenSearchServerlessServiceRolePolicy ポリシーは、IAM コンソールにあります。

AWS 管理ポリシーに対する OpenSearch Service の更新

このサービスが変更の追跡を開始してからの OpenSearch Service の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
`AmazonOpenSearchIngestionServiceRolePolicy` を更新しました	この更新により、OpenSearch Ingestion に対して OpenSearch が作成した VPC エンドポイントを変更するアクセス許可が付与され、VPC 間でパイプラインを共有することができるようになります。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2025 年 8 月 28 日
`AmazonOpenSearchServiceRolePolicy` を更新しました	以下のステートメントをポリシーに追加しました。Amazon OpenSearch Service が `AWSServiceRoleForAmazonOpenSearchService` サービスにリンクされたロールを引き受けると、ポリシー内のこの新しいステートメントにより、OpenSearch は OpenSearch のみが管理する AWS IAM アイデンティティセンター アプリケーションのアクセス範囲を更新できます。 `{ "Effect": "Allow",`	2025 年 3 月 31 日

変更	説明	日付
	"Action": "sso:PutApplicationAccessScope", "Resource": "arn:aws:sso::*:application/*/*", "Condition": { "StringEquals": { "aws:ResourceOrgID": "${aws:PrincipalOrgID}" } } }	
AmazonOpenSearchServerlessServiceRolePolicy を更新	ポリシー AmazonOpenSearchServerlessServiceRolePolicy に SID AllowAOSSCloudwatchMetrics を追加しました。SID は、ポリシーステートメントのオプションの識別子として機能するステートメント ID です。	2024 年 7 月 12 日

変更	説明	日付
「OpenSearchIngestionSelfManagedVpcePolicy 」を追加	OpenSearch Ingestion が、インジェストパイプラインへのセルフマネージド VPC アクセスを有効にし、タグを作成し、インジェスト関連の CloudWatch メトリクスをアカウントに公開することを許可する、新しいポリシーです。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2024 年 6 月 12 日
AmazonOpenSearchDirectQueryGlueCreateAccess を追加	Amazon OpenSearch Service Direct Query Service に、CreateDatabase 、CreatePartition CreateTable 、および へのアクセスを許可しますBatchCreatePartition AWS Glue API。	2024 年 5 月 6 日
「AmazonOpenSearchServiceRolePolicy 」と「AmazonElasticsearchServiceRolePolicy 」の更新	サービスにリンクされたロールが IPv6 アドレスの割り当てと割り当て解除を行うために必要なアクセス許可が追加されました。 廃止された Elasticsearch ポリシー も下位互換性を確保するために更新されました。	2023 年 10 月 18 日

変更	説明	日付
「AmazonOpenSearchIngestionServiceRolePolicy 」を追加	OpenSearch Ingestion が、インジェストパイプラインへの VPC アクセスを有効にし、タグを作成し、インジェスト関連の CloudWatch メトリクスをアカウントに公開することを許可する、新しいポリシーです。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2023 年 4 月 26 日
「AmazonOpenSearchIngestionFullAccess 」を追加	AWS アカウントのために、OpenSearch Ingestion API オペレーションとリソースへの完全なアクセスを許可する新しいポリシーです。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2023 年 4 月 26 日
「AmazonOpenSearchIngestionReadOnlyAccess 」を追加	AWS アカウントのためにすべての OpenSearch Ingestion リソースへの読み取り専用アクセスを許可する新しいポリシーです。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2023 年 4 月 26 日

変更	説明	日付
「AmazonOpenSearchServerlessServiceRolePolicy 」を追加	OpenSearch Serverless のメトリクスデータを Amazon CloudWatchに送信するために必要な最小限のアクセス許可を提供する新しいポリシーです。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2022 年 11 月 29 日
「AmazonOpenSearchServiceRolePolicy 」と「AmazonElasticsearchServiceRolePolicy 」の更新	サービスにリンクされたロールが OpenSearch Service で管理される VPC エンドポイントを作成するために必要なアクセス許可を追加しました。アクションには、リクエストにタグ OpenSearchManaged=true が含まれていないと実行できないものがあります。 廃止された Elasticsearch ポリシー も下位互換性を確保するために更新されました。	2022 年 11 月 7 日

変更	説明	日付
「AmazonOpenSearchServiceRolePolicy 」と「AmazonElasticsearchServiceRolePolicy 」の更新	OpenSearch クラスターメトリクスを Amazon CloudWatch に発行するために必要な PutMetricData アクションのサポートが追加されました。 廃止された Elasticsearch ポリシー も下位互換性を確保するために更新されました。 ポリシーの JSON については、「IAM コンソール」を参照してください。	2022 年 9 月 12 日
「AmazonOpenSearchServiceRolePolicy 」と「AmazonElasticsearchServiceRolePolicy 」の更新	acm リソースタイプのサポートを追加しました。このポリシーは、カスタムエンドポイントが有効なドメインを作成および更新するために、サービスにリンクされたロールが ACM リソースを検証および検証するために必要な最小限の AWS Certificate Manager (ACM) 読み取り専用アクセス許可を提供します。 非推奨の Elasticsearch ポリシーも、下位互換性を確保するために更新されました。	2022 年 7 月 28 日

変更	説明	日付
「AmazonOpenSearchServiceCognitoAccess 」および「AmazonESCognitoAccess 」を更新しました。	UpdateUserPoolClient アクション用のサポートを追加します。Elasticsearch から OpenSearch へのアップグレード時に、Cognito user pool を設定するために必要です。 SetIdentityPoolRoles すべてのリソースへのアクセスを許可するアクションの権限を修正しました。 非推奨の Elasticsearch ポリシーも、下位互換性を確保するために更新されました。	2021 年 12 月 20 日
AmazonOpenSearchServiceRolePolicy を更新	security-group リソースタイプのサポートを追加しました。このポリシーは、VPC アクセスを有効にするために、サービスにリンクされたロールで必要な最小限の Amazon EC2 および Elastic Load Balancing の許可を提供します。	2021 年 9 月 9 日

変更	説明	日付
• 「AmazonOpenSearchServiceFullAccess 」を追加 • 「AmazonESFullAccess 」を廃止	この新しいポリシーは、古いポリシーを置き換えるためのものです。どちらのポリシーも、OpenSearch Service 設定 API および OpenSearch API のすべての HTTP メソッドへのフルアクセスを提供します。きめ細かなアクセスコントロールおよびリソースベースのポリシーは引き続きアクセスを制限できます。	2021 年 9 月 7 日
• 「AmazonOpenSearchServiceReadOnlyAccess 」を追加 • 「AmazonESReadOnlyAccess 」を廃止	この新しいポリシーは、古いポリシーを置き換えるためのものです。どちらのポリシーも、OpenSearch Service 設定 API (es:Describe* 、es:List*、および es:Get*) への読み取り専用アクセスを許可しますが、OpenSearch API の HTTP メソッドへのアクセスは許可しません。	2021 年 9 月 7 日
• 「AmazonOpenSearchServiceCognitoAccess 」を追加 • 「AmazonESCognitoAccess 」を廃止	この新しいポリシーは、古いポリシーを置き換えるためのものです。どちらのポリシーも、Cognito 認証を有効にするために必要な最小限の Amazon Cognito の許可を提供します。	2021 年 9 月 7 日

変更	説明	日付
• 「AmazonOpenSearchServiceRolePolicy」を追加 • 「AmazonElasticsearchServiceRolePolicy」を廃止	この新しいポリシーは、古いポリシーを置き換えるためのものです。どちらのポリシーも、VPC アクセスを有効にするために、サービスにリンクされたロールで必要な最小限の Amazon EC2 および Elastic Load Balancing の許可を提供します。	2021 年 9 月 7 日
変更の追跡を開始しました	Amazon OpenSearch Service は、 AWS管理ポリシーの変更を追跡するようになりました。	2021 年 9 月 7 日

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、Amazon OpenSearch Service が別のサービスに付与する、リソースへのアクセス許可を制限することをお勧めします。aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID にaws:SourceArn の値が含まれていない場合、aws:SourceAccount 値と aws:SourceArn 値の中のアカウントには、同じアカウント ID

を使用する必要があります。クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、aws:SourceAccount を使用します。

aws:SourceArn の値は OpenSearch Service ドメインの ARN でなければなりません。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合は、aws:SourceArn グローバルコンテキスト条件キーを使用して、ARN の未知部分をワイルドカード (*) で表します。例えば、arn:aws:*es*:*:*123456789012*:* です。

次の例では、OpenSearch Service で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Sid":"ConfusedDeputyPreventionExamplePolicy",
      "Effect":"Allow",
      "Principal":{
         "Service":"es.amazonaws.com"
      },
      "Action":"sts:AssumeRole",
      "Condition":{
         "StringEquals":{
            "aws:SourceAccount":"111122223333"
         },
         "ArnLike":{
            "aws:SourceArn":"arn:aws:es:us-east-1:111122223333:domain/my-domain"
         }
      }
   }
}
```

Amazon OpenSearch Service のきめ細かなアクセスコントロール

きめ細かなアクセスコントロールは、Amazon OpenSearch Service のデータへのアクセスをコントロールする追加の方法を提供します。例えば、リクエスト者によっては、検索で 1 つのインデックスのみから結果が返されるようにしたい場合があります。ドキュメント内の特定のフィールドを非表示にしたい場合や、特定のドキュメントをまとめて除外したい場合もあります。

きめ細かなアクセスコントロールには、以下の利点があります。

- ロールベースアクセスコントロール
- インデックスレベル、ドキュメントレベル、フィールドレベルのセキュリティ
- OpenSearch Dashboards マルチテナンシー
- OpenSearch および OpenSearch Dashboards の HTTP 基本認証

トピック

- 全体像: きめ細かなアクセスコントロールと OpenSearch Service セキュリティ
- 主要なコンセプト
- マスターユーザーについて
- きめ細かなアクセスコントロールの有効化
- マスターユーザーとして OpenSearch Dashboards にアクセスする
- 許可の管理
- 推奨される設定
- 制限事項
- マスターユーザーの変更
- 追加のマスターユーザー
- 手動スナップショット
- 統合
- REST API の相違点
- チュートリアル: IAM マスターユーザーと Amazon Cognito 認証を使用してドメインを設定する
- チュートリアル: 内部ユーザーデータベースと HTTP 基本認証でドメインを設定する

全体像: きめ細かなアクセスコントロールと OpenSearch Service セキュリティ

Amazon OpenSearch Service のセキュリティには 3 つの主要なレイヤーがあります。

ネットワーク

最初のセキュリティレイヤーはネットワークであり、リクエストが OpenSearch Service ドメインに到達するかどうかを決定します。ドメインを作成するときに [パブリックアクセス] を選択した場合、インターネットに接続されたクライアントからのリクエストがドメインエンドポイントに到達できます。[VPC アクセス] を選択した場合、クライアントがエンドポイントに到達するためには、クライアントが VPC に接続する必要があります (かつ、関連するセキュリティグループがその接続を許可する必要があります)。詳細については、「the section called “VPC サポート”」を参照してください。

ドメインアクセスポリシー

2 番目のセキュリティレイヤーは、ドメインアクセスポリシーです。リクエストがドメインエンドポイントに到達すると、リソースベースのアクセスポリシーにより、指定された URI へのアクセスリクエストが許可または拒否されます。このアクセスポリシーにより、リクエストは OpenSearch 自体に到達する前に、ドメインの「エッジ」で許可または拒否されます。

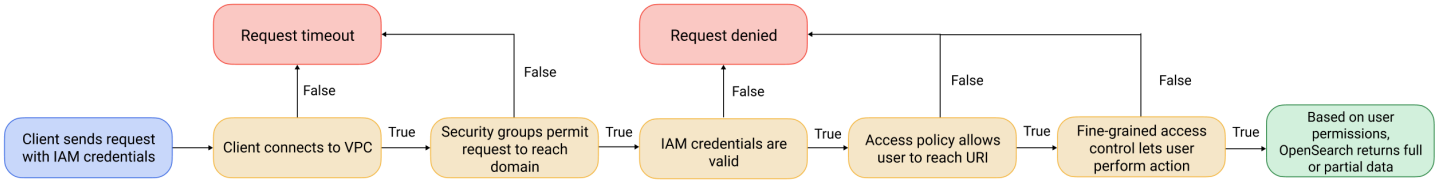
きめ細かなアクセスコントロール

最後の 3 番目のセキュリティレイヤーは、きめ細かなアクセスコントロールです。リソースベースのアクセスポリシーにより、リクエストがドメインエンドポイントに到達することが許可された後、きめ細かなアクセスコントロールにより、ユーザー認証情報が評価されて、ユーザーが認証されるか、リクエストが拒否されます。きめ細かなアクセスコントロールによりユーザーが認証された場合、そのユーザーにマッピングされているすべてのロールが取得され、付与されるすべての許可を使用してリクエストの処理方法が決定されます。

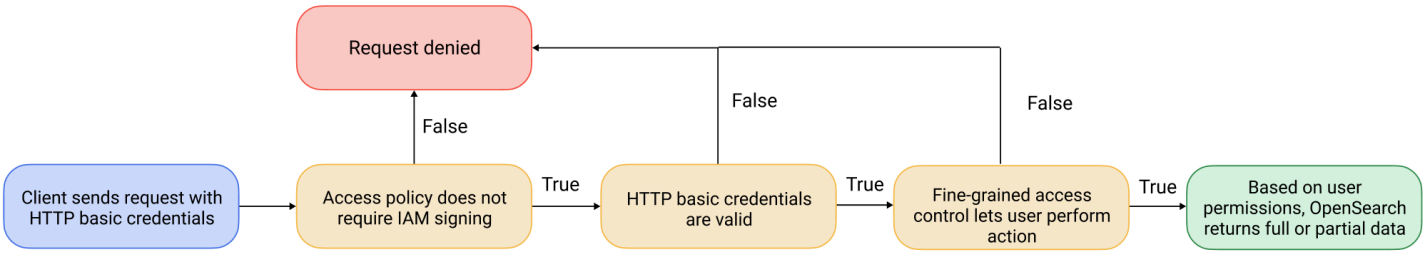
> **Note**
>
> リソースベースのアクセスポリシーに IAM ロールまたはユーザーが含まれている場合、クライアントは AWS 署名バージョン 4 を使用して署名付きリクエストを送信する必要があります。そのため、アクセスポリシーは、特に内部ユーザーデータベースと HTTP 基本認証を使用する場合、きめ細かなアクセスコントロールと競合することがあります。ユーザー名とパ

スワードで、かつ IAM 認証情報で、リクエストに署名することはできません。一般に、きめ細かなアクセスコントロールを有効にする場合は、署名付きリクエストを必須としないドメインアクセスポリシーを使用することをお勧めします。

以下の図表は、きめ細かなアクセスコントロールが有効な VPC アクセスドメイン、IAM ベースのアクセスポリシー、IAM マスターユーザーという、一般的な構成を示しています。



以下の図表は、きめ細かなアクセスコントロールが有効なパブリックアクセスドメイン、IAM プリンシパルを使用しないアクセスポリシー、内部ユーザーデータベース内のマスターユーザーという、別の一般的な構成を示しています。



例

`movies/_search?q=thor` への GET リクエストを考えてみます。ユーザーに `movies` インデックスを検索する許可があることを確認します。そうであれば、ユーザーにその内部のすべてのドキュメントを表示するアクセス許可があることを確認します。レスポンスで、フィールドを省略または匿名化するかを選択します。マスターユーザーの場合、レスポンスは以下のようになります。

```
{
  "hits": {
    "total": 7,
    "max_score": 8.772789,
    "hits": [{
        "_index": "movies",
        "_type": "_doc",
        "_id": "tt0800369",
        "_score": 8.772789,
        "_source": {
```

```
          "directors": [
            "Kenneth Branagh",
            "Joss Whedon"
          ],
          "release_date": "2011-04-21T00:00:00Z",
          "genres": [
            "Action",
            "Adventure",
            "Fantasy"
          ],
          "plot": "The powerful but arrogant god Thor is cast out of Asgard to
 live amongst humans in Midgard (Earth), where he soon becomes one of their finest
 defenders.",
          "title": "Thor",
          "actors": [
            "Chris Hemsworth",
            "Anthony Hopkins",
            "Natalie Portman"
          ],
          "year": 2011
        }
      },
      ...
    ]
  }
}
```

許可がより制限されたユーザーが上記と同じリクエストを発行した場合、レスポンスは以下のようになります。

```
{
  "hits": {
    "total": 2,
    "max_score": 8.772789,
    "hits": [{
        "_index": "movies",
        "_type": "_doc",
        "_id": "tt0800369",
        "_score": 8.772789,
        "_source": {
          "year": 2011,
          "release_date":
 "3812a72c6dd23eef3c750c2d99e205cbd260389461e19d610406847397ecb357",
```

```
          "plot": "The powerful but arrogant god Thor is cast out of Asgard to
 live amongst humans in Midgard (Earth), where he soon becomes one of their finest
 defenders.",
          "title": "Thor"
        }
      },
      ...
    ]
  }
}
```

レスポンスでは、ヒット数、およびヒットあたりのフィールド数が少なくなっています。また、release_date フィールドは匿名化されています。許可のないユーザーが上記と同じリクエストを発行した場合、クラスターによってエラーが返されます。

```
{
  "error": {
    "root_cause": [{
      "type": "security_exception",
      "reason": "no permissions for [indices:data/read/search] and User [name=limited-
user, roles=[], requestedTenant=null]"
    }],
    "type": "security_exception",
    "reason": "no permissions for [indices:data/read/search] and User [name=limited-
user, roles=[], requestedTenant=null]"
  },
  "status": 403
}
```

ユーザーが無効な認証情報を提供した場合、クラスターによって Unauthorized 例外が返されます。

主要なコンセプト

きめ細かなアクセスコントロールを開始するときは、次の概念を考慮してください。

- [ロール] – きめ細かなアクセスコントロールの主要な使用方法。この場合、ロールは IAM ロールとは異なります。ロールには、クラスター全体、インデックス固有、ドキュメントレベル、フィールドレベルの許可の任意の組み合わせが含まれます。
- [マッピング] – ロールを設定したら、1 人以上のユーザーにマッピングします。例えば、3 つのロールを 1 人のユーザーにマッピングできます。1 つは Dashboards へのアクセスを許可し、1 つ

は index1 への読み取り専用アクセスを許可し、もう 1 つは index2 への書き込みアクセスを許可します。または、これらのすべての許可を 1 つのロールに含めることもできます。

- [ユーザー] – OpenSearch クラスターに対してリクエストを行うユーザーまたはアプリケーション。ユーザーは、リクエストを行うときに指定する認証情報 (IAM アクセスキーまたはユーザー名とパスワードのいずれか) を持っています。

マスターユーザーについて

OpenSearch Service のマスターユーザーは、ユーザー名とパスワードの組み合わせ、または基盤となる OpenSearch クラスターへの完全なアクセス許可を持つ IAM プリンシパルのいずれかです。OpenSearch Dashboards 内で内部ユーザー、ロール、ロールマッピングを作成する機能に加えて、OpenSearch クラスターへのすべてのアクセス権を持っている場合、ユーザーはマスターユーザーと見なされます。

OpenSearch Service コンソールまたは CLI を使用して作成されたマスターユーザーは、次の 2 つの定義済みロールに自動的にマッピングされます。

- all_access – すべてのクラスター全体のオペレーションへのフルアクセス、すべてのクラスターインデックスへの書き込み許可、およびすべてのテナントへの書き込み許可を提供します。
- security_manager – セキュリティプラグインへのアクセスと、ユーザーとアクセス許可の管理を提供します。

これらの 2 つのロールを使用すると、ユーザーは OpenSearch Dashboards の [セキュリティ] タブにアクセスして、ユーザーとアクセス許可を管理できます。別の内部ユーザーを作成し、all_access ロールにのみマッピングすると、そのユーザーは [セキュリティ] タブにアクセスできなくなります。追加のマスターユーザーは、all_access と security_manager ロールの両方に明示的にマッピングすることで作成できます。手順については、「the section called “追加のマスターユーザー”」を参照してください。

ドメインのマスターユーザーを作成するときは、既存の IAM プリンシパルを指定するか、内部ユーザーデータベース内にマスターユーザーを作成できます。どちらを使用するかを決定するときは、次の点を考慮してください。

- IAM プリンシパル – マスターユーザーの IAM プリンシパルを選択した場合、クラスターへのすべてのリクエストは AWS 署名バージョン 4 を使用して署名する必要があります。

OpenSearch Service では、IAM プリンシパルのアクセス許可は考慮されません。IAM ユーザーまたはロールは、認証のためにのみ機能します。そのユーザーまたはロールのポリシーは、マスターユーザーの許可には関係しません。許可は、OpenSearch セキュリティプラグインのさまざまなアクセス許可を通じて処理されます。

例えば、IAM プリンシパルに IAM アクセス許可をゼロに割り当てることができます。マシンまたはユーザーがそのユーザーまたはロールに認証できる限り、そのプリンシパルは OpenSearch Service でマスターユーザーの権限を持ちます。

複数のクラスターで同じユーザーを使用する場合、Amazon Cognito を使用して Dashboards にアクセスする場合、または署名バージョン 4 の署名をサポートする OpenSearch クライアントがある場合は、IAM をお勧めします。

- [内部ユーザーデータベース] – 内部ユーザーデータベースにマスターを作成する場合 (ユーザー名とパスワードの組み合わせを使用)、HTTP 基本認証 (および IAM 認証情報) を使用してクラスターにリクエストを行うことができます。ほとんどのクライアントは、curl を含む基本認証をサポートしています。curl は、--aws-sigv4 オプションで AWS 署名バージョン 4 もサポートしています。内部ユーザーデータベースは OpenSearch インデックスに保存されるため、他のクラスターと共有することはできません。

  複数のクラスター間でユーザーを再利用する必要がない場合、Dashboards へのアクセスに Amazon Cognito ではなく HTTP 基本認証を使用する場合、または基本認証のみをサポートするクライアントがある場合は、内部ユーザーデータベースをお勧めします。OpenSearch Service の使用を開始するには、内部ユーザーデータベースが最もシンプルな方法です。

きめ細かなアクセスコントロールの有効化

コンソール、 AWS CLI、または設定 API を使用して、きめ細かなアクセスコントロールを有効にします。手順については、「ドメインの作成と管理」を参照してください。

きめ細かなアクセスコントロールには、OpenSearch または Elasticsearch 6.7 以降が必要です。また、ドメインへのすべてのトラフィックに HTTPS、保管時のデータの暗号化、およびノード間の暗号化を要求します。きめ細かなアクセス制御の、高度な機能の設定方法によっては、さらに多くのリクエストを処理するために、個々のデータノードにコンピューティングリソースとメモリリソースが必要になる場合があります。きめ細かなアクセスコントロールを有効にした後、無効にすることはできません。

既存のドメインでのきめ細かなアクセスコントロールの有効化

OpenSearch または Elasticsearch 6.7 以降を実行している既存のドメインに対して、きめ細かなアクセスコントロールを有効にすることができます。

既存のドメインに対してきめ細かなアクセスコントロールを有効にするには (コンソール)

1. ドメインを選択し、[アクション] および [セキュリティ設定の編集] を選択します。
2. [きめ細かなアクセスコントロールを有効にする] を選択します。
3. マスターユーザーの作成方法を選択します。

   - ユーザー管理に IAM を使用する場合は、[IAM ARN をマスターユーザーとして設定] を選択し、IAM ロールの ARN を指定します。
   - 内部ユーザーデータベースを使用する場合は、[Create master user] (マスターユーザーの作成) を選択し、ユーザー名とパスワードを指定します。
4. (オプション) [Open/IP ベースのアクセスポリシーの移行期間を有効にする] を選択します。この設定により、30 日間の移行期間が有効になります。この期間中、既存のユーザーが中断することなくドメインにアクセスできるようになり、IP ベースのアクセスポリシーは引き続きドメインで動作します。この移行期間中に、管理者が必要なロールを作成し、それらをドメインのユーザーにマップすることをお勧めします。オープンアクセスポリシーまたは IP ベースのアクセスポリシーの代わりに ID ベースのポリシーを使用している場合は、この設定を無効にすることができます。

   また、移行期間中にきめ細かなアクセス制御を使用するようにクライアントを更新する必要があります。たとえば、IAM ロールをきめ細かなアクセスコントロールにマッピングする場合、AWS 署名バージョン 4 でリクエストの署名を開始するようにクライアントを更新する必要があります。きめ細かなアクセス制御を使用して HTTP 基本認証を構成する場合、リクエストに応じた基本認証資格情報を提供するようにクライアントを更新する必要があります。

   移行期間中、ドメインの OpenSearch Dashboards エンドポイントにアクセスするユーザーは、ログインページではなく、Discover ページに直接アクセスします。管理者とマスターユーザーは、Login を選択して管理者の認証情報を使用してログインし、ロールのマッピングを設定できます。

> **Important**
>
> OpenSearch Service は 30 日後に移行期間を自動的に無効にします。必要なロールを作成してユーザーにマップしたら、すぐに終了することをお勧めします。移行期間が終了した後、再度有効化することはできません。

5. [Save changes] (変更の保存) をクリックします。

この変更により、青/緑のデプロイがトリガーされ、その間にクラスターの状態が赤になりますが、すべてのクラスターオペレーションは影響を受けません。

既存のドメインに対してきめ細かなアクセスコントロールを有効にするには (CLI)

きめ細かなアクセスコントロールを使用して移行期間を有効にするには、`AnonymousAuthEnabled` を `true` に設定します。

```
aws opensearch update-domain-config --domain-name test-domain --region us-east-1 \
      --advanced-security-options '{ "Enabled": true,
 "InternalUserDatabaseEnabled":true, "MasterUserOptions": {"MasterUserName":"master-
username","MasterUserPassword":"master-password"},"AnonymousAuthEnabled": true}'
```

default_role について

きめ細かなアクセスコントロールには、ロールマッピングが必要です。ドメインがアイデンティティベースのアクセスポリシーを使用している場合、OpenSearch Service は、既存のユーザーを適切に移行できるように、ユーザーを default_role と呼ばれる新しいロールに自動的にマップします。この一時的なマッピングにより、独自のロールマッピングを作成するまで、ユーザーは IAM 署名付き GET および PUT リクエストを正常に送信できます。

ロールによって、OpenSearch サービスドメインにセキュリティの脆弱性や欠陥が追加されることはありません。独自のロールを設定したら、すぐにデフォルトのロールを削除して、それに応じてマッピングすることをお勧めします。

移行シナリオ

次の表に、既存のドメインできめ細かなアクセス制御を有効にする前と後の各認証方式の動作と、管理者がユーザーをロールに適切にマッピングするために必要な手順について説明します。

認証方法	きめ細かなアクセスコントロールの有効化の前	きめ細かなアクセスコントロールの有効化の後	管理タスク
アイデンティティベースのポリシー	IAM ポリシーを満たすすべてのユーザーが、ドメインにアクセスできます。	移行期間を有効にする必要はありません。 OpenSearch Service は、IAM ポリシーを満たすすべてのユーザーがドメインに引き続きアクセスできるように、これらのユーザーを default_role に自動的にマップします。	1. ドメインでカスタムロールマッピングを作成します。 2. default_role を削除します。
IP ベースのポリシー	許可された IP アドレスまたは CIDR ブロックのすべてのユーザーがドメインにアクセスできます。	30 日間の移行期間中は、許可された IP アドレスまたは CIDR ブロックのすべてのユーザーが引き続きドメインにアクセスできます。	1. ドメインでカスタムロールマッピングを作成します。 2. ロールマッピングの設定に応じて、基本認証情報または IAM 認証情報を提供するようにクライアントを更新します。 3. 移行期間を無効にします。許可された IP アドレスまたは CIDR ブロックからのユーザーは、基本認証または IAM 認証情報なしでリクエストを送信すると、ドメインへのアクセスが失われます。
オープンアクセスポリシー	インターネット上のすべてのユーザーがドメインにアクセスできます。	30 日間の移行期間中は、インターネット上のすべてのユーザーが引き続きドメインにアクセスできます。	1. ドメインでロールマッピングを作成します。 2. ロールマッピングの設定に応じて、基本認証情報または IAM 認証情報を提供するようにクライアントを更新します。

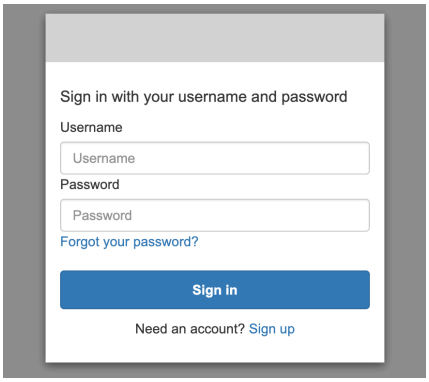
認証方法	きめ細かなアクセスコントロールの有効化の前	きめ細かなアクセスコントロールの有効化の後	管理タスク
			3. 移行期間を無効にします。基本認証または IAM 認証情報なしでリクエストを送信するユーザーは、ドメインにアクセスできなくなります。

マスターユーザーとして OpenSearch Dashboards にアクセスする

きめ細かなアクセスコントロールには、管理タスクをシンプルにする OpenSearch Dashboards プラグインがあります。Dashboards を使用して、ユーザー、ロール、マッピング、アクショングループ、テナントを管理できます。ただし、OpenSearch Dashboards サインインページと基になる認証方法は、ユーザーの管理方法とドメインの設定方法によって異なります。

- ユーザー管理に IAM を使用する場合は、the section called “OpenSearch Dashboards の Amazon Cognito 認証” を使用して、Dashboards にアクセスします。それ以外の方法では、Dashboards の「機能しない」サインインページが表示されます。「the section called “制限事項”」を参照してください。

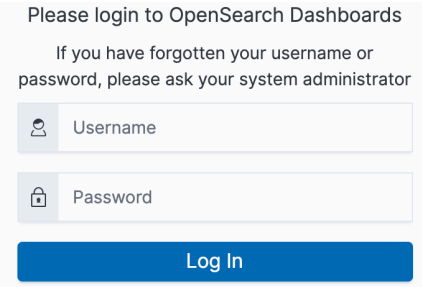
  Amazon Cognito 認証では、ID プールから引き受けたいずれかのロールが、マスターユーザーに指定した IAM ロールと一致する必要があります。この設定の詳細については、「the section called “(オプション) きめ細かなアクセスの設定”」および「the section called “チュートリアル: Cognito 認証によるきめ細かなアクセスコントロール”」を参照してください。

- 内部ユーザーデータベースを使用することを選択した場合は、マスターユーザー名とパスワードを使用して Dashboards にサインインできます。HTTPS 経由で Dashboards にアクセスする必要が

あります。このログイン画面は、Dashboards の Amazon Cognito 認証と SAML 認証の両方に置き換えられます。

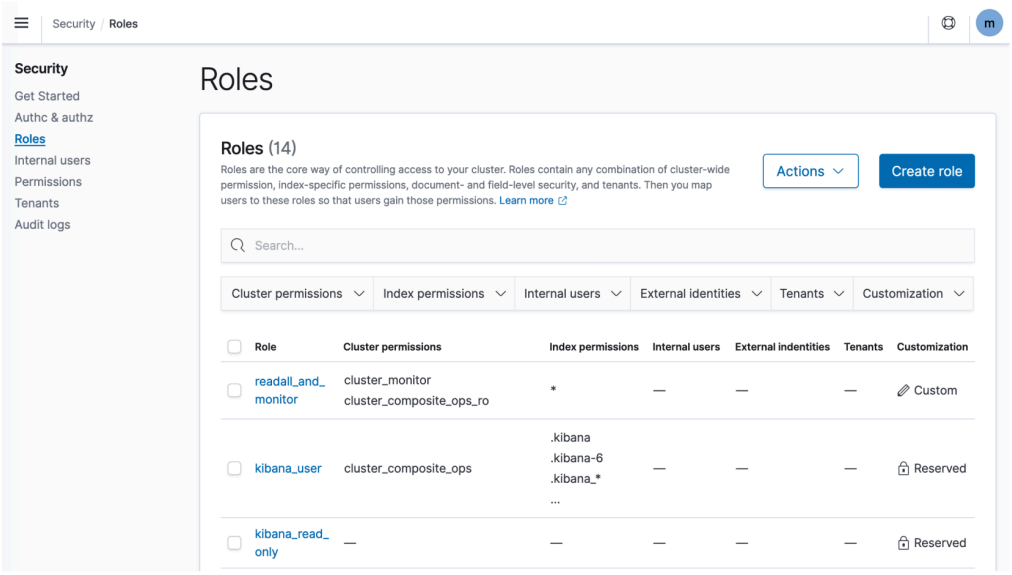
この設定の詳細については、「the section called “チュートリアル: 内部ユーザーデータベースと基本認証”」を参照してください。



- SAML 認証を使用することを選択した場合は、外部 ID プロバイダーからの認証情報を使用してサインインできます。詳細については、「the section called “OpenSearch Dashboards の SAML 認証”」を参照してください。

許可の管理

「the section called “主要なコンセプト”」で説明しているように、ロール、ユーザー、マッピングを使用してきめ細かなアクセスコントロールの許可を管理します。このセクションでは、これらのリソースを作成して適用する方法について説明します。これらの操作を実行するには、マスターユーザーとして Dashboards にサインインすることをお勧めします。



> **Note**
>
> ユーザーに付与するために選択する権限は、ユースケースによって大きく異なります。このドキュメントですべてのシナリオを実行可能にカバーすることはできません。ユーザーに付与するアクセス許可を決定するときは、次のセクションで説明する OpenSearch クラスターとインデックスのアクセス許可を参照し、常に 最小特権の原則 に従ってください。

ロールの作成

OpenSearch Dashboards、または REST API の `_plugins/_security` オペレーションを使用して、きめ細かなアクセスコントロール用の新しいロールを作成できます。詳細については、「ロールの作成」を参照してください。

きめ細かなアクセスコントロールには、多くの事前定義されたロールも含まれます。OpenSearch Dashboards や Logstash などのクライアントは、OpenSearch に対して多種多様なリクエストを行うため、最小限の許可のセットを使用してロールを手動で作成するのが難しくなる場合があります。例えば、`opensearch_dashboards_user` ロールには、ユーザーがインデックスパターン、可視化、ダッシュボード、およびテナントを操作するのに必要な許可が含まれます。このロールは、他のインデックスへのアクセスを許可する追加のロールと共に、Dashboards にアクセスするすべてのユーザーまたはバックエンドロールにマッピングすることをお勧めします。

Amazon OpenSearch Service では、次の OpenSearch ロールは提供されていません。

- `observability_full_access`
- `observability_read_access`
- `reports_read_access`
- `reports_full_access`

Amazon OpenSearch Service には、OpenSearch では利用できないロールがいくつかあります。

- `ultrawarm_manager`
- `ml_full_access`
- `cold_manager`
- `notifications_full_access`
- `notifications_read_access`

クラスターレベルのセキュリティ

クラスターレベルの許可により、_mget、_msearch、_bulk などの広範なリクエストの実施、ヘルスのモニタリング、スナップショットの取得などを可能にするかどうかを制御できます。ロールを作成するときに、[クラスター許可] セクションを使用してこれらの許可を管理します。クラスタレベルの権限の完全なリストについては、「クラスタ権限」を参照してください。

多くの場合、個別の権限ではなく、デフォルトのアクショングループの組み合わせを使用して、目的のセキュリティ体制を実現できます。クラスターレベルのアクショングループのリストについては、「クラスターレベル」を参照してください。

インデックスレベルのセキュリティ

インデックスレベルの許可により、新しいインデックスの作成、インデックスの検索、ドキュメントの読み取りと書き込み、ドキュメントの削除、エイリアスの管理などを可能にするかどうかを制御できます。ロールを作成するときに、[インデックス許可] セクションを使用してこれらのアクセス許可を管理します。インデックスレベルの権限の完全なリストについては、「インデックス権限」を参照してください。

多くの場合、個別の権限ではなく、デフォルトのアクショングループの組み合わせを使用して、目的のセキュリティ体制を実現できます。インデックスレベルのアクショングループのリストについては、「インデックスレベル」を参照してください。

ドキュメントレベルのセキュリティ

ドキュメントレベルのセキュリティにより、インデックス内のどのドキュメントをユーザーが表示できるかを制限できます。ロールを作成するときに、インデックスパターンと OpenSearch クエリを指定します。そのロールにマッピングされたすべてのユーザーは、クエリに一致するドキュメントのみを表示できます。ドキュメントレベルのセキュリティは、検索時に受け取るヒットの数に影響します。

詳細については、「ドキュメントレベルのセキュリティ」を参照してください。

フィールドレベルのセキュリティ

フィールドレベルのセキュリティにより、どのドキュメントフィールドをユーザーが表示できるかを制御できます。ロールを作成するときに、含めるフィールドと除外するフィールドのリストを追加します。フィールドを含めると、そのロールにマッピングされたユーザーはそれらのフィールドのみを表示できます。フィールドを除外すると、除外されたフィールドを除くすべてのフィールドを表示できます。フィールドレベルのセキュリティは、検索時にヒットに含まれるフィールドの数に影響します。

詳細については、「フィールドレベルのセキュリティ」を参照してください。

フィールドのマスキング

フィールドのマスキングは、フィールドレベルのセキュリティに代わるもので、フィールド内のデータを完全に削除するのではなく、匿名化できます。ロールを作成するときに、マスクするフィールドのリストを追加します。フィールドのマスキングは、検索時にフィールドの内容を表示できるかどうかに影響します。

> **Tip**
>
> 標準のマスキングをフィールドに適用すると、OpenSearch Service は不正確な集計結果を引き起こす可能性のある安全でランダムなハッシュを使用します。マスキングされたフィールドで集計を実施するには、代わりにパターンベースのマスキングを使用します。

ユーザーの作成

内部ユーザーデータベースを有効にした場合、OpenSearch Dashboards、または REST API の `_plugins/_security` オペレーションを使用してユーザーを作成できます。詳細については、「ユーザーの作成」を参照してください。

マスターユーザーに IAM を選択した場合は、Dashboards のこの部分は無視してください。その代わりに IAM ロールを作成します。詳細については、IAM ユーザーガイドを参照してください。

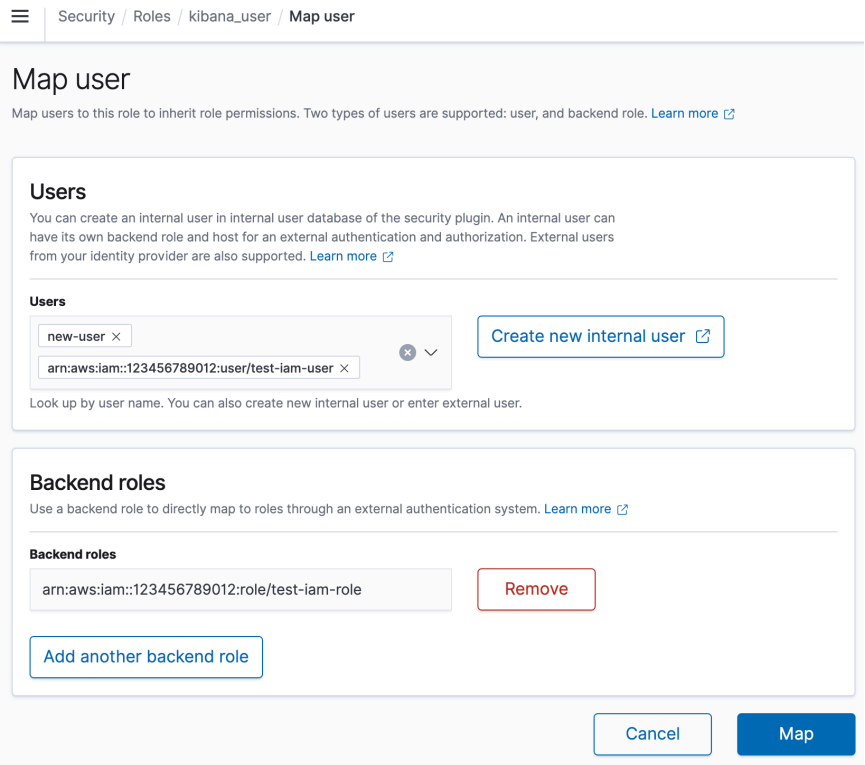
ユーザーへのロールのマッピング

ロールのマッピングは、きめ細かなアクセスコントロールの最も重要な側面です。きめ細かなアクセスコントロールには、使用の開始に役立ついくつかの事前定義されたロールがありますが、ロールをユーザーにマッピングしない限り、クラスターに対するすべてのリクエストは許可エラーという結果になります。

バックエンドロールを使用すると、ロールのマッピングプロセスを簡素化できます。このロールを使えば、同じロールを 100 人のユーザーに割り当てるのではなく、100 人のユーザーが共有している 1 つのバックエンドロールにロールをマッピングできます。バックエンドロールは、IAM ロールまたは任意の文字列にすることができます。

- [Users] (ユーザー) セクションで、ユーザー、ユーザー ARN、および Amazon Cognito ユーザー文字列を指定します。Cognito ユーザー文字列の形式は `Cognito/user-pool-id/username` です。

- [バックエンドロール] セクションで、バックエンドロールと IAM ロール ARN を指定します。



OpenSearch Dashboards、または REST API の `_plugins/_security` オペレーションを使用して、ロールをユーザーにマッピングできます。詳細については、「ユーザーをロールにマッピングする」を参照してください。

アクショングループの作成

アクショングループは、さまざまなリソースで再利用できる許可のセットです。OpenSearch Dashboards、または REST API の `_plugins/_security` オペレーションを使用して新しいアクショングループを作成できますが、ほとんどのユースケースではデフォルトのアクショングループで十分です。デフォルトのアクショングループの詳細については、「デフォルトのアクショングループ」を参照してください。

OpenSearch Dashboards マルチテナンシー

テナントは、インデックスパターン、可視化、ダッシュボード、その他の Dashboards オブジェクトを保存するための領域です。Dashboards マルチテナンシーを使用すると、他の Dashboards ユーザーと作業を安全に共有できます (プライベートに保持できます)。また、テナントを動的に設定できます。どのロールがテナントにアクセスできるか、それらのロールに読み取りアクセスがあるか書

き込みアクセスがあるかを制御できます。グローバルテナントがデフォルトです。詳細については、「OpenSearch Dashboards マルチテナンシー」を参照してください。

現在のテナントを表示したりテナントを変更したりするには

1. OpenSearch Dashboards に移動し、サインインします。
2. 右上隅にあるユーザーアイコンを選択し、[テナントの切り替え] を選択します。
3. 可視化またはダッシュボードを作成する前に、テナントを確認します。他のすべての Dashboards ユーザーと作業を共有する場合は、[グローバル] を選択します。一部の Dashboards ユーザーと作業を共有するには、別の共有テナントを選択します。それ以外の場合は、[プライベート] を選択します。

> **Note**
>
> OpenSearch Dashboards は、テナントごとに個別のインデックスを維持し、`tenant_template` と呼ばれるインデックステンプレートを作成します。テナントインデックスのマッピングが正しく構成されていない場合、OpenSearch Dashboards が誤動作する可能性があるため、`tenant_template` インデックスを削除または変更しないでください。

推奨される設定

きめ細かなアクセスコントロールは他のセキュリティ機能と連携するため、ここでは、ほとんどのユースケースで適切に機能するきめ細かなアクセスコントロールの推奨される設定を示します。

説明	マスターユーザー	ドメインアクセスポリシー
OpenSearch API への呼び出しに IAM 認証情報を使用し、SAML 認証を使用して、Dashboards にアクセスします。Dashboards または REST API を使用して、きめ細かなアクセスコン	IAM ロールまたはユーザー	JSON `{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {`

説明	マスターユーザー	ドメインアクセスポリシー
トロールのロールを管理します。		"AWS": "*" }, "Action": "es:ESHttp*", "Resource": "arn:aws:es: us-east-1 :111122223333 :domain/domain-name /*" }] }
OpenSearch API への呼び出しに IAM 認証情報または基本認証を使用します。Dashboards または REST API を使用して、きめ細かなアクセスコントロールのロールを管理します。 この設定は、特に基本認証のみをサポートする OpenSearch クライアントがある場合、多くの柔軟性を提供します。 既存の ID プロバイダーがある場合は、SAML 認証を使用して、Dashboards にアクセスします。それ以外の場合は、内部ユーザーデータベースで Dashboards ユーザーを管理します。	ユーザー名とパスワード	JSON { "Version":"2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "arn:aws:es: us-east-1 :111122223333 :domain/domain-name /*" }] }

説明	マスターユーザー	ドメインアクセスポリシー

OpenSearch API への呼び出しに IAM 認証情報を使用し、Amazon Cognito を使用して、Dashboards にアクセスします。Dashboards または REST API を使用して、きめ細かなアクセスコントロールのロールを管理します。	IAM ロールまたはユーザー	JSON
		{
		"Version":"2012-10-17",
		"Statement": [
		{
		"Effect": "Allow",
		"Principal": {
		"AWS": "*"
		},
		"Action": "es:ESHttp*",
		"Resource": "arn:aws:es: *us-east-1* :*111122223333* :domain/*domain-name* /*"
		}
		]
		}

説明	マスターユーザー	ドメインアクセスポリシー

OpenSearch API への呼び出しに IAM 認証情報を使用し、Dashboards へのほとんどのアクセスをブロックします。REST API を使用して、きめ細かなアクセスコントロールのロールを管理します。	IAM ロールまたはユーザー	JSON
		{
		"Version":"2012-10-17",
		"Statement": [
		{
		"Effect": "Allow",
		"Principal": {
		"AWS": "*"
		},
		"Action": "es:ESHttp*",
		"Resource": "arn:aws:es: *us-east-1* :*111122223333* :domain/*domain-name* /*"
		},
		{
		"Effect": "Deny",
		"Principal": {
		"AWS": "*"
		},
		"Action": "es:ESHttp*",
		"Resource": "arn:aws:es: *us-east-1* :*111122223333* :domain/*domain-name* /_dashboards*"
		}
		]
		}

制限事項

きめ細かなアクセスコントロールには、いくつかの重要な制限があります。

- hosts のロールマッピングの側面は、ロールをホスト名または IP アドレスにマッピングすることですが、ドメインが VPC 内にある場合は機能しません。ただし、ロールをユーザーとバックエンドロールにマッピングすることは可能です。

- マスターユーザーに IAM を選択し、Amazon Cognito または SAML 認証を有効にしない場合、Dashboards の「機能しない」サインインページが表示されます。
- マスターユーザーに IAM を選択しても、内部ユーザーデータベースにユーザーを作成することはできます。ただし、この設定では HTTP 基本認証が有効になっていないため、これらのユーザー認証情報で署名されたリクエストは拒否されます。
- SQL を使用して、アクセスできないインデックスをクエリすると、「許可なし」のエラーが表示されます。インデックスが存在しない場合、「インデックスなし」のエラーが表示されます。このエラーメッセージの違いは、インデックスの名前を推測できれば、そのインデックスが存在していることがわかる点です。

  この問題を最小限に抑えるために、インデックス名に機密情報を含めないでください。SQL へのすべてのアクセスを拒否するには、ドメインアクセスポリシーに以下の要素を追加します。

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": [
      "*"
    ]
  },
  "Action": [
    "es:*"
  ],
  "Resource": "arn:aws:es:us-east-1:123456789012:domain/my-domain/_plugins/_sql"
}
```

- ドメインバージョンが 2.3 以上で、きめ細かいアクセス制御が有効になっている場合、`max_clause_count` を 1 に設定するとドメインで問題が発生します。このアカウントにはもっと高い数値を設定することをおすすめします。
- きめ細かなアクセスコントロールが設定されていないドメインできめ細かなアクセスコントロールを有効にする場合、直接クエリ用に作成されたデータソースについては、きめ細かなアクセスコントロールロールを自分で設定する必要があります。きめ細かなアクセスロールを設定する方法の詳細については、「Creating Amazon OpenSearch Service data source integrations with Amazon S3」を参照してください。

マスターユーザーの変更

マスターユーザーの詳細を忘れた場合は、コンソール、 AWS CLI、または configuration API を使用して再設定できます。

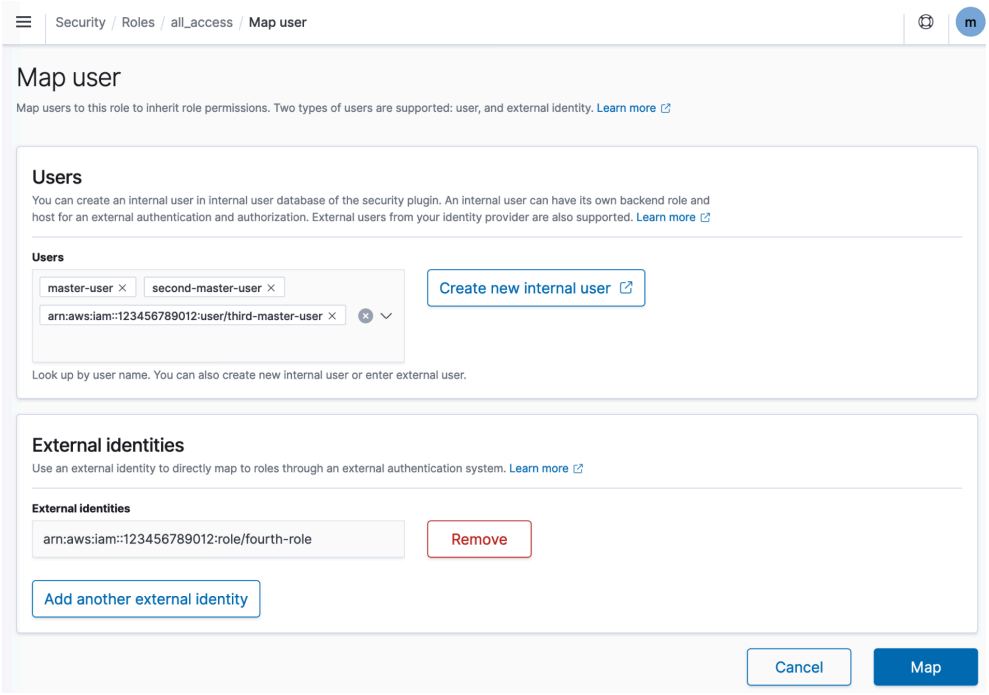
マスターユーザーを変更するには (コンソール)

1. https://console.aws.amazon.com/aos/home/ の Amazon OpenSearch Service コンソールに移動します。
2. ドメインを選択し、[Actions] (アクション)、[Edit security configuration] (セキュリティ設定の編集) を選択します。
3. [マスターユーザーとして IAM ARN を設定] または [マスターユーザーを作成] を選択します。
   - 以前に IAM マスターユーザーを使用していた場合、きめ細かなアクセスコントロールにより、指定した新しい IAM ARN に `all_access` ロールが再マッピングされます。
   - 以前に内部ユーザーデータベースを使用していた場合、きめ細かなアクセスコントロールにより、新しいマスターユーザーが作成されます。新しいマスターユーザーを使用して、古いマスターユーザーを削除できます。
   - 内部ユーザーデータベースから IAM マスターユーザーへの切り替えでは、内部ユーザーデータベースからユーザーを削除しません。代わりに、HTTP 基本認証を無効にするだけです。内部ユーザーデータベースからユーザーを手動で削除するか、HTTP 基本認証を再度有効にする必要がある場合に備えて、ユーザーを保持します。
4. [Save changes] (変更の保存) をクリックします。

追加のマスターユーザー

ドメインの作成時にマスターユーザーを指定しますが、必要に応じて、このマスターユーザーを使用して追加のマスターユーザーを作成できます。OpenSearch Dashboards または REST API の 2 つのオプションがあります。

- Dashboards で [セキュリティ]、[ロール] の順に選択し、新しいマスターユーザーを `all_access` および `security_manager` ロールにマッピングします。



- REST API を使用するには、以下のリクエストを送信します。

```
PUT _plugins/_security/api/rolesmapping/all_access
{
  "backend_roles": [
    "arn:aws:iam::123456789012:role/fourth-master-user"
  ],
  "hosts": [],
  "users": [
    "master-user",
    "second-master-user",
    "arn:aws:iam::123456789012:user/third-master-user"
  ]
}
```

```
PUT _plugins/_security/api/rolesmapping/security_manager
{
  "backend_roles": [
    "arn:aws:iam::123456789012:role/fourth-master-user"
  ],
  "hosts": [],
  "users": [
    "master-user",
    "second-master-user",
    "arn:aws:iam::123456789012:user/third-master-user"
```

```
  ]
}
```

これらのリクエストは現在のロールマッピングを置き換えるため、PUT リクエストに現在のすべてのロールを含めることができるように、最初に GET リクエストを実行します。REST API が特に役立つのは、Dashboards にアクセスできず、IAM ロールを Amazon Cognito から `all_access` ロールにマッピングする場合です。

手動スナップショット

きめ細かなアクセスコントロールにより、手動スナップショットの取得がいくらか複雑になります。スナップショットリポジトリを登録するには、HTTP 基本認証を他のすべての目的で使用する場合でも、the section called “前提条件” で定義されるように、`TheSnapshotRole` を引き受けるための `iam:PassRole` 許可がある IAM ロールに `manage_snapshots` ロールをマッピングする必要があります。

次に、その IAM ロールを使用して、「the section called “手動スナップショットレポジトリの登録”」に概説されているように、署名付きリクエストをドメインに送信します。

統合

OpenSearch Service で他の AWS サービスを使用する場合は、それらのサービスの IAM ロールに適切なアクセス許可を付与する必要があります。例えば、Firehose の配信ストリームでは、`firehose_delivery_role` という名前の IAM ロールがよく使用されます。Dashboards で、きめ細かなアクセスコントロール用のロールを作成し、そのロールに IAM ロールをマッピングします。この場合、新しいロールには以下の許可が必要です。

```
{
  "cluster_permissions": [
    "cluster_composite_ops",
    "cluster_monitor"
  ],
  "index_permissions": [{
    "index_patterns": [
      "firehose-index*"
    ],
    "allowed_actions": [
      "create_index",
      "manage",
```

```
      "crud"
    ]
  }]
}
```

許可は、各サービスが実行するアクションによって異なります。データのインデックスを作成する AWS IoT ルールまたは AWS Lambda 関数には Firehose と同様のアクセス許可が必要になる可能性がありますが、検索のみを実行する Lambda 関数では、より制限されたセットを使用できます。

REST API の相違点

きめ細かなアクセスコントロールの REST API は、OpenSearch/Elasticsearch バージョンによってわずかに異なります。PUT リクエストを行う前に、GET リクエストを行って、想定されるリクエスト本文を確認します。例えば、GET への _plugins/_security/api/user リクエストはすべてのユーザーを返すため、それらの結果を変更して使用して、有効な PUT リクエストを行うことができます。

Elasticsearch 6.x では、ユーザーの作成リクエストは以下のようになります。

```
PUT _opendistro/_security/api/user/new-user
{
  "password": "some-password",
  "roles": ["new-backend-role"]
}
```

OpenSearch または Elasticsearch 7.x では、リクエストは次のようになります (Elasticsearch を使用している場合は、_plugins を _opendistro に変更してください)。

```
PUT _plugins/_security/api/user/new-user
{
  "password": "some-password",
  "backend_roles": ["new-backend-role"]
}
```

さらに、Elasticsearch 6.x では、テナントはロールのプロパティです。

```
GET _opendistro/_security/api/roles/all_access

{
```

```
  "all_access": {
    "cluster": ["UNLIMITED"],
    "tenants": {
      "admin_tenant": "RW"
    },
    "indices": {
      "*": {
        "*": ["UNLIMITED"]
      }
    },
    "readonly": "true"
  }
}
```

OpenSearch と Elasticsearch 7.x では、それらは独自の URI を持つオブジェクトです (Elasticsearch を使用している場合は、_plugins を _opendistro に変更してください)。

```
GET _plugins/_security/api/tenants

{
  "global_tenant": {
    "reserved": true,
    "hidden": false,
    "description": "Global tenant",
    "static": false
  }
}
```

OpenSearch REST API のドキュメントについては、「セキュリティプラグイン API リファレンス」を参照してください。

Tip

内部ユーザーデータベースを使用する場合は、curl を使用してリクエストを行い、ドメインをテストできます。次のサンプルコマンドを試してください。

```
curl -XGET -u 'master-user:master-user-password' 'domain-endpoint/_search'
curl -XGET -u 'master-user:master-user-password' 'domain-endpoint/_plugins/
_security/api/user'
```

チュートリアル: IAM マスターユーザーと Amazon Cognito 認証を使用してドメインを設定する

このチュートリアルでは、きめ細かなアクセスコントロールの一般的な Amazon OpenSearch Service のユースケースとして、OpenSearch Dashboards の Amazon Cognito 認証を受ける IAM マスターユーザーについて説明します。

マスター IAM ロールと制限付き IAM ロールを設定し、それらを Amazon Cognito のユーザーに関連付けます。その後、マスターユーザーは OpenSearch Dashboards にサインインし、制限付きユーザーをロールにマッピングし、きめ細かなアクセスコントロールを使用してユーザーのアクセス許可を制限できます。



これらの手順は、認証に Amazon Cognito ユーザープールを使用しますが、この同じ基本プロセスは、Cognito 認証プロバイダに対して機能するため、異なる IAM ロールを異なるユーザーに割り当てることができます。

このチュートリアルでは、次の手順を実行します。

1. マスター IAM ロールと制限付き IAM ロールを作成する
2. Cognito 認証を使用してドメインを作成する
3. Cognito ユーザープールとアイデンティティプールの設定
4. OpenSearch Dashboards でロールをマッピングする
5. アクセス許可をテストする

ステップ 1: マスター IAM ロールと制限付き IAM ロールを作成する

AWS Identity and Access Management (IAM) コンソールに移動し、2 つの個別のロールを作成します。

- `MasterUserRole` – マスターユーザー。クラスターに対するフルアクセスの許可を持ち、ロールとロールマッピングを管理します。

- `LimitedUserRole` – マスターユーザーよりも制限されたロール。マスターユーザーから制限付きアクセスが許可されます。

ロールの作成手順については、「IAM ユーザーガイド」の「カスタム信頼ポリシーを使用してロールを作成する」を参照してください。

両方のロールに次の信頼ポリシーがある必要があります。これにより、Cognito ID プールがロールを引き受けることができます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Federated": "cognito-identity.amazonaws.com"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "cognito-identity.amazonaws.com:aud": "{identity-pool-id}"
      },
      "ForAnyValue:StringLike": {
        "cognito-identity.amazonaws.com:amr": "authenticated"
      }
    }
  }]
}
```

> **Note**
>
> `identity-pool-id` を Amazon Cognito ID プールの一意の識別子に置き換えます。例えば、`us-east-1:0c6cdba7-3c3c-443b-a958-fb9feb207aa6`。

ステップ 2: Cognito 認証を使用してドメインを作成する

https://console.aws.amazon.com/aos/home/ で Amazon OpenSearch Service コンソールに移動し、次の設定でドメインを作成します。

- OpenSearch 1.0 以降、またはElasticsearch 7.8 以降
- パブリックアクセス
- マスターユーザー (前のステップで作成) として `MasterUserRole` できめ細かなアクセスコントロールが可能
- OpenSearch Dashboardsで有効になっている Amazon Cognito 認証。Cognito 認証を有効にし、ユーザーと ID プールを選択する手順については、「the section called “Amazon Cognito 認証を使用するためのドメインの設定”」を参照してください。
- 次のドメインアクセスポリシー:

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/{domain-name}/*"
    }
  ]
}
```

- ドメインへのすべてのトラフィックに HTTPS を必須とする
- ノード間の暗号化
- 保管中のデータの暗号化

ステップ 3: Cognito ユーザーの設定

ドメインの作成中に、「Amazon Cognito デベロッパーガイド」の「Create a user pool」に従って、Amazon Cognito 内でマスターユーザーと、制限付きユーザーを設定します。最後に、「Amazon Cognito でアイデンティティプールを作成する」の手順に従ってアイデンティティプールを設定します。ユーザープールと ID プールは、同じ AWS リージョンに存在している必要があります。

ステップ 4: OpenSearch Dashboards でロールをマッピングする

ユーザーの設定が完了したので、マスターユーザーとして OpenSearch Dashboards にサインインし、ユーザーをロールにマッピングできます。

1. OpenSearch Service コンソールに戻り、作成したドメインの OpenSearch Dashboards URL に移動します。URL はこの形式に従います: *domain-endpoint*/_dashboards/。
2. master-user 認証情報を使用してサインインします。
3. [Add sample data] (サンプルデータを追加) を選択し、サンプルフライトデータを追加します。
4. 左側のナビゲーションペインで [Security] (セキュリティ)、[Roles] (ロール)、[Create role] (ロールを作成) の順に選択します。
5. ロールに new-role という名前を付けます。
6. [Index] (インデックス) には、opensearch_dashboards_sample_data_fli* (Elasticsearch ドメインの kibana_sample_data_fli*) を指定します。
7. [Index permissions] (インデックスアクセス許可) には、[read] (読み取り) を選択します。
8. [ドキュメントレベルのセキュリティ] で、以下のクエリを指定します。

```
{
  "match": {
    "FlightDelay": true
  }
}
```

9. フィールドレベルのセキュリティでは、[除外] を選択し、FlightNum を指定します。
10. [匿名化] では、Dest を指定します。
11. [作成] を選択します。
12. [マッピングされたユーザー]、[マッピングの管理] を選択します。外部 ID として LimitedUserRole の Amazon リソースネーム (ARN) を追加し、[Map] (マッピング) を選択します。

13. ロールのリストに戻り、[opensearch_dashboards_user] を選択します。[マッピングされたユーザー]、[マッピングの管理] を選択します。バックエンドロールとして `LimitedUserRole` の ARN を追加し、[マップ] を選択します。

ステップ 5: アクセス許可をテストする

ロールが正しくマッピングされると、制限付きユーザーとしてサインインし、アクセス許可をテストできます。

1. 新しいプライベートブラウザウィンドウで、ドメインの OpenSearch Dashboards URL に移動し、`limited-user` 認証情報を使用してサインインして、[Explore on my own] (自力で調べる) を選択します。
2. [開発ツール] に進み、デフォルトの検索を実行します。

   ```
   GET _search
   {
     "query": {
       "match_all": {}
     }
   }
   ```

   許可エラーに注意してください。`limited-user` には、クラスター全体の検索を実行する許可がありません。
3. 別の検索を実行します。

   ```
   GET opensearch_dashboards_sample_data_flights/_search
   {
     "query": {
       "match_all": {}
     }
   }
   ```

   一致するすべてのドキュメントでは、[FlightDelay] フィールドが `true` であり、`Dest` フィールドが匿名化されて、`FlightNum` フィールドはありません。
4. 元のブラウザウィンドウで、`master-user` としてサインインし、[開発ツール] を選択して、同じ検索を実行します。許可、ヒット数、一致するドキュメント、含まれるフィールドが異なっています。

チュートリアル: 内部ユーザーデータベースと HTTP 基本認証でドメインを設定する

このチュートリアルでは、内部ユーザーデータベースのマスターユーザーと OpenSearch Dashboards の HTTP 基本認証という、別の一般的なきめ細かなアクセスコントロールのユースケースについて説明します。その後、マスターユーザーは OpenSearch Dashboards にサインインし、内部ユーザーを作成し、そのユーザーをロールにマッピングし、きめ細かなアクセスコントロールを使用してユーザーのアクセス許可を制限できます。

このチュートリアルでは、次の手順を実行します。

1. マスターユーザーでドメインを作成する
2. OpenSearch Dashboards で内部ユーザーを設定する
3. OpenSearch Dashboards でロールをマッピングする
4. アクセス許可をテストする

ステップ 1: ドメインを作成する

https://console.aws.amazon.com/aos/home/ で Amazon OpenSearch Service コンソールに移動し、次の設定でドメインを作成します。

- OpenSearch 1.0 以降、またはElasticsearch 7.9 以降
- パブリックアクセス
- 内部ユーザーデータベース内のマスターユーザーによるきめ細かなアクセスコントロール (このチュートリアルの残りの部分では `TheMasterUser`)
- Dashboards の Amazon Cognito 認証が無効にされました
- 以下のアクセスポリシー:

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
```

```
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/{domain-name}/*"
    }
  ]
}
```

- ドメインへのすべてのトラフィックに HTTPS を必須とする
- ノード間の暗号化
- 保管中のデータの暗号化

ステップ 2: OpenSearch Dashboards で内部ユーザーを作成する

ドメインを取得したので、OpenSearch Dashboards にサインインして内部ユーザーを作成できます。

1. OpenSearch Service コンソールに戻り、作成したドメインの OpenSearch Dashboards URL に移動します。URL はこの形式に従います: *domain-endpoint*/_dashboards/。
2. TheMasterUser でサインインします。
3. [Add sample data] (サンプルデータを追加) を選択し、サンプルフライトデータを追加します。
4. 左側のナビゲーションペインで、[セキュリティ]、[内部ユーザー]、[内部ユーザーの作成] を選択します。
5. ユーザーに new-user という名前を付け、パスワードを指定します。次に [作成] を選択します。

ステップ 3: OpenSearch Dashboards でロールをマッピングします

ユーザーが設定されたので、ユーザーをロールにマップできます。

1. 次も、OpenSearch Dashboards の [セキュリティ] セクションで [ロール]、[ロールの作成] を選択します。
2. ロールに new-role という名前を付けます。
3. [インデックス] には、インデックスパターンに opensearch_dashboards_sample_data_fli* (Elasticsearch ドメインの kibana_sample_data_fli*) を指定します。
4. アクショングループで、[読み取り] を選択します。

5. [ドキュメントレベルのセキュリティ] で、以下のクエリを指定します。

```
{
  "match": {
    "FlightDelay": true
  }
}
```

6. フィールドレベルのセキュリティでは、[除外] を選択し、FlightNum を指定します。
7. [匿名化] では、Dest を指定します。
8. [作成] を選択します。
9. [マッピングされたユーザー]、[マッピングの管理] を選択します。次に、new-user を [ユーザー] に追加し、[マップ] を選択します。

ステップ 4: アクセス許可をテストします

ロールが正しくマッピングされると、制限付きユーザーとしてサインインし、アクセス許可をテストできます。

1. 新しいプライベートブラウザウィンドウで、ドメインの OpenSearch Dashboards URL に移動し、new-user 認証情報を使用してサインインして、[Explore on my own] (自力で調べる) を選択します。
2. [開発ツール] に進み、デフォルトの検索を実行します。

```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

許可エラーに注意してください。new-user には、クラスター全体の検索を実行する許可がありません。

3. 別の検索を実行します。

```
GET dashboards_sample_data_flights/_search
{
  "query": {
```

```
    "match_all": {}
  }
}
```

一致するすべてのドキュメントでは、[FlightDelay] フィールドが `true` であり、`Dest` フィールドが匿名化されて、`FlightNum` フィールドはありません。

4. 元のブラウザウィンドウで、`TheMasterUser` としてサインインし、[開発ツール] を選択して、同じ検索を実行します。許可、ヒット数、一致するドキュメント、含まれるフィールドが異なっています。

Amazon OpenSearch Service のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS コンプライアンスプログラムの一環として Amazon OpenSearch Service のセキュリティとコンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA が含まれます。

コンプライアンス要件がある場合は、任意のバージョンの OpenSearch または Elasticsearch 6.0 以降を使用することを検討してください。以前のバージョンの Elasticsearch は、保管中のデータの暗号化およびノード間の暗号化の組み合わせを提供しておらず、ニーズを満たすことはほとんどありません。きめ細かなアクセスコントロールがユースケースにとって重要である場合は、任意のバージョンの OpenSearch または Elasticsearch 6.7 以降の使用を検討してください。いずれにしても、ドメインの作成時に特定の OpenSearch または Elasticsearch バージョンを選択しても、コンプライアンスが保証されるわけではありません。

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

Amazon OpenSearch Service の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

OpenSearch Service には、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能があります。

- マルチ AZ ドメインとレプリカシャード
- 自動スナップショットと手動スナップショット

Amazon OpenSearch Service n の JWT 認証と承認

Amazon OpenSearch Service では、認証と承認に JSON Web トークン (JWT) を使用できるようになりました。JWT は、シングルサインオン (SSO) アクセスを付与するために使用される JSON ベースのアクセストークンです。OpenSearch Service の JWT を使用してシングルサインオントークンを作成し、OpenSearch Service ドメインへのリクエストを検証できます。JWT を使用するには、きめ細かなアクセスコントロールを有効にし、有効な RSA または ECDSA PEM 形式のパブリックキーを指定する必要があります。きめ細かなアクセスコントロールの詳細については、「Fine-grained access control in Amazon OpenSearch Service」を参照してください。

OpenSearch Service コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDKs を使用して、JSON ウェブトークンを設定できます。

考慮事項

Amazon OpenSearch Service で JWT を使用する前に、以下を考慮する必要があります。

- PEM 形式の RSA パブリックキーのサイズが大きいため、 AWS コンソールを使用して JWT 認証と承認を設定することをお勧めします。

- JWT のサブジェクトとロールフィールドを指定するときは、有効なユーザーとロールを指定する必要があります。指定しないと、リクエストは拒否されます。
- OpenSearch 2.11 は、JWT 認証に使用できる最も古い互換性のあるバージョンです。

ドメインアクセスポリシーの変更

ドメインを設定して JWT 認証と認可を使用する前に、ドメインアクセスポリシーを更新して JWT ユーザーがドメインにアクセスできるようにする必要があります。更新しないと、受信する JWT 認証リクエストはすべて拒否されます。サブリソース (/*) へのフルアクセスを提供するために推奨されるドメインアクセスポリシーは次のとおりです。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:ESHttp*",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

JWT 認証と認可の設定

ドメイン作成プロセス中、または既存のドメインを更新することで、JWT 認証と承認を有効にすることができます。セットアップ手順は、選択するオプションに応じて若干異なります。

次の手順では、OpenSearch Service コンソールで JWT 認証と承認用に既存のドメインを設定する方法について説明します。

1. [ドメイン設定] で、[OpenSearch の JWT 認証と承認] に移動し、[JWT 認証と承認を有効にする] を選択します。

2. ドメインに使用するパブリックキーを設定します。これを行うには、パブリックキーを含む PEM ファイルをアップロードするか、手動で入力します。

> **Note**
>
> アップロードまたは入力されたキーが有効でない場合、問題を指定する警告がテキストボックスの上に表示されます。

3. (オプション) [その他の設定] で、次のオプションフィールドを設定します。

   - サブジェクトキー — このフィールドを空のままにして JWT のデフォルト `sub` キーを使用できます。
   - ロールキー — このフィールドを空のままにして、JWT のデフォルト `roles` キーを使用できます。

   変更を行ったら、ドメインを保存します。

JWKS エンドポイントを使用して JWT を検証する

静的パブリックキーを設定する代わりに、ID プロバイダーからパブリックキーを動的に取得するように JSON Web Key Set (JWKS) エンドポイント URL を設定できます。JWKS エンドポイントを設定すると、OpenSearch Service は JWT 署名の検証に使用されるパブリックキーを自動的にフェッチしてキャッシュするため、ID プロバイダーが署名キーをローテーションするときにキーを手動で更新する必要がなくなります。

JWKS エンドポイント設定にはOpenSearch バージョン 3.3 以降が必要です。

> **Note**
>
> JWKS URL と静的パブリックキーの両方が設定されている場合、JWKS URL が優先され、静的パブリックキーは無視されます。

下位互換性

この機能は、完全な下位互換性を維持します。

- が指定されJwksUrlていない場合、または null に設定されている場合、システムは既存の静的パブリックキーメカニズムを使用します。
- 既存の JWT 設定は変更することなく引き続き機能します。
- ドメイン設定を更新することで、静的キーと JWKS を切り替えることができます。

JWKS エンドポイントの設定

OpenSearch Service コンソール、、または設定 API を使用して AWS CLI、JWKS エンドポイントを設定できます。

コンソール

JWKS エンドポイントを設定するには

1. ドメイン設定で、OpenSearch の JWT 認証と認可に移動します。
2. JWT 認証と認可を有効にするを選択します。
3. キーソースで、JWKS URL を選択します。
4. ID プロバイダーから提供された JWKS エンドポイント URL を入力します (例: `https://example.com/.well-known/jwks.json`)。
5. (オプション) 追加設定でサブジェクトキーとロールキーフィールドを設定します。
6. [Save changes] (変更の保存) をクリックします。

AWS CLI

次の AWS CLI コマンドは、既存のドメインで JWT 認証用の JWKS エンドポイントを設定します。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --advanced-security-options '{"JWTOptions":{"Enabled":true, "JwksUrl":"https://
example.com/.well-known/jwks.json", "SubjectKey":"sub", "RolesKey":"roles"}}'
```

Configuration API

設定 API への次のリクエストは、既存のドメインで JWT 認証用の JWKS エンドポイントを設定します。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
```

```
  "AdvancedSecurityOptions": {
    "JWTOptions": {
      "Enabled": true,
      "JwksUrl": "https://example.com/.well-known/jwks.json",
      "RolesKey": "optional-roles-key",
      "SubjectKey": "optional-subject-key"
    }
  }
}
```

JWT ヘッダーの要件

JWKS エンドポイントを使用する場合、JWT ヘッダーには、検証に使用する特定のキーを識別するキー ID (kid) を含める必要があります。

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "V-diposfUJIk5jDBFi_QRouiVinG5PowskcSWy5EuCo"
}
```

kid 値は、JWKS エンドポイントレスポンスのキー識別子と一致する必要があります。

JWKS 設定オプションとセキュリティ設定の詳細については、OpenSearch ドキュメントの「JWKS エンドポイントを使用して JWT を検証する」を参照してください。

JWT を使用してテストリクエストを送信する

指定されたサブジェクトとロールのペアを使用して新しい JWT を作成したら、テストリクエストを送信できます。これを行うには、プライベートキーを使用して、JWT を作成したツールを使用してリクエストに署名します。OpenSearch Service は、この署名を検証することで、受信リクエストを検証できます。

> **Note**
>
> JWT にカスタムサブジェクトキーまたはロールキーを指定した場合は、JWT に正しいクレーム名を使用する必要があります。

以下は、JWT トークンを使用してドメインの検索エンドポイントから OpenSearch Service にアクセスする方法の例です。

```
curl -XGET "$search_endpoint" -H "Authorization: Bearer <JWT>"
```

JWT 認証と承認の設定 (AWS CLI)

次の AWS CLI コマンドは、ドメインが存在する場合、OpenSearch の JWT 認証と認可を有効にします。

```
aws opensearch update-domain-config --domain-name <your_domain_name> --advanced-
security-options '{"JWTOptions":{"Enabled":true, "PublicKey": "<your_public_key>",
 "SubjectKey": "<your_subject_key>", "RolesKey": "<your_roles_key>"}}'
```

JWT 認証と承認の設定 (API を介した設定)

設定 API への次のリクエストにより、既存のドメイン上の OpenSearch の JWT 認証と承認を有効にします。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
  "AdvancedSecurityOptions": {
    "JWTOptions": {
      "Enabled": true,
      "PublicKey": "public-key",
      "RolesKey": "optional-roles-key",
      "SubjectKey": "optional-subject-key"
   }
  }
}
```

キーペアの生成

OpenSearch ドメインの JWT を設定するには、プライバシー強化メール (PEM) 形式でパブリックキーを指定する必要があります。Amazon OpenSearch Service は現在 JWT を使用する際に、RSA と ECDSA の 2 つの非対称暗号化アルゴリズムをサポートしています。

共通 OpenSSL ライブラリを使用して RSA キーペアを作成するには、次の手順に従います。

1. `openssl genrsa -out privatekey.pem 2048`
2. `openssl rsa -in privatekey.pem -pubout -out publickey.pem`

この例では、publickey.pem ファイルには Amazon OpenSearch Service で使用するパブリックキーが含まれ、privatekey.pem にはサービスに送信された JWT に署名するためのプライベートキーが含まれます。さらに、JWT を生成するためにプライベートキーが必要な場合は、プライベートキーを一般的に使用される pkcs8 形式に変換することもできます。

アップロードボタンを使用して PEM ファイルをコンソールに直接追加する場合、ファイルの拡張子は .pem である必要があります。.crt、.cert、または .key などの他のファイル拡張子は現在サポートされていません。

Amazon OpenSearch Service のインフラストラクチャセキュリティ

マネージドサービスとして、 は AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク経由で にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

AWS 公開された API コールを使用して、ネットワーク経由で OpenSearch Service 設定 API にアクセスします。許容する最低限必要な TLS バージョンを設定するには、ドメインエンドポイントオプションで TLSSecurityPolicy 値を次のように指定します。

```
aws opensearch update-domain-config --domain-name my-domain --domain-endpoint-options
 '{"TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"}'
```

詳細については、「AWS CLI コマンドリファレンス」を参照してください。

ドメイン設定に応じて、OpenSearch API へのリクエストに署名する必要があることもあります。詳細については、「the section called “OpenSearch Service リクエストの作成と署名”」を参照してください。

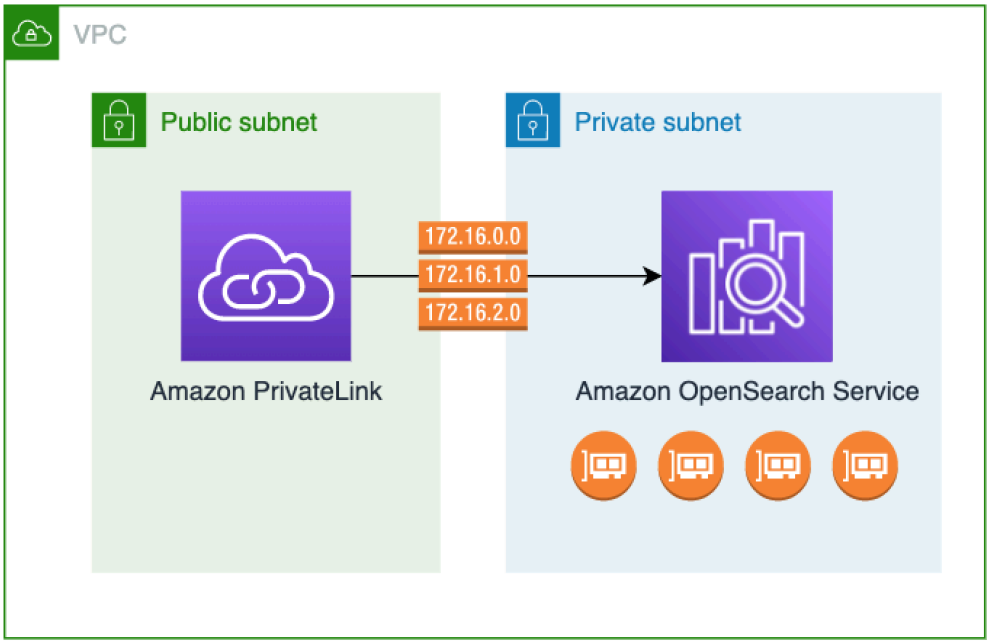
OpenSearch Service は、パブリックアクセスドメインをサポートしており、インターネットに接続された任意のデバイスからリクエストを受信できます。また、パブリックインターネットから分離されている VPC アクセスドメインもサポートしています。

VPC ドメインで VPC 出力オプションを有効にすると、OpenSearch Service はリクエスタ管理の ENIs をサブネットに配置して、ドメインからの出力トラフィックを伝送します。詳細については、「the section called “VPC を介したドメイン出力トラフィックのルーティング”」を参照してください。

OpenSearch Service で管理される VPC エンドポイントを使用して Amazon OpenSearch Service にアクセスする (AWS PrivateLink)

OpenSearch Service が管理する VPC エンドポイント (を使用) を設定することで、Amazon OpenSearch Service ドメインにアクセスできます AWS PrivateLink。これらのエンドポイントにより、VPC と Amazon OpenSearch Service との間にプライベート接続が作成されます。OpenSearch Service VPC ドメインには、インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのようにアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても OpenSearch Service にアクセスできます。

OpenSearch Service ドメインは、同じ VPC、異なる VPC、または異なる AWS アカウント内のパブリックサブネットまたはプライベートサブネットで実行されている追加のエンドポイントを公開するように設定できます。これにより、セキュリティレイヤーを追加して、どこで実行されているかにかかわらずドメインにアクセスできます。インフラストラクチャを管理する必要はありません。次の図は、同じ VPC 内の OpenSearch Service で管理される VPC エンドポイントを示しています。



このプライベート接続を確立するには、OpenSearch Service で管理されるインターフェイス VPC エンドポイント (AWS PrivateLinkを使用) を作成します。インターフェイス VPC エンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスが作成されます。これらは OpenSearch Service に送信されるトラフィックのエントリポイントとして機能する、サービス管理型のネットワークインターフェイスです。 AWS PrivateLinkで課金される OpenSearch Service 管理型 VPC エンドポイントには、標準 AWS PrivateLink インターフェイスエンドポイント料金が適用されます。

OpenSearch と従来の Elasticsearch のすべてのバージョンを実行しているドメインに対して VPC エンドポイントを作成できます。詳細については「AWS PrivateLink ガイド」の「Access AWS のサービス through AWS PrivateLink」を参照してください。

OpenSearch Service に関する考慮事項と制約事項

OpenSearch Service のインターフェイス VPC エンドポイントを設定する前に、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

OpenSearch Service で管理される VPC エンドポイントを使用する場合は、次の点を考慮してください。

- VPC ドメインへの接続には、インターフェイス VPC エンドポイントのみを使用できます。パブリックドメインはサポートされません。
- VPC エンドポイントは、同じ AWS リージョン内のドメインにのみ接続できます。
- VPC エンドポイントでサポートされているプロトコルは HTTPS のみです。HTTP は許可されていません。
- OpenSearch Service は、すべてのサポートされている OpenSearch API オペレーションをインターフェイス VPC エンドポイント経由で呼び出すことをサポートしています。
- アカウントごとに最大 50 個のエンドポイント、ドメインごとに最大 10 個のエンドポイントを設定できます。1 つのドメインに含めることができる認証済みプリンシパルの数は、最大 10 です。
- 現在、 AWS CloudFormation を使用してインターフェイス VPC エンドポイントを作成することはできません。
- インターフェイス VPC エンドポイントは、OpenSearch Service コンソールまたは OpenSearch Service API を使用してのみ作成できます。OpenSearch Service 用のインターフェイス VPC エンドポイントを、Amazon VPC コンソールを使用して作成することはできません。
- OpenSearch Service で管理される VPC エンドポイントには、インターネットからアクセスできません。OpenSearch Service で管理される VPC エンドポイントは、エンドポイントがプロビ

ジョニングされている VPC 内か、エンドポイントがプロビジョニングされている VPC とピア接続されている (ルートテーブルとセキュリティグループによって許可されている) VPC 内でのみアクセスできます。

- OpenSearch Service では、VPC エンドポイントポリシーがサポートされません。セキュリティグループをエンドポイントのネットワークインターフェイスに関連付けて、インターフェイス VPC エンドポイント経由で OpenSearch Service へのトラフィックを制御できます。
- サービスにリンクされたロールは、VPC エンドポイントの作成に使用するのと同じ AWS アカウントにある必要があります。
- OpenSearch Service VPC エンドポイントを作成、更新、削除するには、Amazon OpenSearch Service のアクセス許可に加えて、次の Amazon EC2 アクセス許可が必要です。
  - `ec2:CreateVpcEndpoint`
  - `ec2:DescribeVpcEndpoints`
  - `ec2:ModifyVpcEndpoint`
  - `ec2:DeleteVpcEndpoints`
  - `ec2:CreateTags`
  - `ec2:DescribeTags`
  - `ec2:DescribeSubnets`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeVpcs`

> **Note**
>
> 現在、VPC エンドポイントの作成を OpenSearch Service に限定することはできません。今後のアップデートでこれを可能にするよう取り組んでいます。

ドメインへのアクセスを提供する

ドメインにアクセスする VPC が別の にある場合は AWS アカウント、インターフェイス VPC エンドポイントを作成する前に、所有者のアカウントからドメインを許可する必要があります。

別の の VPC AWS アカウント がドメインにアクセスできるようにするには

1. https://console.aws.amazon.com/aos/home/ で Amazon OpenSearch Service コンソールを開きます。

2. ナビゲーションペインで、[Domains] (ドメイン) を選択し、アクセス権を付与するドメインを開きます。
3. [VPC endpoints] (VPC エンドポイント) タブに移動すると、ドメインにアクセスできるアカウントと対応する VPC が表示されます。
4. [Authorize principal] (プリンシパルを承認) を選択します。
5. ドメインにアクセスするアカウントの AWS アカウント ID を入力します。このステップでは、指定されたアカウントがドメインに対して VPC エンドポイントを作成することを承認します。
6. [承認] を選択します。

VPC ドメインのインターフェイス VPC エンドポイントを作成する

OpenSearch Service コンソールまたは AWS Command Line Interface () を使用して、OpenSearch Service のインターフェイス VPC エンドポイントを作成できますAWS CLI。

OpenSearch Service ドメイン用のインターフェイス VPC エンドポイントを作成するには

1. https://console.aws.amazon.com/aos/home/ で Amazon OpenSearch Service コンソールを開きます。
2. 左のナビゲーションペインで [VPC endpoints] (VPC エンドポイント) を選択します。
3. エンドポイントの作成 を選択します。
4. 現在または AWS アカウント 別の でドメインを接続するかどうかを選択します AWS アカウント。
5. このエンドポイントで接続するドメインを選択します。ドメインが現在の にある場合は AWS アカウント、ドロップダウンを使用してドメインを選択します。ドメインが別のアカウントにある場合は、接続するドメインの Amazon リソースネーム (ARN) を入力します。別のアカウントのドメインを選択するには、所有者にドメインへのアクセス権を付与してもらう必要があります。
6. [VPC] で、OpenSearch Service へのアクセス元の VPC を選択します。
7. [Subnets] (サブネット) で、OpenSearch Service へのアクセス元のサブネットを 1 つまたは複数選択します。
8. [Security groups] (セキュリティグループ) で、エンドポイントネットワークインターフェイスに関連付けるセキュリティグループを選択します。これは、エンドポイントに対して承認するインバウンドトラフィックのポート、プロトコル、およびソースを制限する重要なステップです。セキュリティグループルールでは、OpenSearch Service と通信してエンドポイントのネットワー

クインターフェイスと通信するために、VPC エンドポイントを使用するリソースを許可する必要があります。

9. エンドポイントの作成 を選択します。エンドポイントは 2～5 分でアクティブになるはずです。

設定 API を使用した OpenSearch Service で管理される VPC エンドポイントの操作

次の API オペレーションを使用して、OpenSearch Service で管理される VPC エンドポイントを作成および管理します。

- CreateVpcEndpoint
- ListVpcEndpoints
- UpdateVpcEndpoint
- DeleteVpcEndpoint

VPC ドメインへのエンドポイントアクセスを管理するには、次の API オペレーションを使用します。

- AuthorizeVpcEndpointAccess
- ListVpcEndpointAccess
- ListVpcEndpointsForDomain
- RevokeVpcEndpointAccess

OpenSearch Dashboards の SAML 認証

OpenSearch Dashboards の SAML 認証を使用すると、既存の ID プロバイダーを使用して、OpenSearch または Elasticsearch 6.7 以降を実行している Amazon OpenSearch Service ドメインの Dashboards Single Sign-On (SSO) を提供できます。SAML 認証を使用するには、きめ細かなアクセスコントロールを有効にする必要があります。

Amazon Cognito または内部ユーザーデータベースを介して認証するのではなく、OpenSearch Dashboards の SAML 認証により、サードパーティーの ID プロバイダーを使用して Dashboards へのログイン、詳細なアクセスコントロールの管理、データの検索、視覚化の構築を行うことができます。OpenSearch Service は、Okta、Keycloak、Microsoft Entra ID、Active Directory フェデレーションサービス (ADFS)、Auth0、 などの SAML 2.0 標準を使用するプロバイダーをサポートしています AWS IAM アイデンティティセンター。

Dashboards の SAML 認証は、ウェブブラウザから OpenSearch Dashboards にアクセスするためのものです。お客様のSAML認証情報では、OpenSearch または Dashboards API への直接 HTTP リクエストを行うことはできません。

SAML 設定の概要

このドキュメントは、ユーザーに既存の ID プロバイダーがあり、そのプロバイダーについてある程度の知識があることを前提としています。OpenSearch Service ドメインについてのみ、お客様のプロバイダーの詳細な設定手順は提供できません。

OpenSearch Dashboards のログインフローは、次の 2 つの形式のいずれかになります。

- サービスプロバイダー (SP) が開始されている: Dashboards に移動します (例えば、`https://my-domain.us-east-1.es.amazonaws.com/_dashboards`)。これにより、ログイン画面にリダイレクトされます。ログイン後、ID プロバイダーはお客様を Dashboards にリダイレクトします。
- ID プロバイダー (IdP) 開始: ID プロバイダーに移動してログインし、アプリケーションディレクトリから OpenSearch Dashboards を選択します。

OpenSearch Service には、SP 開始と IdP 開始の 2 つの Single Sign-On URL が用意されていますが、目的の OpenSearch Dashboards ログインフローに一致するもののみが必要です。

どの認証タイプを使用するかにかかわらず、目的は ID プロバイダーを介してログインし、ユーザーネーム (必須) とバックエンドロール (オプションですが、推奨されます) を含む SAML アサーションを受け取ることです。この情報により、きめ細かなアクセスコントロールが、SAML ユーザーに許可を割り当てることができます。外部 ID プロバイダーでは、バックエンドロールは通常「ロール」または「グループ」と呼ばれます。

考慮事項

SAML 認証を設定するときは、以下を考慮してください。

- IdP メタデータファイルのサイズにより、SAML 認証の設定には AWS コンソールを使用することを強くお勧めします。
- ドメインは、一度に 1 つの Dashboards 認証方法のみをサポートします。OpenSearch Dashboards の Amazon Cognito 認証が有効になっている場合は、SAML 認証を有効にする前に無効化しておく必要があります。

- SAML で Network Load Balancer を使用する場合は、最初にカスタムエンドポイントを作成します。詳細については、「???」を参照してください。
- サービスコントロールポリシー (SCP) は、IAM 以外の ID (Amazon OpenSearch Serverless & SAML における SAML、あるいは Amazon OpenSearch Service の 基本的な内部ユーザー許可など) の場合には適用または評価されません。

VPC ドメインの SAML 認証

SAML は、アイデンティティプロバイダーとサービスプロバイダー間の直接的な通信を必要としません。したがって、OpenSearch ドメインがプライベート VPC 内でホストされている場合でも、ブラウザが OpenSearch クラスターとアイデンティティプロバイダーの両方と通信できる限り、SAML を使用できます。ブラウザは、基本的にアイデンティティプロバイダーとサービスプロバイダーの仲介として機能します。SAML 認証フローを説明する便利な図表については、Okta のドキュメントを参照してください。

ドメインアクセスポリシーの変更

SAML 認証を設定する前に、ドメインアクセスポリシーを更新して SAML ユーザーがドメインにアクセスできるようにする必要があります。これを実行しない場合は、アクセス拒否エラーが表示されます。

次のドメインアクセスポリシーをお勧めします。これにより、ドメイン上のサブリソース (/*) にフルアクセスが付与されます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:ESHttp*",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

ポリシーをより制限的にするには、ポリシーに IP アドレス条件を追加できます。この条件は、指定された IP アドレス範囲またはサブネットにのみアクセスを制限します。例えば、次のポリシーでは、192.0.2.0/24 サブネットからのアクセスのみを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24"
          ]
        }
      },
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

> **Note**
>
> オープンドメインアクセスポリシーでは、ドメインできめ細かなアクセスコントロールを有効にする必要があります。有効にしないと、次のエラーが表示されます。
> `To protect domains with public access, a restrictive policy or fine-grained access control is required.`
> マスターユーザーまたは内部ユーザーが堅牢なパスワードで設定されている場合、きめ細かなアクセスコントロールの使用中にポリシーを開いたままにしておいても、セキュリティの観点から許容できることもあります。詳細については、「???」を参照してください。

SP 開始認証または IdP 開始認証の設定

以下のステップでは、OpenSearch Dashboards の SP 開始認証または IdP 開始認証を使用して SAML 認証を有効にする方法について説明します。両方を有効にするために必要な追加のステップについては、「SP 開始認証と IdP 開始認証の両方を有効にする」を参照してください。

ステップ 1: SAML 認証を有効にする

SAML 認証は、ドメインの作成中に有効化、または既存のドメインで [Actions] (アクション)、[Edit security configuration] (セキュリティ設定の編集) の順に選択することで有効化できます。以下のステップは、どちらを選択するかに応じて若干異なります。

ドメイン設定内の [SAML authentication for OpenSearch Dashboards/Kibana] (OpenSearch Dashboards/Kibana 用の SAML 認証) で [Enable SAML authentication] (SAML 認証を有効化) を選択します。

ステップ 2: アイデンティティプロバイダーを設定する

SAML 認証をいつ設定しているかに応じて、以下のステップを実行します。

新しいドメインを作成している場合

新しいドメインを作成中の場合、OpenSearch Service はまだサービスプロバイダーのエンティティ ID または SSO URL を生成できません。アイデンティティプロバイダーは、SAML 認証を適切に有効化するためにこれらの値を必要としますが、これらの値はドメインの作成後しか生成できません。ドメインの作成中にこの相互依存性を回避するには、IdP 設定に一時的な値を指定して必要なメタデータを生成し、ドメインがアクティブになった後でそれらを更新することができます。

カスタムエンドポイントを使用している場合は、URL がどうなるかを推測できます。例えば、カスタムエンドポイントが `www.custom-endpoint.com` の場合、サービスプロバイダーのエンティティ ID は `www.custom-endpoint.com`、IdP 開始の SSO URL は `www.custom-endpoint.com/_dashboards/_opendistro/_security/saml/acs/idpinitiated`、SP 開始の SSO URL は `www.custom-endpoint.com/_dashboards/_opendistro/_security/saml/acs` になります。ドメインを作成される前に、これらの値を使用して ID プロバイダーを設定できます。例については、次のセクションを参照ください。

> **Note**
>
> HTTP リクエストの FQDN は SAML リクエストの FQDN とは異なるため、デュアルスタックエンドポイントでサインインすることはできません。OpenSearch 管理者は、デュアルス

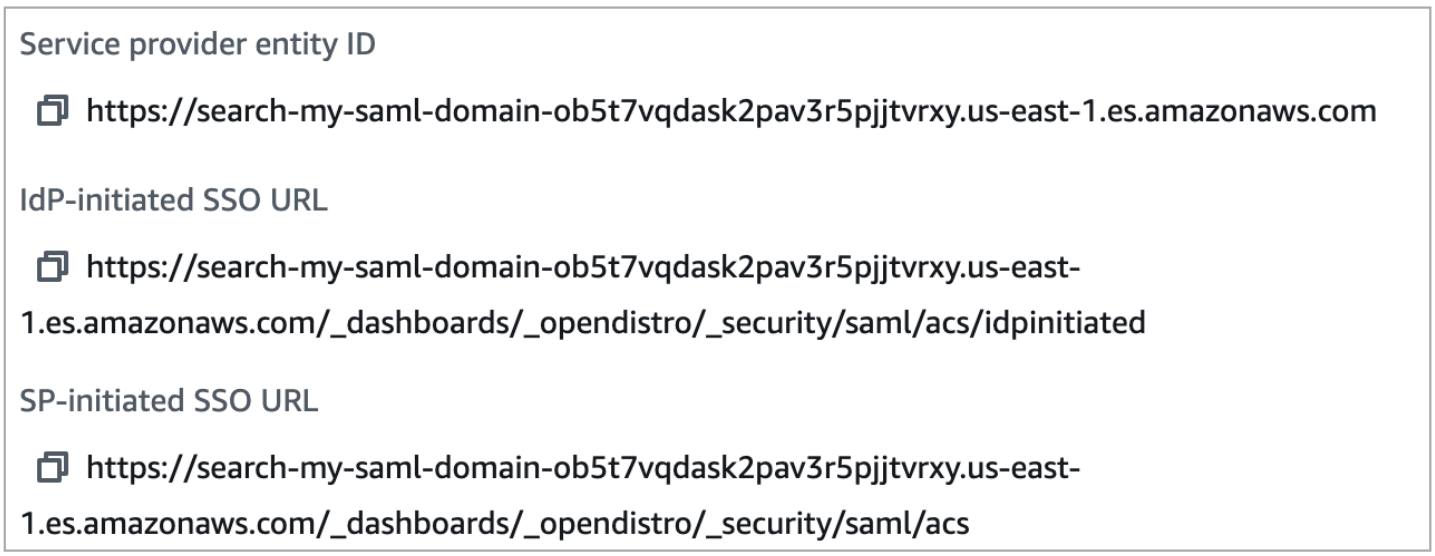
タックエンドポイントを使用してサインインする場合は、カスタムエンドポイントを設定し、CNAME 値をデュアルスタックエンドポイントに設定する必要があります。

カスタムエンドポイントを使用していない場合は、IdP に一時的な値を入力して必要なメタデータを生成し、ドメインがアクティブになった後でそれらを更新することができます。

例えば、Okta では、[Single sign on URL] (シングルサインオン URL) と [Audience URI (SP Entity ID)] (オーディエンス URI (SP エンティティ ID) フィールドに `https://temp-endpoint.amazonaws.com` を入力できます。そうすることで、メタデータの生成が可能になります。その後、ドメインがアクティブになったら、OpenSearch Service から正しい値を取得して、Okta でそれらを更新できます。手順については、「the section called “ステップ 6: IdP URL を更新する”」を参照してください。

既存のドメインを編集している場合

既存のドメインで SAML 認証を有効化している場合は、サービスプロバイダーのエンティティ ID と SSO URL の 1 つをコピーします。使用する URL のガイダンスについては、「the section called “SAML 設定の概要”」を参照してください。



これらの値を使用して、アイデンティティプロバイダーを設定します。これはプロセスの最も複雑な部分ですが、残念ながら、用語とステップはプロバイダーによって大きく異なります。プロバイダーのドキュメントを参照してください。

例えば Okta では、SAML 2.0 ウェブアプリケーションを作成します。[Single sign on URL] (シングルサインオン URL) には、SSO URL を指定します。Audience URI (SP エンティティ ID) では、SP エンティティ ID を指定します。

Okta には、ユーザーおよびバックエンドロールではなく、ユーザーとグループがあります。[Group Attribute Statements] (グループ属性ステートメント) では、`role` を [名前] フィールドに、正規表現 `.+` を [フィルター] フィールドに追加することをお勧めします。このステートメントは、Okta の ID プロバイダーに、ユーザーの認証後に SAML アサーションの `role` フィールドの下に、すべてのユーザーグループを含めるよう指示します。

IAM アイデンティティセンターで、SP エンティティ ID をアプリケーション SAML オーディエンスとして指定します。また、次の属性マッピングも指定する必要があります: `Subject=${user:subject}:format=unspecified` と `Role=${user:groups}:format=uri`

Auth0 では、通常のウェブアプリケーションを作成し、SAML 2.0 アドオンを有効にします。Keycloak では、クライアントを作成します。

ステップ 3: IdP メタデータをインポートする

ID プロバイダーを設定すると、IdP メタデータファイルが生成されます。この XML ファイルには、TLS 証明書、Single Sign-On エンドポイント、ID プロバイダーのエンティティ ID など、プロバイダーに関する情報が含まれています。

IdP メタデータファイルの内容をコピーし、OpenSearch Service コンソールの [IdP からのメタデータ] フィールドにそれを貼り付けます。または、[XML ファイルからインポート] を選択し、ファイルをアップロードします。メタデータファイルは、次のように表示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="entity-id"
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  <md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>tls-certificate</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</
md:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</
md:NameIDFormat>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
 Location="idp-sso-url"/>
```

```
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="idp-sso-url"/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

ステップ 4: SAML フィールドを設定する

IdP メタデータを入力したら、OpenSearch Service コンソールで以下の追加フィールドを設定します。

- [IdP entity ID] (IdP エンティティ ID) – メタデータファイルから `entityID` プロパティの値をコピーし、このフィールドにそれを貼り付けます。多くの ID プロバイダーは、この値も設定後の概要の一部として表示します。一部のプロバイダーは、それを「発行者」と呼んでいます。

  > **Note**
  >
  > IdP エンティティ ID 値は URL 形式である必要があります (例: `https://idp.example.com/...`)。プレーン文字列 (JumpCloud」など) などの URL 以外の値では、500 エラーが発生します。

- [SAML master username] (SAML マスターユーザー名) と [SAML master backend role] (SAML マスターバックエンドロール) – 指定するユーザーおよび/またはロールには、クラスターに対する完全な許可が付与されます。これは新しいマスターユーザーと同等ですが、これらの許可を OpenSearch Dashboards 外で使用することはできません。

  例えば、Okta では、グループ `admins` に属しているユーザー `jdoe` がある可能性があります。`jdoe` を [SAML マスターユーザーネーム] フィールドに追加した場合、そのユーザーのみが完全な許可を受け取ります。`admins` を SAML マスターバックエンドロールフィールドに追加する場合は、`admins` グループに属するすべてのユーザーに完全な許可が付与されます。

  > **Note**
  >
  > SAML アサーションの内容は、SAML マスターユーザーネームおよび SAML マスターロールに使用する文字列と正確に一致する必要があります。一部の ID プロバイダーは、ユーザーネームの前にプレフィックスを追加するので、診断が難しいミスマッチを引き起こす可能性があります。ID プロバイダーのユーザーインターフェイスで、`jdoe` を確認できる場合がありますが、SAML アサーションには `auth0|jdoe` が含まれている可能性があります。常に SAML アサーションからの文字列を使用します。

多くの ID プロバイダーでは、設定プロセス中にサンプルのアサーションを表示できます。また、SAML トレーサーは、実際のアサーションの内容を調べてトラブルシューティングするのに役立ちます。アサーションは次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2:Assertion ID="id67229299299259351343340162"
 IssueInstant="2020-09-22T22:03:08.633Z" Version="2.0"
  xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">idp-issuer</
saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">username</saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData NotOnOrAfter="2020-09-22T22:08:08.816Z"
 Recipient="domain-endpoint/_dashboards/_opendistro/_security/saml/acs"/>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2020-09-22T21:58:08.816Z"
 NotOnOrAfter="2020-09-22T22:08:08.816Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>domain-endpoint</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AuthnStatement AuthnInstant="2020-09-22T19:54:37.274Z">
    <saml2:AuthnContext>

 <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport<
saml2:AuthnContextClassRef>
    </saml2:AuthnContext>
  </saml2:AuthnStatement>
  <saml2:AttributeStatement>
    <saml2:Attribute Name="role" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
      <saml2:AttributeValue
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xs:string">GroupName Match Matches regex ".+" (case-sensitive)
      </saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
</saml2:Assertion>
```

ステップ 5: (オプション) 追加設定を実行する

[Additional settings] (その他の設定) で、次のオプションフィールドを設定します。

- [Subject key] (件名キー) – このフィールドを空のままにしておいて、ユーザーネームの SAML アサーションの `NameID` エレメントを使用することができます。アサーションでこの標準エレメントを使用せず、代わりにユーザーネームをカスタム属性として含める場合は、ここでその属性を指定します。
- [Roles key] (ロールキー) – バックエンドロール (推奨) を使用する場合は、このフィールドでアサーションからの属性 (`role` または `group` など) を指定します。これは、SAML トレーサーなど、どのツールが役立つかという別の状況です。
- [Session time to live] (セッションの有効時間) – デフォルトで、OpenSearch Dashboards は 24 時間後にユーザーをログアウトします。この値は、新しい値を指定することで、60 から 1,440 (24 時間) までの任意の数値に設定できます。

設定に問題がなければ、ドメインを保存します。

ステップ 6: IdP URL を更新する

ドメインの作成中に SAML 認証を有効化した場合は、XML メタデータファイルを生成するために IdP 内で一時的な URL を指定する必要がありました。ドメインステータスが `Active` に変わったら、正しい URL を取得して IdP を変更できます。

URL を取得するには、ドメインを選択してから、[Actions] (アクション)、[Edit security configuration] (セキュリティ設定の編集) の順に選択します。[SAML authentication for OpenSearch Dashboards/Kibana] (OpenSearch Dashboards/Kibana 用の SAML 認証) で、正しいサービスプロバイダーのエンティティ ID と SSO URL を見つけることができます。値をコピーし、それらを使用してアイデンティティプロバイダーを設定することで、ステップ 2 で指定した一時的な URL を置き換えます。

ステップ 7: SAML ユーザーをロールにマップする

ドメインのステータスがアクティブになり、IdP が正しく設定されたら、OpenSearch Dashboards に移動します。

- SP 開始の URL を選択した場合は、*`domain-endpoint`*`/_dashboards` に移動します。特定のテナントに直接ログインするには、URL に `?security_tenant=`*`tenant-name`* を追加できます。

- IdP 開始の URL を選択した場合は、ID プロバイダーのアプリケーションディレクトリに移動します。

どちらの場合も、SAML マスターユーザーまたは SAML マスターバックエンドロールに属しているユーザーとしてログインします。ステップ 7 からの例を続けるには、jdoe、または admins グループのメンバーとしてログインします。

OpenSearch Dashboards がロードされたら、[Security] (セキュリティ)、[Roles] (ロール) の順に選択します。次に、ロールをマップして、他のユーザーが OpenSearch Dashboards にアクセスできるようにします。

例えば、信頼できる同僚 jroe を all_access および security_manager ロールにマッピングします。また、バックエンドロール analysts を readall および opensearch_dashboards_user ロールにマッピングすることもできます。

OpenSearch Dashboards の代わりに API を使用したい場合は、以下のサンプルリクエストを参照してください。

```
PATCH _plugins/_security/api/rolesmapping
[
  {
    "op": "add", "path": "/security_manager", "value": { "users": ["master-user",
 "jdoe", "jroe"], "backend_roles": ["admins"] }
  },
  {
    "op": "add", "path": "/all_access", "value": { "users": ["master-user", "jdoe",
 "jroe"], "backend_roles": ["admins"] }
  },
  {
    "op": "add", "path": "/readall", "value": { "backend_roles": ["analysts"] }
  },
  {
    "op": "add", "path": "/opensearch_dashboards_user", "value": { "backend_roles":
 ["analysts"] }
  }
]
```

SP 開始認証と IdP 開始認証両方の設定

SP 開始認証と IdP 開始認証の両方を設定する場合は、ID プロバイダーを通じて設定する必要があります。例えば、Okta では、次のステップを実行できます。

1. SAML アプリケーション内で、[General] (全般)、[SAML settings] (SAML 設定) に移動します。
2. [シングルサインオン URL] で、IdP 開始 SSO URL を指定します。例えば、`https://search-domain-hash/_dashboards/_opendistro/_security/saml/acs/idpinitiated`。
3. [Allow this app to request other SSO URLs] (このアプリが他の SSO URL をリクエストすることを許可) を有効にします。
4. [リクエスト可能な SSO URL] で、1 つ以上の SP 開始 SSO URL を追加します。例えば、`https://search-domain-hash/_dashboards/_opendistro/_security/saml/acs`。

SAML 認証の設定 (AWS CLI)

次の AWS CLI コマンドは、既存のドメインで OpenSearch Dashboards の SAML 認証を有効にします。

```
aws opensearch update-domain-config \
  --domain-name my-domain \
  --advanced-security-options '{"SAMLOptions":{"Enabled":true,"MasterUserName":"my-idp-user","MasterBackendRole":"my-idp-group-or-role","Idp":{"EntityId":"entity-id","MetadataContent":"metadata-content-with-quotes-escaped"},"RolesKey":"optional-roles-key","SessionTimeoutMinutes":180,"SubjectKey":"optional-subject-key"}}'
```

メタデータ XML では、すべての引用符と改行文字をエスケープする必要があります。例えば、`<KeyDescriptor use="signing">` および改行の代わりに `<KeyDescriptor use=\"signing\">\n` を使用します。 AWS CLIの使用方法の詳細については、「AWS CLI コマンドリファレンス」を参照してください。

SAML 認証の設定 (設定 API)

設定 API への次のリクエストにより、既存のドメイン上の OpenSearch Dashboards の SAML 認証を有効にします。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
  "AdvancedSecurityOptions": {
    "SAMLOptions": {
      "Enabled": true,
      "MasterUserName": "my-idp-user",
```

```
      "MasterBackendRole": "my-idp-group-or-role",
      "Idp": {
        "EntityId": "entity-id",
        "MetadataContent": "metadata-content-with-quotes-escaped"
      },
      "RolesKey": "optional-roles-key",
      "SessionTimeoutMinutes": 180,
      "SubjectKey": "optional-subject-key"
    }
  }
}
```

メタデータ XML では、すべての引用符と改行文字をエスケープする必要があります。例えば、`<KeyDescriptor use="signing">` および改行の代わりに `<KeyDescriptor use=\"signing\">\n` を使用します。設定 API の使用方法の詳細については、「OpenSearch Service API Reference」(OpenSearch Service API リファレンス) を参照してください。

SAML のトラブルシューティング

エラー	詳細
リクエスト: '*/some/path* ' は許可されていません。	正しい SSO URL (ステップ 3) を ID プロバイダーに提供したことを確認します。
SAML を有効にするには、有効な ID プロバイダーメタデータドキュメントを指定してください。	IdP メタデータファイルは SAML 2.0 標準に準拠していません。検証ツールを使用してエラーをチェックします。
SAML 設定オプションは、コンソールでは表示されません。	最新のサービスソフトウェアに更新します。
SAML 設定エラー: SAML 設定を取得中に問題が発生しました。設定を確認してください。	この一般的なエラーは、さまざまな原因で発生することがあります。 • ID プロバイダーに正しい SP エンティティ ID と SSO URL が指定されていることを確認します。 • IdP メタデータファイルを再生成し、IdP エンティティ ID を確認します。更新されたメタデータを AWS コンソールに追加します。

エラー	詳細
	• ドメインアクセスポリシーで、OpenSearch Dashboards と `_plugins/_security/*` へのアクセスが許可されていることを確認します。一般的に、きめ細かなアクセスコントロールを使用するドメインではオープンアクセスポリシーを使用することをお勧めします。 • SAML の設定のステップについては、ID プロバイダーのドキュメントを参照してください。
ロールがありません: このユーザーには使用できるロールがありません。システム管理者に連絡してください。	認証に成功しましたが、ユーザーネームおよび SAML アサーションからのバックエンドロールはどのロールにもマッピングされていないため、許可がありません。これらのマッピングでは、大文字と小文字が区別されます。 システム管理者は、SAML トレーサーのようなツールを使用して、SAML アサーションの内容を確認し、次のリクエストを使用してロールマッピングを確認します。 `GET _plugins/_security/api/rolesmapping`
OpenSearch Dashboards にアクセスしようとすると、ブラウザが HTTP 500 エラーを継続的にリダイレクトまたは受け取ります。	このエラーは、SAML アサーションに合計約 1,500 文字の多数のロールが含まれている場合に発生します。例えば、平均の長さが 20 文字である 80 ロールを渡すと、ウェブブラウザの Cookie のサイズ制限を超える可能性があります。OpenSearch バージョン 2.7 以降、SAML アサーションは最大 5,000 文字のロールをサポートします。

エラー	詳細
ADFS から ログアウトすることはできません。	ADFS では、OpenSearch Service がサポートしていないすべてのログアウトリクエストに署名する必要があります。IdP メタデータファイルから `<SingleLogoutService />` を削除して、OpenSearch Service が独自の内部ログアウトメカニズムを使用するように強制します。
`Could not find entity descriptor for __PATH__.`	OpenSearch Service へのメタデータ XML で提供される IdP のエンティティ ID が、SAML レスポンスのエンティティ ID とは異なります。これを修正するには、それらが一致していることを確認してください。ドメインで [CW アプリケーションエラーログ] を有効にして、SAML 統合の問題をデバッグするためのエラーメッセージを見つけます。
`Signature validation failed. SAML response rejected.`	OpenSearch Service は、メタデータ XML で提供された IdP の証明書を使用して SAML レスポンスの署名を検証できません。これは手動エラーであるか、または IdP が証明書をローテーションした可能性があります。 AWS マネジメントコンソール を通じて OpenSearch Service に提供されるメタデータ XML 内の IdP からの最新の証明書を更新します。
`__PATH__ is not a valid audience for this response.`	SAML レスポンスのオーディエンスフィールドがドメインエンドポイントと一致しません。このエラーを修正するには、ドメインエンドポイントと一致するように SP オーディエンスフィールドを更新します。カスタムエンドポイントを有効にしている場合は、オーディエンスフィールドがカスタムエンドポイントと一致する必要があります。ドメインで [CW アプリケーションエラーログ] を有効にして、SAML 統合の問題をデバッグするためのエラーメッセージを見つけます。

エラー	詳細
ブラウザは、レスポンスで Invalid Request Id ともに HTTP 400 エラーを受け取ります。	このエラーは通常、IdP 開始 URL を形式 *<DashboardsURL>* /_opendistro/_security/saml/acs で設定した場合に発生します。代わりに、URL を形式 *<DashboardsURL>* /_opendistro/_security/saml/acs/idpinitiated で設定します。
レスポンスは __PATH__ の代わりに __PATH__ で受信されました。	SAML レスポンスの宛先フィールドが次の URL 形式のいずれかと一致しません。 • *<DashboardsURL>* /_opendistro/_security/saml/acs • *<DashboardsURL>* /_opendistro/_security/saml/acs/idpinitiated . 使用するログインフロー (SP 開始または IdP 開始) に応じて、OpenSearch URL のいずれかと一致する宛先フィールドに入力します。
レスポンスには InResponseTo 属性がありますが、InResponseTo 属性は想定されていません。	SP 開始ログインフローで IdP 開始 URL を使用しています。代わりに、SP 開始 URL を使用してください。

SAML 認証の無効化

OpenSearch Dashboards の SAML 認証を無効にするには (コンソール)

1. ドメインを選択し、[アクション] から [セキュリティ設定の編集] を選択します。
2. [SAML 認証を有効にする] のチェックを外します。
3. [Save changes] (変更の保存) をクリックします。
4. ドメインの処理が終了したら、次のリクエストを用いてきめ細かなアクセスコントロールのロールマッピングを確認します。

```
GET _plugins/_security/api/rolesmapping
```

Dashboards の SAML 認証を無効にしても、SAML マスターユーザーネームおよび/または SAML マスターバックエンドロールのマッピングを削除しません。これらのマッピングを削除する場合は、内部ユーザーデータベース (有効な場合) を使用して Dashboards にログインするか、API を使用してそれらを削除します。

```
PUT _plugins/_security/api/rolesmapping/all_access
{
  "users": [
    "master-user"
  ]
}
```

IAM アイデンティティセンターによる OpenSearch に対する信頼できる ID の伝達のサポート

Trusted Identity Propagation を介して一元的に設定された AWS IAM Identity Center プリンシパル (ユーザーとグループ) を使用して、OpenSearch Service アプリケーション を介して Amazon OpenSearch OpenSearchドメインにアクセスできるようになりました。OpenSearch の IAM アイデンティティセンターのサポートを有効にするには、IAM アイデンティティセンターの使用を有効にする必要があります。これを行う方法の詳細については、「What is IAM Identity Center?」を参照してください。その他の詳細については、「How to associate OpenSearch domain as datasource in OpenSearch applications?」を参照してください。

OpenSearch Service コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDKs を使用して、IAM Identity Center を設定できます。

> **Note**
>
> IAM アイデンティティセンタープリンシパルは ダッシュボード (クラスターと共存) ではサポートされていません。これらは、一元化された OpenSearch ユーザーインターフェイス (ダッシュボード) でのみサポートされています。

考慮事項

Amazon OpenSearch Service で IAM アイデンティティセンターを使用する前に、以下をチェックする必要があります。

- IAM アイデンティティセンターがアカウントで有効になっている。
- ご利用のリージョンで IAM アイデンティティセンターが利用可能である。
- OpenSearch のドメインバージョンは 1.3 以降である。
- きめ細かなアクセスコントロールがドメインで有効になっている。
- ドメインは IAM アイデンティティセンターインスタンスと同じリージョンにある。
- ドメインと OpenSearch アプリケーションは同じ AWS アカウントに属します。

ドメインアクセスポリシーの変更

IAM アイデンティティセンターを設定する前に、ドメインアクセスポリシーまたは OpenSearch アプリケーションで設定された IAM ロールのアクセス許可を、信頼できる ID の伝達用に更新する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/OpenSearchRole"
            },
            "Action": "es:ESHttp*",
            "Resource": "arn:aws:es:us-east-1:111122223333:domain/example-domain/
*"
        }
    ]
}
```

IAM アイデンティティセンターの認証と認可の設定 (コンソール)

ドメイン作成プロセス時、または既存のドメインを更新することで、IAM アイデンティティセンターの認証と認可を有効にすることができます。セットアップ手順は、選択するオプションに応じて若干異なります。

次の手順では、Amazon OpenSearch Service コンソールで IAM アイデンティティセンターの認証と許可用に既存のドメインを設定する方法について説明します。

1. [ドメイン設定]で、[セキュリティ設定] に移動し、[編集] を選択して IAM アイデンティティセンター認証セクションに移動し、[IAM アイデンティティセンターで認証された API アクセスを有効にする] を選択します。
2. 次のようにサブジェクトキーとロールキーを選択します。

   - [サブジェクトキー] – UserId (デフォルト)、UserName、E メールのいずれかを選択して、対応する属性をドメインにアクセスするプリンシパルとして使用します。
   - ロールキー - GroupId (デフォルト) と GroupName のいずれかを選択して、対応する属性値を IAM アイデンティティセンタープリンシパルに関連付けられたすべてのグループのfine-grained-access-controlのバックエンドロールとして使用します。

変更を行ったら、ドメインを保存します。

きめ細かなアクセスコントロールの設定

OpenSearch ドメインで IAM アイデンティティセンターオプションを有効にしたら、バックエンドロールへのロールマッピングを作成して、IAM アイデンティティセンタープリンシパルへのアクセスを設定できます。プリンシパルのバックエンドロール値は、IAM Identity Center プリンシパルのグループメンバーシップとGroupId または GroupName の RolesKey 設定に基づいています。

> **Note**
>
> Amazon OpenSearch Service は、単一のユーザーに対して最大 100 グループをサポートします。許可されたインスタンスの数を超えて使用しようとすると、きめ細かなアクセスコントロールの認可処理に不整合が発生し、403 エラーメッセージが表示されます。

IAM アイデンティティセンターの認証と認可の設定 (CLI)

```
  aws opensearch update-domain-config \
      --domain-name my-domain \
      --identity-center-options '{"EnabledAPIAccess": true,
 "IdentityCenterInstanceARN": "instance arn",  "SubjectKey": "UserId/UserName/
UserEmail" , "RolesKey": "GroupId/GroupName"}'
```

ドメインでの IAM アイデンティティセンター認証の無効化

OpenSearch ドメインで IAM アイデンティティセンターを無効にするには

1. ドメインを選択し、[アクション] から [セキュリティ設定の編集] を選択します。
2. [IAM アイデンティティセンターで認証された API アクセスを有効にする] のチェックを外します。
3. [Save changes] (変更の保存) をクリックします。
4. ドメインの処理が完了したら、IAM Identity Center プリンシパルに追加されたロールマッピングを削除します。

CLI を使用して IAM アイデンティティセンターを無効にするには、以下を使用できます。

```
aws opensearch update-domain-config \
    --domain-name my-domain \
    --identity-center-options '{"EnabledAPIAccess": false}'
```

OpenSearch Dashboards の Amazon Cognito 認証の設定

Amazon Cognito を使用して、OpenSearch Dashboards の Amazon OpenSearch Service のデフォルトインストールを認証し、保護することができます。Amazon Cognito 認証はオプションであり、OpenSearch または Elasticsearch 5.1 以降を使用しているドメインでのみ使用できます。Amazon Cognito 認証を設定しない場合でも、IP ベースのアクセスポリシーとプロキシサーバー、HTTP 基本認証、または SAML を使用して、Dashboards を保護することができます。

認証プロセスの多くは Amazon Cognito で発生しますが、このセクションでは、Amazon Cognito リソースを OpenSearch Service ドメインで使用するように設定するためのガイドラインと要件を示します。すべての Amazon Cognito リソースに標準の料金が適用されます。

> **Tip**
>
> OpenSearch Dashboards の Amazon Cognito 認証を使用するようにドメインを初めて設定するときは、コンソールを使用することをお勧めします。Amazon Cognito リソースは高度にカスタマイズ可能であり、コンソールにより、該当する機能を識別して理解しやすくなります。

トピック

- 前提条件
- Amazon Cognito 認証を使用するためのドメインの設定
- 認証されたロールの許可
- ID プロバイダの設定
- (オプション) きめ細かなアクセスの設定
- (オプション) サインインページのカスタマイズ
- (オプション) アドバンストセキュリティを設定する
- テスト
- クォータ
- 一般的な設定の問題
- OpenSearch Dashboards の Amazon Cognito 認証を無効にする
- OpenSearch Dashboards の Amazon Cognito 認証を使用するドメインを削除する

前提条件

OpenSearch Dashboards の Amazon Cognito 認証を設定する前に、いくつかの前提条件を満たす必要があります。OpenSearch Service コンソールは、これらのリソースの作成の効率化を支援しますが、各リソースの目的を理解すると、設定およびトラブルシューティングに役立ちます。Dashboards の Amazon Cognito 認証では、次のリソースが必要になります。

- Amazon Cognito ユーザープール

- Amazon Cognito ID プール
- `AmazonOpenSearchServiceCognitoAccess` ポリシーがアタッチされた IAM ロール (`CognitoAccessForAmazonOpenSearch`)

> **Note**
>
> ユーザープールと ID プールは、同じ AWS リージョンに存在している必要があります。同じユーザープール、ID プール、および IAM ロールを使用して、Dashboards の Amazon Cognito 認証を複数の OpenSearch Service ドメインに追加できます。詳細についてはthe section called “クォータ”を参照してください。

ユーザープールについて

ユーザープールには、次の 2 つの主要な機能があります。それらは、ユーザーのディレクトリを作成および管理する機能と、ユーザーによるサインアップとログインを許可する機能です。ユーザープールを作成する手順については、「Amazon Cognito デベロッパーガイド」の「Getting started with user pools」を参照してください。

OpenSearch Service で使用するユーザープールを作成するときは、以下の点を考慮します。

- Amazon Cognito ユーザープールにはドメイン名が必要です。OpenSearch Service はこのドメイン名を使用して、Dashboards にアクセスするためのログインページにユーザーをリダイレクトします。ユーザープールには、ドメイン名以外に、デフォルト以外の設定は必要ありません。
- プールの必須の標準属性 (名前、誕生日、E メールアドレス、電話番号など) を指定する必要があります。ユーザープールの作成後にこれらの属性を変更することはできないため、この時点で関連するものを選択します。
- ユーザープールを作成する際に、ユーザーが自分のアカウントを作成できるかどうか、アカウントのパスワードの最低強度、多要素認証を有効にするかどうかを選択します。外部 ID プロバイダを使用する予定がある場合、これらの設定は重要ではありません。技術的には、ID プロバイダとしてユーザープールを有効にし、さらに外部 ID プロバイダを有効にすることができますが、ほとんどのユーザーはどちらか一方を希望します。

ユーザープール ID の形式は *`region_ID`* です。CLI または AWS SDK AWS を使用して OpenSearch Service を設定する場合は、ID を書き留めます。

ID プールについて

ID プールにより、ユーザーのログイン後に、権限の制限された一時的なロールをユーザーに割り当てることができます。アイデンティティプールを作成する手順については、「Amazon Cognito デベロッパーガイド」の「Identity pools console overview」を参照してください。OpenSearch Service で使用する ID プールを作成するときは、以下の点を考慮します。

- Amazon Cognito コンソールを使用する場合は、[認証されていない ID に対してアクセスを有効にする] チェックボックスをオンにして、ID プールを作成する必要があります。ID プールを作成し、OpenSearch Service ドメインを設定すると、Amazon Cognito によりこの設定は無効になります。
- ID プールに外部 ID プロバイダを追加する必要はありません。Amazon Cognito 認証を使用するように OpenSearch Service を設定すると、先ほど作成したユーザープールを使用するように ID プールが設定されます。
- ID プールを作成した後、認証されていない IAM ロールと認証された IAM ロールを選択する必要があります。これらのロールにより、ログイン前およびログイン後にユーザーに与えられるアクセスポリシーが指定されます。Amazon Cognito コンソールを使用する場合は、これらのロールを自動的に作成できます。認証されたロールを作成した後に、ARN を書き留めます。この形式は `arn:aws:iam::123456789012:role/Cognito_identitypoolnameAuth_Role` です。

ID プール ID の形式は *region*:*ID*-*ID*-*ID*-*ID*-*ID* です。CLI または AWS SDK AWS を使用して OpenSearch Service を設定する場合は、ID を書き留めます。

CognitoAccessForAmazonOpenSearch ロールについて

OpenSearch Service には、Amazon Cognito ユーザーおよび ID プールを設定し、認証のためにそれらを使用するための許可が必要です。この目的のためにAmazonOpenSearchServiceCognitoAccess、 AWS マネージドポリシーである を使用できます。 AmazonESCognitoAccessは、サービスの名前が Amazon OpenSearch Service に変更されたAmazonOpenSearchServiceCognitoAccessときに に置き換えられたレガシーポリシーです。どちらのポリシーも、Amazon Cognito 認証を有効にするために必要な最小限の Amazon Cognito の許可を提供します。ポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「AmazonOpenSearchServiceCognitoAccess」を参照してください。

> **Important**
>
> ドメインで Amazon Cognito 認証を有効にした後、Amazon Cognito App Client の設定を手動で変更 (ID プロバイダーの追加など) しないでください。Amazon OpenSearch Service

は、ドメインの Amazon Cognito オプションを変更するとき、または Elasticsearch-to-OpenSearchへのアップグレード中に、手動の App Client の変更を上書きすることがあります。カスタム App Client 設定を保持するには、ドメインを変更するたびに適用します。

コンソールを使用して OpenSearch Service ドメインを作成または設定する場合、IAM ロールが作成され、`AmazonOpenSearchServiceCognitoAccess` ポリシー (Elasticsearch ドメインの場合は `AmazonESCognitoAccess` ポリシー) がロールに添付されます。このロールのデフォルト名は `CognitoAccessForAmazonOpenSearch` です。

ロールの許可ポリシー `AmazonOpenSearchServiceCognitoAccess` および `AmazonESCognitoAccess` のいずれも、OpenSearch Service が、すべてのアイデンティティおよびユーザープールに対して以下のアクションを実行することを許可します。

- アクション: `cognito-idp:DescribeUserPool`
- アクション: `cognito-idp:CreateUserPoolClient`
- アクション: `cognito-idp:DeleteUserPoolClient`
- アクション: `cognito-idp:UpdateUserPoolClient`
- アクション: `cognito-idp:DescribeUserPoolClient`
- アクション: `cognito-idp:AdminInitiateAuth`
- アクション: `cognito-idp:AdminUserGlobalSignOut`
- アクション: `cognito-idp:ListUserPoolClients`
- アクション: `cognito-identity:DescribeIdentityPool`
- アクション: `cognito-identity:SetIdentityPoolRoles`
- アクション: `cognito-identity:GetIdentityPoolRoles`

AWS CLI またはいずれかの AWS SDKs を使用する場合は、OpenSearch Service ドメインを設定するときに、独自のロールを作成し、ポリシーをアタッチして、このロールの ARN を指定する必要があります。ロールには、次の信頼関係が必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

手順については、「IAM ユーザーガイド」の「 AWS サービスにアクセス許可を委任するロールの作成」および「IAM ID アクセス許可の追加と削除」を参照してください。

Amazon Cognito 認証を使用するためのドメインの設定

前提条件を完了したら、Dashboards の Amazon Cognito を使用するように OpenSearch Service ドメインを設定できます。

> **Note**
>
> Amazon Cognito はすべての で利用できるわけではありません AWS リージョン。サポートされているリージョンのリストについては、Amazon Cognito の「サービスエンドポイント」を参照してください。OpenSearch Service で使用する Amazon Cognito 用に同じリージョンを使用する必要はありません。

Amazon Cognito 認証の設定 (コンソール)

コンソールによって `CognitoAccessForAmazonOpenSearch` ロールが作成されるため、コンソールの設定は非常にシンプルになっています。標準の OpenSearch Service のアクセス許可に加えて、コンソールで OpenSearch Dashboards の Amazon Cognito 認証を使用するドメインを作成するには、次のアクセス許可のセットが必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:DescribeVpcs",
        "cognito-identity:ListIdentityPools",
        "cognito-idp:ListUserPools",
        "iam:CreateRole",
        "iam:AttachRolePolicy"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-
role/CognitoAccessForAmazonOpenSearch"
    }
  ]
}
```

アイデンティティ (ユーザー、ユーザーグループ、またはロール) にアクセス許可を追加するには、「IAM ID アクセス許可の追加 (コンソール)」を参照してください。

CognitoAccessForAmazonOpenSearch がすでに存在する場合、必要なアクセス許可は少なくなります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "cognito-identity:ListIdentityPools",
        "cognito-idp:ListUserPools"
      ],
      "Resource": "*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-
role/CognitoAccessForAmazonOpenSearch"
    }
  ]
}
```

Dashboards の Amazon Cognito 認証を設定するには (コンソール)

1. https://console.aws.amazon.com/aos/home/ で Amazon OpenSearch Service コンソールを開きます。
2. [ドメイン] の下で、設定するドメインを選択します。
3. [アクション] から [セキュリティ設定の編集] を選択します。
4. [Amazon Cognito 認証を有効にする] を選択します。
5. リージョンで、Amazon Cognito ユーザープールと ID プール AWS リージョン を含む を選択します。
6. [Cognito ユーザープール] で、ユーザープールを選択または作成します。詳細については、「the section called “ユーザープールについて”」を参照してください。
7. [Cognito ID プール] で、ID プールを選択または作成します。詳細については、「the section called “ID プールについて”」を参照してください。

   > **Note**
   >
   > [ユーザープールの作成] および [ID プールの作成] リンクから Amazon Cognito コンソールに移動したら、これらのリソースを手動で作成します。このプロセスは自動的には行われません。詳細については、「the section called “前提条件”」を参照してください。

8. [IAM ロール名] で、CognitoAccessForAmazonOpenSearch のデフォルト値を使用するか (推奨)、新しい名前を入力します。詳細については、「the section called “CognitoAccessForAmazonOpenSearch ロールについて”」を参照してください。
9. [Save changes] (変更の保存) をクリックします。

ドメインによる処理が終了したら、追加の設定ステップについて「the section called “認証されたロールの許可”」および「the section called “ID プロバイダの設定”」を参照してください。

Amazon Cognito 認証の設定 (AWS CLI)

--cognito-options パラメータを使用して、OpenSearch Service ドメインを設定します。create-domain コマンドと update-domain-config コマンドの両方で次の構文が使用されます。

```
--cognito-options Enabled=true,UserPoolId="user-pool-id",IdentityPoolId="identity-pool-id",RoleArn="arn:aws:iam::123456789012:role/CognitoAccessForAmazonOpenSearch"
```

例

次の例では、CognitoAccessForAmazonOpenSearch ロールを使用して、Dashboards の Amazon Cognito 認証を有効にするドメインを us-east-1 リージョンで作成し、Cognito_Auth_Role へのドメインアクセスを提供します。

```
aws opensearch create-domain --domain-name my-domain --
region us-east-1 --access-policies '{ "Version": "2012-10-17",
        "Statement":[{"Effect":"Allow","Principal":{"AWS":
 ["arn:aws:iam::123456789012:role/
Cognito_Auth_Role"]},"Action":"es:ESHttp*","Resource":"arn:aws:es:us-
east-1:123456789012:domain/*" }]}' --engine-version "OpenSearch_1.0"
 --cluster-config InstanceType=m4.xlarge.search,InstanceCount=1
 --ebs-options EBSEnabled=true,VolumeSize=10 --cognito-options
 Enabled=true,UserPoolId="us-east-1_123456789",IdentityPoolId="us-
east-1:12345678-1234-1234-1234-123456789012",RoleArn="arn:aws:iam::123456789012:role/
CognitoAccessForAmazonOpenSearch"
```

ドメインによる処理が終了したら、追加の設定ステップについて「the section called “認証されたロールの許可”」および「the section called “ID プロバイダの設定”」を参照してください。

Amazon Cognito 認証 (AWS SDKsの設定

AWS SDKs (Android および iOS SDKs を除く) は、 CreateDomain および オペレーションの CognitoOptionsパラメータを含め、Amazon OpenSearch Service API リファレンスで定義されているすべてのUpdateDomainConfigオペレーションをサポートします。 AWS SDKsAWS「 Software Development Kits」を参照してください。

ドメインによる処理が終了したら、追加の設定ステップについて「the section called “認証されたロールの許可”」および「the section called “ID プロバイダの設定”」を参照してください。

認証されたロールの許可

デフォルトでは、「the section called “ID プールについて”」のガイドラインに従って設定された、認証された IAM ロールには、OpenSearch Dashboards にアクセスするために必要な権限がありません。追加のアクセス権限を持つロールを提供する必要があります。

> **Note**
>
> きめ細かなアクセスコントロールを設定しており、オープンまたは IP ベースのアクセスポリシーを使用している場合は、この手順をスキップできます。

これらのアクセス許可をアイデンティティベースのポリシーに含めることはできますが、認証されたユーザーがすべての OpenSearch Service ドメインにアクセスできるようにする場合を除いて、1 つのドメインにアタッチされたリソースベースのポリシーの方が適切です。

`Principal` には、the section called “ID プールについて” のガイドラインに従って設定した Cognito 認証ロールの ARN を指定します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":[
               "arn:aws:iam::111122223333:role/Cognito_identitypoolname/
Auth_Role"
            ]
         },
         "Action":[
            "es:ESHttp*"
         ],
         "Resource":"arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
      }
   ]
```

```
}
```

リソースベースのポリシーを OpenSearch Service ドメインに追加する手順については、「the section called “アクセスポリシーの設定”」を参照してください。

ID プロバイダの設定

Dashboards の Amazon Cognito 認証を使用するようにドメインを設定すると、OpenSearch Service はユーザープールにアプリクライアントを追加し、認証されたプロバイダとして ID プールにユーザープールを追加します。

> **Warning**
>
> アプリクライアントを名前変更または削除しないでください。

ユーザープールの設定方法に応じて、ユーザーアカウントを手動で作成する必要がある場合や、ユーザーが自分のアカウントを作成できる場合があります。これらの設定で問題ない場合は、それ以上アクションを実行する必要はありません。ただし、多くのユーザーが、外部 ID プロバイダの使用を希望します。

SAML 2.0 ID プロバイダを有効にするには、SAML メタデータドキュメントを提供する必要があります。Login with Amazon、Facebook、Google などのソーシャル ID プロバイダを有効にするには、これらのプロバイダからのアプリ ID とアプリシークレットが必要です。ID プロバイダの任意の組み合わせを有効にすることができます。

ユーザープールを設定する最も簡単な方法は、Amazon Cognito コンソールを使用することです。手順については、「Amazon Cognito デベロッパーガイド」の「User pool sign-in with third party identity providers」および「Application-specific settings with app client」を参照してください。

(オプション) きめ細かなアクセスの設定

デフォルトの ID プール設定では、同じ IAM ロール (`Cognito_`*`identitypool`*`Auth_Role`) にログインするすべてのユーザーが割り当てられます。つまり、すべてのユーザーが同じ AWS リソースにアクセスできます。Amazon Cognito とともにきめ細かなアクセスコントロールを使用する場合、例えば、組織のアナリストが複数のインデックスに対して読み取り専用アクセスを持つようにする一方で、デベロッパーはすべてのインデックスに対して書き込みアクセスを持つようにする場合、次の 2 つのオプションがあります。

- ユーザーグループを作成し、ユーザーの認証トークンに基づいて IAM ロールを選択するように ID プロバイダを設定する (推奨)。
- 1 つ以上のルールに基づいて IAM ロールを選択するように ID プロバイダを設定する。

きめ細かなアクセスコントロールを含むチュートリアルについては、「the section called “チュートリアル: Cognito 認証によるきめ細かなアクセスコントロール”」を参照してください。

> **Important**
>
> デフォルトロールと同様に、Amazon Cognito は追加の各ロールの信頼関係に含まれている必要があります。詳細については、「Amazon Cognito デベロッパーガイド」の「Creating roles for role mapping」を参照してください。

ユーザーグループとトークン

ユーザーグループを作成するときは、グループのメンバー用の IAM ロールを選択します。グループの作成の詳細については、「Amazon Cognito デベロッパーガイド」の「Adding groups to a user pool」を参照してください。

1 つ以上のユーザーグループを作成した後、ID プールのデフォルトロールではなく、グループのロールをユーザーに割り当てるように認証プロバイダを設定できます。[トークンから選択する] を選択してから、[デフォルトの認証されたロールを使用する] または [拒否] を選択して、ID プールでのグループの一部ではないユーザーの処理方法を指定します。

Rules

基本的に、ルールは Amazon Cognito が順番に評価する一連の `if` ステートメントです。例えば、ユーザーの E メールアドレスに `@corporate` が含まれる場合、Amazon Cognito はそのユーザーに `Role_A` を割り当てます。ユーザーの E メールアドレスに `@subsidiary` が含まれる場合、そのユーザーに `Role_B` が割り当てられます。それ以外の場合は、デフォルトの認証されたロールがユーザーに割り当てられます。

詳細については、「Amazon Cognito デベロッパーガイド」の「Using rule-based mapping to assign roles to users」を参照してください。

(オプション) サインインページのカスタマイズ

Amazon Cognito コンソールを使用して、カスタムロゴをアップロードし、サインインページで CSS の変更を行うことができます。手順と CSS プロパティの完全なリストについては、「Amazon Cognito デベロッパーガイド」の「Customizing hosted UI (classic) branding」を参照してください。

(オプション) アドバンストセキュリティを設定する

Amazon Cognito ユーザープールでは、多要素認証、侵害された認証情報の確認、およびアダプティブ認証などのアドバンストセキュリティ機能がサポートされています。詳細については、「Amazon Cognito デベロッパーガイド」の「Using Amazon Cognito user pools security features」を参照してください。

テスト

設定に問題がなければ、ユーザーエクスペリエンスが期待どおりであることを確認します。

OpenSearch Dashboards にアクセスするには

1. ウェブブラウザで `https://`*`opensearch-domain`*`/_dashboards` にアクセスします。特定のテナントに直接ログインするには、URL に `?security_tenant=`*`tenant-name`* を追加します。
2. 任意の認証情報を使用してサインインします。
3. OpenSearch Dashboards がロードされたら、少なくとも 1 つのインデックスパターンを設定します。Dashboards は、そのパターンを使用して、どのインデックスを分析するかを特定します。「*」と入力し、[次のステップ]、[Create index pattern (インデックスパターンの作成)] の順に選択します。
4. データを検索または調査するには、[検出] を選択します。

このプロセスのいずれかのステップが失敗した場合は、トラブルシューティング情報について、「the section called “一般的な設定の問題”」を参照してください。

クォータ

Amazon Cognito には、そのリソースの多くでソフト制限があります。多数の OpenSearch Service ドメインに対して Dashboards 認証を有効にする場合は、必要に応じて Amazon Cognito におけるクォータと制限の引き上げのリクエストについて確認してください。

各 OpenSearch Service ドメインでは、アプリクライアントがユーザープールに追加されます。これにより、認証プロバイダが ID プールに追加されます。10 を超えるドメインに対して OpenSearch Dashboards 認証を有効にすると、「ID プールあたりの Amazon Cognito の最大ユーザープールプロバイダ数」の制限が適用されることがあります。制限を超えた場合、Dashboards の Amazon Cognito 認証を使用する設定を試みている OpenSearch Service ドメインが、処理中の設定状態でスタックする可能性があります。

一般的な設定の問題

一般的な設定の問題と解決策を、以下の表に示します。

OpenSearch Service の設定

問題	ソリューション
`OpenSearch Service can't create the role` (コンソール)	適切な IAM アクセス許可がありません。「the section called “Amazon Cognito 認証の設定 (コンソール)”」で指定されたアクセス許可を追加します。
`User is not authorized to perform: iam:PassRole on resource CognitoAccessForAmazonOpenSearch` (コンソール)	CognitoAccessforAmazonOpenSearch ロールの `iam:PassRole` 許可がありません。次のポリシーをアカウントにアタッチします: JSON `{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iam:PassRole" ], "Resource": "arn:aws:iam:: 123456789012:role/service-role/CognitoAccessForAmazonOpenSearch " } ] }`

問題	ソリューション
	または、IAMFullAccess ポリシーをアタッチできます。
User is not authorized to perform: cognito-identity:ListIdentityPools on resource	Amazon Cognito の読み取り許可がありません。AmazonCognitoReadOnly ポリシーをアカウントにアタッチします。
An error occurred (ValidationException) when calling the CreateDomain operation: OpenSearch Service must be allowed to use the passed role	OpenSearch Service が CognitoAccessForAmazonOpenSearch ロールの信頼関係で指定されていません。ロールで、「the section called “CognitoAccessForAmazonOpenSearch ロールについて”」で指定された信頼関係が使用されていることを確認します。または、コンソールを使用して Amazon Cognito 認証を設定します。コンソールによってロールが作成されます。
An error occurred (ValidationException) when calling the CreateDomain operation: User is not authorized to perform: cognito-idp: *action* on resource: *user pool*	--cognito-options で指定されたロールに、Amazon Cognito にアクセスする許可がありません。ロールに AWS 管理AmazonOpenSearchServiceCognitoAccess ポリシーがアタッチされていることを確認します。または、コンソールを使用して Amazon Cognito 認証を設定します。コンソールによってロールが作成されます。
An error occurred (ValidationException) when calling the CreateDomain operation: User pool does not exist	OpenSearch Service はユーザープールを見つけることができません。ユーザープールを作成し、正しい ID が設定されていることを確認します。ID を見つけるには、Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用します。 `aws cognito-idp list-user-pools --max-results 60 --region region`

問題	ソリューション
`An error occurred (ValidationException) when calling the CreateDomain operation: IdentityPool not found`	OpenSearch Service は ID プールを見つけることができません。ユーザープールを作成し、正しい ID が設定されていることを確認します。ID を見つけるには、Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用します。 `aws cognito-identity list-identity-pools --max-results 60 --region region`
`An error occurred (ValidationException) when calling the CreateDomain operation: Domain needs to be specified for user pool`	ユーザープールにドメイン名がありません。Amazon Cognito コンソールまたは次の AWS CLI コマンドを使用して、ドメイン名を設定できます。 `aws cognito-idp create-user-pool-domain --domain name --user-pool-id id`

OpenSearch Dashboards へのアクセス

問題	ソリューション
目的の ID プロバイダがログインページに表示されない。	「the section called “ID プロバイダの設定”」で指定したように、OpenSearch Service アプリクライアントで ID プロバイダが有効になっていることを確認します。
ログインページが、組織に関連付けられているように見えない。	「the section called “(オプション) サインインページのカスタマイズ”」を参照してください。
ログイン認証情報が受け入れられない。	「the section called “ID プロバイダの設定”」で指定したように ID プロバイダを設定したことを確認します。 ID プロバイダとしてユーザープールを使用している場合は、アカウントが Amazon Cognito コンソールに存在することを確認します。

問題	ソリューション
OpenSearch Dashboards がまったくロードされないか、正しく動作しない。	Amazon Cognito の認証されたロールでは、ドメイン (/*) が Dashboards にアクセスして使用するためには、`es:ESHttp*` の許可が必要です。「the section called “認証されたロールの許可”」で指定したように、アクセスポリシーを追加したことを確認します。
あるタブで OpenSearch Dashboards からサインアウトすると、残りのタブには更新トークンが取り消されたことを示すメッセージが表示されます。	Amazon Cognito 認証使用時に OpenSearch Dashboards セッションからサインアウトすると、OpenSearch Service で AdminUserGlobalSignOut オペレーションが実行されて、すべてのアクティブな OpenSearch Dashboards セッションから強制サインアウトされます。

問題	ソリューション
`Invalid identity pool configuration. Check assigned IAM roles for this pool.`	Amazon Cognito には、認証されたユーザーに代わって IAM ロールを引き受ける許可がありません。ロールの信頼関係を変更して、以下を含めます。 JSON `{ "Version":"2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Federated": "cognito-identity.amazonaws.com" }, "Action": "sts:AssumeRoleWithWebIdentity", "Condition": { "StringEquals": { "cognito-identity.amazonaws.com:aud": " identity-pool-id " }, "ForAnyValue:StringLike": { "cognito-identity.amazonaws.com:amr": "authenticated" } } }] }`
`Token is not from a supported provider of this identity pool.`	ユーザープールからアプリクライアントを削除すると、この一般的でないエラーが発生する可能性があります。新しいブラウザセッションで Dashboards を開いてみます。

OpenSearch Dashboards の Amazon Cognito 認証を無効にする

Dashboards の Amazon Cognito 認証を無効にするには、次の手順を使用します。

Dashboards の Amazon Cognito 認証を無効にするには (コンソール)

1. Amazon OpenSearch Service コンソールを開きます。
2. [Domains] (ドメイン) で、設定するドメインを選択します。
3. [アクション] から [セキュリティ設定の編集] を選択します。
4. [Amazon Cognito 認証を有効にする] を選択解除します。
5. [Save changes] (変更の保存) をクリックします。

> **Important**
>
> Amazon Cognito ユーザープールと ID プールが不要になった場合は、それらを削除します。削除しなかった場合は、利用料金が引き続き発生します。

OpenSearch Dashboards の Amazon Cognito 認証を使用するドメインを削除する

Dashboards の Amazon Cognito 認証を使用するドメインが [処理中] の設定状態で止まらないようにするには、関連する Amazon Cognito ユーザープールと ID プールを削除する前に OpenSearch Service ドメインを削除します。

Amazon OpenSearch Service 用のサービスにリンクされたロールの使用

Amazon OpenSearch Service は AWS Identity and Access Management 、(IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、OpenSearch Service に直接リンクされた一意のタイプの (IAM) ロールです。サービスにリンクされたロールは OpenSearch Service によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用することで、必要な許可を手動で追加する必要がなくなるため、OpenSearch Service の設定が簡単になります。OpenSearch Service は、サービスにリンクさ

れたロールの許可を定義します。特に定義されている場合を除き、OpenSearch Service のみがそのロールを引き受けることができます。定義される許可には、信頼ポリシーと許可ポリシーが含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。サービスにリンクされたロールとアクセス許可ポリシーの更新については、「Amazon OpenSearch Service のドキュメント履歴」を参照してください。

サービスにリンクされたロールをサポートする他のサービスの詳細については、AWS「IAM と連携するサービス」を参照し、「サービスにリンクされたロール」列で「はい」を持つサービスを探します。サービスリンクロールに関するドキュメントをサービスで表示するには、リンクで [はい] を選択します。

トピック

- サービスにリンクされたロールを使用して、VPC ドメインとダイレクトクエリデータソースを作成する
- サービスにリンクされたロールを使用して OpenSearch Serverless コレクションを作成する
- サービスにリンクされたロールを使用した OpenSearch Ingestion パイプラインの作成

サービスにリンクされたロールを使用して、VPC ドメインとダイレクトクエリデータソースを作成する

Amazon OpenSearch Service は AWS Identity and Access Management 、(IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、OpenSearch Service に直接リンクされた一意のタイプの (IAM) ロールです。サービスにリンクされたロールは、OpenSearch Service によって事前定義されており、ユーザーの代わりにサービスから他の AWS サービスを呼び出すために必要なアクセス許可をすべて含んでいます。

OpenSearch Service は AWSServiceRoleForAmazonOpenSearchService と呼ばれるサービスにリンクされたロールを使用します。このロールは、ドメインまたはダイレクトクエリデータソースの VPC アクセスを有効にするために、ロールで必要な最小限の Amazon EC2 および Elastic Load Balancing のアクセス許可を提供します。

従来の Elasticsearch ロール

Amazon OpenSearch Service は、`AWSServiceRoleForAmazonOpenSearchService` と呼ばれるサービスにリンクされたロールを使用します。アカウントには、`AWSServiceRoleForAmazonElasticsearchService` と呼ばれるサービスにリンクされた

ロールも含まれることがあります。これは、非推奨の Elasticsearch API エンドポイントと連携します。

従来の Elasticsearch ロールがアカウントに存在しない場合、OpenSearch ドメインを初めて作成するときに、OpenSearch サービスによって新しい OpenSearch サービスにリンクされたロールが自動的に作成されます。そうでない場合、アカウントでは Elasticsearch ロールが引き続き使用されます。この自動作成を成功させるためには、`iam:CreateServiceLinkedRole` アクションへのアクセス許可が必要です。

アクセス許可

`AWSServiceRoleForAmazonOpenSearchService` サービスリンクロールは、以下のサービスを信頼してロールを引き受けます。

- `opensearchservice.amazonaws.com`

`AmazonOpenSearchServiceRolePolicy` という名前のロール許可ポリシーは、OpenSearch Service が次のアクションを指定されたリソースで完了することを許可します。

- アクション: `acm:DescribeCertificate`。対象リソース: *
- アクション: * 上で `cloudwatch:PutMetricData`
- アクション: * 上で `ec2:CreateNetworkInterface`
- アクション: * 上で `ec2:DeleteNetworkInterface`
- アクション: * 上で `ec2:DescribeNetworkInterfaces`
- アクション: * 上で `ec2:ModifyNetworkInterfaceAttribute`
- アクション: * 上で `ec2:DescribeSecurityGroups`
- アクション: * 上で `ec2:DescribeSubnets`
- アクション: `ec2:DescribeVpcs`。対象リソース: *
- アクション: すべてのネットワークインターフェースと VPC エンドポイントで `ec2:CreateTags`
- アクション: `ec2:DescribeTags`。対象リソース: *
- アクション: すべての VPC、セキュリティグループ、サブネット、およびルートテーブル、ならびにリクエストにタグ `OpenSearchManaged=true` が含まれる場合、すべての VPC エンドポイントで `ec2:CreateVpcEndpoint`

- アクション: すべての VPC、セキュリティグループ、サブネット、およびルートテーブル、ならびにリクエストにタグ `OpenSearchManaged=true` が含まれる場合、すべての VPC エンドポイントで `ec2:ModifyVpcEndpoint`
- アクション: リクエストにタグ `OpenSearchManaged=true` が含まれる場合、すべてのエンドポイントで `ec2:DeleteVpcEndpoints`
- アクション: `ec2:AssignIpv6Addresses`。対象リソース: *
- アクション: * 上で `ec2:UnAssignIpv6Addresses`
- アクション: * 上で `elasticloadbalancing:AddListenerCertificates`
- アクション: `elasticloadbalancing:RemoveListenerCertificates` 上で *

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、権限を設定する必要があります。詳細については、「IAM ユーザーガイド」の「Service-linked role permissions」を参照してください。

サービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。を使用して VPC 対応ドメインまたは直接クエリデータソースを作成すると AWS マネジメントコンソール、OpenSearch Service によってサービスにリンクされたロールが作成されます。この自動作成を成功させるためには、`iam:CreateServiceLinkedRole` アクションへのアクセス許可が必要です。

サービスにリンクされたロールは、IAM コンソール、IAM CLI、または IAM API を使用して手動で作成することもできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの作成」を参照してください。

サービスにリンクされたロールを編集する

OpenSearch Service では、`AWSServiceRoleForAmazonOpenSearchService` サービスにリンクされたロールを編集できません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

サービスリンクロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティ

ティを排除できます。ただし、手動で削除する前に、サービスリンクロールをクリーンアップする必要があります。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。次に、`AWSServiceRoleForAmazonOpenSearchService` ロールの名前 (チェックボックスではありません) を選択します。
3. 選択したロールの [概要] ページで、[アクセスアドバイザー] タブを選択します。
4. [アクセスアドバイザー] タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

> **Note**
>
> OpenSearch Service で `AWSServiceRoleForAmazonOpenSearchService` ロールが使用されているかどうかが不明な場合は、このロールの削除を試みることができます。サービスでロールが使用されている場合、削除は失敗し、ロールが使用されているリソースを表示できます。ロールが使用されている場合は、ロールを削除する前、またはロールを使用しているリソースを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

サービスにリンクされたロールの手動削除

IAM コンソール、API、または AWS CLI からサービスにリンクされたロールを削除します。手順については、IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

サービスにリンクされたロールを使用して OpenSearch Serverless コレクションを作成する

OpenSearch Serverless は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、OpenSearch Service に直接リンクされた一意のタイプの (IAM) ロールです。サービスにリンクされたロールは、OpenSearch Service によって事前定義されており、ユーザーの代わりにサービスから他の AWS サービスを呼び出すために必要なアクセス許可をすべて含んでいます。

OpenSearch Serverless は、AWSServiceRoleForAmazonOpenSearchServerless と呼ばれるサービスにリンクされたロールを使用します。このロールは、サーバーレス関連の CloudWatch メトリクスをアカウントに発行するためにロールで必要なアクセス許可を提供します。AWSServiceRoleForAmazonOpenSearchServerless に関連付けられたロールアクセス許可ポリシーの名前は `AmazonOpenSearchServerlessServiceRolePolicy` です。ポリシーの詳細については、「AWS Managed Policy Reference Guide」の「AmazonOpenSearchServerlessServiceRolePolicy」を参照してください。

OpenSearch Serverless のサービスにリンクされたロールのアクセス許可

OpenSearch Serverless は、AWSServiceRoleForAmazonOpenSearchServerless と呼ばれるサービスにリンクされたロールを使用します。このロールにより、OpenSearch Serverless がユーザーに代わって AWS のサービスを呼び出すことができるようになります。

AWSServiceRoleForAmazonOpenSearchServerless サービスにリンクされたロールは、以下のサービスを信頼してロールを引き受けます。

- `observability.aoss.amazonaws.com`

`AmazonOpenSearchServerlessServiceRolePolicy` という名前のロール許可ポリシーによって、OpenSearch Serverless が次のアクションを指定されたリソースで完了できるようになります。

- アクション: すべての AWS リソースで `cloudwatch:PutMetricData`

> **Note**
>
> ポリシーには条件キー `{"StringEquals": {"cloudwatch:namespace": "AWS/AOSS"}}` が含まれていますが、これはサービスにリンクされたロールが AWS/AOSS CloudWatch 名前空間にのみメトリクスデータを送信できることを意味します。

サービスリンクロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するにはアクセス許可を設定する必要があります。詳細についてはIAM ユーザーガイド の「サービスにリンクされた役割のアクセス許可」を参照してください。

OpenSearch Serverless のサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS マネジメントコンソール、、AWS CLIまたは AWS API で OpenSearch Serverless コレクションを作成すると、OpenSearch Serverless によってサービスにリンクされたロールが作成されます。

> **Note**
>
> コレクションを初めて作成するときは、ID ベースのポリシーで `iam:CreateServiceLinkedRole` を割り当てる必要があります。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は同じ方法でアカウントにロールを再作成できます。OpenSearch Serverless コレクションを作成すると、OpenSearch Serverless がサービスにリンクされたロールを再度作成します。

IAM コンソールを使用して、Amazon OpenSearch Serverless ユースケースでサービスリンクロールを作成することもできます。 AWS CLI または AWS API で、サービス名を使用して`observability.aoss.amazonaws.com`サービスにリンクされたロールを作成します。

```
aws iam create-service-linked-role --aws-service-name
 "observability.aoss.amazonaws.com"
```

詳細については、IAM ユーザーガイドの「サービスリンクロールの作成」を参照してください。このサービスリンクロールを削除しても、同じ方法でロールを再作成できます。

OpenSearch Serverless のサービスにリンクされたロールの編集

OpenSearch Serverless では、AWSServiceRoleForAmazonOpenSearchServerless サービスリンクロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

OpenSearch Serverless のサービスにリンクされたロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを保持しないようにできます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

AWSServiceRoleForAmazonOpenSearchServerless を削除するには、まず 内のすべての OpenSearch Serverless コレクションを削除する必要があります AWS アカウント。

> **Note**
>
> リソースを削除しようとしたときに OpenSearch Serverless でロールが使用されている場合、削除が失敗することがあります。その場合は、数分待ってからオペレーションを再試行してください。

IAM を使用してサービスリンクロールを手動で削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonOpenSearchServerless サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

OpenSearch Serverless のサービスにリンクされたロールがサポートされているリージョン

OpenSearch Serverless は、OpenSearch Serverless が利用可能なすべてのリージョンで、サービスにリンクされたロール AWSServiceRoleForAmazonOpenSearchServerless の使用をサポートしています。サポートされているリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon OpenSearch Serverless endpoints and quotas」を参照してください。

サービスにリンクされたロールを使用した OpenSearch Ingestion パイプラインの作成

Amazon OpenSearch Ingestion は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールとは、OpenSearch Ingestion に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは OpenSearch Ingestion によって事前に定義され、ユーザーに代わって他の AWS サービスを呼び出す際にサービスが必要とするアクセス権限がすべて含まれています。

OpenSearch Ingestion は、[AWSServiceRoleForAmazonOpenSearchIngestionService] という名前のサービスにリンクされたロールを使用します。ただし、セルフマネージド VPC を使用する場合、[AWSServiceRoleForOpensearchIngestionSelfManagedVpce] という名前のサービスにリンクされたロールを使用します。添付されたポリシーは、アカウントと OpenSearch Ingestion のと間に仮想プライベートクラウド (VPC) を作成し、CloudWatch メトリクスをアカウントに発行するために、ロールが必要とするアクセス権限を提供します。

アクセス許可

`AWSServiceRoleForAmazonOpenSearchIngestionService` サービスリンクロールは、以下のサービスを信頼してロールを引き受けます。

- `osis.amazonaws.com`

`AmazonOpenSearchIngestionServiceRolePolicy` というロールアクセス権限ポリシーは、OpenSearch Ingestion に、指定されたリソースで次のアクションを実行することを許可します。

- アクション: `cloudwatch:PutMetricData`。対象リソース: `cloudwatch:namespace": "AWS/OSIS"`
- アクション: `arn:aws:ec2:*:*:network-interface/*` 上で `ec2:CreateTags`
- アクション: `*` 上で `ec2:CreateVpcEndpoint`
- アクション: `*` 上で `ec2:DeleteVpcEndpoints`
- アクション: `*` 上で `ec2:DescribeSecurityGroups`
- アクション: `*` 上で `ec2:DescribeSubnets`
- アクション: `*` 上で `ec2:DescribeVpcEndpoints`
- アクション: `ec2:ModifyVpcEndpoint`。対象リソース: `*`

AWSServiceRoleForOpensearchIngestionSelfManagedVpce サービスリンクロールは、以下のサービスを信頼してロールを引き受けます。

- self-managed-vpce.osis.amazonaws.com

OpenSearchIngestionSelfManagedVpcePolicy というロールアクセス権限ポリシーは、OpenSearch Ingestion に、指定されたリソースで次のアクションを実行することを許可します。

- アクション: ec2:DescribeSubnets。対象リソース: *
- アクション: * 上で ec2:DescribeSecurityGroups
- アクション: * 上で ec2:DescribeVpcEndpoints
- アクション: cloudwatch:PutMetricData 上で cloudwatch:namespace": "AWS/OSIS"

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、権限を設定する必要があります。詳細についてはIAM ユーザーガイド の「サービスにリンクされた役割のアクセス許可」を参照してください。

OpenSearch Ingestion のサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS マネジメントコンソール、、AWS CLIまたは AWS API で OpenSearch Ingestion パイプラインを作成すると、OpenSearch Ingestion によってサービスにリンクされたロールが作成されます。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は同じ方法でアカウントにロールを再作成できます。OpenSearch Ingestion パイプラインを作成すると、OpenSearch Ingestion が、再びサービスにリンクされたロールをユーザーの代わりに作成します。

OpenSearch Ingestion のサービスにリンクされたロールの編集

OpenSearch Ingestion では、AWSServiceRoleForAmazonOpenSearchIngestionService サービスにリンクされたロールは編集できません。サービスリンクロールの作成後は、さまざまなエンティティがロールを参照する可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

OpenSearch Ingestion のサービスにリンクされたロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

サービスリンク役割のクリーンアップ

IAM を使用してサービスにリンクされた役割を削除するには最初に、その役割で使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースを削除する際に OpenSearch Serverless がロールを使用していると、削除できないことがあります。失敗した場合は数分待ってから操作を再試行してください。

`AWSServiceRoleForAmazonOpenSearchIngestionService` または **`AWSServiceRoleForOpensearchIngestionSelfManagedVpce`** が使用している OpenSearch Ingestion リソースを削除するには

1. Amazon OpenSearch Service コンソールに移動し、[Ingestion] を選択します。
2. すべてのパイプラインを削除します。手順については、「the section called “パイプラインの削除”」を参照してください。

OpenSearch Ingestion のサービスにリンクされたロールを削除する

サービスにリンクされたロールは、OpenSearch Ingestion のコンソールから削除できます。

サービスにリンクされたロールを削除するには (コンソール)

1. [IAM console] (IAM コンソール) に入ります。
2. [ロール] を選択し、[AWSServiceRoleForAmazonOpenSearchIngestionService] または [AWSServiceRoleForOpensearchIngestionSelfManagedVpce] ロールを検索します。
3. ロールを選択し、[Delete] (削除) をクリックします。

Amazon OpenSearch Service のサンプルコード

この章には、以下の Amazon OpenSearch Service で使用するための一般的なサンプルコードが含まれます: さまざまなプログラミング言語での HTTP リクエストの署名、HTTP リクエストの本文の圧縮、ドメインを作成するための AWS SDK の使用。

トピック

- Elasticech クライアントの互換性
- Amazon OpenSearch Service での HTTP リクエストの圧縮
- AWS SDKs を使用して Amazon OpenSearch Service を操作する

Elasticech クライアントの互換性

最新バージョンの Elasticsearch クライアントには、人為的に互換性を損なうライセンスチェックやバージョンチェックが含まれる場合があります。次のテーブルに、OpenSearch Service との最大限の互換性で使用するための、これらのクライアントのバージョンに関する推奨事項を示します。

> **Important**
>
> これらのクライアントバージョンは古くて、Log4j を含む最新の依存関係では更新されません。可能であれば、クライアントの OpenSearch バージョンを使用することを強くお勧めします。

クライアント	推奨バージョン
Java 低レベル REST クライアント	7.13.4
Java 高レベル REST クライアント	7.13.4
Python Elasticsear	7.13.4
Ruby Elasticsearch クライアント	7.13.3
Node.js Elasticsearch クライアント	7.13.0

Amazon OpenSearch Service での HTTP リクエストの圧縮

gzip 圧縮を使用して、Amazon OpenSearch Service ドメインで HTTP リクエストとレスポンスを圧縮できます。gzip 圧縮を使用すると、ドキュメントのサイズを縮小し、帯域幅の使用率とレイテンシーを低減できるため、転送速度が向上します。

gzip 圧縮は、OpenSearch または Elasticsearch 6.0 以降を実行しているすべてのドメインでサポートされています。OpenSearch クライアントの中には、gzip 圧縮のサポートが組み込まれているものもあり、また、多くのプログラミング言語には、プロセスを簡素化するライブラリがあります。

gzip 圧縮を有効にする

OpenSearch の類似の設定と混同しないように注意してください。`http_compression.enabled` は OpenSearch Service に固有であり、ドメインで gzip 圧縮を有効または無効にします。OpenSearch または Elasticsearch 7.x を実行しているドメインではデフォルトで gzip 圧縮が有効になっていますが、Elasticsearch 6.x を実行しているドメインではデフォルトで無効になっています。

gzip 圧縮を有効にするには、次のリクエストを送信します。

```
PUT _cluster/settings
{
  "persistent" : {
    "http_compression.enabled": true
  }
}
```

`_cluster/settings` へのリクエストは圧縮解除する必要があるので、クラスター設定を更新するために別のクライアントまたは標準の HTTP 要求を使用する必要がある場合があります。

gzip 圧縮が正常に有効になったことを確認するには、次のリクエストを送信します。

```
GET _cluster/settings?include_defaults=true
```

レスポンスに次の設定が表示されていることを確認します。

```
...
"http_compression": {
```

```
  "enabled": "true"
}
...
```

必要なヘッダー

gzip 圧縮されたリクエストボディを含める場合は、標準の `Content-Type: application/json` ヘッダーを保持し、`Content-Encoding: gzip` ヘッダーを追加します。gzip 圧縮レスポンスを受け入れるには、`Accept-Encoding: gzip` ヘッダーも同様に追加します。OpenSearch クライアントが gzip 圧縮をサポートしている場合、これらのヘッダーが自動的に含まれている可能性があります。

サンプルコード (Python 3)

次のサンプルでは、opensearch-py を使用して圧縮を実行し、リクエストを送信します。このコードは、IAM 認証情報を使用してリクエストに署名します。

```
from opensearchpy import OpenSearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth
import boto3

host = '' # e.g. my-test-domain.us-east-1.es.amazonaws.com
region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

# Create the client.
search = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    http_compress = True, # enables gzip compression for request bodies
    connection_class = RequestsHttpConnection
)

document = {
  "title": "Moneyball",
  "director": "Bennett Miller",
```

```
  "year": "2011"
}

# Send the request.
print(search.index(index='movies', id='1', body=document, refresh=True))

# print(search.index(index='movies', doc_type='_doc', id='1', body=document,
 refresh=True))
```

あるいは、適切なヘッダーを指定し、リクエストボディを自身で圧縮し、リクエストのような標準 HTTP ライブラリを使用します。このコードは、HTTP 基本認証情報を使用してリクエストに署名します。これは、きめ細かなアクセスコントロール使用する場合に、ドメインがサポートする場合があります。

```
import requests
import gzip
import json

base_url = '' # The domain with https:// and a trailing slash. For example, https://my-
test-domain.us-east-1.es.amazonaws.com/
auth = ('master-user', 'master-user-password') # For testing only. Don't store
 credentials in code.

headers = {'Accept-Encoding': 'gzip', 'Content-Type': 'application/json',
           'Content-Encoding': 'gzip'}

document = {
  "title": "Moneyball",
  "director": "Bennett Miller",
  "year": "2011"
}

# Compress the document.
compressed_document = gzip.compress(json.dumps(document).encode())

# Send the request.
path = 'movies/_doc?refresh=true'
url = base_url + path
response = requests.post(url, auth=auth, headers=headers, data=compressed_document)
print(response.status_code)
print(response.text)
```

AWS SDKs を使用して Amazon OpenSearch Service を操作する

このセクションでは、 AWS SDKs を使用して Amazon OpenSearch Service 設定 API を操作する方法の例を示します。これらのコードサンプルは、OpenSearch Service ドメインの作成、更新、削除方法を示しています。

Java

このセクションでは、 のバージョン 1 と 2 の例を示します AWS SDK for Java。

Version 2

この例では、OpenSearchClientBuilder コンストラクタを使用して、 AWS SDK for Java のバージョン 2 から OpenSearch ドメインの作成、その設定の更新、および削除を行います。`waitForDomainProcessing` への呼び出しのコメント (および `deleteDomain` への呼び出しのコメント) を削除し、ドメインをオンラインにして使用できるようにします。

```
package com.example.samples;

import java.util.concurrent.TimeUnit;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.CognitoOptions;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainResponse;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

/**
 * Sample class demonstrating how to use the Amazon Web Services SDK for Java to
 create, update,
 * and delete Amazon OpenSearch Service domains.
 */
```

```
public class OpenSearchSample {

    public static void main(String[] args) {

     String domainName = "my-test-domain";

     // Build the client using the default credentials chain.
        // You can use the CLI and run `aws configure` to set access key, secret
        // key, and default region.

        OpenSearchClient client = OpenSearchClient.builder()
          // Unnecessary, but lets you use a region different than your default.
          .region(Region.US_EAST_1)
          // Unnecessary, but if desired, you can use a different provider chain.
          .credentialsProvider(DefaultCredentialsProvider.create())
                     .build();

        // Create a new domain, update its configuration, and delete it.
        createDomain(client, domainName);
        //waitForDomainProcessing(client, domainName);
        updateDomain(client, domainName);
        //waitForDomainProcessing(client, domainName);
        deleteDomain(client, domainName);
    }

    /**
     * Creates an Amazon OpenSearch Service domain with the specified options.
     * Some options require other Amazon Web Services resources, such as an Amazon
 Cognito user pool
     * and identity pool, whereas others require just an instance type or instance
     * count.
     *
     * @param client
     *            The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *            The name of the domain you want to create
     */

    public static void createDomain(OpenSearchClient client, String domainName) {

     // Create the request and set the desired configuration options

        try {
```

```
            ClusterConfig clusterConfig = ClusterConfig.builder()
                    .dedicatedMasterEnabled(true)
                    .dedicatedMasterCount(3)
                    // Small, inexpensive instance types for testing. Not
 recommended for production.
                    .dedicatedMasterType("t2.small.search")
                    .instanceType("t2.small.search")
                    .instanceCount(5)
                    .build();

            // Many instance types require EBS storage.
            EBSOptions ebsOptions = EBSOptions.builder()
                    .ebsEnabled(true)
                    .volumeSize(10)
                    .volumeType("gp2")
                    .build();

            NodeToNodeEncryptionOptions encryptionOptions =
 NodeToNodeEncryptionOptions.builder()
                    .enabled(true)
                    .build();

            CreateDomainRequest createRequest = CreateDomainRequest.builder()
                    .domainName(domainName)
                    .engineVersion("OpenSearch_1.0")
                    .clusterConfig(clusterConfig)
                    .ebsOptions(ebsOptions)
                    .nodeToNodeEncryptionOptions(encryptionOptions)
                    // You can uncomment this line and add your account ID, a
 username, and the
                    // domain name to add an access policy.
                    // .accessPolicies("{\"Version\":\"2012-10-17\",
\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
[\"arn:aws:iam::123456789012:user/user-name\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:region:123456789012:domain/domain-name/*\"}]}")
                    .build();

            // Make the request.
            System.out.println("Sending domain creation request...");
            CreateDomainResponse createResponse =
 client.createDomain(createRequest);
            System.out.println("Domain status:
 "+createResponse.domainStatus().toString());
```

```
            System.out.println("Domain ID:
"+createResponse.domainStatus().domainId());


        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the configuration of an Amazon OpenSearch Service domain with the
     * specified options. Some options require other Amazon Web Services resources,
such as an
     * Amazon Cognito user pool and identity pool, whereas others require just an
     * instance type or instance count.
     *
     * @param client
     *             The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *             The name of the domain to update
     */

    public static void updateDomain(OpenSearchClient client, String domainName) {

     // Updates the domain to use three data instances instead of five.
        // You can uncomment the Cognito line and fill in the strings to enable
Cognito
        // authentication for OpenSearch Dashboards.

        try {

            ClusterConfig clusterConfig = ClusterConfig.builder()
                    .instanceCount(5)
                    .build();

            CognitoOptions cognitoOptions = CognitoOptions.builder()
                    .enabled(true)
                    .userPoolId("user-pool-id")
                    .identityPoolId("identity-pool-id")
                    .roleArn("role-arn")
                    .build();
```

```
            UpdateDomainConfigRequest updateRequest =
UpdateDomainConfigRequest.builder()
                    .domainName(domainName)
                    .clusterConfig(clusterConfig)
                    //.cognitoOptions(cognitoOptions)
                    .build();

            System.out.println("Sending domain update request...");
            UpdateDomainConfigResponse updateResponse =
client.updateDomainConfig(updateRequest);
            System.out.println("Domain config:
"+updateResponse.domainConfig().toString());


        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an Amazon OpenSearch Service domain. Deleting a domain can take
     * several minutes.
     *
     * @param client
     *             The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *             The name of the domain that you want to delete
     */

    public static void deleteDomain(OpenSearchClient client, String domainName) {

        try {

            DeleteDomainRequest deleteRequest = DeleteDomainRequest.builder()
                    .domainName(domainName)
                    .build();

            System.out.println("Sending domain deletion request...");
            DeleteDomainResponse deleteResponse =
client.deleteDomain(deleteRequest);
            System.out.println("Domain status: "+deleteResponse.toString());
```

```
        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Waits for the domain to finish processing changes. New domains typically take
15-30 minutes
     * to initialize, but can take longer depending on the configuration. Most
updates to existing domains
     * take a similar amount of time. This method checks every 15 seconds and
finishes only when
     * the domain's processing status changes to false.
     *
     * @param client
     *            The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *            The name of the domain that you want to check
     */

    public static void waitForDomainProcessing(OpenSearchClient client, String
domainName) {
        // Create a new request to check the domain status.
        DescribeDomainRequest describeRequest = DescribeDomainRequest.builder()
                .domainName(domainName)
                .build();

        // Every 15 seconds, check whether the domain is processing.
        DescribeDomainResponse describeResponse =
client.describeDomain(describeRequest);
        while (describeResponse.domainStatus().processing()) {
            try {
                System.out.println("Domain still processing...");
                TimeUnit.SECONDS.sleep(15);
                describeResponse = client.describeDomain(describeRequest);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Once we exit that loop, the domain is available
        System.out.println("Amazon OpenSearch Service has finished processing
changes for your domain.");
```

```
        System.out.println("Domain description: "+describeResponse.toString());
    }
}
```

Version 1

この例では AWSElasticsearchClientBuilder を使用して、 AWS SDK for Java のバージョン 1 からレガシーm p Elasticsearch ドメインの作成、その設定の更新、および削除を行います。waitForDomainProcessing への呼び出しのコメント (および deleteDomain への呼び出しのコメント) を削除し、ドメインをオンラインにして使用できるようにします。

```
package com.amazonaws.samples;

import java.util.concurrent.TimeUnit;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.elasticsearch.AWSElasticsearch;
import com.amazonaws.services.elasticsearch.AWSElasticsearchClientBuilder;
import com.amazonaws.services.elasticsearch.model.CreateElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.CreateElasticsearchDomainResult;
import com.amazonaws.services.elasticsearch.model.DeleteElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.DeleteElasticsearchDomainResult;
import
 com.amazonaws.services.elasticsearch.model.DescribeElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.DescribeElasticsearchDomainResult;
import com.amazonaws.services.elasticsearch.model.EBSOptions;
import com.amazonaws.services.elasticsearch.model.ElasticsearchClusterConfig;
import com.amazonaws.services.elasticsearch.model.ResourceNotFoundException;
import
 com.amazonaws.services.elasticsearch.model.UpdateElasticsearchDomainConfigRequest;
import
 com.amazonaws.services.elasticsearch.model.UpdateElasticsearchDomainConfigResult;
import com.amazonaws.services.elasticsearch.model.VolumeType;

/**
 * Sample class demonstrating how to use the Amazon Web Services SDK for Java to
 create, update,
 * and delete Amazon OpenSearch Service domains.
 */

public class OpenSearchSample {

    public static void main(String[] args) {
```

```
        final String domainName = "my-test-domain";

        // Build the client using the default credentials chain.
        // You can use the CLI and run `aws configure` to set access key, secret
        // key, and default region.
        final AWSElasticsearch client = AWSElasticsearchClientBuilder
                .standard()
                // Unnecessary, but lets you use a region different than your
default.
                .withRegion(Regions.US_WEST_2)
                // Unnecessary, but if desired, you can use a different provider
chain.
                .withCredentials(new DefaultAWSCredentialsProviderChain())
                .build();

        // Create a new domain, update its configuration, and delete it.
        createDomain(client, domainName);
        // waitForDomainProcessing(client, domainName);
        updateDomain(client, domainName);
        // waitForDomainProcessing(client, domainName);
        deleteDomain(client, domainName);
    }

    /**
     * Creates an Amazon OpenSearch Service domain with the specified options.
     * Some options require other Amazon Web Services resources, such as an Amazon
Cognito user pool
     * and identity pool, whereas others require just an instance type or instance
     * count.
     *
     * @param client
     *            The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *            The name of the domain you want to create
     */
    private static void createDomain(final AWSElasticsearch client, final String
domainName) {

        // Create the request and set the desired configuration options
        CreateElasticsearchDomainRequest createRequest = new
CreateElasticsearchDomainRequest()
                .withDomainName(domainName)
                .withElasticsearchVersion("7.10")
```

```
                .withElasticsearchClusterConfig(new ElasticsearchClusterConfig()
                        .withDedicatedMasterEnabled(true)
                        .withDedicatedMasterCount(3)
                        // Small, inexpensive instance types for testing. Not
 recommended for production
                        // domains.
                        .withDedicatedMasterType("t2.small.elasticsearch")
                        .withInstanceType("t2.small.elasticsearch")
                        .withInstanceCount(5))
                // Many instance types require EBS storage.
                .withEBSOptions(new EBSOptions()
                        .withEBSEnabled(true)
                        .withVolumeSize(10)
                        .withVolumeType(VolumeType.Gp2));
                // You can uncomment this line and add your account ID, a username,
 and the
                // domain name to add an access policy.
                // .withAccessPolicies("{\"Version\":\"2012-10-17\",
\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
[\"arn:aws:iam::123456789012:user/user-name\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:region:123456789012:domain/domain-name/*\"}]}")


        // Make the request.
        System.out.println("Sending domain creation request...");
        CreateElasticsearchDomainResult createResponse =
 client.createElasticsearchDomain(createRequest);
        System.out.println("Domain creation response from Amazon OpenSearch
 Service:");
        System.out.println(createResponse.getDomainStatus().toString());
    }

    /**
     * Updates the configuration of an Amazon OpenSearch Service domain with the
     * specified options. Some options require other Amazon Web Services resources,
 such as an
     * Amazon Cognito user pool and identity pool, whereas others require just an
     * instance type or instance count.
     *
     * @param client
     *            The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *            The name of the domain to update
     */
```

```
   private static void updateDomain(final AWSElasticsearch client, final String
domainName) {
       try {
           // Updates the domain to use three data instances instead of five.
           // You can uncomment the Cognito lines and fill in the strings to enable
Cognito
           // authentication for OpenSearch Dashboards.
           final UpdateElasticsearchDomainConfigRequest updateRequest = new
UpdateElasticsearchDomainConfigRequest()
                   .withDomainName(domainName)
                   // .withCognitoOptions(new CognitoOptions()
                           // .withEnabled(true)
                           // .withUserPoolId("user-pool-id")
                           // .withIdentityPoolId("identity-pool-id")
                           // .withRoleArn("role-arn")
                   .withElasticsearchClusterConfig(new ElasticsearchClusterConfig()
                           .withInstanceCount(3));

           System.out.println("Sending domain update request...");
           final UpdateElasticsearchDomainConfigResult updateResponse = client
                   .updateElasticsearchDomainConfig(updateRequest);
           System.out.println("Domain update response from Amazon OpenSearch
Service:");
           System.out.println(updateResponse.toString());
       } catch (ResourceNotFoundException e) {
           System.out.println("Domain not found. Please check the domain name.");
       }
   }

   /**
    * Deletes an Amazon OpenSearch Service domain. Deleting a domain can take
    * several minutes.
    *
    * @param client
    *            The client to use for the requests to Amazon OpenSearch Service
    * @param domainName
    *            The name of the domain that you want to delete
    */
   private static void deleteDomain(final AWSElasticsearch client, final String
domainName) {
       try {
           final DeleteElasticsearchDomainRequest deleteRequest = new
DeleteElasticsearchDomainRequest()
                   .withDomainName(domainName);
```

```
            System.out.println("Sending domain deletion request...");
            final DeleteElasticsearchDomainResult deleteResponse =
client.deleteElasticsearchDomain(deleteRequest);
            System.out.println("Domain deletion response from Amazon OpenSearch
Service:");
            System.out.println(deleteResponse.toString());
        } catch (ResourceNotFoundException e) {
            System.out.println("Domain not found. Please check the domain name.");
        }
    }

    /**
     * Waits for the domain to finish processing changes. New domains typically take
15-30 minutes
     * to initialize, but can take longer depending on the configuration. Most
updates to existing domains
     * take a similar amount of time. This method checks every 15 seconds and
finishes only when
     * the domain's processing status changes to false.
     *
     * @param client
     *            The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *            The name of the domain that you want to check
     */
    private static void waitForDomainProcessing(final AWSElasticsearch client, final
String domainName) {
        // Create a new request to check the domain status.
        final DescribeElasticsearchDomainRequest describeRequest = new
DescribeElasticsearchDomainRequest()
                .withDomainName(domainName);

        // Every 15 seconds, check whether the domain is processing.
        DescribeElasticsearchDomainResult describeResponse =
client.describeElasticsearchDomain(describeRequest);
        while (describeResponse.getDomainStatus().isProcessing()) {
            try {
                System.out.println("Domain still processing...");
                TimeUnit.SECONDS.sleep(15);
                describeResponse =
client.describeElasticsearchDomain(describeRequest);
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
            }
        }

        // Once we exit that loop, the domain is available
        System.out.println("Amazon OpenSearch Service has finished processing
 changes for your domain.");
        System.out.println("Domain description response from Amazon OpenSearch
 Service:");
        System.out.println(describeResponse.toString());
    }
}
```

Python

この例では、 の OpenSearchService 低レベル Python クライアントを使用して、ドメイン AWS SDK for Python (Boto) の作成、設定の更新、削除を行います。

```
import boto3
import botocore
from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

my_config = Config(
    # Optionally lets you specify a region other than your default.
    region_name='us-west-2'
)

client = boto3.client('opensearch', config=my_config)

domainName = 'my-test-domain'  # The name of the domain


def createDomain(client, domainName):
    """Creates an Amazon OpenSearch Service domain with the specified options."""
    response = client.create_domain(
        DomainName=domainName,
        EngineVersion='OpenSearch_1.0',
        ClusterConfig={
```

```
            'InstanceType': 't2.small.search',
            'InstanceCount': 5,
            'DedicatedMasterEnabled': True,
            'DedicatedMasterType': 't2.small.search',
            'DedicatedMasterCount': 3
        },
        # Many instance types require EBS storage.
        EBSOptions={
            'EBSEnabled': True,
            'VolumeType': 'gp2',
            'VolumeSize': 10
        },
        AccessPolicies="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"arn:aws:iam::123456789012:user/user-name\"]},\"Action\":
[\"es:*\"],\"Resource\":\"arn:aws:es:us-west-2:123456789012:domain/my-test-domain/*
\"}]}",
        NodeToNodeEncryptionOptions={
            'Enabled': True
        }
    )
    print("Creating domain...")
    print(response)


def updateDomain(client, domainName):
    """Updates the domain to use three data nodes instead of five."""
    try:
        response = client.update_domain_config(
            DomainName=domainName,
            ClusterConfig={
                'InstanceCount': 3
            }
        )
        print('Sending domain update request...')
        print(response)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found. Please check the domain name.')
        else:
            raise error


def deleteDomain(client, domainName):
```

```
    """Deletes an OpenSearch Service domain. Deleting a domain can take several
 minutes."""
    try:
        response = client.delete_domain(
            DomainName=domainName
        )
        print('Sending domain deletion request...')
        print(response)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found. Please check the domain name.')
        else:
            raise error


def waitForDomainProcessing(client, domainName):
    """Waits for the domain to finish processing changes."""
    try:
        response = client.describe_domain(
            DomainName=domainName
        )
        # Every 15 seconds, check whether the domain is processing.
        while response["DomainStatus"]["Processing"] == True:
            print('Domain still processing...')
            time.sleep(15)
            response = client.describe_domain(
                DomainName=domainName)

        # Once we exit the loop, the domain is available.
        print('Amazon OpenSearch Service has finished processing changes for your
 domain.')
        print('Domain description:')
        print(response)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found. Please check the domain name.')
        else:
            raise error


def main():
    """Create a new domain, update its configuration, and delete it."""
```

```
    createDomain(client, domainName)
    waitForDomainProcessing(client, domainName)
    updateDomain(client, domainName)
    waitForDomainProcessing(client, domainName)
    deleteDomain(client, domainName)
```

ノード

この例では、SDK for JavaScript in Node.js のバージョン 3 OpenSearch クライアントを使用して、ドメインの作成、その設定の更新、および削除を行います。

```
var {
    OpenSearchClient,
    CreateDomainCommand,
    DescribeDomainCommand,
    UpdateDomainConfigCommand,
    DeleteDomainCommand
} = require("@aws-sdk/client-opensearch");
var sleep = require('sleep');

var client = new OpenSearchClient();

var domainName = 'my-test-domain'

// Create a new domain, update its configuration, and delete it.
createDomain(client, domainName)
waitForDomainProcessing(client, domainName)
updateDomain(client, domainName)
waitForDomainProcessing(client, domainName)
deleteDomain(client, domainName)

async function createDomain(client, domainName) {
    // Creates an Amazon OpenSearch Service domain with the specified options.
    var command = new CreateDomainCommand({
        DomainName: domainName,
        EngineVersion: 'OpenSearch_1.0',
        ClusterConfig: {
        'InstanceType': 't2.small.search',
        'InstanceCount': 5,
        'DedicatedMasterEnabled': 'True',
        'DedicatedMasterType': 't2.small.search',
        'DedicatedMasterCount': 3
        },
```

```
        EBSOptions:{
            'EBSEnabled': 'True',
            'VolumeType': 'gp2',
            'VolumeSize': 10
        },
        AccessPolicies: "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"arn:aws:iam::123456789012:user/user-name\"]},\"Action\":
[\"es:*\"],\"Resource\":\"arn:aws:es:us-east-1:123456789012:domain/my-test-domain/*
\"}]}",
         NodeToNodeEncryptionOptions:{
            'Enabled': 'True'
        }
    });
    const response = await client.send(command);
    console.log("Creating domain...");
    console.log(response);
}

async function updateDomain(client, domainName) {
    // Updates the domain to use three data nodes instead of five.
    var command = new UpdateDomainConfigCommand({
        DomainName: domainName,
        ClusterConfig: {
        'InstanceCount': 3
        }
    });
    const response = await client.send(command);
    console.log('Sending domain update request...');
    console.log(response);
}

async function deleteDomain(client, domainName) {
    // Deletes an OpenSearch Service domain. Deleting a domain can take several
 minutes.
    var command = new DeleteDomainCommand({
        DomainName: domainName
    });
    const response = await client.send(command);
    console.log('Sending domain deletion request...');
    console.log(response);
}

async function waitForDomainProcessing(client, domainName) {
    // Waits for the domain to finish processing changes.
```

```
    try {
        var command = new DescribeDomainCommand({
            DomainName: domainName
        });
        var response = await client.send(command);

        while (response.DomainStatus.Processing == true) {
            console.log('Domain still processing...')
            await sleep(15000) // Wait for 15 seconds, then check the status again
            function sleep(ms) {
                return new Promise((resolve) => {
                    setTimeout(resolve, ms);
                });
            }
            var response = await client.send(command);
        }
        // Once we exit the loop, the domain is available.
        console.log('Amazon OpenSearch Service has finished processing changes for your
 domain.');
        console.log('Domain description:');
        console.log(response);

    } catch (error) {
        if (error.name === 'ResourceNotFoundException') {
            console.log('Domain not found. Please check the domain name.');
            }
    };
}
```

Amazon OpenSearch Service でのデータのインデックス作成

Amazon OpenSearch Service では REST API を使用するため、ドキュメントのインデックス作成にはさまざまな方法があります。curl などの標準クライアントを使用することも、HTTP リクエストの送信が可能な任意のプログラミング言語を使用することもできます。やり取りのプロセスをさらに単純化するため、OpenSearch Service には多くのプログラミング言語に対応するクライアントが用意されています。上級ユーザーは、直接「the section called “OpenSearch Service へのストリーミングデータをロードする”」に進むことができます。

Amazon OpenSearch Ingestion を使用してデータを取り込むことを強くお勧めします。これは OpenSearch Service 内に構築されたフルマネージド型のデータコレクターです。詳細については、「Amazon OpenSearch Ingestion」を参照してください。

インデックス作成の概要については、OpenSearch のドキュメントを参照してください。

インデックスの命名制限

OpenSearch Service インデックスには、以下の命名制限があります。

- すべての文字を小文字にする必要があります。
- インデックスの名前を _ または - から始めることはできません。
- インデックス名にスペース、カンマ、:、"、*、+、/、\、|、?、#、>、< を含めることはできません。

インデックス、タイプ、またはドキュメント ID 名に機密情報を含めないでください。OpenSearch Service は、これらの名前をその Uniform Resource Identifier (URI) で使用します。サーバーおよびアプリケーションは、HTTP リクエストを頻繁に記録します。このため、URI に機密情報が含まれている場合、不要なデータ漏えいにつながる可能性があります。

```
2018-10-03T23:39:43 198.51.100.14 200 "GET https://opensearch-domain/dr-jane-doe/flu-patients-2018/202-555-0100/ HTTP/1.1"
```

関連する JSON ドキュメントを表示する許可がない場合でも、Dr. Doe の患者 (電話番号は 202-555-0100) が 2018 年にインフルエンザにかかったことを、この架空の口グ行から推測することが可能です。

OpenSearch Service が、インデックス名に実際の IP アドレスまたは認識した IP アドレスを検出した場合 (my-index-12.34.56.78.91 など) では、IP アドレスをマスクします。_cat/indices を呼び出すと、以下のレスポンスが生成されます。

```
green open my-index-x.x.x.x.91    soY19tBERoKo71WcEScidw 5 1 0 0   2kb  1kb
```

不必要な混乱を避けるため、インデックス名には IP アドレスを含めないでください。

レスポンスサイズの削減

_index および _bulk API のレスポンスには、多くの情報が含まれています。この情報は、リクエストのトラブルシューティングや、再試行ロジックの実装には役立ちますが、帯域幅を広く使用する場合があります。この例では、32 バイトのドキュメントのインデックスを作成すると、レスポンスは 339 バイト (ヘッダーを含む) になります。

```
PUT opensearch-domain/more-movies/_doc/1
{"title": "Back to the Future"}
```

レスポンス

```
{
  "_index": "more-movies",
  "_type": "_doc",
  "_id": "1",
  "_version": 4,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "_seq_no": 3,
  "_primary_term": 1
}
```

このレスポンスサイズは一見小さく見えますが、1 日あたり 1,000,000 ドキュメント (1 秒あたり約 11.5 ドキュメントのインデックス) を作成した場合、339 バイト/1 レスポンスは、1 か月あたり 10.17 GB のダウンロードトラフィックになります。

データ転送コストが懸念される場合は、filter_path パラメータを使用して OpenSearch Service レスポンスのサイズを削減できますが、失敗したリクエストの識別や再試行のために必要なフィールドを除外しないように注意してください。このフィールドはクライアントによって異なります。filter_path パラメータは、すべての OpenSearch Service REST API で動作しますが、頻繁に呼び出す API (例: _index、_bulk API) を使用する場合に特に便利です。

```
PUT opensearch-domain/more-movies/_doc/1?filter_path=result,_shards.total
{"title": "Back to the Future"}
```

レスポンス

```
{
  "result": "updated",
  "_shards": {
    "total": 2
  }
}
```

フィールドを含める代わりに、- プリフィックスを使用してフィールドを除外することができます。filter_path は、ワイルドカードもサポートしています。

```
POST opensearch-domain/_bulk?filter_path=-took,-items.index._*
{ "index": { "_index": "more-movies", "_id": "1" } }
{"title": "Back to the Future"}
{ "index": { "_index": "more-movies", "_id": "2" } }
{"title": "Spirited Away"}
```

レスポンス

```
{
  "errors": false,
  "items": [
    {
      "index": {
        "result": "updated",
        "status": 200
      }
    },
    {
      "index": {
        "result": "updated",
```

```
        "status": 200
      }
    }
  ]
}
```

インデックスコーデック

インデックスコーデックは、インデックスに格納されているフィールドを圧縮してディスクに保存する方法を決定します。インデックスコーデックは、圧縮アルゴリズムを指定する静的 `index.codec` 設定によって制御されます。この設定は、インデックスのシャードサイズと操作のパフォーマンスに影響します。

サポートされているコーデックのリストとパフォーマンス特性については、OpenSearch ドキュメントの「サポートされているコーデック」を参照してください。

インデックスコーデックを選択するときは、次の点を考慮してください。

- 既存のインデックスのコーデック設定を変更する手間を省くために、新しいコーデック設定を使用する前に、本番環境以外の環境で代表的なワークロードをテストしてください。詳細については、「インデックスコーデックを変更する」を参照してください。
- Zstandard 圧縮コーデック (`"index.codec": "zstd"` または `"index.codec": "zstd_no_dict"`) を k-NN とセキュリティ分析インデックスに使用することはできません。

Amazon OpenSearch Service へのストリーミングデータをロードする

OpenSearch Ingestion を使用すると、サードパーティのソリューションを使用しなくても、ストリーミングデータを Amazon OpenSearch Service ドメインに直接ロードできます。OpenSearch Ingestion にデータを送信するには、データプロデューサーを設定します。そうすると、サービスによって、指定したドメインまたはコレクションにデータが自動的に配信されます。OpenSearch Ingestion の使用を開始するには、「the section called “チュートリアル: コレクションにデータを取り込む”」を参照してください。

OpenSearch Service のサポートが組み込まれている Amazon Data Firehose や Amazon CloudWatch Logs など、他のソースを使用してストリーミングデータをロードすることもできます。Amazon S3、Amazon Kinesis Data Streams、Amazon DynamoDB などの他のものでは、イベントハンドラ

として AWS Lambda 関数を使用します。Lambda 関数は、データを処理してドメインにストリーミングすることで、新しいデータに応答します。

> **Note**
>
> Lambda では、いくつかの一般的なプログラミング言語がサポートされ、ほとんどの AWS リージョンで利用できます。詳細については、「AWS Lambda デベロッパーガイド」の「Getting started with Lambda」、および「AWS 全般のリファレンス」の「AWS service endpoints」を参照してください。

OpenSearch Ingestion からのストリーミングデータのロード

Amazon OpenSearch Ingestion を使用して OpenSearch Service ドメインにデータをロードできます。OpenSearch Ingestion にデータを送信するようにデータプロデューサーを設定すると、指定したコレクションにデータが自動的に送信されます。また、OpenSearch Ingestion で、送信前にデータを変換するように設定することもできます。詳細については、「Amazon OpenSearch Ingestion」を参照してください。

Amazon S3 からストリーミングデータをロードする

Lambda を使用して、Amazon S3 から OpenSearch Service ドメインにデータを送信できます。S3 バケットに到着する新しいデータにより、Lambda へのイベント通知がトリガーされた後、インデックス作成を実行するカスタムコードが実行されます。

このデータのストリーミング方法には非常に柔軟性があります。オブジェクトメタデータのインデックスを作成したり、オブジェクトがプレーンテキストの場合は、オブジェクト本文の要素を解析してインデックス作成したりすることができます。このセクションでは、正規表現を使用してログファイルを解析し、一致をインデックス作成するシンプルな Python サンプルコードがあります。

前提条件

続行する前に、以下のリソースが必要です。

前提条件	説明
Amazon S3 バケット	詳細については、Amazon Simple Storage Service ユーザーガイドの「最初の S3 バケットを作成する」を参照してください。バケット

前提条件	説明
	は、OpenSearch Service ドメインと同じリージョンに存在する必要があります。
OpenSearch Service ドメイン	Lambda 関数により処理された後のデータのターゲット。詳細については、「the section called “OpenSearch Service ドメインの作成”」を参照してください。

Lambda デプロイパッケージを作成する

デプロイパッケージは、コードとその依存関係を含む ZIP または JAR ファイルです。このセクションには、Python サンプルコードがあります。他のプログラミング言語については、AWS Lambda デベロッパーガイドの「Lambda デプロイパッケージ」を参照してください。

1. ディレクトリを作成します。このサンプルでは、名前 `s3-to-opensearch` を使用します。
2. `sample.py` という名前のディレクトリ内にファイルを作成します。

```
import boto3
import re
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
index = 'lambda-s3-index'
datatype = '_doc'
url = host + '/' + index + '/' + datatype

headers = { "Content-Type": "application/json" }

s3 = boto3.client('s3')

# Regular expressions used to parse some simple log lines
ip_pattern = re.compile('(\d+\.\d+\.\d+\.\d+)')
```

```
time_pattern = re.compile('\[(\d+\/\w\w\w\/\d\d\d\d:\d\d:\d\d:\d\d\s-\d\d\d\d)\]')
message_pattern = re.compile('\"(.+)\"')

# Lambda execution starts here
def handler(event, context):
    for record in event['Records']:

        # Get the bucket name and key for the new file
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

        # Get, read, and split the file into lines
        obj = s3.get_object(Bucket=bucket, Key=key)
        body = obj['Body'].read()
        lines = body.splitlines()

        # Match the regular expressions to each line and index the JSON
        for line in lines:
            line = line.decode("utf-8")
            ip = ip_pattern.search(line).group(1)
            timestamp = time_pattern.search(line).group(1)
            message = message_pattern.search(line).group(1)

            document = { "ip": ip, "timestamp": timestamp, "message": message }
            r = requests.post(url, auth=awsauth, json=document, headers=headers)
```

`region` と `host` の変数を編集します。

3. まだ持っていない場合、pip をインストールしてから、依存関係を新しい package ディレクトリにインストールします。

```
cd s3-to-opensearch

pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

すべての Lambda 実行環境に Boto3 がインストールされているため、デプロイパッケージに含める必要はありません。

4. アプリケーションコードや相互依存性をパッケージ化します。

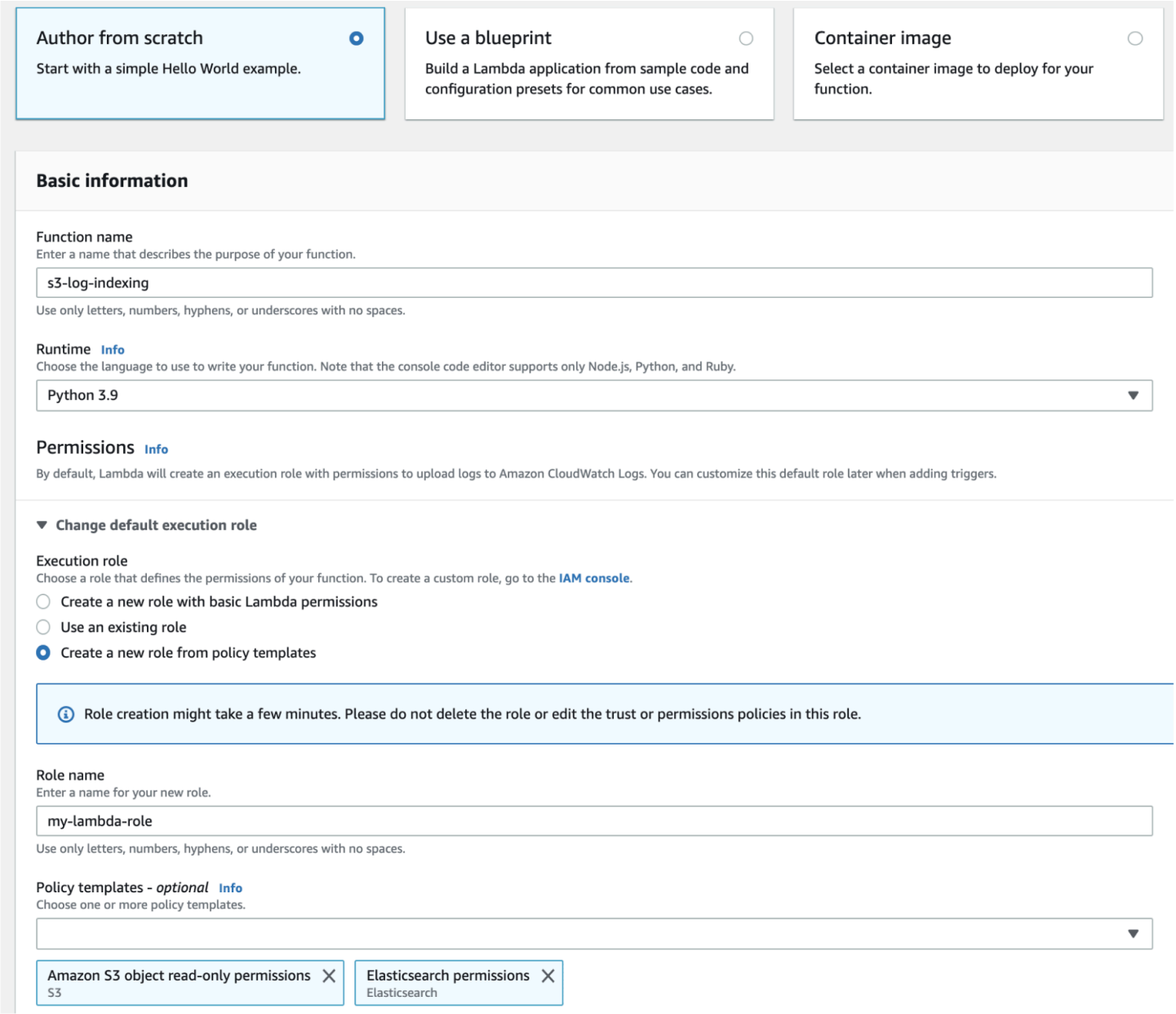
```
cd package
zip -r ../lambda.zip .
```

```
cd ..
zip -g lambda.zip sample.py
```

Lambda 関数を作成する

デプロイパッケージを作成すると、Lambda 関数を作成できます。関数を作成するとき、名前、ランタイム (たとえば、Python 3.8)、IAM ロールを選択します。IAM ロールにより、関数の許可が定義されます。詳細な手順については、AWS Lambda デベロッパーガイドの「コンソールで Lambda 関数を作成する」を参照してください。

この例では、コンソールを使用していることを前提としています。次のスクリーンショットに示すように、Python 3.9 と、S3 読み取り許可および OpenSearch Service 書き込み許可を持つロールを選択します。



関数を作成した後、トリガーを追加する必要があります。この例では、S3 バケットにログファイルが到着するたびにコードを実行します。

1. [トリガーの追加] を選択し、[S3] を選択します。
2. バケットを選択します。
3. [イベントタイプ] で、[PUT] を選択します。
4. [プレフィックス] に `logs/` と入力します。
5. [サフィックス] では、`.log` と入力します。
6. 再帰呼び出しの警告を確認し、[追加] を選択します。

最後に、デプロイパッケージをアップロードすることができます。

1. [~ からアップロード] および [.zip ファイルをアップロード] を選択してから、デプロイパッケージをアップロードするプロンプトに従います。
2. アップロードが終了したら、[ランタイム設定] を編集し、[ハンドラ] を `sample.handler` に変更します。この設定により、トリガー後に実行するファイル (`sample.py`) およびメソッド (`handler`) が Lambda に通知されます。

この時点で、リソースがすべて揃いました (ログファイルのバケット、ログファイルがバケットに追加されるたびに実行される関数、解析とインデックス作成を実行するコード、検索および可視化のための OpenSearch Service ドメイン)。

Lambda 関数をテストする

関数を作成した後、Amazon S3 バケットにファイルをアップロードしてテストできます。次のサンプルログの行を使用して、`sample.log` という名前のファイルを作成します。

```
12.345.678.90 - [10/Oct/2000:13:55:36 -0700] "PUT /some-file.jpg"
12.345.678.91 - [10/Oct/2000:14:56:14 -0700] "GET /some-file.jpg"
```

S3 バケットの `logs` フォルダにファイルをアップロードします。手順については、Amazon Simple Storage Service ユーザーガイドの「バケットにオブジェクトをアップロードする」を参照してください。

次に、OpenSearch Service コンソールまたは OpenSearch Dashboards を使用して、`lambda-s3-index` インデックスに 2 つのドキュメントが含まれていることを確認します。標準検索リクエストを行うこともできます。

```
GET https://domain-name/lambda-s3-index/_search?pretty
{
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "lambda-s3-index",
        "_type" : "_doc",
        "_id" : "vTYXaWIBJWV_TTkEuSDg",
        "_score" : 1.0,
```

```
        "_source" : {
          "ip" : "12.345.678.91",
          "message" : "GET /some-file.jpg",
          "timestamp" : "10/Oct/2000:14:56:14 -0700"
        }
      },
      {
        "_index" : "lambda-s3-index",
        "_type" : "_doc",
        "_id" : "vjYmaWIBJWV_TTkEuCAB",
        "_score" : 1.0,
        "_source" : {
          "ip" : "12.345.678.90",
          "message" : "PUT /some-file.jpg",
          "timestamp" : "10/Oct/2000:13:55:36 -0700"
        }
      }
    ]
  }
}
```

Amazon Kinesis Data Streams からストリーミングデータをロードする

ストリーミングデータは、Kinesis Data Streams から OpenSearch Service にロードできます。データストリームに到達する新しいデータによって、Lambda へのイベント通知がトリガーされ、インデックス作成を実行するカスタムコードが実行されます。このセクションには、いくつかの簡単な Python サンプルコードがあります。

前提条件

続行する前に、以下のリソースが必要です。

前提条件	説明
Amazon Kinesis Data Stream	Lambda 関数のイベントソース。詳細については、「Kinesis Data Streams」を参照してください。
OpenSearch Service ドメイン	Lambda 関数により処理された後のデータのターゲット。詳細については、「the section called “OpenSearch Service ドメインの作成”」を参照してください。

前提条件	説明

IAM ロール	このロールには、以下のように OpenSearch Service、Kinesis、Lambda の基本的な許可が必要です。
	
	JSON
	
	`{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "es:ESHttpPost", "es:ESHttpPut", "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents", "kinesis:GetShardIterator", "kinesis:GetRecords", "kinesis:DescribeStream", "kinesis:ListStreams" ], "Resource": "*" } ] }`
	
	ロールには、次の信頼関係が必要です。
	
	JSON
	
	`{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" },`

前提条件	説明
	"Action": "sts:AssumeRole" }] } 詳細については、IAM ユーザーガイドの「IAM ロールの作成」を参照してください。

Lambda 関数を作成する

「the section called “Lambda デプロイパッケージを作成する”」の手順に従いますが、kinesis-to-opensearch という名前のディレクトリを作成し、sample.py に次のコードを使用します。

```
import base64
import boto3
import json
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
index = 'lambda-kine-index'
datatype = '_doc'
url = host + '/' + index + '/' + datatype + '/'

headers = { "Content-Type": "application/json" }

def handler(event, context):
    count = 0
    for record in event['Records']:
        id = record['eventID']
        timestamp = record['kinesis']['approximateArrivalTimestamp']
```

```
        # Kinesis data is base64-encoded, so decode here
        message = base64.b64decode(record['kinesis']['data'])

        # Create the JSON document
        document = { "id": id, "timestamp": timestamp, "message": message }
        # Index the document
        r = requests.put(url + id, auth=awsauth, json=document, headers=headers)
        count += 1
    return 'Processed ' + str(count) + ' items.'
```

region と host の変数を編集します。

まだ持っていない場合、pip をインストールしてから、次のコマンドを使用して、依存関係をインストールします。

```
cd kinesis-to-opensearch

pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

次に、「the section called “Lambda 関数を作成する”」の手順に従いますが、「the section called “前提条件”」の IAM ロールと、トリガーの以下の設定を指定します。

- [Kinesis ストリーム]: Kinesis ストリーム
- [バッチサイズ]: 100
- [開始位置]: 水平トリム

詳細については、Amazon Kinesis Data Streams デベロッパーガイドの「Amazon Kinesis Data Streams とは」を参照してください。

この時点で、リソースがすべて揃いました (Kinesis データストリーム、ストリームが新しいデータを受信してそのデータのインデックスを作成した後に実行する関数、検索および可視化のための OpenSearch Service ドメイン)。

Lambda 関数をテストする

関数を作成した後、 AWS CLIを使用してデータストリームに新しいレコードを追加することでテストできます。

```
aws kinesis put-record --stream-name test --data "My test data." --partition-key
 partitionKey1 --region us-west-1
```

次に、OpenSearch Service コンソールまたは OpenSearch Dashboards を使用して、lambda-kine-index にドキュメントが含まれていることを確認します。以下のリクエストを使用することもできます。

```
GET https://domain-name/lambda-kine-index/_search
{
  "hits" : [
    {
      "_index": "lambda-kine-index",
      "_type": "_doc",
      "_id":
 "shardId-000000000000:495835116157626994950129608214214566865294366804960870 42",
      "_score": 1,
      "_source": {
        "timestamp": 1523648740.051,
        "message": "My test data.",
        "id":
 "shardId-000000000000:495835116157626994950129608214214566865294366804960870 42"
      }
    }
  ]
}
```

Amazon DynamoDB テーブルからストリーミングデータをロードする

を使用して AWS Lambda 、Amazon DynamoDB から OpenSearch Service ドメインにデータを送信できます。データテーブルに到着する新しいデータにより、Lambda へのイベント通知がトリガーされた後、インデックス作成を実行するカスタムコードが実行されます。

前提条件

続行する前に、以下のリソースが必要です。

前提条件	説明
DynamoDB テーブル	このテーブルにはソースデータが含まれています。詳細については、Amazon DynamoDB デベロッパーガイドの「DynamoDB テーブルの基本的なオペレーション」を参照してください。 このテーブルは、OpenSearch Service ドメインと同じリージョンに存在している必要があり、ストリームが [新しいイメージ] に設定されている必要があります。詳細については、「ストリームの有効化」を参照してください。
OpenSearch Service ドメイン	Lambda 関数により処理された後のデータのターゲット。詳細については、「the section called “OpenSearch Service ドメインの作成”」を参照してください。

IAM ロール	このロールには、以下のように OpenSearch Service、DynamoDB、Lambda の基本的な実行許可が必要です。
	
	JSON
	
	{
	"Version":"2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"es:ESHttpPost",
	"es:ESHttpPut",
	"dynamodb:DescribeStream",
	"dynamodb:GetRecords",
	"dynamodb:GetShardIterator",
	"dynamodb:ListStreams",
	"logs:CreateLogGroup",
	"logs:CreateLogStream",
	"logs:PutLogEvents"
	],
	"Resource": "*"
	}
	]
	}

前提条件	説明
	ロールには、次の信頼関係が必要です。 JSON `{ "Version":"2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }` 詳細については、IAM ユーザーガイドの「IAM ロールの作成」を参照してください。

Lambda 関数を作成する

「the section called “Lambda デプロイパッケージを作成する”」の手順に従いますが、ddb-to-opensearch という名前のディレクトリを作成し、sample.py に次のコードを使用します。

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-east-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
```

```
index = 'lambda-index'
datatype = '_doc'
url = host + '/' + index + '/' + datatype + '/'

headers = { "Content-Type": "application/json" }

def handler(event, context):
    count = 0
    for record in event['Records']:
        # Get the primary key for use as the OpenSearch ID
        id = record['dynamodb']['Keys']['id']['S']

        if record['eventName'] == 'REMOVE':
            r = requests.delete(url + id, auth=awsauth)
        else:
            document = record['dynamodb']['NewImage']
            r = requests.put(url + id, auth=awsauth, json=document, headers=headers)
        count += 1
    return str(count) + ' records processed.'
```

region と host の変数を編集します。

まだ持っていない場合、pip をインストールしてから、次のコマンドを使用して、依存関係をインストールします。

```
cd ddb-to-opensearch

pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

次に、「the section called “Lambda 関数を作成する”」の手順に従いますが、「the section called “前提条件”」の IAM ロールと、トリガーの以下の設定を指定します。

- [テーブル]: DynamoDB テーブル
- [バッチサイズ]: 100
- [開始位置]: 水平トリム

詳細については、Amazon DynamoDB デベロッパーガイドの「DynamoDB Streams と Lambda を用いた新しい項目の処理」を参照してください。

この時点で、すべてのリソースが揃いました (ソースデータの DynamoDB テーブル、テーブルに対する DynamoDB ストリームの変更、ソースデータが変更されてそれらの変更のインデックスを作成した後に実行される関数、検索および可視化のための OpenSearch Service ドメイン)。

Lambda 関数をテストする

関数を作成した後、 AWS CLIを使用して DynamoDB テーブルに新しい項目を追加することでテストできます。

```
aws dynamodb put-item --table-name test --item '{"director": {"S": "Kevin
 Costner"},"id": {"S": "00001"},"title": {"S": "The Postman"}}' --region us-west-1
```

次に、OpenSearch Service コンソールまたは OpenSearch Dashboards を使用して、`lambda-index` にドキュメントが含まれていることを確認します。以下のリクエストを使用することもできます。

```
GET https://domain-name/lambda-index/_doc/00001
{
    "_index": "lambda-index",
    "_type": "_doc",
    "_id": "00001",
    "_version": 1,
    "found": true,
    "_source": {
        "director": {
            "S": "Kevin Costner"
        },
        "id": {
            "S": "00001"
        },
        "title": {
            "S": "The Postman"
        }
    }
}
```

Amazon Data Firehose からストリーミングデータをロードする

Firehose は配信先として OpenSearch Service をサポートしています。OpenSearch Service にストリーミングデータをロードする方法については、「Amazon Data Firehose Developer Guide」の

「Creating a Kinesis Data Firehose Delivery Stream」および「Choose OpenSearch Service for Your Destination」を参照してください。

データを OpenSearch Service にロードする前に、データの変換が必要になる場合があります。Lambda 関数を使用してこのタスクを実行する方法については、このガイドの「Amazon Kinesis Data Firehose データ変換」を参照してください。

配信ストリームを設定する場合、Firehose には「ワンクリック」の IAM ロールが用意されています。このロールにより、OpenSearch Service へのデータ送信、Amazon S3 でのデータバックアップ、および Lambda の使用によるデータ変換に必要なリソースアクセス権が付与されます。このようなロールを手動で作成する作業は複雑になるため、用意されているロールの使用をお勧めします。

Amazon CloudWatch からストリーミングデータをロードする

CloudWatch Logs サブスクリプションを使用して、ストリーミングデータを CloudWatch Logs から OpenSearch Service ドメインにロードできます。Amazon CloudWatch サブスクリプションの詳細については、「サブスクリプションを用いたログデータのリアルタイム処理」を参照してください。設定情報については、Amazon CloudWatch デベロッパーガイドの「CloudWatch Logs データを Amazon OpenSearch Service にストリーミング」を参照してください。

からのストリーミングデータのロード AWS IoT

ルール AWS IoT を使用して からデータを送信できます。詳細については、 AWS IoT デベロッパーガイドの「OpenSearch アクション」を参照してください。

Logstash を用いて Amazon OpenSearch Service へデータをロードする

Logstash のオープンソース版 (Logstash OSS) には、一括 API を使用して Amazon OpenSearch Service ドメインにデータをアップロードする便利な方法があります。このサービスは、Amazon S3 入力プラグインを含む、すべての標準 Logstash 入力プラグインをサポートします。OpenSearch Service は、基本認証と IAM 認証情報の両方をサポートする logstash-output-opensearch 出力プラグインをサポートします。プラグインはバージョン 8.1 以前の Logstash OSS で動作します。

設定

Logstash の設定は、ドメインが使用する認証のタイプによって異なります。

どの認証方法を使用していても、設定ファイルの出力セクションで `ecs_compatibility` を `disabled` に設定する必要があります。Logstash 8.0 では、デフォルトで ECS 互換モードですべてのプラグインが実行される画期的な変更が導入されました。レガシー動作を維持するには、デフォルト値をオーバーライドする必要があります。

きめ細かなアクセス制御の設定

OpenSearch Service ドメインで HTTP Basic 認証によるきめ細かなアクセスコントロールを使用する場合、設定は他の OpenSearch クラスターと同様です。この設定ファイルの例では、Filebeat のオープンソース版 (Filebeat OSS) から入力を受け取ります。

```
input {
  beats  {
    port => 5044
  }
}

output {
  opensearch {
    hosts       => "https://domain-endpoint:443"
    user        => "my-username"
    password    => "my-password"
    index       => "logstash-logs-%{+YYYY.MM.dd}"
    ecs_compatibility => disabled
    ssl_certificate_verification => false
  }
}
```

設定は Beats アプリケーションとユースケースによって異なりますが、Filebeat OSS の設定は次のようになります。

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /path/to/logs/dir/*.log
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
setup.ilm.enabled: false
setup.ilm.check_exists: false
setup.template.settings:
```

```
  index.number_of_shards: 1
output.logstash:
  hosts: ["logstash-host:5044"]
```

IAM ロールの設定

ドメインで IAM ベースのドメインアクセスポリシーを使用する場合や IAM マスターユーザーによるきめ細かなアクセスコントロールを使用する場合、OpenSearch Service へのすべてのリクエストには IAM 認証情報を使用して署名する必要があります。次の ID ベースのポリシーは、ドメインのサブリソースへの、すべての HTTP リクエストを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

Logstash を設定するには、プラグインをその出力に使用するように、設定ファイルを変更します。この設定ファイルの例では、S3 バケット内のファイルから入力を受け取ります。

```
input {
  s3 {
    bucket => "amzn-s3-demo-"
    region => "us-east-1"
  }
}

output {
  opensearch {
    hosts => ["domain-endpoint:443"]
    auth_type => {
```

```
      type => 'aws_iam'
      aws_access_key_id => 'your-access-key'
      aws_secret_access_key => 'your-secret-key'
      region => 'us-east-1'
      }
      index  => "logstash-logs-%{+YYYY.MM.dd}"
      ecs_compatibility => disabled
  }
}
```

設定ファイル内で IAM 認証情報を提供したくない場合は、認証情報をエクスポート (または `aws configure` を実行) できます。

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token"
```

OpenSearch Service ドメインが VPC にある場合、Logstash OSS マシンは VPC に接続できる必要があり、また VPC セキュリティグループを介してドメインにアクセスできる必要があります。詳細については、「the section called “VPC ドメインのアクセスポリシーについて”」を参照してください。

Amazon OpenSearch Serviceでのデータの検索

Amazon OpenSearch Service でドキュメントを検索する一般的な方法がいくつかあります。これには、URI 検索とリクエストボディ検索が含まれます。OpenSearch Service は、カスタムパッケージ、SQL サポート、非同期検索など、検索エクスペリエンスを向上させる追加機能を提供します。包括的な OpenSearch 検索 API リファレンスについては、「OpenSearch ドキュメント」を参照してください。

> **Note**
>
> 次のサンプルリクエストは、OpenSearch API を使用します。一部のリクエストでは古い Elasticsearch バージョンを使用できない可能性があります。

トピック

- URI 検索
- リクエストボディ検索
- 検索結果のページ分割
- Dashboards Query Language
- Amazon OpenSearch Service でのパッケージのインポートと管理
- SQL を用いた Amazon OpenSearch Service のデータのクエリを行う
- Amazon OpenSearch Service のクロスクラスター検索
- Amazon OpenSearch Service 用の Learning to Rank
- Amazon OpenSearch Service の非同期検索
- Amazon OpenSearch Service のポイントインタイム検索
- Amazon OpenSearch Service でのエージェント検索
- Amazon OpenSearch Serviceでのセマンティック検索
- Amazon OpenSearch Service での同時セグメント検索
- Amazon OpenSearch Service における自然言語クエリ生成

URI 検索

Universal Resource Identifier (URI) 検索は、最もシンプルな検索方法です。URI 検索では、HTTP リクエストパラメータとしてクエリを指定します。

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/_search?q=house
```

以下にレスポンスの例を示します。

```
{
  "took": 25,
  "timed_out": false,
  "_shards": {
    "total": 10,
    "successful": 10,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 85,
      "relation": "eq",
    },
    "max_score": 6.6137657,
    "hits": [
      {
        "_index": "movies",
        "_type": "movie",
        "_id": "tt0077975",
        "_score": 6.6137657,
        "_source": {
          "directors": [
            "John Landis"
          ],
          "release_date": "1978-07-27T00:00:00Z",
          "rating": 7.5,
          "genres": [
            "Comedy",
            "Romance"
          ],
          "image_url": "http://ia.media-imdb.com/images/M/
MV5BMTY2OTQxNTc1OF5BMl5BanBnXkFtZTYwNjA3NjI5._V1_SX400_.jpg",
```

```
          "plot": "At a 1962 College, Dean Vernon Wormer is determined to expel the
 entire Delta Tau Chi Fraternity, but those troublemakers have other plans for him.",
          "title": "Animal House",
          "rank": 527,
          "running_time_secs": 6540,
          "actors": [
            "John Belushi",
            "Karen Allen",
            "Tom Hulce"
          ],
          "year": 1978,
          "id": "tt0077975"
        }
      },
      ...
    ]
  }
}
```

デフォルトでは、このクエリは家という語句をすべてのインデックスのすべてのフィールドで検索します。検索を絞り込むには URI でインデックス (movies) およびドキュメントフィールド (title) を指定します。

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search?q=title:house
```

リクエストに追加のパラメータを含めることができますが、サポートされているパラメータはごく一部分の OpenSearch 検索オプションのみを提供します。次のリクエストは 20 の結果 (デフォルトでは 10) を返し、年ごとに (_score ではなく) ソートされます。

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search?
q=title:house&size=20&sort=year:desc
```

リクエストボディ検索

より複雑な検索を実行するには、HTTP リクエストボディと、クエリの OpenSearch ドメイン固有言語 (DSL) を使用します。クエリ DSL では、OpenSearch 検索オプションの全範囲を指定することができます。

> **Note**
>
> テキストフィールドの値に Unicode 特殊文字を含めることはできません。値は特殊文字で区切られた複数の値として解析されます。このような誤った解析により、意図せずに文書がフィルタリングされ、アクセス制御が損なわれる可能性があります。詳しい情報については、OpenSearch ドキュメンテーションの「テキストフィールド内の Unicode 特殊文字に関する注意」を参照してください。

次の match クエリでは、最終的な URI 検索の例と似ています。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "sort": {
    "year": {
      "order": "desc"
    }
  },
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "house"
    }
  }
}
```

> **Note**
>
> _search API はリクエストボディ検索のための HTTP GET および POST を受け入れます。ただし、すべての HTTP クライアントが GET リクエストにリクエストボディを追加することをサポートしているわけではありません。POST は普遍的な選択です。

多くの場合、いくつかのフィールドを検索する必要がある場合がありますが、すべてのフィールドを検索する必要はありません。multi_match クエリを使用します。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
```

```
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title", "plot", "actors", "directors"]
    }
  }
}
```

ブーストフィールド

特定のフィールドをブーストすることで、検索の関連性を向上させることができます。ブーストは乗数で、あるフィールドの一致を他のフィールドの一致より重く重み付けします。次の例では、`title` フィールドでの john の一致は、`plot` フィールドの一致の 2 倍の度合いで、また `actors` もしくは `directors` フィールドの一致の 4 倍の度合いで `_score` に影響を及ぼします。その結果、John Wick や John Carter のような映画が検索結果の最上部にあり、John Travolta 主演の映画は最下部にあります。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "john",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  }
}
```

検索結果のハイライト

`highlight` オプションは、クエリが 1 つ以上のフィールドに一致した場合、`hits` 配列内にある追加のオブジェクトを返すよう OpenSearch に指示します。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
```

```
  },
  "highlight": {
    "fields": {
      "plot": {}
    }
  }
}
```

クエリが plot フィールドのコンテンツに一致した場合、ヒットは次のようになります。

```
{
  "_index": "movies",
  "_type": "movie",
  "_id": "tt0091541",
  "_score": 11.276199,
  "_source": {
    "directors": [
      "Richard Benjamin"
    ],
    "release_date": "1986-03-26T00:00:00Z",
    "rating": 6,
    "genres": [
      "Comedy",
      "Music"
    ],
    "image_url": "http://ia.media-imdb.com/images/M/
MV5BMTIzODEzODE2OF5BMl5BanBnXkFtZTcwNjQ3ODcyMQ@@._V1_SX400_.jpg",
    "plot": "A young couple struggles to repair a hopelessly dilapidated house.",
    "title": "The Money Pit",
    "rank": 4095,
    "running_time_secs": 5460,
    "actors": [
      "Tom Hanks",
      "Shelley Long",
      "Alexander Godunov"
    ],
    "year": 1986,
    "id": "tt0091541"
  },
  "highlight": {
    "plot": [
      "A young couple struggles to repair a hopelessly dilapidated <em>house</em>."
    ]
```

```
  }
}
```

デフォルトでは、OpenSearch は一致する文字列を タグで囲み、一致した文字列の前後 100 文字のコンテキストを提供します。また、区切り記号、スペース、タブ、改行を識別してコンテンツをセンテンスに分割します。これらの設定はすべてカスタマイズ可能です。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  },
  "highlight": {
    "fields": {
      "plot": {}
    },
    "pre_tags": "<strong>",
    "post_tags": "</strong>",
    "fragment_size": 200,
    "boundary_chars": ".,!? "
  }
}
```

Count API

ドキュメントの内容に関心がなく、一致の数だけを知りたい場合は、_search API の代わりに _count API を使用できます。次のリクエストでは、query_string クエリを使用してロマンチックコメディを識別します。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_count
{
  "query": {
    "query_string": {
      "default_field": "genres",
      "query": "romance AND comedy"
    }
  }
```

```
}
```

以下にレスポンスの例を示します。

```
{
  "count": 564,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  }
}
```

検索結果のページ分割

大量の検索結果を表示する必要がある場合は、さまざまな方法を使用してページネーションを実装できます。

ポイントインタイム

ポイントインタイム (PIT) 機能とは、時間が固定されているデータセットにさまざまなクエリを実行できる、検索の一種です。こちらは、OpenSearch、特にディープページネーションを行う場合に推奨される、ページネーションの方法です。PIT は、OpenSearch Service バージョン 2.5 以降で使用できます。PIT の詳細については「???」を参照してください。

`from` と `size` のパラメーター

ページ分割の最も簡単な方法は、`from` と `size` パラメータを使用した方法です。次のリクエストは、検索結果のゼロから始まるインデックスリストのうち、20～39 までの結果を返します。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "from": 20,
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
```

```
  }
}
```

検索ページネーションの詳細については、「OpenSearch ドキュメント」の「ページネーションの結果」を参照してください。

Dashboards Query Language

Dashboards Query Language (DQL) を使用して、OpenSearch Dashboards でデータと視覚化を検索できます。DQL は、用語、ブール値、日付と範囲、およびネストされたフィールドの 4 つの主要なクエリタイプを使用します。

用語のクエリ

用語クエリでは、検索する用語を指定する必要があります。

用語クエリを実行するには、次のように入力します。

```
host:www.example.com
```

ブール値クエリ

ブール演算子 AND、or、および not を使用して、複数のクエリを組み合わせることができます。

ブール値クエリを実行するには、次を貼り付けます。

```
host.keyword:www.example.com and response.keyword:200
```

日付と範囲のクエリ

日付と範囲のクエリを使用して、クエリの前後の日付を検索できます。

- > は、指定した日付より後の日付を検索することを示します。
- < は、指定した日付より前の日付を検索することを示します。

```
@timestamp > "2020-12-14T09:35:33"
```

ネストされたフィールドのクエリ

ネストされたフィールドを持つドキュメントがある場合は、ドキュメントのどの部分を取得するかを指定する必要があります。ネストされたフィールドを含むサンプルドキュメントを次に示します。

```
{"NBA players":[
    {"player-name": "Lebron James",
      "player-position": "Power forward",
      "points-per-game": "30.3"
    },
    {"player-name": "Kevin Durant",
      "player-position": "Power forward",
      "points-per-game": "27.1"
    },
    {"player-name": "Anthony Davis",
      "player-position": "Power forward",
      "points-per-game": "23.2"
    },
    {"player-name": "Giannis Antetokounmpo",
      "player-position": "Power forward",
      "points-per-game":"29.9"
    }
  ]
}
```

DQL を使用して特定のフィールドを取得するには、次を貼り付けます。

```
NBA players: {player-name: Lebron James}
```

ネストされたドキュメントから複数のオブジェクトを取得するには、次を貼り付けます。

```
NBA players: {player-name: Lebron James} and NBA players: {player-name: Giannis
 Antetokounmpo}
```

範囲内を検索するには、次を貼り付けます。

```
NBA players: {player-name: Lebron James} and NBA players: {player-name: Giannis
 Antetokounmpo and < 30}
```

別のオブジェクト内にネストされたオブジェクトがドキュメントにある場合でも、すべてのレベルを指定することでデータを取得できます。これを実行するには、次を貼り付けます。

```
Top-Power-forwards.NBA players: {player-name:Lebron James}
```

Amazon OpenSearch Service でのパッケージのインポートと管理

Amazon OpenSearch Service では、ストップワードや類義語などのカスタム辞書ファイルをアップロードして、プラグインをドメインに関連付けることができます。これらのプラグインは、パッケージ済み、カスタム、またはサードパーティーのいずれかであり、ドメインの機能を柔軟に拡張できます。これらすべてのタイプのファイルの総称は、パッケージです。

- 辞書ファイルは、よくある頻出語を無視したり、「フローズンカスタード」、「ジェラート」、「アイスクリーム」などの類似した用語を同等に扱ったりするように OpenSearch に指示することで、検索結果を改善します。また、日本語 (kuromoji) 分析プラグインで見られるように、ステミングを改善することもできます。
- パッケージ済みプラグインは、パーソナライズされた検索結果を実現する Amazon Personalize プラグインなどの組み込み機能を提供します。これらのプラグインは、`ZIP-PLUGIN` パッケージタイプを使用します。詳細については、「the section called “エンジンバージョンに応じたプラグイン”」を参照してください。
- カスタムプラグインおよびサードパーティープラグインを使用すると、カスタマイズされた機能を追加したり、外部システムと統合したりできるため、ドメインの柔軟性がさらに向上します。パッケージ済みプラグインと同様に、カスタムプラグインを `ZIP-PLUGIN` パッケージとしてアップロードします。サードパーティープラグインの場合は、プラグインのライセンスファイルと設定ファイルも個別のパッケージとしてインポートし、これらすべてをドメインに関連付ける必要があります。

  詳細については、以下の各トピックを参照してください。

  - the section called “カスタムプラグイン”
  - the section called “サードパーティープラグイン”

> **Note**
>
> 1 つのドメインには最大 20 個のプラグインを関連付けることができます。この制限には、オプションプラグイン、サードパーティープラグイン、カスタムプラグインを含む、すべてのプラグインタイプが含まれます。

トピック

- 必要なアクセス許可
- Amazon S3 へのパッケージのアップロード

- パッケージのインポートと関連付け
- OpenSearch でのパッケージの使用
- パッケージの更新
- 新しい辞書でインデックスを手動で更新する
- パッケージの関連付け解除と削除
- Amazon OpenSearch Service でのカスタムプラグイン管理
- Amazon OpenSearch Service でのサードパーティープラグインのインストール

必要なアクセス許可

管理者アクセスのないユーザーは、パッケージを管理するために特定の AWS Identity and Access Management (IAM) アクションが必要です。

- `es:CreatePackage` – パッケーを作成する
- `es:DeletePackage` – パッケージを削除する
- `es:AssociatePackage` – パッケージをドメインに関連付ける
- `es:DissociatePackage` – ドメインからパッケージの関連付けを解除する

カスタムパッケージが存在する Amazon S3 バケットパスまたはオブジェクトに対する許可も必要です。

ドメインアクセスポリシー内ではなく、IAM 内のすべての許可を付与します。詳細については、「the section called “Identity and Access Management”」を参照してください。

Amazon S3 へのパッケージのアップロード

このセクションでは、パッケージ済みプラグインパッケージは既にインストールされていることから、カスタム辞書パッケージをアップロードする方法について説明します。カスタム辞書は、ドメインに関連付ける前に Amazon S3 バケットにアップロードする必要があります。手順については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのアップロード」を参照してください。サポートされているプラグインをアップロードする必要はありません。

辞書に機密情報が含まれている場合は、アップロード時に S3 で管理されたキーによるサーバー側の暗号化を指定します。OpenSearch Service は、 AWS KMS キーを使用して保護する S3 上のファイルにアクセスできません。

ファイルをアップロードしたら、その S3 パスを書き留めます。パスの形式は `s3://amzn-s3-demo-bucket/file-path/file-name` です。

テスト目的で、次のシノニムファイルを使用できます。`synonyms.txt` と名前を付けて保存します。

```
danish, croissant, pastry
ice cream, gelato, frozen custard
sneaker, tennis shoe, running shoe
basketball shoe, hightop
```

Hunspell 辞書など一部の辞書では、複数のファイルを使用するため、ファイルシステム上に独自のディレクトリが必要になります。現時点では、OpenSearch Service では単一ファイルの辞書のみがサポートされています。

パッケージのインポートと関連付け

コンソールは、カスタム辞書を OpenSearch Service にインポートする最も簡単な方法です。Amazon S3 から辞書をインポートすると、OpenSearch Service はパッケージの独自のコピーを保存し、AES-256 を使用して、OpenSearch Service マネージドキーによりそのコピーを自動的に暗号化します。

オプションのプラグインは OpenSearch Service にあらかじめインストールされているので、自分でアップロードする必要はありませんが、プラグインをドメインに関連付ける必要があります。使用可能なプラグインは、コンソールの [パッケージ] 画面に一覧表示されます。

パッケージをドメインにインポートして関連付ける

1. Amazon OpenSearch Service コンソールで、[パッケージ] を選択します。
2. [パッケージのインポート] を選択します。
3. パッケージにわかりやすい名前を付けます。
4. ファイルへの S3 パスを指定し、[インポート] を選択します。
5. [パッケージ] 画面に戻ります。
6. パッケージのステータスが [使用可能] の場合は、パッケージを選択します。
7. [ドメインへの関連付け] を選択します。
8. ドメインを選択して、[次へ] を選択します。パッケージを確認し、[関連付け] を選択します。
9. ナビゲーションペインでドメインを選択し、[パッケージ] タブに進みます。

10. パッケージがカスタム辞書の場合、パッケージが [使用可能] になったら ID を書き留めておきます。[OpenSearch へのリクエスト] でファイルパスとして analyzers/*id* を使用します。

OpenSearch でのパッケージの使用

このセクションでは、カスタム辞書とパッケージ済みプラグインの両タイプのパッケージの使用方法について説明します。

カスタム辞書の使用

ファイルをドメインに関連付けた後は、トークナイザやトークンフィルターの作成時に、synonyms_path、stopwords_path、user_dictionary などのパラメータで使用できます。正確なパラメータは、オブジェクトによって異なります。synonyms_path と stopwords_path はいくつかのオブジェクトでサポートされますが、user_dictionary は kuromoji プラグイン専用です。

IK (中国語) 分析プラグインの場合、カスタム辞書ファイルをカスタムパッケージとしてアップロードしてドメインに関連付けることができます。アップロードしたカスタム辞書ファイルはプラグインによって自動的にピックアップされます。user_dictionary パラメータは必要ありません。ファイルがシノニムファイルの場合は、synonyms_path パラメータを使用します。

次の例では、シノニムファイルを新しいインデックスに追加します。

```
PUT my-index
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "my_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["my_filter"]
          }
        },
        "filter": {
          "my_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F111111111",
            "updateable": true
          }
```

```
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "standard",
        "search_analyzer": "my_analyzer"
      }
    }
  }
}
```

このリクエストは、標準トークナイザとシノニムトークンフィルターを使用するインデックスのカスタムアナライザーを作成します。

- トークナイザは、いくつかのルールのセットに基づいて、文字のストリームをトークン (通常は単語) に分割します。最も単純な例は、空白文字に遭遇するたびに、先行する文字をトークンに分割する空白トークナイザです。より複雑な例は標準トークナイザです。これは、一連の文法ベースのルールを使用して多くの言語で動作します。
- トークンフィルターは、トークンの追加、変更、削除を行います。たとえば、シノニムトークンフィルターは、シノニムリストで単語が見つかったときにトークンを追加します。ストップトークンフィルターは、ストップワードリスト内の単語を検出すると、トークンを削除します。

このリクエストは、マッピングにテキストフィールド (description) を追加し、その検索アナライザーとして新しいアナライザーを使用するよう OpenSearch に指示します。インデックスアナライザーとして、まだ標準アナライザーが使用されていることがわかります。

最後に、トークンフィルターの行 `"updateable": true` をメモします。このフィールドは検索アナライザーにのみ適用され、インデックスアナライザーには適用されず、後で検索アナライザーを更新したい場合に不可欠です。

テスト目的のために、インデックスにいくつかのドキュメントを追加します。

```
POST _bulk
{ "index": { "_index": "my-index", "_id": "1" } }
{ "description": "ice cream" }
{ "index": { "_index": "my-index", "_id": "2" } }
```

```
{ "description": "croissant" }
{ "index": { "_index": "my-index", "_id": "3" } }
{ "description": "tennis shoe" }
{ "index": { "_index": "my-index", "_id": "4" } }
{ "description": "hightop" }
```

次に、シノニムを使用して検索します。

```
GET my-index/_search
{
  "query": {
    "match": {
      "description": "gelato"
    }
  }
}
```

この場合、OpenSearch は次の応答を返します。

```
{
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 0.99463606,
    "hits": [{
      "_index": "my-index",
      "_type": "_doc",
      "_id": "1",
      "_score": 0.99463606,
      "_source": {
        "description": "ice cream"
      }
    }]
  }
}
```

> **Tip**
>
> 辞書ファイルは、そのサイズに比例して Java ヒープ領域を使用します。たとえば、2 GiB の辞書ファイルは、ノード上で 2 GiB のヒープ領域を消費する可能性があります。大きな

ファイルを使用する場合は、収容に十分なヒープ領域がノードにあることを確認してください。JVMMemoryPressure メトリクスをモニタリングし、必要に応じてクラスターをスケーリングします。

パッケージ済みプラグインの使用

OpenSearch Service では、プレインストールされているオプションの OpenSearch プラグインをドメインに関連付けて使用することができます。パッケージ済みプラグインパッケージは特定の OpenSearch バージョンと互換性があり、そのバージョンのドメインにのみ関連付けることができます。ドメインで利用可能なパッケージのリストには、そのドメインバージョンと互換性のある、サポートされているすべてのプラグインが含まれています。プラグインをドメインに関連付けると、ドメインへのインストールプロセスが開始されます。すると、OpenSearch Service にリクエストを送信する際にプラグインを参照して使用できるようになります。

プラグインの関連付けと解除には、ブルー/グリーンデプロイが必要です。詳細については、「the section called “通常は blue/green デプロイの原因となる変更”」を参照してください。

オプションのプラグインには、言語アナライザーやカスタマイズされた検索結果が含まれます。例えば、Amazon Personalize Search Ranking プラグインは、機械学習を使用してお客様の検索結果をパーソナライズします。このプラグインの詳細については、「OpenSearch の検索結果のパーソナライズ」を参照してください。サポートされているすべてのプラグインのリストについては、「the section called “エンジンバージョンに応じたプラグイン”」を参照してください。

Sudachi プラグイン

Sudachi プラグインの場合、ディクショナリファイルを再関連付けしても、すぐにドメインには反映されません。設定変更やその他の更新の一環として、次のブルー/グリーンデプロイがドメインで実行されると、辞書が更新されます。または、更新済みのデータで新しいインデックスを作成し、この新しいパッケージを使用して新しいインデックスを作成し、既存のインデックスを新しいインデックスに再インデックスしてから、古いインデックスを削除することもできます。インデックスの再作成方法を使用する場合は、トラフィックが中断されないようにインデックスエイリアスを使用してください。

さらに、Sudachi プラグインは CreatePackage API オペレーションでアップロードできるバイナリ Sudachi ディクショナリのみをサポートしています。ビルド済みのシステムディクショナリの詳細およびユーザーディクショナリのコンパイルプロセスについては、Sudachi のドキュメントを参照してください。

> **Note**
>
> バイナリディクショナリファイルを Amazon S3 にアップロードする場合は、S3 オブジェクトの Content-Type を に設定する必要がありますbinary/octet-stream。を使用するとapplication/octet-stream、パッケージのインポートが失敗します。

次の例は、Sudachi トークナイザでシステムディクショナリとユーザーディクショナリを使用する方法を示しています。これらのディクショナリは、タイプ TXT-DICTIONARY のカスタムパッケージとしてアップロードし、追加設定でパッケージ ID を指定する必要があります。

```
PUT sudachi_sample
{
  "settings": {
    "index": {
      "analysis": {
        "tokenizer": {
          "sudachi_tokenizer": {
            "type": "sudachi_tokenizer",
            "additional_settings": "{\"systemDict\": \"<system-dictionary-package-
id>\",\"userDict\": [\"<user-dictionary-package-id>\"]}"
        }
        },
        "analyzer": {
          "sudachi_analyzer": {
            "filter": ["my_searchfilter"],
            "tokenizer": "sudachi_tokenizer",
            "type": "custom"
          }
        },
        "filter":{
          "my_searchfilter": {
            "type": "sudachi_split",
            "mode": "search"
          }
        }
      }
    }
  }
}
```

パッケージの更新

パッケージ済みのプラグインパッケージはすでに更新されているため、このセクションではカスタム辞書パッケージの更新方法についてのみ説明します。Amazon S3 に新しいバージョンの辞書をアップロードしても、Amazon OpenSearch Service 上のパッケージは自動的に更新されません。OpenSearch Service はファイルの独自のコピーを保存するため、新しいバージョンを S3 にアップロードする場合は、ファイルを手動で更新する必要があります。

関連付けられた各ドメインはそのファイルの独自のコピーも保存します。検索動作を予測可能に保つために、ドメインは明示的に更新されるまで、現在のパッケージバージョンを引き続き使用します。カスタムパッケージを更新するには、 でファイルを変更し Amazon S3 Control、OpenSearch Service でパッケージを更新してから、更新を適用します。

コンソール

1. OpenSearch Service コンソールで、[パッケージ] を選択します。
2. パッケージを選択し、[更新] を選択します。
3. ファイルへの新しい S3 パスを指定し、[パッケージを更新] を選択します。
4. [パッケージ] 画面に戻ります。
5. パッケージのステータスが [使用可能] に変わったら、それを選択します。次に、関連付けられたドメインを 1 つ以上選択し、[更新の適用] を選択し、確定します。関連付けステータスが [アクティブ] に変わるまで待ちます。
6. 次の手順は、インデックスの設定方法によって異なります。

    - ドメインが OpenSearch または Elasticsearch 7.8 以降を実行していて、更新可能 フィールドが true に設定されている検索アナライザーのみを使用する場合は、それ以上アクションを実行する必要はありません。OpenSearch Service は、_plugins/_refresh_search_analyzers API を使用して、自動的にインデックスを更新します。
    - ドメインが Elasticsearch 7.7 以前を実行していて、インデックスアナライザーを使用しているか、または `updateable` フィールドを使用していない場合は、「the section called “新しい辞書でインデックスを手動で更新する”」を参照してください。

コンソールは最も簡単な方法ですが、 AWS CLI、 SDKs、または設定 API を使用して OpenSearch Service パッケージを更新することもできます。詳細については、「AWS CLI コマンドリファレンス」と「Amazon OpenSearch Service API リファレンス」を参照してください。

AWS SDK

コンソールでパッケージを手動で更新する代わりに、SDK を使用して更新プロセスを自動化できます。次のサンプル Python スクリプトは、新しいパッケージファイルを Amazon S3 にアップロードし、OpenSearch Service でパッケージを更新し、指定されたドメインに新しいパッケージを適用します。更新が正常に完了したことを確認した後、OpenSearch にサンプル呼び出しを行い、新しいシノニムが適用されたことを示します。

host、region、file_name、bucket_name、s3_key、package_id、domain_name、および query の値を指定する必要があります。

```
from requests_aws4auth import AWS4Auth
import boto3
import requests
import time
import json
import sys

host = ''  # The OpenSearch domain endpoint with https:// and a trailing slash. For
 example, https://my-test-domain.us-east-1.es.amazonaws.com/
region = ''  # For example, us-east-1
file_name = ''  # The path to the file to upload
bucket_name = ''  # The name of the S3 bucket to upload to
s3_key = ''  # The name of the S3 key (file name) to upload to
package_id = ''  # The unique identifier of the OpenSearch package to update
domain_name = ''  # The domain to associate the package with
query = ''  # A test query to confirm the package has been successfully updated

service = 'es'
credentials = boto3.Session().get_credentials()
client = boto3.client('opensearch')
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key,
                   region, service, session_token=credentials.token)


def upload_to_s3(file_name, bucket_name, s3_key):
    """Uploads file to S3"""
    s3 = boto3.client('s3')
    try:
        s3.upload_file(file_name, bucket_name, s3_key)
        print('Upload successful')
        return True
    except FileNotFoundError:
```

```
        sys.exit('File not found. Make sure you specified the correct file path.')


def update_package(package_id, bucket_name, s3_key):
    """Updates the package in OpenSearch Service"""
    print(package_id, bucket_name, s3_key)
    response = client.update_package(
        PackageID=package_id,
        PackageSource={
            'S3BucketName': bucket_name,
            'S3Key': s3_key
        }
    )
    print(response)


def associate_package(package_id, domain_name):
    """Associates the package to the domain"""
    response = client.associate_package(
        PackageID=package_id, DomainName=domain_name)
    print(response)
    print('Associating...')


def wait_for_update(domain_name, package_id):
    """Waits for the package to be updated"""
    response = client.list_packages_for_domain(DomainName=domain_name)
    package_details = response['DomainPackageDetailsList']
    for package in package_details:
        if package['PackageID'] == package_id:
            status = package['DomainPackageStatus']
            if status == 'ACTIVE':
                print('Association successful.')
                return
            elif status == 'ASSOCIATION_FAILED':
                sys.exit('Association failed. Please try again.')
            else:
                time.sleep(10)  # Wait 10 seconds before rechecking the status
                wait_for_update(domain_name, package_id)


def sample_search(query):
    """Makes a sample search call to OpenSearch"""
    path = '_search'
```

```
params = {'q': query}
url = host + path
response = requests.get(url, params=params, auth=awsauth)
print('Searching for ' + '"' + query + '"')
print(response.text)
```

> **Note**
>
> を使用してスクリプトを実行するときに「package not found」エラーが表示される場合は AWS CLI、SBoto3S3 が使用している可能性があります。aws configure を実行して正しいリージョンを指定するか、明示的にクライアントにそのリージョンを追加します。
>
> ```
> client = boto3.client('opensearch', region_name='us-east-1')
> ```

新しい辞書でインデックスを手動で更新する

手動によるインデックス更新はカスタム辞書にのみ適用され、パッケージ済みプラグインには適用されません。更新された辞書を使用するには、次の条件のいずれかを満たしている場合、インデックスを手動で更新する必要があります。

- ドメインは Elasticsearch 7.7 以前を実行しています。
- カスタムパッケージをインデックスアナライザーとして使用しています。
- 検索アナライザーとしてカスタムパッケージを使用していますが、[更新可能] フィールドは含まれていません。

新しいパッケージファイルでアナライザーを更新するには、次の 2 つのオプションがあります。

- 更新するインデックスを閉じて開きます。

```
POST my-index/_close
POST my-index/_open
```

- インデックスを再作成します。まず、更新されたシノニムファイル (または完全に新しいファイル) を使用するインデックスを作成します。UTF-8 のみがサポートされていることに注意してください。

```
PUT my-new-index
```

```
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "synonym_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["synonym_filter"]
          }
        },
        "filter": {
          "synonym_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F222222222"
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "synonym_analyzer"
      }
    }
  }
}
```

次に、古いインデックスをその新しいインデックスに再作成します。

```
POST _reindex
{
  "source": {
    "index": "my-index"
  },
  "dest": {
    "index": "my-new-index"
  }
}
```

インデックスアナライザーを頻繁に更新する場合は、インデックスエイリアスを使用して、最新のインデックスへの一貫したパスを維持します。

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "my-index",
        "alias": "latest-index"
      }
    },
    {
      "add": {
        "index": "my-new-index",
        "alias": "latest-index"
      }
    }
  ]
}
```

古いインデックスが必要ない場合は、削除します。

```
DELETE my-index
```

パッケージの関連付け解除と削除

カスタム辞書であれパッケージ済みプラグインであれ、ドメインからパッケージを切り離すと、新しいインデックスを作成するときにそのパッケージを使用できなくなります。パッケージの関連付けが解除されると、パッケージを使用していた既存のインデックスでは使用できなくなります。パッケージを関連付け解除する前に、どのインデックスからも削除する必要があります。削除しないと、関連付けの解除は失敗します。

コンソールは、ドメインからパッケージの関連付けを解除し、OpenSearch Service から削除する最も簡単な方法です。OpenSearch サービスからパッケージを削除しても、それはAmazon S3 上の元の場所から削除されません。

ドメインからパッケージの関連付けを解除する

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、[ドメイン] を選択します。
3. ドメインを選択してから、[パッケージ] タブに移動します。
4. パッケージを選択して、[アクション] を選択し、[関連付け解除] を選択します。選択内容を確認します。
5. パッケージがリストから消えるのを待ちます。ブラウザを更新することが必要な場合があります。
6. パッケージを他のドメインで使用する場合は、ここで停止します。パッケージの削除を続行するには (カスタム辞書の場合)、ナビゲーションペインで [パッケージ] を選択します。
7. パッケージを選択し、[削除] を選択します。

または、 AWS CLI、 SDKs、または 設定 API を使用して、パッケージの関連付けを解除および削除します。詳細については、「AWS CLI コマンドリファレンス」と「Amazon OpenSearch Service API リファレンス」を参照してください。

Amazon OpenSearch Service でのカスタムプラグイン管理

OpenSearch Service のカスタムプラグインを使用すると、言語分析、カスタムフィルタリング、ランキングなどの領域で OpenSearch 機能を拡張できるため、パーソナライズされた検索エクスペリエンスを作成できます。OpenSearch 用のカスタムプラグインは、`org.opensearch.plugins.Plugin` クラスを拡張し、`.zip` ファイルにパッケージ化することで開発できます。

現在、Amazon OpenSearch Service では以下のプラグイン拡張機能がサポートされています。

- AnalysisPlugin – カスタムアナライザー、文字トークナイザ、テキスト処理用のフィルターなどを追加して、分析機能を拡張します。
- SearchPlugin – カスタムクエリタイプ、類似度アルゴリズム、提案オプション、集計を使用して検索機能を強化します。
- MapperPlugin – OpenSearch でカスタムフィールドタイプとそのマッピング設定を作成し、さまざまなタイプのデータの保存方法とインデックス作成方法を定義できます。

- ScriptPlugin – OpenSearch にカスタムスクリプト機能を追加できます。例えば、検索またはインデックス作成中のスコアリング、ソート、フィールド値変換などのオペレーション用のカスタムスクリプトなどです。
- SearchPipelinePlugin – カスタム検索パイプライン機能を追加することで、Amazon OpenSearch Service の検索エクスペリエンスを調整できます。たとえば、調整ノードレベルで検索フェーズ間で実行されるカスタム`SearchPhaseResultsProcessor`実装、複数のスコアリング戦略を組み合わせるためのハイブリッド検索の再ランク付け、複数のデータソースの結果を集約するフェデレーティッド検索パイプラインなどです。

OpenSearch Service コンソールまたはカスタムパッケージの既存の API コマンドを使用して、プラグインをアップロードし、Amazon OpenSearch Service クラスターに関連付けることができます。DescribePackages コマンドを使用して、アカウント内のすべてのパッケージを記述し、OpenSearch バージョンやエラーの詳細などの詳細情報を表示することもできます。OpenSearch Service は、プラグインパッケージのバージョン互換性、セキュリティの脆弱性、許可されたプラグインオペレーションを検証します。カスタムパッケージの詳細については、「the section called “パッケージ”」を参照してください。

OpenSearch のバージョンと AWS リージョン サポート

カスタムプラグインは、以下の AWS リージョンで OpenSearch バージョン 2.15 を実行している OpenSearch Service ドメインでサポートされています。

- 米国東部 (オハイオ) (us-east-2)
- 米国東部 (バージニア北部) (us-east-1)
- 米国西部 (オレゴン) (us-west-2)
- アジアパシフィック (ムンバイ) (ap-south-1)
- アジアパシフィック (ソウル) (ap-northeast-2)
- アジアパシフィック (シンガポール) (ap-southeast-1)
- アジアパシフィック (シドニー) (ap-southeast-2)
- アジアパシフィック (東京) (ap-northeast-1)
- カナダ (中部) (ca-central-1)
- ヨーロッパ (フランクフルト) (eu-central-1)
- 欧州 (アイルランド) (eu-west-1)
- ヨーロッパ (ロンドン) (eu-west-2)
- 欧州 (パリ) (eu-west-3)

- 南米 (サンパウロ) (sa-east-1)

> **Note**
>
> カスタムプラグインには、ユーザーが開発したコードが含まれています。ユーザーが開発したコードに起因する SLA 違反などの問題は、SLA クレジットの対象外です。詳細については、「Amazon OpenSearch Service - サービスレベルアグリーメント」を参照してください。

トピック

- プラグインクォータ
- 前提条件
- トラブルシューティング
- コンソールを使用したカスタムプラグインのインストール
- を使用したカスタムプラグインの管理 AWS CLI
- Amazon OpenSearch Service カスタムパッケージ AWS KMS 統合

プラグインクォータ

- アカウントにつき、リージョンごとに最大 25 個のカスタムプラグインを作成できます。
- プラグインの非圧縮状態での最大サイズは 1 GB です。
- 1 つのドメインに関連付けることができるプラグインの最大数は 20 です。このクォータは、オプション、サードパーティー、カスタムからなるすべてのプラグインタイプの合計に適用されます。
- カスタムプラグインは、OpenSearch バージョン 2.15 以降を実行しているドメインでサポートされています。
- プラグインの `descriptor.properties` ファイルは、2.15.0 のようなエンジンバージョン、つまりパッチバージョンがゼロに設定されている 2.x.x バージョンをサポートしている必要があります。

前提条件

カスタムプラグインをインストールしてドメインに関連付ける際は、次の要件を満たしていることを確認してください。

- `descriptor.properties` ファイル内のプラグインでサポートされているエンジンバージョンが `2.15.0` または `2.x.0` と同等であること。つまり、パッチバージョンがゼロである必要があります。
- ドメインで以下の機能が有効になっていること。
  - ノード間の暗号化
  - 保管時の暗号化
  - `EnforceHTTPS` が「true」に設定されている

    「AWS Config デベロッパーガイド」の「opensearch-https-required」も参照してください。
  - クライアントは Policy-Min-TLS-1-2-PFS-2023-10 をサポートする必要があります。以下のコマンドを使用することでこのサポートを指定できます。*#########*は、ユーザー自身の情報に置き換えてください。

    ```
    aws opensearch update-domain-config \
        --domain-name domain-name \
        --domain-endpoint-options '{"TLSSecurityPolicy":"Policy-Min-TLS-1-2-
    PFS-2023-10" }'
    ```

    詳細については、「Amazon OpenSearch Service API リファレンス」の「DomainEndpointOptions」を参照してください。

トラブルシューティング

システムがエラー `PluginValidationFailureReason : The provided plugin could not be loaded` を返した場合、トラブルシューティング情報については「the section called “バージョンの互換性の問題によりカスタムプラグインのインストールが失敗する”」を参照してください。

コンソールを使用したカスタムプラグインのインストール

サードパーティープラグインをドメインに関連付けるには、まずプラグインのライセンスと設定をパッケージとしてインポートします。

カスタムプラグインをインストールするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインで [パッケージ] を選択します。

3. [パッケージのインポート] を選択します。
4. [名前] に、このプラグインの一意の識別可能な名前を入力します。
5. (オプション) [説明] に、パッケージまたはその目的に関する有用な詳細を入力します。
6. [パッケージタイプ] で [プラグイン] を選択します。
7. [パッケージソース] に、パスを入力するか、Amazon S3 のプラグイン ZIP ファイルを参照して選択します。
8. [OpenSearch エンジンのバージョン] で、プラグインがサポートする OpenSearch のバージョンを選択します。
9. [パッケージ暗号化] で、パッケージの暗号化キーをカスタマイズするかどうかを選択します。デフォルトでは、OpenSearch Service はプラグインパッケージを AWS 所有のキーで暗号化します。代わりにカスタマーマネージドキーを使用することもできます。
10. [インポート] を選択します。

プラグインパッケージをインポートしたら、ドメインに関連付けます。手順については、「the section called “パッケージをドメインにインポートして関連付ける”」を参照してください。

を使用したカスタムプラグインの管理 AWS CLI

を使用して AWS CLI 、多数のカスタムプラグインタスクを管理できます。

タスク

- を使用したカスタムプラグインのインストール AWS CLI
- を使用したカスタムプラグインの更新 AWS CLI
- AWS KMS キーセキュリティを使用してカスタムプラグインを作成または更新する
- を使用して、カスタムプラグインを使用して OpenSearch Service ドメインを新しいバージョンの OpenSearch にアップグレードする AWS CLI
- カスタムプラグインのアンインストールと、関連付け解除ステータスの表示

を使用したカスタムプラグインのインストール AWS CLI

[開始する前に]

カスタムプラグインをドメインに関連付ける前に、Amazon Simple Storage Service (Amazon S3) のバケットにアップロードする必要があります。バケットは、プラグインを使用する AWS リージョン 場所と同じ に配置する必要があります。S3 バケットにオブジェクトを追加する際の詳細につい

ては、「Amazon Simple Storage Service ユーザーガイド」の「オブジェクトのアップロード」を参照してください。

プラグインに機密情報が含まれている場合は、アップロード時に S3 で管理されたキーによるサーバー側の暗号化を指定します。ファイルをアップロードしたら、その S3 パスを書き留めます。パスの形式は `s3://amzn-s3-demo-bucket/file-path/file-name` です。

> **Note**
>
> オプションで、 AWS Key Management Service (AWS KMS) キーを指定して、プラグインの作成時にカスタムプラグインを保護できます。詳細については、「the section called “AWS KMS キーセキュリティを使用してカスタムプラグインを作成または更新する”」を参照してください。

を使用してカスタムプラグインをインストールするには AWS CLI

1. 以下の create-package コマンドを実行して、カスタムプラグインの新しいパッケージを作成します。その際、以下の要件を満たしていることを確認してください。

   - バケットとキーの場所は、コマンドを実行しているアカウントの S3 バケット内のプラグイン `.zip` ファイルを指している必要があります。
   - S3 バケットは、パッケージを作成しているリージョンと同じリージョンにある必要があります。
   - `ZIP-PLUGIN` パッケージでサポートされているのは `.zip` ファイルのみです。
   - `.zip` ファイルの内容は、プラグインが想定するディレクトリ構造に従う必要があります。
   - `--engine-version` の値は `OpenSearch_{MAJOR}.{MINOR}` 形式である必要があります。例: **`OpenSearch_2.17`**。

   *`#########`*は、ユーザー自身の情報に置き換えてください。

```
aws opensearch create-package \
    --package-name package-name \
    --region region \
    --package-type ZIP-PLUGIN \
    --package-source S3BucketName=amzn-s3-demo-bucket,S3Key=s3-key \
    --engine-version opensearch-version
```

2. (オプション) describe-packages コマンドを使用して、検証やセキュリティ脆弱性の検出結果を含む、create-package オペレーションのステータスを表示します。*########*は、ユーザー自身の情報に置き換えてください。

```
aws opensearch describe-packages \
    --region region  \
    --filters '[{"Name": "PackageType","Value": ["ZIP-PLUGIN"]}, {"Name":
 "PackageName","Value": ["package-name"]}]'
```

コマンドは以下のような情報を返します。

```
{
    "PackageDetailsList": [{
        "PackageID": "pkg-identifier",
        "PackageName": "package-name",
        "PackageType": "ZIP-PLUGIN",
        "PackageStatus": "VALIDATION_FAILED",
        "CreatedAt": "2024-11-11T13:07:18.297000-08:00",
        "LastUpdatedAt": "2024-11-11T13:10:13.843000-08:00",
        "ErrorDetails": {
            "ErrorType": "",
            "ErrorMessage": "PluginValidationFailureReason : Dependency Scan
 reported 3 vulnerabilities for the plugin: CVE-2022-23307, CVE-2019-17571,
 CVE-2022-23305"
        },
        "EngineVersion": "OpenSearch_2.15",
        "AllowListedUserList": [],
        "PackageOwner": "OWNER-XXXX"
    }]
}
```

> **Note**
>
> create-package オペレーションの間、Amazon OpenSearch Service は ZIP-PLUGIN の値について、バージョン互換性、サポートされているプラグイン拡張機能、セキュリティの脆弱性をチェックします。セキュリティの脆弱性は、Amazon Inspector サービスを使用してスキャンされます。これらのチェックの結果は、API レスポンスの ErrorDetails フィールドに表示されます。

3. associate-package コマンドを使用して、プラグインを任意の OpenSearch Service ドメインに関連付けます。その際は、前のステップで作成したパッケージのパッケージ ID を使用します。

   > **Tip**
   >
   > 複数のプラグインがある場合は、代わりに associate-packages コマンドを使用します。これにより、1 回のオペレーションで複数のパッケージをドメインに関連付けることができます。

   *#########*は、ユーザー自身の情報に置き換えてください。

   ```
   aws opensearch associate-package \
       --domain-name domain-name \
       --region region \
       --package-id package-id
   ```

   > **Note**
   >
   > プラグインのインストールおよびアンインストールには、ブルー/グリーンデプロイプロセスが使用されます。

4. (オプション) list-packages-for-domain コマンドを使用して、関連付けのステータスを表示します。関連付けステータスは、ワークフローが ASSOCIATING から ACTIVE に進行するにつれて変化します。プラグインのインストールが完了し、プラグインを使用できる状態になると、関連付けステータスは ACTIVE に変わります。

   *#########*を、ユーザー自身の情報に置き換えます。

   ```
   aws opensearch list-packages-for-domain \
       --region region \
       --domain-name domain-name
   ```

を使用したカスタムプラグインの更新 AWS CLI

update-package コマンドを使用してプラグインを変更します。

> **Note**
>
> オプションで、 AWS Key Management Service (AWS KMS) キーを指定して、プラグインを更新するときにカスタムプラグインを保護できます。詳細については、「the section called "AWS KMS キーセキュリティを使用してカスタムプラグインを作成または更新する"」を参照してください。

を使用してカスタムプラグインを更新するには AWS CLI

- 以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch update-package \
    --region region \
    --package-id package-id \
    --package-source S3BucketName=amzn-s3-demo-bucket,S3Key=s3-key \
    --package-description description
```

パッケージを更新した後は、ssociate-package コマンドまたは associate-packages コマンドを使用して、パッケージの更新をドメインに適用できます。

> **Note**
>
> を使用して、プラグインのオペレーションを監査、作成、更新、関連付け、関連付け解除できます AWS CloudTrail。詳細については、「を使用した Amazon OpenSearch Service API コールのモニタリング AWS CloudTrail」を参照してください。

AWS KMS キーセキュリティを使用してカスタムプラグインを作成または更新する

AWS KMS キーを指定することで、プラグイン作成または更新時にカスタムプラグインを保護できます。これを行うには、PackageEncryptionOptions を true に設定し、以下の例に示すように、キーの Amazon リソースネーム (ARN) を指定します。

例: AWS KMS キーセキュリティを使用してカスタムプラグインを作成する

```
aws opensearch create-package \
    --region us-east-2  --package-name my-custom-package \
    --package-type ZIP-PLUGIN \
```

```
    --package-source S3BucketName=amzn-s3-demo-bucket,S3Key=my-s3-key
    --engine-version OpenSearch_2.15
"PackageConfigOptions": {
     ...
  }
  "PackageEncryptionOptions": {
    "Enabled": true,
    "KmsKeyId":"arn:aws:kms:us-east-2:111222333444:key/2ba228d5-1d09-456c-ash9-
daf42EXAMPLE"
  }
```

例: AWS KMS キーセキュリティでカスタムプラグインを更新する

```
aws opensearch update-package \
    --region us-east-2  --package-name my-custom-package \
    --package-type ZIP-PLUGIN \
    --package-source S3BucketName=amzn-s3-demo-bucket,S3Key=my-s3-key
    --engine-version OpenSearch_2.15
"PackageConfigOptions": {
     ...
  }
  "PackageEncryptionOptions": {
    "Enabled": true,
    "KmsKeyId":"arn:aws:kms:us-east-2:111222333444:key/2ba228d5-1d09-456c-ash9-
daf42EXAMPLE"
  }
```

Important

指定した AWS KMS キーが無効または削除されると、関連付けられたクラスターが動作しなくなる可能性があります。

カスタムパッケージと AWS KMS の統合の詳細については、「」を参照してくださいthe section called “Amazon OpenSearch Service カスタムパッケージ AWS KMS 統合”。

を使用して、カスタムプラグインを使用して OpenSearch Service ドメインを新しいバージョンの OpenSearch にアップグレードする AWS CLI

カスタムプラグインを使用する OpenSearch Service ドメインを、OpenSearch の新しいバージョンにアップグレードする必要がある場合は、以下のプロセスを実行してください。

を使用して、カスタムプラグインを持つ OpenSearch Service ドメインを新しいバージョンの OpenSearch にアップグレードするには AWS CLI

1. create-package コマンドを使用して、新しい OpenSearch バージョンを指定するプラグインの新しいパッケージを作成します。

   すべてのエンジンバージョンのプラグインでパッケージ名が同じであることを確認します。パッケージ名を変更すると、ブルー/グリーンデプロイ中にドメインのアップグレードプロセスが失敗します。

2. 「the section called “ドメインのアップグレード”」の手順に従って、ドメインを上位バージョンにアップグレードします。

   このプロセス中、Amazon OpenSearch Service は以前のバージョンのプラグインパッケージの関連付けを解除し、ブルー/グリーンデプロイを使用して新しいバージョンをインストールします。

カスタムプラグインのアンインストールと、関連付け解除ステータスの表示

任意のドメインからプラグインをアンインストールするには、disssociate-package コマンドを使用します。このコマンドを実行すると、関連する設定やライセンスパッケージも削除されます。その後、list-packages-for-domain コマンドを使用して、関連付け解除のステータスを表示できます。

> **Tip**
>
> dissociate-packages コマンドを使用して、1 回のオペレーションでドメインから複数のプラグインをアンインストールすることもできます。

カスタムプラグインをアンインストールして、関連付け解除ステータスを表示するには

1. すべてのインデックスでプラグインを無効にします。これは、プラグインパッケージの関連付けを解除する前に行う必要があります。

   すべてのインデックスからプラグインを無効にする前にプラグインをアンインストールしようとすると、ブルー/グリーンデプロイプロセスは `Processing` 状態でスタックしたままになります。

2. 次のコマンドを実行して、プラグインをアンインストールします。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch dissociate-package \
    --region region \
    --package-id plugin-package-id \
    --domain-name domain name
```

3. (オプション) list-packages-for-domain コマンドを実行して、関連付け解除のステータスを表示します。

Amazon OpenSearch Service カスタムパッケージ AWS KMS 統合

Amazon OpenSearch Service カスタムパッケージは、 を使用して保管中のZIP-PLUGINパッケージを保護するために、デフォルトで暗号化を提供します AWS マネージドキー。

- AWS 所有のキー – Amazon OpenSearch Service カスタムパッケージは、デフォルトでこれらのキーを使用して ZIP-PLUGIN パッケージを自動的に暗号化します。その使用を表示、管理、使用 AWS 所有のキー 、または監査することはできません。しかし、データを暗号化するキーを保護するために何らかの操作を行ったり、プログラムを変更したりする必要はありません。詳細については、「AWS Key Management Service デベロッパーガイド」の 「AWS 所有のキー」を参照してください。
- カスタマーマネージドキー – ZIP-PLUGINカスタムパッケージの作成時にカスタマーマネージドキーを選択すること AWS 所有のキー で、既存の に 2 番目の暗号化レイヤーを追加できます。

  Amazon OpenSearch Service カスタムパッケージは、作成、所有、管理する対称カスタマーマネージドキーを使用して、既存の AWS 所有暗号化に 2 番目の暗号化レイヤーを追加することをサポートします。この暗号化レイヤーはユーザーが完全に制御できるため、以下のタスクを実行できます。

  - キーポリシーの確立と維持
  - AWS Identity and Access Management (IAM) ポリシーと許可の確立と維持
  - キーポリシーの有効化と無効化
  - キー暗号化マテリアルのローテーション
  - タグを追加
  - キーエイリアスの作成
  - キー削除のスケジュール

詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

> **Note**
>
> Amazon OpenSearch Service カスタムパッケージは、 AWS 所有のキー を使用して、保管時の暗号化を無償で自動的に有効にします。ただし、カスタマーマネージドキーを使用する場合、 AWS KMS 料金が適用されます。料金の詳細については、「AWS Key Management Service の料金」を参照してください。

Amazon OpenSearch Service カスタムパッケージサービスが で許可を使用する方法 AWS KMS

OpenSearch Service カスタムパッケージには、カスタマーマネージドキーを使用するための許可が必要です。

カスタマーマネージドキーで暗号化された`ZIP-PLUGIN`パッケージを作成すると、Amazon OpenSearch Service カスタムパッケージサービスは CreateGrant リクエストを送信してユーザーに代わって許可を作成します AWS KMS。の許可 AWS KMS は、OpenSearch Service にアカウントの AWS KMS キーへのアクセスを許可します。OpenSearch Service カスタムパッケージによって作成された許可には、リクエストにカスタムパッケージ ID を持つ暗号化コンテキストが含まれている場合にのみオペレーションを許可する制約があります。

Amazon OpenSearch Service カスタムパッケージは、以下の内部オペレーションのために、カスタマーマネージドキーの使用に関する許可を必要とします。

運用	説明
`DescribeKey`	プラグインパッケージの作成時に入力された対称カスタマーマネージドキー ID が有効 AWS KMS であることを確認する`DescribeKey` リクエストを に送信します。
`GenerateDataKeyWithoutPlaintext`	カスタマーマネージドキーによって暗号化されたデータキーを生成する AWS KMS リクエスト`GenerateDataKeyWithoutPlaintext` を に送信します。
`GenerateDataKey`	`GenerateDataKey` リクエストを に送信 AWS KMS して、内部でコピーするときにパッケージを暗号化するデータキーを生成します。

運用	説明
`Decrypt`	暗号化されたデータキーを復号 AWS KMS 化するための`Decrypt`リクエストを に送信し、データの復号化に使用できます。

許可へのアクセスの取り消しや、カスタマーマネージドキーに対するサービスのアクセスの取り消しは、いつでもできます。これを行うと、OpenSearch Service は、カスタマーマネージドキーによって暗号化されたすべてのデータにアクセスできなくなり、そのデータに依存しているオペレーションが影響を受けます。例えば、OpenSearch Service がアクセスできないプラグインパッケージを関連付けようとすると、オペレーションは `AccessDeniedException` エラーを返します。

カスタマーマネージドキーを作成する

対称カスタマーマネージドキーは、 AWS マネジメントコンソール または AWS KMS APIs を使用して作成できます。

対称カスタマーマネージドキーを作成するには

- 「AWS Key Management Service デベロッパーガイド」の「KMS キーの作成」のステップに従います。

キーポリシー

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。キーポリシーは、カスタマーマネージドキーの作成時に指定できます。詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMSでのキーポリシー」を参照してください。

カスタマーマネージドキーをプラグインリソースで使用するには、キーポリシーで次の API オペレーションを許可する必要があります。

- `kms:CreateGrant` - カスタマーマネージドキーに許可を追加します。指定された AWS KMS キーへのアクセスを制御する権限を付与し、OpenSearch Service カスタムパッケージが必要とするオペレーションを付与するためのアクセスを許可します。許可の使用についての詳細は、「AWS KMS デベロッパーガイド」を参照してください。

  これにより、OpenSearch Service は以下を実行できます。

  - GenerateDataKeyWithoutPlainText を呼び出して暗号化されたデータキーを生成し、さらに検証するために保存します。
  - GenerateDataKey を呼び出してプラグインパッケージを内部でコピーします。
  - Decrypt を呼び出してプラグインパッケージに内部でアクセスします。
  - RetireGrant にサービスが許可するための、廃止するプリンシパルを設定します。
- kms:DescribeKey – カスタマーマネージドキーの詳細を提供し、OpenSearch Service がキーを検証できるようにします。
- kms:GenerateDataKey、kms:GenerateDataKeyWithoutPlaintext、 kms:Decrypt – OpenSearch Service カスタムパッケージが、許可内でこれらのオペレーションを使用できるようにします。

OpenSearch Service カスタムパッケージに追加できるポリシーステートメントの例を次に示します。

```
"Statement" : [
  {
    "Sid" : "Allow access to principals authorized to use OpenSearch Service custom
 packages",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "*"
    },
    "Action" : [
      "kms:CreateGrant",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyWithoutPlaintext",
      "kms:Decrypt"
    ],
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "kms:ViaService" : "custom-packages.region.amazonaws.com"
      },
      "StringEquals" : {
        "kms:EncryptionContext:packageId": "Id of the package"
      }
    }
  },
```

```
  {
    "Sid" : "Allow access to principals authorized to use Amazon OpenSearch Service
 custom packages",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "*"
    },
    "Action" : [
      "kms:DescribeKey"
    ],
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "kms:ViaService" : "custom-packages.region.amazonaws.com"
      }
    }
  }
]
```

ポリシーでのアクセス許可の指定に関する詳細については、「AWS Key Management Service デベロッパーガイド」の「Key policies in AWS KMS」を参照してください。

キーアクセスのトラブルシューティングの詳細については、 AWS Key Management Service デベロッパーガイドの「アクセスAWS KMS 許可のトラブルシューティング」を参照してください。

Amazon OpenSearch Service カスタムパッケージのカスタマーマネージドキーを指定する

カスタマーマネージドキーは、ZIP-PLUGIN パッケージの 2 番目の暗号化レイヤーとして指定できます。

プラグインパッケージを作成するときは、OpenSearch Service カスタムパッケージがプラグインパッケージの暗号化に使用する AWS KMS キー ID を入力してデータキーを指定できます。

AWS KMS key ID — AWS KMS カスタマーマネージドキーのキー識別子。キー ID、キー ARN、エイリアス名、またはエイリアス ARN を入力します。

Amazon OpenSearch Service カスタムパッケージの暗号化コンテキスト

暗号化コンテキストは、データに関する追加のコンテキスト情報を含むキーと値のペアのオプションセットです。

AWS KMS は、追加の認証済みデータとして暗号化コンテキストを使用して、認証済み暗号化をサポートします。データを暗号化するリクエストに暗号化コンテキストを含めると、 は暗号化コンテキストを暗号化されたデータに AWS KMS バインドします。データを復号化するには、そのリクエストに (暗号化時と) 同じ暗号化コンテキストを含めます。

Amazon OpenSearch Service カスタムパッケージの暗号化コンテキスト

Amazon OpenSearch Service カスタムパッケージは、すべての AWS KMS 暗号化オペレーションで同じ暗号化コンテキストを使用します。キーは `packageId`で、値はプラグインパッケージ`package-id`の です。

モニタリングに暗号化コンテキストを使用する

対称カスタマーマネージドキーを使用してプラグインパッケージを暗号化する場合は、監査レコードとログで暗号化コンテキストを使用して、カスタマーマネージドキーがどのように使用されているかを特定することができます。暗号化コンテキストは、 AWS CloudTrail または Amazon CloudWatch Logs によって生成されたログにも表示されます。

暗号化コンテキストを使用して顧客マネージドキーへのアクセスを制御する

対称カスタマーマネージドキー (CMK) へのアクセスを制御するための条件として、キーポリシーと IAM ポリシー内の暗号化コンテキストを使用することもできます。付与する際に、暗号化コンテキストの制約を使用することもできます。

OpenSearch Service カスタムパッケージは、許可で暗号化コンテキスト制約を使用して、アカウントまたはリージョン内のカスタマーマネージドキーへのアクセスを制御します。グラントの制約では、指定された暗号化コンテキストの使用をグラントが許可するオペレーションが必要です。

次に、特定の暗号化コンテキストのカスタマーマネージドキーへのアクセスを付与するキーポリシーステートメントの例を示します。このポリシーステートメントの条件では、暗号化コンテキストを指定する暗号化コンテキスト制約がグラントに必要です。

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
    },
    "Action": "kms:DescribeKey",
    "Resource": "*"
},
```

```
{
    "Sid": "Enable OpenSearch Service custom packages to use the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
    },
    "Action" : [
         "kms:CreateGrant",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals" : {
            "kms:EncryptionContext:packageId": "ID of the package"
         }
    }
}
```

OpenSearch カスタムパッケージサービスの暗号化キーのモニタリング

OpenSearch Service カスタムパッケージサービスリソースで AWS KMS カスタマーマネージドキーを使用する場合、CloudTrail または CloudWatch Logs を使用してOpenSearch カスタムパッケージが に送信するリクエストを追跡できます AWS KMS。

詳細情報

次のリソースは、保管時のデータ暗号化についての詳細を説明しています。

- AWS KMS 基本概念の詳細については、「 AWS Key Management Service デベロッパーガイドAWS KMS keys」の「」を参照してください。
- のセキュリティのベストプラクティスの詳細については AWS KMS、AWS 「ベストプラクティスの規範ガイダンスガイド」を参照してくださいAWS Key Management Service。

Amazon OpenSearch Service でのサードパーティープラグインのインストール

Amazon OpenSearch Service は、選択したパートナーのサードパーティープラグインをサポートしています。これらのプラグインにより、カスタムアナライザー、トークナイザ、暗号化機能などの

追加機能を使用して OpenSearch のセットアップを強化できます。OpenSearch Service ドメインと適切に統合するには、サードパーティー開発者が提供する特定のインストールと設定の手順に従います。

> **Note**
>
> サードパーティー開発者から直接有効なライセンスを取得して維持する必要があります。一部のプロバイダーはプラグインをすべて有効にしない可能性があるため AWS リージョン、プラグインプロバイダーに可用性を確認してください。

OpenSearch Service では、次のサードパーティープラグインを使用できます。

- Portal26 暗号化プラグイン (Titanium-lockbox) – NIST FIPS 140-2 認定暗号化を使用して、インデックス作成時にデータを暗号化します。これには、セキュリティを強化するために暗号化キーを管理できる Bring Your Own Key (BYOK) サポートが含まれています。プラグインは Portal26 によって提供され、OpenSearch バージョン 2.15 以降が必要です。
- 名前一致 (RNI) – 24 以上の言語で名前、組織、住所、日付を一致させて、セキュリティとコンプライアンスを向上させます。プラグインは Babel Street によって提供され、OpenSearch バージョン 2.15 以降が必要です。

トピック

- 前提条件
- サードパーティープラグインのインストール
- 次の手順

前提条件

サードパーティープラグインをインストールする前に、次の手順を実行します。

- プラグイン設定ファイルとライセンスファイルを取得し、Amazon S3 バケットにアップロードします。バケットはドメイン AWS リージョン と同じ にある必要があります。
- サードパーティープラグインは、カスタムプラグインの一種です。ドメインがカスタムプラグインの前提条件を満たしていることを確認してください。

サードパーティープラグインのインストール

サードパーティープラグインを OpenSearch Service ドメインに関連付けるには、まずライセンスパッケージ、設定パッケージ、プラグインパッケージの 3 つの個別のパッケージをアップロードする必要があります。

- ライセンスパッケージには、プラグインに関連付けられたライセンス情報またはメタデータが .json または .xml 形式で含まれています。
- 設定パッケージには、プラグイン設定ファイルと、サポートされているアセットと設定が含まれています。これらのファイルは、プラグインの動作または OpenSearch との統合方法を定義します。
- プラグインパッケージには、OpenSearch が実行する実行可能コードであるコンパイル済みプラグインバイナリが含まれています。これはプラグイン機能の中核です。

両方のパッケージをアップロードすると、プラグインとライセンスを互換性のあるドメインに関連付けることができます。

コンソール

サードパーティープラグインをドメインに関連付けるには、まずプラグインのライセンスと設定をパッケージとしてインポートします。

サードパーティープラグインをインストールするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインで [パッケージ] を選択します。
3. まず、ライセンスパッケージをインポートします。[パッケージのインポート] を選択します。
4. [パッケージタイプ] で、[ライセンス] を選択します。
5. [パッケージソース] には、Amazon S3 のライセンス JSON または XML ファイルへのパスを入力します。
6. [インポート] を選択します。パッケージは、[パッケージ] ページの [ライセンス] タブに表示されます。
7. 次に、プラグイン設定をインポートします。[パッケージのインポート] を再度選択します。
8. [パッケージタイプ] で、[設定] を選択します。
9. [パッケージソース] には、Amazon S3 のプラグイン設定 ZIP ファイルへのパスを入力します。

10. [インポート] を選択します。
11. 最後に、プラグイン自体をインポートします。[パッケージのインポート] を選択します。
12. [パッケージタイプ] で [プラグイン] を選択します。
13. [パッケージソース] には、Amazon S3 のプラグイン ZIP ファイルへのパスを入力します。
14. プラグインがサポートする OpenSearch エンジンのバージョンを選択します。
15. [インポート] を選択します。

サードパーティープラグインをドメインに関連付けるには

1. 次に、プラグインのライセンスと設定をドメインに関連付けます。左側のナビゲーションペインで [ドメイン] を選択します。
2. ドメイン名を選択し、クラスター設定を開きます。
3. [プラグイン] タブに移動します。
4. [パッケージを関連付ける] を選択し、インポートしたプラグイン、ライセンス、および設定パッケージを選択します。
5. [選択] を選択します。
6. [次へ] を選択します。関連付けるパッケージを確認し、[関連付け] を選択します。

CLI

まず、create-package コマンドを使用して、プラグインライセンスを含む新しいパッケージを作成します。S3Key は、ライセンステキストまたはメタデータを含む Amazon S3 の .json ファイルまたは .xml ファイルを指す必要があります。

```
aws opensearch create-package \
  --package-name plugin-license-package \
  --package-type PACKAGE-LICENSE \
  --package-source S3BucketName=my-bucket,S3Key=licenses/my-plugin-license.json
```

create-package コマンドを再度使用して、プラグイン設定を含むパッケージを作成します。S3Key は、プラグインで想定されるディレクトリ構造に準拠する Amazon S3 の .zip ファイルを指す必要があります。

```
aws opensearch create-package \
  --package-name plugin-config-package \
  --package-type PACKAGE-CONFIG \
```

```
  --package-source S3BucketName=my-bucket,S3Key=path/to/package.zip
```

create-package コマンドを再度使用して、プラグイン自体を含むパッケージを作成します。S3Key は、Amazon S3 のプラグイン .zip ファイルを指す必要があります。

```
aws opensearch create-package \
  --package-name plugin-package \
  --package-type ZIP-PLUGIN \
  --package-source S3BucketName=my-bucket,S3Key=path/to/package.zip
```

最後に、associate-package コマンドを使用して、それぞれのパッケージ ID を指定することで、パートナープラグイン、ライセンス、および設定を互換性のあるドメインにリンクします。プラグイン ID を他のパッケージの前提条件として指定します。つまり、他のパッケージより前にドメインに関連付けられている必要があります。

```
aws opensearch associate-packages \
  --domain-name my-domain \
  --package-list '[{"PackageID": "plugin-package-id"},{"PackageID": "license-package-id","PrerequisitePackageIDList":["plugin-package-id"]},{"PackageID":"config-package-id","PrerequisitePackageIDList":["plugin-package-id"]}]'
```

次の手順

関連付けが完了したら、要件に基づき、特定のインデックスでプラグインを有効にするか、必要に応じて設定できます。サードパーティープラグイン機能を特定のインデックスに適用するには、インデックスの作成時にインデックス設定を変更するか、既存のインデックスを更新します。例えば、サードパーティープラグインにカスタムアナライザーが含まれている場合は、インデックス設定でそれを参照します。

プラグイン機能を複数のインデックスに一貫して適用するには、プラグイン設定が含まれたインデックステンプレートを使用します。OpenSearch のセットアップ用に機能を設定する方法については、プラグインのドキュメントを必ず参照してください。

SQL を用いた Amazon OpenSearch Service のデータのクエリを行う

JSON ベースの OpenSearch クエリ DSL を使用するのではなく、SQL を使用して Amazon OpenSearch Service のクエリを行うことができます。SQL を用いたクエリの実施は、その言語にす

でに精通している場合や、それを使用するアプリケーションにドメインを統合する場合に役立ちます。SQL サポートは、OpenSearch または Elasticsearch 6.5 以降を実行しているドメインで利用できます。

> **Note**
>
> このドキュメントでは、OpenSearch Service と SQL プラグインのさまざまなバージョン、および JDBC ドライバーおよび ODBC ドライバーのバージョン互換性について説明します。基本および複雑なクエリ、関数、メタデータクエリ、集計関数の構文についての情報は、オープンソースの「OpenSearch のドキュメント」を参照してください。

次のテーブルを使用して、OpenSearch および Elasticsearch の各バージョンでサポートされている SQL プラグインのバージョンを見つけます。

OpenSearch

OpenSearch バージョン	SQL プラグインバージョン	注目すべき機能
2.19.0	2.19.0.0	
2.18.0	2.18.0.0	
2.17.0	2.17.0.0	
2.15.0	2.15.0.0	
2.13.0	2.13.0.0	
2.11.0	2.11.0.0	PPL 言語とクエリのサポートを追加
2.9.0	2.9.0.0	Spark コネクターを追加し、テーブル関数と PromQL 関数をサポートします
2.7.0	2.7.0.0	`datasource` API を追加
2.5.0	2.5.0.0	
2.3.0	2.3.0.0	`maketime` および `makedate` 日時関数を追加

OpenSearch バージョン	SQL プラグインバージョン	注目すべき機能
1.3.0	1.3.0.0	デフォルトのクエリ制限サイズと、値リスト内から選択する IN 句をサポート
1.2.0	1.2.0.0	可視化応答形式の新しいプロトコルを追加
1.1.0	1.1.0.0	SQL と PPL のフィルターとしてマッチ機能をサポートする
1.0.0	1.0.0.0	データストリームのクエリをサポートする

Open Distro for Elasticsearch

Elasticsearch バージョン	SQL プラグインバージョン	注目すべき機能
7.10	1.13.0	Window 関数の NULL FIRST および LAST、CAST() 関数、SHOW、および DESCRIBE コマンド
7.9	1.11.0	追加の日付/時刻関数、ORDER BY キーワードの追加
7.8	1.9.0	
7.7	1.8.0	
7.3	1.3.0	複数の文字列および数値演算子
7.1	1.1.0	

サンプル呼び出し

SQL を用いてデータのクエリを行うには、次の形式を使用して _sql に HTTP リクエストを送信します。

```
POST domain-endpoint/_plugins/_sql
{
  "query": "SELECT * FROM my-index LIMIT 50"
```

```
}
```

> **Note**
>
> ドメインが OpenSearch ではなく Elasticsearch を実行している場合、フォーマットは `_opendistro/_sql` になります。

注意と相違点

`_plugins/_sql` の呼び出しではリクエストボディにインデックス名が含まれるため、バルク、mget、および msearch オペレーションと同じアクセスポリシーの考慮事項が適用されます。これまでどおり、API オペレーションにアクセス許可を付与するときは、最小権限の原則に従う必要があります。

きめ細かなアクセスコントロールでの SQL の使用に関連するセキュリティ上の考慮事項については、「the section called “きめ細かなアクセスコントロール”」を参照してください。

OpenSearch SQL プラグインには、多くの調整可能な設定があります。OpenSearch Service では、プラグイン設定のパス (`_plugins/_query/settings`) ではなく、`_cluster/settings` パスを使用します。

```
PUT _cluster/settings
{
  "transient" : {
    "plugins.sql.enabled" : true
  }
}
```

従来の Elasticsearch ドメインの場合、`plugins` を `opendistro` に置き換えます。

```
PUT _cluster/settings
{
  "transient" : {
    "opendistro.sql.enabled" : true
  }
}
```

SQL Workbench

SQL Workbench は、オンデマンド SQL クエリの実行、SQL から REST への変換を行えるほか、テキスト、JSON、JDBC、または CSV として結果を表示および保存できる OpenSearch Dashboards ユーザーインターフェイスです。詳細については、「クエリワークベンチ」を参照してください。

SQL CLI

SQL CLI は、`opensearchsql` コマンドで起動できるスタンドアロンの Python アプリケーションです。インストール、構成、および使用のステップについては、「SQL CLI」を参照してください。

JDBC ドライバー

Java Database Connectivity (JDBC) ドライバーにより、OpenSearch Service ドメインをお気に入りのビジネスインテリジェンス (BI) アプリケーションに統合できます。ドライバーをダウンロードするには、こちらをクリックしてください。詳細については、「 GitHub リポジトリ」を参照してください。

ODBC ドライバー

Open Database Connectivity (ODBC) ドライバーは、Windows と macOS 用の、読み取り専用の ODBC ドライバーです。これを使用すれば、Microsoft Excel のようなビジネスインテリジェンスやデータ可視化のアプリケーションを SQL プラグインに接続できます。

ドライバーのインストールの詳細については、GitHub の SQL リポジトリ を参照してください。

Amazon OpenSearch Service のクロスクラスター検索

Amazon OpenSearch Service でのクラスター間検索では、接続された複数のドメインにまたがってクエリと集計を実行できます。さまざまなタイプのワークロードを実行している場合には特に、単一の大きなドメインではなく、複数の小さなドメインを使用する方が合理的です。

ワークロード固有のドメインを使用すると、次のタスクを実行できます。

- 特定のワークロードのインスタンスタイプを選択することにより、各ドメインを最適化する。
- ワークロード間で障害分離の境界を確立する。これは、ワークロードの 1 つに障害が発生しても、その障害はその特定のドメイン内で食い止められ、他のワークロードに影響しないことを意味します。

- ドメイン間でより簡単にスケーリングできる。

クラスター間検索では OpenSearch Dashboards がサポートされているため、すべてのドメインで視覚化とダッシュボード作成を行うことができます。ドメイン間で転送された検索結果には、標準の AWS データ転送料金がかかります。

> **Note**
>
> オープンソースの OpenSearch には、クラスター間検索に関するドキュメントもあります。オープンソースクラスターの設定は、マネージド Amazon OpenSearch Service ドメインとは大きく異なります。特に、OpenSearch Service では、cURL AWS マネジメントコンソール ではなく を使用してクラスター間接続を設定します。さらに、 マネージドサービスは、きめ細かなアクセスコントロールに加えて、クラスター間認証に AWS Identity and Access Management (IAM) を使用します。したがって、オープンソースの OpenSearch ドキュメントではなく、このドキュメントを使用してドメインのクラスター間検索を設定することをお勧めします。

トピック

- 制限事項
- クラスター間検索の前提条件
- クラスター間検索の料金
- 接続のセットアップ
- 接続の削除
- セキュリティのセットアップとサンプルチュートリアル
- OpenSearch Dashboards

制限事項

クラスター間検索には、以下に示す重要な制限事項があります。

- Elasticsearch ドメインを OpenSearch ドメインに接続することはできません。
- セルフマネージド OpenSearch/Elasticsearch クラスターに接続することはできません。
- リージョン間でドメインを接続するには、両方のドメインが Elasticsearch 7.10 以降または OpenSearch 上にある必要があります。

- ドメインでは、最大 20 の送信接続が可能です。同様に、ドメインでは最大 20 の着信接続が可能です。つまり、1 つのドメインは、最大 20 の他のドメインに接続できます。
- ソースドメインは、宛先ドメインと同じかそれ以降のバージョンである必要があります。2 つのドメイン間で双方向接続を設定し、その一方または両方をアップグレードする場合は、まず接続の 1 つを削除する必要があります。
- クラスター間検索で、カスタム辞書や SQL を使用することはできません。
- を使用してドメイン CloudFormation を接続することはできません。
- M3 インスタンスとバースト可能 (T2 および T3) インスタンスではクラスター間検索は使用できません。

クラスター間検索の前提条件

クラスター間検索をセットアップする前に、ドメインが次の要件を満たしていることを確認してください。

- 2 つの OpenSearch ドメイン、またはバージョン 6.7 以降の Elasticsearch ドメイン
- Fine-grained access controlが有効
- ノード間暗号化が有効
- いずれかのドメインが VPC 内にある場合、ドメインは VPC ピアリングまたは Transit Gateway を介して接続する必要があり、セキュリティグループはそれらの間のトラフィックを許可する必要があります。

クラスター間検索の料金

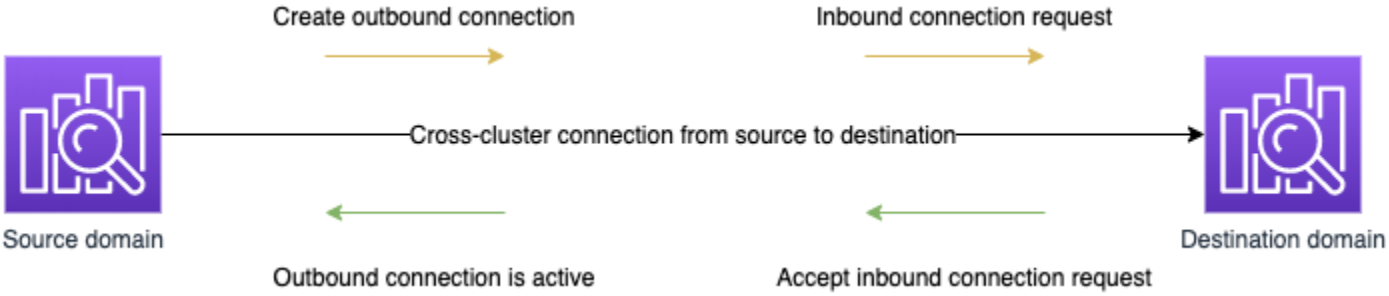
ドメインをまたがって検索する場合も、追加料金はかかりません。

接続のセットアップ

"ソース" ドメインは、クラスター間検索リクエストが発生したドメインを指します。つまり、ソースドメインは、ユーザーからの最初の検索リクエストの送信先です。

"ターゲット" ドメインは、ソースドメインから照会するドメインです。

クラスター間の接続は、ソースからターゲットドメインへの単方向のみです。つまり、ターゲットドメインからソースドメインを照会することはできません。ただし、反対方向で別の接続をセットアップすることは可能です。



ソースドメインは、ターゲットドメインへの "アウトバウンド" 接続を作成します。ターゲットドメインは、ソースドメインから "インバウンド" 接続リクエストを受信します。

接続をセットアップするには

1. ドメインダッシュボードで、ドメインを選択し、[接続] タブに進みます。
2. [アウトバウンド接続] セクションで、[リクエスト] を選択します。
3. [接続のエイリアス] に接続の名前を入力します。
4. AWS アカウント と リージョンのドメインに接続するか、別のアカウントまたはリージョンに接続するかを選択します。
   - AWS アカウント と リージョンのクラスターに接続するには、ドロップダウンメニューからドメインを選択し、リクエストを選択します。
   - 別の AWS アカウント またはリージョンのクラスターに接続するには、リモートドメインの ARN を選択し、リクエストを選択します。リージョン間でドメインを接続するには、両方のドメインで Elasticsearch バージョン 7.10 以降または OpenSearch が実行されている必要があります。
5. クラスタークエリで使用できないクラスターをスキップするには、 [使用できないものをスキップ] を選択します。この設定を行うと、1 つまたは複数のリモートクラスターで障害が発生しても、クラスター間のクエリで部分的な結果が返されるようになります。
6. クラスター間検索では、まず接続リクエストを検証することで、前提条件が満たされているか確認が行われます。ドメインに互換性がないことがわかった場合、接続リクエストは `Validation failed` の状態になります。
7. 接続要求が正常に検証されると、接続リクエストはターゲットドメインに送信され、承認を受けます。この承認が行われるまで、接続は `Pending acceptance` の状態のままです。接続要求がターゲットドメインで承諾されると、状態が `Active` に変わり、ターゲットドメインをクエリに使用できるようになります。

- ドメインページには、ターゲットドメインについて、ドメインの全体的な正常性とインスタンスの正常性に関する詳細が表示されます。ドメインとの間の接続を柔軟に作成、表示、削除、モニタリングできるのは、ドメイン所有者のみです。

接続が確立されると、接続されたドメインのノード間で流れるトラフィックはすべて暗号化されます。VPC ドメインを非 VPC ドメインに接続し、非 VPC ドメインがインターネットからのトラフィックを受信できるパブリックエンドポイントである場合も、ドメイン間のクラスター間トラフィックは暗号化され安全です。

接続の削除

接続を削除すると、インデックスに対するクラスター間のオペレーションがすべて停止します。

1. ドメインダッシュボードで、[接続] タブに移動します。
2. 削除するドメイン接続を選択し、[削除] を選択したら、削除を確定します。

接続を削除するには、ソースドメインまたはターゲットドメインのいずれかでこれらの手順を実行できます。削除した接続は 15 日間 `Deleted` ステータスで表示されます。

アクティブなクラスター間接続のあるドメインは削除できません。ドメインを削除するには、まずそのドメインからすべての受信接続と送信接続を削除します。この操作は、ドメインを削除する前にクラスター間ドメインのユーザーを考慮するために行います。

セキュリティのセットアップとサンプルチュートリアル

1. クラスター間検索リクエストをソースドメインに送信します。
2. ソースドメインでは、ドメインアクセスポリシーに照らし合わせてリクエストを評価します。クラスター間検索にはきめ細かなアクセス制御が必要になるため、ソースドメインでオープンアクセスポリシーを使用することをお勧めします。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/src-domain/*"
    }
  ]
}
```

> **Note**
>
> パスにリモートインデックスを含める場合は、ドメイン ARN で URI を URL エンコードする必要があります。例えば、`arn:aws:es:us-east-1:123456789012:domain/my-domain/local_index,dst:remote_index` ではなく `arn:aws:es:us-east-1:123456789012:domain/my-domain/local_index,dst%3Aremote_index` を使用します。

きめ細かなアクセス制御に加えて制限付きアクセスポリシーを使用する場合は、ポリシーでは最低限でも `es:ESHttpGet` へのアクセスを許可する必要があります。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:user/test-user"
        ]
      },
      "Action": "es:ESHttpGet",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/src-domain/*"
```

```
    }
  ]
}
```

3. ソースドメインに対するきめ細かなアクセス制御では、要求が次のように評価されます。

   - リクエストが、有効な IAM または HTTP 基本認証情報で署名されているか。
   - その場合、検索を実行し、データにアクセスするためのアクセス許可がユーザーにあるか。

   要求がターゲットドメインのデータのみを検索する場合 (例: `dest-alias:dest-index/_search`)、ターゲットドメインに対するアクセス許可のみが必要になります。

   両方のドメインを対象にしたデータ検索のリクエストの場合 (例: `source-index,dest-alias:dest-index/_search`)、両方のドメインに対するアクセス許可が必要です。

   きめ細かいアクセス制御を使用する場合、ユーザーには、関連するインデックスに対して標準の `read` または `search` アクセス許可に加えて `indices:admin/shards/search_shards` アクセス許可も必要になります。

4. ソースドメインは、ターゲットドメインにリクエストを渡します。ターゲットドメインでは、このリクエストをドメインアクセスポリシーに照らし合わせて評価します。ターゲットドメインに対する `es:ESCrossClusterGet` アクセス許可も含める必要があります。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:ESCrossClusterGet",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/dst-domain"
    }
  ]
}
```

es:ESCrossClusterGet アクセス許可が /dst-domain/* ではなく /dst-domain に適用されていることを確認してください。

ただし、この最小ポリシーでは、クラスター間検索のみが許可されます。ドキュメントのインデックス作成や標準検索の実行など、他のオペレーションを実行するには、追加のアクセス許可が必要です。ターゲットドメインでは、以下のポリシーの使用をお勧めします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/dst-domain/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:ESCrossClusterGet",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/dst-domain"
    }
  ]
}
```

> **Note**
>
> ドメイン間のクラスター間検索リクエストはすべて、ノード間の暗号化の一環としてデフォルトで転送中に暗号化されます。

5. ターゲットドメインで検索が実行され、結果がソースドメインに返されます。
6. ソースドメインにより、独自の結果 (存在する場合) とターゲットドメインの結果を組み合わせたものが返されます。
7. テストリクエストには、Postman の使用をお勧めします。

   - ターゲットドメインで、ドキュメントのインデックスを作成します。

   ```
   POST https://dst-domain.us-east-1.es.amazonaws.com/books/_doc/1

   {
     "Dracula": "Bram Stoker"
   }
   ```

   - ソースドメインからこのインデックスを照会するには、クエリ内にターゲットドメインの接続エイリアスを含めます。

   ```
   GET https://src-domain.us-east-1.es.amazonaws.com/<connection_alias>:books/
   _search

   {
       ...
     "hits": [
       {
         "_index": "source-destination:books",
         "_type": "_doc",
         "_id": "1",
         "_score": 1,
         "_source": {
           "Dracula": "Bram Stoker"
         }
       }
     ]
   }
   ```

接続エイリアスは、ドメインダッシュボードの [接続] タブで確認できます。

- domain-a -> domain-b の接続を接続エイリアス cluster_b でセットアップし、domain-a -> domain-c 接続を接続エイリアス cluster_c でセットアップする場合は、次のように domain-a、domain-b、および domain-c を検索します。

```
GET https://src-domain.us-east-1.es.amazonaws.com/
local_index,cluster_b:b_index,cluster_c:c_index/_search
{
  "query": {
    "match": {
      "user": "domino"
    }
  }
}
```

レスポンス

```
{
  "took": 150,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "failed": 0,
    "skipped": 0
  },
  "_clusters": {
    "total": 3,
    "successful": 3,
    "skipped": 0
  },
  "hits": {
    "total": 3,
    "max_score": 1,
    "hits": [
      {
        "_index": "local_index",
        "_type": "_doc",
        "_id": "0",
        "_score": 1,
        "_source": {
```

```
          "user": "domino",
          "message": "This is message 1",
          "likes": 0
        }
      },
      {
        "_index": "cluster_b:b_index",
        "_type": "_doc",
        "_id": "0",
        "_score": 2,
        "_source": {
          "user": "domino",
          "message": "This is message 2",
          "likes": 0
        }
      },
      {
        "_index": "cluster_c:c_index",
        "_type": "_doc",
        "_id": "0",
        "_score": 3,
        "_source": {
          "user": "domino",
          "message": "This is message 3",
          "likes": 0
        }
      }
    ]
  }
}
```

接続のセットアップで使用できないクラスターをスキップしなかった場合、検索リクエストを正常に実行するには、検索するすべての宛先クラスターが使用可能である必要があります。そうでない場合は、リクエスト全体が失敗します ドメインのうち、いずれか 1 つだけが使用できない場合でも、検索結果は返されません。

OpenSearch Dashboards

接続された複数ドメインのデータは、単一のドメインの場合と同じ方法で可視化できますが、connection-alias:index を使用してリモートインデックスにアクセスする必要があります。インデックスパターンは connection-alias:index に一致する必要があります。

Amazon OpenSearch Service 用の Learning to Rank

OpenSearch は、BM-25 と呼ばれる確率的ランキングフレームワークを使用して、関連性スコアを計算します。特定のキーワードがドキュメント内で頻繁に表示される場合、BM-25 はそのドキュメントに対して高い関連性スコアを割り当てます。ただし、このフレームワークでは、クリックスルーデータなどのユーザーの行動は考慮されないため、関連性がさらに改善します。

Learning to Rank はオープンソースの OpenSearch プラグインで、機械学習と行動データを使用してドキュメントの関連性を調整できます。XGBoost および Ranklib ライブラリのモデルを使用して、検索結果のリスコアを行います。Elasticsearch LTプラグインは当初、OpenSource Connections によって開発され、ウィキメディア財団、Snagajob Engineering、Bonsai、Yelp Engineering による多大な貢献があります。OpenSearch バージョンのプラグインは Elasticsearch LTR プラグインから派生しています。

Learning to Rank には、OpenSearch または Elasticsearch 7.7 以降が必要です。Learning to Rank プラグインを使用するには、完全な管理者許可が必要です。詳細についてはthe section called “マスターユーザーの変更”を参照してください。

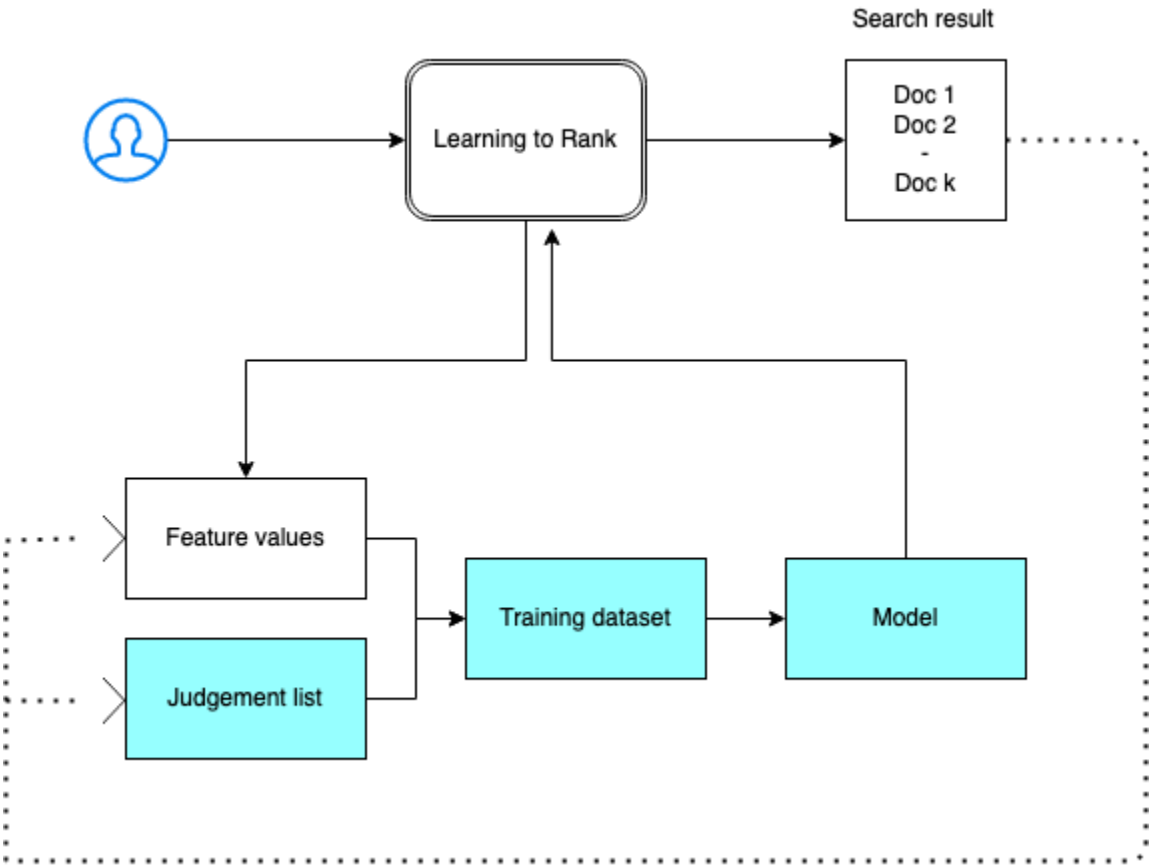
> **Note**
>
> このドキュメントでは、Learning to Rank プラグインの概要と、プラグインの使用を開始する方法を確認できます。詳細なステップや API の説明など、完全なドキュメントについては、Learning to Rank ドキュメントで入手可能です。

トピック

- Learning to Rank を開始する
- Learning to Rank API

Learning to Rank を開始する

判断リストを提供し、トレーニングデータセットを準備し、Amazon OpenSearch Service 以外でモデルをトレーニングする必要があります。青の部分は、OpenSearch サービスの外部で発生します。



ステップ 1: プラグインを初期化する

Learning to Rank プラグインを初期化するには、OpenSearch Service ドメインに次のリクエストを送信します。

```
PUT _ltr
```

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : ".ltrstore"
}
```

このコマンドは、機能セットやモデルなどのメタデータ情報を保存する非表示の `.ltrstore` インデックスを作成します。

ステップ 2: 判断リストを作成する

> **Note**
>
> このステップは、OpenSearch Service 以外で実行する必要があります。

判断リストは、機械学習モデルが学習する例を集めたものです。判断リストには、自分にとって重要なキーワードと、各キーワードのグレード付けされたドキュメントのセットを含める必要があります。

この例では、ムービーデータセットの判断リストがあります。4 のグレードは完全一致を示します。0 のグレードは最も悪い一致を示します。

グレード	キーワード	ドキュメントID	ムービー名
4	rambo	7555	Rambo
3	rambo	1370	Rambo III
3	rambo	1369	Rambo: First Blood Part II
3	rambo	1368	First Blood

判断リストを以下の形式で準備します。

```
4 qid:1 # 7555 Rambo
3 qid:1 # 1370 Rambo III
3 qid:1 # 1369 Rambo: First Blood Part II
3 qid:1 # 1368 First Blood

where qid:1 represents "rambo"
```

判定リストのより完全な例については、「映画の判断」を参照してください。。

この判断リストは、人間の注釈者の助けを借りて手動で作成することも、分析データからプログラムで推測することもできます。

ステップ 3: 機能セットを構築する

機能とは、ドキュメントの関連性に対応するフィールドです。例えば、title、overview、popularity score (再生数) などが表示されます。

各機能の Mustache テンプレートを用いて機能セットを構築します。機能の詳細については、「機能を用いた操作」を参照してください。

この例では、title および overview フィールドを用いて movie_features 機能セットを構築します

```
POST _ltr/_featureset/movie_features
{
  "featureset" : {
      "name" : "movie_features",
      "features" : [
        {
          "name" : "1",
          "params" : [
            "keywords"
          ],
          "template_language" : "mustache",
          "template" : {
            "match" : {
              "title" : "{{keywords}}"
            }
          }
        },
        {
          "name" : "2",
          "params" : [
            "keywords"
          ],
          "template_language" : "mustache",
          "template" : {
            "match" : {
              "overview" : "{{keywords}}"
            }
          }
        }
      ]
    }
}
```

元の .ltrstore インデックスのクエリを行うと、機能セットが返されます。

```
GET _ltr/_featureset
```

ステップ 4: 機能値のログを取る

機能値は、各機能について BM-25 によって計算された関連性スコアです。

機能セットと判断リストを組み合わせて、機能値のログを取ります。機能のログギングの詳細については、「機能のログギングスコア」を参照してください。

この例では、bool クエリは、フィルターを用いてグレード付けされたドキュメントを取得してから、sltr クエリを用いて機能セットを選択します。ltr_log クエリは、ドキュメントと機能を組み合わせて、対応する機能値のログを取ります。

```
POST tmdb/_search
{
  "_source": {
    "includes": [
      "title",
      "overview"
    ]
  },
  "query": {
    "bool": {
      "filter": [
        {
          "terms": {
            "_id": [
              "7555",
              "1370",
              "1369",
              "1368"
            ]
          }
        },
        {
          "sltr": {
            "_name": "logged_featureset",
            "featureset": "movie_features",
            "params": {
              "keywords": "rambo"
            }
```

```
          }
        }
      ]
    }
  },
  "ext": {
    "ltr_log": {
      "log_specs": {
        "name": "log_entry1",
        "named_query": "logged_featureset"
      }
    }
  }
}
```

以下にレスポンスの例を示します。

```
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 4,
      "relation" : "eq"
    },
    "max_score" : 0.0,
    "hits" : [
      {
        "_index" : "tmdb",
        "_type" : "movie",
        "_id" : "1368",
        "_score" : 0.0,
        "_source" : {
          "overview" : "When former Green Beret John Rambo is harassed by local law
 enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and
 rat-a-tat-tats his way into the action-movie hall of fame. Hounded by a relentless
 sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
```

```
          "title" : "First Blood"
        },
        "fields" : {
          "_ltrlog" : [
            {
              "log_entry1" : [
                {
                  "name" : "1"
                },
                {
                  "name" : "2",
                  "value" : 10.558305
                }
              ]
            }
          ]
        },
        "matched_queries" : [
          "logged_featureset"
        ]
      },
      {
        "_index" : "tmdb",
        "_type" : "movie",
        "_id" : "7555",
        "_score" : 0.0,
        "_source" : {
          "overview" : "When governments fail to act on behalf of captive missionaries,
ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween
River in a war-torn region of Thailand to take action.  Although he's still haunted
by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can
hardly turn his back on the aid workers who so desperately need his help.",
          "title" : "Rambo"
        },
        "fields" : {
          "_ltrlog" : [
            {
              "log_entry1" : [
                {
                  "name" : "1",
                  "value" : 11.2569065
                },
                {
                  "name" : "2",
```

```
                "value" : 9.936821
              }
            ]
          }
        ]
      },
      "matched_queries" : [
        "logged_featureset"
      ]
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1369",
      "_score" : 0.0,
      "_source" : {
        "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly
secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to
rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his
life, avenge the death of a woman and bring corrupt officials to justice.",
        "title" : "Rambo: First Blood Part II"
      },
      "fields" : {
        "_ltrlog" : [
          {
            "log_entry1" : [
              {
                "name" : "1",
                "value" : 6.334839
              },
              {
                "name" : "2",
                "value" : 10.558305
              }
            ]
          }
        ]
      },
      "matched_queries" : [
        "logged_featureset"
      ]
    },
    {
      "_index" : "tmdb",
```

```
        "_type" : "movie",
        "_id" : "1370",
        "_score" : 0.0,
        "_source" : {
          "overview" : "Combat has taken its toll on Rambo, but he's finally begun to
 find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for
 his help on a top secret mission to Afghanistan, Rambo declines but must reconsider
 when Trautman is captured.",
          "title" : "Rambo III"
        },
        "fields" : {
          "_ltrlog" : [
            {
              "log_entry1" : [
                {
                  "name" : "1",
                  "value" : 9.425955
                },
                {
                  "name" : "2",
                  "value" : 11.262714
                }
              ]
            }
          ]
        },
        "matched_queries" : [
          "logged_featureset"
        ]
      }
    ]
  }
}
```

前の例では、1 番目の機能には機能値がありません。これは、1368 に等しい ID が付いたドキュメントのタイトルフィールドに「rambo」というキーワードが表示されていないからです。これは、トレーニングデータで欠落している機能値です。

ステップ 2: トレーニングデータセットを作成する

> **Note**
>
> このステップは、OpenSearch Service 以外で実行する必要があります。

次のステップでは、判断リストと機能値を組み合わせて、トレーニングデータセットを作成します。元の判断リストは次のようになります。

```
4 qid:1 # 7555 Rambo
3 qid:1 # 1370 Rambo III
3 qid:1 # 1369 Rambo: First Blood Part II
3 qid:1 # 1368 First Blood
```

それを最終的なトレーニングデータセットに変換します。これは次のようになります。

```
4 qid:1 1:12.318474 2:10.573917 # 7555 rambo
3 qid:1 1:10.357875 2:11.950391 # 1370 rambo
3 qid:1 1:7.010513 2:11.220095 # 1369 rambo
3 qid:1 1:0.0 2:11.220095 # 1368 rambo
```

このステップは手動で実行することも、プログラムを記述して自動化することもできます。

ステップ6：アルゴリズムを選択してモデルを構築する

> **Note**
>
> このステップは、OpenSearch Service 以外で実行する必要があります。

トレーニングデータセットを配置したら、次のステップは XGBoost ライブラリまたは Ranklib ライブラリを使用してモデルを構築することです。XgBoost ライブラリと Ranklib ライブラリを使用すると、Lambdamart や Random Forest などの人気モデルを構築できます。

XGBoost と Ranklib を使用してモデルを構築するステップについては、それぞれ「XGBoost」および「RankLib」ドキュメントを参照してください。Amazon SageMaker を使用して XGBoost モデルを構築するには、「XGBoost アルゴリズム」を参照してください。

ステップ 7: モデルをデプロイする

モデルを構築したら、それを Learning to Rank プラグインに展開します。モデルのデプロイの詳細については、「トレーニングされたモデルのアップロード」を参照してください。。

この例では、Ranklib ライブラリを使用して my_ranklib_model モデルを構築します。

```
POST _ltr/_featureset/movie_features/_createmodel?pretty
{
  "model": {
    "name": "my_ranklib_model",
    "model": {
      "type": "model/ranklib",
      "definition": """## LambdaMART
## No. of trees = 10
## No. of leaves = 10
## No. of threshold candidates = 256
## Learning rate = 0.1
## Stop early = 100

<ensemble>
   <tree id="1" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>0.0</threshold>
            <split pos="left">
               <output>-2.0</output>
            </split>
            <split pos="right">
               <feature>1</feature>
               <threshold>7.010513</threshold>
               <split pos="left">
                  <output>-2.0</output>
               </split>
               <split pos="right">
                  <output>-2.0</output>
               </split>
            </split>
         </split>
         <split pos="right">
```

```
            <output>2.0</output>
         </split>
      </split>
   </tree>
   <tree id="2" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>0.0</threshold>
            <split pos="left">
               <output>-1.67031991481781</output>
            </split>
            <split pos="right">
               <feature>1</feature>
               <threshold>7.010513</threshold>
               <split pos="left">
                  <output>-1.67031991481781</output>
               </split>
               <split pos="right">
                  <output>-1.6703200340270996</output>
               </split>
            </split>
         </split>
         <split pos="right">
            <output>1.6703201532363892</output>
         </split>
      </split>
   </tree>
   <tree id="3" weight="0.1">
      <split>
         <feature>2</feature>
         <threshold>10.573917</threshold>
         <split pos="left">
            <output>1.479954481124878</output>
         </split>
         <split pos="right">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
               <feature>1</feature>
               <threshold>0.0</threshold>
               <split pos="left">
```

```
                  <output>-1.4799546003341675</output>
               </split>
               <split pos="right">
                  <output>-1.479954481124878</output>
               </split>
            </split>
            <split pos="right">
               <output>-1.479954481124878</output>
            </split>
         </split>
      </split>
   </tree>
   <tree id="4" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>0.0</threshold>
            <split pos="left">
               <output>-1.3569872379302979</output>
            </split>
            <split pos="right">
               <feature>1</feature>
               <threshold>7.010513</threshold>
               <split pos="left">
                  <output>-1.3569872379302979</output>
               </split>
               <split pos="right">
                  <output>-1.3569872379302979</output>
               </split>
            </split>
         </split>
         <split pos="right">
            <output>1.3569873571395874</output>
         </split>
      </split>
   </tree>
   <tree id="5" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
```

```
            <threshold>0.0</threshold>
            <split pos="left">
               <output>-1.2721362113952637</output>
            </split>
            <split pos="right">
               <feature>1</feature>
               <threshold>7.010513</threshold>
               <split pos="left">
                  <output>-1.2721363306045532</output>
               </split>
               <split pos="right">
                  <output>-1.2721363306045532</output>
               </split>
            </split>
         </split>
         <split pos="right">
            <output>1.2721362113952637</output>
         </split>
      </split>
   </tree>
   <tree id="6" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
               <feature>1</feature>
               <threshold>0.0</threshold>
               <split pos="left">
                  <output>-1.2110036611557007</output>
               </split>
               <split pos="right">
                  <output>-1.2110036611557007</output>
               </split>
            </split>
            <split pos="right">
               <output>-1.2110037803649902</output>
            </split>
         </split>
         <split pos="right">
            <output>1.2110037803649902</output>
         </split>
```

```
      </split>
   </tree>
   <tree id="7" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
               <feature>1</feature>
               <threshold>0.0</threshold>
               <split pos="left">
                  <output>-1.165616512298584</output>
               </split>
               <split pos="right">
                  <output>-1.165616512298584</output>
               </split>
            </split>
            <split pos="right">
               <output>-1.165616512298584</output>
            </split>
         </split>
         <split pos="right">
            <output>1.165616512298584</output>
         </split>
      </split>
   </tree>
   <tree id="8" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
               <feature>1</feature>
               <threshold>0.0</threshold>
               <split pos="left">
                  <output>-1.131177544593811</output>
               </split>
               <split pos="right">
                  <output>-1.131177544593811</output>
               </split>
```

```
            </split>
            <split pos="right">
               <output>-1.131177544593811</output>
            </split>
         </split>
         <split pos="right">
            <output>1.131177544593811</output>
         </split>
      </split>
   </tree>
   <tree id="9" weight="0.1">
      <split>
         <feature>2</feature>
         <threshold>10.573917</threshold>
         <split pos="left">
            <output>1.1046180725097656</output>
         </split>
         <split pos="right">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
               <feature>1</feature>
               <threshold>0.0</threshold>
               <split pos="left">
                  <output>-1.1046180725097656</output>
               </split>
               <split pos="right">
                  <output>-1.1046180725097656</output>
               </split>
            </split>
            <split pos="right">
               <output>-1.1046180725097656</output>
            </split>
         </split>
      </split>
   </tree>
   <tree id="10" weight="0.1">
      <split>
         <feature>1</feature>
         <threshold>10.357875</threshold>
         <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
```

```
                <feature>1</feature>
                <threshold>0.0</threshold>
                <split pos="left">
                   <output>-1.0838804244995117</output>
                </split>
                <split pos="right">
                   <output>-1.0838804244995117</output>
                </split>
             </split>
             <split pos="right">
                <output>-1.0838804244995117</output>
             </split>
          </split>
          <split pos="right">
             <output>1.0838804244995117</output>
          </split>
       </split>
    </tree>
</ensemble>
"""
    }
  }
}
```

モデルを表示するには、次のリクエストを送信します。

```
GET _ltr/_model/my_ranklib_model
```

ステップ 8: Learning to Rank を用いて検索する

モデルをデプロイしたら、検索する準備ができています。

使用する機能と実行するモデルの名前を用いて `sltr` クエリを実行します。

```
POST tmdb/_search
{
  "_source": {
    "includes": ["title", "overview"]
  },
  "query": {
    "multi_match": {
      "query": "rambo",
      "fields": ["title", "overview"]
```

```
    }
  },
  "rescore": {
    "query": {
      "rescore_query": {
        "sltr": {
          "params": {
            "keywords": "rambo"
          },
          "model": "my_ranklib_model"
        }
      }
    }
  }
}
```

Learning to Rank では、判断リストの最高グレードを割り当てたため、「Rambo」が最初の結果として表示されます。

```
{
  "took" : 12,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 7,
      "relation" : "eq"
    },
    "max_score" : 13.096414,
    "hits" : [
      {
        "_index" : "tmdb",
        "_type" : "movie",
        "_id" : "7555",
        "_score" : 13.096414,
        "_source" : {
          "overview" : "When governments fail to act on behalf of captive missionaries,
 ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween
```

```
River in a war-torn region of Thailand to take action.  Although he's still haunted
by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can
hardly turn his back on the aid workers who so desperately need his help.",
        "title" : "Rambo"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1370",
      "_score" : 11.17245,
      "_source" : {
        "overview" : "Combat has taken its toll on Rambo, but he's finally begun to
find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for
his help on a top secret mission to Afghanistan, Rambo declines but must reconsider
when Trautman is captured.",
        "title" : "Rambo III"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1368",
      "_score" : 10.442155,
      "_source" : {
        "overview" : "When former Green Beret John Rambo is harassed by local law
enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and
rat-a-tat-tats his way into the action-movie hall of fame. Hounded by a relentless
sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
        "title" : "First Blood"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1369",
      "_score" : 10.442155,
      "_source" : {
        "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly
secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to
rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his
life, avenge the death of a woman and bring corrupt officials to justice.",
        "title" : "Rambo: First Blood Part II"
      }
```

```
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "31362",
      "_score" : 7.424202,
      "_source" : {
        "overview" : "It is 1985, and a small, tranquil Florida town is being rocked
by a wave of vicious serial murders and bank robberies. Particularly sickening to the
authorities is the gratuitous use of violence by two “Rambo” like killers who dress
themselves in military garb. Based on actual events taken from FBI files, the movie
depicts the Bureau’s efforts to track down these renegades.",
        "title" : "In the Line of Duty: The F.B.I. Murders"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "13258",
      "_score" : 6.43182,
      "_source" : {
        "overview" : """Will Proudfoot (Bill Milner) is looking for an escape from
his family's stifling home life when he encounters Lee Carter (Will Poulter), the
school bully. Armed with a video camera and a copy of "Rambo: First Blood", Lee plans
to make cinematic history by filming his own action-packed video epic. Together, these
two newfound friends-turned-budding-filmmakers quickly discover that their imaginative
– and sometimes mishap-filled – cinematic adventure has begun to take on a life of its
own!""",
        "title" : "Son of Rambow"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "61410",
      "_score" : 3.9719706,
      "_source" : {
        "overview" : "It's South Africa 1990. Two major events are about to happen:
The release of Nelson Mandela and, more importantly, it's Spud Milton's first year
at an elite boys only private boarding school. John Milton is a boy from an ordinary
background who wins a scholarship to a private school in Kwazulu-Natal, South Africa.
Surrounded by boys with nicknames like Gecko, Rambo, Rain Man and Mad Dog, Spud has
his hands full trying to adapt to his new home. Along the way Spud takes his first
tentative steps along the path to manhood. (The path it seems could be a rather long
```

```
 road). Spud is an only child. He is cursed with parents from well beyond the lunatic
 fringe and a senile granny. His dad is a fervent anti-communist who is paranoid that
 the family domestic worker is running a shebeen from her room at the back of the
 family home. His mom is a free spirit and a teenager's worst nightmare, whether it's
 shopping for Spud's underwear in the local supermarket",
          "title" : "Spud"
        }
      }
    ]
  }
}
```

Learning to Rank プラグインを使用せずに検索すると、OpenSearch は異なる結果を返します。

```
POST tmdb/_search
{
  "_source": {
    "includes": ["title", "overview"]
  },
  "query": {
    "multi_match": {
      "query": "Rambo",
      "fields": ["title", "overview"]
    }
  }
}
```

```
{
  "took" : 5,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 5,
      "relation" : "eq"
    },
    "max_score" : 11.262714,
    "hits" : [
```

```
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1370",
      "_score" : 11.262714,
      "_source" : {
        "overview" : "Combat has taken its toll on Rambo, but he's finally begun to
find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for
his help on a top secret mission to Afghanistan, Rambo declines but must reconsider
when Trautman is captured.",
        "title" : "Rambo III"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "7555",
      "_score" : 11.2569065,
      "_source" : {
        "overview" : "When governments fail to act on behalf of captive missionaries,
ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween
River in a war-torn region of Thailand to take action.  Although he's still haunted
by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can
hardly turn his back on the aid workers who so desperately need his help.",
        "title" : "Rambo"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1368",
      "_score" : 10.558305,
      "_source" : {
        "overview" : "When former Green Beret John Rambo is harassed by local law
enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and
rat-a-tat-tats his way into the action-movie hall of fame. Hounded by a relentless
sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
        "title" : "First Blood"
      }
    },
    {
      "_index" : "tmdb",
      "_type" : "movie",
      "_id" : "1369",
```

```
        "_score" : 10.558305,
        "_source" : {
          "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly
 secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to
 rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his
 life, avenge the death of a woman and bring corrupt officials to justice.",
          "title" : "Rambo: First Blood Part II"
        }
      },
      {
        "_index" : "tmdb",
        "_type" : "movie",
        "_id" : "13258",
        "_score" : 6.4600153,
        "_source" : {
          "overview" : """Will Proudfoot (Bill Milner) is looking for an escape from
 his family's stifling home life when he encounters Lee Carter (Will Poulter), the
 school bully. Armed with a video camera and a copy of "Rambo: First Blood", Lee plans
 to make cinematic history by filming his own action-packed video epic. Together, these
 two newfound friends-turned-budding-filmmakers quickly discover that their imaginative
 – and sometimes mishap-filled – cinematic adventure has begun to take on a life of its
 own!""",
          "title" : "Son of Rambow"
        }
      }
    ]
  }
}
```

モデルがどの程度うまく実行されているかに基づいて、判断リストと機能を調整します。次に、ステップ 2~8 を繰り返して、時間の経過とともにランキング結果を改善します。

Learning to Rank API

Learning to Rank オペレーションを使用して、機能セットとモデルをプログラムで操作します。

ストアを作成する

機能セットやモデルなどのメタデータ情報を保存する非表示の `.ltrstore` インデックスを作成します。

```
PUT _ltr
```

ストアを削除する

非表示の .ltrstore インデックスを削除し、プラグインをリセットします。

```
DELETE _ltr
```

機能セットを作成する

機能セットを作成します。

```
POST _ltr/_featureset/<name_of_features>
```

機能セットを削除する

機能セットを削除します。

```
DELETE _ltr/_featureset/<name_of_feature_set>
```

機能セットを取得する

機能セットを取得します。

```
GET _ltr/_featureset/<name_of_feature_set>
```

モデルを作成する

モデルを作成します。

```
POST _ltr/_featureset/<name_of_feature_set>/_createmodel
```

モデルを削除する

モデルを削除します。

```
DELETE _ltr/_model/<name_of_model>
```

モデルを取得する

モデルを取得します。

```
GET _ltr/_model/<name_of_model>
```

統計を取得する

プラグインがどのように動作するかについての情報を提供します。

```
GET _ltr/_stats
```

また、フィルターを使用して単一の統計を取得することもできます:

```
GET _ltr/_stats/<stat>
```

さらに、情報をクラスター内の単一のノードに制限することもできます。

```
GET _ltr/_stats/<stat>/nodes/<nodeId>

{
  "_nodes" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "cluster_name" : "873043598401:ltr-77",
  "stores" : {
    ".ltrstore" : {
      "model_count" : 1,
      "featureset_count" : 1,
      "feature_count" : 2,
      "status" : "green"
    }
  },
  "status" : "green",
  "nodes" : {
    "DjelK-_ZSfyzstO5dhGGQA" : {
      "cache" : {
        "feature" : {
          "eviction_count" : 0,
          "miss_count" : 0,
          "entry_count" : 0,
          "memory_usage_in_bytes" : 0,
          "hit_count" : 0
```

```
        },
        "featureset" : {
          "eviction_count" : 2,
          "miss_count" : 2,
          "entry_count" : 0,
          "memory_usage_in_bytes" : 0,
          "hit_count" : 0
        },
        "model" : {
          "eviction_count" : 2,
          "miss_count" : 3,
          "entry_count" : 1,
          "memory_usage_in_bytes" : 3204,
          "hit_count" : 1
        }
      },
      "request_total_count" : 6,
      "request_error_count" : 0
    }
  }
}
```

統計は、次のテーブルに示すように、ノードとクラスターの 2 つのレベルで提供されます。

ノードレベルの統計

フィールド名	説明
request_total_count	ランク付けリクエストの合計数。
request_error_count	失敗したリクエストの合計数。
cache	すべてのキャッシュ (機能、機能セット、モデル) の統計情報。キャッシュヒットは、ユーザーがプラグインのクエリを行い、モデルがすでにメモリにロードされているときに発生します。
cache.eviction_count	キャッシュ削除の数。
cache.hit_count	キャッシュヒットの数。

フィールド名	説明
cache.miss_count	キャッシュミスの数。キャッシュミスは、ユーザーがプラグインのクエリを行い、モデルがまだメモリにロードされていない場合に発生します。
cache.entry_count	キャッシュのエントリの数。
cache.memory_usage_in_bytes	使用メモリの合計 (バイト単位)。
cache.cache_capacity_reached	キャッシュ制限に達したかどうかを示します。

クラスターレベルの統計

フィールド名	説明
保存	機能セットとモデルメタデータが保存される場所を示します。(デフォルトは「.ltrstore」です。 それ以外の場合は、ユーザーが指定した名前で、接頭辞に「.ltrstore_」が付きます)。
stores.status	インデックスのステータス。
stores.feature_sets	機能セットの数
stores.features_count	機能の数。
stores.model_count	モデルの数。
ステータス	Feature Store インデックスのステータス (赤、黄、緑) およびサーキットブレーカーの状態 (開または閉) に基づくプラグインのステータス。
cache.cache_capacity_reached	キャッシュ制限に達したかどうかを示します。

キャッシュ統計を取得

キャッシュとメモリの使用状況についての統計を返します。

```
GET _ltr/_cachestats

{
    "_nodes": {
        "total": 2,
        "successful": 2,
        "failed": 0
    },
    "cluster_name": "opensearch-cluster",
    "all": {
        "total": {
            "ram": 612,
            "count": 1
        },
        "features": {
            "ram": 0,
            "count": 0
        },
        "featuresets": {
            "ram": 612,
            "count": 1
        },
        "models": {
            "ram": 0,
            "count": 0
        }
    },
    "stores": {
        ".ltrstore": {
            "total": {
                "ram": 612,
                "count": 1
            },
            "features": {
                "ram": 0,
                "count": 0
            },
            "featuresets": {
                "ram": 612,
                "count": 1
```

```
            },
            "models": {
                "ram": 0,
                "count": 0
            }
        }
    },
    "nodes": {
        "ejF6uutERF20wOFNOXB61A": {
            "name": "opensearch1",
            "hostname": "172.18.0.4",
            "stats": {
                "total": {
                    "ram": 612,
                    "count": 1
                },
                "features": {
                    "ram": 0,
                    "count": 0
                },
                "featuresets": {
                    "ram": 612,
                    "count": 1
                },
                "models": {
                    "ram": 0,
                    "count": 0
                }
            }
        },
        "Z2RZNWRLSveVcz2c6lHf5A": {
            "name": "opensearch2",
            "hostname": "172.18.0.2",
            "stats": {
                ...
            }
        }
    }
}
```

キャッシュをクリア

プラグインキャッシュをクリアします。これを使用して、モデルをリフレッシュします。

```
POST _ltr/_clearcache
```

Amazon OpenSearch Service の非同期検索

Amazon OpenSearch Service の非同期検索を使用すると、バックグラウンドで実行される検索クエリを送信し、リクエストの進行状況をモニタリングし、後のステージで結果を取得できます。検索が完了する前に部分的な結果が取得できるようになります。検索が終了したら、後で取得および分析できるように結果を保存します。

非同期検索では、OpenSearch 1.0 以降、または Elasticsearch 7.10 以降が必要です。

このドキュメントでは、非同期検索の概要を説明します。また、オープンソースの OpenSearch クラスターではなく、マネージド Amazon OpenSearch Service ドメインで非同期検索を使用する制限についても説明します。利用可能な設定、アクセス許可、完全な API リファレンスなど、非同期検索の完全なドキュメントについては、OpenSearch ドキュメントの「Asynchronous search」を参照してください。

サンプルの検索コール

非同期検索を実行するには、以下の形式を使用して HTTP リクエストを `_plugins/_asynchronous_search` に送信します。

```
POST opensearch-domain/_plugins/_asynchronous_search
```

> **Note**
>
> OpenSearch バージョンではなく Elasticsearch 7.10 を使用している場合は、すべての非同期検索リクエストで `_plugins` を `_opendistro` で置き換えます。

次の非同期検索オプションを指定できます。

オプション	説明	デフォルトの値	必須
`wait_for_completio`	結果を待機する予定の時間を指定します。通常の検索と同様に、この時間内に得られるどの	1 秒	いいえ

オプション	説明	デフォルトの値	必須
n_timeout	ような結果も確認できます。ID に基づいて残りの結果をポーリングできます。最大値は 300 秒です。		
keep_on_completion	検索の完了後に結果をクラスターに保存するかどうかを指定します。保存された結果は、後で確認できます。	false	いいえ
keep_alive	結果がクラスターに保存される時間を指定します。例えば、2d は、結果が 48 時間クラスターに保存されることを意味します。保存された検索結果は、この期間の後、または検索がキャンセルされた場合、削除されます。これにはクエリランタイムが含まれることに注意してください。この時間にクエリがオーバーランすると、プロセスではこのクエリが自動的にキャンセルされます。	12 時間	いいえ

リクエスト例

```
POST _plugins/_asynchronous_search/?
pretty&size=10&wait_for_completion_timeout=1ms&keep_on_completion=true&request_cache=false
{
  "aggs": {
    "city": {
      "terms": {
        "field": "city",
        "size": 10
      }
    }
  }
}
```

> **Note**
>
> 標準 _search クエリに適用されるすべてのリクエストパラメータがサポートされています。OpenSearch バージョンではなく Elasticsearch 7.10 を使用している場合は、_plugins を _opendistro で置き換えます。

非同期検索アクセス許可

非同期検索は、きめ細かなアクセスコントロールをサポートしています。ユースケースに適合する許可のミキシングとマッチングの詳細については、非同期検索セキュリティを参照してください。

きめ細かなアクセスコントロールが有効になっているドメインの場合は、ロールに対して以下の最小限の許可が必要です。

```
# Allows users to use all asynchronous search functionality
asynchronous_search_full_access:
  reserved: true
  cluster_permissions:
    - 'cluster:admin/opensearch/asynchronous-search/*'
  index_permissions:
    - index_patterns:
        - '*'
      allowed_actions:
        - 'indices:data/read/search*'

# Allows users to read stored asynchronous search results
asynchronous_search_read_access:
  reserved: true
  cluster_permissions:
    - 'cluster:admin/opensearch/asynchronous-search/get'
```

きめ細かなアクセスコントロールが無効になっているドメインの場合は、IAM アクセスとシークレットキーを使用してすべてのリクエストに署名します。非同期検索 ID を用いて結果にアクセスできます。

非同期検索設定

OpenSearch では、_cluster/settings APIを使用して利用可能なすべての非同期検索設定を変更できます。OpenSearch Service では、以下の設定のみを変更できます。

- plugins.asynchronous_search.node_concurrent_running_searches
- plugins.asynchronous_search.persist_search_failures

クラスター間検索

クラスター間で非同期検索を実行できますが、次の小さな制限があります。

- 非同期検索は、ソースドメインでのみ実行できます。
- クロスクラスター検索クエリの一部としてネットワークラウンドトリップを最小化することはできません。

接続エイリアス cluster_b を用いた domain-a -> domain-b と接続エイリアス cluster_c を用いた domain-a -> domain-c の間の接続をセットアップする場合は、次のように domain-a、domain-b、および domain-c を非同期検索します。

```
POST https://src-domain.us-east-1.es.amazonaws.com/
local_index,cluster_b:b_index,cluster_c:c_index/_plugins/_asynchronous_search/?
pretty&size=10&wait_for_completion_timeout=500ms&keep_on_completion=true&request_cache=false
{
  "size": 0,
  "_source": {
    "excludes": []
  },
  "aggs": {
    "2": {
      "terms": {
        "field": "clientip",
        "size": 50,
        "order": {
          "_count": "desc"
        }
      }
    }
  },
  "stored_fields": [
    "*"
  ],
  "script_fields": {},
  "docvalue_fields": [
    "@timestamp"
```

```
  ],
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "status:404",
            "analyze_wildcard": true,
            "default_field": "*"
          }
        },
        {
          "range": {
            "@timestamp": {
              "gte": 1483747200000,
              "lte": 1488326400000,
              "format": "epoch_millis"
            }
          }
        }
      ],
      "filter": [],
      "should": [],
      "must_not": []
    }
  }
}
```

レスポンス

```
{
  "id" :
 "Fm9pYzJyVG91U19xb0hIQUJnMHJfRFEAAAAAAAknghQ1OWVBczNZQjVEa2dMYTBXaTdEagAAAAAAAAAB",
  "state" : "RUNNING",
  "start_time_in_millis" : 1609329314796,
  "expiration_time_in_millis" : 1609761314796
}
```

詳細については、「the section called “クラスター間検索”」を参照してください。

UltraWarm

UltraWarm インデックスを用いた非同期検索は引き続き機能します。詳細については、「the section called “UltraWarm ストレージ”」を参照してください。

> **Note**
>
> CloudWatch で非同期検索統計をモニタリングできます。 メトリクスの一覧については、「the section called “非同期検索メトリクス”」を参照してください。

Amazon OpenSearch Service のポイントインタイム検索

ポイントインタイム (PIT) とは、時間が固定されているデータセットにさまざまなクエリを実行できる、検索の一種です。ドキュメントは絶えずインデックス付けされ、更新され、削除されているため、同じインデックスに対して同じクエリを異なる時点で実行すると、通常は異なる結果が返されます。PIT では、一定の状態に保たれたデータセット対して、クエリを実行できます。

PIT 検索は、多くの場合 `search_after` 機能と組み合わせて使用されます。これは、OpenSearch、とりわけディープページネーションで推奨されているページネーションの方法です。時間とともに固定したデータセットで機能し、クエリに拘束されず、前方と後方への一貫したページネーションをサポートしているためです。OpenSearch バージョン 2.5 を実行するドメインで PIT を使用できます。

> **Note**
>
> このトピックでは、PIT の概要と、セルフマネージド OpenSearch クラスターではなく、マネージド Amazon OpenSearch Service ドメインで使用するときに考慮すべき点について説明します。包括的な API リファレンスを含む PIT の完全なドキュメントについては、オープンソースの OpenSearch ドキュメントの「Point in Time」を参照してください。

考慮事項

PIT　検索を設定するときは、以下を考慮します。

- OpenSearch バージョン 2.3 を実行するドメインからのアップグレードで、PIT アクションへのきめ細かなアクセスコントロールが必要である場合、そのアクションとロールは手動で追加する必要があります。

- PIT には回復力はありません。ノードの再起動、ノードの終了、ブルー/グリーンデプロイ、OpenSearch プロセスの再起動などにより、PIT のデータはすべて失われます。
- ブルー/グリーンデプロイ中にシャードが再配置された場合、新しいノードに転送されるのはライブデータのセグメントのみです。PIT が保持していたシャードのセグメント (独占的に保持していたものとライブデータと共有されたものの両方) は、古いノードに残ります。
- PIT 検索は現在、非同期検索では使用できません。

PIT の作成

PIT クエリを実行するには、次の形式を使って HTTP リクエストを _search/point_in_time に送信します。

```
POST opensearch-domain/my-index/_search/point_in_time?keep_alive=5m
```

次の PIT オプションを指定できます。

オプション	説明	デフォルトの値	必須
keep_alive	PIT を維持する時間。検索リクエストで PIT にアクセスするたびに、PIT の有効期間が keep_alive パラメータと同じ時間だけ延長されます。時間単位のサフィックスで値を指定します。サポートされている単位: ms (ミリ秒)s、 (秒)m、 (分) 、 h (時間) 、 d (日) 。例: 5 5m 分間。このクエリパラメータは、PIT を作成するときは必須ですが、検索リクエストでは任意となります。		はい
preference	検索の実行に使用されるノードまたはシャードを指定する文字列。	ランダム	いいえ
routing	検索リクエストを特定のシャードにルーティングするよう指定する文字列。	ドキュメントの _id	いいえ

オプション	説明	デフォルトの値	必須
expand_wildcards	ワイルドカードパターンと一致するインデックスのタイプを指定する文字列。コンマ区切り値をサポート。有効な値は以下のとおりです。 • all: 任意のインデックスまたはデータストリームと一致。非表示のものを含む。 • open: 開いているインデックス、非表示でないインデックス、非表示でないデータストリームと一致。 • closed: 閉じているインデックス、非表示でないインデックス、非表示でないデータストリームと一致。 • hidden: 非表示のインデックスまたはデータストリームと一致。開いているものか閉じているもの、または、開いているものと閉じているもの両方と組み合わせねばならない。 • none: ワイルドカードのパターンは使用できない。	open	いいえ
allow_partial_pit_creation	部分的な障害を含む PIT を作成するかどうかを指定するブール値。	true	いいえ

レスポンス例

```
{
    "pit_id":
 "o463QQEPbXktaW5kZXgtMDAwMDAxFnNOWU43ckt3U3IyaFVpbGE1UWEtMncAFjFyeXBsRGJmVFM2RTB6eVg1aVVqQncAA
    "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
    },
```

```
    "creation_time": 1658146050064
}
```

PIT を作成すると、PIT ID が届きます。これは、PIT を使って検索するときに使用する ID です。

ポイントインタイムアクセス許可

PIT は、詳細なアクセス制御をサポートしています。OpenSearch バージョン 2.5 ドメインにアップグレードする予定で、きめ細かなアクセスコントロールが必要である場合は、次のアクセス許可を持つロールを、手動で作成する必要があります。

```
# Allows users to use all point in time search search functionality
point_in_time_full_access:
  reserved: true
  index_permissions:
    - index_patterns:
        - '*'
      allowed_actions:
        - "indices:data/read/point_in_time/create"
        - "indices:data/read/point_in_time/delete"
        - "indices:data/read/point_in_time/readall"
        - "indices:data/read/search"
        - "indices:monitor/point_in_time/segments"


# Allows users to use point in time search search functionality for specific index
# All type operations like list all PITs, delete all PITs are not supported in this
 case

point_in_time_index_access:
  reserved: true
  index_permissions:
    - index_patterns:
        - 'my-index-1'
      allowed_actions:
        - "indices:data/read/point_in_time/create"
        - "indices:data/read/point_in_time/delete"
        - "indices:data/read/search"
        - "indices:monitor/point_in_time/segments"
```

OpenSearch バージョン 2.5 以降のドメインでは、組み込みの point_in_time_full_access ロールを使用できます。詳細については、OpenSearch ドキュメントの「セキュリティモデル」を参照してください。

PIT の設定

OpenSearch では、利用可能なすべての PIT 設定を、_cluster/settings API を使用して変更することができます。OpenSearch Service では、現在設定を変更できません。

クラスター間検索

PIT の作成、PIT ID を使った検索、PIT の一覧表示、クラスター全体での PIT の削除には、次のような軽微の制限があります。

- PIT の一覧を表示したり、すべての PIT を削除したりできるのは、ソースドメイン上でのみです。
- クロスクラスター検索クエリの一部としてネットワークラウンドトリップを最小化することはできません。

詳細については、「the section called “クラスター間検索”」を参照してください。

UltraWarm

UltraWarm インデックスを用いた PIT 検索は、引き続き使用できます。詳細については、「the section called “UltraWarm ストレージ”」を参照してください。

> **Note**
>
> PIT 検索の統計は、CloudWatch でモニタリングできます。メトリクスの一覧については、「the section called “ポイントインタイムメトリクス”」を参照してください。

Amazon OpenSearch Service でのエージェント検索

OpenSearch バージョン 3.3 以降、エージェント検索を使用すると、自動エージェントを使用してユーザーに代わって複雑な検索を実行する AI を活用したプロセスが可能になります。

エージェント検索は、ユーザーのインテントを理解し、適切なツールを調整し、最適化されたクエリを生成するインテリジェントなエージェントシステムを導入します。自然言語インターフェイスを通じて、意思決定の透過的な概要を提供します。

OpenSearch Service を使用すると、 および外部サービスの AI コネクタ AWS のサービスを設定できます。コンソールを使用して、エージェントの構築に使用できる CloudFormation テンプレートを使用して ML モデルを作成することもできます。詳細については、「Bedrock Claude によるエージェント検索の設定」を参照してください。

詳細なドキュメントとstep-by-step実装については、OpenSearch ドキュメントの「エージェント検索」を参照してください。

Amazon OpenSearch Serviceでのセマンティック検索

バージョン 2.19 以降では、自動セマンティックエンリッチメントを使用して、最小限の設定作業でセマンティック検索を実装できます。この機能は、テキストフィールドのスパースエンコーディングを自動的に生成するため、手動取り込みパイプラインのセットアップが不要になります。詳細については、OpenSearch ドキュメントの「自動セマンティックエンリッチメント」を参照してください。

OpenSearch バージョン 2.9 以降では、セマンティック検索を使用して検索クエリに関する理解を深め、検索の関連性を高めることができます。セマンティック検索は、ニューラル検索と K 近傍法 (k-NN) 検索の 2 つの方法のいずれかで使用できます。

OpenSearch Service を使用すると、AWS のサービスの AI コネクタと外部サービスをセットアップできます。コンソールを使用して、 CloudFormation テンプレートを使用して ML モデルを作成することもできます。詳細については、「the section called “CloudFormation テンプレートの統合”」を参照してください。

セマンティック検索を使用するためのステップバイステップガイドを含むセマンティック検索の完全なドキュメントについては、オープンソースの OpenSearch ドキュメントの「Semantic search」を参照してください。

Amazon OpenSearch Service での同時セグメント検索

OpenSearch バージョン 2.17 以降、並列セグメント検索では新しい設定により並列検索動作が制御されます。

- バージョン 2.17 で作成された新しいドメインでは、2xl 以上のノードについて、並列セグメント検索はデフォルトで自動モードに設定されます。
- 既存のドメインを 2.17 にアップグレードした場合、もしクラスターの全体的な CPU 使用率が過去 1 週間で 45% を下回っていれば、インスタンスタイプに基づいて、2xl 以上のすべてのノードについて並列セグメント検索をデフォルトで自動に設定します。

- 詳細については、「Concurrent segment search version 2.17」を参照してください。

OpenSearch バージョン 2.13 以降では、同時セグメント検索を使用して、クエリフェーズ中にセグメントを並行して検索できます。同時セグメント検索の完全なドキュメントについては、オープンソースの OpenSearch ドキュメントの「Concurrent segment search」を参照してください。同時セグメント検索に関連する Amazon CloudWatch メトリクスの詳細については、「Instance metrics」と「UltraWarm metrics」を参照してください。

Amazon OpenSearch Service で現在のセグメント検索を使用する場合、いくつかの追加の制限が適用されます。

- デフォルトでは、OpenSearch Service は最大スライス数メカニズムで 2 つのスライス数を使用します。

Amazon OpenSearch Service における自然言語クエリ生成

Amazon OpenSearch Service の自然言語クエリ生成機能を使用すると、自然言語を使用してオペレーションログとセキュリティログのデータをクエリできます。OpenSearch は、非常にスケーラブルでパフォーマンスの高いログ分析と検索エンジンであり、自然言語を使用してこれらのログを調べることができるため、ログデータを探索する理想的なオプションです。この機能を使用すると、OpenSearch Piped Processing Language (PPL) に依存したり、クエリを構築するときにデータ定義を検索したりすることなく、問題を特定できます。バージョン 2.13 以降の OpenSearch Service ドメインでは、自然言語クエリ生成機能を使用できます。きめ細かなアクセスコントロールを有効にする必要があります。

この機能は、OpenSearch Assistant Toolkit を使用して構築されました。大規模な言語モデルに接続する同様の機能を作成する場合は、ツールキットを使用して独自のエージェントとツールを設定できます。

前提条件

自然言語クエリ生成機能を使用する前に、ドメインには次が必要です。

- バージョン 2.13 以降。
- サービスソフトウェア R20240520-P4 以降。
- きめ細かなアクセスコントロールが有効。詳細については、「the section called “きめ細かなアクセスコントロールの有効化”」を参照してください。

開始方法

自然言語クエリは、詳細なアクセスコントロールが有効になっているバージョン 2.13 以降で作成されたすべてのドメインでデフォルトで有効になっています。

他のドメインの場合は、[自然言語クエリ生成を有効化 と Amazon Q Developer 機能] を選択して有効にします。

有効にしたら、OpenSearch Dashboards の [ログ] ページに移動します。[Event Explorer] を選択し、クエリアシスタントを使用して質問をします。

アクセス許可の設定

既存の OpenSearch Service ドメインで自然言語クエリの生成を有効にすると、ドメインで [query_assistant_access] ロールが定義されない場合があります。きめ細かなアクセスコントロールを使用してドメインのウォームインデックスを管理するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。[query_assistant_access] ロールを手動で作成するには、以下の手順を実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[ロール] を選択します。
2. [ロールの作成] を選択し、次のクラスターアクセス許可を設定します。
   - `cluster:admin/opensearch/ml/config/get`
   - `cluster:admin/opensearch/ml/execute`
   - `cluster:admin/opensearch/ml/predict`
   - `cluster:admin/opensearch/ppl`
3. ロールに [query_assistant_access] という名前を付けます。
4. [ロールの作成] を選択してください。[query_assistant_access] ロールが利用可能になりました。

> **Note**
>
> また、自然言語の質問を使用するインデックスの `indices:admin/mappings/get` および `read` インデックスアクセス許可も必要です。

設定の自動化

Flow Framework は、クエリ生成や会話チャットなどのユースケースの OpenSearch 設定を自動化できる OpenSearch プラグインです。プラグインは自然言語クエリ生成機能を有効にするリソースを追跡するため、フローフレームワークインデックスはクエリアシストを使用する各ドメインのテンプレートを保存します。

Flow Framework では、事前定義されたテンプレートのセットから選択するか、生成モデルのバックエンドとして OpenSearch を準備する機械学習コネクタ、ツール、エージェント、およびその他のコンポーネント用の独自のオートメーションを作成できます。

ベクトル検索

Amazon OpenSearch Service のベクトル検索を使用すると、従来のキーワードマッチングではなく機械学習埋め込みを使用して意味的に類似したコンテンツを検索できます。ベクトル検索は、データ (テキスト、画像、オーディオなど) を、コンテンツの意味的意味をキャプチャする高次元数値ベクトル (埋め込み) に変換します。検索を実行すると、OpenSearch はクエリのベクトル表現を保存されたベクトルと比較して、最も類似した項目を見つけます。

ベクトル検索には、次の主要な要素が含まれます。

ベクトルフィールド

OpenSearch は、設定可能なディメンション (最大 16,000) を持つ高密度ベクトルを保存する `knn_vector` フィールドタイプをサポートしています。

検索方法

- k-NN (k-narest neighbors): k の最も類似したベクトルを検索します
- 近似 k-NN: HNSW (Hierarchical Navigable Small World) などのアルゴリズムを使用して、大規模なデータセットをすばやく検索します

距離メトリクス

以下を含むさまざまな類似度計算をサポートします。

- ユークリッド距離
- コサイン類似度
- ドット積

一般的なユースケース

ベクトル検索では、次の一般的なユースケースがサポートされています。

- セマンティック検索: 一致するキーワードだけでなく、同様の意味を持つドキュメントを検索する
- レコメンデーションシステム: 類似製品、コンテンツ、またはユーザーを提案する
- イメージ検索: 視覚的に似たイメージを検索する
- 異常検出: データパターンの外れ値を特定する
- RAG (Retrieval Augmented Generation: 検索拡張生成): 関連するコンテキストで LLM レスポンスを強化する

機械学習との統合

OpenSearch は、以下の機械学習サービスおよびモデルと統合されています。

- Amazon Bedrock: 基盤モデルを使用して埋め込みを生成する場合
- Amazon SageMaker AI: カスタム ML モデルデプロイ用
- Hugging Face モデル: 事前トレーニング済みの埋め込みモデル
- カスタムモデル: 独自のトレーニング済み埋め込みモデル

ベクトル検索を使用すると、コンテキストと意味を理解する高度な AI を活用したアプリケーションを構築し、従来のテキストマッチング機能をはるかに超えることができます。

Amazon S3 ベクトルから OpenSearch Serverless へのインポート

Amazon S3 Vectors は、ベクトルの保存とクエリをネイティブにサポートする史上初のクラウドオブジェクトストアを提供します。S3 Vectors は、意味的意味や類似性に基づいてクエリを実行できる、費用対効果に優れ、伸縮性自在で耐久性の高いベクトルストレージを提供します。1 秒未満のクエリ応答時間を提供し、ベクトルのアップロード、保存、クエリのコストを最大 90% 削減します。

Amazon S3 Vectors には S3 ベクトルバケットが導入されています。これを使用して、インフラストラクチャをプロビジョニングすることなくベクトルデータを保存、アクセス、クエリできます。ベクトルバケット内では、ベクトルデータをベクトルインデックス内で整理できます。ベクトルバケットは複数のベクトルインデックスを持つことができ、各ベクトルインデックスは数百万のベクトルを保持できます。詳細については、「Amazon S3 ユーザーガイド」の「Amazon S3 Vectors とベクトルバケットの操作」を参照してください。

各ベクトルは以下で構成されます:

- 一意のキー
- ベクトルデータ
- JSON 形式のオプションのメタデータ

ベクトルインデックスは、類似性検索操作のためにユークリッド距離関数とコサイン距離関数をサポートします。

> **Note**
>
> ベクターバケットの主な利点は、膨大なデータセットを極めて低コストで保存できると同時に、ベクター演算のための直接的な API アクセスを提供できることです。

Amazon S3 ベクトルバケットの作成方法などの詳細については、「Amazon S3 ユーザーガイド」の「Amazon S3 Vectors とベクトルバケットの操作」を参照してください。このトピックで説明されている以外の OpenSearch Service との統合の詳細については、「OpenSearch Service での S3 Vectors の使用」を参照してください。

Amazon OpenSearch Service で S3 Vectors を使用すると、クエリの頻度が低い場合にベクトルストレージのコストを削減し、需要の増加に応じてそれらのデータセットを OpenSearch にすばやく移動したり、検索機能を強化したりできます。

OpenSearch Service は Amazon S3 Vectors と統合され、Amazon S3 ベクトルバケットが単独で提供する以上のパフォーマンスと機能を提供します。必要に応じて、この統合を検討してください。

- より高いクエリスループット
- 1 秒未満の検索レイテンシー
- 集計などの高度な分析機能
- テキストデータとベクトルデータを組み合わせたハイブリッド検索

この統合は、複数のアプリケーションが異なるパフォーマンス要件で同じベクトルデータを使用する場合に特に有用です。一部のアプリケーションはコストの影響を受けやすいユースケースのために Amazon S3 ベクトルバケットと直接やり取りでき、他のアプリケーションはパフォーマンスが重要なオペレーションのために OpenSearch 統合を活用できます。

統合アーキテクチャ

この統合では、Amazon OpenSearchAmazon S3 ベクトルインデックスと Amazon OpenSearch Serverless ベクトルコレクション間のデータパイプラインとして Amazon OpenSearch Ingestion (OSI) を使用します。OpenSearch Ingestion は、指定したベクトルインデックスからベクトルデータを自動的にエクスポートし、OpenSearch Serverless ベクトルコレクションに取り込んで、高性能な検索操作を行います。

> **Note**
>
> エクスポート後も、データは S3 ベクトルインデックスに残ります。データのコピーは 2 つあります。

各ベクトルインデックスは、OpenSearch Serverless コレクション内の対応するインデックスにマッピングされます。統合:

- ベクトルディメンションを保持します
- メタデータを保持します
- OpenSearch のベクトル検索機能のデータ構造を最適化します

設定後、OpenSearch Ingestion は Amazon S3 ListVectors API を使用して、指定されたベクトルインデックスからベクトルを消費することでデータエクスポートプロセスを開始します。このサービスはベクトルを並行して処理し、OpenSearch Ingestion と Amazon OpenSearch Serverless の両方のスケーリング制限を尊重しながら取り込み速度を最適化します。

取り込み中、サービスは以下を行います。

- OpenSearch Service の想定形式と一致するようにベクトルデータを変換します
- ベクトル値、メタデータ、距離メトリクスなどの重要な情報を保持します
- インテリジェントな再試行メカニズムを通じて障害シナリオを処理します
- 後で分析するためのデッドレターキューとして使用される Amazon S3 バケットに問題のあるレコードを配置します

統合は、ベクトルディメンション、データセットサイズ、および設定されたスケーリング制限に応じてパフォーマンスで、大規模なデータセットを効率的に処理します。OSI はパイプラインあたり最大 16 人のワーカーをスケールアップでき、OpenSearch Serverless は取り込みの需要に基づいて容量を自動的に調整します。デフォルトでは、OpenSearch は `maxSearch` OpenSearch Serverless 側の OpenSearch Computational Unit (OCU) を 100 に増やします。

> **Note**
>
> 統合では、以下を通じてコスト効率が優先されます。
>
> - エクスポート完了後の自動パイプラインシャットダウン

- OpenSearch Serverless コレクションのスケーリング
- 従量課金方式のリソースモデル

必要な IAM 許可

統合には、サービス間の安全な通信を可能にするための IAM アクセス許可の慎重な設定が必要です。OpenSearch Ingestion には、Amazon S3 ベクトルインデックスからの読み取り、OpenSearch Service ベクトルコレクションへの書き込み、および関連するセキュリティポリシーの管理を行うためのアクセス許可が必要です。

このトピックの後半の手順を使用して統合を有効にすると、アクセス許可管理に次のいずれかのオプションを選択できます。

- 必要なアクセス許可を持つサービスロールを自動的に作成することをシステムに許可する
- 要件を満たす既存のロールを提供します

自動的に作成されたロールには、以下のポリシーが含まれます:

- Amazon S3 ベクトルインデックス API へのアクセス
- OpenSearch Service コレクションの操作の管理
- 失敗した取り込み試行のデッドレターキューオペレーションの処理

既存のロールを指定する場合は、ロールに次の IAM アクセス許可があることを確認します。

(必須): OpenSearch Ingestion と OpenSearch Serverless 間のデータパイプライン権限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowAPIs",
            "Effect": "Allow",
            "Action": [ "aoss:APIAccessAll", "aoss:BatchGetCollection" ],
            "Resource": [ "arn:aws:aoss:*:111122223333:collection/collection-
id" ]
```

```
        },
        {
            "Sid": "allowSecurityPolicy",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateSecurityPolicy",
                "aoss:UpdateSecurityPolicy",
                "aoss:GetSecurityPolicy"
            ],
            "Resource": "*",
            "Condition":{
               "StringLike":{
                  "aoss:collection": [ "collection-name" ]
                },
               "StringEquals": {
                  "aws:ResourceAccount": [ "111122223333" ]
               }
            }
        }
    ]
}
```

(必須): OpenSearch Ingestion と Amazon S3 デッドレターキュー間のデータインジェスト権限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "s3Access",
            "Effect": "Allow",
            "Action": [
              "s3:PutObject"
            ],
            "Resource": [ "arn:aws:s3:::bucket/*" ]
        }
    ]
}
```

(必須): OpenSearch Ingestion と Amazon S3 Vectors 間のデータインジェスト権限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3VectorIndexAccess",
            "Effect": "Allow",
            "Action": [
               "s3vectors:ListVectors",
               "s3vectors:GetVectors"
            ],
            "Resource": [
                "arn:aws:s3vectors:us-east-1:111122223333:bucket/bucket-name/
index/index-name"
            ]
        }
    ]
}
```

(AWS KMS 暗号化が有効になっている場合に必須): OpenSearch Ingestion と Amazon S3 Vectors 間の通信のための復号アクセス許可

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowS3VectorDecryptionOfCustomManagedKey",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [ "arn:aws:kms:us-east-1:111122223333:key/key-id" ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3vectors.us-east-1.amazonaws.com",
```

```
                    "kms:EncryptionContext:aws:s3vectors:arn": [
                        "arn:aws:s3vectors:us-east-1:111122223333:bucket/example-
bucket",
                        "arn:aws:s3vectors:us-east-1:111122223333:bucket/example-
bucket/index/example-index"
                        ]
                }
            }
        }
    ]
}
```

Amazon S3 Vectors と OpenSearch の統合を設定する

次の手順を使用して、Amazon S3 Vectors と OpenSearch Serverless の統合を設定します。

> **Note**
>
> [ベクトルバケット] ページで [OpenSearch へエクスポート] オプションを選択して Amazon S3 コンソールから統合を設定するプロセスを開始した場合、手順に記載されているように、次の手順の一部のステップは適用されません。

Amazon S3 Vectors と OpenSearch Serverless の統合を設定するには

1. Amazon OpenSearch Service コンソールで、[S3 ベクトルインデックスを OpenSearch ベクトルエンジンにインポート] ページを開きます。Amazon S3 コンソールで [OpenSearch へエクスポート] をクリックすると、ページが自動的に表示されます。OpenSearch コンソールで開始する場合は、左側のナビゲーションで [統合] を選択し、[S3 ベクトルインデックスのインポート] を選択します。
2. [ソース] セクションで、Amazon S3 コンソールでを開始した場合は、ベクトルインデックスの名前とその Amazon リソースネーム (ARN) が既に指定されていることを確認します。OpenSearch コンソールで開始した場合は、[S3 ベクトルインデックス ARN] フィールドにインデックス ARN を入力します。
3. [サービスアクセス] セクションでオプションを選択します。既存のロールを選択する場合は、「必要な IAM 許可」の説明に従って、統合に必要なすべてのアクセス許可があることを確認します。

4. (オプション) 追加設定 を展開します。[冗長性を有効にする (アクティブレプリカ)] には、本番環境ではこのオプションを選択したままにすることを推奨します。最初のコレクションを作成すると、OpenSearch Serverless は 2 OCU (1 OCU はインデックス作成用、もう 1 OCU は検索用) をインスタンス化します。高可用性を確保するために、他のアベイラビリティーゾーンでスタンバイノードのセットも起動します。開発とテストの目的で、コレクションの [冗長性を有効化] 設定を無効にすることができます。これにより、2 つのスタンバイレプリカが排除され、2 つの OCU のみがインスタンス化されます。デフォルトでは、冗長アクティブレプリカが有効になっています。これは、アカウントの最初のコレクションのために合計で 4 つの OCU がインスタンス化されることを意味します。

   Amazon OpenSearch Serverless ベクトルのカスタマーマネージド AWS KMS キーを追加するで、カスタマーマネージドキーを使用してベクトルコレクションのデータを暗号化するには、このオプションを選択します。デフォルトでは、OpenSearch は AWS マネージドキーを使用します。

5. Amazon S3 コンソールで [OpenSearch へエクスポート] オプションをクリックしてこのプロセスを開始した場合、[エクスポートの詳細] セクションに、OpenSearch が次に実行するステップが一覧表示されます。準備ができたら、[エクスポート] を選択します。

   OpenSearch Service コンソールでこのプロセスを開始した場合、[インポートの詳細] セクションに、OpenSearch が次に実行するステップが一覧表示されます。準備が完了したら、[インポート] を選択します。

   OpenSearch は履歴ページを開き、OpenSearch Serverless インデックスへの Amazon S3 ベクトルインデックスのすべてのエクスポート/インポートを表示します。

取り込みが正常に完了すると、OSI は不要なコストを回避するためにパイプラインを自動的に停止します。取り込まれたデータは OpenSearch に保持されます。CloudWatch メトリクスを使用して統合の進行状況をモニタリングし、トラブルシューティングのために詳細ログにアクセスできます。

OpenSearch コレクションはアクティブのままで、最初の取り込みが完了した後もクエリで使用できます。以下を実行できます:

- 類似度検索
- 集計
- 分析運用

Amazon S3 ベクトルエンジンを使用した高度な検索機能

Amazon OpenSearch Service では、ベクトルインデックスのベクトルエンジンとして Amazon S3 を使用できるようになりました。この機能を使用すると、ベクトルデータを Amazon S3 にオフロードしつつ、1 秒未満のベクトル検索性能を低コストで維持できます。

この機能を使用すると、OpenSearch は、OpenSearch クラスターのストレージに他のドキュメントフィールドを保持しながら、Amazon S3 OpenSearch ベクトルインデックスにベクトル埋め込みを保存します。このアーキテクチャには次の利点があります:

- 耐久性: S3 Vectors に書き込まれたデータは Amazon S3 に保存され、S3 はイレブンナインのデータ耐久性を実現するよう設計されています。
- スケーラビリティ: クラスターストレージを消費することなく、大きなベクトルデータセットを S3 にオフロードします。
- コスト効率: ベクトルが多いワークロードのストレージコストを最適化します。

OpenSearch には、S3 ベクトルインデックスを使用するための以下の要件があります:

- OpenSearch バージョン 2.19 以降
- OpenSearch Optimized インスタンス
- OpenSearch リリースの最新のパッチバージョン

S3 ベクトルの有効化

新しいドメインを作成するとき、または既存のドメインを更新するときは、[詳細設定] セクションで [エンジンオプションとして S3 ベクトルを有効にする] を選択できます。この設定により、エンジンとして S3 Vectors を利用するときに、OpenSearch が S3 ベクトルバケットを作成できるようになります。このオプションを有効にすると、OpenSearch はドメインの S3 ベクトルを次のように設定します。

1. ドメインで設定された AWS KMS キーに 2 つの新しい許可を作成します。
   - S3 Vectors のバックグラウンドインデックス作成ジョブに対する復号権限付きのグラント
   - OpenSearch が `GenerateDataKey` アクセス許可を持つ S3 ベクトルバケットを作成するためのグラント
2. OpenSearch ドメインで使用される KMS キーを、すべてのベクトルインデックスデータの保管時の暗号化用の CMK として設定します。

S3 ベクトルエンジンを使用したインデックスの作成

ドメインを設定したら、s3vector をインデックスマッピングのバックエンドベクトルエンジンとして使用するフィールドを使用して、1 つ以上の k-NN インデックスを作成できます。ユースケースに基づいて、異なるエンジンタイプで異なるベクトルフィールドを設定できます。

> **Important**
>
> s3vector エンジンは、インデックス作成時のフィールド定義のマッピングにのみ使用できます。インデックスの作成後に s3vector エンジンでマッピングを追加または更新することはできません。

以下は、S3 ベクトルエンジンインデックスを作成する例です。

例: S3 ベクトルエンジンを使用した k-NN インデックスの作成

```
PUT my-first-s3vector-index
{
  "settings": {
    "index": {
      "knn": true
    }
  },
  "mappings": {
    "properties": {
        "my_vector_1": {
          "type": "knn_vector",
          "dimension": 2,
          "space_type": "l2",
          "method": {
            "engine": "s3vector"
          }
        },
        "price": {
          "type": "float"
        }
    }
  }
}
```

例: S3 ベクトルエンジンと FAISS エンジンの両方による k-NN インデックスの作成

この例では、同じインデックス内で複数のベクトルエンジンを使用できるという事実を強調しています。

```
PUT my-vector-index
{
  "settings": {
    "index": {
      "knn": true
    }
  },
  "mappings": {
    "properties": {
        "my_vector_1": {
          "type": "knn_vector",
          "dimension": 2,
          "space_type": "l2",
          "method": {
            "engine": "s3vector"
          }
        },
        "price": {
          "type": "float"
        },
        "my_vector_2": {
            "type": "knn_vector",
            "dimension": 2,
            "space_type": "cosine",
            "method": {
                "name": "hnsw",
                "engine": "faiss",
                "parameters": {
                    "ef_construction": 128,
                    "m": 24
                }
            }
        }
    }
  }
}
```

サポートされていない例: インデックスの作成後の S3 ベクトルエンジンの追加

次のアプローチはサポートされておらず、失敗します。

```
PUT my-first-s3vector-index
{
  "settings": {
    "index": {
      "knn": true
    }
  }
}

PUT my-first-s3vector-index/_mapping
{
  "properties": {
        "my_vector_1": {
          "type": "knn_vector",
          "dimension": 2,
          "space_type": "l2",
          "method": {
            "engine": "s3vector"
          }
        },
        "price": {
          "type": "float"
        }
    }
}
```

機能的制限

インデックスで s3vector エンジンを使用する前に、次の制限を考慮してください。

s3vector エンジンでサポートされていない機能と動作

機能	行動
Split/Shrink/Clone インデックス	これらの API は、knn_vector フィールドの s3vector エンジンで設定されたインデックスで使用すると失敗します。
スナップショット	s3vector エンジンを使用するインデックスはスナップショットをサポートしていません。マネージドドメインの場合: • 自動スナップショットには、s3vector エンジンを使用していないインデックスのみが含まれます。

機能	行動
	• s3vector インデックスの手動スナップショットリクエストは失敗します。 Note スナップショットは point-in-time リカバリではサポートされていませんが、s3vector エンジンは OpenSearch Optimized インスタンスとともにイレブンナインの耐久性を提供します。
UltraWarm 階層	s3vector エンジンで設定されたインデックスは UltraWarm 階層に移行できません。
クラスター間レプリケーション	s3vector エンジンで設定されたインデックスは、クラスター間のレプリケーションをサポートしていません。
誤削除防止	s3vector エンジンを使用するインデックスではスナップショットがサポートされていないため、誤削除防止は使用できません。ドメイン内の他のインデックスは復元できます。
放射状検索	s3vector エンジンを使用するフィールドでは、放射状検索によるクエリはサポートされていません。

ドキュメントのインデックス作成

S3 ベクトルエンジンでインデックスを作成したら、標準 _bulk API を使用してドキュメントを取り込むことができます。OpenSearch は、s3vector エンジンを使用して knn_vector フィールドのベクトルデータをリアルタイムで S3 ベクトルインデックスに自動的にオフロードします。他のフィールドに属するデータ、または異なるエンジンを使用する knn_vector フィールドのデータは、OpenSearch によって独自のストレージ層に保持されます。

確認済みのすべての一括リクエストについて、OpenSearch はすべてのデータ (ベクトルと非ベクトル) の耐久性を保証します。リクエストが否定的な確認を受け取った場合、その一括リクエストのドキュメントの耐久性は保証されません。このようなまれなケースでは、ドキュメント ID を使用し

て、以前に失敗したリクエストを削除した後、このようなリクエストを再試行することをお勧めします。

一括インデックス作成の例

```
POST _bulk
{ "index": { "_index": "my-first-s3vector-index", "_id": "1" } }
{ "my_vector_1": [1.5, 2.5], "price": 12.2 }
{ "index": { "_index": "my-first-s3vector-index", "_id": "2" } }
{ "my_vector_1": [2.5, 3.5], "price": 7.1 }
{ "index": { "_index": "my-first-s3vector-index", "_id": "3" } }
{ "my_vector_1": [3.5, 4.5], "price": 12.9 }
{ "index": { "_index": "my-first-s3vector-index", "_id": "4" } }
{ "my_vector_1": [5.5, 6.5], "price": 1.2 }
{ "index": { "_index": "my-first-s3vector-index", "_id": "5" } }
{ "my_vector_1": [4.5, 5.5], "price": 3.7 }
```

ドキュメントの検索

標準 _search API を使用してインデックスを検索し、テキスト、k-NN、またはハイブリッドクエリを実行できます。s3vector エンジンで設定された knn_vector フィールドに対するクエリの場合、OpenSearch はクエリを対応する S3 ベクトルインデックスに自動的にオフロードします。

> **Note**
>
> s3vector エンジンでは、k-NN 検索クエリは最大値kの 100 をサポートします。つまり、100 個の最も近い近傍の最大値を検索結果に返すことができます。

検索クエリの例

```
GET my-first-s3vector-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector_1": {
        "vector": [2.5, 3.5],
        "k": 2
      }
```

```
    }
  }
}
```

s3vector エンジンを使用して、OpenSearch kNN インデックスでフィルタリングされたベクトル検索を実行できます。OpenSearch はフィルターをポストフィルターとして適用し、特定のヒューリスティックに基づいてオーバーサンプリングメカニズムを使用してリコールとレイテンシーのバランスを取ります。

フィルターを使用した検索クエリの例:

```
GET my-index/_search
{
  "size": 10,
  "query": {
    "knn": {
      "my_vector_field": {
        "vector": [2.5, 3.5, 1.2, 4.8],
        "k": 10,
        "filter": {
          "range": {
            "price": {
              "gte": 10,
              "lte": 100
            }
          }
        }
      }
    }
  }
}
```

サポートされているマッピングパラメータ

s3vector エンジンでは、knn_vector フィールドはマッピングで次のパラメータをサポートします。

ベクトルフィールドパラメータ

[Parameter] (パラメータ)	[Required] (必須)	説明	サポートされる値
type	はい	ドキュメント内に存在するフィールドのタイプ。	knn_vector
dimension	はい	インデックスに取り込まれる各ベクトルのディメンション。	>0, <=4096
space_type	いいえ	ベクトル間の距離を計算するために使用されるベクトル空間。	l2, cosinesimil
method.engine	はい	インデックス作成と検索に使用する近似 k-NN エンジン。	s3vector
method.name	いいえ	最近傍法	""
store	該当なし	このマッピングパラメータの有効化または無効化は、knn_vector データが OpenSearch に保存されないため、no-op です。	サポート外
doc_values	該当なし	このマッピングパラメータの有効化または無効化は、knn_vector データが OpenSearch に保存されないため、no-op です。	サポート外

> **Important**
>
> ネスト化された knn_vector フィールドタイプは s3vector エンジンを使用してサポートされていません

使用量測定と請求

この機能の計測と請求については、「Amazon OpenSearch Service の料金」を参照してください。

s3vector エンジンの無効化

s3vector エンジンを無効にする前に、現在使用しているすべてのインデックスを削除します。そうしないと、エンジンを無効にする試みは失敗します。

また、s3vector エンジンを有効または無効にすると、ドメインでブルー/グリーンデプロイがトリガーされることに注意してください。

s3vector エンジンを無効にするには、ドメイン設定を編集し、S3VectorsEngine.Enabled: false を設定します。

Amazon OpenSearch Service での K 最近傍 (k-NN) 検索

関連する K 最近傍アルゴリズムでは、Amazon OpenSearch Service 用 k-NN を使用すると、ベクトル空間内のポイントを検索し、ユークリッド距離またはコサイン類似度によってそれらのポイントの「最近傍」を見つけることができます。ユースケースには、推奨 (音楽アプリケーションの「おすすめの曲」機能など)、画像認識、不正行為の検出などがあります。

> **Note**
>
> このドキュメントでは、k-NN プラグインの概要と、マネージド OpenSearch Service でプラグインを使用する場合の制限について説明します。簡易および複雑な例、パラメータリファレンス、完全な API リファレンスなど、k-NN プラグインの包括的なドキュメントについては、オープンソースの「OpenSearch ドキュメント」を参照してください。このドキュメントでは、パフォーマンス調整と k-NN 固有のクラスター設定についても説明されています。

k-NN を用いた開始方法

k-NN を使用するには、index.knn 設定でインデックスを作成し、knn_vector データ型の 1 つ以上のフィールドを追加する必要があります。

```
PUT my-index

{
  "settings": {
```

```
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "my_vector1": {
        "type": "knn_vector",
        "dimension": 2
      },
      "my_vector2": {
        "type": "knn_vector",
        "dimension": 4
      }
    }
  }
}
```

knn_vector データ型は、必要な dimension パラメータによって定義された浮動小数点数とともに、10,000 個までの浮動小数点数の単一のリストをサポートします。インデックスを作成したら、そのインデックスにデータを追加します。

```
POST _bulk

{ "index": { "_index": "my-index", "_id": "1" } }
{ "my_vector1": [1.5, 2.5], "price": 12.2 }
{ "index": { "_index": "my-index", "_id": "2" } }
{ "my_vector1": [2.5, 3.5], "price": 7.1 }
{ "index": { "_index": "my-index", "_id": "3" } }
{ "my_vector1": [3.5, 4.5], "price": 12.9 }
{ "index": { "_index": "my-index", "_id": "4" } }
{ "my_vector1": [5.5, 6.5], "price": 1.2 }
{ "index": { "_index": "my-index", "_id": "5" } }
{ "my_vector1": [4.5, 5.5], "price": 3.7 }
{ "index": { "_index": "my-index", "_id": "6" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 10.3 }
{ "index": { "_index": "my-index", "_id": "7" } }
{ "my_vector2": [2.5, 3.5, 5.6, 6.7], "price": 5.5 }
{ "index": { "_index": "my-index", "_id": "8" } }
{ "my_vector2": [4.5, 5.5, 6.7, 3.7], "price": 4.4 }
{ "index": { "_index": "my-index", "_id": "9" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 8.9 }
```

次に、knn クエリタイプを使用してデータを検索できます。

```
GET my-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  }
}
```

この場合、k はクエリが返す最近傍の数ですが、size オプションも含める必要があります。それ以外の場合は、クエリ全体の k 結果ではなく、各シャード (および各セグメント) の k 結果が返されます。k-NN は、最大 10,000 の k 値をサポートします。

knn クエリを他の句と混在させると、返される結果は k の結果よりも少なくなる場合があります。この例では、post_filter 句を使用することにより、結果の数が 2 から 1 に減ります。

```
GET my-index/_search

{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  },
  "post_filter": {
    "range": {
      "price": {
        "gte": 6,
        "lte": 10
      }
    }
  }
}
```

最適なパフォーマンスを維持しながら大量のクエリを処理する必要がある場合は、_msearch API を使用して JSON を使用して一括検索を作成し、1 つのリクエストを送信して複数の検索を実行できます。

```
GET _msearch

{ "index": "my-index"}
{ "query": { "knn": {"my_vector2":{"vector": [2, 3, 5, 6],"k":2 }} } }
{ "index": "my-index", "search_type": "dfs_query_then_fetch"}
{ "query": { "knn": {"my_vector1":{"vector": [2, 3],"k":2 }} } }
```

次の動画は、K-NN クエリの一括ベクトル検索を設定する方法を示しています。

k-NN の違い、チューニング、および制限

OpenSearch では、_cluster/settings APIを使用してすべての k-NN の設定を変更できます。OpenSearch Service では、knn.memory.circuit_breaker.enabled および knn.circuit_breaker.triggered 以外のすべての設定を変更できます。k-NN 統計は、Amazon CloudWatch メトリクスとして含まれています。

特に、knn.memory.circuit_breaker.limit 統計情報およびインスタンスタイプに使用可能な RAM に対して、各データノードの KNNGraphMemoryUsage メトリクスを確認できます。OpenSearch Service は、インスタンスの RAM の半分を Java ヒープ (32 GiB のヒープサイズまで) に使用します。デフォルトでは、k-NN は残りの半分の 50% まで使用するため、32 GiB の RAM を持つインスタンスタイプは 8 GiB のグラフに対応できます (32 * 0.5 * 0.5)。グラフのメモリ使用量がこの値を超えると、パフォーマンスが低下する可能性があります。

バージョン 2.x 以降で作成された k-NN インデックスを、バージョン 2.17 以降のドメインの UltraWarm または コールドストレージ に移行できます。

k-NN インデックスのクリアキャッシュ API とウォームアップ API は、ウォームインデックスに対してブロックされます。インデックスの最初のクエリが開始されると、Amazon S3 からグラフファイルをダウンロードし、グラフをメモリにロードします。同様に、グラフの TTL の有効期限が切れると、ファイルは自動的にメモリから削除されます。

ベクトル取り込み

ベクトル取り込みは、OpenSearch ドメインと OpenSearch Serverless コレクションの迅速な取り込みとインデックス作成に役立ちます。このサービスはドメインまたはコレクションを調べ、ユー

ザーに代わって取り込みパイプラインを作成して OpenSearch にデータをロードします。ドメインまたはコレクションの取り込みとインデックス作成は、ベクトル取り込みによって管理されます。

および 自動最適化機能を有効にすることで、インデックス作成プロセスを高速化ベクトルインデックス作成の GPU アクセラレーションおよび最適化できます。ベクトル取り込みでは、ベクトルデータベースのインデックス作成と取り込みをサポートするために、基盤となるインフラストラクチャ、パッチソフトウェア、スケーリングクラスターを管理する必要はありません。これにより、ニーズに合わせてベクトルデータベースをすばやく構築できます。

仕組み

ベクトル取り込みは、ドメインまたはコレクションとそのインデックスを調べます。ベクトルインデックスフィールドを手動で設定することも、OpenSearch が自動設定を使用できるようにすることもできます。

ベクトル取り込みは、Amazon S3 と OpenSearch 間のデータパイプラインとして OpenSearch Ingestion (OSI) を使用します。このサービスはベクトルを並行して処理し、OSI と OpenSearch の両方のスケーリング制限を尊重しながら取り込み速度を最適化します。

OpenSearch Vector 取り込み料金

特定の時点では、パイプラインを流れるデータがあるかどうかにかかわらず、パイプラインに割り当てられたベクトル取り込み OCUs の数に対してのみ料金が発生します。OpenSearch ベクトル取り込みは、使用量に基づいてパイプライン容量をスケールアップまたはスケールダウンすることで、ワークロードに即座に対応します。

料金の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

前提条件

ベクトル取り込みを使用する前に、次のリソースがあることを確認してください。

- OpenSearch JSON ドキュメントを Parquet または JSONL 形式で含む Amazon S3 バケット
- OpenSearch リソース - ドメインまたはコレクション
- OpenSearch バージョン 2.19以降 (自動最適化統合に必要)

ベクトルデータベースを作成する

ベクトル取り込みジョブ作成ワークフローを使用して、自動ベクトルインデックス調整を設定し、大規模なインデックス構築を高速化します。

> **Note**
>
> このセクションの手順の内容は、ユーザーインターフェイスが確定されると変更される可能性があります。ワークフローは、最新のコンソールエクスペリエンスを反映するために、今後のリリースで更新される可能性があります。

ベクトル取り込みジョブを作成するには

1. 「ベクトル取り込みジョブの詳細」セクションの「名前」に、取り込みジョブの名前を入力します。
2. データソースセクションで、以下を設定します。
   a. Amazon S3 URI の場合は、OpenSearch Service JSON ドキュメントを含む Amazon S3 バケットの場所を入力します。
   b. Amazon S3 を参照を選択して使用可能なバケットから選択するか、表示を選択してバケットコンテンツをプレビューします。
   c. コンテンツタイプで、以下を選択します。
      - ベクトル - ドキュメントには既にベクトルが含まれており、それ以上のベクトル埋め込みの生成は必要ありません。
3. データソースのアクセス許可セクションで、アクセス許可を設定します。
   a. [IAM role] (IAM ロール) で、以下のいずれかのオプションを選択します。
      - 新規ロールの作成
      - 既存のロールを使用する
   b. IAM ロール名には、ロールの名前を入力します。
4. 送信先セクションで、OpenSearch Service エンドポイントを設定します。
   a. Endpoint で、現在のリージョンの互換性のあるドメインまたはコレクションから選択するオプションを選択します。
   b. 次へ を選択して、選択したエンドポイントを続行します。

5. Next を選択して次のステップに進むか、Cancel を選択して保存せずに終了します。

関連機能

ベクトル取り込みは、次の Amazon OpenSearch Service 機能と連携してベクトルデータベースのパフォーマンスを最適化します。

ベクトルインデックス作成の GPU アクセラレーション

GPU アクセラレーションにより、ベクトルインデックスの作成、更新、削除に必要な時間が短縮されます。ベクトル取り込みで使用すると、大規模なベクトルデータベースの取り込みとインデックス作成プロセスを大幅に高速化できます。

自動最適化

自動最適化は、検索レイテンシー、品質、メモリ要件の間の最適なトレードオフを自動的に検出します。ベクトル取り込みでは、取り込みプロセス中に自動最適化レコメンデーションを適用して、ベクトルインデックスが最適に設定されるようにできます。

最良の結果を得るには、ベクトル取り込みを使用して大規模なベクトルデータベースを構築するときに、GPU 加速と自動最適化の両方を有効にすることを検討してください。

Amazon S3 ベクトルインデックスを OpenSearch Service ベクトルエンジンにエクスポートする

選択した Amazon S3 ベクトルインデックスのpoint-in-timeエクスポートを OpenSearch Service にエクスポートします。OpenSearch Service ベクトルエンジンは、高度な検索機能を備えたシンプルでスケーラブルなベクトルストアを提供します。

Amazon S3 ベクトルインデックスを OpenSearch Service ベクトルエンジンにエクスポートするには

1. Source セクションで、Amazon S3 ベクトルインデックスの詳細を確認します。
   - Amazon S3 ベクトルインデックス - ソースインデックスの名前
   - Amazon S3 ベクトルインデックス ARN - インデックスの Amazon リソースネーム
2. サービスアクセスセクションで、OpenSearch Service 認可を設定します。
   a. OpenSearch Service を承認する方法を選択する で、次のいずれかを選択します。

- 新しいサービスロールを作成して使用
- 既存のサービスロールを使用

b. サービスロール名に、サービスロールの名前を入力します。

> **Note**
>
> サービスロール名は 1～64 文字にする必要があります。有効な文字は、a～z、A～Z、0～9、ピリオド (.) です。

c. アクセス許可の詳細を表示 を選択して、必要なアクセス許可を確認します。

3. 追加設定の展開 - 必要に応じてオプションで詳細オプションを設定します。
4. エクスポートの詳細セクションで、次のオプションを設定します。

- OpenSearch Service ベクトルコレクションの作成を自動化 - OpenSearch Service コレクションはベクトルデータの保存に使用されます。サーバーレスコンピューティング容量は OpenSearch Service Compute Units (OCUs) で測定されます。デフォルトでは、最大 OCU 容量は 50 です。
- サービスアクセス用の IAM ロールの作成を自動化する - このロールはOpenSearch Service が Amazon S3 ベクトルインデックスを読み取って OpenSearch Service コレクションに書き込むために使用されます。
- OpenSearch Service 取り込みパイプラインの作成を自動化 - OpenSearch Service 取り込みパイプラインを使用してデータを取り込みます。Amazon S3 バケットは、失敗したイベントをキャプチャして Amazon S3 バケットのデッドレターキュー (DLQ) に保存するためのベストプラクティスとして作成され、トラブルシューティングと分析のための簡単なアクセスを可能にします。

5. エクスポートを選択してエクスポートプロセスを開始するか、キャンセルを選択してエクスポートせずに終了します。

Amazon S3 ベクトル名前空間を OpenSearch Service ベクトルエンジンにインポートする

OpenSearch Service を使用してベクトルデータを分析するには、1 回限りの OpenSearch Service コレクションと IAM アクセス許可の設定が必要です。

Amazon S3 ベクトル名前空間を OpenSearch Service ベクトルエンジンにインポートするには

1. ソースセクションで、Amazon S3 ベクトルインデックスを設定します。
   - Amazon S3 ベクトルインデックス ARN には、Amazon S3 ベクトルインデックスの ARN を入力します。

     > **Note**
     > arn:aws:iam::account-id:vector-bucket-name/*:index の形式である必要があります

2. サービスアクセスセクションで、OpenSearch Service 認可を設定します。
   a. OpenSearch Service を承認する方法を選択する で、次のいずれかを選択します。
      - 新しいサービスロールを作成して使用
      - 既存のサービスロールを使用
   b. サービスロール名に、サービスロールの名前を入力します。

      > **Note**
      > サービスロール名は 1～64 文字にする必要があります。有効な文字は、a～z、A～Z、0～9、ピリオド (.) です。

   c. アクセス許可の詳細を表示 を選択して、必要なアクセス許可を確認します。
3. 追加設定の展開 - 必要に応じてオプションで詳細オプションを設定します。
4. インポートステップセクションで、次の自動化オプションを設定します。
   - OpenSearch Service ベクトルコレクションの作成を自動化 - OpenSearch Service コレクションはベクトルデータの保存に使用されます。サーバーレスコンピューティング容量は OpenSearch Service Compute Units (OCUs) で測定され、デフォルトでは最大 OCU 容量は 50 です。
   - サービスアクセス用の IAM ロールの作成を自動化する - このロールはOpenSearch Service が Amazon S3 ベクトルインデックスを読み取って OpenSearch Service コレクションに書き込むために使用されます。
   - OpenSearch Service 取り込みパイプラインの作成を自動化する - OpenSearch Service 取り込みパイプラインを使用してデータを取り込みます。Amazon S3 バケットは、失敗したイベントをキャプチャして Amazon S3 バケットのデッドレターキュー (DLQ) に保存するためのベス

トプラクティスとして作成され、トラブルシューティングと分析のための簡単なアクセスを可能にします。

5. インポートを選択してインポートプロセスを開始するか、キャンセルを選択してインポートせずに終了します。

ベクトル取り込みジョブとインポート履歴を表示する

ベクトル取り込みジョブは、データセットをベクトル化し、ベクトルインデックスのチューニングを自動化して大規模なインデックス構築を加速するためのパイプラインを作成します。

ベクトル取り込みジョブを表示するには

1. 「ベクトル取り込みジョブ」セクションで、概要情報を表示します。
   - ジョブ - 取り込みジョブの合計数
   - ベクトルデータベースの作成を選択して新しい取り込みジョブを作成する
2. Amazon S3 ベクトルのインポートセクションで、インポートの概要を表示します。
   - インポートの合計 - 完了したインポートの数
   - Amazon S3 ベクトルのインポートを選択して新しいインポートを開始する
3. ベクトル取り込みジョブテーブルで、次の情報を使用してアクティブなジョブをモニタリングします。
   - 名前 - ジョブ名
   - ステータス - 現在のジョブステータス (アクティブなど)
   - データソース - ソースの場所 (s3://location など)
   - 送信先 - ターゲット送信先
   - 最終更新日 - 最新の更新タイムスタンプ
4. 検索ボックスを使用してベクトル取り込みジョブを検索し、特定のジョブを見つけます。
5. ジョブを管理するには、次のアクションから選択します。
   - 削除を選択して選択したジョブを削除する
   - ベクトルデータベースの作成を選択して追加のジョブを作成する
6. Amazon S3 ベクトルのインポート履歴セクションで、インポートイベントを追跡します。
   a. 日付範囲フィルターを使用して、インポート履歴の期間を指定します。

b. Status ドロップダウンを使用して、インポートステータス (任意のステータスなど) でフィルタリングします。

c. 検索ボックスを使用して Amazon S3 ベクトルインデックス na... によるインポートを検索し、特定のインポートを見つけます。

d. 次のようなインポートの詳細を表示します。

- インポート開始日 (UTC+5:30) - インポート開始日時
- インポートステータス - 現在のステータス (進行中、完了、失敗、一部完了)
- Amazon S3 ベクトルインデックス ARN - ソースインデックス識別子
- OpenSearch Service ベクトルコレクション - 送信先コレクション

7. Amazon S3 ベクトルのインポートを選択して、新しいインポートプロセスを開始します。

自動最適化

Auto-optimize は、ベクトルインデックスの最適化を自動化するサービスであり、数週間の手動エキスパート調整を必要とせずに、ユーザーが検索品質、速度、コストのバランスを取ることができます。ユーザー定義のレイテンシーとリコールの要件に基づいてインデックス設定を評価し、最適化の推奨事項を生成するため、最小限の専門知識が必要です。レコメンデーションは通常 30～60 分以内に配信されます。

従来のベクトルインデックス設定では、最適なパフォーマンスを実現するには、かなりの専門知識と実験が必要です。`ef_construction` (インデックスのビルド品質を制御する)`m`、 (グラフ接続の数を決定する）、 `ef_search` (HNSW 検索を制御する）、量子化メソッド (バイナリ量子化 (32x、16x、8x)、スカラー量子化 (4x)) などのパラメータは、検索精度とリソース使用率の両方に大きな影響を与えます。自動最適化は、ハイパーパラメータ最適化アルゴリズムを使用して、定義されたレイテンシーおよびリコール要件内でデータセットに一意に最適なインデックス設定を検出します。

利点

OpenSearch の自動最適化には、次の利点があります。

- 自動パラメータ調整 - アルゴリズム (HNSW)、量子化、再スコアリング、エンジンパラメータによる手動実験を排除し、時間を節約し、ベクトル検索の最適化のための学習曲線を減らします。

- 検索速度の最適化 - デフォルトでは、OpenSearch はインメモリパフォーマンス用に設定されています。自動最適化は、許容可能な検索速度を維持しながら、検索品質とコスト削減を改善する有利なトレードオフを検出します。
- コスト最適化 - 検索品質を最小限に抑え、トレードオフを高速化しながら、インデックスメモリ要件を減らすオプションを見つけることでコストを削減します。
- 検索品質の最適化 - デフォルト設定よりも高いリコールを実現したり、リコール損失を最小限に抑えながら大幅なコスト削減を実現する有利なトレードオフを発見したりできます。

自動最適化は、 などの他の OpenSearch 機能と連携してベクトルインデックス作成の GPU アクセラレーション、ベクトル検索ワークロードの包括的なパフォーマンス最適化を提供します。

仕組み

自動最適化は、ベクトルデータを分析し、最適化の推奨事項を提供するジョブベースのアーキテクチャを介して動作します。キーポイント:

- ユーザーは、Amazon S3 バケットで Parquet または JSONL 形式でデータセットを共有します。
- 許容できるリコールとレイテンシーのしきい値を設定することで、サーバーレスの自動最適化ジョブを設定します。しきい値を緩和することで、サービスがより重要なコスト最適化を検出できます。
- 自動最適化ジョブは、Amazon OpenSearch Service によってフルマネージドされたインフラストラクチャで実行されます。ジョブはドメインまたはコレクション上のリソースを消費しません。ワーカーは並行して実行してインデックス設定を評価し、大規模なデータセットでサンプリングを使用して、通常 30～60 分以内に結果を提供します。
- 各ジョブは、予測可能な定額料金で請求されます。料金の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

前提条件

- データセットの形式とアクセス許可 - データセットを Amazon S3 バケットフォルダ内の 1 つ以上の Parquet ファイルまたは JSONL ファイルとして利用できる必要があります。例えば、次のようになります。
  - Parquet: `s3://dataset-bucket-us-east-1/dataset_folder/first_half.parquet`および `s3://dataset-bucket-us-east-1/dataset_folder/second_half.parquet`
  - JSONL: `s3://dataset-bucket-us-east-1/dataset_folder/data.jsonl`

囲みフォルダ URI を指定します (例: `s3://dataset-bucket-us-east-1/dataset_folder/`)。フォルダには 1 つの形式のファイルが含まれている必要があります。同じフォルダに Parquet ファイルと JSONL ファイルを混在させないでください。このデータセットは、レコメンデーションを生成するために使用されます。フェデレーティッドロールに、そのリソースに対する次の Amazon S3 アクセス許可があることを確認します: `"s3:Get*", "s3:List*", "s3:Describe*"`。

- 正しいデータセットメタデータを指定する - 指定されたデータセットには浮動小数点値の行が含まれている必要があります。各列の名前と各ベクトルの次元は、コンソールで提供されるオプションと一致する必要があります。たとえば、データセットに各768ディメンションの名前が`train_data`付けられたベクトルが含まれている場合、これらの値は自動最適化コンソールと一致する必要があります。
- (ベクトル取り込み機能を使用する場合) - 取り込み機能を利用する場合 (インデックスを自動的に作成してデータを取り込むために自動最適化レコメンデーションを実行する場合) 、データセットを OpenSearch クラスターに取り込むための自動最適化アクセス許可を付与するように OpenSearch クラスターを設定する必要があります。ドメインアクセスポリシーを持つ OpenSearch ドメインの場合、そのポリシーを通じて新しく作成されたロールにアクセス権を付与します。きめ細かなアクセスコントロールを持つ OpenSearch ドメインの場合は、パイプラインロールをバックエンドロールとして追加します。OpenSearch Serverless コレクションの場合、パイプラインロールをデータアクセスポリシーに追加します。
- IAM アクセス許可 - 自動最適化を使用するには、次の IAM アクセス許可が必要です。
  - `opensearch:SubmitAutoOptimizeJob`
  - `opensearch:GetAutoOptimizeJob`
  - `opensearch:DeleteAutoOptimizeJob`
  - `opensearch:CancelAutoOptimizeJob`
  - `opensearch:ListAutoOptimizeJobs`

  > **Note**
  >
  > これらはアイデンティティベースのポリシーです。自動最適化は、リソースベースのポリシーをサポートしていません。

- 認証情報の有効期限 - フェデレーティッドユーザーセッションの認証情報の有効期限が少なくとも 1 時間になるように設定します。非常に大きなデータセットや高次元の場合は、有効期限を最大 3 時間延長することを検討してください。

自動最適化のユースケース

自動最適化は、以下のシナリオで特に重要です。

初期設定の最適化

ベクトル検索アプリケーションを最初にデプロイする場合、最適な HNSW パラメータを決定するには、多くの場合、広範なテストとドメインの専門知識が必要です。自動最適化は、データとワークロードの特性を分析して本番環境に対応した設定を推奨することで、このtrial-and-errorプロセスを排除します。

このユースケースは、ベクトル検索を初めて行うチームや、ベースライン設定を迅速に確立する必要がある他のベクトルデータベースプラットフォームから移行するチームに最適です。

スケーリングの最適化

ベクトルデータセットが数千から数百万のベクトルに増加するにつれて、最初にうまく機能したパラメータは最適ではない可能性があります。自動最適化では、パフォーマンスを大規模に維持するための調整が推奨されます。

コスト削減

ベクトルインデックスは、特に高次元埋め込みでは、大量のコンピューティングリソースとストレージリソースを消費する可能性があります。自動最適化は、より少ないリソースを使用しながら必要なパフォーマンスレベルを維持する、より効率的なパラメータ設定を見つけることで、コストを削減する機会を特定します。

たとえば、自動最適化では、現在のm値 (グラフの接続性) が精度要件に必要な値よりも高いことが検出され、検索品質に影響を与えることなくインデックス作成時間とストレージを削減できます。

パフォーマンスのトラブルシューティング

ベクトル検索オペレーションでクエリパフォーマンスが遅い、またはレイテンシーが高い場合、自動最適化はデータセットを分析し、より最適な設定を特定できます。このサービスは、グラフ接続や検索パラメータの調整など、パフォーマンスのボトルネックに対処するための特定の推奨事項を提供します。

制限事項

- リージョンの可用性 - 自動最適化は、次の AWS リージョンでのみ使用できます。

  - ap-south-1
  - eu-west-1
  - us-west-2
  - us-east-2
  - us–east–1
  - eu-central-1
  - ap-southeast-2
  - ap-northeast-1
  - ap-southeast-1
- コレクションタイプ - 自動最適化は、ベクトル検索コレクションと OpenSearch ドメイン (2.19、3.1、3.3) でのみサポートされています。
- エンジンのサポート

  デプロイタイプ別のエンジンサポート

エンジン	サーバーレス	OpenSearch マネージド
Lucene	いいえ	はい
信条	はい	はい
Nmslib	いいえ	いいえ

- アルゴリズムのサポート - 自動最適化は、HNSW ベースのベクトルインデックスのみをサポートします。
- 同時ジョブ - リージョンごとにアカウントごとに最大 10 個の同時最適化ジョブを実行できます。制限に達した場合、新しいジョブを受け入れることはできません。
- ジョブの期間 - 最適化ジョブは、データセットのサイズ、ディメンション、および必要なパフォーマンスメトリクスに応じて、15 分から数時間かかる場合があります。
- 推奨事項 - 自動最適化は、最大 3 つの推奨事項のみを提案します。
- データセット
  - サポートされている形式: Parquet、JSONL
  - データストア: Amazon S3

請求とコスト

自動最適化は、データセットのサイズと最適化設定に関係なく、成功した最適化ジョブごとに料金を支払うジョブごとの料金モデルを使用します。失敗したジョブやキャンセルされたジョブに対しては課金されません。さらに、自動最適化はマネージド型またはサーバーレスの OpenSearch クラスターとは異なるインフラストラクチャで実行されるため、既存のクラスターのリソース使用率には影響しません。

料金モデル

自動最適化コストは、標準の OpenSearch Serverless または OpenSearch Managed ドメインのコンピューティングおよびストレージコストとは別に請求されます。

料金の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

サポートされているデータ形式

自動最適化は、Amazon S3 に保存されているベクトルデータセットに対して次のデータ形式をサポートします。

Parquet 形式

Parquet は、分析ワークロード用に最適化された列指向ストレージ形式です。各 Parquet ファイルには、ベクトルデータを表す浮動小数点配列の列が含まれている必要があります。

Parquet ファイル構造の例 (テーブルとして表示):

```
| id  | train_data                      |
|-----|---------------------------------|
| 1   | [0.12, 0.45, 0.78, ..., 0.33] |
| 2   | [0.56, 0.89, 0.12, ..., 0.67] |
| 3   | [0.34, 0.67, 0.90, ..., 0.11] |
```

JSONL 形式

JSONL (JSON Lines) は、各行が有効な JSON オブジェクトであるテキスト形式です。各行には、ベクトルデータを表す浮動小数点配列を持つフィールドが含まれている必要があります。

JSONL ファイルの例:

```
{"id": 1, "train_data": [0.12, 0.45, 0.78, 0.33]}
```

```
{"id": 2, "train_data": [0.56, 0.89, 0.12, 0.67]}
{"id": 3, "train_data": [0.34, 0.67, 0.90, 0.11]}
```

形式間の変換

データが別の形式である場合は、次の Python スクリプトを使用してデータを変換できます。

JSON または JSONL を Parquet に変換する

```
#!/usr/bin/env python3
import json
import pyarrow as pa
import pyarrow.parquet as pq
from pathlib import Path
from typing import Any, Dict, List


def load_json_any(path: Path) -> List[Dict[str, Any]]:
    """
    Load JSON that can be:
      - a list of objects
      - a single object
      - JSON Lines (one object per line)
    Returns list[dict].
    """
    text = path.read_text().strip()

    # Try full JSON file
    try:
        obj = json.loads(text)
        if isinstance(obj, list):
            return obj
        if isinstance(obj, dict):
            return [obj]
    except json.JSONDecodeError:
        pass

    # Fallback # JSON Lines
    records = []
    for i, line in enumerate(text.splitlines(), start=1):
        line = line.strip()
        if not line:
```

```
            continue
        try:
            rec = json.loads(line)
        except json.JSONDecodeError as e:
            raise ValueError(f"Invalid JSON on line {i}: {e}")
        if not isinstance(rec, dict):
            raise ValueError(f"Line {i} must contain a JSON object")
        records.append(rec)

    return records


def json_to_parquet(json_path: str, parquet_path: str, compression: str = "snappy"):
    """Convert ANY JSON to Parquet (schema inferred)."""
    records = load_json_any(Path(json_path))
    table = pa.Table.from_pylist(records)
    pq.write_table(table, parquet_path, compression=compression)
    print(f"Wrote {len(records)} rows to {parquet_path}")


if __name__ == "__main__":
    INPUT_JSON = "vectors.jsonl"
    OUTPUT_PARQUET = "vectors.parquet"
    json_to_parquet(INPUT_JSON, OUTPUT_PARQUET)
```

Parquet を JSONL に変換する

```
#!/usr/bin/env python3
import json
import pyarrow.parquet as pq


def parquet_to_jsonl(parquet_path: str, jsonl_path: str):
    """Convert a Parquet file to JSONL format."""
    table = pq.read_table(parquet_path)
    rows = table.to_pylist()
    with open(jsonl_path, "w") as f:
        for row in rows:
            f.write(json.dumps(row) + "\n")
    print(f"Wrote {len(rows)} rows to {jsonl_path}")


if __name__ == "__main__":
```

```
    INPUT_PARQUET = "vectors.parquet"
    OUTPUT_JSONL = "vectors.jsonl"
    parquet_to_jsonl(INPUT_PARQUET, OUTPUT_JSONL)
```

コンソールでの自動最適化の使用

Amazon OpenSearch Service コンソールを使用して、ベクトル取り込みジョブの作成、進行状況のモニタリング、最適化レコメンデーションの表示、それらのレコメンデーションに基づくインデックスの構築を行うことができます。

前提条件

コンソールで自動最適化を使用する前に、次のものが必要です。

- OpenSearch コンソールにアクセスできるアクティブな AWS アカウント。
- タイプベクトル検索またはマネージド OpenSearch ドメインの既存の OpenSearch Serverless コレクション。
- 次のアクションの IAM アクセス許可:
  - `opensearch:SubmitAutoOptimizeJob`
  - `opensearch:GetAutoOptimizeJob`
  - `opensearch:DeleteAutoOptimizeJob`
  - `opensearch:CancelAutoOptimizeJob`
  - `opensearch:ListAutoOptimizeJobs`

  > **Note**
  >
  > これらはアイデンティティベースのポリシーです。 AWS は、リソースを自動最適化するためのリソースベースのポリシーをサポートしていません。

- 認証情報の有効期限が 1 時間以上になるようにフェデレーティッドユーザーセッションを設定します。非常に大きなデータセットや高次元の場合は、有効期限を最大 3 時間延長することを検討してください。

ベクトル取り込みジョブの作成

ベクトル取り込みジョブは、ベクトルデータを分析し、インデックス設定の最適化に関する推奨事項を提供します。

ベクトル取り込みジョブを作成するには

1. で Amazon OpenSearch Service コンソールにサインインしますAWS マネジメントコンソール。
2. ナビゲーションペインで、自動最適化を選択します。
3. ベクトル取り込みジョブの作成 を選択します。
4. ジョブの詳細に、ベクトル取り込みジョブの名前を入力します。この名前は、コンソールでジョブを識別するのに役立ちます。
5. データソースで、以下を設定します。

   a. Amazon S3 URI の場合は、データファイルを含むフォルダ (Parquet または JSONL) の Amazon S3 URI を入力します。URI は、個々のファイルではなく、囲みフォルダを指す必要があります。たとえば、ファイルが `s3://my-bucket/my-folder/file1.parquet`または にある場合は`s3://my-bucket/my-folder/data.jsonl`、 と入力します`s3://my-bucket/my-folder/`。

   > **Note**
   >
   > フォルダには 1 つの形式のファイルが含まれている必要があります。Parquet ファイルと JSONL ファイルを同じフォルダに混在させないでください。

   b. リージョンで、Amazon S3 バケットがある AWS リージョンを選択します。リージョンはバケットの場所と一致する必要があります。
6. OpenSearch ドメインで、既存のドメインまたはコレクションを選択するか、新規作成を選択して作成します。

   > **Note**
   >
   > OpenSearch Managed ドメインまたは OpenSearch Serverless サーバーレスコレクションを指定できます。

7. データソースのアクセス許可で、Amazon S3 バケットと OpenSearch ドメインまたはコレクションにアクセスするアクセス許可を持つ IAM ロールを指定します。ロールには、ドメインまたはコレクション設定に基づいて必要なアクセス許可が必要です。

   - ドメインアクセスポリシーを持つ OpenSearch ドメインの場合、そのポリシーを通じてロールにアクセス権を付与します。

- きめ細かなアクセスコントロールを持つ OpenSearch ドメインの場合は、ロールをバックエンドロールとして追加します。
- OpenSearch Serverless コレクションの場合、データアクセスポリシーに ロールを追加します。

8. [次へ] を選択します。
9. インデックスの設定で、以下を指定します。

   a. フィールド名に、ベクトルデータを含むデータセットのフィールド名を入力します。
   b. スペースタイプでは、ベクトル間の距離を計算するために使用される距離メトリクスを選択します。

      - l2 - ユークリッド距離
      - cosinesimil - コサインの類似性
      - 内部製品 - 内部製品

   c. ディメンションには、各ベクトルの浮動小数点値の数を入力します。
10. パフォーマンス要件で、以下を設定します。

    a. リコールでは、目的の検索品質を 0～1 の 10 進値として指定します。リコール値が高いほど、より関連性の高い結果が返されます。例えば、次のようになります。

       - 0.95 は、クエリベクトルに最も近い 20 個のドキュメントベクトルのうち平均 19 個が返されることを示します。
       - 0.9 は 10 の 9 を示します
       - 0.8 は 10 の 8 を示します

    b. 検索レイテンシーの要件については、レイテンシー許容値を選択します。ささいな要件では、メモリ要件を減らす圧縮方法により、より多くのコスト削減が可能になります。
11. [次へ] を選択します。
12. 設定を確認し、作成を選択します。

ジョブは処理を開始します。Vector Ingestion Jobs テーブルで進行状況をモニタリングできます。

最適化ジョブのモニタリング

ベクトル取り込みジョブのステータスは、自動最適化ランディングページからモニタリングできます。

最適化ジョブをモニタリングするには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、自動最適化を選択します。
3. Vector Ingestion Jobs テーブルには、現在のステータスのすべてのジョブが表示されます。更新されたステータス情報を表示するには、ページを更新します。

> **Note**
>
> 自動更新や通知のメカニズムはありません。ジョブの完了を確認するには、コンソールを手動で更新する必要があります。

ジョブのステータス状態について

自動最適化ジョブのステータス値は次のとおりです。

保留中

ジョブはキューに入れられ、開始を待っています。

実行中

自動最適化ジョブは、データを積極的に分析し、レコメンデーションを生成しています。

完了

自動最適化ジョブが正常に終了しました。すべての分析、評価、レコメンデーションが完了し、表示できるようになります。

失敗

ジョブでエラーが発生しました。ジョブの詳細ページでエラーの詳細を表示して、原因を特定します。

アクティブ

アタッチされたクラスターにインデックスが作成され、データが取り込まれました。

ジョブの期間は、主にデータセットのサイズと現在のサービス負荷によって異なります。一般的なジョブは 15 分から数時間以内に完了します。

ジョブの詳細の表示

設定やステータスなど、特定の最適化ジョブに関する詳細情報を表示できます。

ジョブの詳細を表示するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、自動最適化を選択します。
3. Vector Ingestion Jobs テーブルで、ジョブ名を選択します。
4. ジョブの詳細ページには、次の情報が表示されます。

   - ジョブ名とステータス
   - データソース設定 (Amazon S3 URI とリージョン)
   - OpenSearch ドメインまたはコレクション
   - インデックス設定 (フィールド名、スペースタイプ、ディメンション)
   - パフォーマンス要件 (リコールとレイテンシー)
   - エラーメッセージ (ジョブが失敗した場合)

結果の表示と理解

ジョブが正常に完了すると、最適化の推奨事項を表示できます。

最適化結果を表示するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、自動最適化を選択します。
3. ベクトル取り込みジョブ テーブルで、完了ステータスのジョブを選択します。
4. 結果ページには、次のセクションが表示されます。

   - 結果の概要 - 要件と比較した推定検索品質の再現率と、推奨される上位の設定と比較したインデックスメモリフットプリントを表示します。
   - レコメンデーション - 最大 3 つの最適化レコメンデーションを一覧表示し、設定に最適なレコメンデーションを順に並べます。各レコメンデーションには以下が含まれます。
     - インデックス設定パラメータ

- 検索設定パラメータ
- 期待されるパフォーマンスメトリクス
- メモリフットプリントの見積もり

> **Note**
>
> レコメンデーションは最適な順に並べられますが、特定のユースケースにより適したレコメンデーションを選択できます。自動最適化は、選択したリコール基準に最も近いものを見つけようとします。

レコメンデーションからのインデックスの構築

最適化レコメンデーションを確認したら、レコメンデーション設定を使用してインデックスを手動で作成するか、選択したレコメンデーションを使用してインデックスを自動的に作成できます。

インデックスを自動的に構築するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、自動最適化を選択します。
3. ベクトル取り込みジョブ テーブルで、完了ステータスのジョブを選択します。
4. レコメンデーションを確認し、使用するレコメンデーションを選択します。
5. Build index を選択します。
6. 選択したレコメンデーションを使用してクラスターにインデックスが自動的に作成され、データセットからベクトルデータが取り込まれます。

インデックスを手動で構築するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. ナビゲーションペインで、自動最適化を選択します。
3. ベクトル取り込みジョブ テーブルで、完了ステータスのジョブを選択します。
4. レコメンデーションを確認し、選択したレコメンデーションのインデックス設定と検索設定パラメータを書き留めます。

5. OpenSearch API またはコンソールを使用して、推奨パラメータを含むインデックスを手動で作成します。

関連機能

自動最適化は、他の Amazon OpenSearch Service 機能と連携して、ベクトル検索アプリケーションの構築と最適化に役立ちます。

- ベクトルインデックス作成の GPU アクセラレーション - GPU アクセラレーションを使用してベクトルインデックスの構築を加速し、インデックス作成の時間とコストを削減します。
- ベクトル取り込み - Amazon S3 からドメインまたはコレクションにベクトルデータをすばやく取り込み、インデックスを作成します。

ベクトルインデックス作成の GPU アクセラレーション

GPU アクセラレーションは、大規模なベクトルデータベースをより迅速かつ効率的に構築するのに役立ちます。この機能は、新規または既存の OpenSearch ドメインと OpenSearch Serverless OpenSearch コレクションで有効にできます。この機能は GPU アクセラレーションを使用して、データをベクトルインデックスにインデックス化するために必要な時間を短縮します。

GPU アクセラレーションを使用すると、インデックス作成コストの 4 分の 10Xに増やすことができます。

前提条件

GPU アクセラレーションは、OpenSearch バージョン 3.1以降を実行している OpenSearch ドメインと OpenSearch Serverless コレクションでサポートされています。詳細については、Amazon OpenSearch Service ドメインのアップグレード「」、UpdateDomainConfig」、およびUpdateCollection API」を参照してください。 APIs

仕組み

ベクトルインデックスは、階層ナビゲーション可能スモールワールド (HNSW) グラフなどのデータ構造を構築するために、大量のコンピューティングリソースを必要とします。ドメインまたはコレクションで GPU アクセラレーションを有効にすると、OpenSearch はインデックスビルドを高速化する機会を自動的に検出し、インデックスビルドを GPU インスタンスにオフロードします。OpenSearch Service はユーザーに代わって GPU インスタンスを管理し、必要に応じてドメイ

ンまたはコレクションに割り当てます。つまり、使用率を管理したり、アイドル時間の料金を支払ったりすることはありません。

Compute Units (OCU) - Vector Acceleration による有用な処理に対してのみ料金が発生します。各ベクトル加速 OCU は、約 8 GiB の CPU メモリ、2 つの vCPUs、6 GiB の GPU メモリの組み合わせです。詳細については、「the section called “GPU アクセラレーションの料金”」を参照してください。

ドメインまたはコレクションの GPU アクセラレーションを有効にするには、「」を参照してくださいGPU アクセラレーションの有効化。

GPU アクセラレーションの料金

AWS は、OpenSearch がドメインまたはコレクションのインデックス構築ワークロードを高速化する機会を検出したときに課金します。各ベクトル加速 OCU は、約 8 GiB の CPU メモリ、2 つの vCPUs、6 GiB の GPU メモリの組み合わせです。

AWS は OCU に第 2 レベルの粒度で請求します。アカウントステートメントには、コンピューティングのエントリが OCU 時間で表示されます。

例えば、2 つの vCPU と 1 GiB の GPU メモリを使用して 1 時間 GPU アクセラレーターを使用してインデックスを作成すると、1 OCU が請求されます。GPU アクセラレーターの使用中に 9 GiB の CPU メモリを使用する場合は、2 OCU が請求されます。

OpenSearch Serverless は、コレクションをサポートするのに必要な処理能力とストレージに基づいて 1 OCU 単位で追加の OCU を追加します。コストを抑えるために、アカウントの OCU の最大数を設定できます。

> **Note**
>
> プロビジョニングされる OCU 数は随時変動し、正確ではありません。時間の経過とともに、OpenSearch と OpenSearch Serverless が使用するアルゴリズムは引き続き改善され、システムの使用が最小限に抑えられます。

料金の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

GPU アクセラレーションと書き込みオペレーション

OpenSearch のベクトル取り込みレート (MB/秒) が範囲内にある場合、GPU 加速がアクティブ化されます。OpenSearch ドメインでは、 `index.knn.remote_index_build.size.min`と を使用し

てこの範囲を柔軟に設定できますindex.knn.remote_index_build.size.max。たとえば、下限のデフォルトが 50 MB の場合、更新間隔の間に 768 ディメンションを持つ 15,000 個の全精度ベクトルを書き込むと、デフォルトで GPU アクセラレーションがトリガーされます。

データは、次の API オペレーションを使用して書き込まれます。

- フラッシュ
- 一括
- インデックスの再作成
- [Index] (インデックス)
- 更新
- [Delete] (削除)
- 強制マージ

GPU アクセラレーションは、自動セグメントマージと手動セグメントマージの両方でアクティブ化されます。

サポートされているインデックス設定

Faiss エンジンは GPU アクセラレーションをサポートしています。

以下の設定は GPU アクセラレーターをサポートしていません。

- Faiss 製品の量子化
- 反転ファイルインデックス (IVF)
- 非メトリクススペースライブラリ
- Lucene エンジン

サポートされている AWS リージョン

GPU アクセラレーションは、以下で利用できます AWS リージョン。

- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)
- アジアパシフィック (シドニー)

- アジアパシフィック (東京)
- 欧州 (アイルランド)

GPU アクセラレーションの有効化

、、または AWS SDK を使用して OpenSearch ドメインまたは OpenSearch Serverless コレクションを作成または更新するときに AWS マネジメントコンソール AWS CLI、GPU アクセラレーションを有効にできます。

ドメインまたはコレクションで GPU アクセラレーションを有効にすると、この機能はすべてのインデックスでデフォルトで有効になります。インデックスレベルでこの機能を無効にする必要がある場合は、「」を参照してくださいGPU アクセラレーションベクトルインデックスの作成。

コンソール

次の手順では、OpenSearch Serverless OpenSearch マネジメントコンソールを使用して、OpenSearch ドメインと OpenSearch Serverless コレクションの GPU アクセラレーションを有効にします。

Create new domain

GPU アクセラレーションを有効にして OpenSearch ドメインを作成するには、「」を参照してくださいOpenSearch Service ドメインの作成。

Edit existing domain

1. OpenSearch Service マネジメントコンソールを開きます。
2. ナビゲーションペインで、[ドメイン] を選択します。
3. ドメイン名を選択して、ドメインの詳細ページを開きます。
4. アクションを選択し、ドメインを編集します。
5. 高度な機能セクションで、GPU アクセラレーションを有効にするを選択します。この機能を有効にすると、ベクトルインデックス作成オペレーションが高速化されます。
6. [Save changes] (変更の保存) をクリックします。

Create new collection

GPU アクセラレーションを有効にして OpenSearch Serverless コレクションを作成するには、「」を参照してくださいチュートリアル: Amazon OpenSearch Serverless の開始方法。コレク

ションの作成時に、ベクトル検索コレクションタイプを選択し、ベクトル検索設定で GPU アクセラレーターを有効にしてください。

Edit existing collection

1. OpenSearch Service マネジメントコンソールを開きます。
2. ナビゲーションペインで、コレクションを選択します。
3. コレクション名を選択して、コレクションの詳細ページを開きます。
4. デプロイオプションセクションで、ベクトル GPU アクセラレーションを編集します。
5. GPU アクセラレーションを無効または有効にします。
6. [Save changes] (変更の保存) をクリックします。

AWS CLI

Create new domain

次の AWS CLI 例では、米国東部 (バージニア北部) で GPU アクセラレーションが有効になっている OpenSearch ドメインを作成します。テキストを独自の設定の####に置き換えます。

```
aws opensearch create-domain \
    --domain-name my-domain \
    --engine-version OpenSearch_3.1 \
    --cluster-config InstanceType=r6g.xlarge.search,\
        InstanceCount=1,\
        DedicatedMasterEnabled=true,\
        DedicatedMasterCount=3,\
        DedicatedMasterType=m6g.large.search \
    --ebs-options "EBSEnabled=true,\
        VolumeType=gp3,\
        VolumeSize=2000" \
    --encryption-at-rest-options '{"Enabled":true}' \
    --aiml-options '{"ServerlessVectorAcceleration": {"Enabled": true}}' \
    --node-to-node-encryption-options '{"Enabled":true}' \
    --domain-endpoint-options '{"EnforceHTTPS":true,\
        "TLSSecurityPolicy":"Policy-Min-TLS-1-0-2019-07"}' \
    --access-policies '{"Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "*"},
            "Action": "es:*",
            "Resource": "arn:aws:es:us-east-1:123456789012:domain/my-domain/*"
```

```
        }]}' \
    --advanced-security-options '{
        "Enabled":true,
        "InternalUserDatabaseEnabled":true,
        "MasterUserOptions": {
            "MasterUserName":"USER_NAME",
            "MasterUserPassword":"PASSWORD"
        }}' \
    --region us-east-1
```

Edit existing domain

次の AWS CLI 例では、既存の OpenSearch ドメインの GPU アクセラレーションを有効にします。テキストを独自の設定の####に置き換えます。

```
aws opensearch update-domain-config \
    --domain-name my-domain \
    --cluster-config InstanceType=r7g.16xlarge.search,InstanceCount=3 \
    --aiml-options '{"ServerlessVectorAcceleration": {"Enabled": true}}'
```

Create new collection

次の AWS CLI 例では、米国東部 (バージニア北部) で GPU アクセラレーターが有効になっている OpenSearch Serverless コレクションを作成します。テキストを独自の設定の####に置き換えます。

```
aws opensearchserverless create-collection \
    --name "my-collection" \
    --type "VECTORSEARCH" \
    --description "My vector collection with GPU acceleration" \
    --vector-options '{"ServerlessVectorAcceleration": "ENABLED"}' \
    --region us-east-1
```

Edit existing collection

次の AWS CLI 例では、既存の OpenSearch Serverless コレクションの GPU アクセラレーションを有効にします。テキストを独自の設定の####に置き換えます。

```
aws opensearchserverless update-collection \
    --id 07tjusf2h91cunochc \
    --vector-options '{"ServerlessVectorAcceleration": "ENABLED"}' \
    --region us-east-1
```

GPU アクセラレーションベクトルインデックスの作成

ドメインまたはコレクションで GPU アクセラレーションを有効にしたら、GPU 処理を活用できるベクトルインデックスを作成します。

> **Note**
>
> GPU アクセラレーションを有効にしてドメインを作成する場合、`index.knn.remote_index_build.enabled`設定は`true`デフォルトで です。インデックスを作成するときに、この設定を明示的に設定する必要はありません。コレクションの場合、この設定に値を明示的に指定する必要があります。

Creating index with GPU-acceleration

次の例では、GPU 処理用に最適化されたベクトルインデックスを作成します。このインデックスには、768 次元ベクトル (テキスト埋め込みで一般的) が保存されます。

```
PUT my-vector-index
{
  "settings": {
    "index.knn": true,
    "index.knn.remote_index_build.enabled": true
  },
  "mappings": {
    "properties": {
      "vector_field": {
        "type": "knn_vector",
        "dimension": 768
      },
      "text": {
        "type": "text"
      }
    }
  }
}
```

主要な設定要素:

- `"index.knn": true` - K 最近傍機能を有効にする

- `"index.knn.remote_index_build.enabled": true` - このインデックスの GPU 処理を有効にします。ドメインで GPU アクセラレーションが有効になっている場合、この設定を指定しない場合のデフォルトは ですtrue。コレクションの場合、この設定に値を明示的に指定する必要があります。
- `"dimension": 768` - ベクトルサイズを指定します (埋め込みモデルに基づいて調整)

Creating index without GPU-acceleration

次の例では、GPU 処理が無効になっているベクトルインデックスを作成します。このインデックスには、768 次元ベクトル (テキスト埋め込みで一般的) が保存されます。

```
PUT my-vector-index
{
  "settings": {
    "index.knn": true,
    "index.knn.remote_index_build.enabled": false
  },
  "mappings": {
    "properties": {
      "vector_field": {
        "type": "knn_vector",
        "dimension": 768
      },
      "text": {
        "type": "text"
      }
    }
  }
}
```

ベクトルデータのインデックス作成と強制マージ

ドメインまたはコレクションに GPU アクセラレーションベクトルインデックスを作成したら、標準の OpenSearch オペレーションを使用してベクトルデータを追加し、インデックスを最適化できます。GPU アクセラレーションは、インデックス作成のパフォーマンスと強制マージオペレーションの両方を自動的に強化するため、既存のワークフローを変更することなく、大規模なベクトル検索アプリケーションを迅速に構築して維持できます。

ベクトルデータのインデックス作成

通常どおりにベクトルデータをインデックス化します。GPU アクセラレーションは、インデックス作成および強制マージオペレーションに自動的に適用されます。次の例は、バルク API を使用してベクトルドキュメントをインデックスに追加する方法を示しています。各ドキュメントには、数値と関連するテキストコンテンツを含むベクトルフィールドが含まれています。

```
POST _bulk
{"index": {"_index": "my-vector-index"}}
{"vector_field": [0.1, 0.2, 0.3, ...], "text": "Sample document 1"}
{"index": {"_index": "my-vector-index"}}
{"vector_field": [0.4, 0.5, 0.6, ...], "text": "Sample document 2"}
```

強制マージオペレーション

GPU アクセラレーションは強制マージオペレーションにも適用され、ベクトルインデックスの最適化に必要な時間を大幅に短縮できます。強制マージオペレーションはコレクションではサポートされていないことに注意してください。次の例は、すべてのセグメントを 1 つのセグメントに統合してベクトルインデックスを最適化する方法を示しています。

```
POST my-vector-index/_forcemerge?max_num_segments=1
```

ベストプラクティス

ベクトル検索ワークロードの GPU アクセラレーションの利点を最大化するには、次のベストプラクティスに従ってください。

- インデックスクライアントを増やす - インデックス構築中に GPUs を最大限に活用するには、OpenSearch にデータを取り込むインデックスクライアントの数を増やします。これにより、GPU リソースの並列化と使用率が向上します。
- おおよそのしきい値を調整する - `index.knn.advanced.approximate_threshold`設定を変更して、セグメントインデックスのビルドが小さくならないようにします。これにより、取り込みの全体的な速度が向上します。10,000 の値は良い出発点です。コレクションの場合、この設定に値を明示的に指定する必要があります。
- シャードサイズの最適化 - 少なくとも 100 万のドキュメントがあるシャードを作成してみてください。このドキュメント数より少ないシャードでは、GPU アクセラレーションによる全体的なメリットが得られない場合があります。

Amazon OpenSearch Service での OpenSearch Dashboards の使用

OpenSearch Dashboards は OpenSearch で稼働するように設計された、オープンソースの可視化ツールです。Amazon OpenSearch Service は、すべての OpenSearch Service ドメインとともに Dashboards のインストールを提供します。Dashboards は、ドメイン内のホットデータノードで実行されます。

OpenSearch Dashboards は、単一の OpenSearch ドメイン内のデータを探索および分析するための視覚化ツールです。対照的に、一元化された OpenSearch ユーザーインターフェイス (OpenSearch アプリケーションとも呼ばれます) は、複数の OpenSearch ドメイン、OpenSearch Serverless コレクション、 AWS データソースに接続するクラウドベースの UI です。これには、オブザーバビリティやセキュリティ分析などの特定のユースケース用のワークスペースが含まれており、データセット間で統一されたエクスペリエンスを提供します。ダッシュボードは個々のドメインに関連付けられていますが、一元化された UI により、ドメイン間のデータ統合と分析が可能になります。詳細については、「OpenSearch UI」を参照してください。

OpenSearch Dashboards へのリンクは、OpenSearch Service コンソールのドメインダッシュボードにあります。OpenSearch を実行しているドメインの場合、URL は *domain-endpoint*/_dashboards/ です。レガシーの Elasticsearch を実行しているドメインの場合、URL は *domain-endpoint*/_plugin/kibana です。

このデフォルトの Dashboards インストールを使用したクエリは、300 秒でタイムアウトになります。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service のコンテキストにおける OpenSearch Dashboards について説明します。これには、それに接続するさまざまな方法が含まれます。入門ガイド、ダッシュボードの作成手順、ダッシュボード管理、Dashboards Query Language (DQL) などの包括的なドキュメントについては、オープンソースの OpenSearch ドキュメントの「OpenSearch Dashboards」を参照してください。

Dashboards へのアクセスの制御

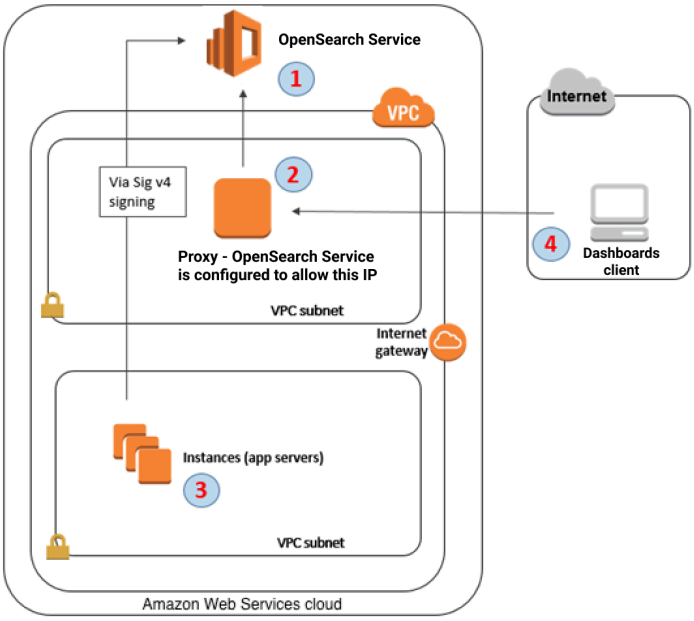
Dashboards では、IAM ユーザーとロールがネイティブにサポートされていませんが、OpenSearch Service には Dashboards へのアクセス権をコントロールするためのいくつかのソリューションが用意されています。

- Dashboards の SAML 認証を有効にします。
- HTTP 基本認証を用いたきめ細かなアクセスコントロールを使用します。
- ダッシュボードの Cognito 認証を設定します。
- パブリックアクセスドメインの場合、プロキシサーバーを使用する (または使用しない) IP ベースのアクセスポリシーを設定します。
- VPC アクセスドメインの場合、プロキシサーバーを使用する (または使用しない) オープンアクセスポリシー、およびセキュリティグループを使用してアクセス許可を制御します。詳細についてはthe section called “VPC ドメインのアクセスポリシーについて”を参照してください。

プロキシを使用して Dashboards から OpenSearch Service にアクセスする

> **Note**
>
> このプロセスは、ドメインでパブリックアクセスが使用されており、Cognito 認証を使用しない場合にのみ適用できます。「the section called “Dashboards へのアクセスの制御”」を参照してください。

Dashboards は JavaScript アプリケーションであるため、リクエストの送信元はユーザーの IP アドレスです。IP ベースのアクセスコントロールは、膨大な数の IP アドレスをホワイトリストに登録する必要があるため、各ユーザーに Dashboards へのアクセスを許可する方法として実用的とは言えません。1 つの回避策は、Dashboards と OpenSearch Service の間にプロキシサーバーを配置することです。これにより、IP ベースのアクセスポリシーを追加し、唯一の IP アドレス (プロキシの IP アドレス) からのリクエストを許可できます。この設定は以下の図のようになります。



1. これは OpenSearch Service ドメインです。IAM は、このドメインへの承認済みアクセスを提供します。追加の IP ベースのアクセスポリシーは、プロキシサーバーへのアクセスを提供します。
2. これは、Amazon EC2 インスタンスで実行されているプロキシサーバーです。
3. 他のアプリケーションは、署名バージョン 4 の署名プロセスを使用して認証済みのリクエストを OpenSearch Service に送信できます。
4. Dashboards クライアントは、プロキシを介して OpenSearch Service ドメインに接続します。

この種の設定を有効にするには、ロールと IP アドレスを指定するリソースベースのポリシーが必要です。ポリシー例を次に示します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Resource":"arn:aws:es:us-west-2:111111111111:domain/my-domain/*",
         "Principal":{
            "AWS":"arn:aws:iam::111111111111:role/allowedrole1"
         },
         "Action":[
            "es:ESHttpGet"
         ],
```

```
            "Effect":"Allow"
         },
         {
            "Effect":"Allow",
            "Principal":{
               "AWS":"*"
            },
            "Action":"es:*",
            "Condition":{
               "IpAddress":{
                  "aws:SourceIp":[
                     "203.0.113.0/24",
                     "2001:DB8:1234:5678::/64"
                  ]
               }
            },
            "Resource":"arn:aws:es:us-west-2:111111111111:domain/my-domain/*"
         }
      ]
}
```

プロキシサーバーを実行する EC2 インスタンスを Elastic IP アドレスを使用して設定することをお勧めします。これにより、必要に応じてインスタンスを置き換え、各インスタンスに同じパブリック IP アドレスをアタッチできます。詳細については、「Elastic IP Addresses」の「Amazon EC2 User Guide」を参照してください。

プロキシサーバーおよび Cognito 認証を使用している場合、`redirect_mismatch` エラーを回避するため、Dashboards と Amazon Cognito に追加の設定が必要になる場合があります。次の `nginx.conf` の例を参照してください。

```
server {
    listen 443;
    server_name $host;
    rewrite ^/$ https://$host/_plugin/_dashboards redirect;

    ssl_certificate          /etc/nginx/cert.crt;
    ssl_certificate_key      /etc/nginx/cert.key;

    ssl on;
    ssl_session_cache  builtin:1000  shared:SSL:10m;
    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
```

```
    ssl_prefer_server_ciphers on;

    location /_plugin/_dashboards {
        # Forward requests to Dashboards
        proxy_pass https://$dashboards_host/_plugin/_dashboards;

        # Handle redirects to Cognito
        proxy_redirect https://$cognito_host https://$host;

        # Update cookie domain and path
        proxy_cookie_domain $dashboards_host $host;
        proxy_cookie_path / /_plugin/_dashboards/;

        # Response buffer settings
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
    }

    location ~ \/(log|sign|fav|forgot|change|saml|oauth2) {
        # Forward requests to Cognito
        proxy_pass https://$cognito_host;

        # Handle redirects to Dashboards
        proxy_redirect https://$dashboards_host https://$host;

        # Update cookie domain
        proxy_cookie_domain $cognito_host $host;
    }
}
```

> **Note**
>
> (オプション) 専用コーディネーターノードのプロビジョニングを選択すると、OpenSearch Dashboards のホスティングが自動的に開始されます。その結果、CPU やメモリなどのデータノードリソースの可用性が向上します。データノードリソースの可用性が向上することで、ドメインの全体的な耐障害性を向上させることができます。

WMS マップサーバーを使用するための Dashboards の設定

OpenSearch Service の Dashboards のデフォルトインストールにはマップサービスが含まれますが、インドおよび中国リージョンのドメインは除きます。マップサービスは、最大 10 のズームレベルをサポートします。

リージョンに関係なく、座標マップの可視化に別の Web Map Service (WMS) サーバーが使用されるように Dashboards を設定できます。リージョンマップの可視化では、デフォルトのマップサービスのみがサポートされます。

WMS マップサーバーを使用できるように Dashboards を設定するには:

1. Dashboards を開きます。
2. [スタックの管理] を選択します。
3. [詳細設定] を選択します。
4. visualization:tileMap:WMSdefaults を見つけます。
5. `enabled` を `true` に変更し、`url` を有効な WMS マップサーバーの URL に変更します。

```
{
  "enabled": true,
  "url": "wms-server-url",
  "options": {
    "format": "image/png",
    "transparent": true
  }
}
```

6. [変更を保存] を選択します。

新しいデフォルト値を可視化に適用するには、Dashboards の再ロードが必要になることがあります。可視化結果を保存した場合は、可視化結果を開いた後に [オプション] を選択します。[WMS マップサーバー] が有効で、[WMS URL)] に優先マップサーバーが指定されていることを確認し、[変更の適用] を選択します。

> **Note**
>
> マップサービスは多くの場合、ライセンス料や制限事項を伴います。マップサーバーを指定する際には、このような点を考慮する必要があります。テストを行うには、アメリカ地質調査所のマップサービスが便利です。

ローカルの Dashboards サーバーを OpenSearch Service に接続する

独自の Dashboards インスタンスを設定するためにすでに時間をかけた場合は、OpenSearch Service から提供されるデフォルトの Dashboards インスタンスの代わりに (または追加で) 独自のインスタンスを使用することもできます。以下の手順は、オープンアクセスポリシーできめ細かなアクセス制御を使用するドメイン用です。

ローカルの Dashboards サーバーを OpenSearch Service に接続するには

1. OpenSearch Service ドメインで、適切な許可を持つユーザーを作成します。
   a. Dashboards で、[セキュリティ]、[内部ユーザー] に進み、[内部ユーザーの作成] を選択します。
   b. ユーザーネームとパスワードを入力し、[作成] を選択します。
   c. [ロール] に進み、ロールを選択します。
   d. [マッピングされたユーザー] を選択し、[マッピングの管理] を選択します。
   e. [ユーザー] で、ユーザーネームを追加し、[マップ] を選択します。
2. 適切なバージョンの OpenSearch セキュリティプラグインをセルフマネージド Dashboards の OSS インストールにダウンロードしてインストールします。
3. ローカルの Dashboards サーバーで、`config/opensearch_dashboards.yml` ファイルを開き、前に作成したユーザー名とパスワードで OpenSearch Service エンドポイントを追加します。

   ```
   opensearch.hosts: ['https://domain-endpoint']
   opensearch.username: 'username'
   opensearch.password: 'password'
   ```

   以下のサンプル `opensearch_dashboards.yml` ファイルを使用できます。

```
server.host: '0.0.0.0'

opensearch.hosts: ['https://domain-endpoint']

opensearchDashboards.index: ".username"

opensearch.ssl.verificationMode: none # if not using HTTPS

opensearch_security.auth.type: basicauth
opensearch_security.auth.anonymous_auth_enabled: false
opensearch_security.cookie.secure: false # set to true when using HTTPS
opensearch_security.cookie.ttl: 3600000
opensearch_security.session.ttl: 3600000
opensearch_security.session.keepalive: false
opensearch_security.multitenancy.enabled: false
opensearch_security.readonly_mode.roles: ['opensearch_dashboards_read_only']
opensearch_security.auth.unauthenticated_routes: []
opensearch_security.basicauth.login.title: 'Please log in using your username and
 password'

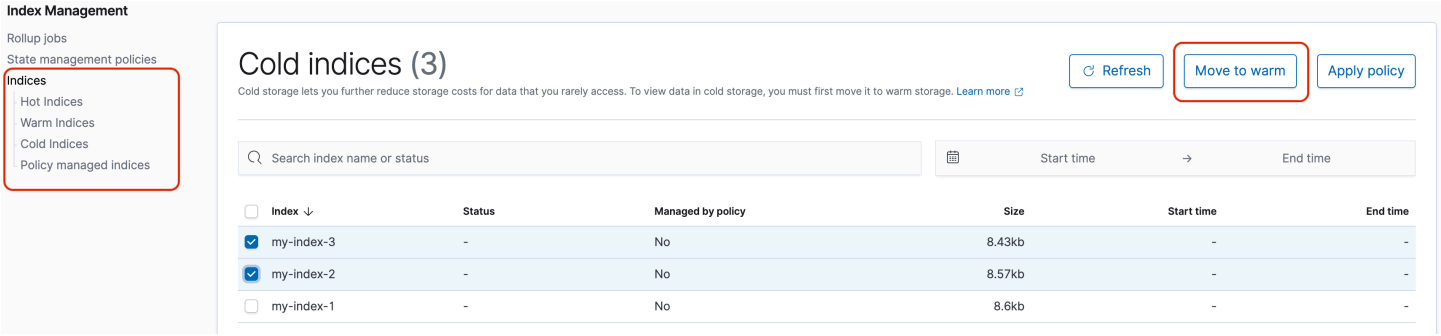
opensearch.username: 'username'
opensearch.password: 'password'
opensearch.requestHeadersWhitelist: [authorization, securitytenant,
 security_tenant]
```

OpenSearch Service のインデックスを表示するには、ローカルの Dashboards サーバーを起動し、[Dev Tools] に進み、以下のコマンドを実行します。

```
GET _cat/indices
```

Dashboards でのインデックスの管理

OpenSearch Service ドメインに Dashboards をインストールすると、ドメイン上の異なるストレージ階層のインデックスを管理するのに便利な UI が提供されます。Dashboards のメインメニューからインデックス管理を選択して、すべてのインデックスをホット UltraWarm、およびコールドストレージで表示し、またインデックスステート管理 (ISM) ポリシーによって管理されるインデックスも表示します。インデックス管理を使用して、ウォームストレージとコールドストレージ間でインデックスを移動し、3 つの階層間の移行をモニタリングします。



UltraWarm またはコールドストレージが有効になっている場合を除き、ホット、ウォーム、コールドインデックスオプションは表示されません。

その他の機能

各 OpenSearch Service ドメインにおける Dashboards のデフォルトインストールには、いくつかの追加機能が含まれています:

- さまざまな OpenSearch プラグインのユーザーインターフェイス
- テナント
- レポート

  レポート作成メニューを使用して、Discover ページおよびダッシュボードまたは可視化の PDF レポートまたは PNG レポートからオンデマンド CSV レポートを生成します。CSV レポートには 10,000 行までの制限があります。
- ガントチャート
- ノートブック

Amazon OpenSearch Service での OpenSearch UI の使用

OpenSearch UI (ユーザーインターフェイス) は、Amazon OpenSearch Service のモダナイズされた運用分析エクスペリエンスであり、複数のソース間でデータを操作するための統一されたビューを提供します。OpenSearch Dashboards がホストされている単一のドメインまたはコレクションでのみ動作するのに対し、OpenSearch UI は AWS クラウドでホストされています。これにより、OpenSearch UI は高可用性を実現し、クラスターのアップグレード中も機能し続け、複数のデータソースにネイティブに接続できます。OpenSearch Dashboards の詳細については、「OpenSearch Dashboards」を参照してください。

OpenSearch UI の主要な機能は以下のとおりです。

- 複数のデータソースのサポート – OpenSearch UI は、複数のデータソースに接続して包括的なビューを作成できます。これには、OpenSearch ドメインとサーバーレスコレクション、Amazon CloudWatch、Amazon Security Lake、Amazon Simple Storage Service (Amazon S3) などの統合 AWS データソースが含まれます。
- アップグレード中のダウンタイムゼロ – OpenSearch UI は AWS クラウドでホストされます。つまり、OpenSearch は引き続き動作し、アップグレードプロセス中にクラスターからデータを取得できます。
- ワークスペース – オブザーバビリティ、セキュリティ分析、検索など、さまざまなワークフローのチーム共同作業のための厳選されたスペース。プライバシー設定を定義し、ワークスペース内の共同作業者のアクセス許可を管理できます。
- シングルサインオン – OpenSearch UI は、 AWS IAM アイデンティティセンター および SAML through AWS Identity and Access Management (IAM) フェデレーションと連携して ID プロバイダーと統合し、エンドユーザー向けのシングルサインオンエクスペリエンスを作成します。
- GenAI を活用した分析 – OpenSearch UI は自然言語クエリ生成をサポートしているため、分析に適したクエリを生成できます。OpenSearch UI は Amazon Q Developer と連携し、Amazon Q チャットを提供するとともに、可視化、アラート概要、インサイト、推奨される異常検知器の生成を支援します。
- 複数クエリ言語のサポート – OpenSearch UI は、Piped Processing Language (PPL)、SQL、Lucene、および Dashboards Query Language (DQL) をサポートしています。
- クロスリージョンおよびクロスアカウントのサポート – OpenSearch UI は、クラスター間検索機能を利用して、異なるアカウントおよび異なるリージョンの OpenSearch ドメインに接続し、集約された分析と視覚化を行うことができます。

開始して最初の OpenSearch UI を作成するには、「the section called “開始方法”」の手順に従います。

OpenSearch UI 用にリリースされた最新の機能については、「the section called “リリース履歴”」を参照してください。

トピック

- Amazon OpenSearch Service ユーザーインターフェイスのリリース履歴
- Amazon OpenSearch Service における OpenSearch ユーザーインターフェースの開始方法
- Amazon OpenSearch Service のエージェント AI
- カスタマーマネージドキーによる OpenSearch UI アプリケーションメタデータの暗号化
- を使用した SAML フェデレーションの有効化 AWS Identity and Access Management
- データソースの関連付けと Virtual Private Cloud アクセス許可の管理
- Amazon OpenSearch Service ワークスペースの使用
- クロスリージョンおよびクロスアカウントデータアクセス
- VPC エンドポイントからの OpenSearch UI へのアクセスの管理
- 保存されたオブジェクトを OpenSearch Dashboards から OpenSearch UI に移行する
- OpenSearch UI エンドポイントとクォータ

Amazon OpenSearch Service ユーザーインターフェイスのリリース履歴

以下の表は、Amazon OpenSearch Service の OpenSearch UI サポートに関する全リリースと、各リリースに含まれる機能および強化点を一覧表示します。

変更	リリース日	説明
機能を追加	2026-06-10	Amazon OpenSearch Service は MCP アプリをサポートするようになりました。これにより、Claude Desktop や VS Code などの互換性のあるエージェント IDEs。AI エージェントは、OpenSearch ドメイン、コレクション、Amazon Managed Service for Prometheus に保存されているログ、トレース、メトリクス、アラートを使用してインシデントを調査し、同じ会話でレンダリングされたインタラクティブな視

変更	リリース日	説明
		覚化の結果を確認できます。詳細については、「the section called “MCP アプリによるエージェントのオブザーバビリティ”」を参照してください。
機能を追加	2026-05-01	OpenSearch UI はOpenSearch ドメインへのクロスリージョンデータアクセスをサポートするようになりました。エンドポイントを切り替えたりデータをレプリケートしたりすることなく、パブリックドメインと VPC ドメインの両方について、単一の OpenSearch UI アプリケーション AWS リージョン とは異なる でホストされているドメインでダッシュボードをクエリまたは構築できます。クロスリージョンデータアクセスは、クロスアカウントデータアクセスと組み合わせることができ、IAM と AWS IAM アイデンティティセンター 認証をサポートします。詳細については、「the section called “OpenSearch ドメインへのクロスリージョンデータアクセス”」を参照してください。
機能を追加	2026-04-09	Amazon OpenSearch Service は、メトリクス、ログ、トレース、AI エージェントトレースをまとめた OpenSearch UI の統合オブザーバビリティエクスペリエンスを提供するようになりました。このリリースでは、Amazon Managed Service for Prometheus とのネイティブ統合が追加されました。これにより、PromQL で Prometheus メトリクスをログとトレースとともにクエリできるほか、OpenTelemetry GenAI セマンティック規則を使用した RED メトリクスと AI エージェントトレースによるアプリケーションモニタリングも利用できます。詳細については、「the section called “メトリクスの検出”」を参照してください。

変更	リリース日	説明
機能を追加	2026-03-31	Amazon OpenSearch Service は、OpenSearch UI のオブザーバビリティワークスペースでログ分析用のエージェント AI 機能を提供するようになりました。エージェントチャットを使用して自然言語でデータを分析し、調査エージェントをトリガーして根本原因分析を自律的に実行し、エージェントメモリを使用して機能ページとセッション間で会話を続けることができます。詳細については、「the section called “エージェントチャット”」を参照してください。
機能を追加	2026-03-12	OpenSearch UI は、OpenSearch ドメインへのクロスアカウントデータアクセスをサポートするようになりました。パブリックドメインと VPC ドメインの両方について、IAM または AWS IAM アイデンティティセンター 認証を使用して異なるアカウントのドメインにアクセスするように、1 つのアカウントでアプリケーションを設定できます。詳細については、「the section called “OpenSearch ドメインへのクロスアカウントデータアクセス”」を参照してください。
機能を追加	2025-12-29	OpenSearch UI は、アプリケーションメタデータを暗号化するためのカスタマーマネージドキー (CMK) をサポートするようになりました。この機能は、暗号化キーを完全に制御することで、規制およびコンプライアンス要件を満たすのに役立ちます。この起動により、OpenSearch UI に保存されたオブジェクトのメタデータサイズ制限も引き上げられるため、複雑なクエリ、広範な視覚化、大規模なワークスペースを作成および保存できます。詳細については、「the section called “暗号化とカスタマーマネージドキー”」を参照してください。

変更	リリース日	説明
機能を追加	2025-11-24	Amazon OpenSearch Service は、パイプ処理言語 (PPL) と自然言語を OpenSearch UI のオブザーバビリティワークスペースのデフォルトのクエリエクスペリエンスにするようになりました。強化されたエクスペリエンスにより、深層分析、ファセット探索、自然言語クエリ用の新しいコマンドが追加され、単一のインターフェイス内でクエリから視覚化に移行できます。詳細については、「the section called “PPL を使用したログのクエリ”」を参照してください。
機能を追加	2025-11-19	Amazon OpenSearch Service が Cluster Insights を提供するようになりました。これは、OpenSearch UI の単一のダッシュボードを通じてクラスターの包括的な運用可視性を提供するモニタリングソリューションです。これにより、ノード、インデックス、シャードにまたがる運用データが統合され、関連するメトリクス、影響を受けるリソース、トップ N クエリ分析、修復ステップが表示されます。OpenSearch 2.17 以降を実行しているドメインでは、追加料金なしで利用できます。詳細については、「クラスターインサイト」を参照してください。
機能を追加	2025-09-15	Amazon OpenSearch Service は、OpenSearch ドメインでインデックス化された時系列データの AI を活用した予測をサポートするようになりました。OpenSearch UI 内で予測と視覚化を生成して、データサイエンスや AI の専門知識を必要とせずに、インフラストラクチャの使用率、アプリケーションのパフォーマンス、ビジネスメトリクスなどのメトリクスの傾向を予測できます。OpenSearch 3.1 以降を実行しているドメインで使用できます。

変更	リリース日	説明
機能を追加	2025-08-08	OpenSearch UI は、IAM フェデレーションを使用して SAML 経由でアクセスするときに、きめ細かなアクセスコントロールをサポートするようになりました。ID プロバイダーのユーザー属性とロールを OpenSearch バックエンドロールにマッピングして、インデックスレベルのアクセス許可とドキュメントレベルのセキュリティを適用して、より詳細なデータアクセスコントロールを実現できます。詳細については、「the section called “IAM による SAML フェデレーションの有効化”」を参照してください。
機能を追加	2025-04-16	OpenSearch UI がクラスター間検索で動作するようになりました。これにより、一つの AWS リージョン で OpenSearch UI を使用して、別のリージョンのクラスターにアクセスできるようになります。これを行うには、アクセスしたいクラスターを、同じリージョン内のクラスターに接続されたリモートクラスターとして設定します。詳細については、「the section called “クラスター間検索”」を参照してください。
機能を追加	2025-03-31	Amazon Q Developer が Amazon OpenSearch Service で一般提供されました。
機能を追加	2025-02-05	Security Assertion Markup Language 2.0) (SAML) から AWS Identity and Access Management (IAM) フェデレーションが OpenSearch UI で動作するようになりました。これにより、エンドユーザー向けの ID プロバイダー主導のシングルサインオン (SSO) エクスペリエンスを作成できます。詳細については、「the section called “IAM による SAML フェデレーションの有効化”」を参照してください。
新しい統合	2024-12-01	Amazon CloudWatch とのゼロ ETL 統合は、ログデータの分析と視覚化を簡素化し、技術的なオーバーヘッドと運用コストを削減します。詳細については、「AWS ニュースブログ」の「New Amazon CloudWatch and Amazon OpenSearch Service launch an integrated analytics experience」を参照してください。

変更	リリース日	説明
新しい統合	2024-12-01	Amazon Security Lake とのゼロ ETL 統合により、組織はセキュリティデータから実用的なインサイトを効率的に検索、分析、取得できます。詳細については、「AWS ニュースブログ」の「セキュリティ分析を簡素化するための Amazon OpenSearch Service と Amazon Security Lake の統合のご紹介」を参照してください。
初回リリース	2024-11-07	OpenSearch UI の最初の一般リリースです。詳細については、「AWS ビッグデータブログ」の「Amazon OpenSearch Service が次世代 OpenSearch UI を発表」を参照してください。

Amazon OpenSearch Service における OpenSearch ユーザーインターフェースの開始方法

Amazon OpenSearch Service では、アプリケーションは OpenSearch ユーザーインターフェイス (OpenSearch UI) のインスタンスです。各アプリケーションは複数のデータソースに関連付けることができ、1 つのソースを複数のアプリケーションに関連付けることができます。サポートされているさまざまな認証オプションを使用して、管理者ごとに複数のアプリケーションを作成できます。

このトピックの情報を使用して、 AWS マネジメントコンソール または を使用して OpenSearch UI アプリケーションを作成するプロセスについて説明します AWS CLI。

トピック

- Amazon OpenSearch Service アプリケーションを作成するために必要なアクセス許可
- OpenSearch UI アプリケーションの作成
- アプリケーション管理者の管理

Amazon OpenSearch Service アプリケーションを作成するために必要なアクセス許可

アプリケーションを作成する前に、タスクに必要なアクセス許可が付与されていることを確認します。必要に応じて、アカウント管理者にお問い合わせください。

全般的なアクセス許可

OpenSearch Service でアプリケーションを使用するには、次のポリシーに示すアクセス許可が必要です。アクセス許可は以下の目的のために設定されます:

- アプリケーションを作成および管理するには、5 つの `es:*Application` アクセス許可が必要です。
- アプリケーションからタグを追加、一覧表示、削除するには、3 つの `es:*Tags` アクセス許可が必要です。
- データソースを関連付けるには `aoss:BatchGetCollection`、`es:DescribeDomain`、`es:GetDirectQueryDataSource` のアクセス許可が必要です。
- データソースにアクセスするには `aoss:APIAccessAll`、`es:ESHttp*`、および 4 つの `opensearch:*DirectQuery*` のアクセス許可が必要です。
- `iam:CreateServiceLinkedRole` は、アカウントにサービスにリンクされたロール (SLR) を作成するためのアクセス許可を Amazon OpenSearch Service に提供します。このロールは、OpenSearch UI アプリケーションが Amazon CloudWatch メトリクスをお使いのアカウントに公開できるようにするために使用されます。詳細については、「サービスにリンクされたロールを使用して、VPC ドメインとダイレクトクエリデータソースを作成する」トピックの「the section called “アクセス許可”」を参照してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "es:CreateApplication",
                "es:DeleteApplication",
                "es:GetApplication",
                "es:ListApplications",
                "es:UpdateApplication",
                "es:AddTags",
                "es:ListTags",
                "es:RemoveTags",
```

```
                "aoss:APIAccessAll",
                "es:ESHttp*",
                "opensearch:StartDirectQuery",
                "opensearch:GetDirectQuery",
                "opensearch:CancelDirectQuery",
                "opensearch:GetDirectQueryResult",
                "aoss:BatchGetCollection",
                "aoss:ListCollections",
                "es:DescribeDomain",
                "es:DescribeDomains",
                "es:ListDomainNames",
                "es:GetDirectQueryDataSource",
                "es:ListDirectQueryDataSources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService"
        }
    ]
}
```

IAM Identity Center 認証を使用するアプリケーションを作成するアクセス許可 (オプション)

デフォルトでは、ダッシュボードアプリケーションは AWS Identity and Access Management (IAM) を使用して認証され、 AWS リソースユーザーのアクセス許可を管理します。ただし、IAM Identity Center を使用してシングルサインオンエクスペリエンスを提供することを選択できます。これにより、既存の ID プロバイダーを使用して OpenSearch UI アプリケーションにログインできます。この場合、このトピックの後半の手順で [IAM Identity Center による認証] オプションを選択し、OpenSearch UI アプリケーションにアクセスするために必要なアクセス許可を IAM Identity Center ユーザーに付与します)。

IAM Identity Center 認証を使用するアプリケーションを作成するには、次のアクセス許可が必要です。*#########*を、ユーザー自身の情報に置き換えます。必要に応じて、アカウント管理者にお問い合わせください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "IDCPermissions",
            "Effect": "Allow",
            "Action": [
                "es:CreateApplication",
                "es:DeleteApplication",
                "es:GetApplication",
                "es:ListApplications",
                "es:UpdateApplication",
                "es:AddTags",
                "es:ListTags",
                "es:RemoveTags",
                "aoss:BatchGetCollection",
                "aoss:ListCollections",
                "es:DescribeDomain",
                "es:DescribeDomains",
                "es:ListDomainNames",
                "es:GetDirectQueryDataSource",
                "es:ListDirectQueryDataSources",
                "sso:CreateApplication",
                "sso:DeleteApplication",
                "sso:PutApplicationGrant",
                "sso:PutApplicationAccessScope",
                "sso:PutApplicationAuthenticationMethod",
                "sso:ListInstances",
                "sso:DescribeApplicationAssignment",
                "sso:DescribeApplication",
                "sso:CreateApplicationAssignment",
                "sso:ListApplicationAssignments",
                "sso:DeleteApplicationAssignment",
                "sso-directory:SearchGroups",
                "sso-directory:SearchUsers",
                "sso:ListDirectoryAssociations",
                "identitystore:DescribeUser",
                "identitystore:DescribeGroup",
                "iam:ListRoles"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "SLRPermission",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService"
        },
        {
            "Sid": "PassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/iam-role-for-identity-
center"
        }
    ]
}
```

OpenSearch UI アプリケーションの作成

次のいずれかの手順を使用して、アプリケーション名、認証方法、および管理者を指定するアプリケーションを作成します。

トピック

- コンソールで IAM 認証を使用する OpenSearch UI アプリケーションの作成
- コンソールで AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成する
- を使用した AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションの作成 AWS CLI

コンソールで IAM 認証を使用する OpenSearch UI アプリケーションの作成

コンソールで IAM 認証を使用する OpenSearch UI アプリケーションを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインの [OpenSearch UI (ダッシュボード)] を選択します。
3. [アプリケーションを作成] を選択します。

4. [アプリケーション名] に、そのアプリケーションの名前を入力します。
5. [IAM Identity Center による認証] チェックボックスをオンにしないでください。認証を使用したアプリケーションの作成については AWS IAM アイデンティティセンター、このトピックのthe section called “コンソールで AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成する”後半の「」を参照してください。
6. (オプション) 作成するアプリケーションの管理者として自動的に追加されます。[OpenSearch アプリケーション管理者管理] エリアでは、他のユーザーに管理者権限を付与できます。

   > **Note**
   >
   > OpenSearch UI アプリケーション管理者ロールは、OpenSearch UI アプリケーションを編集および削除する権限を付与します。アプリケーション管理者は、OpenSearch UI アプリケーションでワークスペースを作成、編集、削除することもできます。

   管理者権限を他のユーザーに付与するには、次のいずれかを選択します。

   - [特定のユーザー (複数可) に管理者の権限を付与] – [OpenSearch アプリケーション管理者] フィールドの [プロパティ] ポップアップリストで、[IAM ユーザー] または

     AWS IAM アイデンティティセンター ユーザーを選択し、管理者権限を付与する個々のユーザーを選択します。
   - [すべてのユーザーに管理者権限を付与] – 組織またはアカウント内のすべてのユーザーに管理者権限が付与されます。
7. (オプション) 暗号化設定を構成します。デフォルトでは、OpenSearch UI メタデータは AWS 所有キーで暗号化されます。暗号化に独自のカスタマーマネージドキー (CMK) を使用するには、「」を参照してくださいthe section called “暗号化とカスタマーマネージドキー”。
8. (オプション) [タグ] エリアで、1 つ以上のタグキーの名前と値のペアをアプリケーションに適用します。

   タグは、リソースに割り当てるオプションのメタデータです。タグを使用すると、目的、所有者、環境などのさまざまな方法でリソースを分類できます。
9. [作成] を選択します。

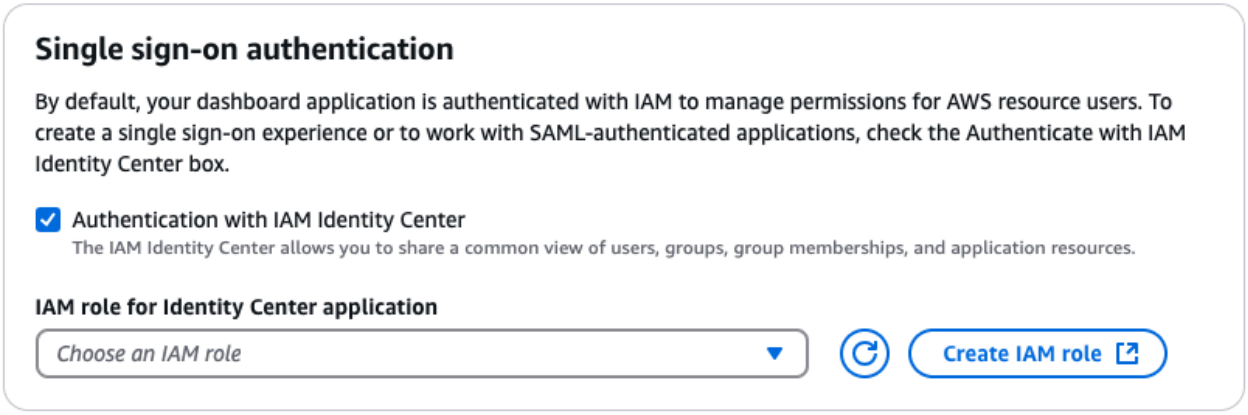
コンソールで AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成する

AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成するには、the section called “IAM Identity Center 認証を使用するアプリケーションを作成するアクセス許可 (オプション)” のこのトピックで前述した IAM アクセス許可が必要です。

コンソールで AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインの [OpenSearch UI (ダッシュボード)] を選択します。
3. [アプリケーションを作成] を選択します。
4. [アプリケーション名] に、そのアプリケーションの名前を入力します。
5. (オプション) 組織またはアカウントのシングルサインオンを有効にするには、次の手順を実行します。

   a. 次の図に示すように、[IAM Identity Center による認証] チェックボックスをオンにします。



   b. 次のいずれかを行います。

   - [Identity Center アプリケーションリストの IAM ロール] で、IAM Identity Center が OpenSearch UI および関連するデータソースにアクセスするために必要なアクセス許可を提供する既存の IAM ロールを選択します。ロールに必要なアクセス許可については、次の箇条書きのポリシーを参照してください。

- 必要なアクセス許可を持つ新しいロールを作成します。IAM ユーザーガイドの次の手順と、指定されたオプションを使用して新しいロールを作成し、必要なアクセス許可ポリシーと信頼ポリシーを使用します。
  - 手順: IAM ポリシーを作成する (コンソール)

    この手順のステップに従って、次のポリシーをポリシーエディタ [JSON] フィールドに貼り付けます:

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "IdentityStoreOpenSearchDomainConnectivity",
            "Effect": "Allow",
            "Action": [
                "identitystore:DescribeUser",
                "identitystore:ListGroupMembershipsForMember",
                "identitystore:DescribeGroup"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledViaLast": "es.amazonaws.com"
                }
            }
        },
        {
            "Sid": "OpenSearchDomain",
            "Effect": "Allow",
            "Action": [
                "es:ESHttp*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "OpenSearchServerless",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

- 手順: カスタム信頼ポリシーを使用してロールを作成する

この手順のステップに従って、[カスタム信頼ポリシー] ボックスのプレースホルダー JSON を次のように置き換えます。

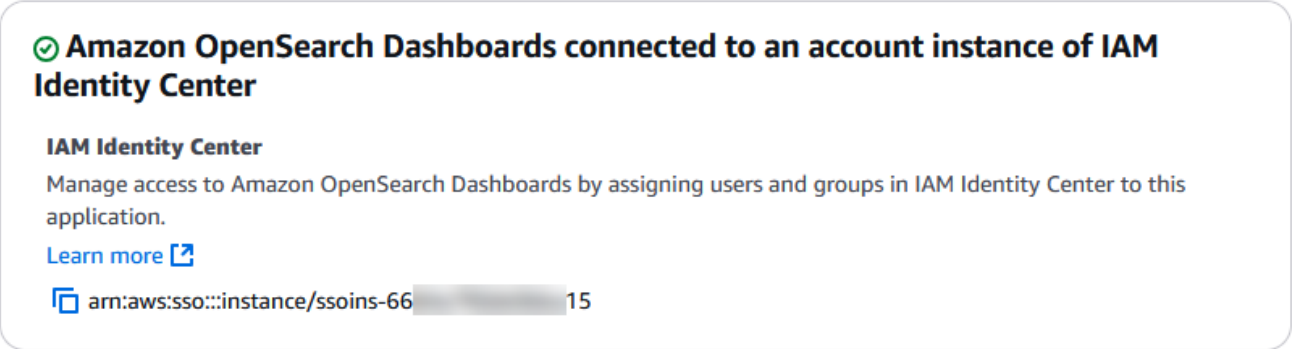
> **Tip**
>
> 既存のロールに信頼ポリシーを追加する場合は、ロールの [信頼関係] タブにポリシーを追加します。

JSON

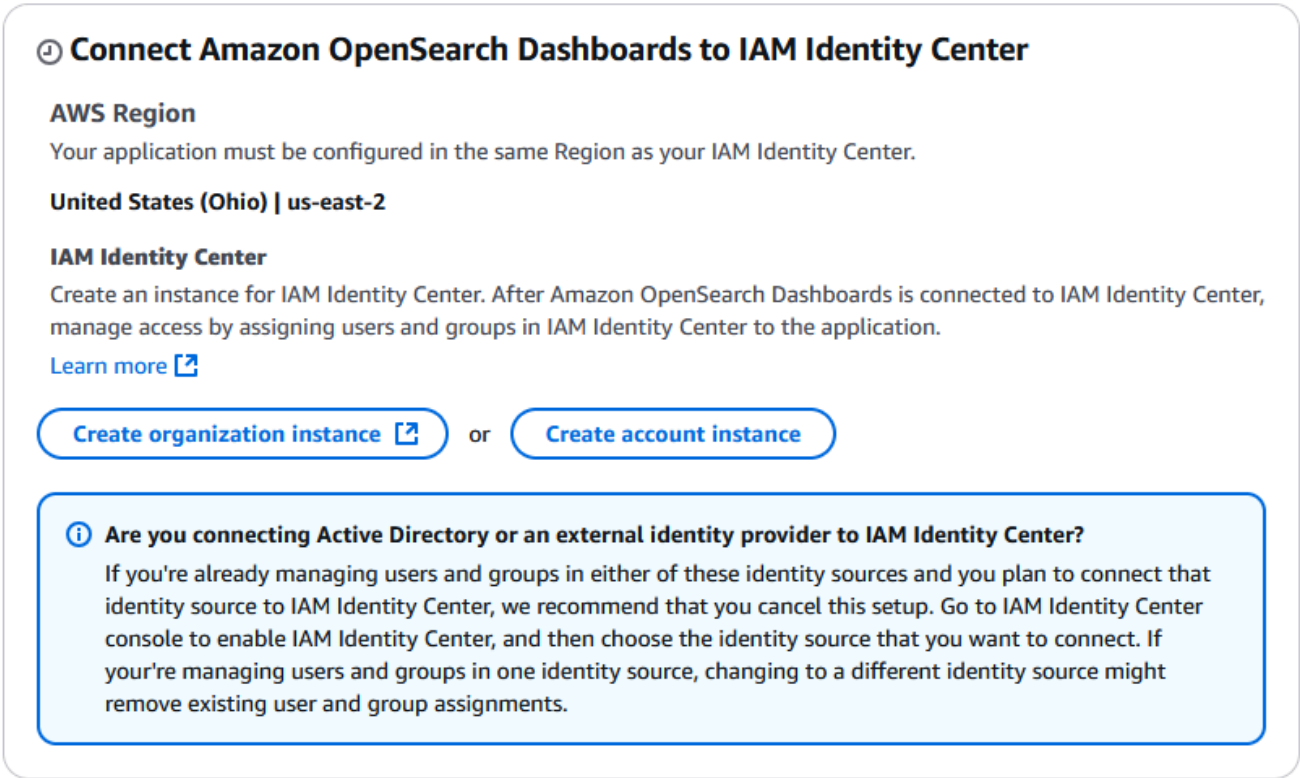
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":
 "application.opensearchservice.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service":
 "application.opensearchservice.amazonaws.com"
            },
            "Action": "sts:SetContext",
            "Condition": {
                "ForAllValues:ArnEquals": {
                "sts:RequestContextProviders":
 "arn:aws:iam::123456789012:oidc-provider/portal.sso.us-
east-1.amazonaws.com/apl/application-id"
                }
```

```
            }
        }
    ]
}
```

c. 組織またはアカウントに IAM Identity Center インスタンスがすでに作成されている場合、次の図に示すように、コンソールは Amazon OpenSearch Dashboards が IAM Identity Center の組織インスタンスに既に接続されていることを報告します。



IAM Identity Center が組織またはアカウントでまだ利用できない場合は、ユーザーまたは必要なアクセス許可を持つ管理者が組織インスタンスまたはアカウントインスタンスを作成できます。[Amazon OpenSearch Dashboards を IAM アイデンティティセンターに接続する] エリアには、次の図に示すように、両方のオプションがあります。

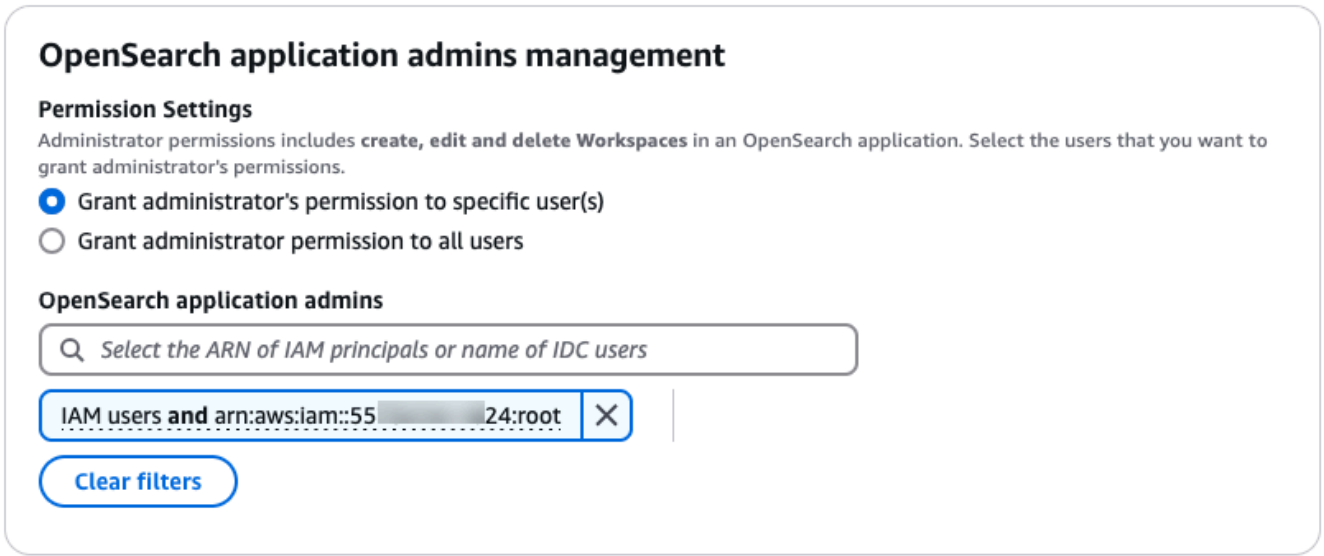


この場合、IAM アイデンティティセンターでテスト用のアカウントインスタンスを作成するか、管理者が IAM アイデンティティセンターで組織インスタンスを作成するようにリクエストできます。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の以下のトピックを参照してください。

> **Note**
>
> IAM アイデンティティセンターの組織インスタンス AWS リージョン とは異なるで OpenSearch UI アプリケーションを作成する場合は、「複数の で IAM アイデンティティセンターを使用する AWS リージョン」を参照してください。

- IAM Identity Center の組織インスタンス
- IAM アイデンティティセンターのアカウントインスタンス
- 有効化 AWS IAM アイデンティティセンター

6. (オプション) 作成するアプリケーションの管理者として自動的に追加されます。[OpenSearch アプリケーションの管理者管理] エリアでは、次の画像に示すように、管理者権限を他のユーザーに付与できます。



> **Note**
>
> OpenSearch UI アプリケーション管理者ロールは、OpenSearch UI アプリケーションを編集および削除する権限を付与します。アプリケーション管理者は、OpenSearch UI アプリケーションでワークスペースを作成、編集、削除することもできます。

管理者権限を他のユーザーに付与するには、次のいずれかを選択します。

- [特定のユーザー (複数可) に管理者の権限を付与] – [OpenSearch アプリケーション管理者] フィールドの [プロパティ] ポップアップリストで、[IAM ユーザー] または

  AWS IAM アイデンティティセンター ユーザーを選択し、管理者権限を付与する個々のユーザーを選択します。
- [すべてのユーザーに管理者権限を付与] – 組織またはアカウント内のすべてのユーザーに管理者権限が付与されます。

7. (オプション) [タグ] エリアで、1 つ以上のタグキーの名前と値のペアをアプリケーションに適用します。

   タグは、リソースに割り当てるオプションのメタデータです。タグを使用すると、目的、所有者、環境などのさまざまな方法でリソースを分類できます。
8. [作成] を選択します。

を使用した AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションの作成 AWS CLI

を使用して AWS IAM アイデンティティセンター 認証を使用する OpenSearch UI アプリケーションを作成するには AWS CLI、以下のオプションを指定して create-application コマンドを使用します。

- `--name` – アプリケーションの名前。
- `--iam-identity-center-options` – (オプション) OpenSearch が認証とアクセスコントロールに使用する IAM Identity Center インスタンスと IAM ロール。

`#########`を、ユーザー自身の情報に置き換えます。

```
aws opensearch create-application \
```

```
    --name application-name \
    --iam-identity-center-options "
          {
          \"enabled\":true,
          \"iamIdentityCenterInstanceArn\":\"arn:aws:sso:::instance/sso-instance\",
          \"iamRoleForIdentityCenterApplicationArn\":\"arn:aws:iam::account-
id:role/role-name\"
          }
    "
```

アプリケーション管理者の管理

OpenSearch UI アプリケーション管理者は、OpenSearch UI アプリケーションを編集および削除できる権限を持つ定義されたロールです。

デフォルトでは、OpenSearch UI アプリケーションの作成者は、OpenSearch UI アプリケーションの最初の管理者です。

コンソールを使用した OpenSearch UI 管理者の管理

AWS マネジメントコンソールの OpenSearch UI アプリケーションには、アプリケーションの作成ワークフロー中またはアプリケーションの作成後の [編集] ページで、管理者を追加できます。

OpenSearch UI アプリケーション管理者ロールは、OpenSearch UI アプリケーションを編集および削除する権限を付与します。アプリケーション管理者は、OpenSearch UI アプリケーションでワークスペースを作成、編集、削除することもできます。

アプリケーションの詳細ページで、IAM プリンシパルの Amazon リソースネーム (ARN) を検索するか、IAM Identity Center のユーザーの名前を検索できます。

コンソールを使用して OpenSearch UI 管理者を管理するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインの [OpenSearch UI (ダッシュボード)] を選択します。
3. [OpenSearch アプリケーション] エリアで、既存のアプリケーションの名前を選択します。
4. [編集] を選択します。
5. 管理者権限を他のユーザーに付与するには、次のいずれかを選択します。

- [特定のユーザー (複数可) に管理者の権限を付与] – [OpenSearch アプリケーション管理者] フィールドの [プロパティ] ポップアップリストで、[IAM ユーザー] または

  AWS IAM アイデンティティセンター ユーザーを選択し、管理者権限を付与する個々のユーザーを選択します。
- [すべてのユーザーに管理者権限を付与] – 組織またはアカウント内のすべてのユーザーに管理者権限が付与されます。

6. [更新] を選択します。

追加の管理者を削除できますが、各 OpenSearch UI アプリケーションは少なくとも 1 人の管理者を保持する必要があります。

を使用した OpenSearch UI 管理者の管理 AWS CLI

AWS CLIを使用して、OpenSearch UI アプリケーション管理者を作成および更新できます。

を使用した OpenSearch UI 管理者の作成 AWS CLI

OpenSearch UI アプリケーションを作成するときに、IAM プリンシパルと IAM Identity Center ユーザーを管理者として追加する例を次に示します。

例 1: IAM ユーザーを管理者として追加する OpenSearch UI アプリケーションを作成する

次のコマンドを実行して、IAM ユーザーを管理者として追加する OpenSearch UI アプリケーションを作成します。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch create-application \
    --name application-name \
    --app-configs "
        {
        \"key\":\"opensearchDashboards.dashboardAdmin.users\",
        \"value\":\"arn:aws:iam::account-id:user/user-id\"
        }
    "
```

例 2: IAM Identity Center を有効にし、OpenSearch Identity Center ユーザー ID を OpenSearch UI アプリケーション管理者として追加する OpenSearch UI アプリケーションを作成する

次のコマンドを実行して、IAM Identity Center を有効にし、IAM Identity Center ユーザー ID を OpenSearch UI アプリケーション管理者として追加する OpenSearch UI アプリケーションを作成します。*#########*を、ユーザー自身の情報に置き換えます。

key は、OpenSearch UI アプリケーションの管理者ロールなど、設定する構成項目を指定します。有効な値は、opensearchDashboards.dashboardAdmin.users および opensearchDashboards.dashboardAdmin.groups です。

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx は、IAM ユーザーの Amazon リソースネーム (ARN) など、キーに割り当てられた値を表します。

```
aws opensearch create-application \
    --name myapplication \
    --iam-identity-center-options "
        {
        \"enabled\":true,
        \"iamIdentityCenterInstanceArn\":\"arn:aws:sso:::instance/ssoins-instance-id\",
        \"iamRoleForIdentityCenterApplicationArn\":\"arn:aws:iam::account-id:role/role-
name\"
        }
    " \
    --app-configs "
        {
        \"key\":\"opensearchDashboards.dashboardAdmin.users\",
        \"value\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"
        }
    "
```

を使用した OpenSearch UI 管理者の更新 AWS CLI

以下は、既存の OpenSearch アプリケーションの管理者として割り当てられた IAM プリンシパルと IAM Identity Center ユーザーを更新する例です。

例 1: 既存の OpenSearch アプリケーションの管理者として IAM ユーザーを追加する

次のコマンドを実行して OpenSearch UI アプリケーションを更新し、IAM ユーザーを管理者として追加します。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch update-application \
```

```
    --id myapplication \
    --app-configs "
        {
        \"key\":\"opensearchDashboards.dashboardAdmin.users\",
        \"value\":\"arn:aws:iam::account-id:user/user-id\"
        }
    "
```

例 2: OpenSearch UI アプリケーションを更新して、IAM Identity Center ユーザー ID を OpenSearch UI アプリケーション管理者として追加する

次のコマンドを実行して、OpenSearch UI アプリケーションを更新し、IAM Identity Center ユーザー ID を OpenSearch UI アプリケーション管理者として追加します。*#########*を、ユーザー自身の情報に置き換えます。

key は、OpenSearch UI アプリケーションの管理者ロールなど、設定する構成項目を指定します。有効な値は、opensearchDashboards.dashboardAdmin.users および opensearchDashboards.dashboardAdmin.groups です。

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx は、IAM ユーザーの Amazon リソースネーム (ARN) など、キーに割り当てられた値を表します。

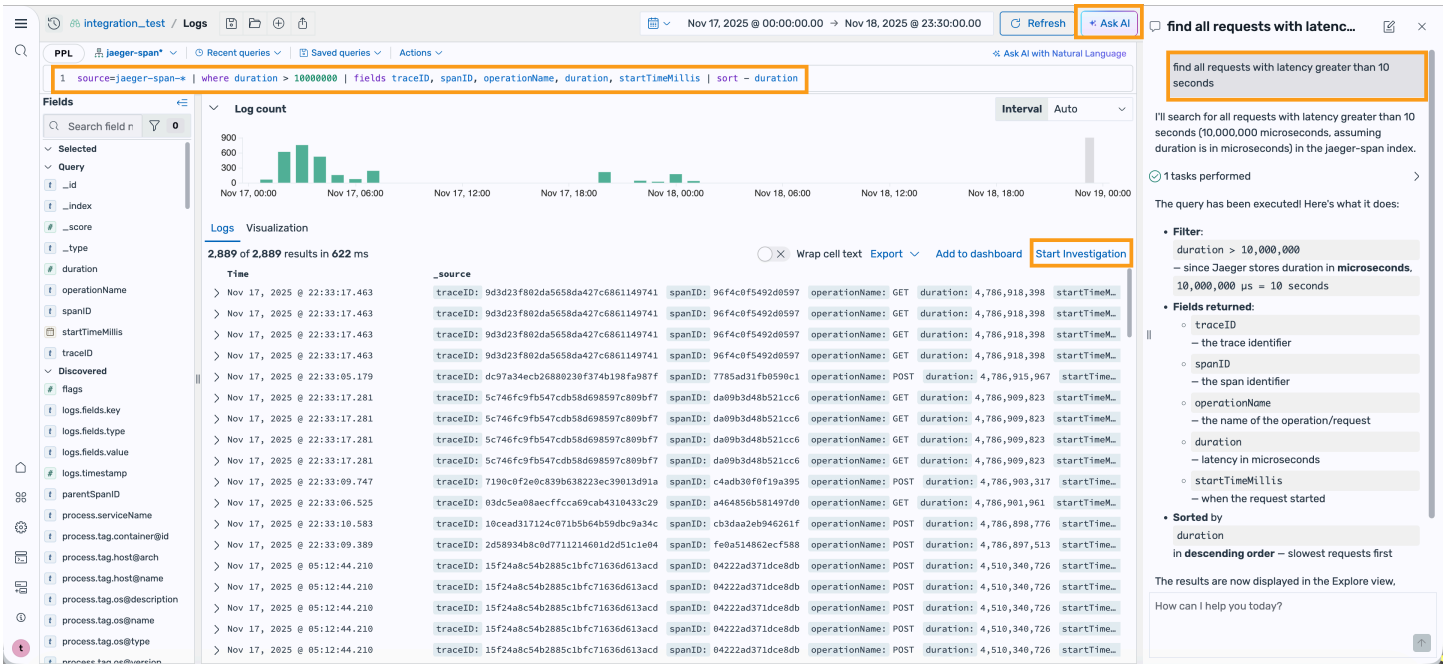
```
aws opensearch update-application \
    --id myapplication \
    --app-configs "
        {
        \"key\":\"opensearchDashboards.dashboardAdmin.users\",
        \"value\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"
        }
    "
```

Amazon OpenSearch Service のエージェント AI

OpenSearch UI には、運用分析を簡素化し、エンジニアリングチームとサポートチームのインシデント調査を加速するために設計された一連のエージェント AI 機能が含まれています。

OpenSearch UI アプリケーションのすべてのページで、AI に尋ねるボタンを選択してエージェントチャットを開きます。エージェントチャットは、表示しているページのコンテキストを理解し、エージェントツールを使用して基盤となるデータを分析し、クエリを更新して Discover で結果を表示

し、質問に明確かつ正確に回答します。複雑な根本原因分析でより詳細な調査が必要な場合は、調査エージェントをトリガーして、複数ステップのワークフローを通じて自律的に計画、実行、および反映し、データ証拠に基づく構造化された仮説を提供できます。両方のエージェント機能を強化するのは、会話や調査内のコンテキストを保持するメモリレイヤーであるエージェントメモリです。これにより、さまざまなウェブセッションにわたって同じ一貫したコンテキストでフォローアップの質問をし続けることができます。



主な機能

- エージェントチャット – OpenSearch UI のすべてのページの AI 機能に埋め込まれているエージェントチャットは、現在のコンテキストを理解し、データを分析して質問に最もよく答えるためのツールを呼び出します。Discover で、チャットインターフェイスに自然言語を入力して Piped Processing Language (PPL) クエリを生成して反復処理するため、実用的なインサイトを得るために PPL の専門家である必要はありません。エージェントチャットがグラフ内の異常を識別し、基盤となるデータと相関させ、分析を生成できる視覚化から直接会話を開始することもできます。複雑な根本原因分析が必要な場合は、`/investigate`スラッシュコマンドを使用して、進行中の会話または新しい指示に基づいて調査エージェントを起動します。
- 調査エージェント – チャット`/investigate`で入力するか、機能ページで調査の開始ボタンを選択してアクティブ化する、目標駆動型の深層調査エージェント。「このレイテンシースパイクの根本原因を見つける」などの調査目標を入力します。調査エージェントは、データと指定された目標を使用して自律的に計画し、クエリと分析を実行し、複数ステップのワークフローを通じて反映します。調査が完了すると、通常は数分以内に、データ証拠に基づいて、可能性別にランク付けされ

た構造化された仮説が生成されます。推論のすべてのステップを完全に透明化できるため、結果を検証して信頼できます。

- エージェントメモリ – 会話や調査内のコンテキストを保持するサービスマネージドメモリレイヤー。一貫したコンテキスト内で質問や調査を継続できます。エージェントメモリを使用すると、エージェントチャットと調査エージェントの両方がセッション全体の継続性を維持できます。

料金と使用制限

エージェント AI 機能は無料で使用できます。トークンベースの使用制限は、不正使用を防ぐためにアカウントごとに適用されます。

エージェント AI 機能は、Amazon Bedrock の基盤モデルを使用します。すべての Amazon Bedrock セキュリティおよびデータガバナンス設定は、エージェント AI 機能に適用されます。詳細については、「Amazon Bedrock でのデータ保護」を参照してください。

セキュリティ

エージェント AI 機能は、既存のセキュリティフレームワーク内で動作し、設定されたアクセスコントロールをバイパスまたは上書きしません。エージェントチャットを操作するか、調査エージェントを起動すると、AI 機能は現在の IAM または IAM アイデンティティセンターの認証情報を使用して認証し、関連するすべてのアクセス許可を継承します。AI アシスタントは、既存のロールベースのアクセスコントロール (RBAC) ポリシーを通じて既に明示的にアクセスできるデータソース、インデックス、視覚化のみをクエリできます。

OpenSearch ドメインがフィールドレベルまたはドキュメントレベルのセキュリティを備えたきめ細かなアクセスコントロールを実装している場合、AI はこれらの制限を尊重し、アクセス許可の範囲外の制限されたフィールドまたはドキュメントのデータを表示することはできません。同様に、エージェントチャットによって生成された PPL クエリや調査中に実行されたデータ取り出しは、ユーザーコンテキストで実行され、設定されたすべてのセキュリティポリシーが完全に適用されるようにします。

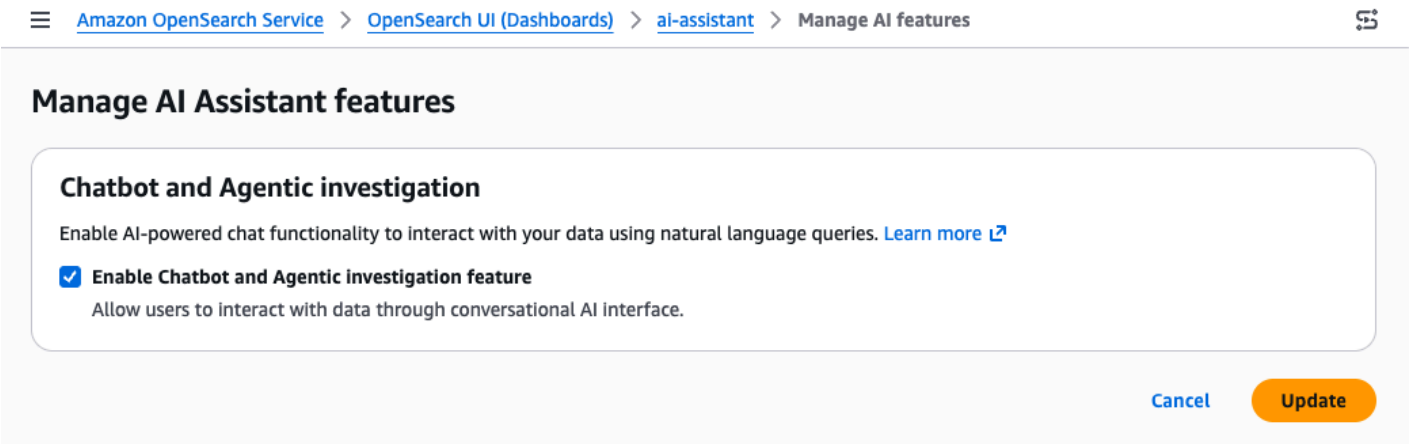
このアクセス許可継承モデルにより、AI 機能を有効にしても、新しいセキュリティの脆弱性は発生せず、管理者は AI インタラクション用に個別のアクセス許可構造を管理する必要もありません。

エージェント AI 機能の有効化と無効化

エージェント AI 機能は、OpenSearch ドメインのいずれかで AI 機能を明示的に無効にしない限り、新しい OpenSearch UI アプリケーションと既存のアプリケーションに対して自動的に有効になります。

AI 機能を有効または無効にするには、 AWS マネジメントコンソール または API を使用できます。

- コンソール – OpenSearch UI アプリケーションの詳細ページに移動し、そこから AI 機能を更新します。



- API – RegisterCapability API を使用して AI 機能を有効にするか、DeregisterCapability API を使用して無効にします。

リージョンの可用性

エージェント AI の機能は次のとおりです AWS リージョン。

- アジアパシフィック (東京) – ap-northeast-1
- アジアパシフィック (シドニー) – ap-southeast-2
- 欧州 (フランクフルト) – eu-central-1
- 欧州 (ストックホルム) - eu-north-1
- 欧州 (スペイン) eu-south-2
- 欧州 (アイルランド) – eu-west-1
- 米国東部 (バージニア北部) – us-east-1
- 米国東部 (オハイオ) – us-east-2
- 米国西部 (オレゴン) – us-west-2

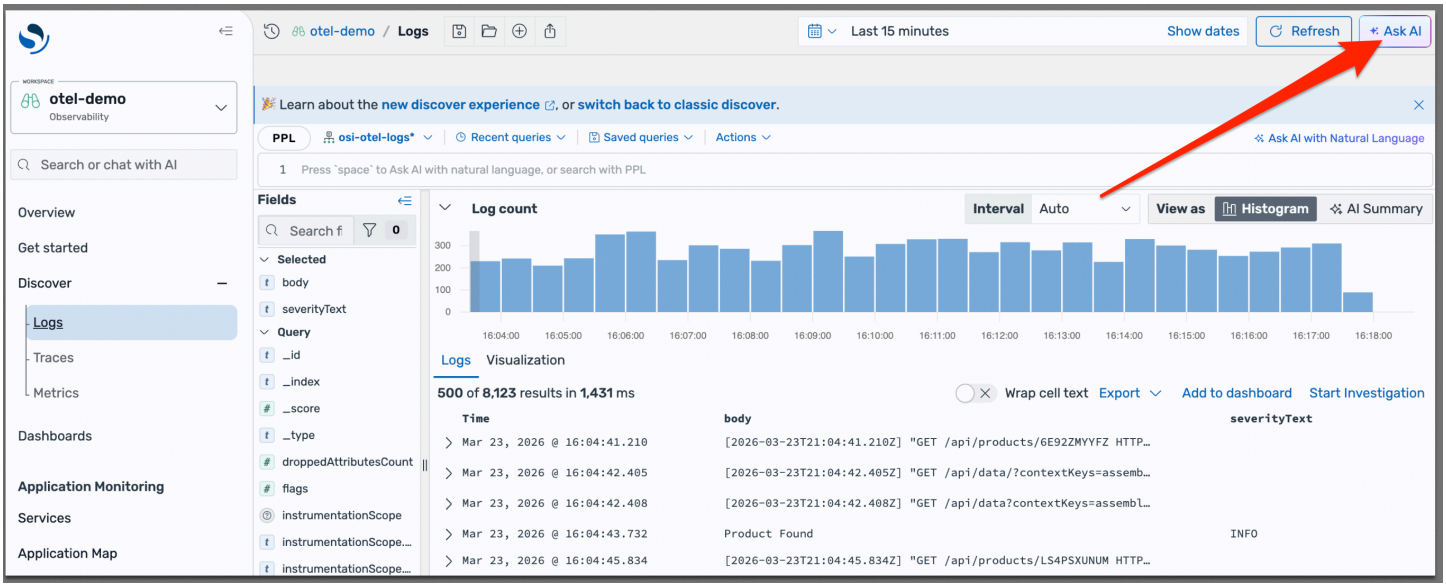
トピック

- Amazon OpenSearch Service のエージェントチャット
- Amazon OpenSearch Service の調査エージェント

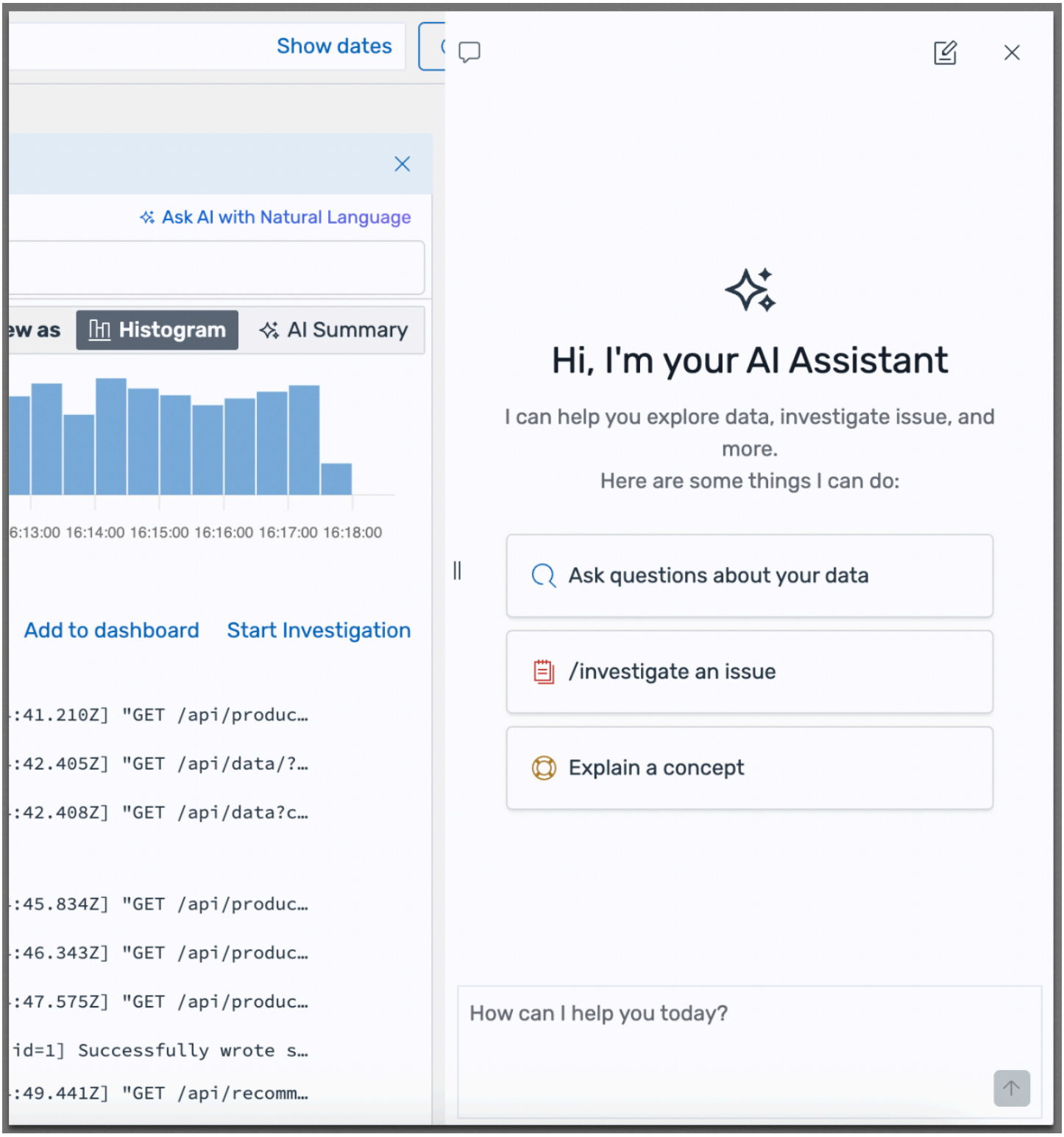
- Amazon OpenSearch Service のエージェントメモリ

Amazon OpenSearch Service のエージェントチャット

エージェントチャットは、OpenSearch UI のすべてのページに埋め込まれた AI アシスタントです。Ask AI ボタンを選択してチャットパネルを開き、データに関する質問、クエリの生成、調査の開始を行うことができます。エージェントチャットは、検出と調査で表示しているページのコンテキストを理解し、エージェントツールを使用して基盤となるデータを分析します。



チャットパネルを開くと、エージェントチャットには、データに関する質問、問題の調査、概念の説明など、開始に役立つオプションが表示されます。以前に会話を開始した場合は、ページ間を移動するときにチャットパネルに表示されたままになるため、中断した場所を続行できます。または、右上隅の新しいチャットボタンを選択して、新しい会話を開始します。



Discover でのエージェントチャットの使用

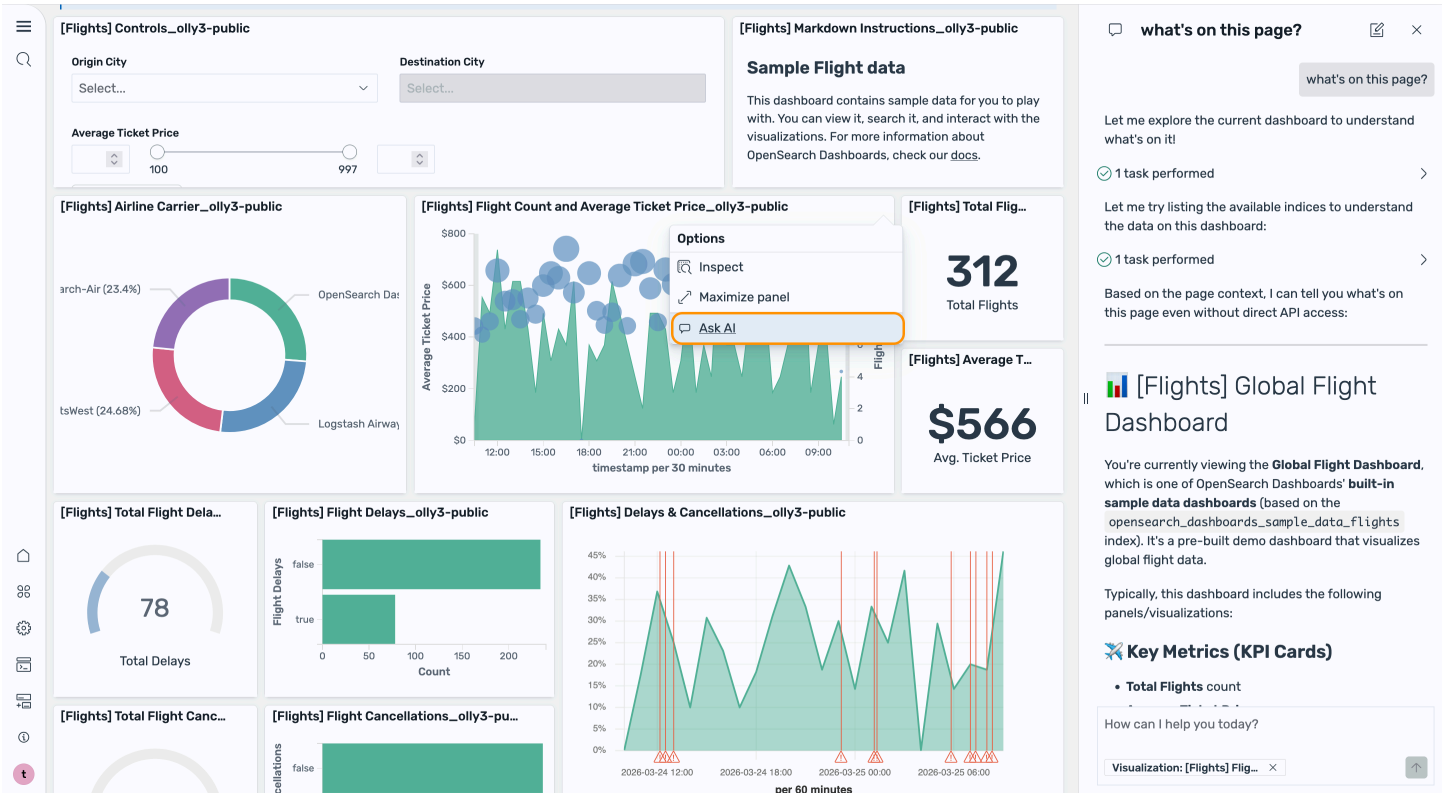
オブザーバビリティワークスペースの検出ページで、チャットインターフェイスに自然言語を入力して PPL クエリを生成できます。エージェントチャットは質問を PPL に変換し、クエリを実行し、結

果を検出ビューに直接表示します。データから実用的なインサイトを得るために、PPL の専門家である必要はありません。

生成されたクエリを絞り込むには、「ステータスコード 500 のフィルターを追加する」など、自然言語でフォローアップの質問をします。エージェントチャットは、現在のクエリのコンテキストを理解し、それに応じて変更します。集計の調整、時間範囲の変更、結果へのフィールドの追加を に依頼することもできます。反復するたびに、Discover ビューが新しいクエリ結果で更新されます。

エージェントチャットとビジュアライゼーションの使用

エージェントチャットとの会話は、視覚化から直接開始できます。視覚化パネルのコンテキストメニューを開き、AI を尋ねるを選択します。エージェントチャットは、視覚化を分析し、グラフ内の異常を識別し、基盤となるデータと関連付けて、分析を生成します。



チャットから調査を開始する

複雑な根本原因分析が必要な場合は、エージェントチャットから直接調査エージェントを起動できます。チャット入力で`/investigate`スラッシュコマンドを使用するか、機能ページで調査の開始ボタンを選択します。

調査エージェントの詳細については、「」を参照してくださいthe section called “調査エージェント”。

サポートされているツール

エージェントチャットは、以下のツールを使用してデータを分析し、質問に回答します。利用可能なツールの最新up-to-dateリストを表示するには、チャットインターフェイスで「使用できるツール」と入力します。

フロントエンドツール

これらのツールは OpenSearch UI を更新します。

- `create_investigation` – 目標、症状、インデックス、時間範囲などの詳細を含む新しいエージェント調査ノートブックを作成します。
- `execute_ppl_query` – 現在のデータセットに対して PPL クエリを実行し、検出ページに結果を表示します。
- `update_time_range` – 現在の検出ページのグローバル時間範囲フィルターを更新します (例：「過去 24 時間」または「先週」) 。

バックエンドツール

これらのツールは OpenSearch データおよび APIs。

- `SearchIndexTool` – DSL クエリを使用してインデックスを検索します。
- `MsearchTool` – 1 つのリクエストで複数の検索オペレーションを実行します。
- `CountTool` – クエリに一致するドキュメントの数を返します。
- `ExplainTool` – ドキュメントがクエリと一致するか一致しない理由について説明します。
- `ListIndexTool` – クラスター内のインデックスをオプションの詳細とともに一覧表示します。
- `IndexMappingTool` – インデックスマッピングと設定を取得します。
- `GetShardsTool` – インデックスのシャード情報を取得します。
- `ClusterHealthTool` – クラスターのヘルス情報を返します。
- `LogPatternAnalysisTool` – ログパターンを分析し、時間範囲を比較し、トレースシーケンス分析を実行します。
- `MetricChangeAnalysisTool` – 2 つの時間範囲間の数値フィールドのパーセンタイル分布を比較します。
- `DataDistributionTool` – オプションでベースラインと比較して、ターゲット時間範囲内のフィールド値の分布を分析します。

- `GenericOpenSearchApiTool` – OpenSearch API エンドポイントを直接呼び出すための柔軟なツールです。

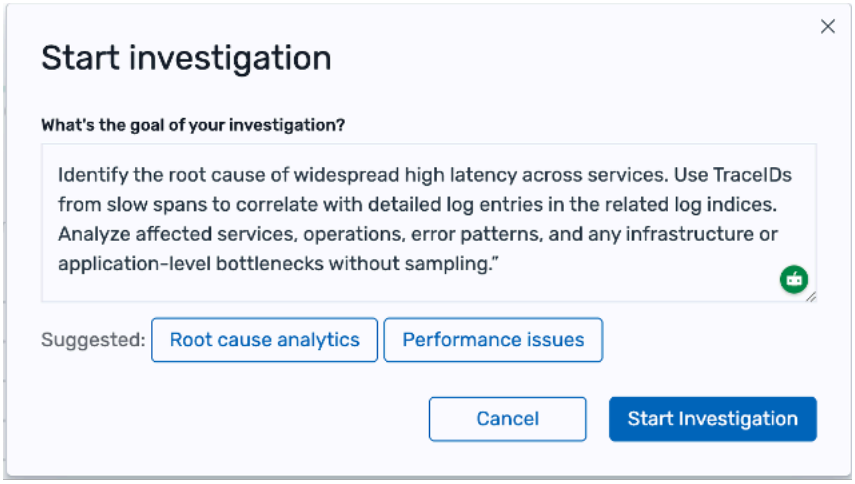
Amazon OpenSearch Service の調査エージェント

調査エージェントは、OpenSearch UI の目標駆動型の深層調査エージェントであり、ユーザーに代わって複雑な問題を自律的に調査します。データおよび指定された目標を使用して計画し、クエリと分析を実行し、複数ステップのワークフローを通じて反映します。調査が完了すると、通常は数分以内に、データ証拠に基づいて、可能性別にランク付けされた構造化された仮説が生成されます。推論のすべてのステップを完全に透明化できるため、結果を検証して信頼できます。

調査の開始

調査は、次の 2 つの方法で開始できます。

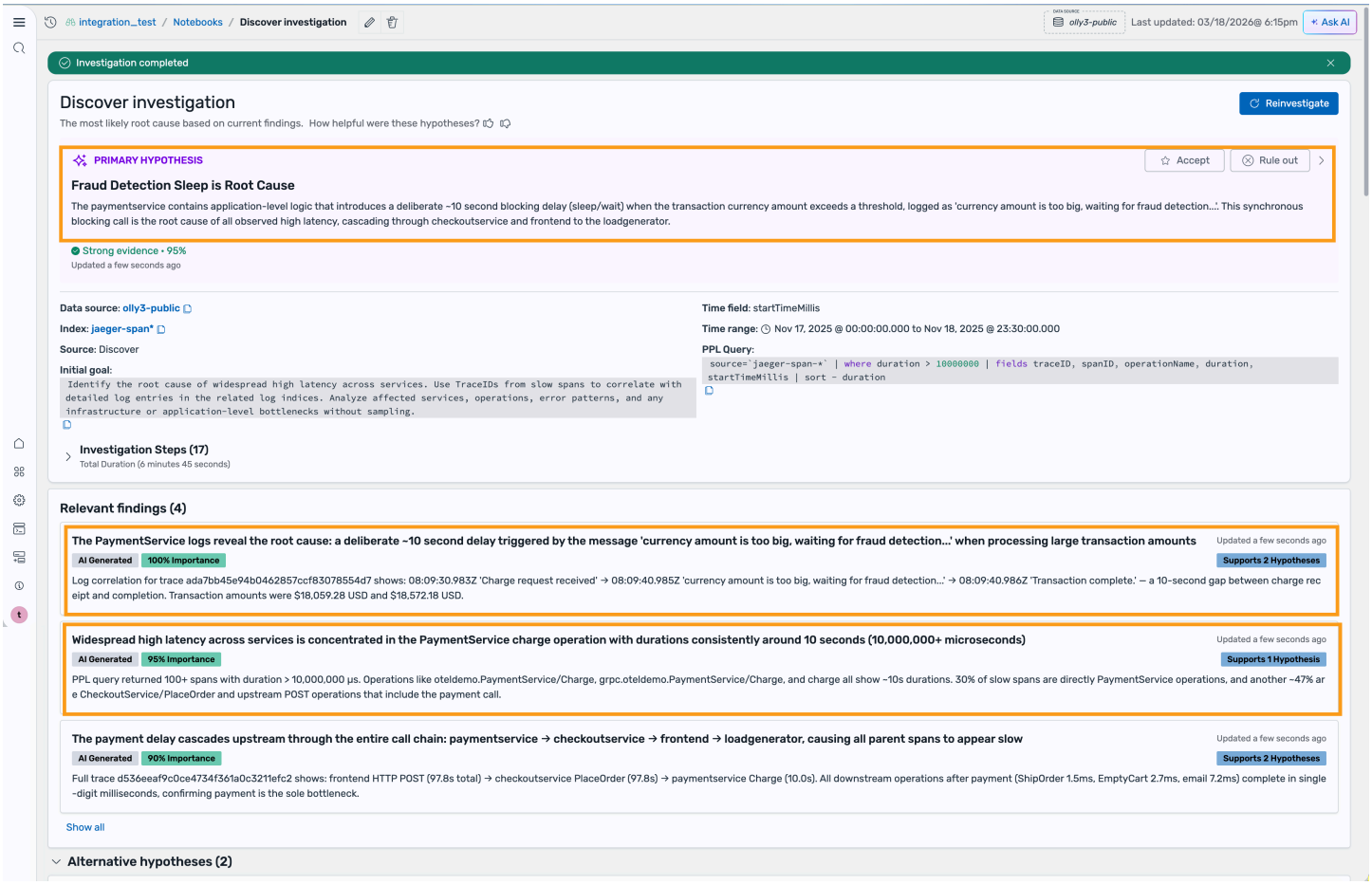
- 検出、視覚化、またはその他のサポートされている機能ページから、調査の開始ボタンを選択します。調査目標を入力し、「根本原因分析」や「パフォーマンスの問題」などの推奨テンプレートから選択できるダイアログが表示されます。
- エージェントチャットから、調査目標を含むチャット入力に`/investigate`スラッシュコマンドを入力します。



エージェントチャットの詳細については、「」を参照してくださいthe section called “エージェントチャット”。

調査結果の確認

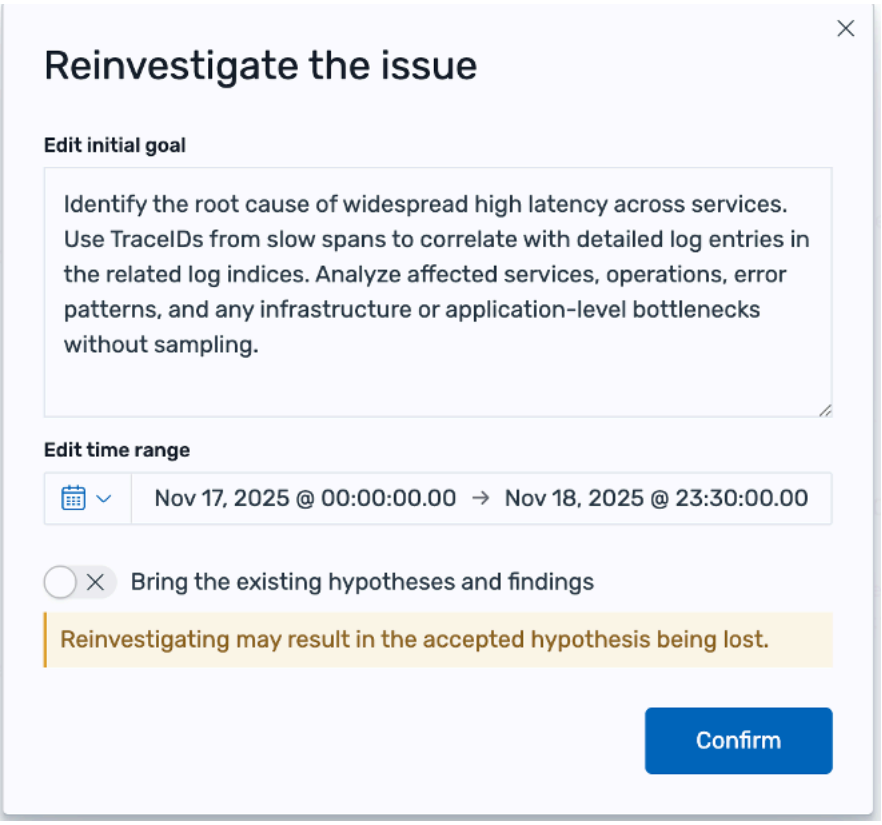
調査が完了すると、調査エージェントは信頼度と裏付けとなる証拠を含む主要な仮説を提示します。結果ページには、実行された調査手順、重要度別にランク付けされた関連する結果、代替仮説が表示されます。



各仮説の背後にある結果を確認し、Accept を選択して仮説を確認するか、Rule out を選択して拒否することができます。可能性の低い代替仮説もレビューできます。評価により一致すれば、最終結論として代替仮説を選択できます。

再調査

調査結果をより明確にする必要がある場合、または調査エージェントが調査の質問に回答できないと判断した場合は、再調査オプションを使用して調査を調整および再実行できます。再調査を選択して最初の目標を編集し、時間範囲を調整し、オプションで既存の仮説と結果を新しい調査に取り込みます。



Amazon OpenSearch Service のエージェントメモリ

エージェントメモリは、エージェントチャットと調査エージェントを強化するサービスマネージドメモリレイヤーです。会話や調査にコンテキストを保持するため、さまざまな機能ページ、ブラウザタブ、ページの更新で一貫したエクスペリエンスが得られます。エージェントメモリは自動的に動作し、ユーザー設定は必要ありません。

エージェントメモリは、OpenSearch エージェントメモリフレームワーク上に構築されています。メモリストレージは、プライバシーのためにユーザー ID によって分離されます。

データ保護

エージェントメモリは無料で使用できます。エージェントメモリに保存されている顧客データは、サービスマネージドキーで暗号化されます。OpenSearch UI アプリケーションのカスタマーマネージドキー (CMK) 暗号化を有効にした場合、メモリデータは代わりに CMK で暗号化されます。メモリは、サービス管理の Amazon OpenSearch Serverless コレクションに保存されます。

CMK 暗号化の詳細については、「」を参照してくださいthe section called “暗号化とカスタマーマネージドキー”。

制限事項

エージェントメモリは、異なる会話スレッド間でコンテキストを保持することはできません。

カスタマーマネージドキーによる OpenSearch UI アプリケーションメタデータの暗号化

ビジュアルアセットと設定は、OpenSearch UI アプリケーションのメタデータとして保存されます。これには、保存されたクエリ、視覚化、ダッシュボードが含まれます。関連付けられたデータソースからのデータはメタデータに保存されません。データソース内のデータの暗号化の詳細については、「Amazon OpenSearch Service for OpenSearch ドメインでのデータ保護」および「サーバーレスコレクションでの Amazon OpenSearch Serverless での暗号化」を参照してください。
OpenSearch

OpenSearch UI メタデータは、保管時の暗号化で保護されます。これにより、不正アクセスが防止されます。暗号化では、 AWS Key Management Service (AWS KMS) を使用して暗号化キーを保存および管理します。デフォルトでは、OpenSearch UI メタデータは AWS 所有キーで暗号化されます。

カスタマーマネージドキー (CMK) 機能を使用して、独自の暗号化キーを管理することもできます。これにより、規制とコンプライアンスの要件を満たすことができます。CMK を使用するには、新しい OpenSearch UI アプリケーションを作成し、作成プロセスで CMK を有効にする必要があります。現在、既存の OpenSearch UI アプリケーションを AWS 所有キーから CMK に更新することはサポートされていません。

カスタマーマネージドキーを使用するタイミング:

- 組織にキー管理に関する規制コンプライアンス要件がある
- 暗号化キーの使用には監査証跡が必要です
- キーローテーションスケジュールを制御する場合
- 既存のキー管理ワークフローと統合する必要があります

カスタマーマネージドキーを使用すると、キーを完全に制御できます。これには、次の機能が含まれます。

- キーポリシーの確立と維持
- IAM ポリシーとグラントの確立と維持

- キーを有効または無効にする
- キーの暗号化マテリアルをローテーションする
- キーにタグを追加する
- キーエイリアスの作成
- キーの削除をスケジュールする

> **Note**
>
> カスタマーマネージドキーは、OpenSearch UI アプリケーション AWS リージョン と同じにある必要があります。別のリージョンのキーを使用することはできません。

トピック

- カスタマーマネージドキーを使用するための前提条件
- コンソールを使用したカスタマーマネージドキー暗号化を使用したアプリケーションの作成
- を使用したカスタマーマネージドキー暗号化によるアプリケーションの作成 AWS CLI
- カスタマーマネージドキーの使用状況のモニタリング
- 暗号化設定の更新

カスタマーマネージドキーを使用するための前提条件

カスタマーマネージドキーを使用して OpenSearch UI アプリケーションメタデータを暗号化する前に、 で対称暗号化キーを作成する必要があります AWS KMS。キーの作成手順については、「 AWS KMS デベロッパーガイド」の「キーの作成」を参照してください。

カスタマーマネージドキーのキーポリシーは、キーを使用するアクセス許可を OpenSearch UI に付与する必要があります。次のキーポリシーを使用して、##########独自の情報に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOpenSearchUIToUseKey",
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "application.opensearchservice.amazonaws.com"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowKeyAdministration",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

ポリシーには、2 つのステートメントが含まれています。

- 最初のステートメントでは、OpenSearch UI が暗号化オペレーションに キーを使用することを許可します。
- 2 番目のステートメントでは、 のユーザーがキーを管理 AWS アカウント できるようにします。これには、キーポリシーの更新、キーの有効化または無効化、キーの削除をスケジュールするアクセス許可が含まれます。ルートプリンシパルを特定の IAM ユーザーまたはロールに置き換えることで、これらのアクセス許可をさらに制限できます。

キーポリシーの詳細については、 AWS KMS デベロッパーガイドの「 でのキーポリシー AWS KMSの使用」を参照してください。

コンソールを使用したカスタマーマネージドキー暗号化を使用したアプリケーションの作成

コンソールで OpenSearch UI アプリケーションを作成するときに、アプリケーションのメタデータを暗号化するためのカスタマーマネージドキーを指定できます。

コンソールを使用してカスタマーマネージドキー暗号化を使用して OpenSearch UI アプリケーションを作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左のナビゲーションペインの [OpenSearch UI (ダッシュボード)] を選択します。
3. [アプリケーションを作成] を選択します。
4. [アプリケーション名] に、そのアプリケーションの名前を入力します。
5. 必要に応じて、認証と管理者の設定を行います。詳細については、「the section called “開始方法”」を参照してください。
6. 「暗号化」セクションの「保管時の暗号化」で、「カスタマーマネージドキーを使用する」を選択します。
7. リストから既存のカスタマーマネージドキーを選択するか、キーの作成を選択して新しいキーを作成します AWS KMS。

   > **Note**
   >
   > キーは、作成するアプリケーション AWS リージョン と同じ にある必要があります。

8. (オプション) アプリケーションにタグを追加します。
9. [作成] を選択します。

を使用したカスタマーマネージドキー暗号化によるアプリケーションの作成 AWS CLI

を使用してカスタマーマネージドキー暗号化を使用して OpenSearch UI アプリケーションを作成するには AWS CLI、 `--kms-key-arn`パラメータを指定して create-application コマンドを使用します。

*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch create-application \
    --name my-application \
    --kms-key-arn arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012
```

`--kms-key-arn` パラメータを指定しない場合、OpenSearch は AWSマネージドキーを使用してアプリケーションのメタデータを暗号化します。

カスタマーマネージドキーの使用状況のモニタリング

OpenSearch UI アプリケーションでカスタマーマネージドキーを使用する場合、 AWS KMS はキーのすべての使用を AWS CloudTrail ログに記録します。これらのログを使用して、キーの使用方法とタイミングをモニタリングできます。ログには、キーにアクセスしたユーザーまたはサービスが表示されます。

AWS AWS KMS は、カスタマーマネージドキーを毎年自動的にローテーションします。必要に応じてキーを手動でローテーションすることもできます。キーローテーションの詳細については、 AWS KMS デベロッパーガイドの「KMS キーのローテーション」を参照してください。

キー使用状況のモニタリングの詳細については、「 AWS KMS デベロッパーガイド」の「 を使用した AWS KMS API コールのログ記録 AWS CloudTrail」を参照してください。

> **Note**
>
> カスタマーマネージドキーを使用すると、 AWS KMS 料金が発生します。料金は、保存された API リクエストとキーの数に基づいています。料金の詳細については、AWS 「 Key Management Service の料金」を参照してください。

暗号化設定の更新

OpenSearch UI アプリケーションを作成した後は、暗号化設定を変更することはできません。別のカスタマーマネージドキーを使用する必要がある場合は、新しいアプリケーションを作成する必要があります。 AWSマネージドキーとカスタマーマネージドキーを切り替える必要がある場合は、必要な暗号化設定で新しいアプリケーションを作成する必要があります。

> **Important**
>
> カスタマーマネージドキーを無効化または削除する前に、次の点を考慮してください。
>
> - キーを無効にすると、アプリケーションは暗号化されたメタデータにアクセスできなくなります。アクセスを復元するには、同じキーを再度有効にする必要があります。

- キーを削除すると、アプリケーションの保存されたオブジェクトに永続的にアクセスできなくなります。これには、クエリ、ビジュアライゼーション、ダッシュボードが含まれます。削除されたキーは復元できません。
- キーステータスを変更する前に、キー ARN を文書化することをお勧めします。

次の手順

アプリケーションの CMK 暗号化を設定したら、次のことができます。

- データソースをアプリケーションに関連付けます。詳細については、「the section called “データソースの関連付けと VPC アクセス許可の管理”」を参照してください。
- チームのワークスペースを作成します。詳細については、「the section called “Amazon OpenSearch Service ワークスペースの使用”」を参照してください。
- キー使用状況 AWS CloudTrail のモニタリングを設定します。詳細については、「the section called “カスタマーマネージドキーの使用状況のモニタリング”」を参照してください。

を使用した SAML フェデレーションの有効化 AWS Identity and Access Management

OpenSearch UI は、多くの ID プロバイダーが使用するオープンスタンダードである Security Assertion Markup Language 2.0 (SAML) をサポートしています。これにより、 AWS Identity and Access Management (IAM) との ID フェデレーションが可能になります。このサポートにより、アカウントまたは組織のユーザーは、IAM ロールを引き受けることで OpenSearch UI に直接アクセスできます。エンドユーザーが外部 ID プロバイダーで認証し、OpenSearch UI の定義されたページに直接ルーティングできる ID プロバイダー主導 (IdP) シングルサインオンエクスペリエンスを作成できます。OpenSearch UI および関連するデータソースにアクセスするためのアクセス許可が異なるさまざまな IAM ロールを引き受けるように、エンドユーザーまたはグループを設定することで、詳細なアクセス制御を実装することもできます。

このトピックでは、OpenSearch UI で SAML を使用するための設定手順を段階的に説明します。これらの手順では、例として Okta アイデンティティおよびアクセス管理アプリケーションを設定する手順を使用します。Azure Active Directory や Ping など、他の ID プロバイダーの設定手順は似ています。

トピック

- ステップ 1: ID プロバイダーアプリケーションを設定する (Okta)
- ステップ 2: Okta AWS の設定をセットアップする
- ステップ 3: IAM で Amazon OpenSearch Service アクセスポリシーを作成する
- ステップ 4: SAML で ID プロバイダーが開始したシングルサインオンエクスペリエンスを検証する
- ステップ 5: SAML 属性ベースの詳細なアクセス制御を設定する

ステップ 1: ID プロバイダーアプリケーションを設定する (Okta)

OpenSearch UI で SAML を使用するには、まず ID プロバイダーを設定します。

タスク 1: Okta ユーザーを作成する

1. 管理者権限を持つユーザーとして、https://login.okta.com/ で Okta 組織にサインインします。
2. 管理者コンソールのナビゲーションペインの [ディレクトリ] で、[ユーザー] を選択します。
3. [Add Person] (ユーザーを追加) を選択します。
4. [名前] には、ユーザーの名前を入力します。
5. [姓] には、ユーザーの姓を入力します。
6. [ユーザー名] には、ユーザーのユーザー名を E メール形式で入力します。
7. [パスワードを設定する] を選択してパスワードを入力します。
8. (オプション) ユーザーが初回サインイン時にパスワードを変更する必要がない場合は、[初回ログイン時にパスワードを変更する必要があります] チェックボックスをオフにします。
9. [保存] を選択します。

タスク 2: グループを作成して割り当てる

1. 管理者権限を持つユーザーとして、https://login.okta.com/ で Okta 組織にサインインします。
2. 管理者コンソールのナビゲーションペインの [ディレクトリ] で、[グループ] を選択します。
3. [グループの追加] を選択します。
4. グループ名を入力し、[保存] を選択します。
5. 新しく作成したグループを選択し、[ユーザーの割り当て] を選択します。
6. プラス記号 ([+]) を選択し、[完了] を選択します。
7. (オプション) ステップ 1～6 を繰り返して、さらにグループを追加します。

タスク 3: Okta アプリケーションを作成する

1. 管理者権限を持つユーザーとして、https://login.okta.com/ で Okta 組織にサインインします。
2. 管理者コンソールのナビゲーションペインの [アプリケーション] で、[アプリケーション] を選択します。
3. [Create App Integration] (アプリの統合の作成) を選択します。
4. サインイン方法として SAML 2.0 を選択し、[次へ] を選択します。
5. アプリ統合用の名前 (**OpenSearch_UI** など) を入力し、[次へ] を選択します。
6. アプリに次の値を入力します。他の値を変更する必要はありません。

   a. 1. シングルサインオン URL の場合は、商用 AWS リージョン**https://signin.aws.amazon.com/saml**に を入力するか、リージョンに固有の URL を入力します。

   b. 2. [オーディエンス URI (SP Entity ID)] には、**urn:amazon:webservices** を入力します。

   c. 3. [名前の形式] には **EmailAddress** を入力します。
7. [次へ] を選択します。
8. [私は Okta の顧客で、内部アプリを追加します] を選択し、次に [これは当社が作成した内部アプリです] を選択します。
9. [Finish] を選択してください。
10. [割り当て] を選択してから、[割り当てる] を選択します。
11. [グループに割り当てる] を選択し、追加するグループの横にある [割り当てる] を選択します。
12. [Done] (完了) をクリックします。

タスク 4: Okta の詳細設定をセットアップする

カスタム SAML アプリケーションを作成したら、次の手順を実行します。

1. 管理者権限を持つユーザーとして、https://login.okta.com/ で Okta 組織にサインインします。

   管理者コンソールの [全般] エリアで、[SAML 設定] で [編集] を選択します。
2. [次へ] を選択します。
3. 次の形式を使用して、[デフォルトのリレーステート] を OpenSearch UI エンドポイントに設定します:

`https://region.console.aws.amazon.com/aos/home?region=region#opensearch/applications/application-id/redirectToDashboardURL`.

以下に例を示します。

```
https://us-east-2.console.aws.amazon.com/aos/home?region=us-east-2#opensearch/applications/abc123def4567EXAMPLE/redirectToDashboardURL
```

4. [属性ステートメント (オプション)] で、以下のプロパティを追加します。

   a. [ロール] 属性を使用して、IAM ロールと ID プロバイダーをカンマ区切り形式で指定します。この同じ IAM ロールと ID プロバイダーは、後のステップで AWS 設定時に使用します。

   b. [RoleSessionName] の [user.login] を設定します。これは、ロールが引き受けられたときに発行される一時的な認証情報の識別子として使用されます。

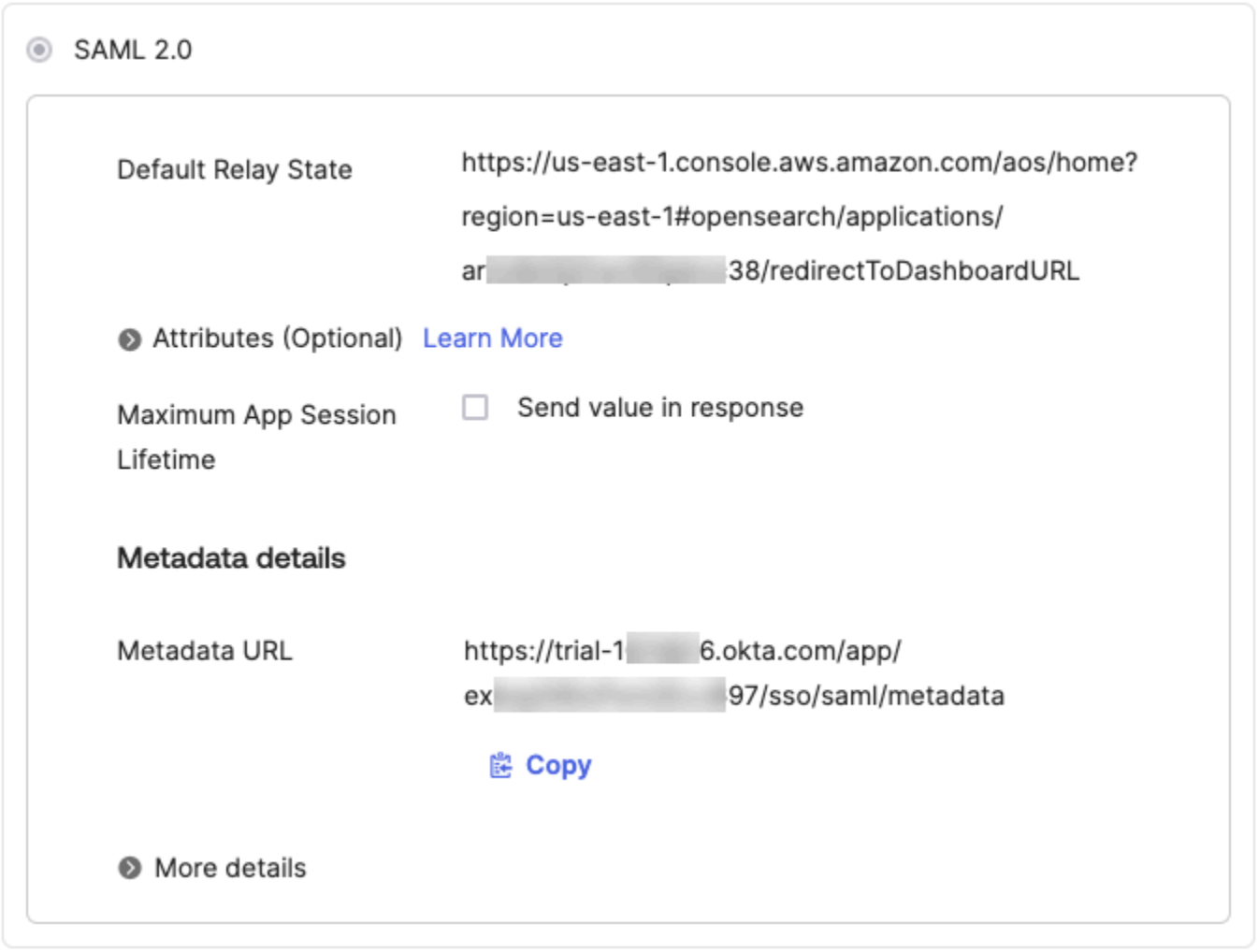
   参考までに:

名前	名前の形式	形式	例
https://aws.amazon.com/SAML/Attributes/Role	未指定	arn:aws:iam:: *aws-account-id* :role/role-name,arn:aws:iam:: *aws-account-id* :saml-provider/ *provider-name*	arn:aws:iam::111222333444:role/oktarole,arn:aws:iam::111222333444:saml-provider/oktaidp
https://aws.amazon.com/SAML/Attribut	未指定	user.login	user.login

名前	名前の形式	形式	例
es/RoleSessionName			

5. 属性プロパティを追加したら、[次へ] を選択し、[終了] を選択します。

属性は、以下の画像に示されているものと同様の形式である必要があります。[デフォルトのリレーステート] 値は、Okta からのシングルサインオン検証を完了した後、アカウントまたは組織のエンドユーザー向けランディングページを定義する URL です。OpenSearch UI の任意のページに設定し、その URL を目的のエンドユーザーに提供できます。



ステップ 2: Okta AWS の設定をセットアップする

Okta AWS の設定を設定するには、次のタスクを実行します。

タスク 1: Okta 情報を収集する

このステップでは、Okta 情報を収集して、後で AWSで設定できるようにする必要があります。

1. 管理者権限を持つユーザーとして、https://login.okta.com/ で Okta 組織にサインインします。
2. [サインオン] タブのページの右下隅で、[SAML セットアップ手順の表示] を選択します。
3. [ID プロバイダーシングルサインオン URL] の値をコピーまたは保存します。この URL は、SQL Workbench/J などのサードパーティー SQL クライアントに接続するときに使用できます。
4. ブロック 4 で ID プロバイダーメタデータを使用し、メタデータファイルを .xml 形式 (例: `metadata.xml`) で保存します。

タスク 2: IAM プロバイダーを作成する

IAM プロバイダーを作成するには、次の手順を実行します。

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインの [アクセス管理] で [ID プロバイダー] を選択します。
3. [プロバイダーを追加] をクリックします。
4. [プロバイダーのタイプ] に、[SAML] を選択します。
5. [プロバイダー名] に名前を入力します。
6. [メタデータドキュメント] で、[ファイルを選択] を選択し、前にダウンロードしたメタデータファイル (.xml) をアップロードします。
7. [プロバイダーを追加] をクリックします。

タスク 3: IAM ロールを作成する

AWS Identity and Access Management ロールを作成するには、次の手順を実行します。

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインの [アクセス管理] で、[ロール] を選択します。
3. [ロールの作成] を選択してください。
4. [信頼されたエンティティタイプ] で、[SAML 2.0 フェデレーション] を選択します。
5. [SAML 2.0 ベースのプロバイダー] として、前に作成した ID プロバイダーを選択します。

6. プログラムによるアクセスと AWS マネジメントコンソール アクセスを許可するを選択します。
7. [次へ] を選択します。
8. アクセス許可ポリシーリストで、 AWS 管理ポリシー AmazonOpenSearchServiceFullAccess OpenSearch Service のアクセス許可を付与するポリシーのチェックボックスをオンにします。
9. [次へ] を選択します。
10. [確認] エリアの [ロール名] に、ロールの名前を入力します (例: **oktarole**)。
11. (オプション) [説明] に、ロールの目的についての簡単な説明を入力してください。
12. [ロールの作成] を選択してください。
13. 先ほど作成したロールに移動し、[信頼関係] タブを選択し、[信頼ポリシーを編集] を選択します。
14. [ステートメントを編集] ペインの 「STS のアクションを追加] で、[TagSession] のボックスを選択します。
15. [ポリシーの更新] を選択してください。

ステップ 3: IAM で Amazon OpenSearch Service アクセスポリシーを作成する

OpenSearch アクセス制御の IAM ロールを設定する方法について説明します。IAM ロールを使用すると、Okta ユーザーグループが OpenSearch リソースにアクセスするための詳細なアクセス制御を実装できます。このトピックでは、2 つのサンプルグループを使用した IAM ロールベースの設定を示します。

Sample group: Alice

リクエスト:

```
GET _plugins/_security/api/roles/alice-group
```

結果:

```
{
  "alice-group": {
    "reserved": false,
    "hidden": false,
    "cluster_permissions": [
```

```
      "unlimited"
    ],
    "index_permissions": [
      {
        "index_patterns": [
          "alice*"
        ],
        "dls": "",
        "fls": [],
        "masked_fields": [],
        "allowed_actions": [
          "indices_all"
        ]
      }
    ],
    "tenant_permissions": [
      {
        "tenant_patterns": [
          "global_tenant"
        ],
        "allowed_actions": [
          "kibana_all_write"
        ]
      }
    ],
    "static": false
  }
}
```

Sample group: Bob

リクエスト:

```
GET _plugins/_security/api/roles/bob-group
```

結果:

```
{
  "bob-group": {
    "reserved": false,
    "hidden": false,
    "cluster_permissions": [
      "unlimited"
```

```
    ],
    "index_permissions": [
      {
        "index_patterns": [
          "bob*"
        ],
        "dls": "",
        "fls": [],
        "masked_fields": [],
        "allowed_actions": [
          "indices_all"
        ]
      }
    ],
    "tenant_permissions": [
      {
        "tenant_patterns": [
          "global_tenant"
        ],
        "allowed_actions": [
          "kibana_all_write"
        ]
      }
    ],
    "static": false
  }
}
```

次の例に示すように、バックエンドロールマッピングを使用して Amazon OpenSearch Service ドメインロールを IAM ロールにマッピングできます。

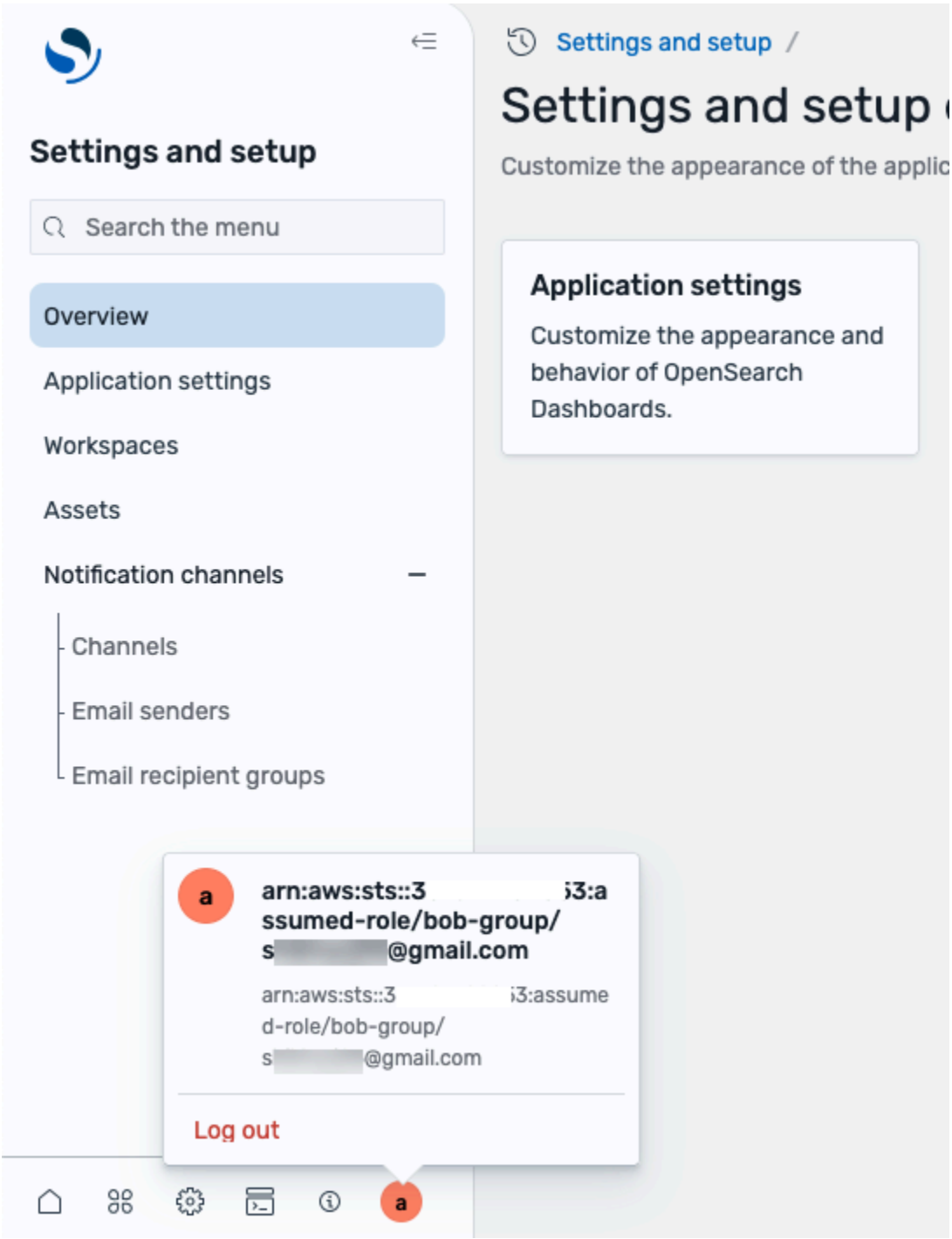
```
{
  "bob-group": {
    "hosts": [],
    "users": [],
    "reserved": false,
    "hidden": false,
    "backend_roles": [
      "arn:aws:iam::111222333444:role/bob-group"
    ],
    "and_backend_roles": []
  },
```

```
  "alice-group": {
    "hosts": [],
    "users": [],
    "reserved": false,
    "hidden": false,
    "backend_roles": [
      "arn:aws:iam::111222333444:role/alice-group"
    ],
    "and_backend_roles": []
  }
}
```

ステップ 4: SAML で ID プロバイダーが開始したシングルサインオンエクスペリエンスを検証する

[デフォルトのリレーステート] URL を開き、Okta 認証ページを開きます。エンドユーザーの認証情報を入力します。OpenSearch UI に自動的にリダイレクトされます。

次の図に示すように、ナビゲーションパネルの下部にあるユーザーアイコンを選択すると、現在の認証情報を確認できます。



ナビゲーションパネルの下部にある開発者ツールにアクセスし、コンソールでクエリを実行することで、ユーザーの詳細なアクセス制御の権限を確認することもできます。次にサンプルクエリを示します:

Example 1: Displays information about the current user

リクエスト:

```
GET _plugins/_security/api/account
```

結果:

```
{
  "user_name": "arn:aws:iam::XXXXXXXXXXXX:role/bob-group",
  "is_reserved": false,
  "is_hidden": false,
  "is_internal_user": false,
  "user_requested_tenant": null,
  "backend_roles": [
    "arn:aws:iam::XXXXXXXXXXXX:role/bob-group"
  ],
  "custom_attribute_names": [],
  "tenants": {
    "global_tenant": true,
    "arn:aws:iam::XXXXXXXXXXXX:role/bob-group": true
  },
  "roles": [
    "bob-group"
  ]
}
```

Example 2: Displays actions permitted for a user

リクエスト:

```
GET bob-test/_search
```

結果:

```
{
  "took": 390,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
```

```
      "value": 1,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "bob-test",
        "_id": "ui01N5UBCIHpjO8Jlvfy",
        "_score": 1,
        "_source": {
          "title": "Your Name",
          "year": "2016"
        }
      }
    ]
  }
}
```

Example 3: Displays actions not permitted for a user

リクエスト:

```
GET alice-test
```

結果:

```
{
  "error": {
    "root_cause": [
      {
        "type": "security_exception",
        "reason": "no permissions for [indices:admin/get]
 and User [name=arn:aws:iam::111222333444:role/bob-group,
 backend_roles=[arn:aws:iam::111222333444:role/bob-group], requestedTenant=null]"
      }
    ],
    "type": "security_exception",
    "reason": "no permissions for [indices:admin/get]
 and User [name=arn:aws:iam::111222333444:role/bob-group,
 backend_roles=[arn:aws:iam::111222333444:role/bob-group], requestedTenant=null]"
  },
  "status": 403
}
```

ステップ 5: SAML 属性ベースの詳細なアクセス制御を設定する

Amazon OpenSearch Service を使用すると、SAML による詳細なアクセス制御を使用して、ID プロバイダーのユーザーとグループを OpenSearch の詳細なアクセス制御のユーザーやロールに動的にマッピングできます。これらのロールを特定の OpenSearch ドメインとサーバーレスコレクションにスコープし、インデックスレベルのアクセス許可とドキュメントレベルのセキュリティを定義できます。

> **Note**
>
> 詳細なアクセス制御の詳細については、「the section called “きめ細かなアクセスコントロール”」を参照してください。

トピック

- 詳細なアクセス制御のための SAML 属性
- タスク 1: 詳細なアクセス制御のために Okta を設定する
- タスク 2: OpenSearch ドメインで SAML を設定する
- タスク 3: OpenSearch Serverless コレクションで SAML を設定する

詳細なアクセス制御のための SAML 属性

subjectKey

E メールやユーザー名などの一意のユーザー属性にマッピングされ、認証するユーザーを識別します。

rolesKey

認可するためのロールまたはアクセス許可を決定する IdP のグループまたはロール属性にマッピングします。

タスク 1: 詳細なアクセス制御のために Okta を設定する

詳細なアクセス制御のために Okta を設定するには

1. [属性ステートメント] セクションに OpenSearch ユーザープリンシパルの新しい属性を追加します。

- 名前: `UserName`
- 値: `${user-email}`

この属性は、認証用の OpenSearch きめ細かなアクセスコントロール設定の [サブジェクトキー] として使用されます。

2. [グループ属性ステートメント] セクションにロール用のグループ属性を追加します:

- 名前: `groups`
- フィルタ: `OpenSearch_xxx`

この属性は、認可するためのグループを OpenSearch の詳細なアクセス制御ロールにマッピングするための [ロールキー] として使用されます。

タスク 2: OpenSearch ドメインで SAML を設定する

OpenSearch ドメインで SAML を設定するには

1. AWS マネジメントコンソールで、OpenSearch UI ユーザーのきめ細かなアクセスコントロールを有効にする OpenSearch Service ドメインを特定します。
2. 特定のドメインの詳細ページに移動します。
3. [セキュリティ設定] タブを選択し、[編集] をクリックします。
4. [IAM フェデレーションを使用する SAML] を拡張します。
5. Okta で定義した `subjectKey` と `roleKey` を入力します。
6. [変更を保存] を選択します。

AWS CLIを使用して詳細なアクセス制御を設定することもできます。

```
aws opensearch create-domain \
--domain-name testDomain \
--engine-version OpenSearch_1.3 \
--cluster-config
 InstanceType=r5.xlarge.search,InstanceCount=1,DedicatedMasterEnabled=false,ZoneAwarenessEnable
 \
```

```
--access-policies '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"*"},"Action":"es:*","Resource":"arn:aws:es:us-
east-1:12345678901:domain/neosaml10/*"}]}' \
--domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
--node-to-node-encryption-options '{"Enabled":true}' \
--encryption-at-rest-options '{"Enabled":true}' \
--advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"********","MasterUserPassword":"********"}, "IAMFederationOptions":
{"Enabled": true,"SubjectKey":"TestSubjectKey","RolesKey":"TestRolesKey"}}' \
--ebs-options "EBSEnabled=true,VolumeType=gp2,VolumeSize=300" \
--no-verify-ssl \
--endpoint-url https://es.us-east-1.amazonaws.com \
--region us-east-1
```

既存のドメインを更新するには:

```
aws opensearch update-domain-config \
--domain-name testDomain \
--advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"********","MasterUserPassword":"********"}, "IAMFederationOptions":
{"Enabled": true,"SubjectKey":"TestSubjectKey","RolesKey":"TestRolesKey"}}' \
--ebs-options "EBSEnabled=true,VolumeType=gp2,VolumeSize=300" \
--no-verify-ssl \
--endpoint-url https://es.us-east-1.amazonaws.com \
--region us-east-1
```

タスク 3: OpenSearch Serverless コレクションで SAML を設定する

OpenSearch Serverless で SAML ベースの詳細なアクセス制御を設定するには

1. を開き AWS マネジメントコンソール 、Amazon OpenSearch Service に移動します。
2. ナビゲーションペインの [サーバーレス] で、[セキュリティ] を選択し、[認証] を選択します。
3. [IAM フェデレーション] セクションで、[編集] を選択します。

   この設定を使用して、SAML 属性ベースの詳細なアクセス制御を制御できます。IAM フェデレーションはデフォルトで無効になっています。
4. [IAM フェデレーションを有効にする] を選択します。
5. Okta で定義した `subjectKey` と `roleKey` の値を入力します。

詳細については、「the section called “詳細なアクセス制御のための SAML 属性”」を参照してください。

6. [Save] (保存) をクリックします。
7. ナビゲーションペインで、[サーバーレス] の [データアクセスポリシー] を選択します。
8. 既存のポリシーを更新するか、新しいポリシーを作成します。
9. ルールを展開し、[プリンシパルを追加] を選択し、[IAM フェデレーションのユーザーとグループ] を選択します。
10. 必要なプリンシパルを追加し、[保存] を選択します。
11. [付与] を選択します。
12. このルールで、次の操作を行います。
    - 選択したプリンシパルに対して定義するアクセス許可を選択します。
    - アクセス許可を適用するコレクションを指定します。
    - 必要に応じて、インデックスレベルのアクセス許可を定義します。

    > **Note**
    >
    > 複数のルールを作成して、異なるプリンシパルグループに異なるアクセス許可を割り当てることができます。

13. 完了したら、[保存] を選択します。
14. [作成] を選択します。

または、CLI を使用して、次に示すようにコレクションのセキュリティ設定を作成することもできます。

```
aws opensearchserverless create-security-config --region "region"  --type iamfederation
 --name "configuration_name" --description "description" --iam-federation-options
 '{"groupAttribute":"GroupKey","userAttribute":"UserKey"}'
```

データソースの関連付けと Virtual Private Cloud アクセス許可の管理

このセクションの手順を使用して、データソースの関連付けを管理し、仮想プライベートクラウド (VPC) に必要なアクセス許可を設定します。

トピック

- データソースと OpenSearch UI アプリケーションの関連付け
- VPC 内のドメインへのアクセスの管理
- VPC 内の OpenSearch Serverless コレクションへのアクセスの設定

データソースと OpenSearch UI アプリケーションの関連付け

OpenSearch UI アプリケーションを作成したら、コンソールまたは を使用して AWS CLI 1 つ以上のデータソースに関連付けることができます。その後、エンドユーザーはこれらのデータソースからデータを取得して、検索、ダッシュボードの操作などを行うことができます。

データソースを OpenSearch UI アプリケーションに関連付ける (コンソール)

コンソールを使用してデータソースを OpenSearch UI アプリケーションに関連付けるには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. [OpenSearch UI (ダッシュボード)] を選択し、OpenSearch UI アプリケーションの名前を選択します。
3. [関連付けられたデータソース] エリアで、[データソースを管理] を選択します。
4. アプリケーションに関連付ける OpenSearch ドメインとコレクションから選択します。

   > **Tip**
   >
   > 探しているデータソースが見つからない場合は、管理者に連絡して必要なアクセス許可を付与してください。詳細については、「the section called “IAM Identity Center 認証を使用するアプリケーションを作成するアクセス許可 (オプション)”」を参照してください。

5. [次へ] を選択してから、[保存] を選択します。

データソースをアプリケーションに関連付けると、アプリケーションの詳細ページで [アプリケーションの起動] ボタンが有効になります。[アプリケーションを起動] を選択して、ワークスペースを作成および管理できる [OpenSearch へようこそ] ページを開くことができます。

ワークスペースの詳しい使い方については、「the section called “Amazon OpenSearch Service ワークスペースの使用”」を参照してください。

VPC 内のドメインへのアクセスの管理

VPC 内の OpenSearch ドメインがアプリケーションに関連付けられている場合、VPC 管理者はコンソールまたは AWS CLIを使用して OpenSearch UI と VPC 間のアクセスを許可する必要があります。

VPC 内のドメインへのアクセスの管理 (コンソール)

AWS マネジメントコンソールを使用して VPC ドメインへのアクセスを設定するには:

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで、[ドメイン] を選択し、VPC ドメインの名前を選択します。

   -または-

   [ドメインを作成] を選択し、ドメインの詳細を設定します。
3. [VPC エンドポイント] タブを選択し、[プリンシパルの承認] を選択します。
4. プリンシパルの承認ダイアログボックスで、他の AWS サービスからプリンシパルの承認を選択し、リストから OpenSearch アプリケーション (ダッシュボード) を選択します。
5. [承認] を選択します。

VPC 内のドメインへのアクセスの管理 (AWS CLI)

を使用して VPC ドメインを承認するには AWS CLI

を使用して VPC ドメインを承認するには AWS CLI、次のコマンドを実行します。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch authorize-vpc-endpoint-access \
 --domain-name domain-name \
 --service application.opensearchservice.amazonaws.com \
```

```
  --region region-id
```

コンソールを使用して VPC ドメインの関連付けを解除するには

関連付けが不要になった場合、VPC ドメイン所有者は次の手順を使用してアクセスを取り消すことができます。

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで、[ドメイン] を選択し、VPC ドメインの名前を選択します。
3. [VPC エンドポイント] タブを選択し、[OpenSearch アプリケーション (ダッシュボード)] の行のボタンを選択します。
4. [アクセスを取り消す] をクリックします。

を使用して VPC ドメインの関連付けを取り消すには AWS CLI

OpenSearch UI アプリケーションとの VPC ドメインの関連付けを取り消すには、次のコマンドを実行します。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch revoke-vpc-endpoint-access \
    --domain-name domain-name \
    --service application.opensearchservice.amazonaws.com \
    --region region-id
```

VPC 内の OpenSearch Serverless コレクションへのアクセスの設定

VPC 内の Amazon OpenSearch Serverless コレクションがアプリケーションに関連付けられている場合、VPC 管理者は新しいネットワークポリシーを作成してコレクションにアタッチすることでアクセスを許可できます。

VPC 内の OpenSearch Serverless コレクションへのアクセスの設定 (コンソール)

コンソールを使用して VPC 内の OpenSearch Serverless コレクションへのアクセスを設定するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションで、[ネットワークポリシー] を選択し、ネットワークポリシーの名前を選択し、[編集] を選択します。

-または-

[ネットワークポリシーを作成] を選択し、ポリシーの詳細を設定します。

3. [アクセスタイプ] エリアで、[プライベート (推奨)] を選択し、[AWS サービスプライベートアクセス] を選択します。
4. 検索フィールドで [サービス] を選択し、次に `application.opensearchservice.amazonaws.com` を選択します。
5. [リソースタイプ] エリアで、[OpenSearch エンドポイントへのアクセスを有効にする] ボックスを選択します。
6. [検索コレクション (複数可) または入力特定のプレフィックス用語 (複数可)] では、検索フィールドに [コレクション名] を選択し、ネットワークポリシーに関連付けるコレクションの名前を入力または選択します。
7. 新しいネットワークポリシーの場合は [作成]、既存のネットワークポリシーの場合は [更新] を選択します。

VPC 内の OpenSearch Serverless コレクションへのアクセスの設定 (AWS CLI)

を使用して VPC 内の OpenSearch Serverless コレクションへのアクセスを設定するには AWS CLI

1. 以下のような .json ファイルを作成します。*#########*を、ユーザー自身の情報に置き換えます。

```
{
    "Description" : "policy-description",
    "Rules": [{
       "ResourceType" : "collection",
        "Resource" : ["collection/collection_name"]
     }],
    "SourceServices" : [
          "application.opensearchservice.amazonaws.com"
      ],
      "AllowFromPublic" : false
}
```

2. VPC 内のコレクションのネットワークポリシーを作成または更新して、OpenSearch UI アプリケーションを操作します。

Create a network policy

以下のコマンドを実行してください。*########*を、ユーザー自身の情報に置き換えます。

```
aws opensearchserverless create-security-policy \
    --type network  \
    --region region \
    --endpoint-url endpoint-url \
    --name network-policy-name \
    --policy file:/path_to_network_policy_json_file
```

コマンドは以下のような情報を返します。

```
{
    "securityPolicyDetail": {
        "createdDate": ******,
        "lastModifiedDate": ******,
        "name": "network-policy-name",
        "policy": [
            {
                "SourceVPCEs": [],
                "AllowFromPublic": false,
                "Description": "",
                "Rules": [
                    {
                        "Resource": [
                            "collection/network-policy-name"
                        ],
                        "ResourceType": "collection"
                    }
                ],
                "SourceServices": [
                    "application.opensearchservice.amazonaws.com"
                ]
            }
        ],
        "policyVersion": "******",
        "type": "network"
    }
}
```

Update a network policy

以下のコマンドを実行してください。*########*を、ユーザー自身の情報に置き換えます。

```
aws opensearchserverless update-security-policy \
    --type network  \
    --region region \
    --endpoint-url endpoint-url \
    --name network-policy-name \
    --policy-version "policy_version_from_output_of_network_policy_creation" \
    --policy file:/path_to_network_policy_json_file
```

コマンドは以下のような情報を返します。

```
{
    "securityPolicyDetail": {
        "createdDate": ******,
        "lastModifiedDate": ******,
        "name": "network-policy-name",
        "policy": [
            {
                "SourceVPCEs": [],
                "AllowFromPublic": false,
                "Description": "",
                "Rules": [
                    {
                        "Resource": [
                            "collection/network-policy-name"
                        ],
                        "ResourceType": "collection"
                    }
                ],
                "SourceServices": [
                    "application.opensearchservice.amazonaws.com"
                ]
            }
        ],
        "policyVersion": "******",
        "type": "network"
    }
}
```

Amazon OpenSearch Service ワークスペースの使用

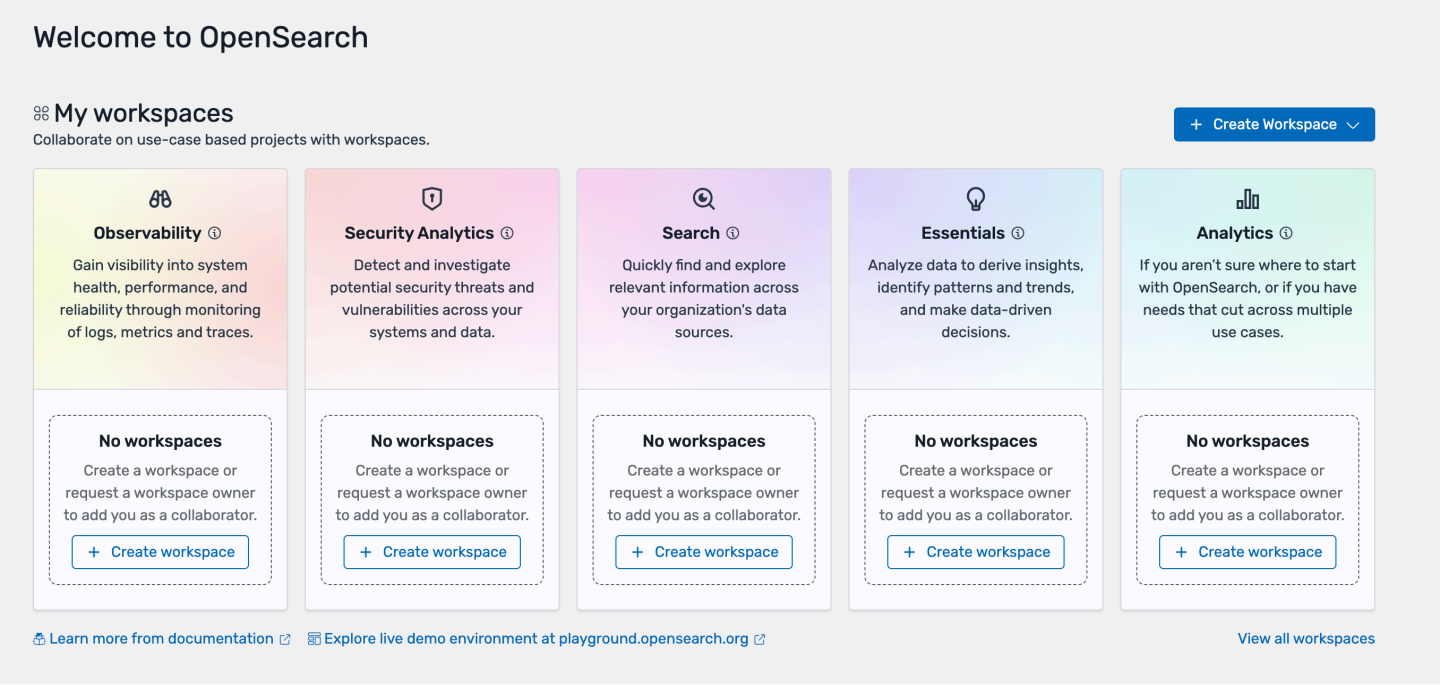
Amazon OpenSearch Service は、複数のユースケース固有のワークスペースの作成をサポートしています。各ワークスペースは、オブザーバビリティ、セキュリティ分析、検索などの一般的なユースケース向けに厳選されたエクスペリエンスを提供します。ワークスペースは共同作業者管理にも対応しているため、目的の共同作業者のみとワークスペースを共有し、それぞれのアクセス許可を管理することもできます。

OpenSearch UI アプリケーションワークスペースの作成

OpenSearch UI アプリケーションが作成され、データソースに関連付けられ、アプリケーションのユーザーアクセス許可が設定されたら、OpenSearch UI アプリケーションを起動してワークスペースを作成できます。

ワークスペースの作成を開始するには、アプリケーションの詳細ページの [アプリケーション起動] ボタンを選択するか、OpenSearch UI アプリケーション URL を使用して、新しいブラウザウィンドウで OpenSearch UI アプリケーションのホームページを開きます。

OpenSearch UI アプリケーションは、ワークスペースを作成するためのオプションを提供し、ホームページ内のすべての既存のワークスペースをユースケース別に分類して一覧表示します。



サポートされているワークスペースの種類に関する詳細については、「the section called “ワークスペースタイプ”」を参照してください。

ワークスペースのプライバシーと共同作業者

ワークスペースのプライバシー設定は、すべてのユーザーのデフォルトのアクセス許可レベルとして定義できます。これは、ワークスペースの作成時または既存のワークスペースの変更時 (ワークスペース [共同作業者] タブ) に行うことができます。選択できるプライバシーオプションは 3 つあります。

- [共同作業者限定] – ワークスペースに明示的に追加した共同作業者のみがアクセスできます。各共同作業者のアクセス許可レベルを定義できます。
- [誰でも閲覧可能] – OpenSearch UI アプリケーションにアクセスできるユーザーは、ワークスペースにアクセスしてアセットを表示できますが、ワークスペースに変更を加えることはできません。
- [誰でも編集可能] – OpenSearch UI アプリケーションにアクセスできるユーザーは、ワークスペースにアクセスし、その中のアセットを表示して、ワークスペース内のアセットを変更することができます。

Workspace Collaborators タブで、IAM ユーザーまたはロールと AWS IAM アイデンティティセンター ユーザーをワークスペースの共同作業者として追加できます。共同作業者が選択できるアクセス許可には 3 つのレベルがあります。

- [読み取り専用] – 共同作業者はワークスペース内のアセットのみを表示できます。ワークスペースが [誰でも編集可能] のプライバシー設定を使用するように構成されている場合、この設定は上書きされます。
- [読み取りと書き込み] – 共同作業者はワークスペース内のアセットを表示および編集できます。[誰でも閲覧可能] のプライバシー設定が適用されているワークスペースの場合、共同編集者は編集を継続できます。
- [管理者] – 共同作業者は設定を更新し、ワークスペースを削除できます。共同作業者は、ワークスペースのプライバシー設定を変更したり、共同作業者を管理したりすることもできます。ワークスペースを作成するユーザーは、ワークスペース管理者に自動的に割り当てられます。

ワークスペースタイプ

Amazon OpenSearch Service には 5 つのワークスペースタイプがあり、それぞれにユースケースごとに異なる機能があります。

- [オブザーバビリティ] ワークスペースは、ログ、メトリクス、トレースのモニタリングを通じて、システムの状態、パフォーマンス、信頼性を可視化するように設計されています。

- [セキュリティ分析] ワークスペースは、システムとデータ全体で潜在的なセキュリティの脅威と脆弱性を検出して調査するように設計されています。
- [検索] ワークスペースは、組織のデータソース全体で関連情報をすばやく検索および探索するように設計されています。
- [Essentials] ワークスペースは、データソースとして OpenSearch Serverless 向けに設計されており、データを分析してインサイトを取得し、パターンと傾向を特定し、データ駆動型の意思決定を迅速に行うことができます。組織のデータソース全体の関連情報は、[Essentials] ワークスペースで検索して確認できます。
- [アナリティクス] (すべての機能) ワークスペースは、多目的ユースケース向けに設計されており、OpenSearch Service UI (ダッシュボード) で使用可能なすべての機能をサポートしています。

クロスリージョンおよびクロスアカウントデータアクセス

OpenSearch UI は、さまざまな AWS アカウントと にわたる OpenSearch AWS リージョンドメインからのデータへのアクセスをサポートしています。要件に応じて 2 つのアプローチから選択できます。次の表は、2 つのアプローチを比較したものです。

> **Note**
>
> クロスアカウントデータアクセスとクラスター間検索はどちらもOpenSearch ドメインでのみ機能します。どちらのアプローチも OpenSearch Serverless コレクションをサポートしていません。

側面	クロスアカウントデータアクセス	クラスター間検索
機能	OpenSearch UI で他のアカウントのドメインを直接データソースとして関連付ける	クラスター間検索接続を使用して、接続されたドメイン間でデータをクエリする
メカニズム	直接アクセス – OpenSearch UI は別のアカウントのターゲットドメインに直接接続します	間接アクセス – リモートドメインにリクエストを中継するには、OpenSearch UI と同じアカウントのローカルドメインが必要です

側面	クロスアカウントデータアクセス	クラスター間検索
クロスアカウントのサポート	はい	はい
クロスリージョンサポート	いいえ – ソースドメインとターゲットドメインは同じ に存在する必要があります AWS リージョン	はい – 送信元ドメインと送信先ドメインは異なる に配置できます AWS リージョン
ドメイン間でデータを結合する	いいえ – 各ドメインは個別のデータソースとして個別にクエリされます	はい – 1 つのクエリで複数の接続されたドメインの結果を集計できます
認証方法	IAM と AWS IAM アイデンティティセンター	IAM (きめ細かなアクセスコントロールを使用)
セットアップの複雑さ	低 – 検証にはクロスアカウント IAM ロールが必要です	高 – クラスター間接続、両方のドメインのアクセスポリシー、きめ細かなアクセスコントロールが必要
OpenSearch UI でのデータソースの可視性	各クロスアカウントドメインは個別のデータソースとして表示されます	リモートドメインは、ローカルソースドメインの接続エイリアスを介してアクセスされます。
リモートドメインへの書き込みアクセス	はい – ターゲットドメインのアクセスポリシーによって制御されます	いいえ – クラスター間検索はリモートドメインへの読み取り専用アクセスを提供します

トピック

- OpenSearch ドメインへのクロスアカウントデータアクセス
- OpenSearch ドメインへのクロスリージョンデータアクセス
- クラスター間検索

OpenSearch ドメインへのクロスアカウントデータアクセス

1 つのアカウントで OpenSearch UI アプリケーションを設定して、異なるアカウントの OpenSearch ドメインにアクセスできます。クロスアカウントデータソースを使用して

OpenSearch UI アプリケーションを作成するときは、ターゲットアカウントの IAM ロールiamRoleForDataSourceArnを指す を指定します。OpenSearch UI は、このロールを引き受け、 es:DescribeDomainを呼び出してドメインアクセシビリティを検証することで、リクエストを検証します。クロスアカウントロールは、データソースの関連付け中のみ使用されます。データプレーンへのアクセスは、ターゲットドメインのアクセスポリシーによって個別に制御されます。

クロスアカウントデータソースのサポートでは、ターゲットドメインできめ細かなアクセスコントロールを有効にする必要があります。きめ細かなアクセスコントロールは、ドメインアクセスポリシー以外にも追加の認可レイヤーを提供するため、個々のインデックス、ドキュメント、フィールドへのアクセスを制御できます。

主要なコンセプト

ソースアカウント

OpenSearch UI アプリケーションをホスト AWS アカウント する 。

ターゲットアカウント

AWS アカウント OpenSearch ドメインが存在する 。

クロスアカウントロール

データソースの関連付け中のみ使用されるターゲットアカウントの IAM ロール。OpenSearch UI はこのロールを引き受けて を呼び出します。これによりes:DescribeDomain、ドメインエンドポイントを取得し、きめ細かなアクセスコントロールが有効になっていることを確認します。これは検出と検証のステップであり、セキュリティの境界ではありません。クロスアカウントロールがデータプレーンアクセスに使用されることはありません。関連付け後、すべてのデータプレーンリクエストは、クロスアカウントロールとは無関係に、ドメインのアクセスポリシーとバックエンドロールマッピングによって承認されます。

IAM Identity Center アプリケーションロール

IAM Identity Center ユーザーデータプレーンアクセスに使用されるソースアカウントの IAM ロール。

クロスアカウントロールの引き受けの仕組み

クロスアカウントデータソースを使用して OpenSearch UI アプリケーションを作成または更新すると、OpenSearch UI は、転送アクセスセッション (FAS) を使用してターゲットアカウントのクロスアカウントロールを引き受けます。FAS は、呼び出し元のプリンシパル自身の IAM IDをsts:AssumeRole呼び出しに伝達します。つまり、次のようになります。

- ターゲットアカウントの信頼ポリシーは、どのソースアカウントプリンシパルがクロスアカウントロールを引き受けることができるかを制御します。
- 引き受けたロールセッションは、 `CreateApplication`または `UpdateApplication`リクエストを開始した発信者の ID を保持します。
- クロスアカウントロールは、 を呼び出す関連付け中にのみ引き受けられます`es:DescribeDomain`。以降のデータプレーンオペレーションには使用されません。

データプレーンアクセスの場合:

- IAM ユーザーは、独自の IAM 認証情報を使用してリクエストに署名します。ターゲットドメインのアクセスポリシーは、これらのリクエストを直接承認します。
- IAM Identity Center ユーザーは、ソースアカウントの IAM Identity Center アプリケーションロール (`iamRoleForIdentityCenterApplicationArn`) で署名されたリクエストを受け取ります。ターゲットドメインのアクセスポリシーとバックエンドロールマッピングは、これらのリクエストを承認します。

前提条件

クロスアカウントデータアクセスを設定する前に、以下があることを確認してください。

- AWS CLI のインストールと設定
- ソースとターゲットの両方へのアクセス AWS アカウント
- きめ細かなアクセスコントロールが有効になっている OpenSearch ドメイン。クロスアカウントデータソースの関連付けは、きめ細かなアクセスコントロールのないドメインではサポートされていません。
- IAM Identity Center フローの場合: AWS IAM アイデンティティセンター 組織インスタンス

シナリオ

認証方法とドメイン設定に一致するシナリオを選択します。

- the section called “シナリオ 1: パブリックドメインにアクセスする IAM ユーザー”
- the section called “シナリオ 2: パブリックドメインにアクセスする IAM Identity Center ユーザー”
- the section called “シナリオ 3: VPC ドメインにアクセスする IAM ユーザー”
- the section called “シナリオ 4: VPC ドメインにアクセスする IAM Identity Center ユーザー”

シナリオ 1: パブリックドメインにアクセスする IAM ユーザー

ステップ 1: クロスアカウント IAM ロールを作成する (ターゲットアカウント)

ソースアカウントがドメイン検証のために引き受けることを許可する IAM ロールをターゲットアカウントに作成します。

クロスアカウントロールを作成するには

1. ソースアカウントがロールを引き受けることを許可する信頼ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::source-account-id:root"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

2. ロールを作成します。

```
aws iam create-role \
  --role-name OpenSearchUIAccessRole \
  --assume-role-policy-document file://trust-policy.json
```

3. `es:DescribeDomain` アクションのみを使用してアクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "es:DescribeDomain",
    "Resource": "arn:aws:es:region:target-account-id:domain/*"
  }]
}
```

4. アクセス許可ポリシーをロールにアタッチします:

```
aws iam put-role-policy \
  --role-name OpenSearchUIAccessRole \
```

```
  --policy-name ValidationOnly \
  --policy-document file://permissions-policy.json
```

ステップ 2: OpenSearch ドメインを作成する (ターゲットアカウント)

きめ細かなアクセスコントロールと暗号化を有効にして、ターゲットアカウントに OpenSearch ドメインを作成します。

```
aws opensearch create-domain \
  --domain-name domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::source-account-
id:root"},"Action":"es:ESHttp*","Resource":"arn:aws:es:region:target-account-
id:domain/domain-name/*"}]}' \
  --region region
```

> **Note**
>
> このアクセスポリシーは、ソースアカウントから IAM プリンシパルへのデータプレーンアクセスをスコープします。アクセスをより制限するには、アカウントのルートプリンシパルを特定の IAM ユーザーまたはロール ARNs。きめ細かなアクセスコントロールは、インデックスとドキュメントへのアクセスを制御するための追加の認可レイヤーを提供します。

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 3: OpenSearch UI アプリケーションを作成する (ソースアカウント)

クロスアカウントデータソースを使用してソースアカウントにアプリケーションを作成します。

```
aws opensearch create-application \
```

```
  --region region \
  --name "cross-account-iam-app" \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/domain-name",
    "dataSourceDescription":"Cross-account domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"*
\"]"}]'
```

ステップ 4: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region region \
  --id application-id
```

- レスポンスからアプリケーションエンドポイント URL に移動します。
- IAM 認証情報を使用してサインインします。
- IAM ユーザーは、独自の認証情報を使用してデータプレーンリクエストに署名します。
- ターゲットドメインアクセスポリシーは、ユーザーがアクセスできるデータを制御します。

シナリオ 2: パブリックドメインにアクセスする IAM Identity Center ユーザー

ステップ 1: クロスアカウント IAM ロールを作成する (ターゲットアカウント)

ソースアカウントがドメイン検証のために引き受けることを許可する IAM ロールをターゲットアカウントに作成します。

クロスアカウントロールを作成するには

1. ソースアカウントがロールを引き受けることを許可する信頼ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
```

```
      "AWS": "arn:aws:iam::source-account-id:root"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

2. ロールを作成します。

```
aws iam create-role \
  --role-name OpenSearchUIAccessRole \
  --assume-role-policy-document file://trust-policy.json
```

3. es:DescribeDomain アクションのみを使用してアクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "es:DescribeDomain",
    "Resource": "arn:aws:es:region:target-account-id:domain/*"
  }]
}
```

4. アクセス許可ポリシーをロールにアタッチします:

```
aws iam put-role-policy \
  --role-name OpenSearchUIAccessRole \
  --policy-name ValidationOnly \
  --policy-document file://permissions-policy.json
```

ステップ 2: OpenSearch ドメインを作成する (ターゲットアカウント)

ターゲットアカウントに OpenSearch ドメインを作成します。と同じコマンドを使用しますがthe section called “ステップ 2: OpenSearch ドメインを作成する (ターゲットアカウント)”、アクセスポリシーを更新して、ソースアカウントからの IAM Identity Center アプリケーションロールを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
```

```
    "Principal": {
      "AWS": "arn:aws:iam::source-account-id:role/NeoIdCAppRole"
    },
    "Action": "es:ESHttp*",
    "Resource": "arn:aws:es:region:target-account-id:domain/domain-name/*"
  }]
}
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 3: IAM Identity Center アプリケーションの IAM ロールを作成する (ソースアカウント)

OpenSearch UI が IAM Identity Center ユーザーデータプレーンアクセスに使用するソースアカウントに IAM ロールを作成します。

IAM Identity Center アプリケーションロールを作成するには

1. 信頼ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:SetContext",
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": "arn:aws:iam::source-account-id:oidc-
provider/portal.sso.region.amazonaws.com/apl/application-id"
        }
      }
    }
  ]
```

```
}
```

2. アクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "OpenSearchDomain",
    "Effect": "Allow",
    "Action": ["es:ESHttp*"],
    "Resource": "*"
  }]
}
```

3. ロールを作成し、ポリシーをアタッチします。

```
aws iam create-role \
  --role-name NeoIdCAppRole \
  --assume-role-policy-document file://neoidc-trust-policy.json

aws iam put-role-policy \
  --role-name NeoIdCAppRole \
  --policy-name NeoIdCAppPermissions \
  --policy-document file://neoidc-permissions-policy.json
```

ステップ 4: IAM Identity Center を使用して OpenSearch UI アプリケーションを作成する (ソースアカウント)

```
aws opensearch create-application \
  --region region \
  --name "cross-account-idc-app" \
  --iam-identity-center-options '{
    "enabled":true,
    "iamIdentityCenterInstanceArn":"arn:aws:sso:::instance/ssoins-instance-id",
    "iamRoleForIdentityCenterApplicationArn":"arn:aws:iam::source-account-
id:role/NeoIdCAppRole"
  }' \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/domain-name",
    "dataSourceDescription":"Cross-account domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
```

```
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"*
\"]"}]'
```

ステップ 5: IAM Identity Center ユーザーとグループを作成して割り当てる

IAM Identity Center ユーザーを作成する

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws identitystore create-user \
  --identity-store-id d-directory-id \
  --user-name user-email \
  --display-name "display-name" \
  --name Formatted=string,FamilyName=last-name,GivenName=first-name \
  --emails Value=user-email,Type=work,Primary=true
```

IAM Identity Center グループを作成し、ユーザーを追加する

以下の コマンドを実行します。

```
aws identitystore create-group \
  --identity-store-id d-directory-id \
  --display-name "OpenSearchUsers" \
  --description "Users with OpenSearch access"

aws identitystore create-group-membership \
  --identity-store-id d-directory-id \
  --group-id group-id \
  --member-id UserId=user-id
```

ユーザーまたはグループをアプリケーションに割り当てる

次のコマンドを実行します。

```
aws sso-admin create-application-assignment \
  --application-arn "arn:aws:sso:::source-account-id:application/ssoins-instance-id/
apl-application-id" \
  --principal-id user-id-or-group-id \
  --principal-type USER
```

ターゲットドメインでバックエンドロールマッピングを設定する

IAM Identity Center グループをターゲットドメインの OpenSearch セキュリティロールにマッピングします。

```
curl -XPUT "https://domain-endpoint/_plugins/_security/api/rolesmapping/all_access" \
  -u admin:master-password \
  -H 'Content-Type: application/json' \
  -d '{
    "backend_roles": ["group-id"],
    "hosts": [],
    "users": []
  }'
```

ステップ 6: 検証とアクセス

```
aws opensearch get-application \
  --region region \
  --id application-id
```

- アプリケーションエンドポイント URL に移動します。
- IAM Identity Center ユーザー認証情報を使用してサインインします。
- IAM Identity Center ユーザーのデータリクエストは、クロスアカウントロールではなく、IAM Identity Center アプリケーションロールで署名されます。
- ドメインコントロールデータアクセス許可のバックエンドロールマッピング。

シナリオ 3: VPC ドメインにアクセスする IAM ユーザー

ステップ 1: クロスアカウント IAM ロールを作成する (ターゲットアカウント)

ソースアカウントがドメイン検証のために引き受けることを許可する IAM ロールをターゲットアカウントに作成します。

クロスアカウントロールを作成するには

1. ソースアカウントがロールを引き受けることを許可する信頼ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
```

```
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::source-account-id:root"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

2. ロールを作成します。

```
aws iam create-role \
  --role-name OpenSearchUIAccessRole \
  --assume-role-policy-document file://trust-policy.json
```

3. es:DescribeDomain アクションのみを使用してアクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "es:DescribeDomain",
    "Resource": "arn:aws:es:region:target-account-id:domain/*"
  }]
}
```

4. アクセス許可ポリシーをロールにアタッチします:

```
aws iam put-role-policy \
  --role-name OpenSearchUIAccessRole \
  --policy-name ValidationOnly \
  --policy-document file://permissions-policy.json
```

ステップ 2: VPC をセットアップする (ターゲットアカウント)

ターゲットアカウントに VPC が既に存在する場合は、このステップをスキップします。

```
# Create VPC
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region region

# Create subnet
```

```
aws ec2 create-subnet \
  --vpc-id vpc-id \
  --cidr-block 10.0.1.0/24 \
  --availability-zone regiona \
  --region region

# Create security group
aws ec2 create-security-group \
  --group-name opensearch-vpc-sg \
  --description "Security group for OpenSearch VPC domain" \
  --vpc-id vpc-id \
  --region region

# Allow inbound HTTPS
aws ec2 authorize-security-group-ingress \
  --group-id security-group-id \
  --protocol tcp \
  --port 443 \
  --cidr 10.0.0.0/16 \
  --region region
```

VPC ドメインの作成について詳しく説明します。

ステップ 3: VPC ドメインを作成する (ターゲットアカウント)

```
aws opensearch create-domain \
  --domain-name vpc-domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --vpc-options "SubnetIds=subnet-id,SecurityGroupIds=security-group-id" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::source-account-
id:root"},"Action":"es:ESHttp*","Resource":"arn:aws:es:region:target-account-
id:domain/vpc-domain-name/*"}]}' \
  --region region
```

> **Note**
>
> このアクセスポリシーは、ソースアカウントから IAM プリンシパルへのデータプレーンアクセスをスコープします。アクセスをより制限するには、アカウントのルートプリンシパルを特定の IAM ユーザーまたはロール ARNs。きめ細かなアクセスコントロールは、インデックスとドキュメントへのアクセスを制御するための追加の認可レイヤーを提供します。

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 4: OpenSearch UI サービスプリンシパル (ターゲットアカウント) の VPC エンドポイントを承認する

> **Important**
>
> これは VPC ドメインに固有の重要なステップです。OpenSearch UI サービスには、VPC エンドポイントへのアクセスを明示的に許可する必要があります。

```
# Authorize the service principal
aws opensearch authorize-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --region region

# Verify authorization
aws opensearch list-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --region region
```

予想されるレスポンス

```
{
  "AuthorizedPrincipalList": [
    {
      "PrincipalType": "AWS_SERVICE",
      "Principal": "application.opensearchservice.amazonaws.com"
    }
  ]
}
```

ステップ 5: OpenSearch UI アプリケーションを作成する (ソースアカウント)

```
aws opensearch create-application \
  --region region \
  --name "cross-account-vpc-iam-app" \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/vpc-domain-name",
    "dataSourceDescription":"Cross-account VPC domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"*
\"]"}]'
```

ステップ 6: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region region \
  --id application-id
```

- レスポンスからアプリケーションエンドポイント URL に移動します。
- IAM 認証情報を使用してサインインします。
- IAM ユーザーは、独自の認証情報を使用してデータプレーンリクエストに署名します。
- ターゲットドメインアクセスポリシーは、ユーザーがアクセスできるデータを制御します。

シナリオ 4: VPC ドメインにアクセスする IAM Identity Center ユーザー

ステップ 1: クロスアカウント IAM ロールを作成する (ターゲットアカウント)

ソースアカウントがドメイン検証のために引き受けることを許可する IAM ロールをターゲットアカウントに作成します。

クロスアカウントロールを作成するには

1. ソースアカウントがロールを引き受けることを許可する信頼ポリシーを作成します。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::source-account-id:root"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

2. ロールを作成します。

```
aws iam create-role \
  --role-name OpenSearchUIAccessRole \
  --assume-role-policy-document file://trust-policy.json
```

3. es:DescribeDomain アクションのみを使用してアクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "es:DescribeDomain",
    "Resource": "arn:aws:es:region:target-account-id:domain/*"
  }]
}
```

4. アクセス許可ポリシーをロールにアタッチします:

```
aws iam put-role-policy \
  --role-name OpenSearchUIAccessRole \
  --policy-name ValidationOnly \
  --policy-document file://permissions-policy.json
```

ステップ 2: VPC をセットアップする (ターゲットアカウント)

ターゲットアカウントに VPC が既に存在する場合は、このステップをスキップします。

```
# Create VPC
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region region
```

```
# Create subnet
aws ec2 create-subnet \
  --vpc-id vpc-id \
  --cidr-block 10.0.1.0/24 \
  --availability-zone regiona \
  --region region

# Create security group
aws ec2 create-security-group \
  --group-name opensearch-vpc-sg \
  --description "Security group for OpenSearch VPC domain" \
  --vpc-id vpc-id \
  --region region

# Allow inbound HTTPS
aws ec2 authorize-security-group-ingress \
  --group-id security-group-id \
  --protocol tcp \
  --port 443 \
  --cidr 10.0.0.0/16 \
  --region region
```

VPC ドメインの作成について詳しく説明します。

ステップ 3: VPC ドメインを作成する (ターゲットアカウント)

と同じコマンドを使用しますがthe section called “ステップ 3: VPC ドメインを作成する (ターゲットアカウント)”、アクセスポリシーを更新して、ソースアカウントからの IAM Identity Center アプリケーションロールを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::source-account-id:role/NeoIdCAppRole"
    },
    "Action": "es:ESHttp*",
    "Resource": "arn:aws:es:region:target-account-id:domain/vpc-domain-name/*"
  }]
}
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 4: OpenSearch UI サービスプリンシパル (ターゲットアカウント) の VPC エンドポイントを承認する

> **Important**
>
> これは VPC ドメインに固有の重要なステップです。OpenSearch UI サービスには、VPC エンドポイントへのアクセスを明示的に許可する必要があります。

```
# Authorize the service principal
aws opensearch authorize-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --region region

# Verify authorization
aws opensearch list-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --region region
```

予想されるレスポンス

```
{
  "AuthorizedPrincipalList": [
    {
      "PrincipalType": "AWS_SERVICE",
      "Principal": "application.opensearchservice.amazonaws.com"
    }
  ]
}
```

ステップ 5: IAM Identity Center アプリケーションの IAM ロールを作成する (ソースアカウント)

OpenSearch UI が IAM Identity Center ユーザーデータプレーンアクセスに使用するソースアカウントに IAM ロールを作成します。

IAM Identity Center アプリケーションロールを作成するには

1. 信頼ポリシーを作成します。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:SetContext",
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": "arn:aws:iam::source-account-id:oidc-
provider/portal.sso.region.amazonaws.com/apl/application-id"
        }
      }
    }
  ]
}
```

2. アクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "OpenSearchDomain",
    "Effect": "Allow",
    "Action": ["es:ESHttp*"],
    "Resource": "*"
  }]
}
```

3. ロールを作成し、ポリシーをアタッチします。

```
aws iam create-role \
  --role-name NeoIdCAppRole \
  --assume-role-policy-document file://neoidc-trust-policy.json
```

```
aws iam put-role-policy \
  --role-name NeoIdCAppRole \
  --policy-name NeoIdCAppPermissions \
  --policy-document file://neoidc-permissions-policy.json
```

ステップ 6: IAM Identity Center を使用して OpenSearch UI アプリケーションを作成する (ソースアカウント)

```
aws opensearch create-application \
  --region region \
  --name "cross-account-vpc-idc-app" \
  --iam-identity-center-options '{
    "enabled":true,
    "iamIdentityCenterInstanceArn":"arn:aws:sso:::instance/ssoins-instance-id",
    "iamRoleForIdentityCenterApplicationArn":"arn:aws:iam::source-account-
id:role/NeoIdCAppRole"
  }' \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/vpc-domain-name",
    "dataSourceDescription":"Cross-account VPC domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"*
\"]"}]'
```

ステップ 7: IAM Identity Center ユーザーとグループを作成して割り当てる

IAM Identity Center ユーザーを作成する

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws identitystore create-user \
  --identity-store-id d-directory-id \
  --user-name user-email \
  --display-name "display-name" \
  --name Formatted=string,FamilyName=last-name,GivenName=first-name \
  --emails Value=user-email,Type=work,Primary=true
```

IAM Identity Center グループを作成し、ユーザーを追加する

以下の コマンドを実行します。

```
aws identitystore create-group \
  --identity-store-id d-directory-id \
  --display-name "OpenSearchUsers" \
  --description "Users with OpenSearch access"

aws identitystore create-group-membership \
  --identity-store-id d-directory-id \
  --group-id group-id \
  --member-id UserId=user-id
```

ユーザーまたはグループをアプリケーションに割り当てる

次のコマンドを実行します。

```
aws sso-admin create-application-assignment \
  --application-arn "arn:aws:sso:::source-account-id:application/ssoins-instance-id/
apl-application-id" \
  --principal-id user-id-or-group-id \
  --principal-type USER
```

ターゲットドメインでバックエンドロールマッピングを設定する

IAM Identity Center グループをターゲットドメインの OpenSearch セキュリティロールにマッピングします。

```
curl -XPUT "https://domain-endpoint/_plugins/_security/api/rolesmapping/all_access" \
  -u admin:master-password \
  -H 'Content-Type: application/json' \
  -d '{
    "backend_roles": ["group-id"],
    "hosts": [],
    "users": []
  }'
```

ステップ 8: 検証とアクセス

```
aws opensearch get-application \
  --region region \
  --id application-id
```

- アプリケーションエンドポイント URL に移動します。

- IAM Identity Center ユーザー認証情報を使用してサインインします。
- IAM Identity Center ユーザーのデータリクエストは、クロスアカウントロールではなく、IAM Identity Center アプリケーションロールで署名されます。
- ドメインコントロールデータアクセス許可のバックエンドロールマッピング。

アプリケーションの管理

クロスアカウントデータソースを使用してアプリケーションを更新する

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch update-application \
  --region region \
  --id application-id \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/domain-1",
    "dataSourceDescription":"First cross-account domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
  },{
    "dataSourceArn":"arn:aws:es:region:target-account-id:domain/domain-2",
    "dataSourceDescription":"Second cross-account domain",
    "iamRoleForDataSourceArn":"arn:aws:iam::target-account-
id:role/OpenSearchUIAccessRole"
  }]'
```

> **Important**
>
> 更新オペレーションは、データソース配列全体を置き換えます。保持するすべてのデータソースを含めます。

アプリケーションを一覧表示する

次のコマンドを実行します。

```
aws opensearch list-applications \
  --region region
```

アプリケーションの削除

次のコマンドを実行します。

```
aws opensearch delete-application \
  --region region \
  --id application-id
```

VPC エンドポイントアクセスの取り消し

次のコマンドを実行します。

```
aws opensearch revoke-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --region region
```

クイックリファレンス

次の表は、ドメインタイプと認証方法の主な違いをまとめたものです。

VPC ドメインと比較したパブリックドメイン

側面	パブリックドメイン	VPC ドメイン
VPC エンドポイント認可	不要	必須 – 認可する必要があります application.opensearchservice.amazonaws.com
ネットワーク設定	なし	HTTPS (443) インバウンドの VPC、サブネット、セキュリティグループ
IAM アクセスポリシー	必須	必須
クロスアカウントロール	クロスアカウントで必須	クロスアカウントで必須

IAM ユーザーと IAM Identity Center ユーザーの比較

側面	IAM ユーザー	IAM Identity Center ユーザー
データプレーンの認証情報	ユーザー独自の IAM 認証情報	IAM Identity Center アプリケーションロール
アクセスコントロール	ドメインアクセスポリシー	ドメインアクセスポリシーとバックエンドロールマッピング
追加セットアップ	なし	IAM Identity Center アプリケーションロール、ユーザー/グループの作成、アプリケーションの割り当て、バックエンドロールマッピング
OpenSearch UI アプリケーション設定	IAM Identity Center オプションなし	`--iam-identity-center-options` が必須

重要な注意事項

- は、 と同じアカウントにある`iamRoleForDataSourceArn`必要がありま`dataSourceArn`す。
- `iamRoleForDataSourceArn` は、クロスアカウントデータソースにのみ必要です。同じアカウントデータソースの場合は省略します。
- クロスアカウントロールには アクセス`es:DescribeDomain`許可のみが必要です。データプレーンへのアクセスには使用されません。
- VPC ドメインの場合、IAM ポリシーと VPC エンドポイント認可の両方を設定する必要があります。
- サポートされているエンジンバージョン: OpenSearch 1.3 以降。
- クロスアカウントデータソースの関連付けでは、ターゲットドメインできめ細かなアクセスコントロールを有効にする必要があります。

トラブルシューティング

問題	解決方法
「ドメインにアクセスできません」でアプリケーションの作成が失敗する	クロスアカウントロールに アクセス`es:DescribeDomain` 許可があり、信頼ポリシーでソースアカウントが許可されていることを確認します。
VPC ドメインの関連付けが失敗する	VPC エンドポイントが に対して承認されていることを確認します`application.opensearchservice.amazonaws.com` 。
IAM ユーザーのデータプレーンアクセスが拒否されました	ターゲットドメインアクセスポリシーが IAM ユーザーまたはロールプリンシパルを許可していることを確認します。
IAM Identity Center ユーザーのデータプレーンアクセスが拒否されました	バックエンドロールマッピングに IAM Identity Center グループ ID が含まれ、ドメインポリシーで IAM Identity Center アプリケーションロールが許可されていることを確認します。
アカウントの不一致エラー	`iamRoleForDataSourceArn` が のドメインと同じアカウントにあることを確認します`dataSourceArn` 。

OpenSearch ドメインへのクロスリージョンデータアクセス

OpenSearch UI アプリケーションは、異なる の OpenSearch ドメインにアクセスする AWS リージョン ように 1 つの AWS リージョンで設定できます。これにより、同じパーティション内の複数の AWS リージョンにわたって OpenSearch ドメインからデータを集約する統合ダッシュボードを作成できます。クロスリージョンデータソースのサポートでは、ターゲットドメインできめ細かなアクセスコントロールを有効にする必要があります。きめ細かなアクセスコントロールは、ドメインアクセスポリシー以外にも追加の認可レイヤーを提供するため、個々のインデックス、ドキュメント、フィールドへのアクセスを制御できます。

主要なコンセプト

アプリケーションリージョン

OpenSearch UI アプリケーションがホスト AWS リージョン されている 。

ターゲット リージョン

AWS リージョン OpenSearch ドメインが存在する 。これは、OpenSearch UI がそのリージョンで使用可能かどうかに関係なく、同じパーティション内の任意のリージョンにすることができます。

クロスアカウントロール

データソースの関連付け中のみ使用されるターゲットアカウントの IAM ロール。OpenSearch UI はこのロールを引き受けて を呼び出します。これにより`es:DescribeDomain`、ドメインエンドポイントを取得し、きめ細かなアクセスコントロールが有効になっていることを確認します。このロールは、ドメインがアプリケーションとは異なるアカウントにある場合にのみ必要です。詳細については、「the section called “OpenSearch ドメインへのクロスアカウントデータアクセス”」を参照してください。

IAM Identity Center アプリケーションロール

IAM Identity Center ユーザーデータプレーンアクセスに使用されるアプリケーションアカウントの IAM ロール。

サポートされているリージョン (VPC ドメインの場合)

VPC ドメインの場合、VPC AWS リージョンエンドポイントを承認するときに OpenSearch UI アプリケーションがホストされている を許可リストに登録する必要があります。この許可リストは、OpenSearch UI が VPC ドメインを呼び出すために必要です。

前提条件

クロスリージョンデータアクセスを設定する前に、以下があることを確認してください。

- AWS CLI のインストールと設定
- アプリケーションリージョンとターゲットリージョンの両方 AWS アカウント の へのアクセス
- きめ細かなアクセスコントロールが有効になっている OpenSearch ドメイン。マルチリージョンデータソースの関連付けは、きめ細かなアクセスコントロールが有効になっているドメインでのみサポートされます。

- クロスアカウントシナリオの場合: ソースとターゲットの両方へのアクセス AWS アカウント
- IAM Identity Center フローの場合: AWS IAM アイデンティティセンター 組織インスタンス。OpenSearch UI アプリケーションは、IAM Identity Center インスタンスと同じリージョンに存在する必要があります。

シナリオ

認証方法とドメイン設定に一致するシナリオを選択します。

- the section called “シナリオ 1: 別のリージョンのパブリックドメインにアクセスする IAM ユーザー”
- the section called “シナリオ 2: 別のリージョンのパブリックドメインにアクセスする IAM Identity Center ユーザー”
- the section called “シナリオ 3: 別のリージョンの VPC ドメインにアクセスする IAM ユーザー”
- the section called “シナリオ 4: 別のリージョンの VPC ドメインにアクセスする IAM Identity Center ユーザー”

各シナリオは、同じアカウントのクロスリージョンアクセスを対象としています。クロスアカウントクロスリージョンアクセスの場合は、これらのシナリオの手順と、「」で説明されているクロスアカウントロールの設定を組み合わせてくださいthe section called “OpenSearch ドメインへのクロスアカウントデータアクセス”。

シナリオ 1: 別のリージョンのパブリックドメインにアクセスする IAM ユーザー

このシナリオでは、1 つのリージョンに OpenSearch UI アプリケーションを作成し、同じアカウント内の別のリージョンにあるパブリック OpenSearch ドメインに接続します。

ステップ 1: OpenSearch ドメインを作成する (ターゲットリージョン)

きめ細かなアクセスコントロールを有効にして、ターゲットリージョンに OpenSearch ドメインを作成します。アクセスポリシーの範囲をアカウントのルートまたは特定の IAM プリンシパルに設定します。

```
aws opensearch create-domain \
  --domain-name domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
```

```
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::account-
id:root"},"Action":"es:ESHttp*","Resource":"arn:aws:es:target-region:account-
id:domain/domain-name/*"}]}' \
  --region target-region
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 2: OpenSearch UI アプリケーションを作成する (アプリケーションリージョン)

クロスリージョンデータソースを使用して、アプリケーションリージョンにアプリケーションを作成します。リージョンはデータソース ARN から自動的に抽出されます。

```
aws opensearch create-application \
  --region application-region \
  --name "cross-region-iam-app" \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:target-region:account-id:domain/domain-name",
    "dataSourceDescription":"Cross-region domain"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"test-
user\"]"}]'
```

ステップ 3: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region application-region \
  --id application-id
```

- レスポンスからアプリケーションエンドポイント URL に移動します。
- IAM 認証情報を使用してサインインします。

- IAM ユーザーは、独自の認証情報を使用してデータプレーンリクエストに署名します。
- ターゲットドメインのアクセスポリシーとバックエンドロールマッピングは、ユーザーがアクセスできるデータを制御します。

シナリオ 2: 別のリージョンのパブリックドメインにアクセスする IAM Identity Center ユーザー

このシナリオでは、1 つのリージョンで IAM アイデンティティセンター認証を使用して OpenSearch UI アプリケーションを作成し、同じアカウント内の別のリージョンのパブリック OpenSearch ドメインに接続します。

ステップ 1: IAM アイデンティティセンターを有効にして OpenSearch ドメインを作成する (ターゲットリージョン)

きめ細かなアクセスコントロールと IAM Identity Center 統合を有効にして、ターゲットリージョンに OpenSearch ドメインを作成します。で `--identity-center-options`パラメータを使用して`IdentityCenterInstanceRegion`、IAM Identity Center インスタンスが配置されているリージョンを指定します。このリージョンは、OpenSearch UI アプリケーションがホストされているリージョンと同じである必要があります。

```
aws opensearch create-domain \
  --domain-name domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --identity-center-options
 '{"EnabledAPIAccess":true,"IdentityCenterInstanceARN":"arn:aws:sso:::instance/
ssoins-instance-id","IdentityCenterInstanceRegion":"idc-
region","RolesKey":"GroupId","SubjectKey":"UserId"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::account-
id:role/NeoIdCAppRole"},"Action":"es:ESHttp*","Resource":"arn:aws:es:target-
region:account-id:domain/domain-name/*"}]}' \
```

```
  --region target-region
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 2: IAM Identity Center アプリケーションの IAM ロールを作成する

OpenSearch UI が IAM Identity Center ユーザーデータプレーンアクセスに使用する IAM ロールを作成します。

IAM Identity Center アプリケーションロールを作成するには

1. sts:AssumeRole ステートメントのみを使用して信頼ポリシーを作成します。次のステップでアプリケーションを作成した後、このポリシーを更新して sts:SetContextステートメントを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. アクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "OpenSearchDomain",
    "Effect": "Allow",
    "Action": ["es:ESHttp*"],
    "Resource": "arn:aws:es:target-region:account-id:domain/domain-name/*"
  }]
}
```

3. ロールを作成し、ポリシーをアタッチします。

```
aws iam create-role \
```

```
  --role-name NeoIdCAppRole \
  --assume-role-policy-document file://neoidc-trust-policy.json

aws iam put-role-policy \
  --role-name NeoIdCAppRole \
  --policy-name NeoIdCAppPermissions \
  --policy-document file://neoidc-permissions-policy.json
```

ステップ 3: IAM アイデンティティセンターで OpenSearch UI アプリケーションを作成する (アプリケーションリージョン)

> **Note**
>
> IAM Identity Center インスタンスが OpenSearch UI アプリケーションリージョンと同じリージョンにあることを確認します。

```
aws opensearch create-application \
  --region application-region \
  --name "cross-region-idc-app" \
  --iam-identity-center-options '{
    "enabled":true,
    "iamIdentityCenterInstanceArn":"arn:aws:sso:::instance/ssoins-instance-id",
    "iamRoleForIdentityCenterApplicationArn":"arn:aws:iam::account-
id:role/NeoIdCAppRole"
  }' \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:target-region:account-id:domain/domain-name",
    "dataSourceDescription":"Cross-region domain"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"test-
user\"]"}]'
```

アプリケーションを作成したら、レスポンスの SSO アプリケーション ID を書き留めます。次に、IAM Identity Center アプリケーションロールの信頼ポリシーを更新して、`sts:SetContext`ステートメントを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:SetContext",
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": "arn:aws:iam::account-id:oidc-provider/
portal.sso.idc-region.amazonaws.com/apl/application-id"
        }
      }
    }
  ]
}
```

```
aws iam update-assume-role-policy \
  --role-name NeoIdCAppRole \
  --policy-document file://updated-trust-policy.json
```

ステップ 4: IAM Identity Center ユーザーとグループを作成して割り当てる

IAM Identity Center ユーザーを作成する

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws identitystore create-user \
  --identity-store-id d-directory-id \
  --user-name user-email \
  --display-name "display-name" \
  --name Formatted=string,FamilyName=last-name,GivenName=first-name \
  --emails Value=user-email,Type=work,Primary=true
```

IAM Identity Center グループを作成し、ユーザーを追加する

以下の コマンドを実行します。

```
aws identitystore create-group \
  --identity-store-id d-directory-id \
  --display-name "OpenSearchUsers" \
  --description "Users with OpenSearch access"

aws identitystore create-group-membership \
  --identity-store-id d-directory-id \
  --group-id group-id \
  --member-id UserId=user-id
```

ユーザーまたはグループをアプリケーションに割り当てる

次のコマンドを実行します。

```
aws sso-admin create-application-assignment \
  --application-arn "arn:aws:sso::account-id:application/ssoins-instance-id/
apl-application-id" \
  --principal-id user-id-or-group-id \
  --principal-type USER
```

ターゲットドメインでバックエンドロールマッピングを設定する

IAM Identity Center グループをターゲットドメインの OpenSearch セキュリティロールにマッピングします。

```
curl -XPATCH "https://domain-endpoint/_plugins/_security/api/rolesmapping/all_access" \
  -u admin:master-password \
  -H 'Content-Type: application/json' \
  -d '[{"op": "add", "path": "/backend_roles", "value": ["group-id"]}]'
```

ステップ 5: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region application-region \
  --id application-id
```

- アプリケーションエンドポイント URL に移動します。
- IAM Identity Center ユーザー認証情報を使用してサインインします。

- IAM Identity Center ユーザーのデータリクエストは、IAM Identity Center アプリケーションロールで署名されます。
- ドメインコントロールデータアクセス許可のバックエンドロールマッピング。

シナリオ 3: 別のリージョンの VPC ドメインにアクセスする IAM ユーザー

このシナリオでは、1 つのリージョンに OpenSearch UI アプリケーションを作成し、同じアカウント内の別のリージョンの VPC OpenSearch ドメインに接続します。VPC ドメインには、追加のネットワーク設定と、クロスリージョンサポートによる明示的な VPC エンドポイント認可が必要です。

ステップ 1: VPC をセットアップする (ターゲットリージョン)

ターゲットリージョンに VPC がすでに存在する場合は、このステップをスキップします。

```
# Create VPC
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region target-region

# Create subnet
aws ec2 create-subnet \
  --vpc-id vpc-id \
  --cidr-block 10.0.1.0/24 \
  --availability-zone target-regiona \
  --region target-region

# Create security group
aws ec2 create-security-group \
  --group-name opensearch-vpc-sg \
  --description "Security group for OpenSearch VPC domain" \
  --vpc-id vpc-id \
  --region target-region

# Allow inbound HTTPS
aws ec2 authorize-security-group-ingress \
  --group-id security-group-id \
  --protocol tcp \
  --port 443 \
  --cidr 10.0.0.0/16 \
  --region target-region
```

VPC ドメインの作成について詳しく説明します。

ステップ 2: VPC ドメインを作成する (ターゲットリージョン)

```
aws opensearch create-domain \
  --domain-name vpc-domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --vpc-options "SubnetIds=subnet-id,SecurityGroupIds=security-group-id" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::account-
id:root"},"Action":"es:ESHttp*","Resource":"arn:aws:es:target-region:account-
id:domain/vpc-domain-name/*"}]}' \
  --region target-region
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 3: クロスリージョンサポート (ターゲットリージョン) を使用して OpenSearch UI サービスプリンシパルの VPC エンドポイントを承認する

```
# Authorize the service principal with cross-region support
aws opensearch authorize-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --service-options '{"SupportedRegions":["target-region","application-region"]}' \
  --region target-region

# Verify authorization
aws opensearch list-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --region target-region
```

予想されるレスポンス

```
{
  "AuthorizedPrincipalList": [
```

```
    {
      "PrincipalType": "AWS_SERVICE",
      "Principal": "application.opensearchservice.amazonaws.com",
      "ServiceOptions": {
        "SupportedRegions": ["target-region", "application-region"]
      }
    }
  ]
}
```

ステップ 4: OpenSearch UI アプリケーションを作成する (アプリケーションリージョン)

```
aws opensearch create-application \
  --region application-region \
  --name "cross-region-vpc-iam-app" \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:target-region:account-id:domain/vpc-domain-name",
    "dataSourceDescription":"Cross-region VPC domain"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"test-
user\"]"}]'
```

ステップ 5: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region application-region \
  --id application-id
```

- レスポンスからアプリケーションエンドポイント URL に移動します。
- IAM 認証情報を使用してサインインします。
- IAM ユーザーは、独自の認証情報を使用してデータプレーンリクエストに署名します。
- ターゲットドメインのアクセスポリシーとバックエンドロールマッピングは、ユーザーがアクセスできるデータを制御します。

シナリオ 4: 別のリージョンの VPC ドメインにアクセスする IAM Identity Center ユーザー

このシナリオでは、1 つのリージョンで IAM アイデンティティセンター認証を使用して OpenSearch UI アプリケーションを作成し、同じアカウント内の別のリージョンの VPC OpenSearch ドメインに接続します。

ステップ 1: VPC をセットアップする (ターゲットリージョン)

ターゲットリージョンに VPC がすでに存在する場合は、このステップをスキップします。

```
# Create VPC
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region target-region

# Create subnet
aws ec2 create-subnet \
  --vpc-id vpc-id \
  --cidr-block 10.0.1.0/24 \
  --availability-zone target-regiona \
  --region target-region

# Create security group
aws ec2 create-security-group \
  --group-name opensearch-vpc-sg \
  --description "Security group for OpenSearch VPC domain" \
  --vpc-id vpc-id \
  --region target-region

# Allow inbound HTTPS
aws ec2 authorize-security-group-ingress \
  --group-id security-group-id \
  --protocol tcp \
  --port 443 \
  --cidr 10.0.0.0/16 \
  --region target-region
```

VPC ドメインの作成について詳しく説明します。

ステップ 2: IAM アイデンティティセンターを有効にして VPC ドメインを作成する (ターゲットリージョン)

きめ細かなアクセスコントロール、IAM Identity Center 統合、VPC 設定を有効にして、ターゲットリージョンに OpenSearch ドメインを作成します。アクセスポリシーを更新して IAM Identity Center アプリケーションロールを許可し、 `--identity-center-options`パラメータを追加します。

```
aws opensearch create-domain \
  --domain-name vpc-domain-name \
  --engine-version OpenSearch_2.19 \
  --cluster-config InstanceType=m5.large.search,InstanceCount=1 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=100" \
  --vpc-options "SubnetIds=subnet-id,SecurityGroupIds=security-group-id" \
  --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"admin","MasterUserPassword":"master-password"}}' \
  --node-to-node-encryption-options '{"Enabled":true}' \
  --encryption-at-rest-options '{"Enabled":true}' \
  --domain-endpoint-options '{"EnforceHTTPS":true,"TLSSecurityPolicy":"Policy-Min-
TLS-1-2-2019-07"}' \
  --identity-center-options
 '{"EnabledAPIAccess":true,"IdentityCenterInstanceARN":"arn:aws:sso:::instance/
ssoins-instance-id","IdentityCenterInstanceRegion":"idc-
region","RolesKey":"GroupId","SubjectKey":"UserId"}' \
  --access-policies '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::account-
id:role/NeoIdCAppRole"},"Action":"es:ESHttp*","Resource":"arn:aws:es:target-
region:account-id:domain/vpc-domain-name/*"}]}' \
  --region target-region
```

続行するActive前に、ドメインのステータスが になるまで待ちます。

ステップ 3: クロスリージョンサポート (ターゲットリージョン) を使用して OpenSearch UI サービスプリンシパルの VPC エンドポイントを承認する

> **Important**
>
> これは、クロスリージョンアクセスを持つ VPC ドメインに固有の重要なステップです。OpenSearch UI サービスには、VPC エンドポイントへのアクセスを明示的に許可し、アプリケーションリージョンをSupportedRegionsリストに含める必要があります。

```
# Authorize the service principal with cross-region support
aws opensearch authorize-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --service-options '{"SupportedRegions":["target-region","application-region"]}' \
  --region target-region

# Verify authorization
aws opensearch list-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --region target-region
```

予想されるレスポンス

```
{
  "AuthorizedPrincipalList": [
    {
      "PrincipalType": "AWS_SERVICE",
      "Principal": "application.opensearchservice.amazonaws.com",
      "ServiceOptions": {
        "SupportedRegions": ["target-region", "application-region"]
      }
    }
  ]
}
```

ステップ 4: IAM Identity Center アプリケーションの IAM ロールを作成する

OpenSearch UI が IAM Identity Center ユーザーデータプレーンアクセスに使用する IAM ロールを作成します。

IAM Identity Center アプリケーションロールを作成するには

1. `sts:AssumeRole` ステートメントのみを使用して信頼ポリシーを作成します。次のステップでアプリケーションを作成した後、このポリシーを更新して `sts:SetContext`ステートメントを追加します。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
   ```

```
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. アクセス許可ポリシーを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "OpenSearchDomain",
    "Effect": "Allow",
    "Action": ["es:ESHttp*"],
    "Resource": "arn:aws:es:target-region:account-id:domain/vpc-domain-name/*"
  }]
}
```

3. ロールを作成し、ポリシーをアタッチします。

```
aws iam create-role \
  --role-name NeoIdCAppRole \
  --assume-role-policy-document file://neoidc-trust-policy.json

aws iam put-role-policy \
  --role-name NeoIdCAppRole \
  --policy-name NeoIdCAppPermissions \
  --policy-document file://neoidc-permissions-policy.json
```

ステップ 5: IAM アイデンティティセンターで OpenSearch UI アプリケーションを作成する (アプリケーションリージョン)

```
aws opensearch create-application \
  --region application-region \
  --name "cross-region-vpc-idc-app" \
  --iam-identity-center-options '{
    "enabled":true,
    "iamIdentityCenterInstanceArn":"arn:aws:sso:::instance/ssoins-instance-id",
    "iamRoleForIdentityCenterApplicationArn":"arn:aws:iam::account-
id:role/NeoIdCAppRole"
```

```
  }' \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:target-region:account-id:domain/vpc-domain-name",
    "dataSourceDescription":"Cross-region VPC domain"
  }]' \
  --app-configs '[{"key":"opensearchDashboards.dashboardAdmin.users","value":"[\"test-
user\"]"}]'
```

アプリケーションを作成したら、レスポンスの SSO アプリケーション ID を書き留めます。次に、IAM Identity Center アプリケーションロールの信頼ポリシーを更新して、`sts:SetContext`ステートメントを追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application.opensearchservice.amazonaws.com"
      },
      "Action": "sts:SetContext",
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": "arn:aws:iam::account-id:oidc-provider/
portal.sso.idc-region.amazonaws.com/apl/application-id"
        }
      }
    }
  ]
}
```

```
aws iam update-assume-role-policy \
  --role-name NeoIdCAppRole \
  --policy-document file://updated-trust-policy.json
```

ステップ 6: IAM Identity Center ユーザーとグループを作成して割り当てる

ユーザー、グループthe section called “ステップ 4: IAM Identity Center ユーザーとグループを作成して割り当てる”を作成し、アプリケーションに割り当て、ターゲットドメインでバックエンドロールマッピングを設定するには、 と同じ手順に従います。

ステップ 7: 検証とアクセス

アプリケーションの詳細を取得してエンドポイント URL を取得します。

```
aws opensearch get-application \
  --region application-region \
  --id application-id
```

- アプリケーションエンドポイント URL に移動します。
- IAM Identity Center ユーザー認証情報を使用してサインインします。
- IAM Identity Center ユーザーのデータリクエストは、IAM Identity Center アプリケーションロールで署名されます。
- ドメインコントロールデータアクセス許可のバックエンドロールマッピング。

アプリケーションの管理

クロスリージョンデータソースを使用してアプリケーションを更新する

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearch update-application \
  --region application-region \
  --id application-id \
  --data-sources '[{
    "dataSourceArn":"arn:aws:es:target-region-1:account-id:domain/domain-1",
    "dataSourceDescription":"Domain in target Region 1"
  },{
    "dataSourceArn":"arn:aws:es:target-region-2:account-id:domain/domain-2",
    "dataSourceDescription":"Domain in target Region 2"
  }]'
```

> **Important**
>
> 更新オペレーションは、データソース配列全体を置き換えます。保持するすべてのデータソースを含めます。

アプリケーションを一覧表示する

次のコマンドを実行します。

```
aws opensearch list-applications \
  --region application-region
```

アプリケーションの削除

次のコマンドを実行します。

```
aws opensearch delete-application \
  --region application-region \
  --id application-id
```

特定のリージョンの VPC エンドポイントアクセスを取り消す

他の を保持しながら特定の AWS リージョンのクロスリージョンアクセスを取り消すには、 `--service-options`パラメータをリージョンとともに使用して取り消します。

```
aws opensearch revoke-vpc-endpoint-access \
  --domain-name vpc-domain-name \
  --service "application.opensearchservice.amazonaws.com" \
  --service-options '{"SupportedRegions":["region-to-revoke"]}' \
  --region target-region
```

クイックリファレンス

次の表は、ドメインタイプ、認証方法、同一リージョンアクセスとクロスリージョンアクセスの主な違いをまとめたものです。

VPC ドメインと比較したパブリックドメイン

側面	パブリックドメイン	VPC ドメイン
VPC エンドポイント認可	不要	必須 – `application.opensearchservice.amazonaws.com` で を承認する必要があります `SupportedRegions`
ネットワーク設定	なし	HTTPS (443) インバウンドの VPC、サブネット、セキュリティグループ
IAM アクセスポリシー	必須	必須

IAM ユーザーと IAM Identity Center ユーザーの比較

側面	IAM ユーザー	IAM Identity Center ユーザー
データプレーンの認証情報	ユーザー独自の IAM 認証情報	IAM Identity Center アプリケーションロール
アクセスコントロール	ドメインアクセスポリシーとバックエンドロールマッピング	ドメインアクセスポリシーとバックエンドロールマッピング
アプリケーションリージョンの制約	任意のリージョン	IAM Identity Center インスタンスと同じリージョンに存在する必要があります
ドメイン設定	標準	`--identity-center-options` で必要 `IdentityCenterInstanceRegion`
追加セットアップ	なし	IAM Identity Center アプリケーションロール、ユーザー/グループの作成、アプリケーションの割り当て、バックエンドロールマッピング

同一リージョンとクロスリージョンの比較

側面	同一リージョン	クロスリージョン
データソース ARN	アプリケーションと同じリージョン	アプリケーションとは異なるリージョン (同じパーティション)
VPC エンドポイント認可	省略 `--service-options`	`--service-options` に を含める `SupportedRegions`
IAM Identity Center ドメイン設定	`IdentityCenterInstanceRegion` が任意	`IdentityCenterInstanceRegion` が必須
クロスパーティションサポート	該当なし	サポートされていない – データソースは同じパーティションに存在する必要があります

重要な注意事項

- クロスリージョンデータソースの関連付けでは、ターゲットドメインできめ細かなアクセスコントロールを有効にする必要があります。
- クロスリージョンデータソースは同じパーティション内にある必要があります。クロスパーティションアクセス (から などaws-cn) aws はサポートされていません。
- データソースリージョンは、データソース ARN から自動的に抽出されます。`CreateApplication` または `UpdateApplication` APIs では、追加のリージョンパラメータは必要ありません。
- 同一アカウントのクロスリージョンデータソースの場合、 `iamRoleForDataSourceArn` は必要ありません。これは、クロスアカウントデータソースにのみ必要です。
- VPC ドメインの場合は、 を呼び出すときに `SupportedRegions`パラメータにアプリケーションリージョンを含める必要がありますAuthorizeVpcEndpointAccess。を省略すると、同じリージョンのアクセスのみが`--service-options`許可されます。
- IAM Identity Center フローの場合、OpenSearch UI アプリケーションは IAM Identity Center インスタンスと同じリージョンに存在する必要があります。
- クロスリージョンドメインを持つ IAM Identity Center フローの場合、クロスリージョントークンイントロスペクションを有効にする`--identity-center-options`には、ターゲットドメイン`IdentityCenterInstanceRegion`に を含める必要があります。

- サポートされているエンジンバージョン: OpenSearch 1.3 以降。

トラブルシューティング

問題	解決方法
「ドメインにアクセスできません」でアプリケーションの作成が失敗する	ドメインがターゲットリージョンに存在し、きめ細かなアクセスコントロールが有効になっていることを確認します。クロスアカウントシナリオでは、クロスアカウントロールに アクセス`es:DescribeDomain` 許可があり、信頼ポリシーでソースアカウントが許可されていることを確認します。
クロスリージョンの VPC ドメインアクセスが失敗する	に含まれているアプリケーションリージョン`application.opensearchservice.amazonaws.com` を使用して、VPC エンドポイントが に対して承認されていることを確認します`SupportedRegions` 。
IAM ユーザーのデータプレーンアクセスが拒否されました	ターゲットドメインアクセスポリシーが IAM ユーザーまたはロールプリンシパルを許可し、きめ細かなアクセスコントロールバックエンドロールマッピングが適切なアクセス許可を付与していることを確認します。
IAM Identity Center ユーザーのデータプレーンアクセスが拒否されました	バックエンドロールマッピングに IAM Identity Center グループ ID が含まれ、ドメインポリシー`IdentityCenterInstanceRegion` で IAM Identity Center アプリケーションロールが許可され、ドメインの OpenSearch UI アプリケーションと同じリージョンに正しく設定されていることを確認します。
クロスパーティションデータソースが拒否されました	クロスパーティションアクセスはサポートされていません。データソース ARN がアプリケーションと同じパーティションにあることを確認します。
クロスリージョンドメインの IAM Identity Center 認証が失敗する	`IdentityCenterInstanceRegion` IAM Identity Center インスタンスが有効になっている正しいリージョンに が設定されていることを確認します。OpenSearch

問題	解決方法
	UI アプリケーションも同じリージョンに存在する必要があります。

クラスター間検索

Amazon OpenSearch Serverless のクラスター間検索を使用すると、接続された複数のドメインにわたってクエリや集計を実行できます。

Amazon OpenSearch Serverless でのクラスター間検索では、ソースドメインと宛先ドメインの概念を使用します。クラスター間検索リクエストは、ソースドメインから送信されます。レプリケート先ドメインは、クエリ元のレプリケート元ドメインの別の AWS アカウント または AWS リージョン (またはその両方) にあることができます。クラスター間検索を使用すると、同じアカウントの OpenSearch UI に関連付けるようにソースドメインを設定し、ターゲットドメインへの接続を作成できます。その結果、別のアカウントまたはリージョンにある場合でも、ターゲットドメインのデータで OpenSearch UI を使用できます。

Amazon OpenSearch Service との間で転送されるデータには、標準の AWS データ転送料金がかかります。OpenSearch Service ドメイン内のノード間で転送されたデータには課金されません。データ「イン」および「アウト」の料金に関する詳細については、「Amazon EC2 オンデマンド料金」ページの「データ転送」をご参照ください。

OpenSearch UI を別のアカウントまたは別のリージョンのクラスターに関連付けるメカニズムとして、クラスター間検索を使用できます。ドメイン間のリクエストは、ノード間暗号化の一環として、デフォルトで転送中に暗号化されます。

> **Note**
>
> オープンソースの OpenSearch ツールも、クラスター間検索について文書化しています。オープンソースクラスターにおけるオープンソースツールの設定は、マネージド Amazon OpenSearch Serverless ドメインとは大きく異なることに注意してください。
> 特に、Amazon OpenSearch Serverless では、cURL リクエストを使用する代わりに AWS マネジメントコンソール を使用してクラスター間接続を設定します。マネージドサービスは、詳細なアクセス制御に加えて、クラスター間認証に AWS Identity and Access Management (IAM) を使用します。

したがって、ドメインのクロスクラスター検索を設定する際には、オープンソースの OpenSearch ドキュメントではなく、このトピックの内容を使用することをお勧めします。

クラスター間検索を使用する場合の機能の違い

通常のドメインと比較して、クラスター間検索を使用して作成されたターゲットドメインには、以下の機能上の違いと要件があります。

- リモートクラスターに書き込んだり、PUT コマンドを実行したりすることはできません。リモートクラスターへのアクセスは読み取り専用です。
- ソースドメインとターゲットドメインの両方が OpenSearch ドメインである必要があります。OpenSearch UI 用に Elasticsearch ドメインまたはセルフマネージド OpenSearch/Elasticsearch OpenSearch クラスターを接続することはできません。
- ドメインは他のドメインに対して最大 20 の接続を持つことができます。これには、送信接続と受信接続の両方が含まれます。
- ソースドメインは、ターゲットドメインと同じかそれ以上のバージョンの OpenSearch である必要があります。2 つのドメイン間で双方向接続を設定したい場合、両ドメインは同じバージョンである必要があります。接続する前に、両方のドメインを最新バージョンにアップグレードすることをお勧めします。双方向接続の設定後にドメインを更新する必要がある場合は、まず接続を削除してから再作成する必要があります。
- リモートクラスターではカスタム辞書や SQL を使用できません。
- を使用してドメイン CloudFormation を接続することはできません。
- M3 インスタンスとバースト可能 (T2 および T3) インスタンスではクラスター間検索は使用できません。
- クラスター間検索は、Amazon OpenSearch Serverless コレクションでは機能しません。

OpenSearch UI におけるクラスター間検索の前提条件

2 つの OpenSearch ドメインでクラスター間検索をセットアップする前に、ドメインが次の要件を満たしていることを確認してください:

- 両方のドメインで詳細なアクセス制御が有効になっています
- 両ドメインでノード間暗号化が有効化されています

トピック

- クラスター間検索を使用したクロスリージョンおよびクロスアカウントデータアクセスのアクセス許可の設定
- ドメイン間の接続の作成
- クラスター間検索による、リージョン間およびアカウント間のデータアクセスに対するセキュリティ設定のテスト
- 接続を削除する

クラスター間検索を使用したクロスリージョンおよびクロスアカウントデータアクセスのアクセス許可の設定

クラスター間検索リクエストをソースドメインに送信すると、ドメインはそのリクエストをドメインアクセスポリシーに対して評価します。クラスター間検索には、詳細なアクセス制御が必要です。ソースドメインでのオープンアクセスポリシーの例を次に示します。

JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": {
          "AWS": [
            "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111222333444:domain/src-domain/*"
    }
  ]
}
```

> **Note**
>
> パスにリモートインデックスを含める場合は、ドメイン ARN で URI を URL エンコードする必要があります。

例えば、次の ARN 形式を使用します:
`:arn:aws:es:us-east-1:111222333444:domain/my-domain/local_index,dst%3Aremote_index`
次の ARN 形式は使用しません:
`arn:aws:es:us-east-1:111222333444:domain/my-domain/local_index,dst:remote_index.`

詳細なアクセス制御に加えて制限付きアクセスポリシーを使用することを選択した場合、ポリシーは最低限 `es:ESHttpGet` へのアクセスを許可する必要があります。以下に例を示します。

JSON

```
{
"Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111222333444:user/john-doe"
        ]
      },
      "Action": "es:ESHttpGet",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/my-domain/*"
    }
  ]
}
```

ソースドメインの詳細なアクセス制御は、リクエストを評価して、有効な IAM または HTTP の基本認証情報で署名されているかどうかを判断します。その場合、詳細なアクセス制御は次に、ユーザーが検索を実行してデータにアクセスするアクセス許可を持っているかどうかを評価します。

検索のアクセス許可要件は次のとおりです:

- リクエストがターゲットドメイン (例: `dest-alias:dest-index/_search`)) のデータのみを検索する場合、アクセス許可はターゲットドメインでのみ必要です。
- リクエストが両方のドメインのデータを検索する場合 (例: `source-index,dest-alias:dest-index/_search`))、両方のドメインでアクセス許可が必要です。

- 詳細なアクセス制御を使用するには、関連するインデックスに対する標準的な読み取りまたは検索権限に加えて、アクセス許可 indices:admin/shards/search_shards が必要です。

ソースドメインは、ターゲットドメインにリクエストを渡します。ターゲットドメインでは、このリクエストをドメインアクセスポリシーに照らし合わせて評価します。ドキュメントのインデックス作成や標準検索の実行など、OpenSearch UI のすべての機能をサポートするには、完全なアクセス許可を設定する必要があります。次は、ターゲットドメインで推奨されるポリシーの例です:

JSON

```
{
"Version":"2012-10-17",
 "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
         "AWS": [
           "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-2:111222333444:domain/my-destination-
domain/*"
    },
    {
    "Effect": "Allow",
        "Principal": {
    "AWS": "*"
      },
      "Action": "es:ESCrossClusterGet",
      "Resource": "arn:aws:es:us-east-2:111222333444:domain/"
    }
  ]
}
```

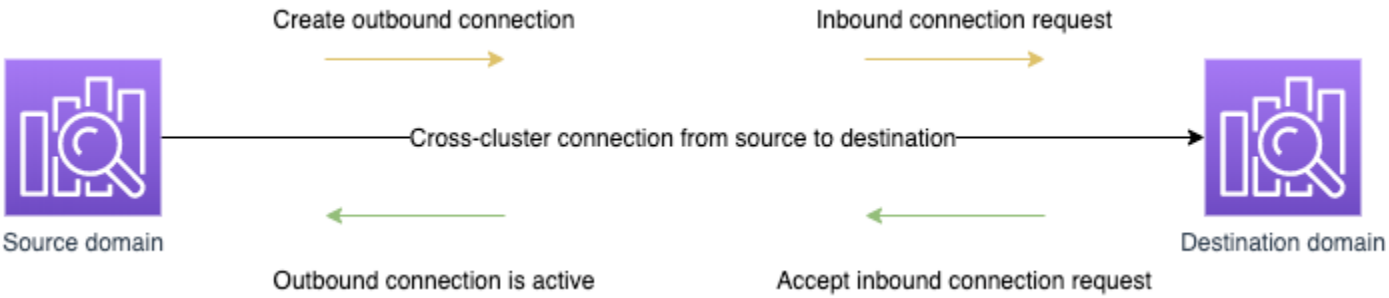
基本的な検索のみを実行する場合、ポリシーの最小要件は、ワイルドカードサポートなしでターゲットドメインに適用される es:ESCrossClusterGet アクセス許可です。例えば、前述のポリシーで

は、ドメイン名を */my-destination-domain* として指定し、*/my-destination-domain/** として指定しません。

この場合、ターゲットドメインが検索を実行し、結果をソースドメインに返します。ソースドメインにより、独自の結果 (存在する場合) とターゲットドメインの結果を組み合わせたものが返されます。

ドメイン間の接続の作成

クロスクラスター検索接続は、ソースドメインからターゲットドメインへの単方向です。つまり、ターゲットドメイン (別のアカウントまたはリージョン) は、OpenSearch UI にローカルなソースドメインをクエリできません。ソースドメインは、ターゲットドメインへのアウトバウンド接続を作成します。ターゲットドメインは、ソースドメインからインバウンド接続リクエストを受信します。



ドメイン間の接続を作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで [ドメイン] を選択します。
3. ソースドメインとして機能するドメインの名前を選択し、[接続] タブを選択します。
4. [アウトバウンド接続] セクションで、[リクエスト] を選択します。
5. [接続のエイリアス] に接続の名前を入力します。接続エイリアスは、ターゲットドメインを選択するために OpenSearch UI で使用されます。
6. [接続モード] では、クラスター間検索またはレプリケーションに [直接接続] を選択します。
7. 検索中に使用できないクラスターをスキップするように接続を指定するには、[使用できないクラスターをスキップ] ボックスを選択します。このオプションを選択すると、1 つ以上のリモートクラスタで障害が発生した場合でも、クラスタ間クエリで部分的な結果が返されるようになります。
8. [ターゲットクラスター] で、[この AWS アカウントのクラスターに接続する] と、[別の AWS アカウントのクラスターに接続する] を選択します。

9. [リモートドメイン ARN] には、クラスターの Amazon リソースネーム (ARN) を入力します。ドメイン ARN は、ドメインの詳細ページの [一般情報] エリアにあります。

   ドメインは次の要件を満たしている必要があります。

   - ARN は形式 `arn:partition:es:regionaccount-id:type/domain-id` である必要があります。例えば、次のようになります。

     ```
     arn:aws:es:us-east-2:111222333444:domain/my-domain
     ```

   - ドメインは、OpenSearch バージョン 1.0 (以降) または Elasticsearch バージョン 6.7 (以降) を使用するように設定する必要があります。
   - 詳細なアクセス制御は、ドメインで有効にする必要があります。
   - ドメインは OpenSearch を実行している必要があります。

10. [リクエスト] を選択します。

クラスター間検索では、まず接続リクエストを検証することで、前提条件が満たされているか確認が行われます。ドメインが互換性がない場合、接続リクエストは `Validation failed` の状態になります。

接続リクエストが正常に検証されると、ターゲットドメインに送信され、承認される必要があります。この承認が行われるまで、接続は `Pending acceptance` の状態のままです。接続要求がターゲットドメインで承諾されると、状態が `Active` に変わり、ターゲットドメインをクエリに使用できるようになります。

ドメインページには、ターゲットドメインについて、ドメインの全体的な正常性とインスタンスの正常性に関する詳細が表示されます。ドメインとの間の接続を柔軟に作成、表示、削除、モニタリングできるのは、ドメイン所有者のみです。

接続が確立されると、接続されたドメインのノード間で流れるトラフィックはすべて暗号化されます。VPC ドメインを非 VPC ドメインに接続し、非 VPC ドメインがインターネットからのトラフィックを受信できるパブリックエンドポイントである場合も、ドメイン間のクラスター間トラフィックは暗号化され、安全に保たれます。

クラスター間検索による、リージョン間およびアカウント間のデータアクセスに対するセキュリティ設定のテスト

クロスクラスター検索によるクロスリージョンおよびクロスアカウントデータアクセスのアクセス許可を設定したら、共同 API 開発用のサードパーティープラットフォームである Postman を使用してセットアップをテストすることをお勧めします。

Postman を使用してセキュリティ設定を行うには

1. ターゲットドメインで、ドキュメントのインデックスを作成します。リクエストの例を次に示します:

```
POST https://dst-domain.us-east-1.es.amazonaws.com/books/_doc/1
{
"Dracula": "Bram Stoker"
}
```

2. ソースドメインからこのインデックスを照会するには、クエリ内にターゲットドメインの接続エイリアスを含めます。接続エイリアスは、ドメインダッシュボードの [接続] タブで確認できます。以下は、リクエストと切り捨てられたレスポンスの例です。

```
GET https://src-domain.us-east-1.es.amazonaws.com/connection_alias:books/_search
{
...
  "hits": [
{
"_index": "source-destination:books",
  "_type": "_doc",
  "_id": "1",
  "_score": 1,
  "_source": {
"Dracula": "Bram Stoker"
  }
}
  ]
}
```

3. (オプション) 1 回の検索で複数のドメインを含む設定を作成できます。例えば、次のように設定したとします。

   接続エイリアス名を `cluster_b` とする domain-a から domain-b への接続

接続エイリアス名を cluster_c とする domain-a から domain-c への接続

この場合、検索にはコンテンツ domain-a、domain-b、および domain-c が含まれます。以下はリクエストとレスポンスのサンプルです:

リクエスト

```
GET https://src-domain.us-east-1.es.amazonaws.com/
local_index,cluster_b:b_index,cluster_c:c_index/_search
{
  "query": {
"match": {
  "user": "domino"
}
  }
}
```

レスポンス:

```
{
"took": 150,
  "timed_out": false,
  "_shards": {
"total": 3,
"successful": 3,
"failed": 0,
"skipped": 0
  },
  "_clusters": {
"total": 3,
"successful": 3,
"skipped": 0
  },
  "hits": {
"total": 3,
"max_score": 1,
"hits": [
  {
"_index": "local_index",
    "_type": "_doc",
    "_id": "0",
    "_score": 1,
```

```
    "_source": {
"user": "domino",
      "message": "This is message 1",
      "likes": 0
    }
  },
  {
"_index": "cluster_b:b_index",
    "_type": "_doc",
    "_id": "0",
    "_score": 2,
    "_source": {
"user": "domino",
      "message": "This is message 2",
      "likes": 0
    }
  },
  {
"_index": "cluster_c:c_index",
    "_type": "_doc",
    "_id": "0",
    "_score": 3,
    "_source": {
"user": "domino",
      "message": "This is message 3",
      "likes": 0
    }
  }
]
  }
}
```

接続のセットアップで使用できないクラスターをスキップしなかった場合、検索リクエストを正常に実行するには、検索するすべての宛先クラスターが使用可能である必要があります。そうでない場合は、リクエスト全体が失敗します ドメインのうち、いずれか 1 つだけが使用できない場合でも、検索結果は返されません。

接続を削除する

接続を削除すると、ターゲットドメインのクラスター間検索オペレーションが停止します。

接続を削除するには、ソースドメインまたはターゲットドメインのいずれかで次の手順を実行できます。接続を削除した後も、15 日間は `Deleted` のステータスで表示されたままになります。

アクティブなクラスター間接続のあるドメインは削除できません。ドメインを削除するには、まずそのドメインからすべての受信接続と送信接続を削除します。この操作は、ドメインを削除する前にクラスター間ドメインのユーザーを考慮するために行います。

接続を削除する方法

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションペインで [ドメイン] を選択します。
3. 削除するドメインの名前を選択し、[接続] タブを選択します。
4. 削除する接続の名前を選択します。
5. [削除] を選択し、削除を確定します。

VPC エンドポイントからの OpenSearch UI へのアクセスの管理

AWS PrivateLinkを使用して、VPC と OpenSearch UI の間にプライベート接続を確立できます。この接続を使用すると、同じ VPC 内にあるかのように OpenSearch UI アプリケーションにアクセスできます。これにより、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect を設定して接続を確立する必要はありません。VPC のインスタンスは、パブリック IP アドレスがなくても OpenSearch UI にアクセスできます。

このプライベート接続を確立するには、まず を使用するインターフェイスエンドポイントを作成します AWS PrivateLink。インターフェースエンドポイントに指定した各サブネットには、エンドポイント用のネットワークインターフェースが自動的に作成されます。これらはリクエスター管理型ネットワークインターフェースであり、OpenSearch UI アプリケーション宛てのトラフィックのエントリポイントとして機能します。

VPC と OpenSearch UI 間のプライベート接続の作成

AWS マネジメントコンソール または を使用して、VPC から OpenSearch UI にアクセスするためのプライベート接続を作成できます AWS CLI。

VPC と OpenSearch UI 間のプライベート接続の作成 (コンソール)

コンソールを使用して VPC と OpenSearch UI 間のプライベート接続を作成するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. 左側のナビゲーションの [サーバーレス] で、[VPC エンドポイント] を選択します。
3. [Create VPC endpoint] (VPC エンドポイントの作成) を選択します。
4. [名前] に、エンドポイントの名前を入力します。
5. [VPC] で OpenSearch UI アプリケーションにアクセスする VPC を選択します。
6. [サブネット] で、OpenSearch UI アプリケーションにアクセスするサブネットを 1 つ選択します。

> **Note**
>
> エンドポイントの IP アドレスと DNS タイプはサブネットタイプに基づいています:
>
> - デュアルスタック: すべてのサブネットに IPv4 および IPv6 のアドレス範囲が設定されている場合。
> - IPv6: すべてのサブネットが IPv6 のみのサブネットの場合。
> - IPv4: すべてのサブネットに IPv4 アドレス範囲がある場合。

7. [セキュリティグループ] で、エンドポイントネットワークインターフェイスに関連付けるセキュリティグループを 1 つ以上選択します。

> **Note**
>
> このステップでは、エンドポイントに対して承認するインバウンドトラフィックのポート、プロトコル、およびソースを制限しています。OpenSearch UI アプリケーションとの通信に VPC エンドポイントを使用するリソースがエンドポイントのネットワークインターフェイスと通信できるように、セキュリティグループルールが設定されていることを確認します。

8. 8. エンドポイントの作成 を選択します。

VPC と OpenSearch UI 間のプライベート接続の作成 (AWS CLI)

を使用して VPC と OpenSearch UI の間にプライベート接続を作成するには AWS CLI

以下のコマンドを実行してください。*#########*を、ユーザー自身の情報に置き換えます。

```
aws opensearchserverless create-vpc-endpoint \
    --region region \
    --endpoint endpoint \
    --name vpc_endpoint_name \
    --vpc-id vpc_id \
    --subnet-ids subnet_ids
```

OpenSearch UI アプリケーションへのアクセスを許可するように VPC エンドポイントポリシーを更新する

プライベート接続を作成したら、VPC エンドポイントポリシーを更新して、アプリケーション ID を指定して VPC エンドポイントポリシーの OpenSearch UI アプリケーションへのアクセスを許可します。

VPC エンドポイントポリシーの更新ついては、「AWS PrivateLink ガイド」の「VPC エンドポイントポリシーの更新」を参照してください。

VPC エンドポイントポリシーに次のステートメントが含まれていることを確認します。*#########*を、ユーザー自身の情報に置き換えます。

```
{
    "Statement": [{
        "Action": ["opensearch:*"],
        "Effect": "Allow",
        "Principal": "*",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "opensearch:ApplicationId": ["opensearch-ui-application-id"]
            }
        }
    }]
}
```

VPC エンドポイントポリシーでの OpenSearch UI へのアクセスの取り消し

OpenSearch UI では、ユーザーが VPC からアプリケーションにアクセスできるようにするには、VPC エンドポイントポリシーに明示的なアクセス許可が必要です。ユーザーが VPC から OpenSearch UI にアクセスする必要がなくなった場合は、エンドポイントポリシーのアクセス許可を削除できます。その後、OpenSearch UI にアクセスしようとすると、ユーザーに `403 forbidden` エラーメッセージが表示されます。

VPC エンドポイントポリシーの更新ついては、「AWS PrivateLink ガイド」の「VPC エンドポイントポリシーの更新」を参照してください。

以下は、VPC からの UI アプリケーションへのアクセスを拒否する VPC エンドポイントポリシーの例です。

```
{
    "Statement": [{
        "Action": ["opensearch:*"],
        "Effect": "Allow",
        "Principal": "*",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "opensearch:ApplicationId": [""]
            }
        }
    }]
}
```

保存されたオブジェクトを OpenSearch Dashboards から OpenSearch UI に移行する

OpenSearch Dashboards に既存のダッシュボード、ビジュアライゼーション、インデックスパターン、およびその他の保存されたオブジェクトがある場合は、OpenSearch UI でそれらを移行して再利用できます。

OpenSearch UI に移行するメリット:

- 高可用性 — OpenSearch UI は でホスト AWS クラウド され、ドメインのアップグレードとメンテナンス中は引き続き使用できますが、OpenSearch Dashboards はドメイン内でホストされ、一時的に使用できなくなります。
- 複数のデータソース – OpenSearch UI は、OpenSearch ドメイン、サーバーレスコレクション、Amazon OpenSearch S3 および Amazon CloudWatch とのデータ接続など、複数のデータソースにまたがる 1 つの統合ペインを提供できますが、各 OpenSearch Dashboards は 1 つのドメインまたはコレクションでのみ使用できます。 Amazon S3
- AI Assistant や WorkSpaces などの追加機能は、OpenSearch UI で利用できます。詳細については、「」を参照してくださいOpenSearch UI。

トピック

- 移行の概要
- 前提条件
- ステップ 1: OpenSearch Dashboards から保存されたオブジェクトをエクスポートする
- ステップ 2: 保存したオブジェクトを OpenSearch UI にインポートする

移行の概要

移行プロセスは、以下の大まかなステップで構成されます。

1. OpenSearch Dashboards から保存済みオブジェクトをエクスポート – OpenSearch Dashboards の保存済みオブジェクト管理 UI またはエクスポート API を使用して、ダッシュボード、ビジュアライゼーション、インデックスパターン、およびその他のオブジェクトを改行区切り JSON (NDJSON) ファイルとしてダウンロードします。
2. OpenSearch UI アプリケーションとワークスペースを作成する – まだ作成していない場合は、OpenSearch UI アプリケーションとワークスペースを作成して、インポートされたオブジェクトを受信します。
3. OpenSearch UI にデータソースを登録する – OpenSearch ドメインを OpenSearch UI アプリケーションに関連付け、ワークスペース内のデータソースとして登録します。インポートされたオブジェクトのインデックスパターンは、このデータソースを参照します。
4. OpenSearch UI への保存済みオブジェクトのインポート – OpenSearch UI の保存済みオブジェクト管理 UI またはインポート API を使用して、NDJSON ファイルをターゲットワークスペースにアップロードします。

5. インポートされたオブジェクトを検証する – OpenSearch UI でダッシュボードとビジュアライゼーションを開き、それらが正しく表示され、関連するドメインまたはコレクションからデータが流れていることを確認します。

前提条件

移行する前に、次の前提条件を確認してください。

- Amazon OpenSearch Service および OpenSearch APIs を呼び出すために必要な OpenSearch IAM アクセス許可があります。詳細については、「the section called “Amazon OpenSearch Service アプリケーションを作成するために必要なアクセス許可”」を参照してください。
- ドメインまたはコレクション、および移行する OpenSearch Dashboards にアクセスできます。
- OpenSearch UI アプリケーションを作成しました。アプリケーションとワークスペースの作成については、「」を参照してくださいthe section called “開始方法”。
- OpenSearch UI アプリケーションに同じドメインまたはコレクションを関連付けています。データソースの関連付けについては、「」を参照してくださいthe section called “データソースの関連付けと VPC アクセス許可の管理”。

> **Note**
>
> OpenSearch UI は OpenSearch バージョン 1.3 以降のみをサポートしています。保存したオブジェクトを移行する前に、OpenSearch ドメインがバージョン 1.3 以降を実行していることを確認します。

ステップ 1: OpenSearch Dashboards から保存されたオブジェクトをエクスポートする

管理 UI またはエクスポート API を使用して、保存されたオブジェクトを OpenSearch Dashboards からエクスポートします。エクスポートでは、選択したすべての保存済みオブジェクトタイプとその依存関係を含む改行区切りの JSON (NDJSON) ファイルが生成されます。

トピック

- OpenSearch Dashboards で手動でエクスポートする
- API を使用したエクスポート

OpenSearch Dashboards で手動でエクスポートする

OpenSearch Dashboards 管理 UI を使用して保存済みオブジェクトをエクスポートするには

1. OpenSearch Dashboards インスタンスを開きます。
2. 左側のナビゲーションパネルで、管理を選択します。
3. ダッシュボード管理で、保存済みオブジェクトを選択します。
4. エクスポートする保存済みオブジェクトを選択します。特定のタイプのすべてのオブジェクトをエクスポートするには、検索バーを使用してタイプでフィルタリングします。すべてのオブジェクトをエクスポートするには、テーブルヘッダーのチェックボックスをオンにします。
5. [エクスポート] を選択します。
6. 保存済みオブジェクトのエクスポートダイアログボックスで、関連するオブジェクトを含めるが選択されていることを確認します。このオプションには、ビジュアライゼーションによって参照されるインデックスパターンなど、選択した保存済みオブジェクトが依存するすべてのオブジェクトが含まれます。このオプションは、依存関係を手動で管理する場合にのみクリアします。
7. エクスポート を選択して、ローカルマシンに`.ndjson`ファイルをダウンロードします。

> **Tip**
>
> 関連オブジェクトを含めるを選択すると、エクスポートされた NDJSON ファイルには、依存するインデックスパターン、ビジュアライゼーション、検索オブジェクトなど、選択したダッシュボードとビジュアライゼーションのレンダリングに必要なすべての保存済みオブジェクトが含まれます。これにより、インポートステップが簡素化され、参照エラーの欠落を回避できます。

API を使用したエクスポート

OpenSearch Dashboards の保存済みオブジェクトのエクスポート API を使用して、保存済みオブジェクトをプログラムでエクスポートできます。これは、移行を自動化したり、エクスポートステップを CI/CD パイプラインに統合したりするのに役立ちます。

> **Note**
>
> OpenSearch ドメインできめ細かなアクセスコントロールが有効になっている場合は、エクスポートリクエストで認証情報を指定する必要があります。ユーザー名とパスワードで -uフ

ラグを追加して、HTTP 基本認証を使用します。認証オプションの詳細については、「the section called “きめ細かなアクセスコントロール”」を参照してください。

次の の例では、関連するオブジェクトを含むすべてのダッシュボードをエクスポートします。*####*を、ユーザー自身の情報に置き換えます。

```
curl -X POST \
    "https://dashboards-endpoint/_dashboards/api/saved_objects/_export" \
    -u 'master-username:master-password' \
    -H "Content-Type: application/json" \
    -H "osd-xsrf: true" \
    -d '{
        "type": ["dashboard", "visualization", "index-pattern", "search"],
        "includeReferencesDeep": true,
        "excludeExportDetails": false
    }' \
    -o saved-objects-export.ndjson
```

ドメインできめ細かなアクセスコントロールが有効になっていない場合は、 -uフラグを省略できます。

特定の保存済みオブジェクトを ID でエクスポートするには、 の代わりに objectsパラメータを使用しますtype。

```
curl -X POST \
    "https://dashboards-endpoint/_dashboards/api/saved_objects/_export" \
    -u 'master-username:master-password' \
    -H "Content-Type: application/json" \
    -H "osd-xsrf: true" \
    -d '{
        "objects": [
            {"type": "dashboard", "id": "dashboard-id"},
            {"type": "visualization", "id": "visualization-id"}
        ],
        "includeReferencesDeep": true
    }' \
    -o saved-objects-export.ndjson
```

> **Note**
>
> 保存済みオブジェクト IDs を検索するには、保存済みオブジェクト API を使用して、特定のタイプのすべてのオブジェクトを一覧表示します。次の の例では、すべてのダッシュボードを一覧表示します。
>
> ```
> curl -X GET \
>     "https://dashboards-endpoint/_dashboards/api/saved_objects/_find?
> type=dashboard" \
>     -u 'master-username:master-password'
> ```
>
> レスポンスには、保存された各オブジェクトの ID が含まれます。OpenSearch Dashboards でオブジェクトを表示するときに、ブラウザ URL で ID を見つけることもできます。

ステップ 2: 保存したオブジェクトを OpenSearch UI にインポートする

保存したオブジェクトをエクスポートした後、NDJSON ファイルを OpenSearch UI に手動でインポートするか、API 経由でインポートできます。

トピック

- OpenSearch UI に手動でインポートする
- API 経由でのインポート

OpenSearch UI に手動でインポートする

OpenSearch UI 管理 UI を使用して保存済みオブジェクトをインポートするには

1. OpenSearch UI アプリケーションを開き、ターゲットワークスペースに移動します。
2. ワークスペースで、上部のナビゲーションからアセットを選択するか、ワークスペースアセットページに移動します。
3. Import を選択して、Import assets ダイアログを開きます。
4. ファイルの選択 を選択し、OpenSearch Dashboards からエクスポートした.ndjsonファイルを選択します。
5. 競合管理では、次のいずれかを選択します。

- 一意の IDs を使用して新しいアセットを作成する (デフォルト) – インポートされたすべてのオブジェクトに新しい IDs を生成し、既存のアセットとの競合を回避します。
- 既存のアセットをチェックする – 既存のオブジェクトとの競合をチェックします。選択したら、次のいずれかのサブオプションを選択します。
  - 競合を自動的に上書きする – 同じ ID を持つ既存のアセットは自動的に置き換えられます。
  - 競合に対するアクションのリクエスト – 各競合を個別に解決するように求められます。

6. [インポート] を選択します。
7. インポートの概要を確認します。

API 経由でのインポート

AWS 署名バージョン 4 認証で API を使用して保存済みオブジェクトをインポートするには、まずデータソース ID を取得し、インポートリクエストで使用する必要があります。*#########*を、ユーザー自身の情報に置き換えます。

ステップ 1: ワークスペースのデータソース ID を取得する:

```
curl -X GET \
    "https://opensearch-ui-endpoint/w/workspace-id/api/saved_objects/_find?type=data-
source" \
    --aws-sigv4 "aws:amz:region:opensearch" \
    --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
    -H "x-amz-security-token: $AWS_SESSION_TOKEN" \
    -H "osd-xsrf: true"
```

> **Note**
>
> レスポンスにはデータソース ID が含まれます。OpenSearch UI でデータソースを表示するときに、ブラウザ URL でデータソース ID を確認することもできます。

ステップ 2: ステップ 1 のデータソース ID を使用して保存されたオブジェクトをインポートします。

```
curl -X POST \
    "https://opensearch-ui-endpoint/w/workspace-id/api/saved_objects/_import?
overwrite=true&dataSourceId=data-source-id" \
```

```
    --aws-sigv4 "aws:amz:region:opensearch" \
    --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
    -H "x-amz-security-token: $AWS_SESSION_TOKEN" \
    -H "osd-xsrf: true" \
    -F "file=@saved-objects-export.ndjson"
```

> **Note**
>
> これらの例では、curl の組み込み`--aws-sigv4`オプション (curl 7.75 以降で利用可能) を使用してリクエストに署名します。コマンドを実行する前に、認証情報を環境変数として設定します AWS 。`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、および`AWS_SESSION_TOKEN` (一時的な認証情報を使用している場合) 。

OpenSearch UI エンドポイントとクォータ

プログラムで AWS サービスに接続するには、 エンドポイントを使用します。サービスクォータ (制限とも呼ばれます) は、 AWS アカウントのサービスリソースまたはオペレーションの最大数です。

Amazon OpenSearch UI は、OpenSearch Dashboards 用の次世代 OpenSearch ユーザーインターフェイスです。OpenSearch ダッシュボードにアクセスするためのエンドポイントを提供します。このトピックを使用して、OpenSearch UI のサービスエンドポイントとサービスクォータを検索します。

他の OpenSearch サービスの詳細については、「サービスエンドポイントとクォータ」を参照してください。

OpenSearch UI エンドポイント

OpenSearch UI は次のリージョンで利用できます。

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (ムンバイ)	ap-south-1	opensearch.ap-south-1.amazonaws.com es.ap-south-1.amazonaws.com	HTTPS HTTPS
欧州 (パリ)	eu-west-3	opensearch.eu-west-3.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
		es.eu-west-3.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	opensearch.us-east-2.amazonaws.com es.us-east-2.amazonaws.com	HTTPS HTTPS
欧州 (アイルランド)	eu-west-1	opensearch.eu-west-1.amazonaws.com es.eu-west-1.amazonaws.com	HTTPS HTTPS
欧州 (フランクフルト)	eu-central-1	opensearch.eu-central-1.amazonaws.com es.eu-central-1.amazonaws.com	HTTPS HTTPS
南米 (サンパウロ)	sa-east-1	opensearch.sa-east-1.amazonaws.com es.sa-east-1.amazonaws.com	HTTPS HTTPS
米国東部 (バージニア北部)	us-east-1	opensearch.us-east-1.amazonaws.com es.us-east-1.amazonaws.com	HTTPS HTTPS
欧州 (ロンドン)	eu-west-2	opensearch.eu-west-2.amazonaws.com es.eu-west-2.amazonaws.com	HTTPS HTTPS
アジアパシフィック (東京)	ap-northeast-1	opensearch.ap-northeast-1.amazonaws.com es.ap-northeast-1.amazonaws.com	HTTPS HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
米国西部 (オレゴン)	us-west-2	opensearch.us-west-2.amazonaws.com es.us-west-2.amazonaws.com	HTTPS HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	opensearch.ap-southeast-1.amazonaws.com es.ap-southeast-1.amazonaws.com	HTTPS HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	opensearch.ap-southeast-2.amazonaws.com es.ap-southeast-2.amazonaws.com	HTTPS HTTPS
カナダ (中部)	ca-central-1	opensearch.ca-central-1.amazonaws.com es.ca-central-1.amazonaws.com	HTTPS HTTPS
欧州 (ストックホルム)	eu-north-1	opensearch.eu-north-1.amazonaws.com es.eu-north-1.amazonaws.com	HTTPS HTTPS
アジアパシフィック (香港)	ap-east-1	opensearch.ap-east-1.amazonaws.com es.ap-east-1.amazonaws.com	HTTPS HTTPS
アジアパシフィック (ソウル)	ap-northeast-2	opensearch.ap-northeast-2.amazonaws.com es.ap-northeast-2.amazonaws.com	HTTPS HTTPS
アジアパシフィック (大阪)	ap-northeast-3	opensearch.ap-northeast-3.amazonaws.com es.ap-northeast-3.amazonaws.com	HTTPS HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (ハイデラバード)	ap-south-2	opensearch.ap-south-2.amazonaws.com es.ap-south-2.amazonaws.com	HTTPS HTTPS
欧州 (スペイン)	eu-south-2	opensearch.eu-south-2.amazonaws.com es.eu-south-2.amazonaws.com	HTTPS HTTPS
米国西部 (北カリフォルニア)	us-west-1	opensearch.us-west-1.amazonaws.com es.us-west-1.amazonaws.com	HTTPS HTTPS
欧州 (チューリッヒ)	eu-central-2	opensearch.eu-central-2.amazonaws.com es.eu-central-2.amazonaws.com	HTTPS HTTPS
ヨーロッパ (ミラノ)	eu-south-1	opensearch.eu-south-1.amazonaws.com es.eu-south-1.amazonaws.com	HTTPS HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	es.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	es.us-gov-west-1.amazonaws.com	HTTPS

OpenSearch UI サービスクォータ

AWS アカウントには、OpenSearch UI リソースに関連する次のクォータがあります。

名前	デフォルト	引き上げ可能	注意事項
アカウントごと、リージョンごとの OpenSearch UI アプリケーション	30	はい	リージョンごとにアカウントごとに作成できる OpenSearch UI アプリケーションの最大数。 サービスクォータを使用して制限を 50 に引き上げ、自動的に承認を受けることができます。上限の引き上げをリクエストするには、サポートチケットを送信します。

Amazon OpenSearch Service でのインデックス管理

Amazon OpenSearch Service にデータを追加した後には、そのデータのインデックスを再作成したり、インデックスのエイリアスを設定したり、インデックスをよりコスト効率の高いストレージに移動したり、インデックスを完全に削除したりすることが必要になる場合がよくあります。この章では、ストレージ階層化ソリューション、インデックス状態管理、およびその他のインデックスオペレーションについて説明します。OpenSearch インデックス API の詳細については、「OpenSearch ドキュメント」を参照してください。

トピック

- Amazon OpenSearch Service ドメインの OpenSearch 最適化インスタンス
- Amazon OpenSearch Service の多層ストレージ
- Amazon OpenSearch Service の UltraWarm ストレージ
- Amazon OpenSearch Service 用コールドストレージ
- Amazon OpenSearch Service でのインデックスステート管理
- インデックスロールアップにより Amazon OpenSearch Service のインデックスを要約する
- Amazon OpenSearch Service でのインデックスの変換
- Amazon OpenSearch Service のクラスター間レプリケーション
- リモート再インデックスを使用した Amazon OpenSearch Service インデックスの移行
- Data Streams を用いた Amazon OpenSearch Service での時系列データの管理

Amazon OpenSearch Service ドメインの OpenSearch 最適化インスタンス

Amazon OpenSearch Service の OpenSearch 最適化インスタンスファミリーは、大量のデータを保存するための費用対効果の高いソリューションです。OpenSearch 最適化インスタンスを持つドメインは、ローカルストレージをプライマリとして使用し、データは到着時に Amazon S3 に同期的にコピーされます。このストレージ構造により、高い耐久性を備えたインデックス作成スループットが改善します。OR1、OR2、OM2 は Amazon Elastic Block Store (Amazon EBS) gp3または`io1`ボリュームをローカルで使用しますが、OI2 インスタンスはローカル NVMe ディスクを使用します。また、OpenSearch 最適化インスタンスファミリーは、障害発生時の自動データ復旧をサポートします。OpenSearch 最適化インスタンスタイプのオプションについては、「the section called “現行世代のインスタンスタイプ”」を参照してください。

ログ分析、オブザーバビリティ、セキュリティ分析などの大規模な運用分析ワークロードのインデックス作成を行っている場合は、OpenSearch 最適化インスタンスによるパフォーマンスやコンピューティング効率の向上からメリットを得ることができるでしょう。さらに、OpenSearch 最適化インスタンスが提供する自動データ復旧は、ドメインの全体的な信頼性を向上させます。

OpenSearch Service は、ストレージ関連の OpenSearch 最適化メトリクスを Amazon CloudWatch に送信します。使用可能なメトリクスのリストについては、??? を参照してください。

OpenSearch 最適化インスタンスはオンデマンドまたはリザーブドインスタンス料金で利用でき、Amazon EBS と Amazon S3 でプロビジョニングされたインスタンスとストレージについて時間単位の料金が発生します。

トピック

- 制限事項
- 取り込みスループットを向上させるための調整
- OpenSearch 最適化インスタンスと他のインスタンスの違い
- OpenSearch 最適化インスタンスと UltraWarm インスタンスの違い
- OpenSearch 最適化インスタンスを使用したドメインのプロビジョニング

制限事項

ドメインに OpenSearch 最適化インスタンスを使用する場合は、次の制限を考慮してください。

- 新しく作成されたドメインは、OpenSearch バージョン 2.11 以降を実行している必要があります。
- 既存のドメインは OpenSearch バージョン 2.15 以降を実行している必要があります。
- ドメインでは、保管時の暗号化が有効になっている必要があります。詳細については、「???」を参照してください。
- ドメインが専用マスターノードを使用している場合は、Graviton インスタンスを使用する必要があります。専用マスターノードの詳細については、「???」を参照してください。
- OpenSearch 最適化インスタンスのインデックスの更新間隔は 10 秒以上である必要があります。OpenSearch 最適化インスタンスのデフォルトの更新間隔は 10 秒です。
- OpenSearch 最適化 (OR1) インスタンスへの移行は元に戻せません。OR1 インスタンスを使用するクラスターを標準インスタンスタイプに戻すことはできません。先に進む前に、移行を慎重に計画してください。

取り込みスループットを向上させるための調整

OpenSearch 最適化インスタンスから最適なインデックス作成スループットを得るには、以下を実行することをお勧めします。

- 大きめのバルクサイズを使用して、バッファ使用率を向上させましょう。推奨サイズは 10 MB です。
- 複数のクライアントを使用して並列処理のパフォーマンスを向上させます。
- アクティブなプライマリシャードの数をデータノードの数に合わせて設定し、リソース使用率を最大化します。

OpenSearch 最適化インスタンスと他のインスタンスの違い

OpenSearch 最適化インスタンスは、以下の点で他のインスタンスとは異なります。

- OpenSearch 最適化インスタンスの場合、インデックス作成はプライマリシャードでのみ実行されます。
- OpenSearch 最適化インスタンスがレプリカで設定されている場合、インデックス作成レートが実際に設定されているよりも低くなることがあります。例えば、プライマリシャードが 1 つ、レプリカシャードが 1 つある場合、実際のインデックス作成レートが 2,000 であっても、表示されるインデックス作成レートは 1,000 となる可能性があります。
- OpenSearch 最適化インスタンスは、リモートソースに送信する前にバッファオペレーションを実行します。その結果、取り込みのレイテンシーが長くなります。

  > **Note**
  >
  > `IndexingLatency` メトリクスはトランスログを同期する時間を反映しないため、影響を受けません。

- レプリカシャードは、プライマリシャードの数秒後になる場合があります。`ReplicationLagMaxTime` Amazon CloudWatch メトリクスを使用してラグをモニタリングすることができます。

OpenSearch 最適化インスタンスと UltraWarm インスタンスの違い

OpenSearch Service には、大量の読み取り専用データをコストパフォーマンスに優れた方法で保存できる UltraWarm インスタンスが用意されています。OpenSearch 最適化インスタンスと

UltraWarm インスタンスのいずれも、データを Amazon EBS にローカルで、Amazon S3 にリモートで保存します。ただし、OpenSearch 最適化インスタンスと UltraWarm インスタンスはいくつかの重要な点で異なります。

- OpenSearch 最適化インスタンスは、ローカルストアとリモートストアの両方にデータのコピーを保持します。UltraWarm インスタンスでは、ストレージコストを削減するために、データは主にリモートストアに保持されます。使用状況パターンに応じて、データをローカルストレージに移動できます。
- OpenSearch 最適化インスタンスはアクティブであり、読み取りと書き込み両方のオペレーションを受け入れますが、UltraWarm インスタンスのデータは、手動でホットストレージに戻すまで読み取り専用となります。
- UltraWarm は、データの耐久性においてインデックススナップショットに依存しています。一方、OpenSearch 最適化インスタンスは、バックグラウンドでレプリケーションとリカバリを実行します。赤いインデックスの場合、OpenSearch 最適化インスタンスは Amazon S3 のリモートストレージから欠落しているシャードを自動的に復元します。復旧時間は、復旧するデータの量によって異なります。

UltraWarm ストレージの詳細については、「???」を参照してください。

OpenSearch 最適化インスタンスを使用したドメインのプロビジョニング

AWS マネジメントコンソール または AWS Command Line Interface () を使用して新しいドメインを作成するときに、データノードの OpenSearch 最適化インスタンスを選択できますAWS CLI。既存のツールを使用して、データのインデックスを作成し、クエリを実行できます。

コンソール

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで [Domains] (ドメイン) を選択します。
3. [ドメインを作成] をクリックします。
4. [データノードの数] セクションで、[インスタンスファミリー] メニューを展開し、[OpenSearch 最適化] を選択します。
5. インスタンスタイプとその他のストレージ設定を選択します。
6. [暗号化] セクションで、[保管時のデータの暗号化を有効にする] が選択されていることを確認します。

7. ドメインの残りの部分を設定し、[作成] を選択します。

AWS CLI

を使用して OpenSearch 最適化ストレージを使用するドメインをプロビジョニングするには AWS CLI、 で特定のインスタンスタイプサイズ (OR1、OR2、OM2、OI2 など) の値を指定する必要がありますInstanceType。

次の例では、サイズが 2xlarge である OR1 インスタンスを使用するドメインを作成し、保管時の暗号化を有効にします。

```
aws opensearch create-domain \
  --domain-name test-domain \
  --engine-version OpenSearch_2.11 \
  --cluster-config
 "InstanceType=or1.2xlarge.search,InstanceCount=3,DedicatedMasterEnabled=true,DedicatedMasterTy
 \
  --ebs-options "EBSEnabled=true,VolumeType=gp3,VolumeSize=200" \
  --encryption-at-rest-options Enabled=true \
  --advanced-security-options
 "Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions={MasterUserName=test-
user,MasterUserPassword=test-password}" \
  --node-to-node-encryption-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true \
  --access-policies '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"*"},"Action":"es:*","Resource":"arn:aws:es:us-
east-1:account-id:domain/test-domain/*"}]}'
```

次の例では、サイズの OI2 インスタンスを使用してドメインを作成しlarge、保管時の暗号化を有効にします。OI2 インスタンスはローカル NVMe ストレージを使用するため、EBS 設定は必要ありません。

```
aws opensearch create-domain \
  --domain-name test-domain-oi2 \
  --engine-version OpenSearch_2.11 \
  --cluster-config
 "InstanceType=oi2.2xlarge.search,InstanceCount=3,DedicatedMasterEnabled=true,DedicatedMasterTy
 \
  --encryption-at-rest-options Enabled=true \
  --advanced-security-options
 "Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions={MasterUserName=test-
user,MasterUserPassword=test-password}" \
```

```
  --node-to-node-encryption-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true \
  --access-policies '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"*"},"Action":"es:*","Resource":"arn:aws:es:us-
east-1:account-id:domain/test-domain-oi2/*"}]}'
```

Amazon OpenSearch Service の多層ストレージ

Amazon OpenSearch Service の多層ストレージは、さまざまなストレージ階層にわたってデータを管理することで、パフォーマンスとコストの両方を最適化するインテリジェントなデータ管理ソリューションです。このアーキテクチャにより、組織はアクセス頻度の低いデータをよりコスト効率の高いウォームストレージに移動しながら、アクセス頻度の高いデータを高性能ホットストレージに保持することで、パフォーマンスとコストのトレードオフを効率的にバランスさせることができます。

Amazon OpenSearch Service には、ホット/ウォームストレージ階層用の 2 つのアーキテクチャオプションがあります。

- OpenSearch 多層ストレージアーキテクチャ
  - Amazon S3 をローカルインスタンスストレージと組み合わせる
  - OpenSearch Optimized Instances を利用
  - ウォーム階層での書き込みオペレーションをサポート
  - ホット階層とウォーム階層間のシームレスなデータ移行をサポート
  - OpenSearch 3.3 以降で利用可能
  - Cold Tier をサポートしていません
- UltraWarm ベースのアーキテクチャ
  - Amazon S3 をローカルインスタンスストレージと組み合わせる
  - UltraWarm インスタンスを使用
  - 読み取り専用のウォーム階層ワークロード向けに最適化
  - Elasticsearch バージョン 6.8 以降およびすべての OpenSearch バージョンで利用可能
  - コールド階層をサポート

> **Note**
>
> このドキュメントでは、多層アーキテクチャのみに焦点を当てています。Ultrawarm ストレージアーキテクチャについては、「Ultrawarm および Cold Storage」を参照してください。

多層ストレージアーキテクチャ

トピック

- 主な利点
- 前提条件
- 制限事項
- 主要事項
- 多層ドメインの作成
- 階層移行の管理
- セキュリティ設定
- ベストプラクティス
- メトリクスのモニタリング

主な利点

- Writable Warm: ウォームインデックスの書き込みオペレーションをサポート
- シームレスな移行: ストレージ層間のデータのシームレスな移動
- コスト最適化: 非アクティブデータをコスト効率の高いウォームストレージに移動することでストレージコストを削減
- パフォーマンスの向上: ホット階層内の頻繁にアクセスされるデータに対して高いパフォーマンスを維持します。
- 柔軟なデータ管理: ワークロード要件に最適なアーキテクチャを選択する
- 自動管理: ストレージ階層全体のデータライフサイクル管理を簡素化

前提条件

- エンジンバージョン: OpenSearch 3.3 以降
- インスタンスファミリー:
  - ホットノード: OR1, OR2, OM2、または OI2
  - ウォームノード: OI2
- セキュリティ: Node-to-node暗号化、保管時の暗号化、HTTPS の適用

制限事項

- Ultrawarm がまだ有効になっていない OpenSearch 最適化インスタンスを持つすべてのドメインで動作します
- コールド階層のサポートなし

主要事項

- ホットからウォームへの移行では、マルチ層アーキテクチャでの強制マージはトリガーされません。必要に応じて、ユーザーはインデックス管理ポリシーを使用して強制マージをオーケストレーションできます。
- インデックス作成とは別に、ウォームノードはバックグラウンドマージオペレーションも実行するようになりました (ホットに似ています)。
- ウォームインデックスに対するすべての検索リクエストはプライマリシャードにルーティングされ、レプリカはプライマリシャードがダウンしている場合にのみ読み取りを提供します。
- このアーキテクチャでは、ウォームインデックスの自動スナップショットもサポートされています。
- クロスクラスターレプリケーションはホットインデックスでのみサポートされています
- Shrink、Split、Clone などのインデックス APIs はウォームインデックスでは機能しません。

多層ドメインの作成

ステップ 1: ドメインを作成する

```
aws opensearch create-domain \
      --domain-name my-domain \
      --engine-version OpenSearch_3.3 \
```

```
      --cluster-config
 InstanceCount=3,InstanceType=or2.large.search,DedicatedMasterEnabled=true,DedicatedMasterType=
 \
      --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11 \
      --node-to-node-encryption-options Enabled=true \
      --encryption-at-rest-options Enabled=true \
      --domain-endpoint-options EnforceHTTPS=true,TLSSecurityPolicy=Policy-Min-
TLS-1-2-2019-07 \
      --advanced-security-options
 '{"Enabled":true,"InternalUserDatabaseEnabled":true,"MasterUserOptions":
{"MasterUserName":"user_name","MasterUserPassword":"your_pass"}}' \
      --access-policies '{"Version": "2012-10-17",        "Statement":
[{"Effect":"Allow","Principal":"*","Action":"es:*","Resource":"*"}]}' \
      --region us-east-1
```

ステップ 2: ウォームノードを検証する

```
aws opensearch describe-domain-nodes --domain-name my-domain --region us-east-1
```

サンプルレスポンス (抜粋):

```
{
      "NodeType": "Warm",
      "InstanceType": "oi2.large.search",
      "NodeStatus": "Active"
    }
```

階層移行の管理

多層ドメインは以下をサポートします。

- 簡素化されたエクスペリエンスのための新しい階層化 APIs
- 互換性のためのレガシー UltraWarm APIs

新しい階層化 APIs

インデックスをウォームに移行します。

```
curl -XPOST 'https://localhost:9200/index-name/_tier/warm'
```

レスポンス:

```
{"acknowledged": true}
```

インデックスをホットに移行します。

```
curl -XPOST 'https://localhost:9200/index-name/_tier/hot'
```

レスポンス:

```
{"acknowledged": true}
```

階層化ステータスを確認します。

```
curl -XGET 'https://localhost:9200/index-name/_tier'
```

レスポンスの例:

```
{
      "tiering_status": {
         "index": "index-name",
         "state": "RUNNING_SHARD_RELOCATION",
         "source": "HOT",
         "target": "WARM",
         "start_time": 1745836500563,
         "shard_level_status": {
           "running": 0,
           "total": 100,
           "pending": 100,
           "succeeded": 0
         }
      }
    }
```

詳細なシャードビュー:

```
curl 'https://localhost:9200/index1/_tier?detailed=true'
```

進行中の移行をすべて一覧表示する (テキスト):

```
curl 'https://localhost:9200/_tier/all'
```

進行中の移行 (JSON) をすべて一覧表示します。

```
curl 'https://localhost:9200/_tier/all?format=json'
```

ターゲット階層でフィルタリング:

```
curl 'https://localhost:9200/_tier/all?target=_warm'
```

互換性のためのレガシー UltraWarm APIs

ウォームに移行する:

```
curl -XPOST localhost:9200/_ultrawarm/migration/index2/_warm
```

ホットへの移行:

```
curl -XPOST localhost:9200/_ultrawarm/migration/index2/_hot
```

ステータスを確認します。

```
curl -XGET localhost:9200/_ultrawarm/migration/index2/_status
```

セキュリティ設定

既存の Amazon OpenSearch Service ドメインで多層ストレージを有効にした場合、`storage_tiering_manager`ロールはドメインで定義されていない可能性があります。きめ細かなアクセスコントロールを使用してドメインのウォームインデックスを管理するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。`storage_tiering_manager`ロールを手動で作成するには、以下のステップを実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	権限
storage_tiering_cluster	インデックス:admin/_tier/all

グループ名	権限
storage_tiering_index_read	インデックス:admin/_tier/get、インデックス:admin/get
storage_tiering_index_write	インデックス:admin/_tier/hot_to_warm、インデックス:admin/_tier/warm_to_hot

3. [ロール]、[ロールの作成] の順に選択します。
4. ロールに `storage_tiering_manager` という名前を付けます。
5. クラスターの許可 では、`storage_tiering_cluster` および `cluster_monitor` を選択します。
6. [インデックス] では、* と入力します。
7. インデックスのアクセス許可については、`storage_tiering_index_read`、、`storage_tiering_index_write`および を選択しますindices_monitor。
8. [作成] を選択します。
9. ロールを作成したら、多層インデックスを管理する任意のユーザーまたはバックエンドロールにマッピングします。

ベストプラクティス

Amazon OpenSearch Service ドメインに多層ストレージを実装するときは、次のベストプラクティスを考慮してください。

- データアクセスパターンを定期的に見直して階層の割り当てを最適化する
- パフォーマンスメトリクスをモニタリングして、効率的なリソース使用率を確保する
- 新しい階層化 APIs を使用してデータ移行をきめ細かく制御する

メトリクスのモニタリング

多層ストレージドメインは、ウォーム階層のパフォーマンスをモニタリングするための追加のメトリクスを提供します。これらのメトリクスには、既存の UltraWarm メトリクスと OpenSearch Optimized Instances アーキテクチャに固有の新しいメトリクスの両方が含まれます。

新しいメトリクス

メトリクス名	ノードレベルの統計	クラスターレベルの統計	詳細度
WarmIndexingLatency	平均	平均	1 分
WarmIndexingRate	平均	Average、Maximum、Sum	1 分
WarmThreadpoolIndexingQueue	最大値	合計、最大、平均	1 分
WarmThreadpoolIndexingRejected	最大値	合計	1 分
WarmThreadpoolIndexingThreads	最大値	合計、平均	1 分

Amazon OpenSearch Service の UltraWarm ストレージ

UltraWarm は、大量の読み取り専用データをコストパフォーマンスに優れた方法で Amazon OpenSearch Service に保存できます。標準データノードは「ホット」ストレージを使用します。このストレージは、各ノードにアタッチされたインスタンスストアまたは Amazon EBS ボリュームの形をとります。ホットストレージは、新しいデータのインデックス作成と検索において、可能な限り高速なパフォーマンスを提供します。

UltraWarm ノードは、接続されたストレージではなく、Amazon S3 と高度なキャッシュソリューションを使用してパフォーマンスを向上させます。アクティブに書き込みを行っていないインデックス、クエリの頻度が低いインデックス、および同じパフォーマンスが必要とされないインデックスについては、UltraWarm により、データの GiB あたりのコストが大幅に削減されます。ウォームインデックスはホットストレージに戻されない限り読み取り専用であるため、UltraWarm はログなどのイミュータブルなデータに最適です。

OpenSearch では、ウォームインデックスは他のインデックスと同じように動作します。同じ API を使用してそれらのクエリを行うことも、OpenSearch Dashboards で可視化を作成することもできます。

トピック

- 前提条件
- UltraWarm ストレージの要件とパフォーマンスに関する考慮事項
- UltraWarm 料金設定
- UltraWarm を有効にする
- UltraWarm ストレージへのインデックスの移行
- 移行を自動化する
- 移行の調整
- 移行をキャンセルする
- ホットインデックスとウォームインデックスを一覧表示する
- ウォームインデックスをホットストレージに戻す
- スナップショットからのウォームインデックスの復元
- ウォームインデックスの手動スナップショット
- ウォームインデックスをコールドストレージへ移行する
- KNN インデックスのベストプラクティス
- UltraWarm を無効にする

前提条件

UltraWarm には、重要な前提条件がいくつかあります。

- UltraWarm には、OpenSearch または Elasticsearch 6.8 以降が必要です。
- ウォームストレージを使用するには、ドメインに専用のマスターノードが必要です。
- スタンバイ付きマルチ AZ ドメインを使用する場合、ウォームノードの数は、使用するアベイラビリティーゾーンの数の倍数である必要があります。
- ドメインでデータノードに T2 または T3 インスタンスタイプが使用されている場合、ウォームストレージを使用することはできません。
- インデックスが近似 k-NN (`"index.knn":true`) を使用している場合、バージョン 2.17 以降であれば、ウォームストレージに移すことができます。2.17 より前のバージョンのドメインは、2.17 にアップグレードすればこの機能を利用できますが、2.x より前のバージョンで作成された KNN インデックスについては、UltraWarm ストレージに移行することはできません。

- ドメインできめ細かなアクセスコントロールが使用されている場合、UltraWarm API 呼び出しを行うには、ユーザーが ultrawarm_manager ロールにマッピングされている必要があります。

> **Note**
>
> ultrawarm_manager ロールは、既存の OpenSearch Service ドメインによっては定義されない場合があります。Dashboards にロールが表示されない場合は、それを手動で作成する必要があります。

許可の設定

既存の OpenSearch Service ドメインで UltraWarm を有効した場合、ultrawarm_manager ロールがドメインで定義されない場合があります。きめ細かなアクセスコントロールを使用してドメインのウォームインデックスを管理するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。ultrawarm_manager ロールを手動で作成するには、以下のステップを実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	許可
ultrawarm_cluster	• cluster:admin/ultrawarm/migration/list • cluster:monitor/nodes/stats
ultrawarm_index_read	• indices:admin/ultrawarm/migration/get • indices:admin/get
ultrawarm_index_write	• indices:admin/ultrawarm/migration/warm • indices:admin/ultrawarm/migration/hot • indices:monitor/stats • indices:admin/ultrawarm/migration/cancel

3. [ロール]、[ロールの作成] の順に選択します。
4. ロールに ultrawarm_manager という名前を付けます。

5. [クラスターの許可] では、`ultrawarm_cluster` および `cluster_monitor` を選択します。
6. [インデックス] では、`*` と入力します。
7. [インデックスの許可] では、`ultrawarm_index_read`、`ultrawarm_index_write`、および `indices_monitor` を選択します。
8. [作成] を選択します。
9. ロールを作成したら、任意のユーザーまたは UltraWarm インデックスを管理するバックエンドロールにそれをマッピングします。

UltraWarm ストレージの要件とパフォーマンスに関する考慮事項

the section called “ストレージ要件の計算” で説明されているように、ホットストレージ内のデータには、レプリカ、Linux の予約済み領域、OpenSearch Service 予約済み領域という大きなオーバーヘッドが発生します。例えば、1 つのレプリカシャードを持つ 20 GiB プライマリシャードには、約 58 GiB のホットストレージが必要です。

UltraWarm では Amazon S3 を使用するため、このオーバーヘッドは発生しません。UltraWarm ストレージ要件を計算するときは、プライマリシャードのサイズのみを考慮します。S3 のデータの耐久性はレプリカの必要性を排除し、S3 はオペレーティングシステムやサービスの考慮事項を抽象化します。同じ 20 GiB シャードには、20 GiB のウォームストレージが必要です。`ultrawarm1.large.search` インスタンスをプロビジョニングする場合、プライマリシャードには最大ストレージの 20 TiB すべてを使用できます。インスタンスタイプの概要と、それぞれが対応できるストレージの最大容量については、「the section called “UltraWarm ストレージのクォータ”」を参照してください。

UltraWarm では、最大シャードサイズを 50 GiB にすることをお勧めします。各 UltraWarm インスタンスタイプに割り当てられる CPU コアの数と RAM の量は、同時に検索できるシャードの数のアイデアを提供します。S3 の UltraWarm ストレージにはプライマリシャードのみがカウントされますが、OpenSearch Dashboards および `_cat/indices` は、すべてのプライマリシャードとレプリカシャードの合計として UltraWarm インデックスサイズを引き続きレポートすることに注意してください。

例えば、各 `ultrawarm1.medium.search` インスタンスには 2 つの CPU コアがあり、S3 で最大 1.5 TiB のストレージに対応できます。これらのインスタンスのうちの 2 つには、3 TiB のストレージが組み合わされており、各シャードが 50 GiB の場合、約 62 のシャードになります。クラスターへのリクエストがこれらのシャードのうちの 4 つしか検索しない場合、パフォーマンスが優れている可能性があります。リクエストが広範で、それらの 62 すべてを検索する場合、4 つの CPU コアがオペレーションを実施しにくくなる可能性があります。`WarmCPUUtilization` および

`WarmJVMMemoryPressure` UltraWarm メトリクスをモニタリングして、インスタンスがワークロードをどのように処理するかを理解してください。

検索が広範または頻繁である場合、インデックスをホットストレージに残すことを検討してください。他の OpenSearch ワークロードと同様に、UltraWarm がニーズを満たしているかどうかを判断する最も重要なステップは、現実的なデータセットを使用して代表的なクライアントテストを実施することです。

UltraWarm 料金設定

ホットストレージでは、プロビジョニングした分に対して料金を支払います。インスタンスにはアタッチされた Amazon EBS ボリュームが必要で、インスタンスストアが含まれるインスタンスもあります。そのストレージが空かいっぱいかに関係なく、同じ料金を支払います。

UltraWarm ストレージでは、使用した分に対して料金を支払います。`ultrawarm1.large.search` インスタンスは S3 で最大 20 TiB のストレージに対応できますが、1 TiB のデータを格納する場合、1 TiB のデータに対してのみ課金されます。他のすべてのノードタイプと同様に、UltraWarm ノードごとに時間料金も支払います。詳細については、「the section called “料金”」を参照してください。

UltraWarm を有効にする

コンソールは、ウォームストレージを使用するドメインを作成する最も簡単な方法です。ドメインの作成時に、[ウォームデータノードを有効にする] を選択し、必要なウォームノード数を設定します。前提条件を満たしていれば、既存のドメインでも同じ基本プロセスを使用できます。ドメインの状態が [処理中] から [アクティブ] になっても、UltraWarm が使用可能になるには数時間かかることがあります。

スタンバイ付きマルチ AZ ドメインを使用する場合、ウォームノードの数は、アベイラビリティーゾーンの数の倍数である必要があります。詳細については、「the section called “Multi-AZ with Standby”」を参照してください。

UltraWarm は、AWS CLI または設定 API を使用して有効にすることもできます。具体的には、`ClusterConfig` の `WarmEnabled`、`WarmCount`、`WarmType` オプションを使用します。

> **Note**
>
> ドメインはウォームノードの最大数をサポートします。詳細については、「the section called “クォータ”」を参照してください。

CLI コマンドの例

次の AWS CLI コマンドは、3 つのデータノード、3 つの専用マスターノード、6 つのウォームノード、およびきめ細かなアクセスコントロールが有効になっているドメインを作成します。

```
aws opensearch create-domain \
  --domain-name my-domain \
  --engine-version Opensearch_1.0 \
  --cluster-config
 InstanceCount=3,InstanceType=r6g.large.search,DedicatedMasterEnabled=true,DedicatedMasterType=
 \
  --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11 \
  --node-to-node-encryption-options Enabled=true \
  --encryption-at-rest-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true,TLSSecurityPolicy=Policy-Min-
TLS-1-2-2019-07 \
  --advanced-security-options
 Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions='{MasterUserName=master-
user,MasterUserPassword=master-password}' \
  --access-policies '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":["123456789012"]},"Action":
["es:*"],"Resource":"arn:aws:es:us-west-1:123456789012:domain/my-domain/*"}]}' \
  --region us-east-1
```

詳細については、「AWS CLI コマンドのリファレンス」を参照してください。

設定 API リクエストの例

設定 API に対する次のリクエストは、有効にされたきめ細かなアクセスコントロールおよび制限されたアクセスポリシーを持つ、3 つのデータノード、3 つの専用マスターノード、および 6 つのウォームノードを持つドメインを作成します。

```
POST https://es.us-east-2.amazonaws.com/2021-01-01/opensearch/domain
{
  "ClusterConfig": {
    "InstanceCount": 3,
    "InstanceType": "r6g.large.search",
    "DedicatedMasterEnabled": true,
    "DedicatedMasterType": "r6g.large.search",
    "DedicatedMasterCount": 3,
    "ZoneAwarenessEnabled": true,
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
```

```
    },
    "WarmEnabled": true,
    "WarmCount": 6,
    "WarmType": "ultrawarm1.medium.search"
  },
  "EBSOptions": {
    "EBSEnabled": true,
    "VolumeType": "gp2",
    "VolumeSize": 11
  },
  "EncryptionAtRestOptions": {
    "Enabled": true
  },
  "NodeToNodeEncryptionOptions": {
    "Enabled": true
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
  },
   "AdvancedSecurityOptions": {
    "Enabled": true,
    "InternalUserDatabaseEnabled": true,
    "MasterUserOptions": {
      "MasterUserName": "master-user",
      "MasterUserPassword": "master-password"
    }
  },
  "EngineVersion": "Opensearch_1.0",
  "DomainName": "my-domain",
  "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"123456789012\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:us-east-1:123456789012:domain/my-domain/*\"}]}"
}
```

詳細については、「Amazon OpenSearch Service API リファレンス」を参照してください。

UltraWarm ストレージへのインデックスの移行

インデックスへの書き込みが終了し、可能な限り高速な検索パフォーマンスを必要としなくなった場合は、インデックスをホットから UltraWarm に移行します。

```
POST _ultrawarm/migration/my-index/_warm
```

次に、移行のステータスを確認します。

```
GET _ultrawarm/migration/my-index/_status

{
  "migration_status": {
    "index": "my-index",
    "state": "RUNNING_SHARD_RELOCATION",
    "migration_type": "HOT_TO_WARM",
    "shard_level_status": {
      "running": 0,
      "total": 5,
      "pending": 3,
      "failed": 0,
      "succeeded": 2
    }
  }
}
```

移行を実施するには、インデックスヘルスが緑である必要があります。複数のインデックスを連続してすばやく移行する場合、_cat API と同様に、すべての移行の概要をプレーンテキストで取得できます。

```
GET _ultrawarm/migration/_status?v

index    migration_type state
my-index HOT_TO_WARM    RUNNING_SHARD_RELOCATION
```

OpenSearch Service は、一度に 1 つのインデックスを UltraWarm に移行します。キューには最大 200 の移行を設定できます。制限を超えるリクエストは拒否されます。キュー内の移行の現在の個数を確認するには、HotToWarmMigrationQueueSize メトリクスをモニタリングします。インデックスは、移行プロセス全体を通して使用でき、ダウンタイムは発生しません。

移行プロセスには次の状態があります。

```
PENDING_INCREMENTAL_SNAPSHOT
RUNNING_INCREMENTAL_SNAPSHOT
FAILED_INCREMENTAL_SNAPSHOT
PENDING_FORCE_MERGE
RUNNING_FORCE_MERGE
FAILED_FORCE_MERGE
```

```
PENDING_FULL_SNAPSHOT
RUNNING_FULL_SNAPSHOT
FAILED_FULL_SNAPSHOT
PENDING_SHARD_RELOCATION
RUNNING_SHARD_RELOCATION
FINISHED_SHARD_RELOCATION
```

これらの状態が示すように、スナップショット、シャード再配置、強制マージ中に移行が失敗する可能性があります。スナップショットまたはシャード再配置中の障害は、通常、ノードの障害または S3 接続の問題が原因です。通常、ディスク領域の不足は、強制マージ失敗の根本的な原因です。

移行が完了すると、同じ _status リクエストでエラーが返されます。その時点でインデックスをチェックすると、ウォームインデックスに固有の設定が表示されます。

```
GET my-index/_settings

{
  "my-index": {
    "settings": {
      "index": {
        "refresh_interval": "-1",
        "auto_expand_replicas": "false",
        "provided_name": "my-index",
        "creation_date": "1599241458998",
        "unassigned": {
          "node_left": {
            "delayed_timeout": "5m"
          }
        },
        "number_of_replicas": "1",
        "uuid": "GswyCdR0RSq0SJYmzsIpiw",
        "version": {
          "created": "7070099"
        },
        "routing": {
          "allocation": {
            "require": {
              "box_type": "warm"
            }
          }
        },
        "number_of_shards": "5",
        "merge": {
```

```
          "policy": {
            "max_merge_at_once_explicit": "50"
          }
        }
      }
    }
  }
}
```

- number_of_replicas は、ディスク領域を消費しないパッシブレプリカの数です。
- routing.allocation.require.box_type は、インデックスが標準データノードではなくウォームノードを使用するように指定します。
- merge.policy.max_merge_at_once_explicit は、移行中に同時にマージするセグメントの数を指定します。

ウォームストレージのインデックスは、ホットストレージに戻されない限り読み取り専用です。これにより、UltraWarm はログなどのイミュータブルなデータに最適となります。インデックスのクエリを行ってそれらを削除することはできますが、個々のドキュメントを追加、更新、削除することはできません。それらを行おうとすると、次のエラーが発生する場合があります。

```
{
  "error" : {
    "root_cause" : [
      {
        "type" : "cluster_block_exception",
        "reason" : "index [indexname] blocked by: [TOO_MANY_REQUESTS/12/disk usage
 exceeded flood-stage watermark, index has read-only-allow-delete block];"
      }
    ],
    "type" : "cluster_block_exception",
    "reason" : "index [indexname] blocked by: [TOO_MANY_REQUESTS/12/disk usage exceeded
 flood-stage watermark, index has read-only-allow-delete block];"
  },
  "status" : 429
}
```

移行を自動化する

インデックスが特定の経過時間に達した後、または他の条件を満たした後の移行プロセスは、the section called “インデックスステート管理” を使用して自動化することをお勧めします。このワークフローを示すサンプルポリシーを参照してください。

移行の調整

UltraWarm ストレージへのインデックスの移行には、強制マージが必要です。各 OpenSearch インデックスは、ある数のシャードで構成され、各シャードはある数の Lucene セグメントで構成されています。強制マージオペレーションは、削除対象としてマークされたドキュメントをパージし、ディスク領域を節約します。デフォルトでは、UltraWarm はインデックスを 1 つのセグメントにマージしますが、k-NN インデックスは例外となり、セグメント数のデフォルト値には 20 が使用されます。

`index.ultrawarm.migration.force_merge.max_num_segments` 設定を使用して、この値を最大 1,000 のセグメントに変更することができます。値を大きくすると、移行プロセスが高速になりますが、移行終了後のウォームインデックスのクエリレイテンシーが長くなります。設定を変更するには、次のリクエストを行います。

```
PUT my-index/_settings
{
  "index": {
    "ultrawarm": {
      "migration": {
        "force_merge": {
          "max_num_segments": 1
        }
      }
    }
  }
}
```

移行プロセスのこの段階でかかる時間を確認するには、`HotToWarmMigrationForceMergeLatency` メトリクスをモニタリングします。

移行をキャンセルする

UltraWarm は、キュー内の移行を順次処理します。移行がキューに入っていても、まだ開始していない場合は、次のリクエストを使用して移行をキューから削除できます。

```
POST _ultrawarm/migration/_cancel/my-index
```

ドメインできめ細かなアクセスコントロールを使用する場合は、このリクエストを行うための indices:admin/ultrawarm/migration/cancel 許可を持っている必要があります。

ホットインデックスとウォームインデックスを一覧表示する

UltraWarm では、ホットインデックスとウォームインデックスの管理に役立つ _all と同様の 2 つのオプションが追加されています。すべてのウォームインデックスまたはホットインデックスのリストについては、次のリクエストを実行します。

```
GET _warm
GET _hot
```

これらのオプションは、インデックスを指定する次のようなリクエストで使用できます。

```
_cat/indices/_warm
_cluster/state/_all/_hot
```

ウォームインデックスをホットストレージに戻す

インデックスに再度書き込む必要がある場合は、ホットストレージに移行し直します。

```
POST _ultrawarm/migration/my-index/_hot
```

ウォームストレージからホットストレージへの移行は、一度に最大 10 個までキューに入れることができます。OpenSearch Service は、移行リクエストをキューに入れられた順序で一度に 1 つずつ処理します。現在の個数を確認するには、WarmToHotMigrationQueueSize メトリクスをモニタリングします。

移行が完了したら、インデックス設定をチェックして、ニーズを満たしていることを確認します。インデックスはホットストレージに戻り、1 つのレプリカが作成されます。

スナップショットからのウォームインデックスの復元

UltraWarm では、自動スナップショット用の標準リポジトリに加えて、ウォームインデックス用の 2 つ目のリポジトリである cs-ultrawarm が追加されます。このリポジトリ内の各スナップショットには、1 つのインデックスしか含まれていません。ウォームインデックスを削除した場合、その

スナップショットは cs-ultrawarm リポジトリに 14 日間残ります。これは、他の自動スナップショットと同様です。

cs-ultrawarm からスナップショットを復元すると、ホットストレージではなくウォームストレージに復元されます。cs-automated-enc リポジトリと cs-automated リポジトリのスナップショットは、ホットストレージに復元されます。

UltraWarm スナップショットをウォームストレージに復元するには

1. 復元するインデックスを含む最新のスナップショットを特定します。

```
GET _snapshot/cs-ultrawarm/_all?verbose=false

{
  "snapshots": [{
    "snapshot": "snapshot-name",
    "version": "1.0",
    "indices": [
      "my-index"
    ]
  }]
}
```

> **Note**
>
> デフォルトでは、GET _snapshot/<repo> オペレーションは、リポジトリ内の各スナップショットの開始時間、終了時間、期間などの詳細なデータ情報を表示します。GET _snapshot/<repo> オペレーションは、リポジトリ内の各スナップショットのファイルから情報を取得します。開始時間、終了時間、期間は不要で、スナップショットの名前とインデックス情報のみが必要な場合、処理時間を最小限に抑えてタイムアウトを防ぐために、スナップショットを一覧表示する際に verbose=false パラメータを使用することをお勧めします。

2. インデックスがすでに存在する場合は、それを削除します。

```
DELETE my-index
```

インデックスを削除しない場合は、それをホットストレージに戻し、それの再インデックスを行います。

3. スナップショットの復元:

```
POST _snapshot/cs-ultrawarm/snapshot-name/_restore
```

UltraWarm は、この復元リクエストで指定したインデックス設定をすべて無視しますが、rename_pattern および rename_replacement のようなオプションを指定することができます。OpenSearch スナップショットの復元オプションの概要については、「OpenSearch ドキュメント」を参照してください。

ウォームインデックスの手動スナップショット

ウォームインデックスの手動スナップショットを取得することは可能ですが、推奨されません。自動化された cs-ultrawarm リポジトリには、移行中に取得された各ウォームインデックスのスナップショットがすでに含まれており、追加料金は発生しません。

デフォルトでは、OpenSearch Service は手動スナップショットにウォームインデックスを含めません。例えば、次の呼び出しには、ホットインデックスしか含まれていません。

```
PUT _snapshot/my-repository/my-snapshot
```

ウォームインデックスの手動スナップショットを取得することを選択した場合は、いくつかの重要な考慮事項が適用されます。

- ホットインデックスとウォームインデックスを混合することはできません。例えば、以下のコマンドは失敗します。

```
PUT _snapshot/my-repository/my-snapshot
{
  "indices": "warm-index-1,hot-index-1",
  "include_global_state": false
}
```

ホットインデックスとウォームインデックスの混合が含まれている場合は、ワイルドカード (*) ステートメントも失敗します。

- スナップショットあたり 1 つのウォームインデックスしか含めることができません。例えば、以下のコマンドは失敗します。

```
PUT _snapshot/my-repository/my-snapshot
```

```
{
  "indices": "warm-index-1,warm-index-2,other-warm-indices-*",
  "include_global_state": false
}
```

このリクエストは成功します。

```
PUT _snapshot/my-repository/my-snapshot
{
  "indices": "warm-index-1",
  "include_global_state": false
}
```

- 手動スナップショットは、もともとウォームインデックスが含まれていても、常にホットストレージに復元されます。

ウォームインデックスをコールドストレージへ移行する

クエリ頻度の低いデータが UltraWarm にある場合は、それをコールドストレージに移行することを検討してください。コールドストレージは、ときどきしかアクセスしないデータや、使用されなくなったデータを対象としています。コールドインデックスを読み書きすることはできませんが、クエリを行う必要があるときはいつでも無償でウォームストレージに移行できます。手順については、「コールドストレージへのインデックスの移行」を参照してください。

KNN インデックスのベストプラクティス

- Ultrawarm/Cold 階層は、すべての KNN インデックスエンジンタイプで使用できます。Lucene エンジンおよびディスク最適化ベクトル検索を使用する KNN インデックスに推奨されます。これらは、グラフデータをオフヒープメモリに完全にロードする必要がないためです。FAISS や NMSLIB などのネイティブインメモリエンジンで使用する場合、アクティブに検索されるシャードのグラフサイズを考慮し、それに応じて UltraWarm インスタンス (uw.large インスタンスタイプが望ましい) をプロビジョニングする必要があります。例えば、2 つの uw.large インスタンスが設定されている場合、それぞれのインスタンスには約 knn.memory.circuit_breaker.limit * 61 GiB のオフヒープメモリが利用可能になります。すべてのウォームクエリによって対象とされるシャードの累積グラフサイズが利用可能なオフヒープメモリを超えない場合、最適なパフォーマンスが得られます。利用可能なメモリがグラフのロードに必要な量よりも少ない場合、エビクションや、オフヒープメモリが利用可能になるまでの待機が発生するため、レイテンシーに影響します。そのため、インメモリエンジンが使用されてい

るユースケースや、エンジンに関係なく検索スループットが高いユースケースには、uw.medium インスタンスを使用することは推奨されません。

- KNN インデックスは、UltraWarm に移行する際に 1 つのセグメントに強制マージされることはありません。これにより、グラフサイズがインメモリエンジンにとって大きくなりすぎることが原因で、ホットノードやウォームノードで OOM 問題が発生する影響を回避できます。シャードあたりのセグメント数の増加により、ローカルキャッシュスペースの消費が増え、ウォーム階層に移行できるインデックスが少なくなる可能性があります。既存の設定を使用し、ウォーム階層への移行前にその設定値を上書きすることで、インデックスを 1 つのセグメントに強制マージすることもできます。詳細については、「the section called “移行の調整”」を参照してください。
- インデックスの検索頻度が低く、レイテンシーの影響を受けやすいワークロードを処理しないユースケースがある場合は、それらのインデックスを UltraWarm 階層に移行することができます。そうすることで、このような優先度の低いインデックスへのクエリは UltraWarm 階層のコンピューティングに処理させて、ホット階層のコンピューティングインスタンスをスケールダウンすることができます。また、優先度の低いインデックスと高いインデックスのクエリで消費されるリソースを分離できるため、互いに影響を与えないようにすることができます。

UltraWarm を無効にする

UltraWarm を無効にする最も簡単な方法は、コンソールを使用することです。ドメインを選択し、[アクション] から [クラスター設定の編集] を選択します。[UltraWarm データノードを有効にする] の選択を解除し、[変更を保存する] を選択します。 AWS CLI および設定 API で `WarmEnabled` オプションを使用することもできます。

UltraWarm を無効にする前に、すべてのウォームインデックスを削除する か、ホットストレージに移行する 必要があります。ウォームストレージが空になったら、5 分待ってから UltraWarm を無効にします。

Amazon OpenSearch Service 用コールドストレージ

コールドストレージでは、アクセス頻度の低いデータや履歴データを Amazon OpenSearch Service ドメインに保存し、他のストレージ階層よりも低コストで、オンデマンドで分析できます。古いデータに対して定期的な調査や法医学分析を行う必要がある場合は、コールドストレージが適しています。コールドストレージに適したデータの実際的な例としては、アクセス頻度の低いログ、コンプライアンス要件を満たすために保存する必要があるデータ、履歴価値のあるログなどがあります。

UltraWarm ストレージと同様に、コールドストレージは Amazon S3 によってバックアップされます。コールドデータをクエリする必要がある場合は、既存の UltraWarm ノードに選択的にアタッチ

できます。コールドデータの移行とライフサイクルは、手動で、またはインデックスステート管理ポリシーを使用して管理できます。

トピック

- 前提条件
- コールドストレージの要件とパフォーマンスに関する考慮事項
- コールドストレージの料金
- コールドストレージを有効にする
- OpenSearch Dashboards でのコールドインデックスの管理
- コールドストレージへのインデックスの移行
- コールドストレージへの移行の自動化
- コールドストレージへの移行のキャンセル
- コールドインデックスの一覧表示
- ウォームストレージへのインデックスの移行
- スナップショットからのコールドインデックスの復元
- コールドストレージからウォームストレージへの移行のキャンセル
- コールドインデックスメタデータの更新
- コールドインデックスの削除
- コールドストレージを無効にする

前提条件

コールドストレージには、次の前提条件があります。

- コールドストレージには、OpenSearch または Elasticsearch バージョン 7.9 以降が必要です
- OpenSearch Service ドメインでコールドストレージを有効にするには、同じドメインでウォームストレージも有効にしておく必要があります。
- コールドストレージを使用するには、ドメインに専用のマスターノードがある必要があります。
- ドメインでデータノードに T2 または T3 インスタンスタイプが使用されている場合、コールドストレージを使用することはできません。
- インデックスが近似 k-NN (`"index.knn":true`) を使用している場合、バージョン 2.17 以降ではインデックスをコールドストレージに移動できます。2.17 以前のバージョンのドメインの場合、

2.17 にアップグレードしてこの機能を使用することはできますが、2.x 以前のバージョンで作成された KNN インデックスをコールドストレージに移動することはできません。

- ドメインがきめ細かなアクセスコントロールを使用する場合、管理者以外のユーザーは、コールドインデックスを管理するために OpenSearch Dashboards の `cold_manager` ロールにマッピングされている必要があります。

> **Note**
>
> `cold_manager` ロールが既存の OpenSearch Service ドメインに存在しない可能性があります。Dashboards にロールが表示されない場合は、それを手動で作成する必要があります。

許可の設定

既存の OpenSearch Service ドメインでコールドストレージを有効にした場合、`cold_manager` ロールが定義されていない可能性があります。ドメインがきめ細かなアクセスコントロールを使用する場合、管理者以外のユーザーは、コールドインデックスを管理するためにこのロールにマッピングされている必要があります。`cold_manager` ロールを手動で作成するには、以下のステップを実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	許可
`cold_cluster`	• `cluster:monitor/nodes/stats` • `cluster:admin/ultrawarm*` • `cluster:admin/cold/*`
`cold_index`	• `indices:monitor/stats` • `indices:data/read/minmax` • `indices:admin/ultrawarm/migration/get` • `indices:admin/ultrawarm/migration/cancel`

3. [ロール]、[ロールの作成] の順に選択します。
4. ロールに cold_manager という名前を付けます。

5. [クラスターの許可] の場合、作成した cold_cluster グループを選択します。
6. [インデックス] の場合、* と入力します。
7. インデックスの許可] の場合、作成した cold_index グループを選択します。
8. [作成] を選択します。
9. ロールを作成したら、コールドインデックスを管理する任意のユーザーまたはバックエンドロールにそれをマッピングします。

コールドストレージの要件とパフォーマンスに関する考慮事項

コールドストレージは Simple Storage Service (Amazon S3) を使用するため、レプリカ、Linux 予約容量、OpenSearch Service 予約容量など、ホットストレージのオーバーヘッドは発生しません。コールドストレージには、コンピューティング性能がアタッチされていないため、特定のインスタンスタイプはありません。コールドストレージには、任意の量のデータを保存できます。Amazon CloudWatch の ColdStorageSpaceUtilization メトリクスをモニタリングして、使用しているコールドストレージ容量の量を確認します。

コールドストレージの料金

UltraWarm ストレージと同様に、コールドストレージでは、データストレージに対してのみ課金されます。コールドデータにはコンピューティングコストはないので、コールドストレージにデータがない場合、課金されることはありません。

コールドストレージとウォームストレージ間でデータを移動する場合、転送料金は発生しません。インデックスがウォームストレージとコールドストレージ間で移行されている間、インデックスの 1 つのコピーに対してのみ、継続して料金が発生します。移行が完了すると、インデックスは、移行先のストレージ階層に従って課金されます。コールドストレージの料金の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

コールドストレージを有効にする

コンソールは、コールドストレージを使用するドメインを作成する最も簡単な方法です。ドメインの作成中に、まず [ウォームデータノードを有効にする] を選択します。これは、同じドメインでウォームストレージを有効にしておく必要があるためです。次に、[コールドストレージを有効にする] を選択します。

前提条件を満たしている限り、既存のドメインでも同じプロセスを使用できます。ドメインの状態が [処理中] から [アクティブ] になっても、コールドストレージが使用可能になるには数時間かかることがあります。

AWS CLI または設定 API を使用して、コールドストレージを有効にすることもできます。

CLI コマンドの例

次の AWS CLI コマンドは、3 つのデータノード、3 つの専用マスターノード、コールドストレージが有効で、きめ細かなアクセスコントロールが有効になっているドメインを作成します。

```
aws opensearch create-domain \
  --domain-name my-domain \
  --engine-version Opensearch_1.0 \
  --cluster-
config ColdStorageOptions={Enabled=true},WarmEnabled=true,WarmCount=4,WarmType=ultrawarm1.mediu
 \
  --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11 \
  --node-to-node-encryption-options Enabled=true \
  --encryption-at-rest-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true,TLSSecurityPolicy=Policy-Min-
TLS-1-2-2019-07 \
  --advanced-security-options
 Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions='{MasterUserName=master-
user,MasterUserPassword=master-password}' \
  --region us-east-2
```

詳細については、「AWS CLI コマンドのリファレンス」を参照してください。

設定 API リクエストの例

設定 API に対する次のリクエストにより、3 つのデータノード、3 つの専用マスターノード、有効になっているコールドストレージ、および有効になっているきめ細かなアクセスコントロールがあるドメインが作成されます。

```
POST https://es.us-east-2.amazonaws.com/2021-01-01/opensearch/domain
{
  "ClusterConfig": {
    "InstanceCount": 3,
    "InstanceType": "r6g.large.search",
    "DedicatedMasterEnabled": true,
    "DedicatedMasterType": "r6g.large.search",
    "DedicatedMasterCount": 3,
    "ZoneAwarenessEnabled": true,
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
     },
```

```
    "WarmEnabled": true,
    "WarmCount": 4,
    "WarmType": "ultrawarm1.medium.search",
    "ColdStorageOptions": {
       "Enabled": true
     }
  },
  "EBSOptions": {
    "EBSEnabled": true,
    "VolumeType": "gp2",
    "VolumeSize": 11
  },
  "EncryptionAtRestOptions": {
    "Enabled": true
  },
  "NodeToNodeEncryptionOptions": {
    "Enabled": true
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
  },
   "AdvancedSecurityOptions": {
    "Enabled": true,
    "InternalUserDatabaseEnabled": true,
    "MasterUserOptions": {
      "MasterUserName": "master-user",
      "MasterUserPassword": "master-password"
    }
  },
  "EngineVersion": "Opensearch_1.0",
  "DomainName": "my-domain"
}
```

詳細については、「Amazon OpenSearch Service API リファレンス」を参照してください。

OpenSearch Dashboards でのコールドインデックスの管理

OpenSearch Service ドメインの既存の Dashboards インターフェイスを使用して、ホット、ウォーム、およびコールドインデックスを管理できます。Dashboards を使用すると、CLI や設定 API を使用することなく、ウォームストレージとコールドストレージ間でインデックスを移行し、インデックスの移行ステータスをモニタリングできます。詳細については、OpenSearch Dashboards でのインデックスの管理を参照してください。

コールドストレージへのインデックスの移行

インデックスをコールドストレージに移行する場合、データの時間範囲を指定して簡単に検出できるようにします。インデックス内のデータに基づいてタイムスタンプフィールドを選択するか、開始タイムスタンプと終了タイムスタンプを手動で指定するか、それを指定しないように選択することができます。

パラメータ	サポートされている値	説明
`timestamp_field`	インデックスマッピングの日付/時刻フィールド。	指定されたフィールドの最小値と最大値が計算され、コールドインデックスの `start_time` および `end_time` メタデータとして保存されます。
`start_time` ：および `end_time`	以下の形式のいずれか。 • strict_date_optional_time。例えば、`yyyy-MM-dd'T'HH:mm:ss.SSSZ` 、`yyyy-MM-dd` などです。 • エポック時間 (ミリ秒)	指定された値は、コールドインデックスの `start_time` および `end_time` メタデータとして保存されます。

タイムスタンプを指定しない場合は、代わりに `?ignore=timestamp` をリクエストに追加します。

次のリクエストは、ウォームインデックスをコールドストレージに移行し、そのインデックスのデータの開始時刻と終了時刻を提供します。

```
POST _ultrawarm/migration/my-index/_cold
  {
    "start_time": "2020-03-09",
    "end_time": "2020-03-09T23:00:00Z"
  }
```

次に、移行のステータスを確認します。

```
GET _ultrawarm/migration/my-index/_status

{
  "migration_status": {
    "index": "my-index",
    "state": "RUNNING_METADATA_RELOCATION",
    "migration_type": "WARM_TO_COLD"
  }
}
```

OpenSearch Service は、一度に 1 つのインデックスをコールドストレージに移行します。キューには最大 100 の移行を設定できます。制限を超えるリクエストは拒否されます。キュー内の移行の現在の個数を確認するには、WarmToColdMigrationQueueSize メトリクスをモニタリングします。移行プロセスには次の状態があります。

```
ACCEPTED_COLD_MIGRATION - Migration request is accepted and queued.
RUNNING_METADATA_MIGRATION - The migration request was selected for execution and
 metadata is migrating to cold storage.
FAILED_METADATA_MIGRATION - The attempt to add index metadata has failed and all
 retries are exhausted.
PENDING_INDEX_DETACH - Index metadata migration to cold storage is completed. Preparing
 to detach the warm index state from the local cluster.
RUNNING_INDEX_DETACH - Local warm index state from the cluster is being removed. Upon
 success, the migration request will be completed.
FAILED_INDEX_DETACH - The index detach process failed and all retries are exhausted.
```

コールドストレージへの移行の自動化

インデックスが特定の経過時間に達した後、または他の条件を満たした後の移行プロセスは、インデックスステート管理を使用して自動化することができます。ポリシーの例を参照してください。これは、インデックスをホットストレージから UltraWarm へ、またコールドストレージへと自動的に移行する方法を示しています。

> **Note**
>
> インデックスステート管理ポリシーを使用してインデックスをコールドストレージに移動するには、明示的な timestamp_field が必要です。

コールドストレージへの移行のキャンセル

コールドストレージへの移行がキューに追加されているか、失敗状態にある場合は、次のリクエストを使用して移行をキャンセルできます。

```
POST _ultrawarm/migration/_cancel/my-index

{
  "acknowledged" : true
}
```

ドメインがきめ細かなアクセスコントロールを使用している場合、このリクエストを行うには `indices:admin/ultrawarm/migration/cancel` 許可が必要です。

コールドインデックスの一覧表示

クエリを実行する前に、コールドストレージ内のインデックスを一覧表示して、詳細な分析のために UltraWarm に移行するインデックスを決定できます。次のリクエストは、すべてのコールドインデックスをインデックス名でソートして一覧表示します。

```
GET _cold/indices/_search
```

レスポンス例

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 3,
  "indices" : [
    {
      "index" : "my-index-1",
      "index_cold_uuid" : "hjEoh26mRRCFxRIMdgvLmg",
      "size" : 10339,
      "creation_date" : "2021-06-28T20:23:31.206Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-2",
      "index_cold_uuid" : "0vIS2n-oROmOWDFmwFIgdw",
      "size" : 6068,
      "creation_date" : "2021-07-15T19:41:18.046Z",
```

```
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-3",
      "index_cold_uuid" : "EaeXOBodTLiDYcivKsXVLQ",
      "size" : 32403,
      "creation_date" : "2021-07-08T00:12:01.523Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    }
  ]
}
```

フィルタリング

プレフィックスベースのインデックスパターンと時間範囲のオフセットに基づいて、コールドインデックスをフィルタリングできます。

次のリクエストは、event-* のプレフィックスパターンに一致するインデックスを一覧表示します。

```
GET _cold/indices/_search
 {
   "filters":{
      "index_pattern": "event-*"
   }
 }
```

レスポンス例

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 1,
  "indices" : [
    {
      "index" : "events-index",
      "index_cold_uuid" : "4eFiab7rRfSvp3slrIsIKA",
      "size" : 32263273,
      "creation_date" : "2021-08-18T18:25:31.845Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
```

```
    }
  ]
}
```

次のリクエストは、2019-03-01 および 2020-03-01 の間の start_time と end_time のメタデータフィールドを持つインデックスを返します。

```
GET _cold/indices/_search
{
  "filters": {
    "time_range": {
      "start_time": "2019-03-01",
      "end_time": "2020-03-01"
    }
  }
}
```

レスポンス例

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 1,
  "indices" : [
    {
      "index" : "my-index",
      "index_cold_uuid" : "4eFiab7rRfSvp3slrIsIKA",
      "size" : 32263273,
      "creation_date" : "2021-08-18T18:25:31.845Z",
      "start_time" : "2019-05-09T00:00Z",
      "end_time" : "2019-09-09T23:00Z"
    }
  ]
}
```

ソート

コールドインデックスは、インデックス名やサイズなどのメタデータフィールドによって並べ替えることができます。次のリクエストは、サイズを降順で並べ替えたすべてのインデックスを一覧表示します。

```
GET _cold/indices/_search
```

```
 {
 "sort_key": "size:desc"
 }
```

レスポンス例

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 5,
  "indices" : [
    {
      "index" : "my-index-6",
      "index_cold_uuid" : "4eFiab7rRfSvp3slrIsIKA",
      "size" : 32263273,
      "creation_date" : "2021-08-18T18:25:31.845Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-9",
      "index_cold_uuid" : "mbD3ZRVDRI6ONqgEOsJyUA",
      "size" : 57922,
      "creation_date" : "2021-07-07T23:41:35.640Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-5",
      "index_cold_uuid" : "EaeXOBodTLiDYcivKsXVLQ",
      "size" : 32403,
      "creation_date" : "2021-07-08T00:12:01.523Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    }
  ]
}
```

その他の有効なソートキーは、start_time:asc/desc、end_time:asc/desc、および index_name:asc/desc です。

Pagination (ページ分割)

コールドインデックスのリストをページ分割できます。page_size パラメータを使用して、ページごとに返されるインデックスの数を設定します (デフォルトは 10)。コールドインデックスのすべての _search リクエストは、後続の呼び出しに使用できる pagination_id を返します。

次のリクエストは、コールドインデックスの _search リクエストの結果をページ分割で表示し、次の 100 件の結果を表示します。

```
GET _cold/indices/_search?page_size=100
{
"pagination_id": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY"
}
```

ウォームストレージへのインデックスの移行

前のセクションでフィルタリング基準を用いてコールドインデックスのリストを絞り込んだ後、それらを UltraWarm に移行してデータのクエリを行い、それを使用して可視化を作成できます。

次のリクエストは、2 つのコールドインデックスをウォームストレージに移行します。

```
POST _cold/migration/_warm
 {
 "indices": "my-index1,my-index2"
 }


{
  "acknowledged" : true
}
```

移行のステータスを確認して移行 ID を取得するには、次のリクエストを送信します。

```
GET _cold/migration/_status
```

レスポンス例

```
{
  "cold_to_warm_migration_status" : [
    {
      "migration_id" : "tyLjXCA-S76zPQbPVHkOKA",
```

```
      "indices" : [
        "my-index1,my-index2"
      ],
      "state" : "RUNNING_INDEX_CREATION"
    }
  ]
}
```

インデックス固有の移行情報を取得するには、インデックス名を含めます。

```
GET _cold/migration/my-index/_status
```

インデックスを指定するのではなく、現在の移行ステータス別にインデックスを一覧表示できます。有効な値は、_failed、_accepted、_all です。

次のコマンドは、単一の移行リクエスト内のすべてのインデックスのステータスを取得します。

```
GET _cold/migration/_status?migration_id=my-migration-id
```

ステータスリクエストを使用して移行 ID を取得します。移行の詳細については、&verbose=true を追加します。

コールドストレージからウォームストレージには 10 またはそれ未満のバッチ数でインデックスを移行できます。最大で 100 インデックスを同時に移行できます。制限を超えるリクエストは拒否されます。現在、移行中の個数を確認するには、ColdToWarmMigrationQueueSize メトリクスをモニタリングします。移行プロセスには次の状態があります。

```
ACCEPTED_MIGRATION_REQUEST - Migration request is accepted and queued.
RUNNING_INDEX_CREATION - Migration request is picked up for processing and will create
 warm indexes in the cluster.
PENDING_COLD_METADATA_CLEANUP - Warm index is created and the migration service will
 attempt to clean up cold metadata.
RUNNING_COLD_METADATA_CLEANUP - Cleaning up cold metadata from the indexes migrated to
 warm storage.
FAILED_COLD_METADATA_CLEANUP - Failed to clean up metadata in the cold tier.
FAILED_INDEX_CREATION - Failed to create an index in the warm tier.
```

スナップショットからのコールドインデックスの復元

削除されたコールドインデックスを復元する必要がある場合は、「the section called “スナップショットからのウォームインデックスの復元”」の手順に従ってから、そのインデックスを再度コー

ルド階層に移行することで、ウォーム層に復元できます。削除されたコールドインデックスをコールド階層に直接復元することはできません。OpenSearch Service は、コールドインデックスが削除された後、それらを 14 日間保持します。

コールドストレージからウォームストレージへの移行のキャンセル

コールドストレージからウォームストレージへのインデックスの移行がキューに追加されているか、失敗状態にある場合は、次のリクエストを用いてそれをキャンセルできます。

```
POST _cold/migration/my-index/_cancel

{
  "acknowledged" : true
}
```

インデックスのバッチ (一度に最大 10 個) の移行をキャンセルするには、移行 ID を指定します。

```
POST _cold/migration/_cancel?migration_id=my-migration-id

{
  "acknowledged" : true
}
```

ステータスリクエストを使用して移行 ID を取得します。

コールドインデックスメタデータの更新

コールドインデックスの `start_time` および `end_time` フィールドを更新できます。

```
PATCH _cold/my-index
 {
 "start_time": "2020-01-01",
 "end_time": "2020-02-01"
 }
```

コールドストレージのインデックスの `timestamp_field` を更新することはできません。

> **Note**
>
> OpenSearch Dashboards は PATCH メソッドをサポートしていません。curl、Postman、または他のメソッドを使用して、コールドメタデータを更新します。

コールドインデックスの削除

ISM ポリシーを使用していない場合は、コールドインデックスを手動で削除できます。次のリクエストは、コールドインデックスを削除します。

```
DELETE _cold/my-index

{
  "acknowledged" : true
}
```

コールドストレージを無効にする

OpenSearch Service コンソールは、コールドストレージを無効にする最も簡単な方法です。ドメインを選択し、[アクション] から [クラスター設定の編集] を選択し、[コールドストレージを有効にする] の選択を解除します。

CLI または設定 API AWS を使用するには、 で `ColdStorageOptions`を設定します`"Enabled"="false"`。

コールドストレージを無効にする前に、すべてのコールドインデックスを削除するか、ウォームストレージに移行する必要があります。そうしないと、無効アクションは失敗します。

Amazon OpenSearch Service でのインデックスステート管理

Amazon OpenSearch Service の Index State Management (ISM) を使用すると、日常的なタスクを自動化するカスタム管理ポリシーを定義し、それらをインデックスおよびインデックスパターンに適用できます。これにより、インデックスオペレーションを実行するために、外部プロセスを設定および管理する必要がなくなります。

ポリシーには、デフォルトの状態と、移行するインデックスの状態一覧が含まれています。各状態内で、実行するアクションのリストと、移行をトリガーする条件を定義できます。一般的なユースケー

スは、一定期間が経過した古いインデックスを定期的に削除することです。例えば、30 日後にインデックスを `read_only` 状態に移行して、90 日後に最終的に削除するポリシーを定義できます。

ポリシーをインデックスにアタッチすると、ISM は 5～8 分ごと (1.3 より前のクラスターに対しては 30～48 分ごと) に実行されるジョブを作成し、ポリシーアクションを実行して条件を確認してから、インデックスを別の状態に移行します。このジョブを実行する基本時間は 5 分ごとです。さらに、0～60% のランダムジッターが追加されることで、すべてのインデックスからのアクティビティが同時に急増することがなくなります。クラスターの状態が赤の場合、ISM はジョブを実行しません。

ISM では、OpenSearch または Elasticsearch 6.8 が必要です。

> **Note**
>
> このドキュメントでは、ISM の概要といくつかのサンプルポリシーについて説明します。また、Amazon OpenSearch Service ドメインの ISM とセルフマネージド OpenSearch クラスターの ISM の違いについても説明します。包括的なパラメータリファレンス、各設定の説明、API リファレンスを含む ISM の完全なドキュメントについては、OpenSearch ドキュメントの「Index State Management」を参照してください。

> **Important**
>
> インデックステンプレートを使用して、新しく作成されたインデックスに ISM ポリシーを適用することはできなくなりました。ISM テンプレートフィールドを使用して、新しく作成されたインデックスを引き続き自動的に管理することができます。この更新では、この設定を使用する既存の CloudFormation テンプレートに影響する重大な変更が導入されました。

ISM ポリシーを作成する

インデックス状態管理を開始するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. ISM ポリシーを作成するドメインを選択します。
3. ドメインのダッシュボードから、OpenSearch Dashboards の URL に移動し、マスターユーザー名とパスワードでサインインします。URL はこの形式に従います。

```
domain-endpoint/_dashboards/
```

4. OpenSearch Dashboards 内の左側のナビゲーションパネルを開き、[Index Management] (インデックス管理)、それから [Create policy] (ポリシーの作成) を選択します。
5. ビジュアルエディタ または JSON エディタ を使用してポリシーを作成します。より構造化された方法でポリシーを定義できるビジュアルエディタを使用することをお勧めします。ポリシーの作成については、以下の サンプルポリシー を参照してください。
6. ポリシーを作成したら、それを 1 つ以上のインデックスにアタッチします。

```
POST _plugins/_ism/add/my-index
{
  "policy_id": "my-policy-id"
}
```

> **Note**
>
> ドメインで従来の Elasticsearch バージョンを実行している場合は、_plugins の代わりに _opendistro を使用します。

または、OpenSearch Dashboards でインデックスを選択し、[ポリシーの適用] を選択します。

サンプルポリシー

次のサンプルポリシーは、一般的な ISM ユースケースを自動化する方法を示しています。

ホットからウォーム、またコールドストレージまで

このサンプルポリシーでは、インデックスをホットストレージから UltraWarm に移動し、最終的にコールドストレージに移動します。次に、インデックスを削除します。

インデックスは、最初は hot 状態です。10 日後、ISM はインデックスを warm 状態に移動します。その 80 日後、90 日が経過したインデックスは cold 状態に移動します。1 年後、サービスは、インデックスを削除中であるという通知を Amazon Chime ルームに送信してから、完全にインデックスを削除します。

コールドインデックスでは、通常の delete オペレーションではなく、cold_delete オペレーションが必要であることに注意してください。また、ISM でコールドインデックスを管理するには、データに明示的な timestamp_field が必要であることに注意してください。

```
{
  "policy": {
    "description": "Demonstrate a hot-warm-cold-delete workflow.",
    "default_state": "hot",
    "schema_version": 1,
    "states": [{
        "name": "hot",
        "actions": [],
        "transitions": [{
          "state_name": "warm",
          "conditions": {
            "min_index_age": "10d"
          }
        }]
      },
      {
        "name": "warm",
        "actions": [{
          "warm_migration": {},
          "retry": {
            "count": 5,
            "delay": "1h"
          }
        }],
        "transitions": [{
          "state_name": "cold",
          "conditions": {
            "min_index_age": "90d"
          }
        }]
      },
      {
        "name": "cold",
        "actions": [{
            "cold_migration": {
              "timestamp_field": "<your timestamp field>"
            }
          }
        ],
```

```
        "transitions": [{
          "state_name": "delete",
          "conditions": {
             "min_index_age": "365d"
          }
        }]
      },
      {
        "name": "delete",
        "actions": [{
          "notification": {
            "destination": {
              "chime": {
                "url": "<URL>"
              }
            },
            "message_template": {
              "source": "The index {{ctx.index}} is being deleted."
            }
          }
        },
        {
          "cold_delete": {}
        }]
      }
    ]
  }
}
```

レプリカの数を減らす

このサンプルポリシーはもう少し単純で、ディスク容量を節約するために 7 日後にレプリカ数をゼロに減らし、21 日後にインデックスを削除します。このポリシーは、インデックスが重要ではなく、書き込みリクエストをこれ以上受信しないことを前提としています。レプリカ数をゼロにすると、データ損失のリスクがあります。

```
{
  "policy": {
    "description": "Changes replica count and deletes.",
    "schema_version": 1,
    "default_state": "current",
    "states": [{
        "name": "current",
```

```
        "actions": [],
        "transitions": [{
          "state_name": "old",
          "conditions": {
            "min_index_age": "7d"
          }
        }]
      },
      {
        "name": "old",
        "actions": [{
          "replica_count": {
            "number_of_replicas": 0
          }
        }],
        "transitions": [{
          "state_name": "delete",
          "conditions": {
            "min_index_age": "21d"
          }
        }]
      },
      {
        "name": "delete",
        "actions": [{
          "delete": {}
        }],
        "transitions": []
      }
    ]
  }
}
```

インデックスのスナップショットを撮る

このサンプルポリシーでは、`snapshot` オペレーションを使用して、インデックスに少なくとも 1 つのドキュメントが含まれているとすぐに、インデックスのスナップショットを撮ります。`repository` は、Amazon S3 に登録した手動スナップショットリポジトリの名前です。`snapshot` は、スナップショットの名前です。スナップショットの前提条件およびリポジトリを登録する手順については、「the section called “インデックススナップショットの作成”」を参照してください。

```
{
  "policy": {
    "description": "Takes an index snapshot.",
    "schema_version": 1,
    "default_state": "empty",
    "states": [{
        "name": "empty",
        "actions": [],
        "transitions": [{
          "state_name": "occupied",
          "conditions": {
            "min_doc_count": 1
          }
        }]
      },
      {
        "name": "occupied",
        "actions": [{
          "snapshot": {
            "repository": "<my-repository>",
            "snapshot": "<my-snapshot>"
            }
          }],
          "transitions": []
      }
    ]
  }
}
```

ISM テンプレート

ism_template フィールドをポリシーにセットアップすると、テンプレートパターンと一致するインデックスを作成したときに、ポリシーがそのインデックスに自動的にアタッチされるようになります。この例では、「log」で始まる名前で作成したインデックスは、自動的に ISM ポリシーmy-policy-id と一致します。

```
PUT _plugins/_ism/policies/my-policy-id
{
  "policy": {
    "description": "Example policy.",
    "default_state": "...",
    "states": [...],
```

```
    "ism_template": {
      "index_patterns": ["log*"],
      "priority": 100
    }
  }
}
```

より詳細な例については、「自動ロールオーバー用 ISM テンプレートを使用したサンプルポリシー」を参照してください。

相違点

OpenSearch および Elasticsearch と比較して、ISM for Amazon OpenSearch Service にはいくつかの違いがあります。

ISM オペレーション

- OpenSearch Service は、3 つのISM 固有のオペレーションである `warm_migration`、`cold_migration`、`cold_delete` をサポートします。
  - ドメインで UltraWarm が有効になっている場合、`warm_migration` アクションはインデックスをウォームストレージに移行します。
  - ドメインでコールドストレージが有効になっている場合、`cold_migration` アクションはインデックスをコールドストレージに移行し、`cold_delete` アクションはインデックスをコールドストレージから削除します。

  これらのアクションのいずれかが設定されたタイムアウト期間内に完了しない場合でも、ウォームインデックスへの移行は継続されます。上記のアクションのいずれかに error_notification を設定すると、タイムアウト期間内にアクションが完了しなかった場合に、そのアクションが失敗したことが通知されますが、この通知は参考用です。実際のオペレーションに固有のタイムアウトはなく、最終的に成功または失敗するまで実行が継続されます。
- ドメインが OpenSearch または Elasticsearch 7.4 以降を実行している場合、OpenSearch Service は ISM `open` および `close` オペレーションをサポートします。
- ドメインが OpenSearch または Elasticsearch 7.7 以降を実行している場合、OpenSearch サービスは ISM `snapshot` オペレーションをサポートします。

コールドストレージ ISM オペレーション

コールドインデックスでは、次の ISM API を使用するときに ?type=_cold パラメータを指定する必要があります。

- ポリシーを追加する
- ポリシーを削除する
- ポリシーを更新する
- 失敗したインデックスを再試行する
- インデックスを説明する

これらのコールドインデックス用の API には、次のような違いがあります。

- ワイルドカード演算子は、最後に使用する場合を除き、サポートされていません。例えば、_plugins/_ism/<add, remove, change_policy, retry, explain>/logstash-* はサポートされていますが、_plugins/_ism/<add, remove, change_policy, retry, explain>/iad-*-prod はサポートされていません。
- 複数のインデックスの名前とパターンはサポートされていません。例えば、_plugins/_ism/<add, remove, change_policy, retry, explain>/app-logs はサポートされていますが、_plugins/_ism/<add, remove, change_policy, retry, explain>/app-logs,sample-data はサポートされていません。

ISM の設定

OpenSearch と Elasticsearch を使用すると、_cluster/settings API を使用して利用可能なすべての ISM 設定を変更することができます。Amazon OpenSearch Service では、次の ISM 設定のみを変更できます。

- クラスターレベルの設定:
  - plugins.index_state_management.enabled
  - plugins.index_state_management.history.enabled
- インデックスレベルの設定:
  - plugins.index_state_management.rollover_alias

チュートリアル: Index State Management プロセスの自動化

このチュートリアルでは、日常的なインデックス管理タスクを自動化して、それらをインデックスおよびインデックスパターンに適用する ISM ポリシーの実装方法を示します。

Amazon OpenSearch Service の Index State Management (ISM) を使用すると、反復的なインデックス管理アクティビティを自動化できるため、追加のツールを使用してインデックスのライフサイクルを管理する必要がなくなります。インデックスが作成されてから経過した時間、サイズ、その他の条件に基づいてこれらのオペレーションを自動化するポリシーを、すべて Amazon OpenSearch Service ドメイン内から作成できます。

OpenSearch Service は、アクティブな書き込みと低レイテンシーの分析用のデフォルトの「ホット」状態、最大 3 ペタバイトの読み取り専用データ用の UltraWarm、無制限の長期アーカイブ用のコールドストレージの 3 つのストレージ階層をサポートします。

このチュートリアルでは、日次インデックスで時系列データを処理するサンプルユースケースをご紹介します。このチュートリアルでは、アタッチされた各インデックスの自動スナップショットを 24 時間後に作成するポリシーを設定します。その後、2 日後にインデックスをデフォルトのホット状態から UltraWarm ストレージに移行し、30 日後にコールドストレージに移行して、最後に 60 日後にインデックスを削除します。

前提条件

- OpenSearch Service のドメインでは、Elasticsearch バージョン 6.8 が実行されている必要があります。
- ドメインで UltraWarm とコールドストレージが有効になっている必要があります。
- ドメインの手動スナップショットリポジトリを登録する必要があります。
- ユーザーロールには、OpenSearch Service コンソールにアクセスするための十分な許可が必要です。必要に応じて、ドメインへのアクセスを検証して設定します。

ステップ 1: ISM ポリシーを設定する

まず、OpenSearch Dashboards で ISM ポリシーを設定します。

1. OpenSearch Service コンソールのドメインダッシュボードから、OpenSearch Dashboards の URL に移動して、マスターユーザー名とパスワードを使用してサインインします。URL はこの形式に従います: *domain-endpoint*/_dashboards/。

2. OpenSearch Dashboards で、[Add sample data] (サンプルデータを追加) を選択し、1 つ以上のサンプルインデックスをドメインに追加します。
3. 左側のナビゲーションパネルを開き、[Index Management] (インデックス管理)、[Create policy] (ポリシーの作成) の順に選択します。
4. ポリシーには `ism-policy-example` という名前を付けます。
5. デフォルトのポリシーを次のポリシーに置き換えます。

```
{
  "policy": {
    "description": "Move indexes between storage tiers",
    "default_state": "hot",
    "states": [
      {
        "name": "hot",
        "actions": [],
        "transitions": [
          {
            "state_name": "snapshot",
            "conditions": {
              "min_index_age": "24h"
            }
          }
        ]
      },
      {
        "name": "snapshot",
        "actions": [
          {
            "retry": {
              "count": 5,
              "backoff": "exponential",
              "delay": "30m"
            },
            "snapshot": {
              "repository": "snapshot-repo",
              "snapshot": "ism-snapshot"
            }
          }
        ],
        "transitions": [
          {
            "state_name": "warm",
```

```
            "conditions": {
              "min_index_age": "2d"
            }
          }
        ]
      },
      {
        "name": "warm",
        "actions": [
          {
            "retry": {
              "count": 5,
              "backoff": "exponential",
              "delay": "1h"
            },
            "warm_migration": {}
          }
        ],
        "transitions": [
          {
            "state_name": "cold",
            "conditions": {
              "min_index_age": "30d"
            }
          }
        ]
      },
      {
        "name": "cold",
        "actions": [
          {
            "retry": {
              "count": 5,
              "backoff": "exponential",
              "delay": "1h"
            },
            "cold_migration": {
              "start_time": null,
              "end_time": null,
              "timestamp_field": "@timestamp",
              "ignore": "none"
            }
          }
        ],
```

```
        "transitions": [
          {
            "state_name": "delete",
            "conditions": {
              "min_index_age": "60d"
            }
          }
        ]
      },
      {
        "name": "delete",
        "actions": [
          {
            "cold_delete": {}
          }
        ],
        "transitions": []
      }
    ],
    "ism_template": [
      {
        "index_patterns": [
          "index-*"
        ],
        "priority": 100
      }
    ]
  }
}
```

> **Note**
>
> ism_template フィールドは、新しく作成され、かつ、指定された index_patterns のいずれかに一致するインデックスにポリシーを自動的にアタッチします。この場合は、index- で始まるすべてのインデックスです。このフィールドは、環境内のインデックス形式に一致するように変更できます。詳細については、「ISM テンプレート」を参照してください。

6. ポリシーの snapshot セクションで、*snapshot-repo* を、ドメイン用に登録したスナップショットリポジトリの名前に置き換えます。オプションで、*ism-snapshot* を置き換えることもできます。これは、作成時のスナップショットの名前となります。
7. [作成] を選択します。ポリシーが [State management policies] (状態管理ポリシー) ページに表示されるようになりました。

ステップ 2: ポリシーを 1 つ以上のインデックスにアタッチする

ポリシーを作成したら、クラスター内の 1 つ以上のインデックスにアタッチします。

1. [Hot indicies] (ホットインデックス) タブにアクセスし、ステップ 1 で追加したすべてのサンプルインデックスをリスト化する opensearch_dashboards_sample を検索します。
2. すべてのインデックスを選択し、[Apply policy] (ポリシーを適用) を選択してから、作成したばかりの ism-policy-example ポリシーを選択します。
3. [Apply] (適用) を選択します。

[Policy managed indices] (ポリシー管理インデックス) ページで、インデックスがさまざまな状態を経るのをモニタリングできます。

インデックスロールアップにより Amazon OpenSearch Service のインデックスを要約する

Amazon OpenSearch Service のインデックスロールアップを使用すると、要約されたインデックスに古いデータを定期的にロールアップすることで、ストレージコストを削減できます。

関心のあるフィールドを選択し、インデックスロールアップを使用して、より粗いタイムバケットに集約されたフィールドのみを持つ新しいインデックスを作成します。同じクエリパフォーマンスで、数か月または数年の履歴データをわずかなコストで保存できます。

インデックスロールアップには、OpenSearch または Elasticsearch 7.9 以降が必要です。

> **Note**
>
> このドキュメントは、Amazon OpenSearch Service でのインデックスロールアップジョブの作成を開始するのに役立ちます。使用可能なすべての設定のリストや API リファレンスを含

む包括的なドキュメントについては、OpenSearch ドキュメントの「Index rollups」を参照してください。

インデックスロールアップジョブの作成

開始するには、OpenSearch Dashboards でインデックス管理を選択します。[ロールアップジョブ] を選択し、[ロールアップジョブの作成] を選択します。

ステップ 1: インデックスを設定する

ソースインデックスとターゲットインデックスを設定します。ソースインデックスは、ロールアップするインデックスです。ターゲットインデックスは、インデックスロールアップの結果が保存される場所です。

インデックスロールアップジョブを作成した後は、そのインデックス選択を変更することはできません。

ステップ 2: 集計とメトリクスを定義する

ロールアップする集計 (用語およびヒストグラム) とメトリクス (平均、合計、最大、最小、および値のカウント) を持つ属性を選択します。多くのスペースを節約することはないので、非常にきめ細かな属性を多く追加しないようにしてください。

ステップ 3: スケジュールを指定する

インデックスが取り込まれるときにロールアップするスケジュールを指定します。インデックスロールアップジョブは、デフォルトでは有効になっています。

ステップ 4: 確認して作成する

設定を確認し、[作成] を選択します。

ステップ 5: ターゲットインデックスを検索する

標準 `_search` API を使用して、ターゲットインデックスを検索できます。ターゲットインデックス内のデータの内部構造にアクセスすることはできません。これは、プラグインがターゲットインデックスに合わせてバックグラウンドでクエリを自動的に書き換えるからです。これは、ソースインデックスとターゲットインデックスで同じクエリを使用できるようにするためのものです。

ターゲットインデックスのクエリを行うには、`size` を 0 に設定します。

```
GET target_index/_search
{
  "size": 0,
  "query": {
    "match_all": {}
  },
  "aggs": {
    "avg_cpu": {
      "avg": {
        "field": "cpu_usage"
      }
    }
  }
}
```

> **Note**
>
> OpenSearch バージョン 2.2 以降では、1 回のリクエストで複数のロールアップインデックスを検索できます。2.2 より前の OpenSearch バージョンと従来の Elasticsearch OSS バージョンでは、検索ごとに 1 つのロールアップインデックスのみがサポートされます。

Amazon OpenSearch Service でのインデックスの変換

一方、インデックスロールアップジョブを使用すると、古いデータを凝縮されたインデックスにロールアップしてデータの詳細度を減らすことができますが、変換ジョブを使用すると、特定のフィールドを中心にしたデータの異なる要約ビューを作成できるため、さまざまな方法でデータを可視化または分析できます。

インデックス変換には、OpenSearch Dashboards のユーザーインターフェイスと REST API があります。この機能には OpenSearch 1.0 以降が必要です。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service ドメインでインデックス変換の使用を開始するのに役立つ、インデックス変換の概要を説明します。包括的なドキュメントと REST API リファレンスについては、オープンソースの OpenSearch ドキュメントの「Index transforms」を参照してください。

インデックス変換ジョブの作成

クラスターにデータがない場合は、OpenSearch Dashboards 内のサンプルフライトデータを使用して、変換ジョブを試してください。データを追加したら、OpenSearch Dashboards を起動します。次に [インデックス管理]、[変換ジョブ]、および [変換ジョブの作成] を選択します。

ステップ 1: インデックスを選択する

[インデックス] セクションで、ソースとターゲットインデックスを選択します。既存のターゲットインデックスを選択することも、そのインデックス名を入力して新しいターゲットインデックスを作成することもできます。

ソースインデックスのサブセットだけを変換する場合は、[データフィルターの追加] を選択し、OpenSearch クエリ DSL を使用して、ソースインデックスのサブセットを指定します。

ステップ 2: フィールドを選択する

インデックスを選択したら、変換ジョブで使用するフィールドと、グループ化または集約を使用するかどうかを選択します。

- グループ化を使用して、変換されたインデックス内の別のバケットにデータを配置できます。例えば、サンプルフライトデータ内のすべての空港の目的地をグループ化する場合は、`DestAirportID` フィールドを `DestAirportID_terms` フィールドのターゲットフィールドにグループ化すると、変換ジョブの終了後に、変換されたインデックスにグループ化された空港 ID を見つけることができます。
- 一方、集約では、簡単な計算を実行できます。例えば、変換ジョブに集約を含めると、すべての飛行機のチケットの合計を計算する `sum_of_total_ticket_price` の新しいフィールドを定義できます。その後、変換されたインデックス内の新しいデータを分析できます。

ステップ 3: スケジュールを指定する

変換ジョブはデフォルトで有効になっており、スケジュールに基づいて実行されます。変換実行間隔については、間隔を分、時間、日数で指定します。

ステップ 4: 確認してモニタリングする

設定を確認し、[作成] を選択します。次に、[ジョブステータスの変換] 列をモニタリングします。

ステップ5：ターゲットインデックスを検索する

ジョブが終了したら、標準 _search API を使用してターゲットインデックスを検索します。

例えば、DestAirportID フィールドに基づいてフライトデータを変換する変換ジョブを実行した後、SFO の値を持つすべてのフィールドを返すために、次のリクエストを実行することができます。

```
GET target_index/_search
{
  "query": {
    "match": {
      "DestAirportID_terms" : "SFO"
    }
  }
}
```

Amazon OpenSearch Service のクラスター間レプリケーション

Amazon OpenSearch Service のクラスター間レプリケーションを使用すると、1 つの OpenSearch Service ドメインから別のドメインにユーザーインデックス、マッピング、メタデータをレプリケートできます。クラスター間レプリケーションでは、障害発生時の災害対策が保証され、地理的に離れたデータセンター間でデータをレプリケートしてレイテンシーを削減できます。ドメイン間で転送されるデータには、標準のデータ転送 AWS 料金がかかります。

クラスター間レプリケーションは、ローカルまたはフォロワーインデクスがリモートまたはリーダーインデックスからデータを取り出すアクティブ/パッシブレプリケーションモデルに従います。リーダーインデックスは、データのソース (データの複製元) のインデックスを意味します。リーダーインデックスは、データのターゲット (データの複製先) のインデックスを意味します。

クラスター間レプリケーションは、Elasticsearch 7.10 または OpenSearch 1.1 以降を実行するドメインで使用できます。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service の観点からクラスター間のレプリケーションを設定する方法について説明します。これには、 を使用してクラスター間接続 AWS マネジメントコンソール を設定することが含まれます。これは、セルフマネージド OpenSearch クラスターでは不可能です。設定リファレンスと包括的な API リファレ

ンスを含む詳細なドキュメントについては、OpenSearch ドキュメントの「Cross-cluster replication」を参照してください。

トピック

- 制限事項
- 前提条件
- アクセス許可の要件
- クラスター間接続のセットアップ
- レプリケーションの開始
- レプリケーションの確認
- レプリケーションの一時停止と再開
- レプリケーションの開始
- 自動フォロー
- 接続されたドメインのアップグレード

制限事項

クラスター間レプリケーションには、次の制約事項があります。

- Amazon OpenSearch Service ドメインとセルフマネージド OpenSearch クラスターまたは Elasticsearch クラスターの間でデータをレプリケートすることはできません。
- 1 つのフォロワードメインから別のフォロワードメインにインデックスをレプリケートすることはできません。インデックスを複数のフォロワードメインにレプリケートする場合、レプリケートできるのは単一のリーダードメインからのみです。
- ドメインは、インバウンド接続とアウトバウンド接続の組み合わせを介して最大 20 のその他のドメインに接続できます。
- クラスター間接続を最初に設定する際には、リーダードメインがフォロワードメインと同じか、またはそれ以降のバージョンである必要があります。
- を使用してドメイン CloudFormation を接続することはできません。
- M3 インスタンスとバースト可能 (T2 および T3) インスタンスではクラスター間レプリケーションを使用できません。

- UltraWarm インデックスとコールドインデックス間でデータをレプリケートすることはできません。両方のインデックスがホットストレージにある必要があります。
- リーダードメインからインデックスを削除してもフォロワードメインの対応するインデックスが自動的に削除されることはありません。
- クラスター間のレプリケーションは、デフォルト とオプトイン の間ではサポートされていません。両方のドメインは、デフォルトのリージョンまたはオプトインリージョンにある必要があります。

前提条件

クラスター間レプリケーションをセットアップする前に、ドメインが次の要件を満たしていることを確認してください。

- Elasticsearch 7.10 または OpenSearch 1.1 以降
- Fine-grained access controlが有効
- ノード間暗号化が有効
- リーダーインデックスは を `index.soft_deletes.enabled`に設定する必要がありますtrue。Elasticsearch 7.0 または OpenSearch 1.0 以降で作成されたインデックスでは、この設定がデフォルトで有効になっています。ただし、Elasticsearch 6.x で作成されてアップグレードされたインデックスは を保持しますsoft_deletes=false。このようなインデックスをレプリケートするには、まずインデックスを再作成する必要があります。

  インデックスでソフト削除が有効になっているかどうかを確認するには:

  ```
  GET <index-name>/_settings?
  include_defaults=true&flat_settings=true&filter_path=*.settings.index.soft_deletes.enabled
  ```

  `soft_deletes` が の場合false、レプリケーションを開始する前にデータを新しいインデックスに再インデックスします。

アクセス許可の要件

レプリケーションを開始するには、リモート (リーダー) ドメインの `es:ESCrossClusterGet` アクセス許可を含める必要があります。リモートドメイン上で以下の IAM ポリシーが推奨されます。このポリシーでは、ドキュメントのインデックス作成や標準検索の実行など、他のオペレーションも実行できます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/leader-domain/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:ESCrossClusterGet",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/leader-domain"
    }
  ]
}
```

`es:ESCrossClusterGet` アクセス許可が `/leader-domain/*` ではなく `/leader-domain` に適用されていることを確認してください。

管理者以外のユーザーがレプリケーションアクティビティを実行するには、適切なアクセス許可にマッピングする必要もあります。ほとんどのアクセス許可は、特定の REST API オペレーションに対応します。例えば、`indices:admin/plugins/replication/index/_resume` アクセス許可を使用すると、インデックスのレプリケーションを再開できます。アクセス許可の完全な一覧については、OpenSearch ドキュメントの「Replication permissions」を参照してください。

> **Note**
>
> レプリケーションを開始してレプリケーションルールを作成するコマンドは、特殊なケースです。リーダードメインとフォロワードメインでバックグラウンドプロセスが呼び出されるため、リクエストで `leader_cluster_role` と `follower_cluster_role` を渡す必要があります。OpenSearch Service は、すべてのバックエンドレプリケーションタスクでこれらのロールを使用します。これらのロールのマッピングと使用の詳細については、OpenSearch ドキュメントの「Map the leader and follower cluster roles」を参照してください。

クラスター間接続のセットアップ

1 つのドメインから別のドメインにインデックスをレプリケートするには、ドメイン間でクラスター間接続を設定する必要があります。ドメイン間の接続する最も簡単な方法は、ドメインダッシュボードの [接続] タブを使用することです。設定 API または AWS CLI を使用することもできます。クラスター間レプリケーションは「プル」モデルに従うため、フォロワードメインから接続を開始します。

> **Note**
>
> 以前にクラスター間検索を実行するために 2 つのドメインを接続したことのある場合、その同じ接続をレプリケーションに使用することはできません。コンソールで、接続は `SEARCH_ONLY` とマークされます。以前に接続された 2 つのドメイン間でレプリケーションを実行するには、接続を削除し、再度作成する必要があります。これを実行すれば、クラスター間検索とクラスター間レプリケーションの両方で接続が使用できるようになります。

接続をセットアップするには

1. Amazon OpenSearch Service コンソールで、フォロワードメインを選択し、[接続] タブに移動して [リクエスト] を選択します。
2. [接続エイリアス] に接続の名前を入力します。
3. AWS アカウント と リージョンのドメインに接続するか、別のアカウントまたはリージョンに接続するかを選択します。
   - AWS アカウント および リージョンのドメインに接続するには、ドメインを選択し、リクエストを選択します。

- 別の AWS アカウント またはリージョンのドメインに接続するには、リモートドメインの ARN を指定し、リクエストを選択します。

OpenSearch Service が接続リクエストを検証します。ドメインに互換性がない場合、接続は失敗します。正常に完了した検証は承認のために宛先ドメインに送信されます。宛先ドメインでリクエストが承認されたら、レプリケーションを開始できます。

クラスター間レプリケーションは双方向レプリケーションをサポートします。したがって、ドメイン A からドメイン B へのアウトバウンド接続と、ドメイン B からドメイン A への別のアウトバウンド接続を作成できます。その後、ドメイン A がドメイン B のインデックスに従い、ドメイン B がドメイン A のインデックスに従うようにレプリケーションを設定できます。

レプリケーションの開始

クラスター間接続を確立したら、データのレプリケーションを開始できます。まず、レプリケートするインデックスをリーダードメインに作成します。

```
PUT leader-01
```

そのインデックスをレプリケートするには、次のコマンドをフォロワードメインに送信します。

```
PUT _plugins/_replication/follower-01/_start
{
   "leader_alias": "connection-alias",
   "leader_index": "leader-01",
   "use_roles":{
      "leader_cluster_role": "all_access",
      "follower_cluster_role": "all_access"
   }
}
```

接続エイリアスは、ドメインダッシュボードの [接続] タブで確認できます。

この例では、管理者がリクエストを発行していると仮定し、説明を簡単にするために `all_access` を `leader_cluster_role` と `follower_cluster_role` に使用します。ただし、本番環境では、リーダーインデックスとフォロワーインデックスの両方でレプリケーションユーザーを作成してマッピングすることをお勧めします。ユーザー名は同一である必要があります。これらのロールのマッピング方法の詳細については、OpenSearch ドキュメントの「Map the leader and follower cluster roles」を参照してください。

レプリケーションの確認

レプリケーションが行われていることを確認するには、レプリケーションステータスを取得します。

```
GET _plugins/_replication/follower-01/_status

{
  "status" : "SYNCING",
  "reason" : "User initiated",
  "leader_alias" : "connection-alias",
  "leader_index" : "leader-01",
  "follower_index" : "follower-01",
  "syncing_details" : {
    "leader_checkpoint" : -5,
    "follower_checkpoint" : -5,
    "seq_no" : 0
  }
}
```

リーダーとフォロワーのチェックポイントの値は負の整数で始まり、ユーザーが持っているシャードの数を反映します (1 つのシャードの場合は -1、5 つのシャードの場合は -5 など)。変更を加えるたびに、値は正の整数で増加します。2 つの値が同じ場合、インデックスが完全に同期されていることを意味します。これらのチェックポイント値を使用して、ドメイン間のレプリケーションのレイテンシーを測定できます。

レプリケーションをさらに検証するには、リーダーインデックスにドキュメントを追加します。

```
PUT leader-01/_doc/1
{
   "Doctor Sleep":"Stephen King"
}
```

そして、それがフォロワーインデックスに表示されることを確認します。

```
GET follower-01/_search

{
    ...
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "follower-01",
```

```
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "Doctor Sleep" : "Stephen King"
        }
      }
    ]
  }
}
```

レプリケーションの一時停止と再開

問題を修復する場合や、リーダードメインの負荷を削減する必要がある場合は、レプリケーションを一時的に一時停止できます。次のリクエストをフォロワードメインに送信します。空のリクエストボディを含めてください。

```
POST _plugins/_replication/follower-01/_pause
{}
```

次に、ステータスを取得して、レプリケーションが一時停止していることを確認します。

```
GET _plugins/_replication/follower-01/_status

{
  "status" : "PAUSED",
  "reason" : "User initiated",
  "leader_alias" : "connection-alias",
  "leader_index" : "leader-01",
  "follower_index" : "follower-01"
}
```

変更作業が完了したら、レプリケーションを再開します。次のリクエストをフォロワードメインに送信します。空のリクエストボディを含めてください。

```
POST _plugins/_replication/follower-01/_resume
{}
```

12 時間以上一時停止してからレプリケーションを再開することはできません。レプリケーションを停止し、フォロワーインデックスを削除し、リーダーのレプリケーションを再開する必要があります。

レプリケーションの開始

レプリケーションを完全に停止すると、フォロワーインデックスはリーダーのフォローを解除し、標準インデックスになります。停止したレプリケーションを再開することはできません。

フォロアードメインからレプリケーションを停止します。空のリクエストボディを含めてください。

```
POST _plugins/_replication/follower-01/_stop
{}
```

自動フォロー

指定したパターンに一致するインデックスを自動的にレプリケートする 1 つのリーダードメインに対して、レプリケーションルールのセットを定義できます。リーダードメイン上のインデックスがパターンの 1 つ (books* など) に一致すると、一致するフォロワーインデックスがフォロワードメインに作成されます。OpenSearch Service は、パターンに一致する既存のインデックスに加えて、ユーザーが作成した新しいインデックスを複製します。フォロワードメインにすでに存在するインデックスはレプリケートされません。

すべてのインデックス (システムによって作成されたインデックスおよびフォロワードメインに既に存在するインデックスを除く) をレプリケートするには、ワイルドカード (*) パターンを使用します。

レプリケーションルールの作成

フォロアードメインにレプリケーションルールを作成し、クラスター間接続の名前を指定します。

```
POST _plugins/_replication/_autofollow
{
   "leader_alias" : "connection-alias",
   "name": "rule-name",
   "pattern": "books*",
   "use_roles":{
      "leader_cluster_role": "all_access",
      "follower_cluster_role": "all_access"
   }
}
```

接続エイリアスは、ドメインダッシュボードの [接続] タブで確認できます。

この例では、管理者がリクエストを発行していると仮定し、説明を簡単にするために all_access をリーダードメインルールとフォロワードメインルールとして使用します。ただし、本番環境では、リーダーインデックスとフォロワーインデックスの両方でレプリケーションユーザーを作成してマッピングすることをお勧めします。ユーザー名は同一である必要があります。これらのロールのマッピング方法の詳細については、OpenSearch ドキュメントの「Map the leader and follower cluster roles」を参照してください。

ドメイン上の既存のレプリケーションルールのリストを取得するには、auto-follow stats API オペレーションを使用します。

ルールをテストするには、リーダードメインのパターンに一致するインデックスを作成します。

```
PUT books-are-fun
```

そして、そのレプリカがフォロアードメインに表示されていることを確認します。

```
GET _cat/indices

health status index          uuid                   pri rep docs.count docs.deleted
 store.size pri.store.size
green  open   books-are-fun  ldfHO78xYYdxRMULuiTvSQ     1   1          0            0
     208b           208b
```

レプリケーションルールの削除

レプリケーションルールを削除すると、OpenSearch Service は、パターンに一致する新規インデックスのレプリケーションを停止しますが、ユーザーがレプリケーションを停止するまで、既存のレプリケーションアクティビティは継続します。

フォロアードメインからレプリケーションルールを削除します。

```
DELETE _plugins/_replication/_autofollow
{
   "leader_alias" : "connection-alias",
   "name": "rule-name"
}
```

接続されたドメインのアップグレード

クラスター間接続が可能な 2 つのドメインのエンジンバージョンをアップグレードするには、まずフォロワードメインをアップグレードしてから、リーダードメインをアップグレードします。それら

の間の接続を削除しないでください。削除すると、レプリケーションが一時停止し、再開できなくなります。

リモート再インデックスを使用した Amazon OpenSearch Service インデックスの移行

リモート再インデックスを使用すると、1 つの Amazon OpenSearch Service ドメインから別のドメインにインデックスをコピーできます。OpenSearch Service ドメインまたはセルフマネージド OpenSearch クラスターおよび Elasticsearch クラスターからインデックスを移行できます。

リモートドメインとインデックスは、データのソース (データのコピー元) のドメインとインデックスを意味します。ローカルドメインとインデックスは、データのターゲット (データのコピー先) のドメインとインデックスを意味します。

リモートでの再インデックス作成には、ローカルドメインで OpenSearch 1.0 以降、または Elasticsearch 6.7 以降が必要です。リモートドメインは、ローカルドメインと同じまたはそれより前のメジャーバージョンである必要があります。Elasticsearch のバージョンは、OpenSearch バージョンより前と見なされます。つまり、Elasticsearch ドメインから OpenSearch ドメインにデータを再インデックスできます。同じメジャーバージョン内では、リモートドメインは任意のマイナーバージョンにすることができます。例えば、Elasticsearch 7.10.x から 7.9 へのリモート再インデックスはサポートされますが、OpenSearch 1.0 から Elasticsearch 7.10.x へのリモート再インデックスはサポートされません。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service ドメイン間でデータを再インデックスする方法について説明します。詳細な手順やサポートされているオプションを含む、`reindex` オペレーションの完全なドキュメントは、OpenSearch ドキュメントの「Reindex document」を参照してください。

トピック

- 前提条件
- OpenSearch Service インターネットドメイン間でデータを再インデックスする
- リモートが VPC にあるときの OpenSearch Service ドメイン間でのデータの再インデックス
- OpenSearch Service 以外のドメイン間でデータの再インデックスを行う

- 大規模なデータセットの再インデックスを行う
- リモート再インデックス設定

前提条件

リモート再インデックスには、次の要件があります。

- リモートドメインは、ローカルドメインからアクセス可能である必要があります。VPC 内に存在するリモートドメインでは、ローカルドメインに VPC へのアクセスが必要です。このプロセスはネットワーク構成によって異なりますが、VPN またはマネージドネットワークへの接続、もしくは、ネイティブの VPC エンドポイント接続の使用が必要となる場合がほとんどです。詳細についてはthe section called “VPC サポート”を参照してください。
- リクエストは、他の REST リクエストと同様にリモートドメインによって承認される必要があります。リモートドメインできめ細かなアクセスコントロールが有効になっている場合は、リモートドメインで再インデックスを実行し、リモートドメインでインデックスを読み取るための許可が必要です。セキュリティに関するその他の考慮事項については、「the section called “きめ細かなアクセスコントロール”」を参照してください。
- 再インデックスプロセスを開始する前に、ローカルドメインに目的の設定でインデックスを作成することをお勧めします。
- ドメインでデータノードに T2 または T3 インスタンスタイプが使用されている場合、リモート再インデックスを使用することはできません。

OpenSearch Service インターネットドメイン間でデータを再インデックスする

最も基本的なシナリオは、リモートインデックスがパブリックにアクセス可能なエンドポイントを持つローカルドメイン AWS リージョン と同じ にあり、IAM 認証情報に署名していることです。

リモートドメインから、再インデックス元のリモートインデックスと、再インデックス先のローカルインデックスを指定します。

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://remote-domain-endpoint:443"
    },
```

```
    "index": "remote_index"
  },
  "dest": {
    "index": "local_index"
  }
}
```

検証チェックを行うには、リモートドメインのエンドポイントの末尾に 443 を追加する必要があります。

インデックスがローカルドメインにコピーされたことを確認するには、次のリクエストをローカルドメインに送信します。

```
GET local_index/_search
```

リモートインデックスがローカルドメインとは異なるリージョンにある場合は、次のサンプルリクエストのように、リージョン名を渡します。

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://remote-domain-endpoint:443",
      "region": "eu-west-1"
    },
    "index": "remote_index"
  },
  "dest": {
    "index": "local_index"
  }
}
```

AWS GovCloud (US) や中国リージョンのように分離されたリージョンの場合、IAM ユーザーがそれらのリージョンで認識されないため、エンドポイントにアクセスできない可能性があります。

リモートドメインが基本的な認証で保護されている場合は、ユーザーネームとパスワードを指定します。

```
POST _reindex
{
  "source": {
    "remote": {
```

```
      "host": "https://remote-domain-endpoint:443",
      "username": "username",
      "password": "password"
    },
    "index": "remote_index"
  },
  "dest": {
    "index": "local_index"
  }
}
```

リモートが VPC にあるときの OpenSearch Service ドメイン間でのデータの再インデックス

すべての OpenSearch Service ドメインは、独自の内部仮想プライベートクラウド (VPC) インフラストラクチャで構成されています。既存の OpenSearch Service VPC に新しいドメインを作成すると、VPC 内の各データノードに Elastic Network Interface が作成されます。

ターゲットの再インデックスオペレーションはターゲットの OpenSearch Service ドメインから実行される (独自のプライベート VPC 内で実行される) ため、ローカルドメインの VPC にアクセスする方法が必要になります。これを行うには、組み込みの VPC エンドポイント接続機能を使用して 経由で接続を確立するか AWS PrivateLink、プロキシを設定します。

ローカルドメインで OpenSearch バージョン 1.0 以降を使用している場合は、 コンソールまたは AWS CLI を使用して AWS PrivateLink 接続を作成できます。 AWS PrivateLink 接続により、ローカル VPC 内のリソースは、同じ 内のリモート VPC 内のリソースにプライベートに接続できます AWS リージョン。

VPC エンドポイント接続を作成するには、再インデックスを作成するソースドメインがローカル VPC にあり、ソースドメインと宛先ドメインの両方が同じ AWS リージョンにある必要があります。

を使用してデータを再インデックスする AWS マネジメントコンソール

コンソールでリモートの再インデックスを使用すれば、VPC エンドポイント接続を共有している 2 つのドメイン間でインデックスをコピーできます。

1. Amazon OpenSearch Service コンソール (https://console.aws.amazon.com/aos/) に移動します。
2. 左側のナビゲーションペインで [Domains] (ドメイン) を選択します。

3. ローカルドメイン (データをコピーするドメイン) を選択します。選択すると、ドメインの詳細ページが開きます。一般情報の下にある [接続] タブを選択し、[リクエスト] を選択します。
4. [接続のリクエスト] ページで、接続モードに [VPC エンドポイント接続] を選択し、その他の関連情報を入力します。これらの詳細には、データをコピーするドメインであるリモートドメインが含まれます。次に、[Request] (リクエスト) を選択します。
5. リモートドメインの詳細ページに移動し、[接続] タブを選択してから [インバウンド接続] テーブルを見つけます。先ほど接続を作成したドメイン (ローカルドメイン) の名前の横にあるチェックボックスをオンにします。[承認] を選択します。
6. ローカルドメインに戻り、[Connections] (接続) タブを選択してから [Outbound connections] (アウトバウンド接続) テーブルを見つけます。2 つのドメイン間の接続がアクティブになると、テーブルの [Endpoint] (エンドポイント) 列でエンドポイントが利用可能になります。エンドポイントをコピーします。
7. ローカルドメインのダッシュボードを開き、左側のナビゲーションにある [Dev Tools] (開発ツール) を選択します。リモートのドメインインデックスがローカルのドメインにまだ存在していないことを確認するには、次の GET リクエストを実行します。*remote-domain-index-name* を実際のインデックス名で置き換えます。

```
GET remote-domain-index-name/_search
{
   "query":{
      "match_all":{}
   }
}
```

   出力に、インデックスが見つからなかったことを示すエラーが表示されます。
8. GET リクエストの下で、以下のような POST リクエストを作成し、リモートホストとしてエンドポイントを使用します。

```
POST _reindex
{
   "source":{
      "remote":{
         "host":"connection-endpoint",
         "username":"username",
         "password":"password"
      },
      "index":"remote-domain-index-name"
   },
```

```
   "dest":{
      "index":"local-domain-index-name"
   }
}
```

このリクエストを実行します。

9. GET リクエストをもう 1 度実行します。そうすると、出力にローカルインデックスが存在すると表示されます。このインデックスをクエリして、OpenSearch がリモートインデックスからすべてのデータをコピーしたことを確認できます。

OpenSearch Service API オペレーションでデータを再インデックスする

API でリモートの再インデックスを使用すれば、VPC エンドポイント接続を共有している 2 つのドメイン間でインデックスをコピーできます。

1. CreateOutboundConnection API オペレーションを使用して、ローカルドメインからリモートドメインへの新しい接続をリクエストします。

```
POST https://es.region.amazonaws.com/2021-01-01/opensearch/cc/outboundConnection

{
   "ConnectionAlias": "remote-reindex-example",
   "ConnectionMode": "VPC_ENDPOINT",
   "LocalDomainInfo": {
      "AWSDomainInformation": {
         "DomainName": "local-domain-name",
         "OwnerId": "aws-account-id",
         "Region": "region"
      }
   },
   "RemoteDomainInfo": {
      "AWSDomainInformation": {
         "DomainName": "remote-domain-name",
         "OwnerId": "aws-account-id",
         "Region": "region"
      }
   }
}
```

ConnectionId が応答で返されます。この ID は次のステップで使用するので保存しておきます。

2. AcceptInboundConnection API オペレーションを接続 ID とともに使用して、ローカルドメインからのリクエストを承認します。

```
PUT https://es.region.amazonaws.com/2021-01-01/opensearch/cc/
inboundConnection/ConnectionId/accept
```

3. DescribeOutboundConnections API オペレーションを使用して、リモートドメインのエンドポイントを取得します。

```
{
    "Connections": [
        {
            "ConnectionAlias": "remote-reindex-example",
            "ConnectionId": "connection-id",
            "ConnectionMode": "VPC_ENDPOINT",
            "ConnectionProperties": {
                "Endpoint": "connection-endpoint"
            },
            ...
        }
    ]
}
```

connection-endpoint はステップ 5 で使用するので保存しておきます。

4. リモートのドメインインデックスがローカルのドメインにまだ存在していないことを確認するには、次の GET リクエストを実行します。*remote-domain-index-name* を実際のインデックス名で置き換えます。

```
GET local-domain-endpoint/remote-domain-index-name/_search
{
   "query":{
      "match_all":{}
   }
}
```

出力に、インデックスが見つからなかったことを示すエラーが表示されます。

5. POST リクエストを作成し、次のようにエンドポイントをリモートホストとして使用します。

```
POST local-domain-endpoint/_reindex
{
   "source":{
      "remote":{
         "host":"connection-endpoint",
         "username":"username",
         "password":"password"
      },
      "index":"remote-domain-index-name"
   },
   "dest":{
      "index":"local-domain-index-name"
   }
}
```

このリクエストを実行します。

6. GET リクエストをもう 1 度実行します。そうすると、出力にローカルインデックスが存在すると表示されます。このインデックスをクエリして、OpenSearch がリモートインデックスからすべてのデータをコピーしたことを確認できます。

リモートドメインが VPC 内でホストされ、VPC エンドポイントの接続機能の使用を希望しない場合は、パブリックにアクセス可能なエンドポイントを使ってプロキシを設定します。この場合、OpenSearch Service には、VPC にトラフィックを送信する機能がないため、パブリックエンドポイントが必要になります。

VPC モードでドメインを実行する場合、1 つまたは複数のエンドポイントが VPC に配置されます。ただし、これらのエンドポイントは VPC 内のドメインに入るトラフィックのみを対象としており、VPC 自体へのトラフィックは許可されません。

リモートの再インデックスコマンドは、リモートドメインから実行されるため、発信元のトラフィックはそれらのエンドポイントを使ってリモートドメインにアクセスすることはできません。このユースケースでプロキシが必要となるのはそのためです。プロキシドメインには、公開認証機関 (CA) によって署名された証明書が必要です。自己署名証明書またはプライベート CA 署名証明書はサポートされていません。

OpenSearch Service 以外のドメイン間でデータの再インデックスを行う

リモートインデックスが OpenSearch Service の外部でホストされている場合 (セルフマネージド EC2 インスタンスなど)、`external` パラメータを `true` に設定します。

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://remote-domain-endpoint:443",
      "username": "username",
      "password": "password",
      "external": true
    },
    "index": "remote_index"
  },
  "dest": {
    "index": "local_index"
  }
}
```

この場合、ユーザーネームとパスワードを用いた基本的な認証のみがサポートされます。リモートドメインには、(ローカルの OpenSearch Service ドメインと同じ VPC にある場合でも) パブリックにアクセス可能なエンドポイントと、パブリック CA によって署名された証明書が必要です。自己署名証明書またはプライベート CA 署名証明書はサポートされていません。

大規模なデータセットの再インデックスを行う

リモート再インデックスは、次のデフォルト値を使用してリモートドメインにスクロールリクエストを送信します。

- 5 分の検索コンテキスト
- 30 秒のソケットタイムアウト
- 1,000 のバッチサイズ

データに合わせてこれらのパラメータを調整することをお勧めします。大きなドキュメントの場合は、バッチサイズを小さくするか、タイムアウトを長くするかを検討してください。一般的なガイドラインとして、100 MB を超えるドキュメントは「ラージ」と見なす必要があり、バッチサイズは 100 以下が必要になる場合があります。10 MB ~ 100 MB のドキュメントでは、バッチサイズを 500 にすることをお勧めします。50 MB を超えるドキュメントのソケットタイムアウトを 60 秒以上に調整します。詳細については、「Paginate results」を参照してください。

```
POST _reindex?pretty=true&scroll=10h&wait_for_completion=false
{
```

```
  "source": {
    "remote": {
      "host": "https://remote-domain-endpoint:443",
      "socket_timeout": "60m"
    },
    "size": 100,
    "index": "remote_index"
  },
  "dest": {
    "index": "local_index"
  }
}
```

パフォーマンスを向上させるために、ローカルインデックスに次の設定を追加することもお勧めします。

```
PUT local_index
{
  "settings": {
    "refresh_interval": -1,
    "number_of_replicas": 0
  }
}
```

再インデックスプロセスが完了したら、目的のレプリカの数を設定し、更新間隔の設定を削除できます。

クエリを介して選択したドキュメントのサブセットのみの再インデックスを行うには、次のリクエストをローカルドメインに送信します。

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://remote-domain-endpoint:443"
    },
    "index": "remote_index",
    "query": {
      "match": {
        "field_name": "text"
      }
    }
```

```
  },
  "dest": {
    "index": "local_index"
  }
}
```

リモート再インデックスはスライスをサポートしていないため、同じリクエストに対して複数のスクロールオペレーションを並行して実行することはできません。

リモート再インデックス設定

OpenSearch Service では、標準の再インデックスオプションに加えて、次のオプションをサポートしています。

オプション	有効な値	説明	必須
外部	ブール値	リモートドメインが OpenSearch Service ドメインでない場合か、または 2 つの VPC ドメイン間でインデックスを再作成する場合は、`true` のように指定します。	いいえ
リージョン	String	リモートドメインが別のリージョンにある場合は、リージョン名を指定します。	いいえ

Data Streams を用いた Amazon OpenSearch Service での時系列データの管理

時系列データを管理する一般的なワークフローには、ロールオーバーインデックスエイリアスの作成、書き込みインデックスの定義、バッキングインデックスの共通のマッピングと設定の定義など、複数の手順が含まれます。

Amazon OpenSearch Service の Data Streams は、この初期設定プロセスを簡素化するのに役立ちます。Data Streams は、通常は追加のみであるアプリケーションログなどの時間ベースのデータに対してすぐに機能します。

Data Streams には OpenSearch バージョン 1.0 以降が必要です。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service ドメインでデータストリームを開始するのに役立つ基本的な手順について説明します。包括的なドキュメントについては、OpenSearch ドキュメンテーションの「Data streams」を参照してください。

Data Streams の使用開始

データストリームは、内部的に複数のバッキングインデックスで構成されます。検索リクエストはすべてのバッキングインデックスにルーティングされますが、インデックス作成リクエストは最新の書き込みインデックスにルーティングされます。

ステップ 1: インデックステンプレートを作成する

データストリームを作成するには、まずインデックスのセットをデータストリームとして設定するインデックステンプレートを作成する必要があります。data_stream オブジェクトは、通常のインデックステンプレートではなくデータストリームであることを示します。インデックスパターンは、次のデータストリームの名前と一致します。

```
PUT _index_template/logs-template
{
  "index_patterns": [
    "my-data-stream",
    "logs-*"
  ],
  "data_stream": {},
  "priority": 100
}
```

この場合、取り込まれた各ドキュメントには @timestamp フィールドがある必要があります。また、独自のカスタムタイムスタンプフィールドを data_stream オブジェクトのプロパティとして定義できます。

```
PUT _index_template/logs-template
{
  "index_patterns": "my-data-stream",
  "data_stream": {
    "timestamp_field": {
      "name": "request_time"
    }
  }
}
```

ステップ 2: データストリームの作成

インデックステンプレートを作成したら、データストリームを作成せずにデータの取り込みを直接開始できます。

data_stream オブジェクトと一致するインデックステンプレートがあるので、OpenSearch は自動的にデータストリームを作成します。

```
POST logs-staging/_doc
{
  "message": "login attempt failed",
  "@timestamp": "2013-03-01T00:00:00"
}
```

手順 3: データストリームにデータを取り込む

データストリームにデータを取り込むには、通常のインデックス API を使用できます。インデックスを作成するすべてのドキュメントにタイムスタンプフィールドがあることを確認します。タイムスタンプフィールドのないドキュメントを取り込もうとすると、エラーが発生します。

```
POST logs-redis/_doc
{
  "message": "login attempt",
  "@timestamp": "2013-03-01T00:00:00"
}
```

ステップ 4: データストリームの検索

通常のインデックスやインデックスエイリアスを検索するのと同じように、データストリームを検索できます。検索オペレーションは、すべてのバッキングインデックス (ストリーム内に存在するすべてのデータ) に適用されます。

```
GET logs-redis/_search
{
  "query": {
    "match": {
      "message": "login"
    }
  }
}
```

手順 5: データストリームのロールオーバー

インデックスステート管理 (ISM) ポリシーをセットアップして、データストリームのロールオーバープロセスを自動化できます。ISM ポリシーは、作成時にバッキングインデックスに適用されます。ポリシーをデータストリームに関連付けると、そのデータストリームの将来のバッキングインデックスにのみ影響します。また、`rollover_alias` 設定を提供する必要はありません。ISM ポリシーがバッキングインデックスからこの情報を推測するからです。

> **Note**
>
> バッキングインデックスをコールドストレージに移行する場合、OpenSearch はこのインデックスをデータストリームから削除します。インデックスを UltraWarm に戻しても、インデックスは元のデータストリームの一部ではなく、独立したままです。インデックスがデータストリームから削除された後、ストリームに対して検索しても、インデックスのデータは返されません。
>
> スナップショット復旧の制限: データストリームバッキングインデックスに UltraWarm 階層データが含まれている場合、スナップショット復旧ではデータストリームの状態が完全に復元されない場合があります。完全なデータストリームの復旧を目的としたスナップショットを作成する前に、すべてのバッキングインデックスがホットストレージにあることを確認します。

> **Warning**
>
> データストリームの書き込みインデックスは、コールドストレージに移行できません。データストリーム内のデータをコールドストレージに移行する場合は、移行前にデータストリームをロールオーバーする必要があります。

ステップ 6: OpenSearch Dashboards で Data Streams を管理する

OpenSearch Dashboards から Data Streams を管理するには、[OpenSearch Dashboards] を開き、[インデックス管理] を選択し、[インデックス] または [ポリシーマネージドインデックス] を選択します。

ステップ 7: データストリームを削除する

削除オペレーションは、まずデータストリームのバッキングインデックスを削除してから、データストリーム自体を削除します。

データストリームとその非表示のバッキングインデックスすべてを削除するには、次の手順に従います。

```
DELETE _data_stream/name_of_data_stream
```

Amazon OpenSearch Service でのデータのモニタリング

アラートと異常検出を用いて Amazon OpenSearch Service 内のデータの積極的なモニタリングを行います。アラートをセットアップして、データが特定のしきい値を超えたときに通知を受信します。異常検出では、機械学習の使用により、ストリーミングデータ内の異常値が自動的に検出されます。異常検出とアラートを組み合わせると、異常が検出された場合にすぐに通知を受けることができます。

トピック

- Amazon OpenSearch Service でのアラートの設定
- Amazon OpenSearch Service での異常検出

Amazon OpenSearch Service でのアラートの設定

1 つ以上のインデックスのデータが特定の条件を満たしたときに通知されるように Amazon OpenSearch Service でアラートを設定します。例えば、アプリケーションで 1 時間に 5 つを超える HTTP 503 エラーが記録された場合に E メールが届くようにしたり、過去 20 分以内に新しいドキュメントにインデックスが付けられなかった場合にデベロッパーに連絡したりできます。

アラートには OpenSearch または Elasticsearch 6.2 以降が必要です。

> **Note**
>
> このドキュメントでは、アラートの概要と、Amazon OpenSearch Service ドメインでのアラートとオープンソースの OpenSearch クラスターでのアラートとの違いについて説明します。包括的な API リファレンス、複合モニタで使用可能なリクエストフィールドのリスト、使用可能なトリガー変数とアクション変数の説明など、アラートに関する詳細なドキュメントについては、OpenSearch ドキュメントの「Alerting」を参照してください。

トピック

- アラートの許可
- アラートの開始方法
- 通知
- 相違点

アラートの許可

アラートはきめ細かなアクセスコントロールをサポートしています。ユースケースに適合する許可のミキシングとマッチングの詳細については、OpenSearch ドキュメントのアラートセキュリティを参照してください。

OpenSearch Dashboards 内の [アラート] ページにアクセスするには、少なくとも `alerting_read_access` 事前定義ロールにマッピングするか、同等のアクセス許可を付与する必要があります。このロールは、アラート、送信先、モニタを表示するアクセス許可を付与しますが、アラートの確認や送信先やモニタの変更を行うアクセス許可を付与しません。

アラートの開始方法

アラートを作成するには、モニタを設定します。モニタとは、定義されたスケジュールで実行され、OpenSearch インデックスをクエリするジョブです。また、1 つまたは複数のトリガーも設定します。イベントを生成する条件をトリガーで定義します。最後に、アクションを設定します。アラートがトリガーされるとアクションが実行されます。

アラートを開始するには

1. OpenSearch Dashboards のメインメニューから [アラート] を選択し、[モニタを作成] を選択します。
2. クエリごと、バケットごと、クラスターごとのメトリクス、またはドキュメントごとのモニタを作成します。手順については、「Create a monitor」を参照してください。
3. [Triggers] (トリガー) では、1 つまたは複数のトリガーを作成します。手順については、「Create triggers」(トリガーの作成) を参照してください。
4. [Actions] (アクション) では、アラートの通知チャネルを設定します。Slack、Amazon Chime、カスタム webhook、Amazon SNS のいずれかを選択します。ご想像のとおり、通知にはチャネルへの接続が必要です。例えば、Slack チャンネルに通知したり、カスタム webhook をサードパーティーのサーバーに送信したりするために、OpenSearch Service ドメインは適切なインターネット接続を備えている必要があります。OpenSearch Service ドメインがカスタム webhook にアラートを送信するには、カスタム webhook にパブリック IP アドレスが必要です。

> **Tip**
>
> アクションが正常にメッセージを送信した後、そのメッセージへのアクセスの保護 (Slack チャネルへのアクセスなど) はユーザーの責任となります。ドメインに機密デー

タが含まれている場合は、アクションなしでトリガーを使用し、Dashboards でアラートを定期的にチェックすることを検討してください。

通知

アラートは、OpenSearch 通知の統合システムである通知と統合されます。通知により、どの通信サービスを使用するかを設定したり、関連する統計やトラブルシューティング情報を確認したりできます。包括的なドキュメントについては、OpenSearch ドキュメンテーションの「Notifications」(通知) を参照してください。

通知を使用するには、ドメインが OpenSearch バージョン 2.3 以降を実行している必要があります。

> **Note**
>
> OpenSearch 通知は OpenSearch Service 通知とは別のものです。OpenSearch Service 通知は、サービスソフトウェアの更新、Auto-Tune の強化、およびその他の重要なドメインレベルの情報に関する詳細を提供します。OpenSearch 通知はプラグイン固有です。

OpenSearch バージョン 2.0 以降、アラート送信先は通知チャネルに置き換わりました。送信先は正式に廃止され、今後すべてのアラート通知はチャネルを通じて管理されます。

ドメインをバージョン 2.3 以降にアップグレードすると (OpenSearch Service の 2.x のサポートは 2.3 から始まるため)、既存の送信先は自動的に通知チャネルに移行されます。送信先が移行できなかった場合、モニタは通知チャネルに移行されるまでその送信先を使用し続けます。詳細については、OpenSearch ドキュメントの「Questions about destinations」(送信先に関する質問) を参照してください。

通知の使用を開始するには、OpenSearch Dashboards にサインインして、[Notifications] (通知)、[Channels] (チャネル)、[Create channel] (チャネルを作成) を選択します。

Amazon Simple Notification Service (Amazon SNS) は、通知でサポートされているチャネルタイプです。ユーザーを認証するには、Amazon SNS へのフルアクセス権をユーザーに付与するか、Amazon SNS へのアクセス許可を持つ IAM ロールをユーザーに割り当てる必要があります。手順については、[Amazon SNS as a channel type] (チャネルタイプとしての Amazon SNS) を参照してください。

相違点

オープンソースバージョンの OpenSearch と比較して、Amazon OpenSearch Service でのアラートにはいくつかの顕著な違いがあります。

アラート設定

OpenSearch Service では、次のアラート設定を変更できます。

- `plugins.scheduled_jobs.enabled`
- `plugins.alerting.alert_history_enabled`
- `plugins.alerting.alert_history_max_age`
- `plugins.alerting.alert_history_max_docs`
- `plugins.alerting.alert_history_retention_period`
- `plugins.alerting.alert_history_rollover_period`
- `plugins.alerting.filter_by_backend_roles`

他のすべての設定では、変更できないデフォルト値が使用されます。

アラートを無効にするには、次のリクエストを送信します。

```
PUT _cluster/settings
{
  "persistent" : {
    "plugins.scheduled_jobs.enabled" : false
  }
}
```

次のリクエストでは、デフォルトでの 30 日後ではなく、7 日後に履歴インデックスを自動的に削除するようにアラートを設定します。

```
PUT _cluster/settings
{
  "persistent": {
    "plugins.alerting.alert_history_retention_period": "7d"
  }
}
```

以前にモニタを作成したことがあり、日次アラートインデックスの作成を停止する場合は、すべてのアラート履歴インデックスを削除します。

```
DELETE .plugins-alerting-alert-history-*
```

履歴インデックスのシャード数を減らすには、インデックステンプレートを作成します。次のリクエストは、1 つのシャードと 1 つのレプリカにアラートを送信するための履歴インデックスを設定します。

```
PUT _index_template/template-name
{
  "index_patterns": [".opendistro-alerting-alert-history-*"],
  "template": {
    "settings": {
      "number_of_shards": 1,
      "number_of_replicas": 1
    }
  }
}
```

データ損失の許容値によっては、ゼロレプリカの使用を検討することもできます。インデックステンプレートの作成と管理の詳細については、OpenSearch ドキュメントのインデックステンプレートを参照してください。

Amazon OpenSearch Service での異常検出

Amazon OpenSearch Service の異常検出機能は、ランダムカットフォレスト (RCF) アルゴリズムを使用して、ほぼリアルタイムで OpenSearch データ内の異常を自動的に検出します。RCFは、受信データストリームのスケッチをモデル化する教師なしの機械学習アルゴリズムです。このアルゴリズムにより、受信データポイントごとに、`anomaly grade` および `confidence score` の値が計算されます。異常検出は、これらの値を使用して、データの通常の変動と異常を区別します。

異常検出プラグインとアラートプラグインを組み合わせると、異常が検出された場合にすぐに通知を受けることができます。

異常検出は、OpenSearch バージョンまたは Elasticsearch 7.4 以降を実行しているドメインで使用できます。`t2.micro` と `t2.small` を除くすべてのインスタンスタイプで、異常検出がサポートされています。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service のコンテキストにおける異常検出の概要を説明します。詳細な手順、API リファレンス、使用可能なすべての設定のリファレンス、視覚化とダッシュボードを作成する手順を含む包括的なドキュメントについては、オープンソースの OpenSearch ドキュメントの「Anomaly detection」を参照してください。

前提条件

異常検出には、次のような前提条件があります。

- 異常検出には OpenSearch または Elasticsearch 7.4 以降が必要です。
- 異常検出では、きめ細かなアクセスコントロールを Elasticsearch バージョン 7.9 以降とすべての バージョンの OpenSearch で利用できます。Elasticsearch 7.9 より前のバージョンでは、ディテクターを作成、表示、管理できるのは管理者ユーザーのみです。
- ドメインできめ細かなアクセスコントロールを使用する場合、管理者以外のユーザーは、ディテクターを表示するには OpenSearch Dashboards の `anomaly_read_access` ロールに、またはディテクターを作成および管理するには `anomaly_full_access` にマッピングされている必要があります。

異常検出の開始方法

開始するには、OpenSearch Dashboards で [異常検出] を選択します。

ステップ 1: ディテクターを作成する

ディテクターは、個別の異常検出タスクです。複数のディテクターを作成し、すべてのディテクターを同時に実行すると、それぞれ別々のソースからのデータを分析できます。

ステップ 2: ディテクターに機能を追加する

機能とは、異常をチェックするインデックス内のフィールドを指します。ディテクターでは、1 つ以上の機能について異常を検出できます。機能ごとに、以下の集計の 1 つを選択する必要があります: `average()`、`sum()`、`count()`、`min()`、または `max()`。

> **Note**
>
> `count()` の集計方法は、OpenSearch および Elasticsearch 7.7 以降でのみ利用できます。Elasticsearch 7.4では、次のようなカスタム式を使用します。
>
> ```
> {
>   "aggregation_name": {
>      "value_count": {
>         "field": "field_name"
>      }
>   }
> }
> ```

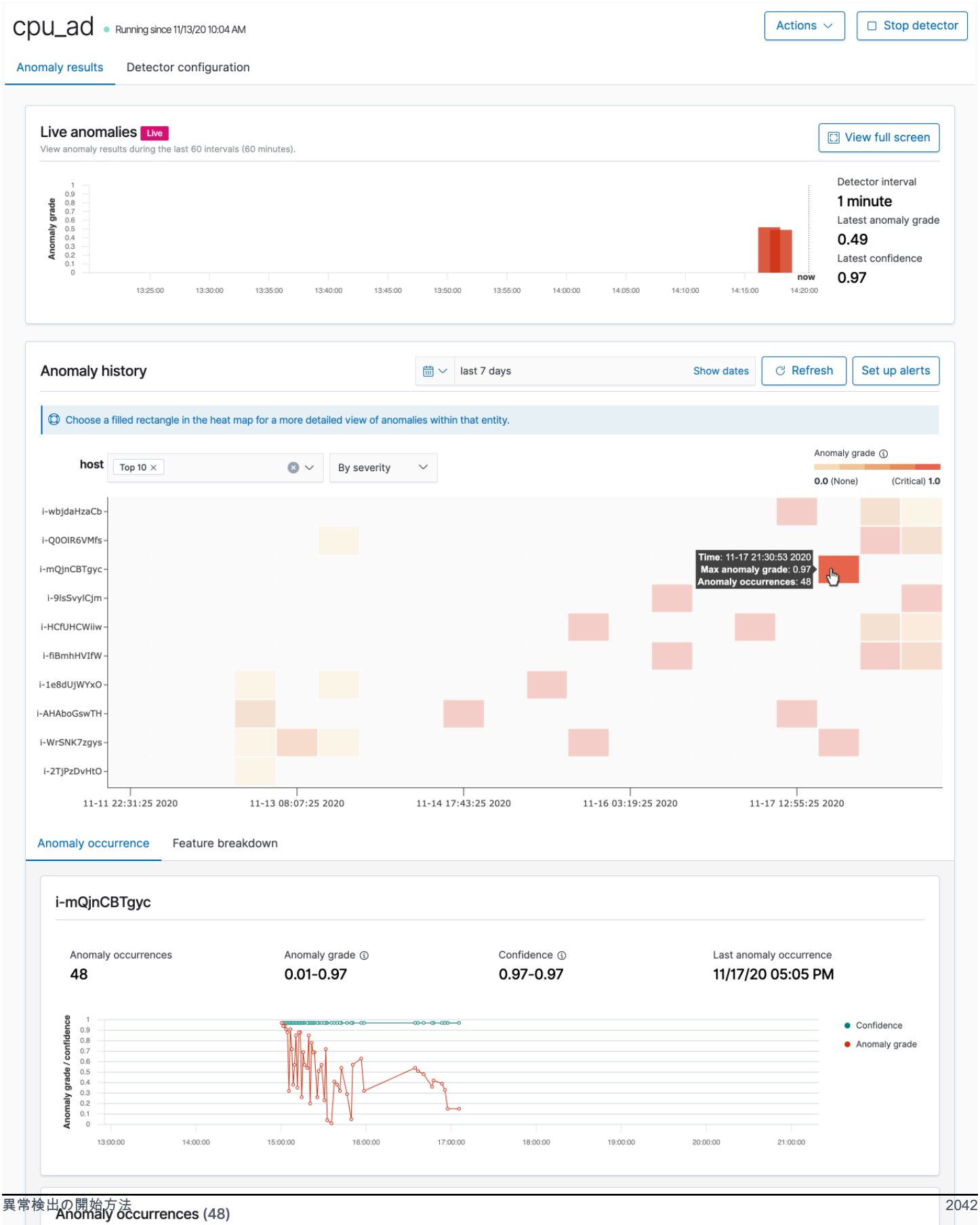
集計方法によって、検出される異常の性質も異なります。例えば、`min()` を選択した場合、ディテクターは機能の最小値に基づいて異常を検出します。`average()` を選択した場合、ディテクターは機能の平均値に基づいて異常を検出します。ディテクターごとに追加できる機能は、最大 5 つです。

以下のオプション設定を指定できます (Elasticsearch 7.7 以降で使用可能)。

- Category field (カテゴリフィールド) - IPアドレス、製品 ID、国コードなどのディメンションでデータを分類またはスライスします。
- Window size (ウィンドウサイズ) - 検出ウィンドウで考慮するデータストリームからの集計間隔の数を設定します。

機能を設定したならば、サンプルの異常をプレビューし、必要に応じて機能の設定を調整します。

ステップ 3: 結果を観察する



- Live anomalies (ライブ異常) - 過去 60 間隔のライブ異常結果を表示します。たとえば、間隔を 10 に設定すると、過去 600 分間の結果が表示されます。このグラフは 30 秒ごとに更新されます。
- Anomaly history (異常履歴) - 対応する測定指標とともに、異常の等級を示します。
- Feature breakdown (機能分類) - 集計方法に基づいて機能を分類します。ディテクターの日時の範囲は変更できます。
- Anomaly occurrence (異常出現) - 検出された各異常の `Start time`、`End time`、`Data confidence`、および `Anomaly grade` を示します。

  カテゴリフィールドを設定すると、異常エンティティの結果を関連付ける、追加のヒートマップチャートが表示されます。塗りつぶされた長方形を選択すると、異常のより詳細なビューが表示されます。

ステップ 4: アラートのセットアップ

異常が検出されたときに通知を受け取るためのモニタを作成するには、[アラートのセットアップ] を選択します。プラグインにより、アラートを設定できる [モニタを追加] ページにリダイレクトされます。

チュートリアル: 異常検出で高い CPU 使用率を検出する

このチュートリアルでは、高い CPU 使用率を検出するために Amazon OpenSearch Service で異常ディテクターを作成する方法を示します。OpenSearch Dashboards を使用して、CPU 使用率をモニタリングし、CPU 使用率が指定されたしきい値を超えたときにアラートを生成するようにディテクターを設定します。

> **Note**
>
> これらのステップは最新バージョンの OpenSearch に適用され、過去のバージョンでは少し異なる場合があります。

前提条件

- Elasticsearch 7.4 以降、または OpenSearch の任意のバージョンを実行する OpenSearch Service ドメインが必要です。
- CPU 使用率データを含むアプリケーションログファイルをクラスターに取り込む必要があります。

ステップ 1: ディテクターを作成する

まず、CPU 使用率データの異常を識別するディテクターを作成します。

1. OpenSearch Dashboards の左側のパネルメニューを開き、[Anomaly Detection] (異常検出) を選択してから、[Create detector] (ディテクターを作成) を選択します。
2. ディテクターに **high-cpu-usage** という名前を付けます。
3. データソース用に、異常を特定する CPU 使用率ログファイルを含むインデックスを選択します。
4. データの [Timestamp field] (Timestamp フィールド) を選択します。必要に応じて、データフィルターを追加できます。このデータフィルターは、データソースのサブセットのみを分析し、無関係なデータのノイズを低減します。
5. [Detector interval] (ディテクターの間隔) を 2 分に設定します。この間隔は、ディテクターがデータを収集する時間 (1 分単位) を定義します。
6. [Window delay] (ウィンドウの遅延) で、[1-minute] (1 分間) の遅延を追加します。この遅延により、ウィンドウ内のすべてのデータが確実に存在しているようにするために、処理時間が長くなります。
7. [次へ] を選択します。異常検出ダッシュボードで、ディテクター名の下にある [Configure model] (モデルを設定) を選択します。
8. [Feature name] (機能名) で、**max_cpu_usage** を入力します。[Feature state] (機能の状態) で、[Enable feature] (機能を有効にする) を選択します。
9. [Find anomalies based on] (次に基づいて異常を検索) で、[Field value] (フィールド値) を選択します。
10. [Aggregation method] (集計方法) で、[**max()**] を選択します。
11. [Field] (フィールド) で、異常をチェックするデータ内のフィールドを選択します。例えば、`cpu_usage_percentage` と呼ばれることがあります。
12. 他のすべての設定をデフォルトのままにして、[Next] (次へ) を選択します。
13. ディテクタージョブの設定を無視して、[Next] (次へ) を選択します。
14. ポップアップウィンドウで、(自動または手動で) ディテクターをいつ開始するかを選択し、[Confirm] (確認) を選択します。

これでディテクターが設定され、初期化された後に、ディテクターパネルの [Real-time results] (リアルタイム結果) セクションで CPU 使用率のリアルタイム結果を確認できるようになります。[Live

anomalies] (ライブ異常) セクションには、データがリアルタイムで取り込まれているときに発生する異常が表示されます。

ステップ 2: アラートを設定する

ディテクターを作成したので、ディテクター設定で指定された条件を満たす CPU 使用率を検出した場合に Slack にメッセージを送信するよう促すアラートを呼び出すモニタを作成します。1 つ以上のインデックスのデータがアラートを呼び出す条件を満たすと、Slack の通知が届きます。

1. OpenSearch Dashboards の左側のパネルメニューを開き、[アラート] を選択してから、[モニタを作成] を選択します。
2. モニタ名前を指定します。
3. [モニタの種類] で、[クエリごとのモニタ] を選択します。クエリごとのモニタは、指定されたクエリを実行し、トリガーを定義します。
4. [Monitor defining method] (モニタリングの定義方法) で [Anomaly detector] (異常ディテクター) を選択し、前のセクションで作成したディテクターを [Detector] (ディテクター) ドロップダウン メニューから選択します。
5. [スケジュール] で、モニタがデータを収集する頻度とアラートを受け取る頻度を選択します。このチュートリアルでは、7 分ごとに実行するようにスケジュールを設定します。
6. [Triggers] (トリガー) セクションで、[Add trigger] (トリガーを追加) を選択します。[Trigger name] (トリガー名) で、**`High CPU usage`** を入力します。このチュートリアルでは、[Severity level] (重大度レベル) で、最高レベルの重大度である [1] を選択します。
7. [Anomaly grade threshold] (異常グレードのしきい値) で、[IS ABOVE] (超える) を選択します。その下のメニューで、適用するグレードしきい値を選択します。このチュートリアルでは、[Anomaly grade] (異常グレード) を [0.7] に設定します。
8. [Anomaly confidence threshold] (異常の信頼度のしきい値) で、[IS ABOVE] (超える) を選択します。その下のメニューで、[Anomaly grade] (異常グレード) と同じ数値を入力します。このチュートリアルでは、[Anomaly confidence threshold] (異常の信頼度のしきい値) を [0.7] に設定します。
9. [Actions] (アクション) セクションで、[Destination] (宛先) を選択します。[Name] (名前) フィールドで、宛先の名前を選択します。[Type] (タイプ) メニューで、[Slack] を選択します。[Webhook URL] (ウェブフック URL) フィールドで、アラートを受信するウェブフック URL を入力します。詳細については、「Sending messages using incoming webhooks」(着信ウェブフックを使用したメッセージの送信) を参照してください。
10. [作成] を選択します。

関連リソース

- the section called “[アラート]”

- the section called “異常検出”
- 異常検出 API

Amazon OpenSearch Service 用の機械学習

ML Commons は、トランスポートおよび REST API コールを通じて一連の一般的な機械学習 (ML) アルゴリズムを提供する OpenSearch プラグインです。これらの呼び出しは、ML リクエストごとに適切なノードとリソースを選択し、ML タスクを監視して稼働時間を確保します。これにより、既存のオープンソースの ML アルゴリズムを活用し、新しい ML 機能の開発に必要な労力を削減できます。プラグインの詳細については、OpenSearch ドキュメントの「Machine learning」をご覧ください。この章では、Amazon OpenSearch Service でプラグインを使用する方法について説明します。

トピック

- 用の Amazon OpenSearch Service ML コネクタ AWS のサービス
- サードパーティプラットフォーム用の Amazon OpenSearch Service ML コネクタ
- CloudFormation を使用してセマンティック検索用のリモート推論をセットアップする
- サポートされていない ML Commons 設定
- OpenSearch Service フローフレームワークテンプレート

用の Amazon OpenSearch Service ML コネクタ AWS のサービス

Amazon OpenSearch Service 機械学習 (ML) コネクタを別の で使用する場合は AWS のサービス、OpenSearch Service をそのサービスに安全に接続する IAM ロールを設定する必要があります。 AWS のサービス これにより、Amazon SageMaker AI と Amazon Bedrock を含めるようにコネクタを設定できます。このチュートリアルでは、OpenSearch Service から SageMaker ランタイムへのコネクタを作成する方法について説明します。コネクタの詳細については、「Supported connectors」を参照してください。

トピック

- 前提条件
- OpenSearch Service コネクタを作成する

前提条件

コネクタを作成するには、Amazon SageMaker AI ドメインエンドポイント、および OpenSearch Service へのアクセスを付与する IAM ロールが必要です。

Amazon SageMaker AI ドメインを設定する

機械学習モデルをデプロイするには、「Amazon SageMaker AI デベロッパーガイド」の「Deploy a Model in Amazon SageMaker AI」を参照してください。AI コネクタを作成するために必要な、モデルのエンドポイント URL をメモしておきます。

IAM ロールを作成する

IAM ロールを設定して、SageMaker ランタイムのアクセス許可を OpenSearch Service に委任します。新しいロールを作成するには、「IAM ユーザーガイド」の「IAM ロールの作成 (コンソール)」を参照してください。なお、同じ権限セットを持っていれば、既存のロールを使用できます。AWS マネージドロールを使用する代わりに新しいロールを作成する場合は、このチュートリアルopensearch-sagemaker-roleの を独自のロールの名前に置き換えます。

1. 以下のマネージド IAM ポリシーを新しいロールにアタッチして、OpenSearch Service が SageMaker AI エンドポイントにアクセスできるようにします。ロールにポリシーをアタッチするには、「Adding IAM identity permissions」を参照してください。

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Action": [
                   "sagemaker:InvokeEndpointAsync",
                   "sagemaker:InvokeEndpoint"
               ],
               "Effect": "Allow",
               "Resource": "*"
           }
       ]
   }
   ```

2. ロールの信頼関係を編集するには、「ロールの信頼ポリシーの変更」の手順に従ってください。次のポリシーで、*service-principal* を OpenSearch Service または OpenSearch Serverless の次のいずれかのサービスプリンシパルに置き換えます。

   OpenSearch Service の場合

   ```
   opensearchservice.amazonaws.com
   ```

OpenSearch Serverless の場合

```
ml.opensearchservice.amazonaws.com
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ml.opensearchservice.amazonaws.com"
                ]
            }
        }
    ]
}
```

aws:SourceAccount および aws:SourceArn 条件キーを使用してアクセスを特定のドメインに制限することをおすすめします。SourceAccount はドメインの所有者に属する AWS アカウント ID で、 SourceArnはドメインの ARN です。例えば、次の条件ブロックを信頼ポリシーに追加できます。

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account-id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"
    }
}
```

アクセス許可の設定

コネクタを作成するには、IAM ロールを OpenSearch Service に渡すアクセス許可が必要です。さらに、es:ESHttpPost アクションへのアクセスも必要です。これらの両方の許可を付与するには、

リクエストの署名に認証情報が使用されている IAM ロールまたはユーザーに次のポリシーをアタッチします。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/opensearch-sagemaker-
role"
        },
        {
            "Effect": "Allow",
            "Action": "es:ESHttpPost",
            "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
        }
    ]
}
```

ユーザーまたはロールがロールを渡すための iam:PassRole アクセス許可を持っていない場合、次のステップでリポジトリを登録しようとすると、承認エラーが発生することがあります。

OpenSearch Dashboards に ML ロールをマッピングします (きめ細かなアクセスコントロールを使用している場合)

きめ細かいアクセス制御では、コネクタの設定時に追加の手順が必要になります。HTTP 基本認証を他のすべての目的で使用する場合でも、opensearch-sagemaker-role を渡すための iam:PassRole 許可を持っている IAM ロールまたはユーザーに ml_full_access ロールをマップする必要があります。

1. OpenSearch Service ドメインの OpenSearch Dashboards プラグインに移動します。OpenSearch Service コンソールのドメインダッシュボードに Dashboards エンドポイントがあります。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[ml_full_access] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。

4. [バックエンドロール] で、opensearch-sagemaker-role を渡すためのアクセス許可がある ロールの ARN を追加します。

```
arn:aws:iam::account-id:role/role-name
```

5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

OpenSearch Service コネクタを作成する

コネクタを作成するには、OpenSearch Service ドメインのエンドポイントに POST リクエストを送信します。署名付きリクエストを送信するには、curl、サンプル Python クライアント、Postman、またはその他の方法を使用できます。Kibana コンソールでは、POST リクエストは使用できないことにご注意ください。 リクエストは以下のような形式です。

```
POST domain-endpoint/_plugins/_ml/connectors/_create
{
   "name": "sagemaker: embedding",
   "description": "Test connector for Sagemaker embedding model",
   "version": 1,
   "protocol": "aws_sigv4",
   "credential": {
      "roleArn": "arn:aws:iam::account-id:role/opensearch-sagemaker-role"
   },
   "parameters": {
      "region": "region",
      "service_name": "sagemaker"
   },
   "actions": [
      {
         "action_type": "predict",
         "method": "POST",
         "headers": {
            "content-type": "application/json"
         },
         "url": "https://runtime.sagemaker.region.amazonaws.com/endpoints/endpoint-id/
invocations",
         "request_body": "{ \"inputs\": { \"question\": \"${parameters.question}\",
 \"context\": \"${parameters.context}\" } }"
      }
   ]
```

```
}
```

ドメインが仮想プライベートクラウド (VPC) に存在する場合は、リクエストが正常に AI コネクタを作成するにはコンピュータが VPC に接続されていることが必要です。VPC へのアクセスはネットワーク構成によって異なりますが、通常は VPN あるいは社内ネットワークへの接続が必要になります。OpenSearch Service ドメインにアクセスできるかを確認するには、ウェブブラウザで `https://your-vpc-domain.region.es.amazonaws.com` を開き、デフォルトの JSON 応答を受信していることを確認します。

Python クライアントのサンプリング

Python クライアントは、HTTP リクエストよりも自動化が簡単で、再利用性も高いです。Python クライアントで AI コネクタを作成するには、以下のサンプルコードを Python ファイルに保存します。このクライアントには AWS SDK for Python (Boto3)、`requests`、`requests-aws4auth` のパッケージが必要です。

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

host = 'domain-endpoint/'
region = 'region'
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

# Register repository
path = '_plugins/_ml/connectors/_create'
url = host + path

payload = {
   "name": "sagemaker: embedding",
   "description": "Test connector for Sagemaker embedding model",
   "version": 1,
   "protocol": "aws_sigv4",
   "credential": {
      "roleArn": "arn:aws:iam::account-id:role/opensearch-sagemaker-role"
   },
   "parameters": {
      "region": "region",
      "service_name": "sagemaker"
```

```
   },
   "actions": [
      {
         "action_type": "predict",
         "method": "POST",
         "headers": {
            "content-type": "application/json"
         },
         "url": "https://runtime.sagemaker.region.amazonaws.com/endpoints/endpoint-id/
invocations",
         "request_body": "{ \"inputs\": { \"question\": \"${parameters.question}\",
 \"context\": \"${parameters.context}\" } }"
      }
   ]
}
headers = {"Content-Type": "application/json"}

r = requests.post(url, auth=awsauth, json=payload, headers=headers)
print(r.status_code)
print(r.text)
```

サードパーティプラットフォーム用の Amazon OpenSearch Service ML コネクタ

このチュートリアルでは、OpenSearch Service から Cohere へのコネクタを作成する方法について説明します。コネクタの詳細については、「Supported connectors」を参照してください。

Amazon OpenSearch Service 機械学習 (ML) コネクタを外部リモートモデルで使用する場合は、特定の権限がある認証情報を AWS Secrets Managerに保存する必要があります。これは API キーでも、ユーザー名とパスワードの組み合わせでもかまいません。つまり、OpenSearch Service のアクセスを Secrets Manager からの読み取りを許可する IAM ロールも作成する必要があるということです。

トピック

- 前提条件
- OpenSearch Service コネクタを作成する

前提条件

Cohere または OpenSearch Service を使用する外部プロバイダーのコネクタを作成するには、認証情報 AWS Secrets Managerを保存する OpenSearch Service へのアクセスを許可する IAM ロールが必要です。また、認証情報は Secrets Manager に保存する必要があります。

さらに、次のネットワーク要件に注意してください。

- ML Connector リクエストは OpenSearch VPC から発信されないため、外部プロバイダーはパブリック IP アドレスからのインバウンドトラフィックを許可する必要があります。
- コネクタが外部エンドポイントと通信できるようにするplugins.ml_commons.trusted_connector_endpoints_regexには、外部プロバイダー URL をクラスター設定に追加する必要があります。

IAM ロールを作成する

IAM ロールを設定して、OpenSearch Service に Secrets Manager のアクセス許可を委任します。既存の SecretManagerReadWrite ロールも使用できます。新しいロールを作成するには、「IAM ユーザーガイド」の「IAM ロールの作成 (コンソール)」を参照してください。 AWS マネージドロールを使用する代わりに新しいロールを作成する場合は、このチュートリアルopensearch-secretmanager-roleの を独自のロールの名前に置き換えます。

1. 以下のマネージド IAM ポリシーを新しいロールにアタッチして、OpenSearch Service が Secrets Manager の値にアクセスできるようにします。ロールにポリシーをアタッチするには、「IAM ID アクセス許可の追加」を参照してください。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

2. ロールの信頼関係を編集するには、「ロールの信頼ポリシーの変更」の手順に従ってください。次のポリシーで、*service-principal* を OpenSearch Service または OpenSearch Serverless OpenSearch の次のいずれかのサービスプリンシパルに置き換えます。

   OpenSearch Service の場合

   ```
   opensearchservice.amazonaws.com
   ```

   OpenSearch Serverless の場合

   ```
   ml.opensearchservice.amazonaws.com
   ```

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Action": [
                   "sts:AssumeRole"
               ],
               "Effect": "Allow",
               "Principal": {
                   "Service": [
                       "service-principle"
                   ]
               }
           }
       ]
   }
   ```

   aws:SourceAccount および aws:SourceArn 条件キーを使用してアクセスを特定のドメインに制限することをお勧めします。SourceAccount はドメインの所有者に属する AWS アカウント ID で、 SourceArnはドメインの ARN です。例えば、次の条件ブロックを信頼ポリシーに追加できます。

   ```
   "Condition": {
       "StringEquals": {
           "aws:SourceAccount": "account-id"
       },
       "ArnLike": {
           "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"
       }
   ```

```
}
```

アクセス許可の設定

コネクタを作成するには、IAM ロールを OpenSearch Service に渡すアクセス許可が必要です。さらに、`es:ESHttpPost` アクションへのアクセスも必要です。これらの両方の許可を付与するには、リクエストの署名に認証情報が使用されている IAM ロールまたはユーザーに次のポリシーをアタッチします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::111122223333:role/opensearch-secretmanager-role"
    },
    {
      "Effect": "Allow",
      "Action": "es:ESHttpPost",
      "Resource": "arn:aws:es:us-east-1:111122223333:domain/domain-name/*"
    }
  ]
}
```

ユーザーまたはロールがロールを渡すための `iam:PassRole` アクセス許可を持っていない場合、次のステップでリポジトリを登録しようとすると、承認エラーが発生することがあります。

セットアップ AWS Secrets Manager

認可の認証情報を Secrets Manager に保存するには、「AWS Secrets Manager User Guide」の「Create an AWS Secrets Manager secret」を参照してください。

Secrets Manager がキーと値のペアをシークレットとして受け入れると、次の形式の ARN を受け取ります: `arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret-a1b2c3`。次のステップでコネクタを作成するときに、この ARN はキーと使用するので記録しておいてください。

OpenSearch Dashboards に ML ロールをマッピングします (きめ細かなアクセスコントロールを使用している場合)

きめ細かいアクセス制御では、コネクタの設定時に追加の手順が必要になります。HTTP 基本認証を他のすべての目的で使用する場合でも、`opensearch-sagemaker-role` を渡すための `iam:PassRole` 許可を持っている IAM ロールまたはユーザーに `ml_full_access` ロールをマップする必要があります。

1. OpenSearch Service ドメインの OpenSearch Dashboards プラグインに移動します。OpenSearch Service コンソールのドメインダッシュボードに Dashboards エンドポイントがあります。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[ml_full_access] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. [バックエンドロール] で、`opensearch-sagemaker-role` を渡すためのアクセス許可があるロールの ARN を追加します。

   ```
   arn:aws:iam::account-id:role/role-name
   ```

5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

OpenSearch Service コネクタを作成する

コネクタを作成するには、OpenSearch Service ドメインのエンドポイントに `POST` リクエストを送信します。署名付きリクエストを送信するには、curl、サンプル Python クライアント、Postman、またはその他の方法を使用できます。Kibana コンソールでは、`POST` リクエストは使用できないことにご注意ください。 リクエストは以下のような形式です。

```
POST domain-endpoint/_plugins/_ml/connectors/_create
{
    "name": "Cohere Connector: embedding",
    "description": "The connector to cohere embedding model",
    "version": 1,
    "protocol": "http",
    "credential": {
        "secretArn": "arn:aws:secretsmanager:region:account-id:secret:cohere-key-id",
        "roleArn": "arn:aws:iam::account-id:role/opensearch-secretmanager-role"
    },
```

```
    "actions": [
        {
            "action_type": "predict",
            "method": "POST",
            "url": "https://api.cohere.ai/v1/embed",
            "headers": {
                "Authorization": "Bearer ${credential.secretArn.cohere-key-used-in-
secrets-manager}"
            },
            "request_body": "{ \"texts\": ${parameters.texts}, \"truncate\": \"END\" }"
        }
    ]
}
```

このリクエストのリクエスト本文は、オープンソースのコネクタリクエストのリクエスト本文と 2 つの点で異なります。`credential` フィールド内では、OpenSearch Service が Secrets Manager から読み取ることを許可する IAM ロールの ARN と、シークレットの ARN を渡します。`headers` フィールドでは、シークレットキーとそれが ARN からのものであるという事実を用いてシークレットを参照します。

ドメインが仮想プライベートクラウド (VPC) に存在する場合は、リクエストが AI コネクタを正常に作成するには、コンピュータが VPC に接続されていることが必要です。VPC へのアクセスはネットワーク構成によって異なりますが、通常は VPN あるいは社内ネットワークへの接続が必要になります。OpenSearch Service ドメインにアクセスできるかを確認するには、ウェブブラウザで `https://your-vpc-domain.region.es.amazonaws.com` を開き、デフォルトの JSON 応答を受信していることを確認します。

Python クライアントのサンプリング

Python クライアントは、HTTP リクエストよりも自動化が簡単で、再利用性も高いです。Python クライアントで AI コネクタを作成するには、以下のサンプルコードを Python ファイルに保存します。このクライアントには AWS SDK for Python (Boto3)、`requests`、`requests-aws4auth` のパッケージが必要です。

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

host = 'domain-endpoint/'
region = 'region'
service = 'es'
```

```
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

path = '_plugins/_ml/connectors/_create'
url = host + path

payload = {
    "name": "Cohere Connector: embedding",
    "description": "The connector to cohere embedding model",
    "version": 1,
    "protocol": "http",
    "credential": {
        "secretArn": "arn:aws:secretsmanager:region:account-id:secret:cohere-key-id",
        "roleArn": "arn:aws:iam::account-id:role/opensearch-secretmanager-role"
    },
    "actions": [
        {
            "action_type": "predict",
            "method": "POST",
            "url": "https://api.cohere.ai/v1/embed",
            "headers": {
                "Authorization": "Bearer ${credential.secretArn.cohere-key-used-in-
secrets-manager}"
            },
            "request_body": "{ \"texts\": ${parameters.texts}, \"truncate\": \"END\" }"
        }
    ]
}

headers = {"Content-Type": "application/json"}

r = requests.post(url, auth=awsauth, json=payload, headers=headers)
print(r.status_code)
print(r.text)
```

CloudFormation を使用してセマンティック検索用のリモート推論をセットアップする

OpenSearch バージョン 2.9 以降では、セマンティック検索によるリモート推論を使用して独自の機械学習 (ML) モデルをホストできます。リモート推論は ML Commons プラグインを使用します。

リモート推論を使用すると、Amazon SageMaker AI や Amazon Bedrock などの ML サービス上でモデル推論をリモートでホストし、ML コネクタを使用して Amazon OpenSearch Service に接続することができます。

リモート推論を簡単にセットアップできるように、Amazon OpenSearch Service ではコンソールで AWS CloudFormation テンプレートが提供されます。CloudFormation は、インフラストラクチャ AWS のサービス をコードとして扱うことで、 AWS およびサードパーティーのリソースをプロビジョニング、管理できる です。

OpenSearch CloudFormation テンプレートはモデルのプロビジョニングプロセスを自動化するため、OpenSearch Service ドメインでモデルを簡単に作成し、モデル ID を使用してデータを取り込み、ニューラル検索クエリを実行できます。

OpenSearch Service バージョン 2.12 以降でニューラルスパースエンコーダーを使用する場合は、リモートでデプロイするのではなく、ローカルでトークナイザモデルを使用することをお勧めします。詳細については、OpenSearch ドキュメントの「Sparse encoding models」を参照してください。

トピック

- 使用可能な CloudFormation テンプレート
- 前提条件
- Amazon Bedrock テンプレート
- Bedrock Claude を使用したエージェント検索の設定
- MCP サーバー統合テンプレート
- Amazon SageMaker テンプレート
- セマンティックハイライトテンプレートに対するリモート推論

使用可能な CloudFormation テンプレート

次の AWS CloudFormation 機械学習 (ML) テンプレートを使用できます。

Amazon Bedrock テンプレート

Amazon Titan Text Embeddings の統合

> Amazon Bedrock のホストされた ML モデルに接続し、個別のモデルデプロイが不要になり、事前定義された Amazon Bedrock エンドポイントを使用します。詳細については、「Amazon Bedrock ユーザーガイド」の「Amazon Titan Text Embeddings」を参照してください。

Cohere 埋め込み統合

Cohere Embed モデルへのアクセスを提供し、特定のテキスト処理ワークフロー向けに最適化されています。詳細については、「Cohere ドキュメント」サイトの「Embed」を参照してください。

Amazon Titan Multimodal Embeddings

テキスト埋め込みとイメージ埋め込みの両方をサポートし、マルチモーダル検索機能を有効にします。詳細については、「Amazon Bedrock ユーザーガイド」の「Amazon Titan Multimodal Embeddings」を参照してください。

MCP サーバー統合テンプレート

MCP サーバー統合

Amazon Bedrock AgentCore ランタイムをデプロイし、エージェントエンドポイントを提供し、インバウンド認証とアウトバウンド認証を処理し、エンタープライズ認証用の OAuth をサポートします。

Amazon SageMaker テンプレート

Amazon SageMaker によるテキスト埋め込みモデルとの統合

Amazon SageMaker ランタイムにテキスト埋め込みモデルをデプロイし、モデルアーティファクトアクセス用の IAM ロールを作成し、セマンティック検索用の ML コネクタを確立します。

SageMaker を介したスパースエンコーダーとの統合

ニューラル検索用のスパースエンコーディングモデルを設定し、コネクタ管理用の AWS Lambda 関数を作成し、すぐに使用できるようにモデル ID を返します。

前提条件

CloudFormation テンプレートを OpenSearch Service で使用するには、次の前提条件を満たす必要があります。

OpenSearch Service ドメインをセットアップする

CloudFormation テンプレートを使用するには、バージョン 2.9 以降の Amazon OpenSearch Service ドメインをセットアップし、詳細なアクセスコントロールを有効にする必要がありま

す。OpenSearch Service バックエンドロールを作成して、コネクタを作成するアクセス許可を ML Commons プラグインに付与します。

CloudFormation テンプレートは、デフォルト名 `LambdaInvokeOpenSearchMLCommonsRole` で Lambda IAM ロールを作成します。別の名前使用する場合は、デフォルト名をオーバーライドできます。この IAM ロールがテンプレートによって作成されたら、OpenSearch Service ドメインを呼び出すアクセス許可を Lambda 関数に付与する必要があります。そのためには、以下の手順に従って、`ml_full_access` という名前のロールを OpenSearch Service バックエンドロールにマッピングします。

1. OpenSearch Service ドメインの OpenSearch Dashboards プラグインに移動します。OpenSearch Service コンソールのドメインダッシュボードに Dashboards エンドポイントがあります。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[ml_full_access] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. [バックエンドロール] で、ドメインを呼び出すアクセス許可を必要とする Lambda ロールの ARN を追加します。

   ```
   arn:aws:iam::account-id:role/role-name
   ```

5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

ロールをマッピングしたら、ドメインのセキュリティ設定に移動し、Lambda IAM ロールを OpenSearch Service アクセスポリシーに追加します。

でアクセス許可を有効にする AWS アカウント

には、テンプレートに AWS のサービス 選択した SageMaker ランタイムまたは Amazon Bedrock とともに、CloudFormation と Lambda へのアクセス許可 AWS アカウント が必要です。

Amazon Bedrock を使用している場合、モデルも登録する必要があります。モデルを登録するには、「Amazon Bedrock ユーザーガイド」の「Model access」を参照してください。

独自の Amazon S3 バケットを使用してモデルアーティファクトを提供する場合は、CloudFormation IAM ロールを S3 アクセスポリシーに追加する必要があります。詳細については、 IAM ユーザーガイドの「IAM ID アクセス許可の追加および削除」を参照してください。

Amazon Bedrock テンプレート

Amazon Bedrock CloudFormation テンプレートは、OpenSearch Service と Amazon Bedrock の間にコネクタを作成するために必要な AWS リソースをプロビジョニングします。

まず、テンプレートは、将来の Lambda 関数が OpenSearch Service ドメインにアクセスすることを許可する IAM ロールを作成します。次に、テンプレートは、ドメインで ML Commons プラグインを使用してコネクタを作成する Lambda 関数を作成します。OpenSearch Service がコネクタを作成すると、リモート推論のセットアップが完了し、Amazon Bedrock API オペレーションを使用してセマンティック検索を実行できるようになります。

> **Note**
>
> Amazon Bedrock は独自の ML モデルをホストしているため、SageMaker ランタイムにモデルをデプロイする必要はありません。代わりに、テンプレートは Amazon BedRock 用の事前定義されたエンドポイントを使用し、エンドポイントのプロビジョニング手順をスキップします。

Amazon Bedrock CloudFormation テンプレートを使用するには

1. Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインから、[統合] を選択します。
3. [Amazon Bedrock を介して Amazon Titan Text Embeddings モデルと統合] で、[ドメインを設定]、[パブリックドメインを設定] を選択します。
4. プロンプトの指示に従ってモデルをセットアップします。

> **Note**
>
> OpenSearch Service では、Amazon VPC ドメインを構成するための専用テンプレートも提供されています。このテンプレートを使用する場合は、Lambda 関数用の Amazon VPC ID を指定する必要があります。

さらに、OpenSearch Service では、Cohere モデルと Amazon Titan Multimodal Embeddings モデルに接続するための次の Amazon Bedrock テンプレートが用意されています:

- Integration with Cohere Embed through Amazon Bedrock
- Integrate with Amazon Bedrock Titan Multi-modal

Bedrock Claude を使用したエージェント検索の設定

エージェント検索は、自律型エージェントを活用して、ユーザーインテントを理解し、適切なツールをオーケストレーションし、最適化されたクエリを生成し、自然言語インターフェイスを通じて意思決定の透過的な概要を提供することで、ユーザーに代わって複雑な検索を実行します。これらのエージェントは、Bedrock Claude などの推論モデルを利用しています。

以下のステップに従って、エージェント検索用に Bedrock Claude モデルを自動的に設定する CloudFormation テンプレートを開き、実行します。また、OpenSearch Dashboards の AI Search Flows プラグインでエージェントを設定および作成する方法について説明します。

Bedrock Claude アクセスの有効化

1. 前提条件: ドメインがきめ細かなアクセスコントロールを使用している場合は、テンプレートを実行する前にバックエンドロール`arn:aws:iam::`*`your-account-id`*`:role/`*`LambdaInvokeOpenSearchMLCommonsRole`*として`ml_full_access`ロールにマッピングします。この IAM ロールが存在しない場合、CloudFormation によって自動的に作成されます。マッピングの設定方法の詳細については、「」を参照してくださいOpenSearch Dashboards に ML ロールをマッピングします (きめ細かなアクセスコントロールを使用している場合)。
2. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
3. 左側のナビゲーションペインから、[統合] を選択します。
4. エージェント検索用の Bedrock Claude との統合で、ドメインの設定を選択します。ドメインがバージョン 3.3 以降であることを確認します。
5. CloudFormation テンプレートで、OpenSearch Service ドメインエンドポイントを入力し、モデルを選択します。残りのフィールドはオプションまたは事前入力済みです。スタックの作成をクリックし、プロビジョニングが完了するまで待ちます。
6. Amazon OpenSearch Service コンソールからドメインを選択し、ドメインを選択します。OpenSearch Dashboards URL をクリックして、OpenSearch Dashboards にアクセスします。

エージェントの構築とエージェント検索の実行

1. OpenSearch Dashboards から、左側にあるメニューを開きます。OpenSearch Plugins > AI Search Flows を選択してプラグインにアクセスします。
2. ワークフローページで新しいワークフロータブを選択し、エージェント検索カードで作成をクリックします。
3. 検索設定に一意の名前を指定し、作成をクリックします。
4. エージェントの設定 で、新しいエージェントの作成 をクリックします。新しく作成した Bedrock Claude モデルを選択し、エージェントの作成をクリックします。ボタンが無効になっている場合は、詳細設定 > LLM インターフェイスをチェックし、有効なインターフェイスが選択されていることを確認します。CloudFormation のすべてのモデルは Bedrock Claude モデルになるため、Bedrock Claude をまだ選択していない場合は、エージェントの作成をクリックします。
5. テストフローで、エージェント検索を実行してみてください。自然言語検索クエリを指定し、検索をクリックします。

AI Search Flows プラグインの完全なドキュメントについては、OpenSearch ドキュメントの「エージェント検索の設定」を参照してください。

エージェント検索の仕組みの詳細については、OpenSearch ドキュメントの「エージェント検索」を参照してください。

MCP サーバー統合テンプレート

モデルコンテキストプロトコル (MCP) サーバーテンプレートを使用すると、OpenSearch がホストする MCP サーバーを Amazon Bedrock AgentCore にデプロイできるため、AI エージェントと OpenSearch ツールの統合の複雑さを軽減できます。詳細については、「Amazon Bedrock AgentCore とは何ですか?」を参照してください

テンプレートの機能

このテンプレートには、MCP サーバーをデプロイおよび管理するための以下の主要な機能が含まれています。

マネージド MCP サーバーのデプロイ

Amazon Bedrock AgentCore ランタイムを使用して [opensearch-mcp-server-py] をデプロイし、基盤となる MCP サーバーにリクエストをプロキシするエージェントエンドポイントを提供します。詳細については、GitHub の「opensearch-mcp-server-py」を参照してください。

認証とセキュリティ

インバウンド認証 (ユーザーから MCP サーバーへ) とアウトバウンド認証 (MCP サーバーから OpenSearch へ) の両方を処理し、エンタープライズ認証用の OAuth をサポートします。

> **Note**
>
> MCP サーバーテンプレートは、次の AWS リージョンでのみ使用できます:
>
> - 米国東部 (バージニア北部)
> - 米国西部 (オレゴン)
> - 欧州 (フランクフルト)
> - アジアパシフィック (シドニー)

MCP サーバーテンプレートを使用するには

MCP サーバーテンプレートをデプロイし、OpenSearch ドメインに接続するには、次の手順に従います。

1. Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインから、[統合] を選択します。
3. [MCP サーバー統合] テンプレートを検索します。
4. [ドメインの設定] を選択します。次に、OpenSearch ドメインエンドポイントを入力します。

対応するオプションパラメータが指定されていない場合、テンプレートは AgentCore Runtime と以下のコンポーネントを作成します。

- Amazon ECR リポジトリ
- オーソライザーとしての Amazon Cognito ユーザープール
- AgentCore Runtime によって使用される実行ロール

この手順を完了したら、作成後のステップに従う必要があります:

1. Amazon OpenSearch Service の場合: 実行ロール ARN を OpenSearch バックエンドロールにマッピングして、ドメインへのアクセスを制御します。

   Amazon OpenSearch Serverless の場合: 実行ロールがコレクションにアクセスできるようにするデータアクセスポリシーを作成します。
2. オーソライザーから OAuth アクセストークンを取得します。次に、このトークンを使用して、CloudFormation スタック出力にリストされている URL で MCP サーバーにアクセスします。

詳細については、「OpenSearch Serverless でのポリシーアクション」を参照してください。

AI エージェントとの統合

デプロイ後、MCP サーバーを任意の MCP 互換エージェントと統合できます。詳細については、「Amazon Bedrock デベロッパーガイド」の「Invoke your deployed MCP server」を参照してください。

デベロッパー統合

MCP サーバーのエンドポイントをエージェント設定に追加できます。Kiro CLI、カスタムエージェント、またはその他の MCP 互換エージェントでも使用できます。

エンタープライズデプロイ

中央ホスト型エージェントは、OpenSearch を構成要素の 1 つとして複数のサービスに接続できます。このエージェントは、OAuth およびエンタープライズ認証システムをサポートし、複数のユーザーとユースケースをサポートするようにスケーリングします。

Strands Agents フレームワークの使用例

```
import os
import requests
from strands import Agent
from strands.tools.mcp import MCPClient
from mcp.client.streamable_http import streamablehttp_client

def get_bearer_token(discovery_url: str, client_id: str, client_secret: str):
    response = requests.get(discovery_url)
    discovery_data = response.json()
    token_endpoint = discovery_data['token_endpoint']
```

```
    data = {
        'grant_type': 'client_credentials',
        'client_id': client_id,
        'client_secret': client_secret
    }
    headers = {
        'Content-Type': 'application/x-www-form-urlencoded'
    }

    response = requests.post(token_endpoint, data=data, headers=headers)
    token_data = response.json()
    return token_data['access_token']

if __name__ == "__main__":
    discovery_url = os.environ["DISCOVERY_URL"]
    client_id = os.environ["CLIENT_ID"]
    client_secret = os.environ["CLIENT_SECRET"]
    mcp_url = os.environ["MCP_URL"]

    bearer_token = get_bearer_token(discovery_url, client_id, client_secret)

    opensearch_mcp_client = MCPClient(lambda: streamablehttp_client(mcp_url, {
        "authorization": f"Bearer {bearer_token}",
        "Content-Type": "application/json"
    }))

    with opensearch_mcp_client:
        tools = opensearch_mcp_client.list_tools_sync()
        agent = Agent(tools=tools)
        agent("list indices")
```

詳細については、OpenSearch ウェブサイトの「Hosting OpenSearch MCP Server with Amazon Bedrock AgentCore」を参照してください。

Amazon SageMaker テンプレート

Amazon SageMaker CloudFormation テンプレートは、ニューラルプラグインとセマンティック検索をセットアップするために複数の AWS リソースを定義します。

まず、[Amazon SageMaker を使用したテキスト埋め込みモデルとの統合] テンプレートを使用して、SageMaker ランタイムにテキスト埋め込みモデルをサーバーとしてデプロイします。モデルエンドポイントを提供しない場合、CloudFormation は SageMaker ランタイムが Amazon S3 からモデ

ルアーティファクトをダウンロードしてサーバーにデプロイすることを許可する IAM ロールを作成します。エンドポイントを提供する場合、CloudFormation は、Lambda 関数が OpenSearch Service ドメインにアクセスすることを許可する IAM ロールを作成します。ロールが既に存在する場合は、ロールを更新して再利用します。エンドポイントは ML Commons プラグインで ML コネクタに使用されるリモートモデルを提供します。

次に、[Amazon Sagemaker を使用したスパースエンコーダーとの統合] テンプレートを使用して、ドメインがリモート推論コネクタをセットアップした Lambda 関数を作成します。OpenSearch Service でコネクタを作成すると、リモート推論は SageMaker ランタイムのリモートモデルを使用してセマンティック検索を実行できます。テンプレートはドメイン内のモデル ID を返すので、検索を開始できます。

Amazon SageMaker AI CloudFormation テンプレートを使用するには

1. Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインから、[統合] を選択します。
3. 各 Amazon SageMaker AI テンプレートで、[ドメインを設定]、[パブリックドメインを設定] を選択します。
4. CloudFormation コンソールのプロンプトに従って、スタックをプロビジョニングし、モデルを設定します。

> **Note**
>
> OpenSearch Service は VPC ベースの OpenSearch ドメインを設定するための個別のテンプレートも提供します。このテンプレートを使用する場合は、Lambda 関数の VPC ID を指定する必要があります。

セマンティックハイライトテンプレートに対するリモート推論

セマンティックハイライトは、キーワードの完全一致のみに依存するのではなく、クエリの意味とコンテキストを分析することで結果の関連性を高める高度な検索機能です。この機能は、機械学習モデルを使用して検索クエリとドキュメントコンテンツ間のセマンティック類似性を評価することで、ドキュメント内で特にコンテキスト関連性が高い文章やフレーズを抽出して強調表示します。従来のハイライト手法が正確な語句の一致に焦点を当てているのに対し、セマンティックハイライトは AI モデルを活用し、クエリと周辺テキスト両方のコンテキスト情報をもとに各文を評価します。これにより、検索語がハイライト対象の文に正確に含まれていない場合でも、関連性の高い情報を

抽出することが可能になります。このアプローチは、ユーザーが文字通りの語句一致よりもセマンティックな関連性を重視する AI 駆動型の検索実装に役立ちます。これにより、検索管理者は、単なるキーワードの出現箇所ではなく、意味のあるコンテンツスパンを強調する、よりインテリジェントでコンテキストに応じた検索エクスペリエンスを提供できます。詳細については、「Using semantic highlighting」を参照してください。

次の手順を使用して、セマンティックハイライト用に Amazon SageMaker モデルを自動的に設定する CloudFormation テンプレートを開き、実行します。

セマンティックハイライト CloudFormation テンプレートを使用するには

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールを開きます。
2. 左側のナビゲーションペインから、[統合] を選択します。
3. [Amazon SageMaker 統合によるセマンティックハイライトの有効化] で、[ドメインの設定]、[パブリックドメインの設定] を選択します。
4. プロンプトの指示に従ってモデルをセットアップします。

> **Note**
>
> OpenSearch Service は VPC ベースの OpenSearch ドメインを設定するための個別のテンプレートも提供します。このテンプレートを使用する場合は、Lambda 関数の VPC ID を指定する必要があります。

サポートされていない ML Commons 設定

Amazon OpenSearch Service では、次の ML Commons 設定の使用をサポートしていません。

- `plugins.ml_commons.allow_registering_model_via_url`
- `plugins.ml_commons.allow_registering_model_via_local_file`

> **Important**
>
> 本番稼働用クラスターでは、クラスター設定 `plugins.ml_commons.only_run_on_ml_node` を無効にしないでください (`false` に設定しないでください)。この保護を無効にするオプションは開発を容易にするためのものです

が、本番稼働用クラスターはコネクタを使用する必要があります。詳細については、「the section called “のコネクタ AWS のサービス”」を参照してください。

ML Commons 設定の詳細については、「ML Commons cluster settings」を参照してください。

OpenSearch Service フローフレームワークテンプレート

Amazon OpenSearch Service フローフレームワークテンプレートを使用すると、一般的なユースケース用のテンプレートを提供することで、複雑な OpenSearch Service のセットアップと前処理タスクを自動化できます。例えば、フローフレームワークテンプレートを使用して機械学習のセットアップタスクを自動化できます。Amazon OpenSearch Service フローフレームワークテンプレートは、JSON または YAML ドキュメントでのセットアッププロセスに関する簡単な説明を提供します。これらのテンプレートでは、会話チャットまたはクエリ生成、AI コネクタ、ツール、エージェント、および OpenSearch Service を生成モデルのバックエンドで使用するために準備するその他のコンポーネントの自動ワークフロー設定について説明されています。

Amazon OpenSearch Service フローフレームワークテンプレートは、特定のニーズに合わせてカスタマイズできます。カスタムフローフレームワークテンプレートの例については、「flow-framework」を参照してください。OpenSearch Service が提供するテンプレートについては、「workflow-templates」を参照してください。詳細な手順、API リファレンス、使用可能なすべての設定のリファレンスを含む包括的なドキュメントについては、オープンソースの OpenSearch ドキュメントの「automating configuration」を参照してください。

> **Note**
>
> フローフレームワークは、OpenSearch Service 2.17 のバックエンドロールフィルタリングをサポートしていません。

OpenSearch Service での ML コネクタの作成

Amazon OpenSearch Service フローフレームワークテンプレートを使用すると、ml-commons で提供される Create Connector API を使用して、ML コネクタを設定およびインストールできます。ML コネクタを使用して、OpenSearch Service を他の AWS サービスやサードパーティーのプラットフォームに接続できます。詳細については、「Creating connectors for third-party ML platforms」を参照してください。Amazon OpenSearch Service フローフレームワーク API を使用す

ると、OpenSearch Service のセットアップタスクと前処理タスクを自動化し、ML コネクタの作成に使用できます。

OpenSearch Service でコネクタを作成する前に、以下を実行する必要があります。

- Amazon SageMaker AI ドメインを作成する
- IAM ロールを作成します。
- パスロールの許可を設定します。
- OpenSearch Dashboards でフローフレームワークロールと ml-commons ロールをマッピングします。

AWS サービス用に ML コネクタを設定する方法の詳細については、AWS 「 サービスの Amazon OpenSearch Service ML コネクタ」を参照してください。OpenSearch Service ML コネクタをサードパーティープラットフォームで使用する方法の詳細については、「のサービス用の Amazon OpenSearch Service ML コネクタ」を参照してください。

フローフレームワークサービスを使用したコネクタの作成

コネクタを使用してフローフレームワークテンプレートを作成するには、OpenSearch Service ドメインエンドポイントに POST リクエストを送信する必要があります。署名付きリクエストを送信するには、cURL、サンプル Python クライアント、Postman、またはその他の方法を使用できます。POST リクエストは以下のような形式です。

```
POST /_plugins/_flow_framework/workflow
{
  "name": "Deploy Claude Model",
  "description": "Deploy a model using a connector to Claude",
  "use_case": "PROVISION",
  "version": {
    "template": "1.0.0",
    "compatibility": [
      "2.12.0",
      "3.0.0"
    ]
  },
  "workflows": {
    "provision": {
      "nodes": [
        {
          "id": "create_claude_connector",
```

```
          "type": "create_connector",
          "user_inputs": {
            "name": "Claude Instant Runtime Connector",
            "version": "1",
            "protocol": "aws_sigv4",
            "description": "The connector to Bedrock service for Claude model",
            "actions": [
              {
                "headers": {
                  "x-amz-content-sha256": "required",
                  "content-type": "application/json"
                },
                "method": "POST",
                "request_body": "{ \"prompt\":\"${parameters.prompt}\",
 \"max_tokens_to_sample\":${parameters.max_tokens_to_sample},
 \"temperature\":${parameters.temperature},  \"anthropic_version\":
\"${parameters.anthropic_version}\" }",
                "action_type": "predict",
                "url": "https://bedrock-runtime.us-west-2.amazonaws.com/model/
anthropic.claude-instant-v1/invoke"
              }
            ],
            "credential": {
                "roleArn": "arn:aws:iam::account-id:role/opensearch-secretmanager-
role"
             },
            "parameters": {
              "endpoint": "bedrock-runtime.us-west-2.amazonaws.com",
              "content_type": "application/json",
              "auth": "Sig_V4",
              "max_tokens_to_sample": "8000",
              "service_name": "bedrock",
              "temperature": "0.0001",
              "response_filter": "$.completion",
              "region": "us-west-2",
              "anthropic_version": "bedrock-2023-05-31"
            }
          }
        }
      ]
    }
  }
}
```

ドメインが仮想プライベートクラウド (Amazon VPC) に存在する場合は、リクエストで AI コネクタを正常に作成するには、Amazon VPC に接続している必要があります。Amazon VPC へのアクセスはネットワーク構成によって異なりますが、通常は VPN あるいは社内ネットワークへの接続が必要になります。OpenSearch Service ドメインにアクセスできるかを確認するには、ウェブブラウザで `https://your-vpc-domain.region.es.amazonaws.com` を開き、デフォルトの JSON 応答を受信していることを確認します。(`############`を独自の値に置き換えます。

Python クライアントのサンプリング

Python クライアントは、HTTP リクエストよりも自動化が容易で、再利用性が向上します。Python クライアントで AI コネクタを作成するには、以下のサンプルコードを Python ファイルに保存します。このクライアントには、AWS SDK for Python (Boto3)、Requests:HTTP for Humans、および requests-aws4auth 1.2.3 パッケージが必要です。

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

host = 'domain-endpoint/'
region = 'region'
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

path = '_plugins/_flow_framework/workflow'
url = host + path

payload = {
  "name": "Deploy Claude Model",
  "description": "Deploy a model using a connector to Claude",
  "use_case": "PROVISION",
  "version": {
    "template": "1.0.0",
    "compatibility": [
      "2.12.0",
      "3.0.0"
    ]
  },
  "workflows": {
    "provision": {
      "nodes": [
```

```
        {
          "id": "create_claude_connector",
          "type": "create_connector",
          "user_inputs": {
            "name": "Claude Instant Runtime Connector",
            "version": "1",
            "protocol": "aws_sigv4",
            "description": "The connector to Bedrock service for Claude model",
            "actions": [
              {
                "headers": {
                  "x-amz-content-sha256": "required",
                  "content-type": "application/json"
                },
                "method": "POST",
                "request_body": "{ \"prompt\":\"${parameters.prompt}\",
 \"max_tokens_to_sample\":${parameters.max_tokens_to_sample},
 \"temperature\":${parameters.temperature},  \"anthropic_version\":
\"${parameters.anthropic_version}\" }",
                "action_type": "predict",
                "url": "https://bedrock-runtime.us-west-2.amazonaws.com/model/
anthropic.claude-instant-v1/invoke"
              }
            ],
            "credential": {
                "roleArn": "arn:aws:iam::account-id:role/opensearch-secretmanager-
role"
             },
            "parameters": {
              "endpoint": "bedrock-runtime.us-west-2.amazonaws.com",
              "content_type": "application/json",
              "auth": "Sig_V4",
              "max_tokens_to_sample": "8000",
              "service_name": "bedrock",
              "temperature": "0.0001",
              "response_filter": "$.completion",
              "region": "us-west-2",
              "anthropic_version": "bedrock-2023-05-31"
            }
          }
        }
      ]
    }
  }
```

```
}

headers = {"Content-Type": "application/json"}

r = requests.post(url, auth=awsauth, json=payload, headers=headers)
print(r.status_code)
print(r.text)
```

事前定義されたワークフローテンプレート

Amazon OpenSearch Service には、一般的な機械学習 (ML) のユースケース用に複数のワークフローテンプレートが用意されています。テンプレートを使用すると、複雑なセットアップが簡単になるほか、セマンティック検索や会話検索などのユースケースの多数のデフォルト値を活用できます。ワークフローテンプレートは、Create Workflow API を呼び出すときに指定できます。

- OpenSearch Service が提供するワークフローテンプレートを使用するには、use_case クエリパラメータとしてテンプレートのユースケースを指定します。
- カスタムワークフローテンプレートを使用するには、リクエスト本文に完全なテンプレートを指定します。カスタムテンプレートの例については、JSON テンプレートの例または YAML テンプレートの例を参照してください。

テンプレートのユースケース

この表は、使用可能なさまざまなテンプレートの概要、テンプレートの説明、および必要なパラメータを示しています。

テンプレートのユースケース	説明	必須パラメータ
bedrock_titan_embedding_model_deploy	Amazon Bedrock 埋め込みモデルを作成してデプロイする (デフォルトでは titan-embed-text-v1	create_connector.credential.roleArn
bedrock_titan_embe	Amazon Bedrock マルチモーダル埋め込みモデルを作成してデプロイする (デフォルトでは titan-embed-text-v1	create_connector.credential.roleArn

テンプレートのユースケース	説明	必須パラメータ
dding_model_deploy		
cohere_embedding_model_deploy	Cohere 埋め込みモデルを作成してデプロイする (デフォルトでは embed-english-v3.0)。	create_connector.credential.roleArn , create_connector.credential.secretArn
cohere_chat_model_deploy	Cohere チャットモデルを作成してデプロイする (デフォルトでは Cohere コマンド)。	create_connector.credential.roleArn , create_connector.credential.secretArn
open_ai_embedding_model_deploy	OpenAI 埋め込みモデルを作成してデプロイする (デフォルトでは text-embedding-ada-002)。	create_connector.credential.roleArn , create_connector.credential.secretArn
openai_chat_model_deploy	OpenAI チャットモデルを作成してデプロイする (デフォルトでは gpt-3.5-turbo)。	create_connector.credential.roleArn , create_connector.credential.secretArn
semantic_search_with_cohere_embedding	セマンティック検索を設定し、Cohere 埋め込みモデルをデプロイする。Cohere モデルの API キーを指定する必要があります。	create_connector.credential.roleArn , create_connector.credential.secretArn

テンプレートのユースケース	説明	必須パラメータ
semantic_search_with_cohere_embedding_query_enricher	セマンティック検索を設定し、Cohere 埋め込みモデルをデプロイする。ニューラルクエリのデフォルトのモデル ID を設定する query_enricher 検索プロセッサを追加する。Cohere モデルの API キーを指定する必要があります。	create_connector.credential.roleArn , create_connector.credential.secretArn
multimodal_search_with_bedrock_titan	Amazon Bedrock マルチモーダルモデルをデプロイし、text_image_embedding プロセッサとマルチモーダル検索用の k-NN インデックスを使用して取り込みパイプラインを設定する。 AWS 認証情報を入力する必要があります。	create_connector.credential.roleArn

> **Note**
>
> シークレット ARN を必要とするすべてのテンプレートの場合、デフォルトは「キー」のキー名を持つシークレットを AWS Secrets Manager に保存します。

事前トレーニング済みモデルを含むデフォルトのテンプレート

Amazon OpenSearch Service には、オープンソースの OpenSearch Service では利用できない 2 つの追加のデフォルトワークフローテンプレートが用意されています。

テンプレートのユースケース	説明
semantic_search_with_local_model	セマンティック検索を設定し、事前トレーニング済みモデル (msmarco-distilbert-base-tas-b) をデプロイします。ニューラルクエリのデフォルトのモデル ID を設定

テンプレートのユースケース	説明
	し、「my-nlp-index」というリンクされた k-NN インデックスを作成する neural_query_enricher 検索プロセッサを追加します。
hybrid_search_with_local_model	ハイブリッド検索を設定し、事前トレーニング済みモデル (msmarco-distilbert-base-tas-b) をデプロイします。ニューラルクエリのデフォルトのモデル ID を設定し、「my-nlp-index」というリンクされた k-NN インデックスを作成する neural_query_enricher 検索プロセッサを追加します。

アクセス許可を設定する

バージョン 2.13 以降で新しいドメインを作成する場合、アクセス許可は既に設定されています。バージョン 2.11 以前の既存の OpenSearch Service ドメインでフローフレームワークを有効にしてからバージョン 2.13 以降にアップグレードする場合は、flow_framework_manager ロールを定義する必要があります。きめ細かなアクセスコントロールを使用してドメインのウォームインデックスを管理するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。flow_framework_manager ロールを手動で作成するには、以下のステップを実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	許可
flow_framework_full_access	• cluster:admin/opensearch/flow_framework/* • cluster_monitor
flow_framework_read_accesss	• cluster:admin/opensearch/flow_framework/workflow/get

グループ名	許可
	• cluster:admin/opensearch/flow_framework/workflow/search • cluster:admin/opensearch/flow_framework/workflow_state/get • cluster:admin/opensearch/flow_framework/workflow_state/search

3. [ロール]、[ロールの作成] の順に選択します。
4. ロール [flow_framework_manager] に名前を付けます。
5. クラスターの許可 では、flow_framework_full_access および flow_framework_read_access を選択します。
6. [インデックス] では、* と入力します。
7. [インデックスの許可] では、indices:admin/aliases/get、indices:admin/mappings/get、および indices_monitor を選択します。
8. [作成] を選択します。
9. ロールを作成したら、任意のユーザーまたはフローフレームワークインデックスを管理するバックエンドロールにそれをマッピングします。

Amazon OpenSearch Service の AI エージェント統合

OpenSearch コミュニティには、AI エージェントが Amazon OpenSearch Service と直接連携できるようにする 2 つのオープンソースプロジェクトがあります。

- the section called “エージェントスキル” – 検索アプリケーションの構築、分散トレースの調査、クラスターのデプロイなど、一般的な OpenSearch ワークフローの実行方法を AI コーディングエージェントに教えるタスクに焦点を当てた命令セットのコレクション AWS。スキルは、Claude コード、カーソル、Kiro など、エージェントスキル標準をサポートするエージェントと連携し、実行中のサーバーは必要ありません。
- the section called “MCP サーバー” – OpenSearch APIs呼び出し可能なツールとして公開するモデルコンテキストプロトコル (MCP) サーバー。AI アシスタントとエージェントフレームワークがインデックスを検索し、クラスターの状態を検査し、ライブ OpenSearch Service ドメインと OpenSearch Serverless コレクションに対して診断を実行できるようにします。コーディング IDEs、Claude Desktop、Kiro CLI、および Strands Agents や LangGraph などのエージェントフレームワークと統合されます。

どちらの統合も、マネージド OpenSearch Service ドメインと OpenSearch Serverless コレクションと連携します。

OpenSearch エージェントスキル

今日の開発者は、Kiro、Claude Code、Cursor などのエージェント IDEs を使用して、アイデアから作業プロトタイプに数分で移行できます。ただし、新しいアイデアを試す場合でも、概念実証を構築する場合でも、本番稼働用システムを実行する場合でも、エクスペリエンスは急速に複雑になります。検索結果は期待どおりに動作せず、レイテンシーの急増ではログを掘り下げる必要があり、 にデプロイすると、深い専門知識を必要とする設定決定 AWS が導入されます。ハイレベルなインテントをクエリ DSLs、インデックス設定、マルチステップワークフローに変換するには、AI エージェント側であっても、依然としてかなりの時間がかかります。

OpenSearch エージェントスキルは、OpenSearch インテリジェンスをエージェントに直接取り込むスキルのリポジトリを提供することで、これに対処します。各スキルは、特定のワークフローのドメインの知識、ベストプラクティス、複数ステップの実行ロジックをカプセル化するため、エージェントは結果を取得するだけでなく、どのように達成されたかを理解します。スキルは、Kiro、Claude Code、カーソルなど、エージェントスキル標準をサポートするコーディングエージェントと連携します。

エージェントスキルでできること

ローカル検索アプリケーションを構築する

このopensearch-launchpadスキルは、検索アプリケーションの構築と反復にインテント主導のエクスペリエンスをもたらします。自然言語の要件やサンプルデータを、キーワード、セマンティック、ハイブリッド検索用に最適化されたマッピング、取り込みパイプライン、ML モデル統合を備えた完全に設定された OpenSearch インデックスに変換し、すぐにテストして反復できる作業検索エクスペリエンスを生成します。

プロンプトの例:

```
Build a semantic search application for product documentation
```

エージェントはインデックスマッピングを設定し、埋め込みモデルと取り込みパイプラインを設定し、作業中の検索 API と UI を返します。これにより、各決定を順を追って説明します。

ログ分析によるインシデントの調査

このlog-analyticsスキルは、エージェントインテリジェンスをオブザーバビリティワークフローにもたらします。パイプ処理言語 (PPL) クエリを手動で作成したり、サービス間でログデータを切り替えたりする代わりに、インテントを表現し、スキルはエラーパターンの検出から異常分析、根本原因の相関まで、複雑さを処理します。

プロンプトの例:

```
Investigate why my service is returning 500s and correlate with recent traces
```

エージェントは PPL クエリを実行してエラーパターンとログボリュームの異常を表示し、ログエラーを特定のトレース IDs に関連付けて根本原因分析を高速化します。PPL 構文をマスターしたり、トレースデータを手動でナビゲートしたりする必要はありません。

分散トレースの調査

このtrace-analyticsスキルは、分散トレースを OpenSearch 内で直接調査します。スロースパン、エラースパン、サービス依存関係を識別し、トレース IDs を使用してログとトレースを関連付けて、オブザーバビリティスタック全体の根本原因を表面化します。

プロンプトの例:

```
Which service is causing the p99 latency regression in the checkout flow?
```

エージェントはサービスマップを構築し、スロースパンを識別してボトルネックを特定します。サービス間でトレースデータを手動でナビゲートする必要はありません。

OpenSearch を にデプロイする AWS

このaws-setupスキルは、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションをプロビジョニングし、アクセスポリシーと暗号化ポリシーを設定し、アプリケーションをマネージドドメインとサーバーレスコレクション用の個別のガイドで接続する方法をエージェントに説明します。

プロンプトの例:

```
Set up an OpenSearch Serverless collection for my search application
```

エージェントはプロビジョニング、ポリシー設定、検証を処理し、各ステップを案内します。

使用可能なスキル

スキルはカテゴリ別に整理されています。完全なコレクションまたは個々のスキルをインストールできます。

Category	スキル	その内容
検索	opensearch-launchpad	検索アプリケーションを最初からスキャフォールドします。BM25 辞書検索、セマンティック検索、ハイブリッド検索、エージェント検索について説明します。モデル選択ガイド、評価戦略、サンプルデータが含まれます。
オブザーバビリティ	log-analytics	Piped Processing Language (PPL) を使用してログをクエリおよび分析します。エージェントがエラーパターンを検出し、異常を表面化し、インデックス間でイベントを関連付けるのに役立ちます。
オブザーバビリティ	trace-analytics	分散トレースを調査します。システムを通じて低速スパンを識別し、サービスマップを構築し、AI エージェントツールの呼び出しをトレースします。

Category	スキル	その内容
Cloud	`aws-setup`	OpenSearch をデプロイします AWS。プロビジョニング、アクセスポリシー、ネットワーク設定をカバーする、Amazon OpenSearch Service ドメインと OpenSearch Serverless コレクション専用のガイドが含まれています。

プロジェクトはアクティブに開発中です。最新のスキルリストとコントリビューションガイダンスについては、opensearch-agent-skills リポジトリを参照してください。

前提条件

前提条件はスキルによって異なります。フルコレクションを使用するには、以下が必要です。

- エージェントスキル (Claude Code、Cursor、または Kiro) をサポートするコーディングエージェント。
- Node.js を npxで使用してPATH、`skills`インストーラを実行します。
- uv および Python 3.11 以降。スキルは uv を使用して、分離された環境でスクリプトを実行します。
- ローカルクラスターをスピンアップするスキル (など) を使用する場合は、Docker をインストールして実行しますopensearch-launchpad。
- AWS `aws-setup` スキルを使用する予定がある場合は、適切なアクセス許可で設定された 認証情報。必要なアクセス許可のガイダンスについては、「」を参照してくださいthe section called “アイデンティティベースのポリシー”。

スキルのインストールと使用

スキルは`skills`、 で呼び出す CLI を介してインストールされますnpx。インストーラはエージェントを検出し、スキルファイルを正しい場所に配置します。

コレクション全体をインストールするには:

```
npx skills add opensearch-project/opensearch-agent-skills
```

単一のスキルをインストールするには、@skill-nameサフィックスを使用します。--full-depth フラグには、エージェントがフォローアップの質問に必要なすべての参考資料が含まれます。

```
npx skills add opensearch-project/opensearch-agent-skills@opensearch-launchpad --full-
depth
npx skills add opensearch-project/opensearch-agent-skills@log-analytics --full-depth
npx skills add opensearch-project/opensearch-agent-skills@trace-analytics --full-depth
npx skills add opensearch-project/opensearch-agent-skills@aws-setup --full-depth
```

一般的なインストールオプション:

-a *agent-name*

などの特定のエージェントに をインストールします-a claude-code。

-g

スキルがマシン上のすべてのプロジェクトで使用できるように、グローバルに をインストールします。

--all

CLI がマシンで検出するすべてのエージェントに をインストールします。

インストール後、エージェントを再起動して新しいスキルファイルを取得します。次に、OpenSearch." エージェントはスキル指示を読み取り、必要なスクリプトを直接実行します。

aws-setup スキルには、両方の OpenSearch Service 製品ファミリー専用のガイドが含まれています。Amazon OpenSearch Service マネージドドメインの場合、スキルはドメインの作成、アクセスポリシーの設定、およびアプリケーションの接続について説明します (「」を参照Introduction)。OpenSearch Serverless では、コレクションの作成、データアクセスポリシーと暗号化ポリシーの設定、データの取り込みについて説明します (「」を参照Amazon OpenSearch Serverless)。実行中のクラスターに対するインタラクティブなクエリの場合は、スキルを とペアリングしますthe section called “MCP サーバー”。

セキュリティに関する考慮事項

スキルをインストールして実行すると、エージェントが記述したスクリプトとコマンドを実行します。サードパーティーの開発者ツールと同様にスキルを扱います。

- インストールする前に、ソースリポジトリのスキルSKILL.mdファイルを確認します。

- 開発環境またはサンドボックス環境でスキルを実行してから、本番稼働用リソースをポイントします。
- `aws-setup` スキルを実行するときは、最小特権の IAM 認証情報を使用します。これは、 AWS アカウント および AWS リージョン デプロイ先の に限定されます。
- 存続期間の長いインフラストラクチャにコミットする前に、スキルが作成する AWS リソースを確認します。スキルは、ユーザーに代わってドメイン、コレクション、IAM ロール、ネットワークリソースをプロビジョニングする場合があります。

その他のリソース

- GitHub の opensearch-agent-skills – 新しいスキルを作成するためのソース、問題、開発者ガイド。
- エージェントスキルの仕様 – スキルが実装するオープンスタンダード。
- the section called “MCP サーバー” – 実行中のクラスターに対するインタラクティブクエリ用のコンパニオン MCP サーバー。

OpenSearch MCP サーバー

OpenSearch MCP サーバー (`opensearch-mcp-server-py`) は、OpenSearch の Model Context Protocol のオープンソース実装です。OpenSearch API APIs「クラスター内にどのようなインデックスが存在するか」などの質問をすることができます。または「過去 1 時間の最も遅いクエリを表示する」とエージェントがユーザーに代わって API コールを処理します。

サーバーは、マネージド OpenSearch Service ドメインと OpenSearch Serverless コレクションの両方で動作します。コーディング IDEs (Kiro、Claude Code、Cursor)、デスクトップ AI アシスタント (Claude Desktop)、エージェントフレームワーク (Strands Agents、LangGraph) に接続できます。

前提条件

- Python 3.10 以降。
- uv (推奨) または `pip`。
- 到達可能な OpenSearch エンドポイント: OpenSearch Service ドメイン、OpenSearch Serverless コレクション、またはセルフマネージド OpenSearch クラスター。
- 公開する OpenSearch APIs を呼び出すアクセス許可を持つ認証情報。OpenSearch Service ドメインの IAM ポリシーについては、「」を参照してくださいthe section called “アイデンティティ

ベースのポリシー”。OpenSearch Serverless データアクセスポリシーについては、「」を参照してくださいthe section called “データアクセスコントロール”。

インストール

でサーバーを実行するuvxか (インストール不要) 、ローカルにインストールします。

```
# Run directly with uvx (recommended - no install step)
uvx opensearch-mcp-server-py

# Or install with pip
pip install opensearch-mcp-server-py
```

ほとんどのユーザーは、サーバーを自動的に起動するように IDE または AI アシスタントを設定します。例the section called “エージェントフレームワーク”については、the section called “IDE で を設定する”「」と「」を参照してください

認証

環境変数 (単一クラスターモード) または YAML 設定ファイル (マルチクラスターモード) のクラスターごとに認証を設定します。サーバーは認証方法を、認証なし、ヘッダーベース、IAM ロール、基本認証、 AWS アクセスキーの優先順位で適用します。

IAM ロール (SigV4) – OpenSearch Service ドメインに推奨

```
export OPENSEARCH_URL="https://your-domain-endpoint"
export AWS_IAM_ARN="arn:aws:iam::123456789012:role/YourOpenSearchRole"
export AWS_REGION="us-east-1"
```

ロールには、サーバーが使用する OpenSearch APIs を呼び出すアクセス許可が必要です。サンプルポリシーthe section called “追加のサンプルポリシー”については、「」を参照してください。

AWS 認証情報またはプロファイル

```
export OPENSEARCH_URL="https://your-domain-endpoint"
export AWS_REGION="us-east-1"
export AWS_PROFILE="your-aws-profile"
```

OpenSearch Serverless コレクション

サーバーが ではなくaossサービス名でリクエストに署名AWS_OPENSEARCH_SERVERLESS=trueするように を設定しますes。プリンシパルに、コレクションのデータアクセスポリシーを通じてアクセス権が付与されていることを確認します (「」を参照the section called “データアクセスコントロール”)。

```
export OPENSEARCH_URL="https://collection-id.us-east-1.aoss.amazonaws.com"
export AWS_OPENSEARCH_SERVERLESS="true"
export AWS_REGION="us-east-1"
export AWS_PROFILE="your-aws-profile"
```

基本認証

```
export OPENSEARCH_URL="https://your-domain-endpoint"
export OPENSEARCH_USERNAME="username"
export OPENSEARCH_PASSWORD="password"
```

> **Important**
>
> ソース管理にチェックインできる設定ファイルにパスワードを埋め込まないでください。オペレーティングシステムのシークレットマネージャーまたはシェル環境を使用して、認証情報を提供します。

コーディング IDE で を設定する

MCP をサポートするコーディング IDEs は、JSON 設定ファイルを使用して、起動するサーバーを宣言します。サーバーは stdio 経由で実行されるため、プロジェクトを開くと IDE によってサブプロセスとして起動されます。

Kiro

プロジェクトディレクトリ (ワークスペースレベル) または ~/.kiro/settings/mcp.json (ユーザーレベル、すべてのプロジェクトで使用可能) .kiro/settings/mcp.jsonで を作成または編集します。Kiro はローカルマシンで実行されるため、 AWS_PROFILE を使用して既存の~/.aws/credentialsファイルから認証情報を取得します。

Amazon OpenSearch Service ドメイン:

```
{
  "mcpServers": {
    "opensearch": {
      "command": "uvx",
      "args": ["opensearch-mcp-server-py"],
      "env": {
        "OPENSEARCH_URL": "https://your-domain-endpoint.region.es.amazonaws.com",
        "AWS_REGION":     "us-east-1",
        "AWS_PROFILE":    "your-aws-profile"
      }
    }
  }
}
```

OpenSearch Serverless コレクション:

```
{
  "mcpServers": {
    "opensearch": {
      "command": "uvx",
      "args": ["opensearch-mcp-server-py"],
      "env": {
        "OPENSEARCH_URL":            "https://collection-id.region.aoss.amazonaws.com",
        "AWS_OPENSEARCH_SERVERLESS": "true",
        "AWS_REGION":                "us-east-1",
        "AWS_PROFILE":               "your-aws-profile"
      }
    }
  }
}
```

保存後、Kiro は自動的にサーバーを再接続します。その後、「クラスター内のインデックスを一覧表示する」や「クラスターの状態は？」などの質問をすることができます。

Claude Code

サーバーをプロジェクトの `.mcp.json` ファイルに追加します。Kiro と同様に、 `AWS_PROFILE` を使用してローカル AWS 認証情報で認証します。

Amazon OpenSearch Service ドメイン:

```
{
```

```
  "mcpServers": {
    "opensearch": {
      "command": "uvx",
      "args": ["opensearch-mcp-server-py"],
      "env": {
        "OPENSEARCH_URL": "https://your-domain-endpoint.region.es.amazonaws.com",
        "AWS_REGION":     "us-east-1",
        "AWS_PROFILE":    "your-aws-profile"
      }
    }
  }
}
```

OpenSearch Serverless コレクション:

```
{
  "mcpServers": {
    "opensearch": {
      "command": "uvx",
      "args": ["opensearch-mcp-server-py"],
      "env": {
        "OPENSEARCH_URL":            "https://collection-id.region.aoss.amazonaws.com",
        "AWS_OPENSEARCH_SERVERLESS": "true",
        "AWS_REGION":                "us-east-1",
        "AWS_PROFILE":               "your-aws-profile"
      }
    }
  }
}
```

サーバーを CLI 経由でグローバルに追加して、すべてのプロジェクトで使用できるようにすることもできます。

```
claude mcp add opensearch \
  --command uvx \
  --args opensearch-mcp-server-py \
  --env OPENSEARCH_URL=https://your-domain-endpoint.region.es.amazonaws.com \
  --env AWS_REGION=us-east-1 \
  --env AWS_PROFILE=your-aws-profile
```

Claude デスクトップ

設定 > デベロッパー を開き、 を編集しますclaude_desktop_config.json。

```
{
  "mcpServers": {
    "opensearch": {
      "command": "uvx",
      "args": ["opensearch-mcp-server-py"],
      "env": {
        "OPENSEARCH_URL": "https://your-domain-endpoint",
        "AWS_REGION": "us-east-1",
        "AWS_PROFILE": "your-aws-profile"
      }
    }
  }
}
```

Kiro CLI

Kiro CLI を使用する場合は、上記のものと同じ JSON 構造.kiro/settings/mcp.jsonを使用して、プロジェクトの にサーバーを追加します。Kiro CLI は、そのディレクトリでセッションを開始すると、設定を自動的に取得します。

エージェントフレームワークで を使用する

OpenSearch MCP サーバーを Python エージェントフレームワークに直接統合できるため、自律型エージェントはより大きなワークフローの一部として OpenSearch にプログラムでアクセスできます。

Strands Agents

Strands Agents は、組み込み MCP サポートとデフォルトのモデルプロバイダーとしての Amazon Bedrock を備えた AWSネイティブエージェント SDK です。次の例では、適切な環境変数を設定して、Strands エージェントを OpenSearch Service ドメインまたは OpenSearch Serverless コレクションに接続します。OpenSearch Serverless コレクションに接続するtrue場合は AWS_OPENSEARCH_SERVERLESS に設定し、マネージドドメインの場合は省略します。

```
import os
from strands import Agent
```

```
from strands.tools.mcp import MCPClient
from mcp import stdio_client, StdioServerParameters

# For a managed domain:
#   OPENSEARCH_URL = https://<domain-endpoint>.<region>.es.amazonaws.com
#
# For an OpenSearch Serverless collection, also set:
#   AWS_OPENSEARCH_SERVERLESS = true
#   OPENSEARCH_URL = https://<collection-id>.<region>.aoss.amazonaws.com
opensearch_client = MCPClient(
    lambda: stdio_client(
        StdioServerParameters(
            command="uvx",
            args=["opensearch-mcp-server-py"],
            env={
                "OPENSEARCH_URL":            os.environ["OPENSEARCH_URL"],
                "AWS_REGION":                os.environ["AWS_REGION"],
                "AWS_IAM_ARN":               os.environ["AWS_IAM_ARN"],
                # Set to "true" for OpenSearch Serverless, omit for managed domains
                "AWS_OPENSEARCH_SERVERLESS":
 os.environ.get("AWS_OPENSEARCH_SERVERLESS", "false"),
            },
        )
    )
)

with opensearch_client:
    agent = Agent(tools=opensearch_client.list_tools_sync())
    response = agent("List all indexes and show the document count for each")
    print(response)
```

ストランドは、デフォルトのモデルプロバイダーとして Amazon Bedrock を使用します。リージョンで Claude に対して AWS 認証情報が設定され、モデルアクセスが有効になっていることを確認します。詳細については、Strands Bedrock プロバイダーのドキュメントを参照してください。

LangGraph

LangGraph は、ステートフルエージェントを構築するための低レベルのオーケストレーションフレームワークです。次の例では`langchain-mcp-adapters`、 を使用して、Amazon Bedrock がサポートする LangGraph ReAct エージェントに OpenSearch MCP ツールをロードします。Strands と同様に、OpenSearch Serverless コレクションに接続する`true`ときは`AWS_OPENSEARCH_SERVERLESS`に設定します。

```
import asyncio
import os
from langchain_aws import ChatBedrock
from langchain_mcp_adapters.client import MultiServerMCPClient
from langgraph.prebuilt import create_react_agent

async def main():
    async with MultiServerMCPClient(
        {
            "opensearch": {
                "command": "uvx",
                "args": ["opensearch-mcp-server-py"],
                "env": {
                    # Managed domain:  https://<domain-
endpoint>.<region>.es.amazonaws.com
                    # Serverless:      https://<collection-
id>.<region>.aoss.amazonaws.com
                    "OPENSEARCH_URL":            os.environ["OPENSEARCH_URL"],
                    "AWS_REGION":                os.environ["AWS_REGION"],
                    "AWS_IAM_ARN":               os.environ["AWS_IAM_ARN"],
                    # Set to "true" for OpenSearch Serverless, omit for managed domains
                    "AWS_OPENSEARCH_SERVERLESS":
 os.environ.get("AWS_OPENSEARCH_SERVERLESS", "false"),
                },
                "transport": "stdio",
            }
        }
    ) as mcp_client:
        tools = mcp_client.get_tools()
        model = ChatBedrock(
            model_id="anthropic.claude-3-5-sonnet-20241022-v2:0",
            region_name=os.environ["AWS_REGION"],
        )
        agent = create_react_agent(model, tools)
        result = await agent.ainvoke(
            {"messages": [{"role": "user", "content": "Check cluster health and list
 all indexes"}]}
        )
        print(result["messages"][-1].content)

asyncio.run(main())
```

必要なパッケージをインストールします。

```
pip install langchain-aws langchain-mcp-adapters langgraph
```

セキュリティに関する考慮事項

MCP サーバーは、指定した認証情報で実行されます。これらの認証情報が許可されていると、AI アシスタントがユーザーに代わってトリガーする可能性があります。以下のプラクティスに従います。

- 最小特権の認証情報を使用します。エージェントが必要とするインデックスとアクションをスコープとする専用の IAM ロールまたは OpenSearch ユーザーを作成します。管理者認証情報の再利用は避けてください。
- 開発と本番を分離します。探索する非本番稼働用クラスターにサーバーを向けます。本番稼働用アクセスが必要な場合は、明示的なクラスター名でマルチモードを使用します。
- フィルターツール。ワークフローで不要なツールカテゴリを無効にします。デフォルト設定を無効にする`OPENSEARCH_DISABLED_CATEGORIES=core_tools`ように を設定するか、`OPENSEARCH_ENABLED_CATEGORIES` を使用して特定のカテゴリのみを有効にします。
- 認証情報を保護します。静的アクセスキーよりもプロファイルと IAM ロールを優先 AWS します。ソースコントロールの設定ファイルにシークレットをコミットしないでください。
- ツールの出力を確認します。MCP ツールのレスポンスは、コンテキストとして言語モデルにフィードバックされます。AI プロバイダーに公開したくない機密データを含むインデックスに対してサーバーを実行しないでください。

トラブルシューティング

アシスタントに OpenSearch ツールが表示されない

設定ファイルが有効な JSON であることを確認し、クライアントを完全に再起動します。ほとんどのクライアントは、起動時にのみ MCP サーバーをロードします。

ツール呼び出しは 403 Forbidden を返します

認証情報に、ツールが呼び出す API に対するアクセス許可がありません。OpenSearch Service ドメインの場合は、ロールにアタッチされているドメインアクセスポリシーと IAM ポリシーを確認します。OpenSearch Serverless の場合は、コレクションのデータアクセスポリシーを確認します。

OpenSearch Serverless の署名不一致エラー

`AWS_OPENSEARCH_SERVERLESS=true` が設定されていることを確認します (またはマルチモード`is_serverless: true`)。これがないと、サーバーは ではなく`es`サービス名で署名します`aoss`。

必要なツールが利用できない

デフォルト以外のカテゴリにあるかどうかを確認し、YAML 設定`enabled_categories`で `OPENSEARCH_ENABLED_CATEGORIES`または で有効にします。

詳細については、opensearch-mcp-server-py リポジトリで問題を開きます。

その他のリソース

- GitHub の opensearch-mcp-server-py – ソース、問題、リリースノート。
- ユーザーガイド – ストリーミングトランスポートと Kubernetes デプロイを含む完全な設定リファレンス。
- Strands Agents – 組み込み MCP サポートを備えた AWSネイティブエージェント SDK。
- LangGraph – ステートフルエージェント向けの低レベルのオーケストレーションフレームワーク。
- the section called “エージェントスキル” – タスクに焦点を当てたワークフローのコンパニオンスキルコレクション。

Amazon OpenSearch Service のセキュリティ分析

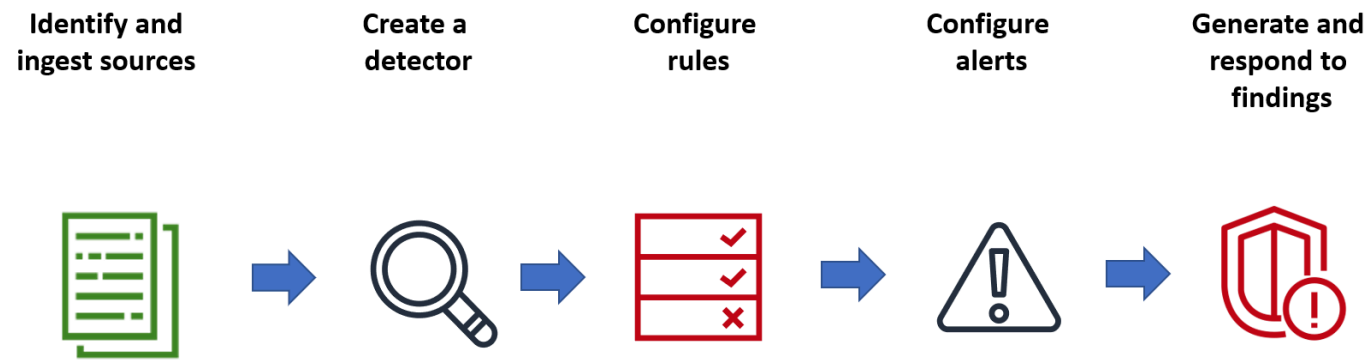
セキュリティ分析は OpenSearch のソリューションの 1 つで、組織のインフラストラクチャの可視化や異常なアクティビティのモニタリング、潜在的なセキュリティ脅威のリアルタイム検出、事前に設定された宛先へのアラート発行などを行います。セキュリティルールを継続的に評価し、自動生成されたセキュリティ検出結果を確認することにより、セキュリティイベントのログの、悪意のあるアクティビティをモニタリングすることができます。さらにセキュリティ分析では、自動アラートを作成し、これを Slack やメールなどの指定された通知チャネルに送信することができます。

セキュリティ分析のプラグインを使用すれば、一般的な脅威をすぐに検出し、ファイアウォールログや Windows ログ、認証監査ログといった既存のセキュリティイベントログから、重要なセキュリティインサイトを生成することができます。セキュリティ分析を使用するには、ドメインが OpenSearch バージョン 2.5 以降を実行している必要があります。

> **Note**
>
> このドキュメントでは、Amazon OpenSearch Service のセキュリティ分析の概要を説明します。主要な概念を定義し、アクセス許可を設定する手順を提供します。設定ガイド、API リファレンス、使用可能なすべての設定のリファレンスを含む包括的なドキュメントについては、OpenSearch ドキュメントの「Security Analytics」を参照してください。

セキュリティ分析のコンポーネントと概念

数多くのツールや機能が、セキュリティ分析のオペレーションの基盤を構成しています。セキュリティ分析のプラグインを構成している主な要素には、ディテクター、ログタイプ、ルール、検出結果、アラートなどがあります。



ログタイプ

OpenSearch は複数のログのタイプをサポートしており、各タイプですぐにマッピングを行います。ディテクターを作成するときにログタイプを指定して時間間隔を設定すると、そこから、セキュリティ分析は関連するルールセットを自動的にアクティブ化し、指定された間隔で実行します。

ディテクター

ディテクターは、ログタイプのさまざまなサイバーセキュリティ脅威を、データインデックス全体で特定します。ユーザーは、カスタムルールと、システムで発生するイベントを評価するパッケージ済みの Sigma ルールの、両方を使用するようにディテクターを設定します。すると、ディテクターがこれらのイベントからセキュリティ検出結果を生成します。ディテクターの詳細については、OpenSearch ドキュメントの「Creating detectors」(ディテクターの作成) を参照してください。

Rules

脅威検出ルールは、ディテクターが、取り込まれたログデータに適用してセキュリティイベントを特定する際の条件を定義します。セキュリティ分析では、ユーザーの要件に合うように、ルールをインポート、作成、カスタマイズすることができます。また、ログから一般的な脅威を検出するための、パッケージ済みの、オープンソースの Sigma ルールも用意されています。セキュリティ分析は、MITRE ATT&CK 組織が管理している、敵対相手の戦術やテクニックに関する、増加を続けるナレッジベースに、数多くのルールをマッピングします。ルールを作成し使用するには、OpenSearch Dashboards と API の両方を使用できます。ルールの詳細については、OpenSearch ドキュメントの「Working with rules」(ルールの使用方法) を参照してください。

検出結果

ディテクターによって、ルールとログイベントが一致すると、検出結果が生成されます。各検出結果には、選択したルール、ログタイプ、ルールの重要度から成る、一意の組み合わせが含まれています。検出結果に示される脅威は、必ずしも、今すぐ対処しなければならないものではありませんが、常に重要なイベントが取り出されています。検出結果の詳細については、OpenSearch ドキュメントの「Working with findings」(検出結果の使用方法) を参照してください。

アラート

ディテクターを作成するときは、アラートをトリガーする条件を 1 つ以上指定することができます。アラートは、Slack やメールなど、優先チャンネルに送信される通知です。ディテクターが 1 つまたは複数のルールに一致したときにアラートがトリガーされるように設定します。通知メッセージ

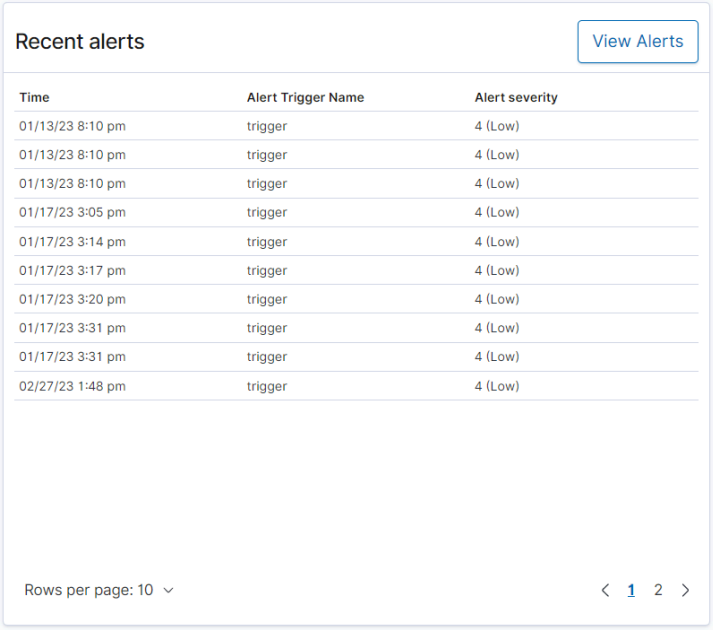
はカスタマイズできます。アラートの詳細については、OpenSearch ドキュメントの「Working with alerts」(アラートの使用方法) を参照してください。

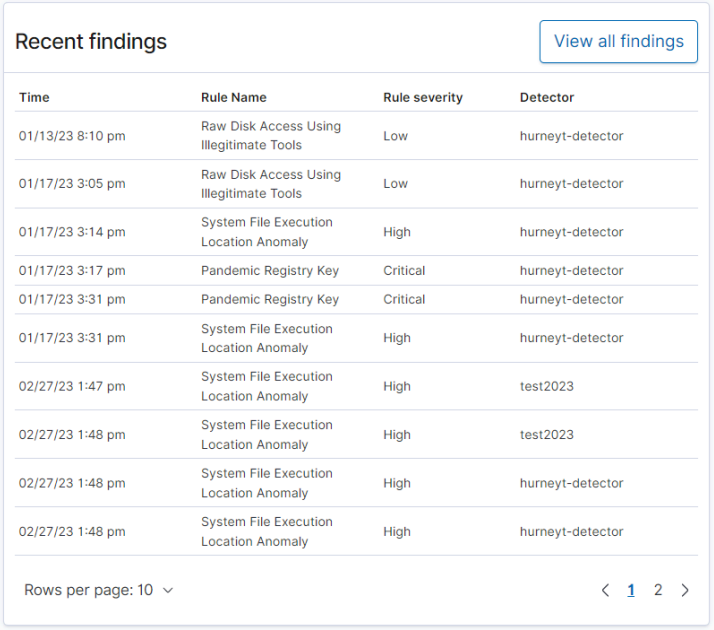
セキュリティ分析を理解する

OpenSearch Dashboards を使用すれば、Security Analytics のプラグインを視覚化してインサイトを得ることができます。[概要] 画面では、検出結果とアラート数、最近の検出結果とアラート、頻出の検出ルール、ディテクターの一覧といった情報を確認できます。複数の画面を、それらをまとめた 1 か所の画面で確認できます。例えば、次のグラフには、特定の期間におけるさまざまなログタイプの検出結果とアラートの傾向が示されています。

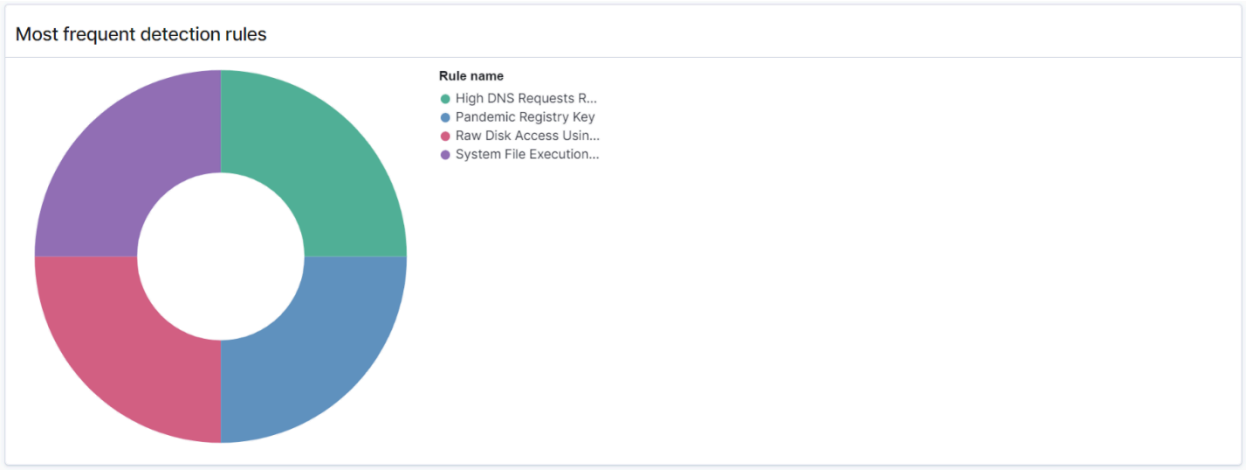


ページの下へ進むと、最新の検出結果とアラートを確認できます。





さらに、最も多くトリガーされたルールが、アクティブなディテクター全体に広がっていることを確認できます。ログタイプ全体でさまざまな種類の悪意あるアクティビティを検出し、調査するのに役立ちます。



最後に、設定済みのディテクターのステータスを確認できます。このパネルからは、ディテクターのワークフローの作成に移動することもできます。



Security Analytics の設定を行うには、[Rules] (ルール) ページでルールを作成し、このルールを使って [Detectors] (ディテクター) ページにディテクターを記述します。Security Analytics の検出結果に特化した画面を表示するには、[Findings] (検出結果) と [Alerts] (アラート) のページを使用します。

アクセス許可の設定

既存の OpenSearch Service ドメインで Security Analytics を有効にした場合、そのドメインで `security_analytics_manager` ロールが定義されていない可能性があります。きめ細かなアクセスコントロールを使用してドメインのウォームインデックスを管理するには、管理者以外のユーザーがこのロールにマッピングされている必要があります。`security_analytics_manager` ロールを手動で作成するには、以下のステップを実行します。

1. OpenSearch Dashboards で、[セキュリティ] に進み、[許可] を選択します。
2. [アクショングループの作成] を選択し、以下のグループを設定します。

グループ名	許可
`security_analytics_full_access`	• `cluster:admin/opensearch/securityanalytics/alerts/*` • `cluster:admin/opensearch/securityanalytics/detector/*` • `cluster:admin/opensearch/securityanalytics/findings/*` • `cluster:admin/opensearch/securityanalytics/mapping/*` • `cluster:admin/opensearch/securityanalytics/rule/*`

グループ名	許可
security_analytics_read_access	• cluster:admin/opensearch/securityanalytics/alerts/get • cluster:admin/opensearch/securityanalytics/detector/get • cluster:admin/opensearch/securityanalytics/detector/search • cluster:admin/opensearch/securityanalytics/findings/get • cluster:admin/opensearch/securityanalytics/mapping/get • cluster:admin/opensearch/securityanalytics/mapping/view/get • cluster:admin/opensearch/securityanalytics/rule/get • cluster:admin/opensearch/securityanalytics/rule/search

3. [ロール]、[ロールの作成] の順に選択します。
4. ロールに security_analytics_manager という名前を付けます。
5. クラスターの許可 では、security_analytics_full_access および security_analytics_read_access を選択します。
6. [インデックス] では、* と入力します。
7. [Index permissions] (インデックスのアクセス許可) で、indices:admin/mapping/put と indices:admin/mappings/get を選択します。
8. [作成] を選択します。
9. ロールを作成したら、任意のユーザーまたは Security Analytics インデックスを管理するバックエンドロールに、それをマッピングします。

トラブルシューティング

そのようなインデックスエラーはありません

ディテクターがない状態で Security Analytics ダッシュボードを開くと、右下に `[index_not_found_exception] no such index [.opensearch-sap-detectors-config]` と書かれた通知が表示されることがあります。この通知は無視してかまいません。ディテクターを作成すると数秒で消え、再び表示されることはありません。

Amazon OpenSearch Service の運用上のベストプラクティス

この章では、Amazon OpenSearch Service ドメインを運用する際のベストプラクティスと、多くのユースケースに適用される一般的なガイドラインを提供します。各ワークロードは一意で、固有の特性があるため、すべてのユースケースに完全に適した一般的な推奨事項はありません。最も重要なベストプラクティスは、ドメインを連続的なサイクルでデプロイ、テスト、調整して、ワークロードに最適な設定、安定性、およびコストを見つけることです。

トピック

- モニタリングとアラート
- シャード戦略
- 安定性
- パフォーマンス
- セキュリティ
- コスト最適化
- Amazon OpenSearch Service で推奨される CloudWatch アラーム
- Amazon OpenSearch Service ドメインのサイジング
- Amazon OpenSearch Service 用ペタバイトスケール
- Amazon OpenSearch Service の専用コーディネーターノード
- Amazon OpenSearch Service の専用マスターノード

モニタリングとアラート

次のベストプラクティスは、OpenSearch Service ドメインのモニタリングに適用されます。

CloudWatch アラームを設定する

OpenSearch Service は Amazon CloudWatch にパフォーマンスメトリクスを発行します クラスターとインスタンスのメトリクスを定期的に確認し、ワークロードのパフォーマンスに基づいて推奨される CloudWatch アラームを設定します。

ログの発行を有効にする

OpenSearch Service は、OpenSearch エラーログ、検索スローログ、インデックス作成スローログ、および監査ログを Amazon CloudWatch Logs に発行します。検索スローログ、インデックス作成スローログ、およびエラーログは、パフォーマンスと安定性の問題のトラブルシューティングに役立ちます。きめ細かいアクセスコントロールを有効にした場合にのみ使用できる監査ログは、ユーザーアクティビティを追跡します。詳細については、OpenSearch ドキュメントの「ログ」を参照してください。

検索スローログとインデックス作成スローログは、検索およびインデックス作成オペレーションのパフォーマンスを理解し、トラブルシューティングするための重要なツールです。すべての本番ドメインのために、検索およびインデックスの低速ログ配信を有効にします。ログ記録のしきい値も設定する必要があります。設定しない場合、CloudWatch はログをキャプチャしません。

シャード戦略

シャードは、OpenSearch Service ドメイン内のデータノード全体でワークロードを分散します。インデックスを適切に設定すると、ドメイン全体のパフォーマンスを向上させるのに役立ちます。

OpenSearch Service にデータを送信すると、そのデータをインデックスに送信することになります。インデックスはデータベーステーブルに似ています。ドキュメントが行、フィールドが列に相当します。インデックスを作成する際に、作成するプライマリシャードの数を OpenSearch に指示します。プライマリシャードは、完全なデータセットの独立したパーティションです。OpenSearch Service は、インデックスのプライマリシャード全体でデータを自動的に分散します。インデックスのレプリカを設定することもできます。各レプリカシャードは、そのインデックスのプライマリシャードのコピーの完全なセットで構成されます。

OpenSearch Service は、クラスター内のデータノード全体で各インデックスのシャードをマッピングします。これにより、インデックスのプライマリシャードとレプリカシャードが別々のデータノードに確実に存在するようになります。最初のレプリカは、インデックスにデータのコピーが確実に 2 個あるようにします。常に少なくとも 1 個のレプリカを使用する必要があります。追加のレプリカは、さらなる冗長性と読み取りキャパシティーを提供します。

OpenSearch は、インデックス作成のリクエストを、インデックスに属するシャードを含むすべてのデータノードに送信します。インデックス作成リクエストは、まずプライマリシャードを含むデータノードに送信され、その後にレプリカシャードを含むデータノードに送信されます。検索リクエストは、コーディネーターノードによって、インデックスに属するすべてのシャードのプライマリシャードまたはレプリカシャードにルーティングされます。

たとえば、プライマリシャードごとに 5 つのプライマリシャードと 1 つのレプリカを持つインデックスの場合、各インデックス作成リクエストは 10 個のシャードに触れます。一方、検索リクエストは n 個のシャードに送信されます。n はプライマリシャードの数です。5 個のプライマリシャードと 1 個のレプリカがあるインデックスの場合、各検索クエリは、そのインデックスの 5 個のシャード (プライマリまたはレプリカ) にタッチします。

シャードとデータノード数を決定する

次のベストプラクティスを使用して、ドメインのシャードとデータノード数を決定します。

シャードのサイズ — ディスク上のデータのサイズは、ソースデータのサイズの直接的な結果であり、インデックスを作成するデータが増えるにつれて変化します。ソースとインデックスの比率は 1:10 から 10:1 またはそれ以上に大きく変動する可能性がありますが、通常は 1:1.10 前後です。この比率を使用して、ディスク上のインデックスサイズを予測できます。また、一部のデータにインデックスを付け、実際のインデックスサイズを取得して、ワークロードの比率を判断することもできます。インデックスのサイズを予測したら、各シャードが 10 ~ 30 GiB (検索ワークロードの場合) または 30 ~ 50 GiB (ログワークロードの場合) になるようにシャード数を設定します。50 GiB が最大になるはずです。増量に備えて準備しておいてください。

シャード数 — データノードへのシャードの分散は、ドメインのパフォーマンスに大きな影響を与えます。複数のシャードを持つインデックスがある場合は、できるかぎりシャード数がデータノード数の倍数になるようにしてください。これは、シャードがデータノード全体に均等に分散されるようにするのに役立ち、ノードがホットになるのを防ぎます。例えば、プライマリシャードが 12 個ある場合、データノード数は 2、3、4、6、または 12 になります。ただし、シャード数よりもシャードサイズの方が重要です。5 GiB のデータがある場合、1 個のシャードを使用するべきです。

データノードあたりのシャード — ノードが保持できるシャードの総数は、ノードの Java 仮想マシン (JVM) ヒープメモリに比例します。ヒープメモリの GiB あたりのシャード数が 25 個以下になるように試みます。例えば、32 GiB のヒープメモリを持つノードのシャード数は 800 個以下である必要があります。シャード配分はワークロードパターンによって異なりますが、Elasticsearch および OpenSearch 1.1 ~ 2.15 では 1 ノードあたり 1,000 シャード、OpenSearch 2.17 以降では同 4,000 シャードに制限されています。cat/allocation API は、シャードの数とデータノード全体のシャードストレージの合計のクイックビューを提供します。

シャードと CPU の比率 — シャードがインデックス作成または検索リクエストに関わっている場合、vCPU を使用してリクエストを処理します。ベストプラクティスとして、シャードあたり 1.5 vCPU の初期スケールポイントを使用します。インスタンスタイプに 8 個の vCPUs がある場合は、各ノードのシャード数が 6 個以下になるようにデータノード数を設定します。これは近似値である

ことに注意してください。必ずワークロードをテストし、それに応じてクラスターをスケールしてください。

ストレージボリューム、シャードサイズ、インスタンスタイプに関する推奨事項については、次のリソースを参照してください。

- the section called “ドメインのサイジング”
- the section called “ペタバイトスケール”

ストレージスキューを回避する

ストレージスキューは、クラスター内の 1 つ以上のノードが、1 つ以上のインデックスについて、他のノードよりも高い割合のストレージを保持している場合に発生します。不均等な CPU 使用率、断続的で不均等なレイテンシー、データノード全体での不均等なキューイングなどはストレージスキューを示します。スキューの問題があるかどうかを判断するには、次のトラブルシューティングセクションを参照してください。

- the section called “ノードシャードとストレージスキュー”
- the section called “インデックスシャードとストレージスキュー”

安定性

次のベストプラクティスは、安定した正常な OpenSearch Service ドメインを維持するために適用されます。

OpenSearch で最新の状態に保つ

サービスソフトウェア更新

OpenSearch Service では、機能追加やドメイン強化を行うソフトウェア更新を定期的にリリースしています。更新では、OpenSearch または Elasticsearch のエンジンバージョンは変更されません。DescribeDomain API オペレーションを定期的に実行するようにスケジュールを設定し、`UpdateStatus` が `ELIGIBLE` の場合にサービスソフトウェアの更新を開始することをお勧めします。特定の期間 (通常は 2 週間) 内にドメインを更新しない場合、OpenSearch Service は自動的に更新を実行します。

OpenSearch のバージョンアップグレード

OpenSearch Service は、コミュニティが維持するバージョンの OpenSearch のサポートを定期的に追加します。最新の OpenSearch バージョンが利用可能になったら、必ずアップグレードしてください。

OpenSearch Service は、OpenSearch と OpenSearch Dashboards (ドメインが従来のエンジンを実行している場合は Elasticsearch と Kibana) の両方を同時にアップグレードします。クラスターに専用マスターノードがある場合、アップグレードはダウンタイムなしで完了します。ない場合、クラスターがマスターノードを選択している間、アップグレード後数秒間応答しなくなることがあります。OpenSearch Dashboards は、アップグレードの一部または全期間で使用できなくなる場合があります。

ドメインをアップグレードするには 2 つの方法があります。

- インプレースアップグレード — 同じクラスターを維持するため、このオプションの方が簡単です。
- スナップショット/リストアアップグレード — このオプションは、新しいクラスターで新しいバージョンをテストしたり、クラスター間で移行したりするのに適しています。

使用するアップグレードプロセスにかかわらず、開発とテスト専用のドメインを維持し、本番ドメインをアップグレードする前に新しいバージョンにアップグレードすることをお勧めします。テストドメインを作成するときに、デプロイタイプのために [Development and testing] (開発とテスト) を選択します。ドメインのアップグレードの直後に、必ずすべてのクライアントを互換性のあるバージョンにアップグレードしてください。

スナップショットパフォーマンスの向上

スナップショットの処理が停止するのを防ぐには、専用マスターノードのインスタンスタイプがシャードカウントに一致する必要があります。詳細については、「the section called “専用マスターノードのインスタンスタイプの選択”」を参照してください。また、各ノードの Java ヒープメモリの GiB あたりのシャード数は、推奨される 25 シャードよりも少なくしてください。詳細については、「the section called “シャード数の選択”」を参照してください。

専用マスターノードを有効にする

専用のマスターノードにより、クラスターの安定性が向上します。専用マスターノードではクラスター管理タスクを実行しますが、インデックスデータは保持せず、クライアントリクエストにも応答しません。このクラスター管理タスクのオフロードにより、ドメインの安定性が向上し、一部の設定変更をダウンタイムなしで行うことができます。

3 つのアベイラビリティーゾーンにまたがるドメインの安定性を最適化するために、3 個の専用マスターノードを有効にして使用します。スタンバイが有効のマルチ AZ でデプロイすると、3 つの専用マスターノードが設定されます。インスタンスタイプの推奨事項については、「the section called "専用マスターノードのインスタンスタイプの選択"」を参照してください。

複数のアベイラビリティゾーンにデプロイする

サービス中断が発生した場合にデータの損失を防ぎ、クラスターのダウンタイムを最小限に抑えるため、同じ AWS リージョン内の 2 つまたは 3 つのアベイラビリティーゾーン間にノードを分散できます。ベストプラクティスは、スタンバイが有効のマルチ AZ を使用してデプロイすることです。これにより、3 つのアベイラビリティーゾーンが設定され、2 つのゾーンがアクティブ、1 つのゾーンがスタンバイとして機能し、インデックスごとに 2 つのレプリカシャードが割り当てられます。この設定では、OpenSearch Service が対応するプライマリシャードとは異なる AZ にレプリカシャードを分散できます。アベイラビリティーゾーン間のクラスター通信には、AZ 間のデータ転送料金はかかりません。

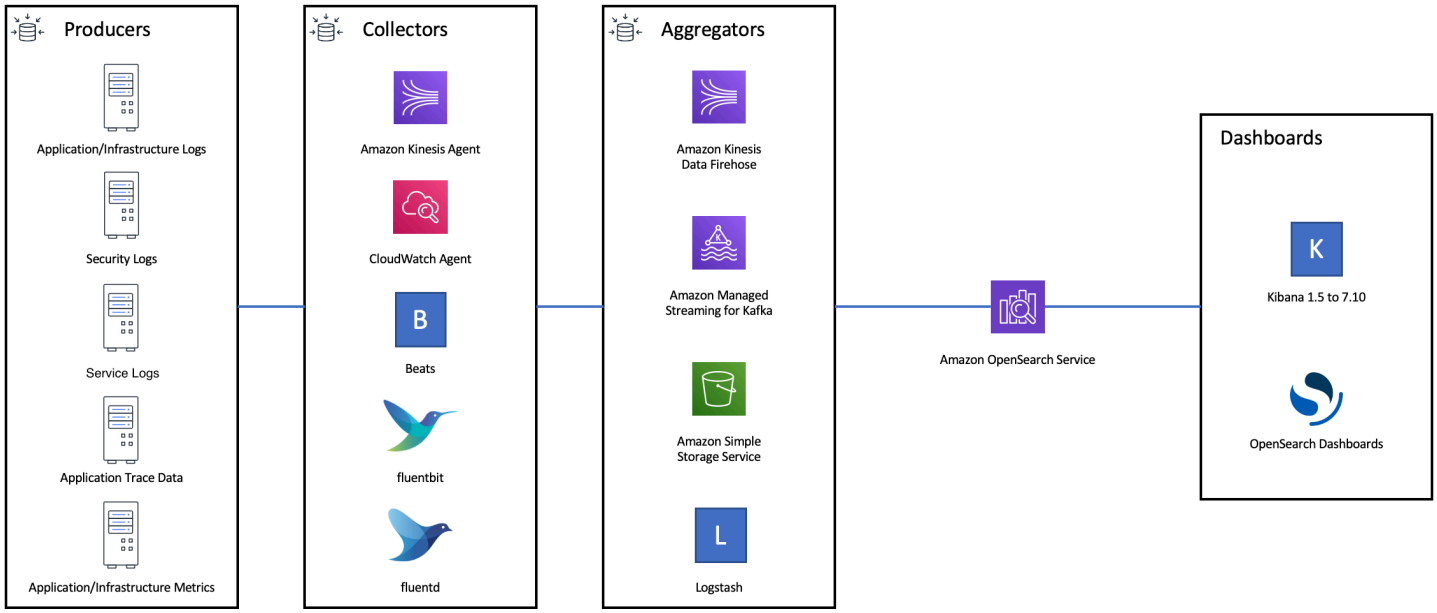
アベイラビリティーゾーンは、各 リージョン内の独立した場所です。2 つの AZ 設定では、1 つのアベイラビリティーゾーンを失うことは、すべてのドメインキャパシティーの半分を失うことを意味します。3 つのアベイラビリティゾーンに移行すると、1 つのアベイラビリティゾーンを失うことによる影響が軽減されます。

取り込みフローとバッファリングを制御する

_bulk API オペレーションを使用して、全体のリクエスト数を制限することをお勧めします。1 つのドキュメントを含む 5,000 のリクエストを送信するよりも、5,000 のドキュメントを含む 1 つの `_bulk` リクエストを送信する方が効率的です。

最適な運用安定性のために、インデックス作成リクエストのアップストリームフローを制限したり、一時停止したりする必要がある場合があります。インデックス作成リクエストのレートを制限することは、対処しなければクラスターで過負荷となる可能性のある、予期しないまたは時折のリクエストの急増に対処するための重要なメカニズムです。アップストリームアーキテクチャにフロー制御メカニズムを構築することを検討してください。

次の図は、ログ取り込みアーキテクチャの複数のコンポーネントオプションを示しています。急激なトラフィックの急増や短時間のドメインメンテナンスのために受信データをバッファリングするのに十分なスペースを確保できるように、集約レイヤーを設定します。



検索ワークロードのマッピングを作成する

検索ワークロードでは、OpenSearch がドキュメントとそのフィールドを格納し、それらのインデックスを作成する方法を定義するマッピングを作成します。新しいフィールドが誤って追加されるのを防ぐために dynamic を strict に設定します。

```
PUT my-index
{
  "mappings": {
    "dynamic": "strict",
    "properties": {
      "title": { "type" : "text" },
      "author": { "type" : "integer" },
      "year": { "type" : "text" }
    }
  }
}
```

インデックステンプレートを使用する

インデックスの作成時にインデックスを設定する方法を OpenSearch に伝えるために、インデックステンプレートを使用できます。インデックスを作成する前に、インデックステンプレートを設定します。その後、インデックスを作成すると、テンプレートから設定とマッピングが継承されます。1 個のインデックスに複数のテンプレートを適用できるため、1 個のテンプレートで設定を指定し、別のテンプレートでマッピングを指定できます。この戦略では、複数のインデックスにまたがる共通設

定用の 1 個のテンプレートと、より具体的な設定やマッピング用の個別のテンプレートを使用できます。

次の設定は、テンプレートでの設定に役立ちます。

- プライマリシャードとレプリカシャードの数
- 更新間隔 (インデックスを更新して最近の変更を検索できるようにする頻度)
- 動的マッピングコントロール
- 明示的なフィールドマッピング

次のテンプレート例には、これらの各設定が含まれています。

```
{
   "index_patterns":[
      "index-*"
   ],
   "order": 0,
   "settings": {
      "index": {
         "number_of_shards": 3,
         "number_of_replicas": 1,
         "refresh_interval": "60s"
      }
   },
   "mappings": {
      "dynamic": false,
      "properties": {
         "field_name1": {
            "type": "keyword"
         }
      }
   }
}
```

ほとんど変更されない場合でも、設定とマッピングを OpenSearch で一元的に定義することは、複数のアップストリームクライアントを更新するよりも管理が簡単です。

Index State Management でインデックスを管理する

ログや時系列データを管理している場合は、Index State Management (ISM) を使用することをお勧めします。ISM では、通常のインデックスライフサイクル管理タスクを自動化できます。ISM を使

用すると、インデックスエイリアスのロールオーバーを呼び出すポリシーを作成し、インデックスのスナップショットを作成し、ストレージ階層間でインデックスを移動し、古いインデックスを削除できます。シャードスキューを回避するための代替データライフサイクル管理戦略として、ISM ロールオーバーオペレーションを使用することもできます。

まず、ISM ポリシーを設定します。例については、「the section called “サンプルポリシー”」を参照してください。その後、ポリシーを 1 つ以上のインデックスにアタッチします。ポリシーに ISM テンプレートフィールドを含めると、OpenSearch Service は、指定されたパターンに一致するすべてのインデックスにポリシーを自動的に適用します。

未使用インデックスの削除

クラスター内のインデックスを定期的に確認し、使用されていないインデックスを特定します。これらのインデックスが S3 に保存されるようにスナップショットを作成してから、削除します。未使用のインデックスを削除すると、シャード数が減り、ノード全体でよりバランスの取れたストレージの分散とリソースの使用が可能になります。アイドル状態であっても、インデックスは内部的なインデックスのメンテナンス作業中に一部のリソースを消費します。

未使用のインデックスを手動で削除する代わりに、ISM を使用して自動的にスナップショットを作成し、一定期間後にインデックスを削除することができます。

高可用性を実現するために複数のドメインを使用する

複数のリージョンで 99.9% のアップタイムを超える高可用性を実現するには、2 つのドメインの使用を検討してください。小規模またはゆっくりと変化するデータセットの場合、クロスクラスターレプリケーションを設定して、アクティブ/パッシブモデルを維持できます。このモデルでは、リーダードメインのみが書き込みの対象となりますが、どちらのドメインも読み取りの対象とすることができます。データセットが大きく、データが急速に変化する場合は、すべてのデータがアクティブ-アクティブモデルの両方のドメインに独立して書き込まれるように、取り込みパイプラインで二重配信を設定します。

フェイルオーバーを念頭に置いて、アップストリームとダウンストリームのアプリケーションを設計します。フェイルオーバープロセスは、他のディザスタリカバリプロセスと一緒にテストしてください。

パフォーマンス

次のベストプラクティスは、最適なパフォーマンスを実現するために、ドメインの調整に適用されます。

一括リクエストのサイズと圧縮を最適化する

一括サイズ設定は、データ、分析、およびクラスター設定によって異なりますが、適切な開始点は一括リクエストあたり 3～5 MiB です。

gzip 圧縮を使用して OpenSearch ドメインからリクエストを送信し、レスポンスを受信して、リクエストとレスポンスのペイロードサイズを削減します。OpenSearch Python クライアントで、またはクライアント側から次のヘッダーを含めることによって、gzip 圧縮を使用できます。

- `'Accept-Encoding': 'gzip'`
- `'Content-Encoding': 'gzip'`

一括リクエストのサイズを最適化するには、3 MiB の一括リクエストサイズから始めます。その後、インデックス作成のパフォーマンスが改善しなくなるまで、リクエストサイズを徐々に大きくします。

> **Note**
>
> Elasticsearch バージョン 6.x を実行しているドメインで gzip 圧縮を有効にするには、クラスターレベルで `http_compression.enabled` を設定する必要があります。この設定は、Elasticsearch バージョン 7.x および OpenSearch のすべてのバージョンではデフォルトで true です。

一括リクエストのレスポンスのサイズを小さくする

OpenSearch レスポンスのサイズを小さくするには、`filter_path` パラメータを使用して不要なフィールドを除外します。失敗したリクエストを識別または再試行するために必要なフィールドを除外しないようにしてください。詳細な説明と例については、the section called “レスポンスサイズの削減” を参照してください。

更新間隔を調整する

OpenSearch インデックスには、最終的な読み取り整合性があります。更新オペレーションにより、インデックスに対して実行されたすべての更新が検索可能になります。デフォルトの更新間隔は 1 秒です。つまり、OpenSearch は、インデックスが書き込まれている間、更新を毎秒実行します。

インデックスの更新頻度が低い (更新間隔が長い) ほど、インデックス作成の全体的なパフォーマンスが向上します。更新間隔を長くすることのトレードオフは、インデックスが更新されてから新し

いデータが検索可能になるまでの時間が長くなることです。全体的なパフォーマンスを向上させるには、更新間隔を許容できる限り長く設定します。

すべてのインデックスの `refresh_interval` パラメータを 30 秒以上に設定することをお勧めします。

Auto-Tune を有効にする

Auto-Tune は、OpenSearch クラスターのパフォーマンスと使用状況のメトリクスを使用して、ノードのキューサイズ、キャッシュサイズ、および Java 仮想マシン (JVM) 設定の変更を提案します。これらのオプションの変更により、クラスターの速度と安定性が向上します。いつでもデフォルトの OpenSearch Service 設定に戻すことができます。Auto-Tune は、明示的に無効にしない限り、新しいドメインでデフォルトで有効になります。

すべてのドメインで Auto-Tune を有効にし、定期的なメンテナンスウィンドウを設定するか、定期的に推奨事項を確認することをお勧めします。

セキュリティ

ドメインのセキュリティ保護には、次のベストプラクティスが適用されます。

きめ細かなアクセスコントロールを有効にする

きめ細かなアクセスコントロールにより、OpenSearch Service ドメイン内の特定のデータにアクセスできるユーザーを制御できます。一般化されたアクセスコントロールと比較して、きめ細かいアクセスコントロールでは、各クラスター、インデックス、ドキュメント、およびフィールドに、独自の指定アクセスポリシーが設定されます。アクセス基準は、アクセスをリクエストする人の役割や、データに対して実行しようとしているアクションなど、さまざまな要因に基づくことができます。例えば、あるユーザーにはインデックスへの書き込みアクセスを許可し、別のユーザーには変更を加えずにインデックスのデータを読み取るためだけのアクセスを許可する場合があります。

きめ細かなアクセスコントロールは、セキュリティやコンプライアンスの問題を生じさせずに、アクセス要件の異なるデータが同じストレージスペースに存在できるようにします。

ドメインできめ細かいアクセスコントロールを有効にすることをお勧めします。

VPC 内にドメインをデプロイする

OpenSearch Service ドメインを仮想プライベートクラウド (VPC) に配置することは、インターネットゲートウェイ、NAT デバイス、VPN 接続なしで、VPC 内で OpenSearch Service と他のサービス

間の安全な通信を実現するのに役立ちます。すべてのトラフィックは AWS クラウド内に安全に保持されます。論理的な隔離により、VPC 内に存在するドメインには、パブリックエンドポイントを使用するドメインに比較して、より拡張されたセキュリティレイヤーがあります。

VPC 内にドメインを作成することをお勧めします。

制限的なアクセスポリシーを適用する

ドメインが VPC 内にデプロイされている場合でも、セキュリティをレイヤーで実装するのがベストプラクティスです。現在のアクセスポリシーの設定を確認してください。

制限的なリソースベースのアクセスポリシーをドメインに適用し、設定 API と OpenSearch API オペレーションへのアクセス権を付与する際は最小特権の原則に従ってください。原則として、アクセスポリシーで匿名ユーザープリンシパル `"Principal": {"AWS": "*" }` を使用することは避けてください。

ただし、きめ細かなアクセスコントロールを有効にする場合など、オープンアクセスポリシーの使用が許容される状況もあります。オープンアクセスポリシーを使用すると、特定のクライアントやツールなど、リクエストの署名が困難または不可能な場合にもドメインにアクセスできます。

保管中の暗号化を有効にする

OpenSearch Service ドメインでは、データへの不正アクセスの防止に役立つ保管中のデータの暗号化が提供されます。保管時の暗号化では、暗号化キーの保存と管理に AWS Key Management Service (AWS KMS) を使用し、暗号化を実行するために 256 ビットキー (AES-256) を使用する Advanced Encryption Standard アルゴリズムを使用します。

ドメインに機密データが保存されている場合、保管中のデータの暗号化を有効にします。

ノード間の暗号化を有効にする

ノード間の暗号化では、OpenSearch Service 内でデフォルトのセキュリティ機能の上に追加のセキュリティレイヤーを提供します。OpenSearch 内でプロビジョニングされたノード間のすべての通信のために Transport Layer Security (TLS) を実装します。ノード間の暗号化により、HTTPS 経由で OpenSearch Service ドメインに送信されたデータは、ノード間で配布およびレプリケートされる間、転送中も暗号化されたままになります。

ドメインに機密データが保存されている場合は、ノード間の暗号化を有効にします。

によるモニタリング AWS Security Hub CSPM

AWS Security Hub CSPM を使用して、セキュリティのベストプラクティスに関連する OpenSearch Service の使用状況をモニタリングします。Security Hub CSPM によってセキュリティコントロールが使用されてリソース設定およびセキュリティ標準が評価され、さまざまなコンプライアンスフレームワークに準拠できるようにサポートします。Security Hub CSPM を使用して OpenSearch Service リソースを評価する方法の詳細については、AWS Security Hub 「 ユーザーガイド」の「 Amazon OpenSearch Service コントロール」を参照してください。

コスト最適化

次のベストプラクティスは、OpenSearch Service のコストを最適化および節約するために適用されます。

最新世代のインスタンスタイプを使用する

OpenSearch Service は、より低いコストでより優れたパフォーマンスを提供する新しい Amazon EC2 インスタンスタイプを常に採用しています。常に最新世代のインスタンスを使用することをお勧めします。

本番稼働用ドメインで T2 や `t3.small` インスタンスを使用しないようにしてください。それらは負荷の高い状態では不安定になることがあるためです。`r6g.large` インスタンスは、小規模な本番ワークロードのためのオプションです (データノードと専用マスターノードの両方として)。

最新の Amazon EBS gp3 ボリュームを使用する

OpenSearch データノードは、迅速なインデックス作成とクエリを提供するために、低レイテンシーかつ高スループットのストレージを必要とします。Amazon EBS gp3 ボリュームを使用することによって、以前提供されていた Amazon EBS gp2 ボリュームタイプよりも 9.6% 低いコストで、より優れたベースラインパフォーマンス (IOPS およびスループット) を得ることができます。gp3 を使用すると、ボリュームサイズにかかわらず、追加の IOPS とスループットをプロビジョニングできます。これらのボリュームはバーストクレジットを使用しないため、前世代のボリュームよりも安定性が優れています。gp3 ボリュームタイプでは、データノードあたりのボリュームサイズの上限も、gp2 ボリュームタイプの 2 倍に拡大されます。これらのより大きなボリュームを使用すると、データノードあたりのストレージ容量を増やすことによって、パッシブデータのコストを削減できます。

時系列のログデータのために UltraWarm とコールドストレージを使用する

ログ分析のために OpenSearch を使用している場合は、データを UltraWarm またはコールドストレージに移動してコストを削減します。Index State Management (ISM) を使用して、ストレージ階層間でデータを移行し、データ保持を管理します。

UltraWarm は、大量の読み取り専用データをコストパフォーマンスに優れた方法で OpenSearch Service に保存できます。UltraWarm はストレージのために Amazon S3 を使用します。つまり、データはイミュータブルであり、必要なコピーは 1 つだけです。インデックス内のプライマリシャードのサイズに相当するストレージの料金のみをお支払いいただきます。UltraWarm クエリのレイテンシーは、クエリの処理に必要な S3 データの量が増えると高くなります。データがノードにキャッシュされると、UltraWarm インデックスに対するクエリは、ホットインデックスに対するクエリと同様に実行されます。

コールドストレージも S3 を使用します。コールドデータをクエリする必要がある場合は、既存の UltraWarm ノードに選択的にアタッチできます。コールドデータは UltraWarm と同じマネージドストレージコストを発生させますが、コールドストレージ内のオブジェクトは UltraWarm ノードリソースを消費しません。そのため、コールドストレージは UltraWarm ノードのサイズや数に影響を及ぼすことなく、大量のストレージキャパシティを提供します。

ホットストレージから移行するデータが約 2.5 TiB になると、UltraWarm は費用対効果が高くなります。フィルレートをモニタリングし、そのデータ量に達する前にインデックスを UltraWarm に移動するように計画します。

リザーブドインスタンスの推奨事項を確認する

パフォーマンスとコンピューティング消費量の適切なベースラインを把握したら、リザーブドインスタンス (RI) の購入を検討してください。割引は、前払いなしの 1 年間の予約で約 30% から始まり、全額前払いの 3 年間の契約では最大 50% まで増加する可能性があります。

少なくとも 14 日間にわたって安定した稼働状態を確認できたら、「AWS Cost Management ユーザーガイド」の「Accessing reservation recommendations」を参照してください。Amazon OpenSearch Service の見出しには、特定の RI 購入の推奨事項と予測される節約額が表示されます。

Amazon OpenSearch Service で推奨される CloudWatch アラーム

CloudWatch アラームは、CloudWatch メトリクスがある程度の時間にわたって指定された値を超えたときにアクションを実行します。たとえば、クラスター AWS のヘルスステータスが `red`1 分以上

の場合は、E メールで送信できます。このセクションでは、Amazon OpenSearch Service で推奨されるいくつかのアラームとそのアラームへの対応方法について説明します。

を使用して、これらのアラームを自動的にデプロイできます CloudFormation。サンプルスタックについては、関連する GitHub リポジトリを参照してください。

> **Note**
>
> CloudFormation スタックをデプロイすると、`KMSKeyError` アラームおよび `KMSKeyInaccessible` アラームは `Insufficient Data` 状態になっています。これは、これらのメトリクスがドメインの暗号化キーに問題が発生した場合にのみ表示されるからです。

アラームの設定の詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch でのアラームの使用」を参照してください。

アラーム	問題
`ClusterStatus.red` maximum is >= 1 for 1 minute, 1 consecutive time	少なくとも 1 つのプライマリシャードとそのレプリカがノードに割り当てられていません。「the section called “赤のクラスター状態”」を参照してください。
`ClusterStatus.yellow` maximum is >= 1 for 1 minute, 5 consecutive times	少なくとも 1 つのレプリカシャードがノードに割り当てられていません。「the section called “黄色のクラスター状態”」を参照してください。
`FreeStorageSpace` minimum is <= 20480 for 1 minute, 1 consecutive time	クラスターのノードの空きストレージ容量が 20 GiB に下がっています。「the section called “使用可能なストレージ領域の不足”」を参照してください。この値は MiB 単位です。20480 ではなく、各ノードのストレージ容量の 25% に設定することをお勧めします。
`ClusterIndexWritesBlocked` is >= 1 for	クラスターは書き込みリクエストをブロックしています。「the section called “ClusterBlockException”」を参照してください。

アラーム	問題
5 minutes, 1 consecutive time	
Nodes minimum is < x for 1 day, 1 consecutive time	x はクラスター内のノード数です。このアラームは、1 日以内にクラスター内の少なくとも 1 つのノードに到達できなかったことを示します。「the section called “障害が発生したクラスターノード”」を参照してください。

Automated SnapshotFailure maximum is >= 1 for 1 minute, 1 consecutive time	自動スナップショットが失敗しました。多くの場合、この失敗によってクラスター状態が赤になります。「the section called “赤のクラスター状態”」を参照してください。
	
	すべての自動スナップショットの概要および障害に関する情報を取得するには、次のリクエストのいずれかを試してください。
	
	`GET domain_endpoint /_snapshot/cs-automated/_all`
	`GET domain_endpoint /_snapshot/cs-automated-enc/_all`

CPUUtilization または WarmCPUUtilization maximum is >= 80% for 15 minutes, 3 consecutive times	100% の CPU 使用率が一時的に発生することもありますが、高い使用率が継続的に発生するのは問題です。より大規模なインスタンスタイプを使用するか、インスタンスを追加することを検討してください。
JVMMemory Pressure maximum is >= 95% for 1 minute, 3 consecutive times	使用量が増加した場合にクラスターでメモリ不足エラーが発生する可能性があります。垂直スケーリングを検討してください。OpenSearch Service は、インスタンスの RAM の半分を Java ヒープ (32 GiB のヒープサイズまで) に使用します。インスタンスは最大 64 GiB の RAM まで垂直スケーリングでき、それ以上はインスタンスを追加することで水平方向にスケーリングできます。
OldGenJVM MemoryPressure maximum is >= 80% for 1 minute, 3 consecutive times	

アラーム	問題
MasterCPU Utilization maximum is >= 50% for 15 minutes, 3 consecutive times	より大規模なインスタンスタイプを専用マスターノードとして使用することを検討してください。クラスターの安定性と Blue/Green デプロイに関わるため、専用マスターノードの CPU 使用率はデータノードよりも低くする必要があります。
MasterJVM MemoryPressure maximum is >= 95% for 1 minute, 3 consecutive times	
MasterOldGenJVMMemoryPressure maximum is >= 80% for 1 minute, 3 consecutive times	
KMSKeyError is >= 1 for 1 minute, 1 consecutive time	ドメイン内の保管中のデータの暗号化に使用される AWS KMS 暗号化キーは無効になっています。通常のオペレーションを復元するために、再度有効にしてください。詳細については、「the section called “保管中の暗号化”」を参照してください。
KMSKeyInaccessible is >= 1 for 1 minute, 1 consecutive time	ドメイン内の保管中のデータの暗号化に使用される AWS KMS 暗号化キーが削除されたか、OpenSearch Service への許可が取り消されました。この状態にあるドメインを復元することはできません。ただし、手動スナップショットがある場合は、それを使用して新しいドメインに移行できます。詳細についてはthe section called “保管中の暗号化”を参照してください。
shards.active is >= 30000 for 1 minute, 1 consecutive time	アクティブなプライマリとレプリカの両方のシャードの合計数は、30,000よりも多くなります。インデックスを頻繁にローテーションさせ過ぎている可能性があります。特定の年齢に達したら、ISM を使用してインデックスを削除することを検討してください。

アラーム	問題
5*xx* alarms >= 10% of OpenSearchRequests	1 つ以上のデータノードが過負荷になっているか、アイドルタイムアウト時間内にリクエストが完了しない可能性があります。より大きなインスタンスタイプに切り替えるか、クラスターにさらにノードを追加することを検討してください。シャードおよびクラスターアーキテクチャ用のベストプラクティスをフォローしていることを確認してください。
MasterReachableFromNode maximum is < 1 for 5 minutes, 1 consecutive time	このアラームは、マスターノードが停止しているか、連絡不能であることを示します。これらの障害は通常、ネットワーク接続の問題または AWS 依存関係の問題の結果です。
ThreadpoolWriteQueue average is >= 100 for 1 minute, 1 consecutive time	クラスタ-では、インデックス作成の同時実行性が高くなっています。インデックス作成リクエストを点検して抑制するか、クラスターリソースを増やします。
ThreadpoolSearchQueue average is >= 500 for 1 minute, 1 consecutive time	クラスターでは、検索の同時実行性が高くなっています。クラスターのスケーリングを検討してください。検索キューのサイズを大きくすることもできますが、そうすると、メモリ不足エラーが発生する可能性があります。
ThreadpoolSearchQueue maximum is >= 5000 for 1 minute, 1 consecutive time	

アラーム	問題
Increase in `ThreadpoolSearchRejected` SUM is >=1{ math expression DIFF( )} for 1 minute, 1 consecutive time	これらのアラームは、パフォーマンスや安定性に影響を及ぼす可能性のあるドメインの問題を通知します。
Increase in `ThreadpoolWriteRejected` SUM is >=1{ math expression DIFF( )} for 1 minute, 1 consecutive time	

> **Note**
>
> メトリクスを表示するのみであれば、「the section called “クラスターメトリクスのモニタリング”」を参照してください。

検討した方が良いその他のアラーム

定期的に使用する OpenSearch Service の特徴に応じて、次のアラームを設定することを検討してください。

アラーム	問題
`WarmFreeStorageSpace` is >= 10%	無料ウォームストレージの合計の 10% に達しました。`WarmFreeStorageSpace` は、無料ウォームストレージスペースの合計を測定します (MiB)。UltraWarm は、アタッチされたディスクではなく Amazon S3 を使用します。

アラーム	問題
HotToWarmMigrationQueueSize is >= 20 for 1 minute, 3 consecutive times	多数のインデックスが、ホットストレージから UltraWarm ストレージへ同時に移動しています。クラスターのスケーリングを検討してください。
HotToWarmMigrationSuccessLatency is >= 1 day, 1 consecutive time	デイリーインデックスを動かそうとして HotToWarmMigrationSuccessCount x レイテンシーが 24 時間を超える場合に通知されるよう、このアラームを設定してください。
WarmJVMMemoryPressure maximum is >= 95% for 1 minute, 3 consecutive times	使用量が増加した場合にクラスターでメモリ不足エラーが発生する可能性があります。垂直スケーリングを検討してください。OpenSearch Service は、インスタンスの RAM の半分を Java ヒープ (32 GiB のヒープサイズまで) に使用します。インスタンスは最大 64 GiB の RAM まで垂直スケーリングでき、それ以上はインスタンスを追加することで水平方向にスケーリングできます。
WarmOldGenJVMMemoryPressure maximum is >= 80% for 1 minute, 3 consecutive times	
WarmToColdMigrationQueueSize is >= 20 for 1 minute, 3 consecutive times	多数のインデックスが、UltraWarm ストレージからコールドストレージへ同時に移動しています。クラスターのスケーリングを検討してください。

アラーム	問題
HotToWarmMigrationFailureCount is >= 1 for 1 minute, 1 consecutive time	スナップショット、シャード再配置、または強制マージ中に、移行が失敗する可能性があります。スナップショットまたはシャード再配置中の障害は、通常、ノードの障害または S3 接続の問題が原因です。通常、ディスク領域の不足は、強制マージ失敗の根本的な原因です。
WarmToColdMigrationFailureCount is >= 1 for 1 minute, 1 consecutive time	インデックスメタデータをコールドストレージに移動させようとして失敗すると、通常、移行は失敗します。ウォームインデックスクラスター状態が削除されたときにも、障害が発生する可能性があります。
WarmToColdMigrationLatency is >= 1 day, 1 consecutive time	デイリーインデックスを動かそうとして WarmToColdMigrationSuccessCount x レイテンシーが 24 時間を超える場合に通知されるよう、このアラームを設定してください。
AlertingDegraded is >= 1 for 1 minute, 1 consecutive time	アラートインデックスが赤色であるか、1 つ以上のノードがスケジュールどおりでないことを意味します。
ADPluginUnhealthy is >= 1 for 1 minute, 1 consecutive time	異常検出プラグインが正しく動作していません。これは、障害率が高いか、使用されているインデックスの 1 つが赤色であるためです。
AsynchronousSearchFailureRate is >= 1 for 1 minute, 1 consecutive time	少なくとも 1 つの非同期検索が直前に失敗しました。これは、おそらくコーディネータノードが失敗したことを意味します。非同期検索リクエストのライフサイクルは、コーディネータノードでのみ管理されているので、コーディネータがダウンすると、リクエストは失敗します。

アラーム	問題
`AsynchronousSearchStoreHealth` is >= 1 for 1 minute, 1 consecutive time	残存するインデックス内の非同期検索レスポンスストアの状態は、赤色です。大量の非同期レスポンスを保存している可能性があり、クラスターが不安定になる可能性があります。非同期検索レスポンスを 10 MB 以下に制限してください。
`SQLUnhealthy` is >= 1 for 1 minute, 3 consecutive times	SQL プラグインが 5 xxレスポンスコードを返しているか、または無効なクエリ DSL を OpenSearch に渡しています。クライアントがプラグインに対して行っているリクエストのトラブルシューティングを行います。
`LTRStatus.red` is >= 1 for 1 minute, 1 consecutive time	Learning to Rankプラグインの実行に必要なインデックスの内、少なくとも 1 つにプライマリシャードがなく、機能しません。

Amazon OpenSearch Service ドメインのサイジング

Amazon OpenSearch Service ドメインのサイズを決定するための完璧な方法はありません。ただし、ストレージのニーズ、サービス、および OpenSearch 自体を理解することから始めることで、ハードウェアのニーズに関する、情報に基づいた初期予測を行うことができます。ドメインのサイジングに取りかかる際の最も重要な検討材料として、そうした予測結果を代表的なワークロードを使用したテストに活用し、パフォーマンスを監視します。

トピック

- ストレージ要件の計算
- シャード数の選択
- インスタンスタイプとテストの選択

ストレージ要件の計算

ほとんどの OpenSearch ワークロードは、次の 2 つのカテゴリに大別できます。

- 長期存続インデックス: データを 1 つまたは複数の OpenSearch インデックスとして処理し、ソースデータの変更に伴ってそれらのインデックスを定期的に更新するコードを記述した場合。一般的な例としては、ウェブサイト、ドキュメント、e コマースの検索などがあります。
- ローリングインデックス: データは継続的に一時的なインデックスに受け渡され、インデックス作成時間や保持期間が指定されます (例: 2 週間保持される一連の日次のインデックス)。代表的な例は、ログ分析、時系列処理、クリックストリーム分析などです。

長期存続インデックスのワークロードの場合、使用されるストレージ容量はディスク上のソースデータを調べることで簡単に判断できます。データのソースが複数ある場合は、すべてのソースを単純に合計します。

ローリングインデックスについては、一般的な期間内に生成されるデータ量に保持期間を掛けます。たとえば、1 時間あたり 200 MiB のログデータが生成されるとすると、1 日で 4.7 GiB になります。この場合、保持期間が 2 週間であれば、いずれかの時点でデータは 66 GiB に達します。

ただし、ソースデータのサイズはストレージ要件の 1 つの要素に過ぎません。次の点についても考慮する必要があります。

- レプリカの数: 各レプリカはプライマリシャードの完全なコピーであり、インデックスのストアサイズはプライマリシャードとレプリカシャードによって取られたサイズを示します。デフォルトでは、各 OpenSearch インデックスに対してレプリカが 1 つ作成されます。データ損失を防ぐため、少なくとも 1 つは作成することをお勧めします。レプリカによって、検索パフォーマンスを改善することもできます。読み込みが多いワークロードが発生する場合は、より多くのレプリカがあるとよいでしょう。`PUT /my-index/_settings` を使用して、インデックスの `number_of_replicas` 設定を更新します。
- OpenSearch インデックス作成のオーバーヘッド: ディスク上のインデックスのサイズは一定ではありません。多くの場合、ソースデータとインデックスの合計サイズはソースの 110% で、インデックスはソースデータの 10% までです。データのインデックスを作成したら、`_cat/indices?v` API と `pri.store.size` 値を使用して正確なオーバーヘッドを計算できます。`_cat/allocation?v` も有用な要約を提供します。
- オペレーティングシステムの予約スペース: デフォルトでは、Linux は、重要なプロセス、システム復元、そしてディスクの断片化問題に対する保護目的で、`root` ユーザー用にファイルシステムの 5% を予約します。
- OpenSearch Service のオーバーヘッド: OpenSearch Service は、セグメントマージ、ログ、およびその他の内部オペレーション用として、インスタンスごとにストレージ容量の 20% (最大 20 GiB) を予約します。

この 20 GiB の上限があるため、予約容量の合計はドメインに存在するインスタンスの数によって大きく異なります。たとえば、ドメインに 3 つの `m6g.xlarge.search` インスタンスがあり、それぞれのストレージ容量が 500 GiB であるとすると、合計は 1.46 TiB になります。この場合、予約される容量はわずか 60 GiB です。ドメインに 10 の `m3.medium.search` インスタンスがあり、それぞれのストレージ容量が 100 GiB の場合は、合計で 0.98 TiB になります。最初のドメインの方が 50% 大きいですが、後者の予約容量は合計 200 GiB です。

次の式では、オーバーヘッドの「ワーストケース」の見積もりを適用します。この見積もりには、ノード障害やアベイラビリティーゾーンの停止による影響を最小限に抑えるのに役立つ追加の空き容量が含まれています。

つまり、任意の時点で 66 GiB のデータが存在し、レプリカを 1 つ必要とする場合、最小ストレージ要件は 66 * 2 * 1.1/0.95/0.8 = 191 GiB になります。この計算は、次のように定型化できます。

ソースデータ * (1 + レプリカ数) * (1 + インデックス作成オーバーヘッド) / (1 - Linux 予約スペース) / (1 - OpenSearch Service のオーバーヘッド) = 必要な最小ストレージ

あるいは、この簡素化されたバージョンを使用することもできます。

ソースデータ * (1 + レプリカの数) * 1.45 = 必要な最小ストレージ

ストレージ容量の不足は、クラスターが不安定になる最も一般的な原因の 1 つです。したがって、インスタンスタイプ、インスタンス数、およびストレージボリュームを選択するときは、数値をクロスチェックする必要があります。

ストレージに関するその他の考慮事項は次のとおりです。

- 最小ストレージ要件が 1 PB を超える場合は、「the section called “ペタバイトスケール”」を参照してください。
- ローリングインデックスがあり、ホットウォームアーキテクチャを使用する場合は、「the section called “UltraWarm ストレージ”」を参照してください。

シャード数の選択

ストレージ要件を理解したら、インデックス作成の戦略を調査できます。OpenSearch Service のデフォルトでは、各インデックスは 5 つのプライマリシャードと 1 つのレプリカ (合計 10 個のシャード) に分割されます。この動作は、デフォルトで 1 つのプライマリシャードと 1 つのレプリカシャー

ドに分割するオープンソースの OpenSearch とは異なります。既存のインデックスに対してプライマリシャードの数を簡単に変更することはできません。したがって、シャード数は最初のドキュメントでインデックスを作成する前に決定する必要があります。

シャード数を選択することの全体的な目標は、クラスター内のすべてのデータノード間でインデックスを均等に分散させることです。ただし、これらのシャードは大きすぎたり多すぎたりしてはいけません。一般的なガイドラインとして、検索レイテンシーが主要なパフォーマンス目標であるワークロードではシャードのサイズを 10～30 GiB、ログ分析などの書き込みが多いワークロードでは 30～50 GiB の範囲に維持することをお勧めします。

シャードが大きいと、OpenSearch の障害復旧が困難になる可能性があります。一方、各シャードはある程度の CPU とメモリを使用するため、小規模なシャードが多数ある場合には、パフォーマンスの問題やメモリ不足の エラーが発生する可能性があります。つまり、基盤となる OpenSearch Service のインスタンスで問題なく処理できるようにシャードは小さくする必要がありますが、小さすぎるとハードウェアに不必要な負荷をかけてしまうということです。

たとえば、66 GiB のデータがあるとします。その数値は今後も増加する予定がなく、各シャードのサイズは約 30 GiB を維持する必要があります。この場合、シャード数は約 3 つとなります (66 * 1.1/30 = 3)。この計算は、次のように定型化できます。

(ソースデータ + 拡張の余地) * (1 + インデックス作成オーバーヘッド) / 必要なシャードサイズ = プライマリシャードの概数

この式は、時間の経過に伴うデータ拡張にも対応できます。同じ 66 GiB のデータが来年は 4 倍になると想定される場合、シャード数はおよそ (66 + 198) * 1.1/30 = 10 となります。ただし、追加分の 198 GiB のデータは現時点では存在しません。将来に向けてのこのような準備として不必要な小さいシャードを作成し、現時点で CPU とメモリを大量に消費することがないようにしてください。この場合、シャード当たりのサイズは 66 * 1.1/10 シャード = 7.26 GiB となります。これは余分なリソースを消費する場合があり、推奨サイズ範囲を下回ります。中間をとったアプローチとしてシャード数を 6 つにすると、シャードのサイズは現時点で 12 GiB、将来的には 48 GiB となります。その後、シャードが 50 GiB を超えたときには、3 つのシャードを使用してデータのインデックスを再作成することもできます。

かなりまれな問題の場合、ノードあたりのシャード数を制限する必要があります。シャードのサイズを適切に設定した場合、通常はこの制限に達するかなり前にディスク容量が不足します。たとえば、`m6g.large.search` インスタンスの最大ディスクサイズは 512 GiB です。ディスク使用率が 80% 未満に維持されており、シャードのサイズを 20 GiB に設定した場合、約 20 個のシャードを収容できます。Elasticsearch 7. 以降、および OpenSearch 2.15 までのすべてのバージョンには、1 ノードあたり 1,000 シャードの制限があります。ノードあたりの最大シャードを調整す

るには、cluster.max_shards_per_node 設定を設定してください。OpenSearch 2.17 以降では、OpenSearch Service は JVM ヒープメモリ 16 GB 当り 1,000 シャード、最大 1 ノード当り 4,000 シャードをサポートします。例については、「Cluster settings」(クラスターの設定) を参照してください。シャード数の詳細については、「the section called “シャード数のクォータ”」を参照してください。

シャードのサイズを適切に設定すると、ほとんどの場合この制限未満に維持できますが、Java ヒープの GiB ごとにシャードの数を検討することもできます。ノードごとに、Java ヒープの GiB あたりのシャード数を 25 以下にしてください。例えば、m5.large.search インスタンスに 4 GiB のヒープがあるとすると、各ノードのシャード数は 100 以下にすることになります。そのシャード数では、各シャードのサイズが約 5 GiB になり、推奨値をかなり下回ります。

インスタンスタイプとテストの選択

ストレージ要件を計算して必要なシャード数を選択したら、ハードウェアに関する決定ができるようになります。ハードウェア要件はワークロードによって大きく異なるものの、基本的な推奨事項は提供されています。

各インスタンスタイプのストレージ制限は一般的に、軽度のワークロードで必要とされる CPU とメモリの量にマッピングされています。たとえば、m6g.large.search インスタンスが最大で 512 GiB の EBS ボリュームサイズ、vCPU x 2 コア、8 GiB のメモリを備えているとします。クラスターには多数のシャードがあり、高負荷の集計処理、頻繁なドキュメント更新、または大量のクエリ処理が発生している場合、それらのリソースではニーズを満たせない可能性があります。クラスターがこのようなカテゴリに分類される場合はまず、ストレージ要件の容量 100 GiB ごとに vCPU x 2 コア、メモリ 8 GiB に近い構成になるようにします。

> **Tip**
>
> 各インスタンスタイプに割り当てられるハードウェアリソースの概要については、「Amazon OpenSearch Service の料金表」を参照してください。

ただし、上記に記載されたリソースでも不十分になる場合があります。一部の OpenSearch ユーザーからは、要件を満たすためにこれらのリソースが重ねて必要になるという報告を受けています。ワークロードに適したハードウェアを見つけるには、知識に基づいた初期予測を作成し、代表的なワークロードでテストし、調整して、再度テストする必要があります。

ステップ 1: 初期見積もりを作成する

開始の際に、スプリットブレインの状態 (コミュニケーションでの時間の経過により、クラスターが 2 つのマスターノードを持つようになる) などの潜在的な OpenSearch の問題を防ぐために、3 つ以上のノード数を選択することをお勧めします。専用マスターノードが 3 つの場合は、レプリケーション用のデータノードは少なくとも 2 つにすることをお勧めします。

ステップ 2: ノードごとのストレージ要件を計算する

ストレージ要件が 184 GiB で、推奨最小数である 3 つのノードがある場合、各ノードで必要とするストレージの容量は 184/3 = 61 GiB という計算になります。この例では、3 つの `m6g.large.search` インスタンスを選択してそれぞれで 90 GiB の EBS ストレージボリュームを使用すれば、安全策として時間の経過に伴う拡張にも備えることができます。この構成では 6 つの vCPU コアと 24 GiB のメモリを利用でき、軽度のワークロードに適しています。

より大規模な例として、ストレージ要件が 14 TiB (14,336 GiB) で高い負荷が発生する状況を想定します。この場合、まずは 2 * 144 = 288 の vCPU コアと 8 * 144 = 1,152 GiB のメモリを使用したテストが考えられます。これらの数値は、約 18 `i3.4xlarge.search` インスタンスを使用する結果となります。高速なローカルストレージが必要ない場合、それぞれが 1 TiB の EBS ストレージボリュームを使用する 18 個の `r6g.4xlarge.search` インスタンスをテストすることもできます。

クラスターに数百テラバイトのデータが含まれる場合は、「the section called “ペタバイトスケール”」を参照してください。

ステップ 3: 代表的なテストを実行する

クラスターを設定したら、先ほど計算したシャードの数を使用してインデックスを追加し、実践的なデータセットを使用して代表的なクライアントテストを実行し、CloudWatch メトリクスのモニタリングによりクラスターでのワークロードの処理状況を確認できます。

ステップ 4: 成功または反復する

パフォーマンスがニーズを満たすものであれば、テストは成功し、CloudWatch メトリクスは正常であり、クラスターは使用する準備ができたことになります。リソースの異常な使用状況を検出するために、必ず CloudWatch アラームを設定してください。

パフォーマンスが許容できないものであれば、テストが失敗したか、または `CPUUtilization` か `JVMMemoryPressure` が高いことになります。別のインスタンスタイプを選択 (またはインスタンスを追加) して、テストを続行する必要があるかもしれません。インスタンスを追加すると、OpenSearch はクラスター全体にわたってシャードを自動的に再分散します。

過負荷クラスターの過剰容量は、負荷の低いクラスターの不足よりも簡単に測定できるため、必要以上に大きなクラスターから開始することをお勧めします。次に、追加のリソースを持つ効率的なクラスターにテストおよびスケールダウンして、アクティビティの増加期間中に安定した運用を確保します。

本番稼働用クラスター (1 つまたは複数) では複雑なステータスが発生しますが、専用マスターノードを活用することでパフォーマンスとクラスターの信頼性を向上させることができます。

Amazon OpenSearch Service 用ペタバイトスケール

Amazon OpenSearch Service ドメインは、最大 10 PB のアタッチストレージを提供します。ドメインは 1,000 個の`OR1.16xlarge.search`インスタンスタイプで設定でき、それぞれに 36 TB のストレージがあります。規模がまったく異なるため、このサイズのドメインに関する推奨事項は一般的な推奨事項とは異なります。このセクションでは、ドメインの作成、コスト、ストレージ、シャードのサイズに関する考慮事項について説明します。

このセクションでは頻繁に`i3.16xlarge.search`インスタンスタイプを参照しますが、他のいくつかのインスタンスタイプを使用してドメインストレージの合計を 10 PB にすることができます。

ドメインの作成

このサイズのドメインは、ドメインあたり 80 インスタンスというデフォルトの制限を超えています。ドメインあたり最大 1000 インスタンスのサービス制限の引き上げをリクエストするには、 AWS サポートセンターでケースを開きます。

料金

このサイズのドメインを作成する前に、Amazon OpenSearch Service の料金表ページを確認して、関連コストが希望に沿っていることを確認します。ホットウォームアーキテクチャがユースケースに合っているかどうかを確認するために the section called “UltraWarm ストレージ” を調べます。

Storage

`i3` インスタンスタイプは、高速な不揮発性メモリエクスプレス (NVMe) ローカルストレージを提供するよう特別に設計されたものです。このローカルストレージは Amazon Elastic Block Store (EBS) と比較してパフォーマンスが向上する傾向があるため、OpenSearch Service でこのインスタンスタイプを選択した場合は EBS ボリュームを選択することはできません。必要に応じて EBS ストレージを使用する場合は、`r6.12xlarge.search` など、別のインスタンスタイプを使用します。

シャードのサイズと数

OpenSearch の一般的なガイドラインは、シャードあたり 50 GB を超えないようにすることです。大きなドメインに対応するために必要なシャードの数および `i3.16xlarge.search` インスタンスが利用可能なリソースが指定されている場合は、シャードのサイズは 100 GB をお勧めします。

たとえば、450 TB のソースデータがあり、レプリカを 1 つにする場合、最小ストレージ要件は 450 TB * 2 * 1.1/0.95 = 1.04 PB です。この計算の説明については、「the section called “ストレージ要件の計算”」を参照してください。1.04 PB/15 TB = 70 インスタンスですが、時間と共に変動するデータ量を考慮して、ストレージセーフティネットを提供し、ノードの障害を処理するために、90 個以上の `i3.16xlarge.search` インスタンスを選択することができます。各インスタンスにより最小ストレージ要件に 20 GiB が追加されますが、このサイズのディスクでは、この 20 GiB はほぼ無視できます。

シャード数の制御には手間がかかります。OpenSearch ユーザーは、インデックスを毎日ローテーションし、データを 1~2 週間保持することがよくあります。このような状況では、「アクティブ」なシャードと「非アクティブ」なシャードを区別すると便利な場合があります。アクティブなシャードとは、書き込みや読み取りがアクティブに行われているシャードです。非アクティブなシャードとは、いくつかの読み取りリクエストがあるものの大部分はアイドルな状態のサービスです。一般的には、アクティブなシャードの数を数千個未満に維持する必要があります。アクティブなシャードの数が 10,000 に近づくと、パフォーマンスと安定性に大きなリスクが出現します。

プライマリシャードの数は次の式を使用して計算します。450,000 GB * 1.1/(100 GB/シャード) = 4,950 シャード。レプリカを考慮してこの数を 2 倍にすると 9,900 シャードになり、すべてのシャードがアクティブの場合は大きな懸念事項になります。ただし、インデックスをローテーションして、任意の指定日にシャードの数の 1/7 または 1/14 (それぞれ 1,414 または 707 シャード) のみがアクティブな場合、クラスターはうまく機能します。この場合も、ドメインのサイズ決定と設定において最も重要なステップは、現実的なデータセットを使用して代表的なクライアントテストを実行することです。

Amazon OpenSearch Service の専用コーディネーターノード

Amazon OpenSearch Service の専用コーディネーターノードは、データノードから調整タスクをオフロードする特殊なノードです。これらのタスクには、検索リクエストの管理や OpenSearch Dashboards のホスティングなどが含まれます。これらの機能を分離することで、専用コーディネーターノードはデータノードの負荷を軽減し、データノードがデータストレージ、インデックス作成、

検索オペレーションに集中できるようにします。これにより、クラスター全体のパフォーマンスとリソース使用率が向上します。

さらに、専用コーディネーターノードは、VPC 設定に必要なプライベート IP アドレスの数を減らすため、より効率的なネットワーク管理につながります。この設定により、ワークロードの特性によっては、インデックス作成スループットが最大 15%、クエリパフォーマンスが 20% 向上します。

専用コーディネーターノードを使用すべきシチュエーション

専用コーディネーターノードは、以下のシナリオで最も有益です。

- 大規模なクラスター – 大量のデータや複雑なクエリがある環境では、調整タスクを専用ノードにオフロードすることで、クラスターのパフォーマンスを向上させることができます。
- 頻繁なクエリ – 頻繁な検索クエリや集計を含むワークロード、特に複雑な日付ヒストグラムや複数の集計を含むワークロードでは、クエリ処理の高速化により大きなメリットが得られます。
- ダッシュボードの高負荷使用 – OpenSearch Dashboards はリソースを大量に消費するツールです。この責任を専用コーディネーターノードにオフロードすることで、データノードへの負担が軽減されます。

アーキテクチャと動作

OpenSearch クラスターでは、専用コーディネーターノードが 2 つの主要な責任を処理します。

- リクエスト処理 – これらのノードは検索リクエストを受信し、関連するデータが保存されている適切なデータノードに転送します。その後、複数のデータノードからの結果を 1 つのグローバル結果セットに統合してクライアントに返します。
- ダッシュボードホスティング – コーディネーターノードは OpenSearch Dashboards を管理します。これにより、OpenSearch Dashboards をホストし、関連するトラフィックを処理するという追加の負担からデータノードを解放できます。

VPC ドメインでは、専用コーディネーターノードには、データノードではなく Elastic Network Interface (ENI) が割り当てられます。この配置により、VPC に必要なプライベート IP アドレスの数を減らすことができ、ネットワーク効率が向上します。通常、専用コーディネーターノードは、データノード全体の約 10% を占めます。

要件と制限

専用コーディネーターノードには、次の要件と制限があります。

- 専有コーディネーターノードは、すべての OpenSearch バージョンと Elasticsearch バージョン 6.8 から 7.10 でサポートされています。
- 専用コーディネーターノードを有効にするには、ドメインで専用マスターノードが有効になっている必要があります。詳細については、「the section called “専用マスターノード”」を参照してください。
- 専用コーディネーターノードをプロビジョニングすると、追加コストが発生する可能性があります。ただし、リソース効率の向上とパフォーマンスの向上により、特に大規模または複雑なクラスターにおける投資は正当化されるでしょう。

専用コーディネーターノードのプロビジョニング

既存のドメインに専用コーディネーターノードをプロビジョニングするには、次の手順を実行します。コーディネーターノードをプロビジョニングする前に、ドメインで専用マスターノードが有効になっていることを確認してください。

コンソール

で専用コーディネーターノードをプロビジョニングするには AWS マネジメントコンソール

1. https://console.aws.amazon.com/aos/home で Amazon OpenSearch Service コンソールにサインインします。
2. [ドメイン] を選択し、変更するドメインを選択します。
3. [クラスターの設定] セクションで、[編集] を選択します。
4. [専用コーディネーターノードを有効にする] を選択します。
5. プロビジョニングするコーディネーターノードのインスタンスタイプと数を選択します。
6. [Save changes] (変更の保存) をクリックします。ドメインが更新されるまでに数分かかる場合があります。

AWS CLI

を使用して専用コーディネーターノードをプロビジョニングするには AWS CLI、update-domain-config コマンドを使用します。次の例では、ドメインに 3 つの `r6g.large.search` コーディネーターノードをプロビジョニングします。

```
aws opensearch update-domain-config \
  --domain-name my-opensearch-domain \
```

```
 --cluster-config
InstanceCount=3,InstanceType=r6g.large.search,DedicatedCoordinatorCount=3,ZoneAwarenessEnabled
```

このコマンドは、専用コーディネーターノードを有効にし、コーディネーターノードのインスタンスタイプと数を設定し、ゾーン認識を有効にて可用性を高めます。

ベストプラクティス

専用コーディネーターノードを使用する場合は、次のベストプラクティスを考慮してください。

- ほとんどのユースケースについては汎用インスタンスを使用してください。コストとパフォーマンスのバランスの取れたアプローチを提供します。メモリ最適化インスタンスは、複雑な集約や大規模な検索など、大量のメモリリソースを必要とするワークロードに最適です。
- 開始点として、データノードの 5%～10% を専用コーディネーターノードとしてプロビジョニングすることをお勧めします。例えば、ドメインに特定のインスタンスタイプのデータノードが 90 個ある場合は、同じインスタンスタイプのコーディネーターノードを 5～9 個プロビジョニングすることを検討してください。

  > **Note**
  >
  > 使用可能なインスタンスタイプは、リージョンごとに異なります。コーディネーターノードのインスタンスタイプを選択するときは、選択したインスタンスタイプが対象リージョンで使用可能かどうかを確認します。ドメインを作成または変更するときに、OpenSearch Service コンソールで利用可能なインスタンスタイプを確認できます。

- 単一障害点のリスクを最小限に抑えるには、少なくとも 2 つの専用コーディネーターノードをプロビジョニングします。これにより、1 つのノードに障害が発生しても、クラスターは引き続き動作を継続できます。
- クロスリージョン検索を使用している場合は、ターゲットドメインに専用のコーディネーターノードをプロビジョニングしてください。ソースドメインは通常、調整タスクを処理しません。
- インデックス作成が多い環境では、最適なパフォーマンスを得るために、データノードのインスタンスサイズに一致する CPU 最適化インスタンスを検討してください。
- メモリを大量に消費するワークロードでは、メモリ需要の増加を管理するために、専用コーディネーターノードに少し大きめのインスタンスタイプを使用します。
- `CoordinatorCPUUtilization` Amazon CloudWatch メトリクスを追跡してください。一貫して 80% を超えている場合は、負荷を処理するためにより大きなコーディネーターノードまたは追加のコーディネーターノードが必要であることを示している可能性があります。

- データノードに合わせて専用コーディネーターノードのサイズを設定しましょう。例えば、4xlarge データノードを使用する場合は、4xlarge 汎用コーディネーターノードから始めます。
- 個々のリクエストまたはレスポンスで非常に高いメモリ (GB 単位) が必要でない限り、コーディネーターノードのインスタンス数を減らす代わりに、小さいインスタンスを複数使用します。例えば、6 つの 8xlarge 汎用インスタンスよりは 12 個の 4xl インスタンスを選択しましょう。

クラスターサイズ別のノードの推奨事項

クラスターサイズに基づいて専用コーディネーターノードをプロビジョニングするための出発点として、次のガイドラインを参照ください。ワークロードの特性とパフォーマンスメトリクスに基づいて、ノードの数とタイプを調整します。

クラスターサイズ	推奨コーディネーターノード	インスタンスタイプ
小規模 (最大 50 ノード)	3 ~ 5 ノード	汎用
中規模 (50 ~ 100 ノード)	5 ~ 9 ノード	メモリ最適化
大規模 (100 ノード以上)	10 ~ 15 ノード	メモリ最適化

Amazon OpenSearch Service の専用マスターノード

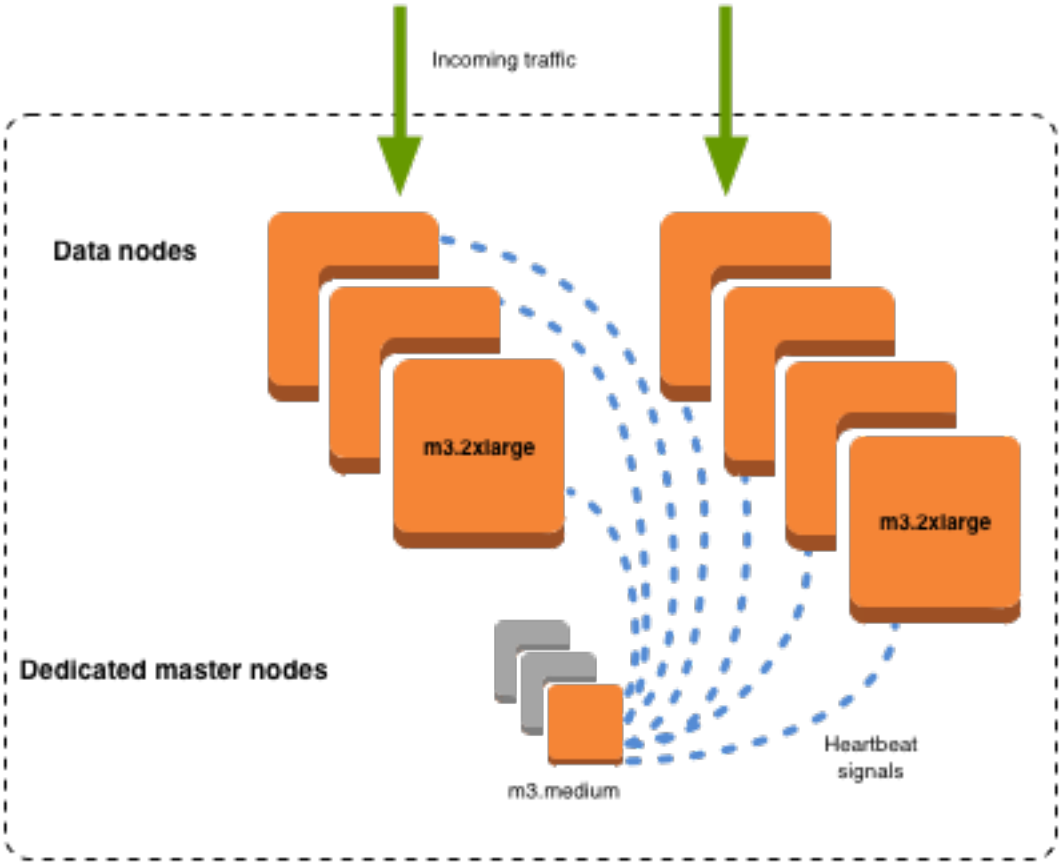
Amazon OpenSearch Service では、クラスターの安定性を向上するために専用マスターノードを使用します。専用マスターノードではクラスター管理タスクを実行しますが、データは保持せず、データのアップロードリクエストにも応答しません。このように、クラスター管理タスクをオフロードすると、ドメインの安定性が向上します。他のすべてのノードタイプと同様に、専用マスターノードごとに時間料金を支払います。

専用マスターノードで実行するクラスター管理タスクは次のとおりです。

- クラスター内のすべてのノードを追跡します。
- クラスター内のインデックスの数を追跡します。

- 各インデックスに属するシャード数を追跡します。
- クラスター内のノードのルーティング情報を保持します。
- クラスター内のインデックス作成やノードの作成/削除など、状態が変化したときにクラスターの状態を更新します。
- クラスターの状態に施された変更を、クラスター内のすべてのノードにわたって複製します。
- クラスター内のデータノードの可用性をモニタリングするハートビートシグナル (定期的なシグナル) を送信することで、すべてのクラスターノードの状態をモニタリングします。

次の図は、10 のインスタンスを持つ OpenSearch Service ドメインを表しています。インスタンスのうち 7 つはデータノードで、3 つは専用マスターノードです。アクティブになっているのは、専用マスターノードの 1 つだけです。2 つの灰色の専用マスターノードは、アクティブな専用マスターノードに障害が発生した場合のバックアップとして待機します。すべてのデータアップロードリクエストは、7 つのデータノードにより保存され、すべてのクラスター管理タスクは、アクティブな専用マスターノードにオフロードされます。



専用マスターノードの数の選択

スタンバイが有効のマルチ AZ を使用することを推奨します。これにより、本番稼働用の各 OpenSearch Service ドメインに 3 つの専用マスターノードが追加されます。スタンバイなしのマルチ AZ 、またはシングル AZ でデプロイする場合でも、3 つの専用マスターノードを推奨します。偶数の専用マスターノードを選択しないでください。専用マスターノードの数を選択するときは、次の点を考慮してください。

- 障害発生時にバックアップがないため、専用マスターノードの 1 つが OpenSearch Service によって明示的に禁止されています。1 つの専用マスターノードのみを持つドメインを作成しようとすると、検証例外が表示されます。
- 2 つの専用マスターノードがある場合、クラスターには、障害が発生した場合に新しいマスターノードを選択するために必要なノードのクォーラムがありません。

  クォーラムは、専用マスターノードの数/2 + 1 (直近の整数まで切り捨て) です。ここでは 2/2 + 1 = 2 となります。専用マスターノード 1 つが失敗し、バックアップは 1 つのみであれば、クラスターにはクォーラムがなく、新しいマスターを選択できません。
- 推奨される専用マスターノード数は 3 つです。マスターノードが失敗したときに 2 つのバックアップノードが使用でき、必要なクォーラム (2) で新しいマスターを選択できます。
- 専用マスターノードを 4 つにすると 3 つより良くなるわけではなく、複数のアベイラビリティーゾーンを使用した場合、問題が発生する可能性があります。
  - マスターノード 1 つが失敗した場合、クォーラム (3) で新しいマスターを選択します。2 つのノードが失敗したした場合、そのクォーラムは失われ、専用マスターノードが 3 つの場合と同じ状況になります。
  - 3 つのアベイラビリティーゾーン (AZ) 設定では、2 つの AZ に 1 つの専用マスターノードがあり、1 つの AZ に 2 つの専用マスターノードがあります。その AZ で中断が発生した場合、残りの 2 つの AZ には、新しいマスターを選択するために必要なクォーラム (3) がありません。
- 5 つの専用マスターノードは 3 つと同様に機能し、クォーラムを維持しながら 2 つのノードを失うことができます。ただし、一度にアクティブになる専用マスターノードは 1 つしかないため、この設定では 4 つのアイドルノードに対して料金を支払うことになります。多くのお客様は、このレベルのフェイルオーバーは過剰だと考えています。

クラスターに偶数のマスター対象ノードがある場合、OpenSearch および Elasticsearch バージョン 7.x 以降では 1 つのノードが無視されるため、投票設定は常に奇数になります。この場合、基本的に専用マスターノード 4 つは 3 つ (2 つは 1 つ) に相当します。

> **Note**
>
> 新しいマスターノードを選択するために必要なクォーラムがクラスターにない場合は、クラスターへの書き込みおよび読み取りリクエストの両方が失敗します。この動作は OpenSearch のデフォルトとは異なります。

専用マスターノードのインスタンスタイプの選択

OpenSearch Service のドメインとインスタンスクォータ

専用マスターノードでは検索およびクエリリクエストを処理しませんが、そのサイズは、管理可能なインスタンスサイズや、インスタンス数、インデックス数、シャード数と大きな相関があります。本番稼働用クラスターでは、少なくとも、専用マスターノードの以下のサイズ設定に関する推奨事項をお勧めします。

これらの推奨事項は一般的なワークロードに基づいており、お客様の要件によって異なります。多数のシャードまたはフィールドのマッピングがあるクラスターは、大きなインスタンスタイプからメリットを得ることができます。詳細については、「Amazon OpenSearch Service で推奨される CloudWatch アラーム」を参照して、より大きなインスタンスタイプを使用する必要があるかどうかを判断してください。

RAM	Elasticsearch および OpenSearch Service 1.x から 2.15 での最大ノードサポート	Elasticsearch および OpenSearch Service 1.x から 2.15 での最大シャードサポート	OpenSearch Service 2.17 以降での最大ノードサポート	OpenSearch Service 2.17 以降での最大シャードサポート
2 GB	該当しない	該当しない	10	1K
4 GB	該当しない	該当しない	10	5K
8 GB	10	10K	30	15K
16 GB	30	30K	60	30K
32 GB	75	40K	120	60K

RAM	Elasticsearch および OpenSearch Service 1.x から 2.15 での最大ノードサポート	Elasticsearch および OpenSearch Service 1.x から 2.15 での最大シャードサポート	OpenSearch Service 2.17 以降での最大ノードサポート	OpenSearch Service 2.17 以降での最大シャードサポート
64 GB	125	75K	240	120K
128 GB	200	75K	480	240K
256 GB	該当しない	該当しない	1002	500K

Amazon OpenSearch Service の全般的なリファレンス

Amazon OpenSearch Service は、さまざまなインスタンス、オペレーション、プラグイン、他のリソースをサポートしています。

トピック

- Amazon OpenSearch Service のコスト最適化手法
- Amazon OpenSearch Service でサポートされるインスタンスタイプ
- Amazon OpenSearch Service のエンジンバージョン別の機能
- Amazon OpenSearch Service のエンジンバージョンに応じたプラグイン
- Amazon OpenSearch Service でサポートされているオペレーション
- Amazon OpenSearch Service クォータ
- Amazon OpenSearch Service のリザーブドインスタンス
- Amazon OpenSearch Service でサポートされている他のリソース

Amazon OpenSearch Service のコスト最適化手法

以下は、Amazon OpenSearch Service の使用中にコストを最適化するために最もよく使用される手法の一部です。マネージドクラスターとサーバーレスの両方です。すべてのワークロードは一意であるため、これらの戦略を特定の使用パターンに照らして評価し、本番環境に適用する前にテスト環境で検証します。

トピック

- 1-Yearまたは複数年のコミットメントで大きな節約
- Amazon OpenSearch Service マネージドクラスターのコスト最適化
- Amazon OpenSearch Service Serverless のコスト最適化

1-Yearまたは複数年のコミットメントで大きな節約

Amazon OpenSearch Service では、最低料金や使用要件なしで使用した分のみお支払いいただきます。Amazon OpenSearch Service には 2 つのデプロイモデルがあります。

- マネージドクラスターの場合、インスタンス時間、ストレージ、データ転送に対して課金されます。料金は、選択したインスタンスタイプとストレージ階層によって異なります。インスタン

スの場合は、リザーブドインスタンスの料金を使用するか、オンデマンド料金よりも Database Savings Plans で保存できます。

- Serverless の場合、コンピューティングとストレージに対して個別に課金されます。コンピューティング容量は OpenSearch Compute Units (OCUs は、データのインデックス作成やクエリの実行に必要な CPU、メモリ、I/O リソースに対応します。Serverless は Database Savings Plans でもカバーされています。

Database Savings Plans は、1 年間の一貫した使用量にコミット ($/時間で測定) するときに Amazon OpenSearch Service のコストを削減する柔軟な料金モデルです。この節約プランは、エンジンバージョン、インスタンスファミリー、サイズ、デプロイオプションに関係なく、または前払い AWS リージョン なしで、対象となるサーバーレスおよびプロビジョニングされたインスタンスの使用に自動的に適用されます。詳細については、 Database Savings Plans の料金ページを参照してください。

リザーブドインスタンスは、1 年間または 3 年間の特定のインスタンス設定にコミットするときに Amazon OpenSearch Service のコストを削減する料金モデルを提供します。リザーブドインスタンスは、特定のインスタンスタイプ、サイズ、および の容量を予約する場合、オンデマンド料金と比較して大幅な割引を提供します AWS リージョン。支払いオプションには、全額前払い、一部前払い、前払いなしなどがあります。前払いの支払いには割引が適用されます。詳細については、Amazon OpenSearch Service リザーブドインスタンスの料金ページを参照してください。

Amazon OpenSearch Service マネージドクラスターのコスト最適化

派生ソース — _source フィールドの保存をスキップします

派生ソースは、 `_source`フィールドを保存するオーバーヘッドを排除するストレージ最適化機能です。

- OpenSearch は、取り込まれたすべてのドキュメントを 2 回保存します。1 回は `_source`フィールド (raw ドキュメント) に、もう 1 回は検索用のインデックス付きフィールドとして保存します。
- `_source` フィールド単独では、大量のストレージ領域を消費する可能性があります。多くの場合、インデックスストレージ全体の 30 ~ 50% です。
- 派生ソースでは、保存をスキップ`_source`し、必要に応じてインデックス付きフィールドから動的に再構築します (検索、取得、mget、再インデックス、または更新オペレーション中) 。
- これはオプトインであり、複合インデックス設定を使用してインデックスの作成時に有効になります。

- OpenSearch 3.1 以降がサポートされているすべてのリージョンで使用できます。

最適: 元の raw ドキュメントを頻繁に取得する必要はなく、フィールドを検索して集計する必要がある分析ワークロードとログワークロード。

詳細については、オープンソースのドキュメントと最新情報のお知らせを参照してください。

OR1 / OR2 / OM2 インスタンス — OpenSearch 最適化インスタンスファミリー

OR1 以降の OR2 および OM2 インスタンスは、セグメントレプリケーションを介したレプリカストレージに Amazon S3 を使用します。

- OR2: OR1 よりもインデックス作成スループットが最大 26% 高く、R7g インスタンスよりも 70% 高くなります。
- OM2: OR1 よりもインデックス作成スループットが最大 15% 高く、M7g インスタンスよりも 66% 高くなります。
- どちらも同じアーキテクチャを使用しています。プライマリストレージにはローカル EBS、耐久性には S3 です。
- レプリカストレージコストを排除 - 高価な EBS ボリュームの代わりに S3 に保存されるレプリカ (耐久性 9 分の 11)。
- 前世代のインスタンスと比較して、最大 30% の価格パフォーマンスの向上。
- 浅いスナップショット v2 をサポート — I/O オーバーヘッドのないほぼ即時のスナップショット。

最適: 大量の運用分析とログ分析ワークロードのインデックス作成。

詳細については、OR2 および OM2 What's New announcement」およびthe section called “OpenSearch 最適化インスタンス”「 developer guide」トピックを参照してください。

インデックスロールアップ — 履歴時系列データを集計する

インデックスロールアップは、古い時系列データを要約してより粗い時間間隔に圧縮し、ストレージボリュームを大幅に削減します。

- IoT/センサーデータ: 最近の期間、1 秒あたりのデータをホットストレージに保持します。古いデータについては、時間単位または日単位の概要にロールアップします。
- システムメトリクス: 過去 30 日間の詳細なメトリクスを保持し、古いデータを時間単位または日単位の概要に集約します。

- ログデータ: アクティブなトラブルシューティングウィンドウ (1 週間など) の完全な詳細を保持し、古い期間に要約されたエラーパターンを維持します。
- ISM ポリシーと組み合わせて、単一のライフサイクルポリシーでロールアップと階層の移行を自動化します。
- 秒単位から数時間単位に集約する場合、秒単位から数分単位に集約する場合よりも大幅な節約になります。

詳細については、Index Rollups ブログ記事を参照してください。

インデックス状態管理 — フルデータライフサイクルを自動化する

ISM ポリシーは、ストレージ階層とライフサイクルアクションによるインデックスの移動を自動化します。

- インデックスを自動的に移行する: 年齢、サイズ、ドキュメント数に基づいて、ホットから UltraWarm へ、コールドから削除へ。
- 階層が移行する前にロールアップをトリガーして、データ量を減らします。
- ロールオーバーポリシー (インデックスが 50 GB に達した場合や 30 日が経過した場合など) を設定して、インデックスの増加を制御します。
- 読み取り専用インデックスの強制マージを自動化して、削除されたドキュメントからストレージを再利用します。
- ロールアップと組み合わせることで、大規模な時系列データセットの節約を最大化できます。

適切なサイズのインスタンスタイプと数

Well-Architected OpenSearch レンズの主なガイダンスと適切なサイジングのベストプラクティス:

- 常に最新世代のインスタンスを使用してください (例えば、Graviton3 インスタンスは Graviton2-basedインスタンスよりもパフォーマンスが最大 25% 向上します) 。
- gp2 の代わりに gp3 EBS ボリュームを使用する — 追加料金なしで低コストでパフォーマンスが向上します。
- インスタンスタイプをワークロードに一致: 検索量が多い場合はメモリ最適化、インデックス作成量が多い場合はコンピューティング最適化。
- 専用クラスターマネージャーノードを評価する: 3 つ以上のデータノードにのみ必要です。マスターノードサイズの過剰プロビジョニングは避けてください。

- CloudWatch メトリクスをモニタリングしてオーバープロビジョニングを検出します。持続的な CPU が 40% 未満、JVM が 50% 未満、ストレージが 50% 未満の場合は、無駄の兆候です。
- 最適な範囲: CPU 60～80%、JVM 65～85%、持続的なワークロードのストレージ 70～85%。

詳細については、「Right-Sizing Best Practices」ブログ記事を参照してください。

シャードの最適化 — オーバーシャーディングを回避する

オーバーシャーディングは隠れたコストドライバーです。小さなシャードが多すぎると CPU、メモリ、JVM ヒープが浪費されます。

- 推奨されるシャードサイズ: ワークロードに応じて、シャードあたり 10～50 GiB。
- Java ヒープの GiB あたり 25 個以下のシャード、データノードあたり 1,000 個以下のシャード。
- ISM ロールオーバーポリシーを使用してインデックスの増加を制御し、無制限のシャードの拡散を回避します。
- 耐久性が許容されるレプリカ数を減らします (OR1/OR2 インスタンスはレプリカを完全に不要にします)。
- 読み取り専用インデックスに強制マージを使用して、セグメント数を減らし、ストレージを再利用します。

詳細については、「必要なシャードの数」を参照してください。

Amazon S3 でのゼロ ETL / 直接クエリ

クエリはほとんど行われないが、引き続きアクセス可能なデータの場合、Direct Query (S3 ではゼロ ETL) では、S3 データを OpenSearch から直接クエリでき、取り込む必要はありません。

- 取り込みコストなし — データは S3 に残ります。
- アーカイブデータのホット階層ストレージコストはかかりません。
- Pay-per-queryコンピューティングモデル。
- 視覚化のための OpenSearch Dashboards をサポートします。
- 秒または分のレイテンシーは許容されます。リアルタイムのユースケースでは許容されません。

取り込み時のサンプリングと圧縮

データが OpenSearch に到達する前にコストを削減します。

- サンプリング: 大量のログストリームの代表的なサブセットのみを取り込みます (デバッグログの 10% など)。
- インデックス圧縮: 最適な圧縮コーデックを有効にして、ストレージフットプリントを削減します。
- フィールドフィルタリング: インデックス作成の前に高カーディナリティ、低値フィールドを削除します (古いログの raw スタックトレースなど)。
- 保持ポリシー: コンプライアンス要件に沿った最大保持期間を定義します。データを無期限に保存しないでください。

延長サポートコストの回避 — エンジンバージョンを最新の状態に保つ

Amazon OpenSearch Service は、延長サポートのエンジンバージョンに対して、正規化されたインスタンス時間あたりの定額料金を請求します。

- サポートされていない古いバージョンにとどまると、インスタンスコストに加えて追加料金が発生します。
- 延長サポート料金が発生しないように、現在サポートされているバージョンにアップグレードします。

コスト配分タグと CloudWatch モニタリング

プロアクティブコストガバナンスは無駄を防ぎます。

- OpenSearch ドメインにコスト配分タグを適用して、チームまたはワークロードごとの詳細なコスト追跡を行います。
- ストレージ使用率、JVM プレッシャー、CPU の CloudWatch アラームを設定して、オーバープロビジョニングを早期に検出します。
- 一貫して使用率が低いドメインを特定するには、 AWS Cost Explorer を使用します。
- 自動調整の評価 — JVM ヒープサイズやその他の設定を自動的に調整して、パフォーマンスを向上させ、リソースの無駄を減らします。

Amazon OpenSearch Service Serverless のコスト最適化

ディスク最適化ベクトル検索 (ベクトルコレクション)

ディスク最適化ベクトル検索は、ベクトルワークロードの最も強力なコスト削減手法の 1 つです。圧縮されたベクトルのみを RAM に保持し、完全精度ベクトルをディスクに保存することで、インメモリモードのコストのごく一部でベクトル検索を実行します。

仕組み:

- 標準 (`in_memory`) モードでは、完全な HNSW グラフが RAM にロードされます。RAM は大規模に非常に高価になります。
- `on_disk` モードでは、圧縮 (量子化) ベクトルのみが候補生成のためにメモリに保持されます。完全精度ベクトルは、最終的な再スコアリングフェーズ (2 フェーズ検索) でのみディスクから取得されます。
- これにより、高い検索品質を維持しながら RAM 要件が大幅に削減されます。
- デフォルト`on_disk`モードは 32 倍のバイナリ量子化を使用します。メモリ要件をインメモリモードと比較して 97% 削減します。
- 圧縮レベルをサポート: 2x (FP16)、4x (バイト)、8x、16x、32x (バイナリ)。
- 100 ~ 200 ミリ秒の範囲の P90 レイテンシー — 1 桁ミリ秒の応答時間を必要としないワークロードに適しています。

コスト削減ベンチマーク:

データセット	リコール@100	P90 レイテンシー	コスト削減
Cohere TREC-RAG	0.94	104 ミリ秒	83%
Tasb-1M	0.96	7 ミリ秒	67%
Marco-1M	0.99	7 ミリ秒	67%

最適: RAG パイプライン、セマンティック検索、ドキュメント取得、および P90 レイテンシーが 100 ~ 200 ミリ秒でコスト削減が優先されるベクトルワークロード。

> **Note**
>
> この変更を既存のインデックス付きデータに適用するには、インデックスを再作成する必要があります。などの外部パイプラインツールを使用して、データを新しいインデックスに再インデックスできます。

詳細については、Disk-Optimized Vector Search ブログ記事、Quantization Techniques ブログ記事、および を参照してくださいthe section called “ベクトル検索コレクションの使用”。

Vector Auto-Optimize (ベクトルコレクション)

自動最適化は、ベクトルインデックス設定を自動的に評価し、ベクトルの専門知識を必要とせずに、検索品質、レイテンシー、メモリコストの最適なトレードオフを推奨します。

- 最適化に関する推奨事項を 1 時間未満で提供します。
- ベクトル取り込みパイプラインと統合されています。
- 数十億規模のベクトルデータベースの GPU アクセラレーションインデックス作成と組み合わせることができます。

詳細については、「Auto-Optimize」ブログ記事を参照してください。

GPU アクセラレーションベクトルインデックス作成 (ベクトルコレクション)

GPU アクセラレーションは、HNSW ベクトルインデックス作成をサーバーレス GPUs、大きなベクトルインデックスを構築する時間と OCU コストを大幅に削減します。

- CPU のみのインデックス作成と比較して、インデックスの構築時間が 6.4 倍から 13.8 倍高速になります。
- 書き込みが多いベクトルワークロードのインデックス作成 OCU コストを最大 75% 削減します。
- GPUsは動的にアタッチされます。GPU アクセラレーションがアクティブな場合にのみ OCUs に対して料金が発生します。
- 1 時間以内に 10 億規模のベクトルデータベースを構築できます。
- Vector Acceleration OCUs。

最適: 大規模な初期ベクトル取り込み、またはインデックスの再構築にコストがかかる頻繁なモデル再トレーニングシナリオ。

詳細については、GPU アクセラレーションのブログ記事を参照してください。

最大 OCU 制限を設定する (すべてのコレクションタイプ)

Amazon OpenSearch Service Serverless は、需要に基づいて OCUs を自動スケーリングします。上限がないと、コストが予期せず急増する可能性があります。ランナウェイスケーリングを防ぐために、アカウントレベルまたはコレクショングループごとに最大 OCU 制限を設定します。システムはピークロード中にこの制限までスケールアップしますが、それを超えることはありません。

許可される値:

- アカウントレベル: 最大 1,700 OCUs (16 の倍数に制限されません)。
- コレクショングループ: 1、2、4、8、16、および 16 ~ 1,696 OCUs。

CloudWatch メトリクス (`OCUUtilization`) をモニタリングして、時間の経過とともに最大 OCU 設定のサイズを適正化します。

> **Note**
>
> 使用率が最大 OCU 上限に達すると、コストが含まれていてもパフォーマンスが大幅に低下する可能性があります。上限に達しても、OCU スパイクの根本的な原因は解決されません。ベクトルコレクションの場合、根本原因は通常インメモリベクトルです。これは、ベクトルインデックス作成の最適化、インデックスサイズの縮小、リコールと精度のトレードオフの調整によって直接対処する必要があります。

> **Tip**
>
> 保守的な最大 OCU から開始し、CloudWatch が上限付近で持続的な使用率を示している場合にのみ増加します。常に上限に達した場合は、単に制限を引き上げるのではなく、根本原因、特にベクトルコレクションのインメモリベクトルの使用を調査します。

詳細については、「the section called “キャパシティ制限の管理”」を参照してください。

保持期間の最適化 (時系列コレクション)

データライフサイクルポリシーは、指定された保持期間後に時系列コレクションからデータを自動的に削除し、無制限のストレージの増加を防ぎます。時系列コレクションのみがデータライフサイクルポリシーをサポートしており、検索コレクションとベクトルコレクションはサポートしていません。

時系列コレクションの OCU 数は、ローカルストレージに保持する必要がある最新のデータの量によって直接決まります。時系列コレクションは、ローカル OCU ストレージ内のデータの最新の部分のみを保持します。古いデータは S3 にオフロードされ、OCUs の数はそれに応じてスケーリングされます。

必要な OCUs = max (最小 OCUs、保持期間内にデータを保持するために必要な OCUs)

データライフサイクルポリシーの設定:

- インデックスまたはインデックスパターンごとに保持期間を 24 時間から 3,650 日に設定します。
- Amazon OpenSearch Service Serverless は、ベストエフォートベース (通常は 48 時間以内、または保持期間の 10% 以内) でデータを自動的に削除します。
- ルールは、コレクションレベル、インデックスパターンレベル、または個々のインデックスレベルで適用できます。より具体的なルールが優先されます。

サイジングの例:

- 1 TiB/日取り込み、保持期間が 30 日 = 約 1 TiB のホットデータ = インデックス作成の場合は 20 OCUs、検索の場合は 20 OCUs。
- 7 日間の保持期間に短縮 = 約 233 GiB のホットデータ = インデックス作成の場合は約 4 OCUs、検索の場合は 4 OCUs。

保持期間が短いほど、ローカルストレージのホットデータが少なくなり、必要な OCUsが少なくなり、コンピューティング料金が低くなります。保持期間を実際のビジネス要件とコンプライアンス要件に合わせる — デフォルトでは、データを無期限に保持しないでください。

詳細については、「the section called “データライフサイクルポリシーを使用する”」を参照してください。

不要なデータ (すべてのコレクションタイプ) を保存しない

直接取り込まれるデータの量を減らすと、コンピューティング (OCUs) とストレージの両方のコストが削減されます。

- 取り込み時にフィールドをフィルタリングする: パイプラインを使用して、コレクションに到達する前に低値のフィールドを削除します。
- 重複データや冗長データを取り込まないでください: パイプラインレベルで重複排除します。
- 適切なインデックスマッピングを使用する: 保存されているが検索されていないフィールドでインデックス作成を無効にします (`"index": false`)。
- 検索コレクションの場合: 検索値なしでストレージを膨張させる大きなバイナリ BLOB や raw テキストを保存しないでください。

マルチテナントワークロードのコレクショングループ (すべてのコレクションタイプ)

コレクショングループを使用すると、異なる KMS キーを持つ複数のコレクションが同じセキュリティ境界内で OCU リソースを共有できるため、マルチテナントアーキテクチャのコストを大幅に削減できます。テナントごとまたはコレクションごとに複数の KMS キーを使用するお客様に適用されます。

- 以前は、各一意の KMS キーには専用 OCUs。これにより、テナントあたりの分離コストが大幅に高くなります。
- コレクショングループを使用すると、個別の暗号化キーを持つテナントは OCU 容量を共有できます。
- 多数の小規模なテナントワークロードで最大 90% のコスト削減。
- 最小 OCUs (保証されたベースライン、コールドスタートなし) と最大 OCUs (コスト上限) の両方をサポートします。
- 異なるネットワークアクセスタイプ (パブリックと VPC) のコレクションは、同じグループに共存できます。
- CloudWatch メトリクスは、リソースの消費量とレイテンシーをグループごとに可視化します。

最適: SaaS プロバイダー、マルチテナントプラットフォーム、または多数の小さなコレクションを持つワークロードで、それぞれに独自の KMS キーが必要です。

詳細については、コレクショングループのブログ記事、最新情報のお知らせ、および を参照してくださいthe section called “コレクショングループ”。

Amazon OpenSearch Service でサポートされるインスタンスタイプ

Amazon OpenSearch Service では、次のインスタンスタイプがサポートされています。すべてのリージョンで、すべてのインスタンスタイプがサポートされているわけではありません。可用性に関する詳細については、「Amazon OpenSearch Service の料金」を参照してください。

お客様のユースケースに適切なインスタンスタイプ関する情報については the section called “ドメインのサイジング”、the section called “EBS ボリュームサイズのクォータ”、および the section called “ネットワークのクォータ” を参照してください。

次のインスタンスファミリーは VPC 暗号化コントロールをサポートしていないことに注意してください。T2、T3、R3、I3、C4、M4、R4、C5、M5、R5、C6g、M6g、R6g、および R6gd。VPC 暗号化コントロールは、トラフィックフローの暗号化ステータスをモニタリングするための一元的な権限制御を提供するセキュリティおよびコンプライアンス機能です。これにより、クリアテキスト通信が可能なリソースを特定し、最終的にはリージョンでの VPC 内での転送、および複数の VPC 間での転送時に暗号化を強制するメカニズムを提供できるようになります。詳細については、VPC 暗号化コントロールのドキュメントを参照してください。

現行世代のインスタンスタイプ

最適なパフォーマンスを得るために、新しい OpenSearch Service ドメインを作成するときには、以下のインスタンスタイプを使用することをお勧めします。

インスタンスタイプ	インスタンス	制限事項
i4i	i4i.large.search i4i.xlarge.search i4i.2xlarge.search i4i.4xlarge.search	i4i インスタンスタイプには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要であり、EBS ボリュームストレージはサポートされません。

インスタンスタイプ	インスタンス	制限事項
	i4i.8xlarge.search i4i.12xlarge.search i4i.16xlarge.search i4i.24xlarge.search i4i.32xlarge.search	
i7i	i7i.large.search i7i.xlarge.search i7i.2xlarge.search i7i.4xlarge.search i7i.8xlarge.search i7i.12xlarge.search i7i.16xlarge.search	i7i インスタンスタイプには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要であり、EBS ボリュームストレージはサポートされていません。

インスタンスタイプ	インスタンス	制限事項
i8g	i8g.large.search i8g.xlarge.search i8g.2xlarge.search i8g.4xlarge.search i8g.8xlarge.search i8g.12xlarge.search i8g.16xlarge.search	i8g インスタンスタイプには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要であり、EBS ストレージボリュームをサポートしていません。

インスタンスタイプ	インスタンス	制限事項
i8ge	i8ge.large.search i8ge.xlarge.search i8ge.2xlarge.search i8ge.3xlarge.search i8ge.6xlarge.search i8ge.12xlarge.search i8ge.18xlarge.search	i8ge インスタンスタイプには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要であり、EBS ストレージボリュームをサポートしていません。

インスタンスタイプ	インスタンス	制限事項
Graviton3	c7g.large.search c7g.xlarge.search c7g.2xlarge.search c7g.4xlarge.search c7g.8xlarge.search c7g.12xlarge.search c7g.16xlarge.search m7g.medium.search m7g.large.search m7g.xlarge.search m7g.2xlarge.search m7g.4xlarge.search	Graviton3 は gp3 ストレージのみをサポートします。Graviton3 インスタンス (c7g、m7g、r7g) は、すべての OpenSearch バージョンと Elasticsearch バージョン 7.9 および 7.10 をサポートしています。

インスタンスタイプ	インスタンス	制限事項
	m7g.8xlarge.search	
	m7g.12xlarge.search	
	m7g.16xlarge.search	
	r7g.medium.search	
	r7g.large.search	
	r7g.xlarge.search	
	r7g.2xlarge.search	
	r7g.4xlarge.search	
	r7g.8xlarge.search	
	r7g.12xlarge.search	
	r7g.16xlarge.search	
	r7gd.large.search	
	r7gd.xlarge.search	

インスタンスタイプ	インスタンス	制限事項
	r7gd.2xlarge.search	
	r7gd.4xlarge.search	
	r7gd.8xlarge.search	
	r7gd.12xlarge.search	
	r7gd.16xlarge.search	

インスタンスタイプ	インスタンス	制限事項
OR1	or1.medium.search or1.large.search or1.xlarge.search or1.2xlarge.search or1.4xlarge.search or1.8xlarge.search or1.12xlarge.search or1.16xlarge.search	• OR1 インスタンスタイプには OpenSearch 2.11 以降が必要です。 • OR1 インスタンスは、他の Graviton インスタンスタイプのマスターノード (C6g、M6g、R6g) とのみ互換性があります。

インスタンスタイプ	インスタンス	制限事項
OR2	or2.medium.search or2.large.search or2.xlarge.search or2.2xlarge.search or2.4xlarge.search or2.8xlarge.search or2.12xlarge.search or2.16xlarge.search	• OR2 および OM2 インスタンスタイプには OpenSearch 2.11 以降が必要で、Graviton ベースのマスターノードインスタンスタイプ (C6g、M6g、R6g) とのみ互換性があります。

インスタンスタイプ	インスタンス	制限事項
OM2	om2.large.search om2.xlarge.search om2.2xlarge.search om2.4xlarge.search om2.8xlarge.search om2.12xlarge.search om2.16xlarge.search	• OR2 および OM2 インスタンスタイプには OpenSearch 2.11 以降が必要で、Graviton ベースのマスターノードインスタンスタイプ (C6g、M6g、R6g) とのみ互換性があります。

インスタンスタイプ	インスタンス	制限事項
OI2	oi2.large.search oi2.xlarge.search oi2.2xlarge.search oi2.4xlarge.search oi2.8xlarge.search oi2.12xlarge.search oi2.16xlarge.search oi2.24xlarge.search oi2.32xlarge.search	• OI2 インスタンスタイプには OpenSearch 2.11 以降が必要です。 • これらのインスタンスは NVMe ストレージでプロビジョニングされ、個別の EBS ストレージは必要ありません。 • これらのインスタンス (最大 8xlarge) は、OpenSearch 3.3 以降のウォームノードとしても使用できます。

インスタンスタイプ	インスタンス	制限事項

Im4gn	`im4gn.large.search`
	`im4gn.xlarge.search`
	`im4gn.2xlarge.search`
	`im4gn.4xlarge.search`
	`im4gn.8xlarge.search`
	`im4gn.16xlarge.search`
		• Im4gn インスタンスタイプには、Elasticsearch バージョン 7.9 以降または任意の OpenSearch バージョンが必要であり、EBS ストレージボリュームはサポートされません。
		• Im4gn インスタンスは、他の Graviton インスタンスタイプ (C6g、M6g、R6g、R6gd) とのみ互換性があります。Graviton インスタンスと非 Graviton インスタンスは、同じクラスター内で結合できません。

T3	`t3.small.search` `t3.medium.search`	• T3 インスタンスタイプには、Elasticsearch 5.6 以降または任意のバージョンの OpenSearch が必要です。 • T3 インスタンスタイプは、ドメインがスタンバイなしでプロビジョニングされている場合にのみ使用できます。詳細については、「the section called “Multi-AZ without Standby”」を参照してください。 • T3 インスタンスタイプは、ドメインのインスタンス数が 10 以下の場合のみ使用できます。 • T3 インスタンスタイプでは、UltraWarm ストレージ、コールドストレージ、または Auto-Tune がサポートされません。

インスタンスタイプ	インスタンス	制限事項
c7i	c7i.large.search c7i.xlarge.search c7i.2xlarge.search c7i.4xlarge.search c7i.8xlarge.search c7i.12xlarge.search c7i.16xlarge.search	• c7i インスタンスには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
m7i	m7i.large.search m7i.xlarge.search m7i.2xlarge.search m7i.4xlarge.search m7i.8xlarge.search m7i.12xlarge.search m7i.16xlarge.search	• m7i インスタンスには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
r7i	r7i.large.search r7i.xlarge.search r7i.2xlarge.search r7i.4xlarge.search r7i.8xlarge.search r7i.12xlarge.search r7i.16xlarge.search	• r7i インスタンスには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
c8g	c8g.large.search c8g.xlarge.search c8g.2xlarge.search c8g.4xlarge.search c8g.8xlarge.search c8g.12xlarge.search c8g.16xlarge.search	• c8g インスタンスには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
m8g	m8g.medium.search m8g.large.search m8g.xlarge.search m8g.2xlarge.search m8g.4xlarge.search m8g.8xlarge.search m8g.12xlarge.search m8g.16xlarge.search	• m8g インスタンスには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
r8g	r8g.medium.search r8g.large.search r8g.xlarge.search r8g.2xlarge.search r8g.4xlarge.search r8g.8xlarge.search r8g.12xlarge.search r8g.16xlarge.search	• r8g インスタンスには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要であり、GP3 ストレージボリュームのみサポートされます。

インスタンスタイプ	インスタンス	制限事項
r8gd	r8gd.medium.search r8gd.large.search r8gd.2xlarge.search r8gd.4xlarge.search r8gd.8xlarge.search r8gd.12xlarge.search r8gd.16xlarge.search	• r8gd インスタンスタイプには、Elasticsearch バージョン 7.9 以降または任意のバージョンの OpenSearch が必要であり、EBS ストレージボリュームはサポートされません。

旧世代のインスタンスタイプ

アプリケーションが旧世代のインスタンスタイプ用に最適化されており、アップグレードはまだこれからというユーザー向けに、OpenSearch Service では、それらの旧世代のインスタンスを提供しています。最高のパフォーマンスを得るには、現世代のインスタンスタイプの使用をお勧めしますが、以下の旧世代のインスタンスタイプも引き続きサポートします。

インスタンスタイプ	インスタンス	制限事項
C4	c4.large.search c4.xlarge.search c4.2xlarge.search c4.4xlarge.search c4.8xlarge.search	
I2	i2.xlarge.search i2.2xlarge.search	
M3	m3.medium.search m3.large.search m3.xlarge.search m3.2xlarge.search	• M3 インスタンスタイプでは、保管時のデータの暗号化、きめ細かなアクセス制御、クラスター間検索がサポートされません。 • OpenSearch バージョンによっては、M3 インスタンスタイプに追加の制限があります。詳細については the section called “無効な M3 インスタンスタイプ”を参照してください。
M4	m4.large.search	

インスタンスタイプ	インスタンス	制限事項
	m4.xlarge.search m4.2xlarge.search m4.4xlarge.search m4.10xlarge.search	
R3	r3.large.search r3.xlarge.search r3.2xlarge.search r3.4xlarge.search r3.8xlarge.search	R3 インスタンスタイプは、保管時のデータの暗号化またはきめ細かなアクセスコントロールをサポートしていません。

インスタンスタイプ	インスタンス	制限事項
R4	r4.large.search r4.xlarge.search r4.2xlarge.search r4.4xlarge.search r4.8xlarge.search r4.16xlarge.search	
i4g	i4g.large.search i4g.xlarge.search i4g.2xlarge.search i4g.4xlarge.search i4g.8xlarge.search i4g.16xlarge.search	i4g インスタンスタイプには、Elasticsearch バージョン 7.9 以降または任意の OpenSearch バージョンが必要であり、EBS ストレージボリュームはサポートされません。

インスタンスタイプ	インスタンス	制限事項
C5	c5.large.search c5.xlarge.search c5.2xlarge.search c5.4xlarge.search c5.9xlarge.search c5.18xlarge.search	C5 インスタンスタイプには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要です。

インスタンスタイプ	インスタンス	制限事項
C6g	c6g.large.search c6g.xlarge.search c6g.2xlarge.search c6g.4xlarge.search c6g.8xlarge.search c6g.12xlarge.search	• C6g インスタンスタイプには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要です。 • C6g インスタンスは、他の Graviton インスタンスタイプ (Im4gn、M6g、R6g、R6gd) とのみ互換性があります。Graviton インスタンスと非 Graviton インスタンスは、同じクラスター内で結合できません。

インスタンスタイプ	インスタンス	制限事項
I3	i3.large.search i3.xlarge.search i3.2xlarge.search i3.4xlarge.search i3.8xlarge.search i3.16xlarge.search	
M5	m5.large.search m5.xlarge.search m5.2xlarge.search m5.4xlarge.search m5.12xlarge.search	M5 インスタンスタイプには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要です。

インスタンスタイプ	インスタンス	制限事項

M6g	`m6g.large.search`
	`m6g.xlarge.search`
	`m6g.2xlarge.search`
	`m6g.4xlarge.search`
	`m6g.8xlarge.search`
	`m6g.12xlarge.search`
		• M6g インスタンスタイプには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要です。
		• M6g インスタンスは、他の Graviton インスタンスタイプ (Im4gn、C6g、R6g、R6gd) とのみ互換性があります。Graviton インスタンスと非 Graviton インスタンスは、同じクラスター内で結合できません。

インスタンスタイプ	インスタンス	制限事項
R5	r5.large.search r5.xlarge.search r5.2xlarge.search r5.4xlarge.search r5.12xlarge.search	R5 インスタンスタイプには、Elasticsearch 5.1 以降または任意のバージョンの OpenSearch が必要です。

インスタンスタイプ	インスタンス	制限事項
R6g	r6g.large.search r6g.xlarge.search r6g.2xlarge.search r6g.4xlarge.search r6g.8xlarge.search r6g.12xlarge.search	• R6g インスタンスタイプには、Elasticsearch 7.9 以降または任意のバージョンの OpenSearch が必要です。 • R6g インスタンスは、他の Graviton インスタンスタイプ (Im4gn、C6g、M6g、R6gd) とのみ互換性があります。Graviton インスタンスと非 Graviton インスタンスは、同じクラスター内で結合できません。

インスタンスタイプ	インスタンス	制限事項
R6gd	r6gd.large.search r6gd.xlarge.search r6gd.2xlarge.search r6gd.4xlarge.search r6gd.8xlarge.search r6gd.12xlarge.search r6gd.16xlarge.search	• R6gd インスタンスタイプには、Elasticsearch バージョン 7.9 以降または OpenSearch の任意のバージョンが必要であり、EBS ストレージボリュームはサポートされません。 • R6gd インスタンスは、他の Graviton インスタンスタイプ (Im4gn、C6g、M6g、R6g) とのみ互換性があります。Graviton インスタンスと非 Graviton インスタンスは、同じクラスター内で結合できません。
T2	t2.micro.search t2.small.search t2.medium.search	• T2 インスタンスタイプは、ドメインのインスタンス数が 10 以下の場合のみ使用できます。 • t2.micro.search インスタンスタイプは、Elasticsearch 1.5 および 2.3 のみをサポートします。 • T2 インスタンスタイプでは、保管中のデータの暗号化、きめ細かなアクセス制御、UltraWarm ストレージ、コールドストレージ、クラスター間検索、または Auto-Tune がサポートされません。

> **Tip**
>
> 専用マスターノードおよびデータノードの異なるインスタンスタイプをお勧めする場合があります。

Amazon OpenSearch Service のエンジンバージョン別の機能

OpenSearch サービスの多くの機能には、最小の OpenSearch バージョン要件または従来の Elasticsearch OSS バージョン要件があります。機能の最小バージョンを満たしていても、その機能がドメインで利用できない場合は、ドメインのサービスソフトウェアを更新します。

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
多層アーキテクチャ	3.3	含まれない
派生ソース	3.1	含まれない
スターツリーインデックス	2.17	含まれない
Amazon Q のサポート	2.17	含まれない
カスタムプラグイン	2.15	含まれない
同時セグメント検索	2.13	含まれない
自然言語クエリの生成	2.13	含まれない
マルチモーダルセマン	2.11	含まれない

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
ティック検索		
直接クエリデータソース	2.11	含まれない
機械学習コネクタ	2.9	含まれない
セマンティック検索	2.9	含まれない
ポイントインタイム検索	2.5	含まれない
通知	2.3	含まれない
ML Commons	1.3	含まれない
クラスター間レプリケーション	1.1	7.10
インデックス変換	1.0	含まれない
専用コーディネーターノード	1.0	6.8
VPC サポート	1.0	1.0

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
ドメインへのすべてのトラフィックに HTTPS を要求する		
マルチ AZ のサポート		
専用マスターノード		
カスタムパッケージ		
カスタムエンドポイント		
スローログの公開		
エラーログの公開	1.0	5.1
保管中のデータの暗号化		
OpenSearch Dashboards のCognito 認証		

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
インプレースアップグレード		
時間単位の自動スナップショット	1.0	5.3
ノード間の暗号化	1.0	6.0
Java 高レベル REST クライアントのサポート		
HTTP リクエストとレスポンス圧縮		
[アラート]	1.0	6.2
SQL	1.0	6.5
クラスター間検索	1.0	6.7
きめ細かなアクセスコントロール		

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
OpenSearch Dashboards の SAML 認証		
Auto-Tune		
リモート再インデックス		
UltraWarm	1.0	6.8
インデックスステート管理		
K-NN (ユークリッド距離)	1.0	7.1
異常検出	1.0	7.4
コサイン類似度による k-NN	1.0	7.7
Learning to Rank		
パイプ処理言語	1.0	7.9

機能	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
OpenSearch Dashboards のレポート		
OpenSearch Dashboards のトレース分析		
ARM ベースの Graviton インスタンス		
コールドストレージ		
ハミング距離、L1 ノルム距離、k-NN 用 Painless スクリプティング	1.0	7.10
非同期検索		
Curator のサポート	含まれない	5.1

これらの機能の一部と追加の機能を有効にするプラグインについては、「the section called “エンジンバージョンに応じたプラグイン”」を参照してください。各バージョンの OpenSearch API の詳細については、「the section called “サポートされているオペレーション”」を参照してください。

Amazon OpenSearch Service のエンジンバージョンに応じたプラグイン

Amazon OpenSearch Service ドメインには、OpenSearch コミュニティからのプラグインがパッケージされています。サービスはプラグインを自動的にデプロイおよび管理しますが、ドメインに選択した OpenSearch または従来の Elasticsearch OSS のバージョンに応じて異なるプラグインをデプロイします。

以下の表は、OpenSearch のバージョン別のプラグインと、従来の Elasticsearch OSS の互換バージョンを一覧表示しています。このプラグインには、対話する可能性のあるプラグインのみが含まれており、包括的ではありません。OpenSearch Service は追加のプラグインを使用して、スナップショット用の S3 リポジトリプラグインや、最適化とモニタリング用の OpenSearch Performance Analyzer プラグインなど、コアサービス機能を有効にします。ドメインで実行されているすべてのプラグインの完全なリストについては、次のリクエストを実行します。

```
GET _cat/plugins?v
```

プラグイン	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
HanLP	2.11	サポートされていません
ヘブライ語分析	2.11	サポートされていません
Amazon Personalize 検索ランキング	2.9	サポートされていません
Neural Search	2.9	サポートされていません
セキュリティ分析	2.5	サポートされていません
OpenSearch 通知	2.3	サポートされていません

プラグイン	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
ML Commons	1.3	サポートされていません
Sudachi 分析 (日本語に推奨)	1.3	サポートされていません
STConvert	1.3	サポートされていません
Pinyin 分析	1.3	サポートされていません
Nori 分析	1.3	サポートされていません
OpenSearch の観測性	1.2	サポートされていません
OpenSearch クラスター間レプリケーション	1.1	7.10
OpenSearch の非同期検索	1.0	7.10
IK (中国語) 分析	1.0	7.7
ベトナム語分析		
タイ語分析		
Learning to Rank		

プラグイン	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
OpenSearch の異常検出	1.0	7.4
OpenSearch k-NN	1.0	7.1
OpenSearch のインデックス状態管理	1.0	6.8
OpenSearch のセキュリティ	1.0	6.7
OpenSearch SQL	1.0	6.5
OpenSearch のアラート設定	1.0	6.2
Ukrainian Analysis	1.0	5.3
Mapper Size	1.0	5.3
Mapper Murmur3	1.0	5.1
Ingest User Agent Processor	1.0	5.1

プラグイン	最低限必要な OpenSearch バージョン	最低限必要な Elasticsearch バージョン
Ingest Attachment Processor	1.0	5.1
Stempel Polish Analysis	1.0	5.1
Smart Chinese Analysis	1.0	5.1
Seunjeon 韓国語分析	1.0	5.1
Phonetic Analysis	1.0	2.3
Japanese (kuromoji) Analysis	1.0	すべてのドメインで含まれる
ICU Analysis	1.0	すべてのドメインで含まれる

オプションプラグイン

Amazon OpenSearch Service は、プリインストールされているデフォルトのプラグインに加えて、いくつかのオプションの言語アナライザープラグインをサポートしています。 AWS マネジメントコンソール および を使用して AWS CLI 、プラグインをドメインに関連付けたり、プラグインをドメインから関連付け解除したり、すべてのプラグインを一覧表示したりできます。オプションのプラグインパッケージは特定の OpenSearch バージョンと互換性があり、そのバージョンのドメインにのみ関連付けることができます。

Sudachi プラグイン の場合、辞書ファイルを再関連付けしても、すぐにドメインに反映されないことに注意してください。設定変更やその他の更新の一環として、次のブルー/グリーンデプロイがドメインで実行されると、辞書が更新されます。または、更新済みのデータで新しいインデックスを作成し、この新しいパッケージを使用して新しいインデックスを作成し、既存のインデックスを新しいインデックスに再インデックスしてから、古いインデックスを削除することもできます。インデックスの再作成方法を使用する場合は、トラフィックが中断されないようにインデックスエイリアスを使用してください。

オプションプラグインは ZIP-PLUGIN パッケージタイプを使用します。オプションのプラグインについての詳細は、「the section called “パッケージ”」を参照してください。

Amazon OpenSearch Service でサポートされているオペレーション

OpenSearch Service は多くのバージョンの OpenSearch と従来の Elasticsearch OSS をサポートしています。以下のセクションでは、各バージョンで OpenSearch Service がサポートするオペレーションを示します。

トピック

- API の重要な相違点

API の重要な相違点

新しい List API

多数のインデックスとシャードを持つ大規模なクラスターをサポートするために、_list/indices や _list/shards など、ページ分割をサポートする新しい List API が導入されました。List API は、インデックスとシャードに関する統計をページ分割形式で取得します。これにより、多くのインデックスを含むレスポンスを処理するタスクが効率化されます。

- `_list/indices`: _list/indices
- `_list/shards`: _list/shards

既存の API の変更

大規模なクラスターをサポートするために、`_cluster/stats` API へのメトリクスフィルターの追加に対するサポートを追加しました。これにより `_cluster/stats/<metric>/nodes/<node-`

filters> や _cluster/stats/<metric>/<index_metric>/nodes/<node-filters> など、関連する統計レスポンスのみを取得することができます。詳細については、「_cluster/stats」を参照してください。

cancel_after_time_interval リクエストパラメータの指定によるタスクキャンセルのサポートを _cat/shards API に追加しました。詳細については、「_cat/shards」を参照してください。

_cat API のレスポンスサイズの制限

データノードとウォームノードの合計インスタンス数が 200 を超える大規模なクラスターをサポートするため、_cat/segments API が返すインデックス数に 10,000 個 の上限を設けています。レスポンスのインデックス数がこの上限を超えると、API は 429 エラーを返します。これを回避するには、_cat/segments/<index-pattern> などのインデックスパターンフィルターをクエリ内で指定してください。

設定と統計

OpenSearch Service は、「フラットな」設定フォームを使用する _cluster/settings API への PUT リクエストのみを受け入れます。拡張設定フォームを使用するリクエストは拒否します。

```
// Accepted
PUT _cluster/settings
{
  "persistent" : {
    "action.auto_create_index" : false
  }
}

// Rejected
PUT _cluster/settings
{
  "persistent": {
    "action": {
      "auto_create_index": false
    }
  }
}
```

高レベル Java REST クライアントは拡張フォームを使用するため、設定リクエストを送信する必要がある場合は、低レベルクライアントを使用します。

Elasticsearch 5.3 まで、OpenSearch Service ドメインに対する _cluster/settings API では、HTTP PUT メソッドのみがサポートされており、GET メソッドはサポートされていませんでした。OpenSearch およびより新しいバージョンの Elasticsearch では、以下の例に示すように、GET メソッドもサポートされるようになっています。

```
GET https://domain-name.region.es.amazonaws.com/_cluster/settings?pretty
```

戻り値の例を次に示します。

```
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation": {
          "cluster_concurrent_rebalance": "2",
          "node_concurrent_recoveries": "2",
          "disk": {
            "watermark": {
              "low": "1.35gb",
              "flood_stage": "0.45gb",
              "high": "0.9gb"
            }
          },
          "node_initial_primarirecoveries": "4"
        }
      }
    },
    "indices": {
      "recovery": {
        "max_bytper_sec": "40mb"
      }
    }
  }
}
```

オープンソース OpenSearch クラスターと OpenSearch Service からの応答を特定の設定および統計 API の応答について比較した場合、不足しているフィールドに気付くかもしれません。OpenSearch Service は、サービス内部を公開する特定の情報 (_nodes/stats からのファイルシステムデータパスや、_nodes からのオペレーティングシステム名およびバージョンなど) を編集します。

縮小

_shrink API により、アップグレード、設定変更、ドメインの削除が失敗する場合があります。Elasticsearch バージョン 5.3 または 5.1 を実行するドメインでは、使用しないことをお勧めします。これらのバージョンは収縮させたインデックススナップショットの復元の失敗を引き起こす可能性があるバグがあります。

他の Elasticsearch または OpenSearch バージョンの _shrink API を使用している場合は、縮小操作を開始する前に次のリクエストを作成します。

```
PUT https://domain-name.region.es.amazonaws.com/source-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": "name-of-the-node-to-shrink-to",
    "index.blocks.read_only": true
  }
}
```

縮小操作の完了後に次のリクエストを作成します。

```
PUT https://domain-name.region.es.amazonaws.com/source-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}

PUT https://domain-name.region.es.amazonaws.com/shrunken-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}
```

新しい List API

多数のインデックスとシャードを持つ大規模なクラスターをサポートするために、_list/indices や _list/shards など、ページ分割をサポートする新しい List API が導入されました。List API は、インデックスとシャードに関する統計をページ分割形式で取得します。これにより、多くのイン

デックスを含むレスポンスを処理するタスクが効率化されます。_list/indices の詳細については、「List indices」を参照してください。_list/shards の詳細については、「List shards」を参照してください。

既存の API の変更

大規模なクラスターをサポートするために、_cluster/stats/<metric>/nodes/<node-filters> と _cluster/stats/<metric>/<index_metric>/nodes/<node-filters> にサポートを追加しました。_cluster/stats の詳細については、「Cluster stats」を参照してください。

_cat API のレスポンスサイズの制限

データノードとウォームノードの合計インスタンス数が 200 を超える大規模なクラスターをサポートするため、_cat/segments API が返すインデックス数に 10,000 の上限を設けています。レスポンスのインデックス数がこの上限を超えると、API は 429 エラーを返します。これを回避するには、クエリ内でインデックスパターンフィルター (_cat/segments/<index-pattern> など) を指定できます。

さらに、cancel_after_time_interval リクエストパラメータの指定によるタスクキャンセルのサポートが _cat/shards API で用可能になりました。この詳細については、「CAT shards」を参照してください。

専用マスターノードのインスタンスタイプの選択

以下の表は、専用マスターノードに適したインスタンスタイプを選択する際の推奨事項を示しています。

RAM	サポートされるノードの最大数	サポートされるシャードの最大数
2 GB	10	1,000
4 GB	10	5,000
8 GB	30	15,000
16 GB	60	30,000

RAM	サポートされるノードの最大数	サポートされるシャードの最大数
32 GB	120	60,000
64 GB	240	120,000
128 GB	480	240,000
256 GB	1002	500,000

OpenSearch バージョン 2.19

OpenSearch 2.19 のオペレーションについては、「OpenSearch REST API リファレンス」、または特定のプラグインの API リファレンスを参照してください。このリリースの変更の詳細については、「2.19 リリースノート」を参照してください。

OpenSearch バージョン 2.17

OpenSearch 2.17 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

> **Note**
>
> OpenSearch 2.17 以降、`cluster.max_shards_per_node` 設定を変更することはできません。OpenSearch 2.17 以降では、OpenSearch Service は JVM ヒープメモリ 16 GB 当り 1,000 シャード、最大 1 ノード当り 4,000 シャードをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* `/_forcemerge` 、*/index-name* `/update/`*id*、および */index-name* `/_close` など)
- `/_alias`
- `/_delete_by_query` [1]
- `/_explain`
- `/_field_caps`
- `/_field_stats`
- `/_flush`
- `/_refresh`
- `/_reindex` [1]
- `/_render`
- `/_resolve/index`
- `/_rollover`

- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ[4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.search.request.slowlog.level
  - cluster.search.request.slowlog.threshold.warn
  - cluster.search.request.slowlog.threshold.info
  - cluster.search.request.slowlog.threshold.debug
- /_ingest/pipeline
- /_list
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_scripts [3]
- /_search[2]
- /_search/pipeline
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

  - cluster.search.request.slowlog.threshold.trace
  - search.phase_took_enabled
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

> **Note**
>
> 現在、cluster.max_shards_per_node 設定機能の変更は、スタンバイのマルチ AZ (アベイラビリティーゾーン) をご利用のお客様では有効になっていません。

OpenSearch バージョン 2.15

OpenSearch 2.15 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/pipeline
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.total.limit
- cluster.max_shards_per_node
- cluster.search.request.slowlog.level
- cluster.search.request.slowlog.threshold.warn
- cluster.search.request.slowlog.threshold.info
- cluster.search.request.slowlog.threshold.debug
- cluster.search.request.slowlog.threshold.trace
- search.phase_took_enabled

- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。

3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 2.13

OpenSearch 2.13 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/pipeline
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks

  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
  - cluster.search.request.slowlog.level
  - cluster.search.request.slowlog.threshold.warn
  - cluster.search.request.slowlog.threshold.info
  - cluster.search.request.slowlog.threshold.debug
  - cluster.search.request.slowlog.threshold.trace
  - search.phase_took_enabled
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 2.11

OpenSearch 2.11 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/pipeline
- /_search/point_in_time
- /_search profile
- /_shard_stores

- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_shrink [5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。

4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 2.9

OpenSearch 2.9 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/pipeline
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]

- indices.breaker.fielddata.limit
- indices.breaker.request.limit
- indices.breaker.total.limit
- cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 2.7

OpenSearch 2.7 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.total.limit
- cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Opensearch バージョン 2.5

OpenSearch 2.5 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-*
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_refresh
- /_reindex [1]
- /_render

name /update/*id*、および /*index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ[4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_security_analytics
- /_plugins/_sm
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search/point_in_time
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Opensearch バージョン 2.3

OpenSearch 2.3 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split

- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- _plugins/_notifications
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。

4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Opensearch バージョン 1.3

OpenSearch 1.3 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ml
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.fielddata.limit
- indices.breaker.request.limit
- indices.breaker.total.limit
- cluster.max_shards_per_node

- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 1.2

OpenSearch 1.2 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- /_count
- /_dashboards

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 1.1

OpenSearch 1.1 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores

- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_shrink [5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデ

フォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。

3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

OpenSearch バージョン 1.0

OpenSearch 1.0 では、OpenSearch Service は次のオペレーションをサポートしています。ほとんどのオペレーションについては、OpenSearch REST API リファレンスか、または特定のプラグインの API リファレンスを参照してください。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_plugins/_asynchronous_search
- /_plugins/_alerting
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template

- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_dashboards
- /_plugins/_anomaly_detection
- /_plugins/_ism
- /_plugins/_ppl
- /_plugins/_security
- /_plugins/_sql
- /_plugins/_transforms
- /_percolate
- /_rank_eval
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 7.10

Elasticsearch 7.10 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_index_template [6]
- /_ingest/pipeline
- /_index_template
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_asynchronous_search
- /_opendistro/_anomaly_detection
- /_opendistro/_ism
- /_opendistro/_ppl
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex [1]
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template [6]
- /_update_by_query [1]
- /_validate

- cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count

- /_plugins/_replication
- /_rank_eval

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。
6. 従来のインデックステンプレート (_template) は、Elasticsearch 7.8 で開始する組み合わせ可能なテンプレート (_index_template) によって置き換えられました。組み合わせ可能なテンプレートは、従来のテンプレートよりも優先されます。指定されたインデックスに一致する組み合わせ可能なテンプレートがない場合、従来のテンプレートは引き続き一致して、適用できます。_template オペレーションは OpenSearch および以降のバージョンの Elasticsearch OSS でも動作しますが、2 つのテンプレートタイプに対する GET 呼び出しは異なる結果を返します。

Elasticsearch バージョン 7.9

Elasticsearch 7.9 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (/*index-name* /_forcemerge 、/*index-*

- /_delete_by_query [1]
- /_explain

- /_refresh
- /_reindex [1]

name /update/*id*、および /*index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_field_caps
- /_field_stats
- /_flush
- /_index_template [6]
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_anomaly_detection
- /_opendistro/_ism
- /_opendistro/_ppl
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_render
- /_resolve/index
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template [6]
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 OpenSearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。
6. 従来のインデックステンプレート (_template) は、Elasticsearch 7.8 で開始する組み合わせ可能なテンプレート (_index_template) によって置き換えられました。組み合わせ可能なテンプレートは、従来のテンプレートよりも優先されます。指定されたインデックスに一致する組み合わせ可能なテンプレートがない場合、従来のテンプレートは引き続き一致して、適用できます。_template オペレーションは OpenSearch および以降のバージョンの Elasticsearch OSS でも動作しますが、2 つのテンプレートタイプに対する GET 呼び出しは異なる結果を返します。

Elasticsearch バージョン 7.8

Elasticsearch 7.8 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores

- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_index_template [6]
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_anomaly_detection
- /_opendistro/_ism
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_shrink [5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template [6]
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用

Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。

5. 「the section called “縮小”」を参照してください。
6. 従来のインデックステンプレート (_template) は、Elasticsearch 7.8 で開始する組み合わせ可能なテンプレート (_index_template) によって置き換えられました。組み合わせ可能なテンプレートは、従来のテンプレートよりも優先されます。指定されたインデックスに一致する組み合わせ可能なテンプレートがない場合、従来のテンプレートは引き続き一致して、適用できます。_template オペレーションは OpenSearch および以降のバージョンの Elasticsearch OSS でも動作しますが、2 つのテンプレートタイプに対する GET 呼び出しは異なる結果を返します。

Elasticsearch バージョン 7.7

Elasticsearch 7.7 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-name* /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_ltr
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- action.search.shard_count.limit
- indices.breaker.fielddata.limit
- indices.breaker.request.limit
- indices.breaker.total.limit
- cluster.max_shards_per_node

- /_opendistro/_anomaly_detection
- /_opendistro/_ism
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 7.4

Elasticsearch 7.4 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge 、*/index-*

- /_cluster/state
- /_cluster/stats
- /_count

- /_refresh
- /_reindex [1]
- /_render

name /update/*id*、および */index-name* /_close など)
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_anomaly_detection
- /_opendistro/_ism
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_rollover
- /_scripts [3]
- /_search [2]
- /_search profile
- /_shard_stores
- /_shrink [5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。

2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 7.1

Elasticsearch 7.1 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]

- `/_cluster/settings` を使用する複数のプロパティ [4]:
  - `action.auto_create_index`
  - `action.search.shard_count.limit`
  - `indices.breaker.fielddata.limit`
  - `indices.breaker.request.limit`
  - `indices.breaker.total.limit`
  - `cluster.max_shards_per_node`
- `/_opendistro/_ism`
- `/_opendistro/_security`
- `/_opendistro/_sql`
- `/_percolate`
- `/_plugin/kibana`
- `/_rank_eval`
- `/_validate`

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに `/_tasks` 操作を使用することをお勧めします。
2. メッセージ本文の `/_search/scroll` に対する DELETE リクエストでは、`"Content-Length"` を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。`scroll_id` 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、`scroll_id` 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.8

Elasticsearch 6.8 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_ism
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

• cluster.blocks.read_only		

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.7

Elasticsearch 6.7 では、OpenSearch Service は次のオペレーションをサポートします。

• インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く • /_alias • /_aliases • /_all • /_analyze	• /_cluster/state • /_cluster/stats • /_count • /_delete_by_query [1] • /_explain • /_field_caps • /_field_stats • /_flush • /_ingest/pipeline • /_mapping	• /_refresh • /_reindex [1] • /_render • /_rollover • /_scripts [3] • /_search[2] • /_search profile • /_shard_stores • /_shrink[5] • /_snapshot

- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用

Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。

5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.5

Elasticsearch 6.5 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.request.limit
- indices.breaker.total.limit

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.4

Elasticsearch 6.4 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]

- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用

Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。

5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.3

Elasticsearch 6.3 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.request.limit
- indices.breaker.total.limit

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.2

Elasticsearch 6.2 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]

- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_percolate
- /_plugin/kibana
- /_rank_eval
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用

Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。

5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 6.0

Elasticsearch 6.0 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_scripts [3]
- /_search [2]
- /_search profile
- /_shard_stores
- /_shrink [5]
- /_snapshot
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.request.limit
- indices.breaker.total.limit

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 5.6

Elasticsearch 5.6 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_render
- /_rollover
- /_scripts [3]
- /_search [2]
- /_search profile
- /_shard_stores
- /_shrink [5]
- /_snapshot
- /_stats

- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex [1]
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用

Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。

5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 5.5

Elasticsearch 5.5 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ[4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query[1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex[1]
- /_render
- /_rollover
- /_scripts[3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query[1]
- /_validate

- indices.breaker.request.limit
- indices.breaker.total.limit

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. スクリプトの使用に関する考慮事項については、「the section called “サポートされている他のリソース”」を参照してください。
4. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
5. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 5.3

Elasticsearch 5.3 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_render
- /_rollover
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[4]
- /_snapshot
- /_stats
- /_status

- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ [3]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex [1]
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. PUT メソッドを参照してください。GET メソッドの詳細については、「the section called “API の重要な相違点”」を参照してください。このリストは、OpenSearch Service がサポートする汎用 Elasticsearch オペレーションのみを参照し、異常検出や ISM などのプラグイン固有のサポートされているオペレーションは含まれません。
4. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 5.1

Elasticsearch 5.1 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /update/*id* など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings を使用する複数のプロパティ (PUT のみ):
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query [1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex [1]
- /_render
- /_rollover
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[3]
- /_snapshot
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query [1]
- /_validate

- indices.breaker.total.limit

1. クラスター設定を変更すると、これらの操作が完了する前に中断される可能性があります。リクエストが正常に完了したことを確認するには、これらの操作とともに /_tasks 操作を使用することをお勧めします。
2. メッセージ本文の /_search/scroll に対する DELETE リクエストでは、"Content-Length" を HTTP ヘッダーに指定する必要があります。ほとんどのクライアントでは、このヘッダーがデフォルトで追加されます。scroll_id 値で = 文字に関する問題を回避するには、クエリ文字列ではなくリクエストボディを使用して、scroll_id 値を OpenSearch Service に渡します。
3. 「the section called “縮小”」を参照してください。

Elasticsearch バージョン 2.3

Elasticsearch 2.3 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション (*/index-name* /_forcemerge および */index-name* /_recovery など)、*/index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cache/clear (インデックスのみ)
- /_cat (/_cat/nodeattrs を除く)
- /_cluster/health
- /_cluster/settings を使用する複数のプロパティ (PUT のみ):
  - indices.breaker.fielddata.limit
- /_cluster/stats
- /_count
- /_flush
- /_mapping
- /_mget
- /_msearch
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_render
- /_search
- /_snapshot
- /_stats
- /_status
- /_template

- indices.breaker.request.limit
- indices.breaker.total.limit
- threadpool.get.queue_size
- threadpool.bulk.queue_size
- threadpool.index.queue_size
- threadpool.percolate.queue_size
- threadpool.search.queue_size
- threadpool.suggest.queue_size

Elasticsearch バージョン 1.5

Elasticsearch 1.5 では、OpenSearch Service は次のオペレーションをサポートします。

- インデックスパス内のすべてのオペレーション、/*index-name* /_optimize および /*index-name* /_warmer など、/*index-name* /_close を除く
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat
- /_cluster/health
- /_cluster/settings を使用する複数のプロパティ (PUT のみ):
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_cluster/stats
- /_count
- /_flush
- /_mapping
- /_mget
- /_msearch
- /_nodes
- /_percolate
- /_plugin/kibana
- /_plugin/kibana3
- /_plugin/migration
- /_refresh
- /_search
- /_snapshot
- /_stats
- /_status
- /_template

- threadpool.get.queue_size
- threadpool.bulk.queue_size
- threadpool.index.queue_size
- threadpool.percolate.queue_size
- threadpool.search.queue_size
- threadpool.suggest.queue_size

Amazon OpenSearch Service クォータ

AWS アカウントには、サービスごとに AWS 、以前は制限と呼ばれていたデフォルトのクォータがあります。特に明記していない限り、クォータはリージョン固有です。

OpenSearch サービスのドメインとインスタンス、Amazon OpenSearch Serverless、および Amazon OpenSearch Ingestion のクォータを確認するには、「AWS 全般のリファレンス」の「Amazon OpenSearch Service クォータ」を参照してください。

で OpenSearch Service のクォータを表示するには AWS マネジメントコンソール、Service Quotas コンソールを開きます。ナビゲーションペインで、[AWS services] (サービス) を選択し、[Amazon OpenSearch Service] を選択します。クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト を参照してください。

> **Note**
>
> VPC ドメインで VPC 経由で Egress を有効にする場合は、OpenSearch Service が VPC に作成する Egress Network Interface 用に、選択した各サブネットに追加の IP アドレスを予約します。詳細については、「the section called “制限と考慮事項”」を参照してください。

ウォームノードストレージクォータ

次の表に、ウォームノードインスタンスタイプと、各タイプで使用できるストレージの最大量を示します。OpenSearch Optimized OI2 インスタンスの場合、ローカルストレージの 80% がキャッシュとして利用でき、アドレス可能なウォームストレージの最大数はキャッシュストレージサイズの 5 倍です。

たとえば、OI2 インスタンスに 468 GB のローカルストレージがある場合:

- キャッシュストレージ = 375 GB (ローカルストレージの 80%)
- 最大アドレス可能なウォームストレージ = 1875 GB (5 x 375 GB キャッシュ)

インスタンスタイプ	インスタンスストレージ (GB)	キャッシュサイズ (GB)	最大アドレス可能なウォームストレージ (GB)
oi2.large.search	468	375	1875
oi2.xlarge.search	937	750	3750
oi2.2xlarge.search	1875	1500	7500
oi2.4xlarge.search	3750	3000	15000
oi2.8xlarge.search	7500	6000	30000

UltraWarm ストレージのクォータ

次のテーブルに、UltraWarm インスタンスタイプと、各タイプで使用できるストレージの最大容量を示します。UltraWarm の詳細については、「the section called “UltraWarm ストレージ”」を参照してください。

インスタンスタイプ	最大ストレージ
ultrawarm1.medium.search	1.5 TiB
ultrawarm1.large.search	20 TiB

1 AZ あたりのデータノード数

次の表に、AZ デプロイのデータノードの総数を示します。全体的な限度は各上限あたりのデータノード数を示しており、各タイプで使用できるホットノードとウォームノード両方のノードカウントストレージも含んでいます。

AZ 設定	ホットノードカウント上限	ウォームノードカウント上限	全体的な限度 (ホット + ウォーム)
1 – AZ	334	250	334
2 – AZ	668	500	668
3 – AZ	1002	750	1002

インスタンスファミリー別の合計ノード制限

次の表に、インスタンスファミリー別のノード制限の合計を示します。

インスタンスファミリー	ElasticSearch OpenSearch 2.15 まで	OpenSearch 2.17 以降	デフォルトの制限
T2	10	10	10
T3	10	10	10
M3、C4、M4、R4、C5、M5、R5、I2、I3	10	200	80
Graviton 2、Graviton 3、Graviton 4	200	400	80
C7i、R7i、M7i、i4i、i4g、i8g、i7i、i8ge	200	400	80
OR1.medium.search OR1.large.search OR2.medium.search OR2.large.search OM2.large.search	200	400	80

インスタンスファミリー	ElasticSearch OpenSearch 2.15 まで	OpenSearch 2.17 以降	デフォルトの制限
OI2.large.search			
OR1.xlarge.search 以降 OR2.xlarge.search 以降 OM2.xlarge.search 以降 OI2.xlarge.search 以降	200	1002	80
Ultrawarm1	150	750	150

EBS ボリュームサイズのクォータ

次の表は、OpenSearch Service がサポートする各インスタンスタイプの EBS ボリュームの最小サイズと最大サイズを示しています。どのインスタンスタイプにインスタンスストレージが含まれるか、およびハードウェアのその他の詳細については、「Amazon OpenSearch Service の料金」を参照してください。

- ドメインの作成時に [EBS ボリュームタイプ] で磁気ストレージを選択した場合、最大ボリュームサイズは、t2.small と t2.medium、および磁気ストレージをサポートしていないすべての Graviton インスタンス (M6g、C6g、R6g、R6gd) を除くすべてのインスタンスタイプで 100 GiB になります。次のテーブルに示されている最大サイズを使用する場合は、いずれかの SSD オプションを選択してください。
- 一部の旧世代のインスタンスタイプにはインスタンスストレージが含まれますが、EBS ストレージをサポートしています。これらのインスタンスタイプの 1 つに対して EBS ストレージを選択した場合、ストレージボリュームは加算されません。EBS ボリュームまたはインスタンスストレージを使用できますが、両方を使用することはできません。
- EBS gp3 ボリュームの場合、サポートされる最大制限は 80,000 IOPS と 2,000 MB/秒のスループットです。インスタンスタイプによっては、ドメインの有効な制限が低くなる場合があることに注意してください。インスタンス固有の最大制限については、「インスタンスタイプの制限」を参照してください。

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
t2.micro.search	10 GiB	35 GiB	該当なし
t2.small.search	10 GiB	35 GiB	該当なし
t2.medium.search	10 GiB	35 GiB	該当なし
t3.small.search	10 GiB	100 GiB	100 GiB
t3.medium.search	10 GiB	200 GiB	200 GiB
m3.medium.search	10 GiB	100 GiB	該当なし
m3.large.search	10 GiB	512 GiB	該当なし
m3.xlarge.search	10 GiB	512 GiB	該当なし
m3.2xlarge.search	10 GiB	512 GiB	該当なし
m4.large.search	10 GiB	512 GiB	該当なし
m4.xlarge.search	10 GiB	1 TiB	該当なし
m4.2xlarge.search	10 GiB	1.5 TiB	該当なし
m4.4xlarge.search	10 GiB	1.5 TiB	該当なし
m4.10xlarge.search	10 GiB	1.5 TiB	該当なし
m5.large.search	10 GiB	512 GiB	1 TiB
m5.xlarge.search	10 GiB	1 TiB	2 TiB
m5.2xlarge.search	10 GiB	1.5 TiB	3 TiB
m5.4xlarge.search	10 GiB	3 TiB	6 TiB
m5.12xlarge.search	10 GiB	9 TiB	18 TiB
m6g.large.search	10 GiB	512 GiB	1 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
m6g.xlarge.search	10 GiB	1 TiB	2 TiB
m6g.2xlarge.search	10 GiB	1.5 TiB	3 TiB
m6g.4xlarge.search	10 GiB	3 TiB	6 TiB
m6g.8xlarge.search	10 GiB	6 TiB	12 TiB
m6g.12xlarge.search	10 GiB	9 TiB	18 TiB
c4.large.search	10 GiB	100 GiB	該当なし
c4.xlarge.search	10 GiB	512 GiB	該当なし
c4.2xlarge.search	10 GiB	1 TiB	該当なし
c4.4xlarge.search	10 GiB	1.5 TiB	該当なし
c4.8xlarge.search	10 GiB	1.5 TiB	該当なし
c5.large.search	10 GiB	256 GiB	256 GiB
c5.xlarge.search	10 GiB	512 GiB	512 GiB
c5.2xlarge.search	10 GiB	1 TiB	1 TiB
c5.4xlarge.search	10 GiB	1.5 TiB	1.5 TiB
c5.9xlarge.search	10 GiB	3.5 TiB	3.5 TiB
c5.18xlarge.search	10 GiB	7 TiB	7 TiB
c6g.large.search	10 GiB	256 GiB	256 GiB
c6g.xlarge.search	10 GiB	512 GiB	512 GiB
c6g.2xlarge.search	10 GiB	1 TiB	1 TiB
c6g.4xlarge.search	10 GiB	1.5 TiB	1.5 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
c6g.8xlarge.search	10 GiB	3 TiB	3 TiB
c6g.12xlarge.search	10 GiB	4.5 TiB	4.5 TiB
r3.large.search	10 GiB	512 GiB	該当なし
r3.xlarge.search	10 GiB	512 GiB	該当なし
r3.2xlarge.search	10 GiB	512 GiB	該当なし
r3.4xlarge.search	10 GiB	512 GiB	該当なし
r3.8xlarge.search	10 GiB	512 GiB	該当なし
r4.large.search	10 GiB	1 TiB	該当なし
r4.xlarge.search	10 GiB	1.5 TiB	該当なし
r4.2xlarge.search	10 GiB	1.5 TiB	該当なし
r4.4xlarge.search	10 GiB	1.5 TiB	該当なし
r4.8xlarge.search	10 GiB	1.5 TiB	該当なし
r4.16xlarge.search	10 GiB	1.5 TiB	該当なし
r5.large.search	10 GiB	1 TiB	2 TiB
r5.xlarge.search	10 GiB	1.5 TiB	3 TiB
r5.2xlarge.search	10 GiB	3 TiB	6 TiB
r5.4xlarge.search	10 GiB	6 TiB	12 TiB
r5.12xlarge.search	10 GiB	12 TiB	24 TiB
r6g.large.search	10 GiB	1 TiB	2 TiB
r6g.xlarge.search	10 GiB	1.5 TiB	3 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
r6g.2xlarge.search	10 GiB	3 TiB	6 TiB
r6g.4xlarge.search	10 GiB	6 TiB	12 TiB
r6g.8xlarge.search	10 GiB	8 TiB	16 TiB
r6g.12xlarge.search	10 GiB	12 TiB	24 TiB
r6gd.large.search	該当なし	該当なし	該当なし
r6gd.xlarge.search	該当なし	該当なし	該当なし
r6gd.2xlarge.search	該当なし	該当なし	該当なし
r6gd.4xlarge.search	該当なし	該当なし	該当なし
r6gd.8xlarge.search	該当なし	該当なし	該当なし
r6gd.12xlarge.search	該当なし	該当なし	該当なし
r6gd.16xlarge.search	該当なし	該当なし	該当なし
i2.xlarge.search	10 GiB	512 GiB	該当なし
i2.2xlarge.search	10 GiB	512 GiB	該当なし
i3.large.search	該当なし	該当なし	該当なし
i3.xlarge.search	該当なし	該当なし	該当なし
i3.2xlarge.search	該当なし	該当なし	該当なし
i3.4xlarge.search	該当なし	該当なし	該当なし
i3.8xlarge.search	該当なし	該当なし	該当なし
i3.16xlarge.search	該当なし	該当なし	該当なし
or1.medium.search	20 GiB	該当なし	768 GiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
or1.large.search	20 GiB	該当なし	1532 GiB
or1.xlarge.search	20 GiB	該当なし	3 TiB
or1.2xlarge.search	20 GiB	該当なし	6 TiB
or1.4xlarge.search	20 GiB	該当なし	12 TiB
or1.8xlarge.search	20 GiB	該当なし	16 TiB
or1.12xlarge.search	20 GiB	該当なし	24 TiB
or1.16xlarge.search	20 GiB	該当なし	36 TiB
or2.medium.search	20 GiB	該当なし	768 GiB
or2.large.search	20 GiB	該当なし	1532 GiB
or2.xlarge.search	20 GiB	該当なし	3 TiB
or2.2xlarge.search	20 GiB	該当なし	6 TiB
or2.4xlarge.search	20 GiB	該当なし	12 TiB
or2.8xlarge.search	20 GiB	該当なし	16 TiB
or2.12xlarge.search	20 GiB	該当なし	24 TiB
or2.16xlarge.search	20 GiB	該当なし	36 TiB
om2.large.search	20 GiB	該当なし	768 GiB
om2.xlarge.search	20 GiB	該当なし	2 Tib
om2.2xlarge.search	20 GiB	該当なし	3 Tib
om2.4xlarge.search	20 GiB	該当なし	6 Tib
om2.8xlarge.search	20 GiB	該当なし	12 Tib

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
om2.12xlarge.search	20 GiB	該当なし	18 Tib
om2.16xlarge.search	20 GiB	該当なし	24 Tib
im4gn.large.search	該当なし	該当なし	該当なし
im4gn.xlarge.search	該当なし	該当なし	該当なし
im4gn.2xlarge.search	該当なし	該当なし	該当なし
im4gn.4xlarge.search	該当なし	該当なし	該当なし
im4gn.8xlarge.search	該当なし	該当なし	該当なし
im4gn.16xlarge.search	該当なし	該当なし	該当なし
C7g.large.search	10 GiB	該当なし	256 GiB
C7g.xlarge.search	10 GiB	該当なし	512 GiB
C7g.2xlarge.search	10 GiB	該当なし	1 TiB
C7g.4xlarge.search	10 GiB	該当なし	1.5 TiB
C7g.8xlarge.search	10 GiB	該当なし	3 TiB
C7g.12xlarge.search	10 GiB	該当なし	4.5 TiB
C7g.16xlarge.search	10 GiB	該当なし	6 TiB
M7g.medium.search	10 GiB	該当なし	512 GiB
M7g.large.search	10 GiB	該当なし	768 GiB
M7g.xlarge.search	10 GiB	該当なし	2 TiB
M7g.2xlarge.search	10 GiB	該当なし	3 TiB
M7g.4xlarge.search	10 GiB	該当なし	6 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
M7g.8xlarge.search	10 GiB	該当なし	12 TiB
M7g.12xlarge.search	10 GiB	該当なし	18 TiB
M7g.16xlarge.search	10 GiB	該当なし	24 TiB
R7g.medium.search	10 GiB	該当なし	768 GiB
R7g.large.search	10 GiB	該当なし	1.5 TiB
R7g.xlarge.search	10 GiB	該当なし	3 TiB
R7g.2xlarge.search	10 GiB	該当なし	6 TiB
R7g.4xlarge.search	10 GiB	該当なし	12 TiB
R7g.8xlarge.search	10 GiB	該当なし	16 TiB
R7g.12xlarge.search	10 GiB	該当なし	24 TiB
R7g.16xlarge.search	10 GiB	該当なし	36 TiB
R7gd.large.search	該当なし	該当なし	該当なし
R7gd.xlarge.search	該当なし	該当なし	該当なし
R7gd.2xlarge.search	該当なし	該当なし	該当なし
R7gd.4xlarge.search	該当なし	該当なし	該当なし
R7gd.8xlarge.search	該当なし	該当なし	該当なし
R7gd.12xlarge.search	該当なし	該当なし	該当なし
R7gd.16xlarge.search	該当なし	該当なし	該当なし
i4i.large.search	該当なし	該当なし	該当なし
i4i.xlarge.search	該当なし	該当なし	該当なし

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
i4i.2xlarge.search	該当なし	該当なし	該当なし
i4i.4xlarge.search	該当なし	該当なし	該当なし
i4i.8xlarge.search	該当なし	該当なし	該当なし
i4i.12xlarge.search	該当なし	該当なし	該当なし
i4i.16xlarge.search	該当なし	該当なし	該当なし
i4i.24xlarge.search	該当なし	該当なし	該当なし
i4i.32xlarge.search	該当なし	該当なし	該当なし
i4g.large.search	該当なし	該当なし	該当なし
i4g.xlarge.search	該当なし	該当なし	該当なし
i4g.2xlarge.search	該当なし	該当なし	該当なし
i4g.4xlarge.search	該当なし	該当なし	該当なし
i4g.8xlarge.search	該当なし	該当なし	該当なし
i4g.16xlarge.search	該当なし	該当なし	該当なし
i8g.large.search	該当なし	該当なし	該当なし
i8g.xlarge.search	該当なし	該当なし	該当なし
i8g.2xlarge.search	該当なし	該当なし	該当なし
i8g.4xlarge.search	該当なし	該当なし	該当なし
i8g.8xlarge.search	該当なし	該当なし	該当なし
i8g.12xlarge.search	該当なし	該当なし	該当なし
i8g.16xlarge.search	該当なし	該当なし	該当なし

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
i7i.large.search	該当なし	該当なし	該当なし
i7i.xlarge.search	該当なし	該当なし	該当なし
i7i.2xlarge.search	該当なし	該当なし	該当なし
i7i.4xlarge.search	該当なし	該当なし	該当なし
i7i.8xlarge.search	該当なし	該当なし	該当なし
i7i.12xlarge.search	該当なし	該当なし	該当なし
i7i.16xlarge.search	該当なし	該当なし	該当なし
i8ge.large.search	該当なし	該当なし	該当なし
i8ge.xlarge.search	該当なし	該当なし	該当なし
i8ge.2xlarge.search	該当なし	該当なし	該当なし
i8ge.3xlarge.search	該当なし	該当なし	該当なし
i8ge.6xlarge.search	該当なし	該当なし	該当なし
i8ge.12xlarge.search	該当なし	該当なし	該当なし
i8ge.18xlarge.search	該当なし	該当なし	該当なし
c7i.large.search	10 GiB	該当なし	256 GiB
c7i.xlarge.search	10 GiB	該当なし	512 GiB
c7i.2xlarge.search	10 GiB	該当なし	1 TiB
c7i.4xlarge.search	10 GiB	該当なし	1.5 TiB
c7i.8xlarge.search	10 GiB	該当なし	3 TiB
c7i.12xlarge.search	10 GiB	該当なし	4.5 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
c7i.16xlarge.search	10 GiB	該当なし	6 TiB
m7i.large.search	10 GiB	該当なし	768 GiB
m7i.xlarge.search	10 GiB	該当なし	2 TiB
m7i.2xlarge.search	10 GiB	該当なし	3 TiB
m7i.4xlarge.search	10 GiB	該当なし	6 TiB
m7i.8xlarge.search	10 GiB	該当なし	12 TiB
m7i.12xlarge.search	10 GiB	該当なし	18 TiB
m7i.16xlarge.search	10 GiB	該当なし	24 TiB
r7i.large.search	10 GiB	該当なし	1.5 TiB
r7i.xlarge.search	10 GiB	該当なし	3 TiB
r7i.2xlarge.search	10 GiB	該当なし	6 TiB
r7i.4xlarge.search	10 GiB	該当なし	12 TiB
r7i.8xlarge.search	10 GiB	該当なし	16 TiB
r7i.12xlarge.search	10 GiB	該当なし	24 TiB
r7i.12xlarge.search	10 GiB	該当なし	36 TiB
c8g.large	10 GiB	該当なし	256 GiB
c8g.xlarge	10 GiB	該当なし	512 GiB
c8g.2xlarge	10 GiB	該当なし	1 TiB
c8g.4xlarge	10 GiB	該当なし	1.5 TiB
c8g.8xlarge	10 GiB	該当なし	3 TiB

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
c8g.12xlarge	10 GiB	該当なし	4.5 TiB
c8g.16xlarge	10 GiB	該当なし	6 TiB
m8g.medium	10 GiB	該当なし	512 GiB
m8g.large	10 GiB	該当なし	768 GiB
m8g.xlarge	10 GiB	該当なし	2 TiB
m8g.2xlarge	10 GiB	該当なし	3 TiB
m8g.4xlarge	10 GiB	該当なし	6 TiB
m8g.8xlarge	10 GiB	該当なし	12 TiB
m8g.12xlarge	10 GiB	該当なし	18 TiB
m8g.16xlarge	10 GiB	該当なし	24 TiB
r8g.medium	10 GiB	該当なし	768 GiB
r8g.large	10 GiB	該当なし	1532 GiB
r8g.xlarge	10 GiB	該当なし	3 TiB
r8g.2xlarge	10 GiB	該当なし	6 TiB
r8g.4xlarge	10 GiB	該当なし	12 TiB
r8g.8xlarge	10 GiB	該当なし	16 TiB
r8g.12xlarge	10 GiB	該当なし	24 TiB
r8g.16xlarge	10 GiB	該当なし	36 TiB
r8gd.medium	該当なし	該当なし	該当なし
r8gd.large	該当なし	該当なし	該当なし

インスタンスタイプ	最小 EBS サイズ	最大 EBS サイズ (gp2)	最大 EBS サイズ (gp3)
r8gd.xlarge	該当なし	該当なし	該当なし
r8gd.2xlarge	該当なし	該当なし	該当なし
r8gd.4xlarge	該当なし	該当なし	該当なし
r8gd.8xlarge	該当なし	該当なし	該当なし
r8gd.12xlarge	該当なし	該当なし	該当なし
r8gd.16xlarge	該当なし	該当なし	該当なし
oi2.large.search	該当なし	該当なし	該当なし
oi2.xlarge.search	該当なし	該当なし	該当なし
oi2.2xlarge.search	該当なし	該当なし	該当なし
oi2.4xlarge.search	該当なし	該当なし	該当なし
oi2.8xlarge.search	該当なし	該当なし	該当なし
oi2.12xlarge.search	該当なし	該当なし	該当なし
oi2.16xlarge.search	該当なし	該当なし	該当なし

ネットワークのクォータ

次の表は、HTTP リクエストペイロードの最大サイズを示しています。

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
t2.micro.search	10 MiB
t2.small.search	10 MiB
t2.medium.search	10 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
t3.small.search	10 MiB
t3.medium.search	10 MiB
m3.medium.search	10 MiB
m3.large.search	10 MiB
m3.xlarge.search	100 MiB
m3.2xlarge.search	100 MiB
m4.large.search	10 MiB
m4.xlarge.search	100 MiB
m4.2xlarge.search	100 MiB
m4.4xlarge.search	100 MiB
m4.10xlarge.search	100 MiB
m5.large.search	10 MiB
m5.xlarge.search	100 MiB
m5.2xlarge.search	100 MiB
m5.4xlarge.search	100 MiB
m5.12xlarge.search	100 MiB
m6g.large.search	10 MiB
m6g.xlarge.search	100 MiB
m6g.2xlarge.search	100 MiB
m6g.4xlarge.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
m6g.8xlarge.search	100 MiB
m6g.12xlarge.search	100 MiB
c4.large.search	10 MiB
c4.xlarge.search	100 MiB
c4.2xlarge.search	100 MiB
c4.4xlarge.search	100 MiB
c4.8xlarge.search	100 MiB
c5.large.search	10 MiB
c5.xlarge.search	100 MiB
c5.2xlarge.search	100 MiB
c5.4xlarge.search	100 MiB
c5.9xlarge.search	100 MiB
c5.18xlarge.search	100 MiB
c6g.large.search	10 MiB
c6g.xlarge.search	100 MiB
c6g.2xlarge.search	100 MiB
c6g.4xlarge.search	100 MiB
c6g.8xlarge.search	100 MiB
c6g.12xlarge.search	100 MiB
r3.large.search	10 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
r3.xlarge.search	100 MiB
r3.2xlarge.search	100 MiB
r3.4xlarge.search	100 MiB
r3.8xlarge.search	100 MiB
r4.large.search	100 MiB
r4.xlarge.search	100 MiB
r4.2xlarge.search	100 MiB
r4.4xlarge.search	100 MiB
r4.8xlarge.search	100 MiB
r4.16xlarge.search	100 MiB
r5.large.search	100 MiB
r5.xlarge.search	100 MiB
r5.2xlarge.search	100 MiB
r5.4xlarge.search	100 MiB
r5.12xlarge.search	100 MiB
r6g.large.search	100 MiB
r6g.xlarge.search	100 MiB
r6g.2xlarge.search	100 MiB
r6g.4xlarge.search	100 MiB
r6g.8xlarge.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
r6g.12xlarge.search	100 MiB
r6gd.large.search	100 MiB
r6gd.xlarge.search	100 MiB
r6gd.2xlarge.search	100 MiB
r6gd.4xlarge.search	100 MiB
r6gd.8xlarge.search	100 MiB
r6gd.12xlarge.search	100 MiB
r6gd.16xlarge.search	100 MiB
i2.xlarge.search	100 MiB
i2.2xlarge.search	100 MiB
i3.large.search	100 MiB
i3.xlarge.search	100 MiB
i3.2xlarge.search	100 MiB
i3.4xlarge.search	100 MiB
i3.8xlarge.search	100 MiB
i3.16xlarge.search	100 MiB
or1.medium.search	10 MiB
or1.large.search	100 MiB
or1.xlarge.search	100 MiB
or1.2xlarge.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
or1.4xlarge.search	100 MiB
or1.8xlarge.search	100 MiB
or1.12xlarge.search	100 MiB
or1.16xlarge.search	100 MiB
or2.medium.search	100 MiB
or2.large.search	100 MiB
or2.xlarge.search	100 MiB
or2.2xlarge.search	100 MiB
or2.4xlarge.search	100 MiB
or2.8xlarge.search	100 MiB
or2.12xlarge.search	100 MiB
or2.16xlarge.search	100 MiB
om2.large.search	10 MiB
om2.xlarge.search	100 MiB
om2.2xlarge.search	100 MiB
om2.4xlarge.search	100 MiB
om2.8xlarge.search	100 MiB
om2.12xlarge.search	100 MiB
om2.16xlarge.search	100 MiB
im4gn.large.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
im4gn.xlarge.search	100 MiB
im4gn.2xlarge.search	100 MiB
im4gn.4xlarge.search	100 MiB
im4gn.8xlarge.search	100 MiB
im4gn.16xlarge.search	100 MiB
i4i.large.search	該当なし
i4i.xlarge.search	該当なし
i4i.2xlarge.search	該当なし
i4i.4xlarge.search	該当なし
i4i.8xlarge.search	該当なし
i4i.12xlarge.search	該当なし
i4i.16xlarge.search	該当なし
i4i.24xlarge.search	該当なし
i4i.32xlarge.search	該当なし
i4g.large.search	該当なし
i4g.xlarge.search	該当なし
i4g.2xlarge.search	該当なし
i4g.4xlarge.search	該当なし
i4g.8xlarge.search	該当なし
i4g.16xlarge.search	該当なし

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
i8g.large.search	該当なし
i8g.xlarge.search	該当なし
i8g.2xlarge.search	該当なし
i8g.4xlarge.search	該当なし
i8g.8xlarge.search	該当なし
i8g.12xlarge.search	該当なし
i8g.16xlarge.search	該当なし
i7i.large.search	100 MiB
i7i.xlarge.search	100 MiB
i7i.2xlarge.search	100 MiB
i7i.4xlarge.search	100 MiB
i7i.8xlarge.search	100 MiB
i7i.12xlarge.search	100 MiB
i7i.16xlarge.search	100 MiB
i8ge.large.search	100 MiB
i8ge.xlarge.search	100 MiB
i8ge.2xlarge.search	100 MiB
i8ge.3xlarge.search	100 MiB
i8ge.6xlarge.search	100 MiB
i8ge.12xlarge.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
i8ge.18xlarge.search	100 MiB
c7i.large.search	100 MiB
c7i.xlarge.search	100 MiB
c7i.2xlarge.search	100 MiB
c7i.4xlarge.search	100 MiB
c7i.8xlarge.search	100 MiB
c7i.12xlarge.search	100 MiB
c7i.16xlarge.search	100 MiB
m7i.large.search	100 MiB
m7i.xlarge.search	100 MiB
m7i.2xlarge.search	100 MiB
m7i.4xlarge.search	100 MiB
m7i.8xlarge.search	100 MiB
m7i.12xlarge.search	100 MiB
m7i.16xlarge.search	100 MiB
r7i.large.search	100 MiB
r7i.xlarge.search	100 MiB
r7i.2xlarge.search	100 MiB
r7i.4xlarge.search	100 MiB
r7i.8xlarge.search	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
r7i.12xlarge.search	100 MiB
r7i.16xlarge.search	100 MiB
c8g.large	100 MiB
c8g.xlarge	100 MiB
c8g.2xlarge	100 MiB
c8g.4xlarge	100 MiB
c8g.8xlarge	100 MiB
c8g.12xlarge	100 MiB
c8g.16xlarge	100 MiB
m8g.medium	100 MiB
m8g.large	100 MiB
m8g.xlarge	100 MiB
m8g.2xlarge	100 MiB
m8g.4xlarge	100 MiB
m8g.8xlarge	100 MiB
m8g.12xlarge	100 MiB
m8g.16xlarge	100 MiB
r8g.medium	100 MiB
r8g.large	100 MiB
r8g.xlarge	100 MiB

インスタンスタイプ	HTTP リクエストペイロードの最大サイズ
r8g.2xlarge	100 MiB
r8g.4xlarge	100 MiB
r8g.8xlarge	100 MiB
r8g.12xlarge	100 MiB
r8g.16xlarge	100 MiB
r8gd.medium	100 MiB
r8gd.large	100 MiB
r8gd.xlarge	100 MiB
r8gd.2xlarge	100 MiB
r8gd.4xlarge	100 MiB
r8gd.8xlarge	100 MiB
r8gd.12xlarge	100 MiB
r8gd.16xlarge	100 MiB
oi2.large.search	100 MiB
oi2.xlarge.search	100 MiB
oi2.2xlarge.search	100 MiB
oi2.4xlarge.search	100 MiB
oi2.8xlarge.search	100 MiB
oi2.12xlarge.search	100 MiB
oi2.16xlarge.search	100 MiB

シャードサイズのクォータ

次のセクションでは、さまざまなインスタンスファミリーの最大シャードサイズを示します。

インスタンスタイプ	Multi-AZ without Standby	Multi-AZ with Standby
R5、C5、M5、C7i、M7i、R7i	該当なし	65 GiB
I3、i4i、i4g、i8g、i7i	該当なし	65 GiB
R6g、C6g、M6g、R6gd、C7g、M7g、R7g、R7gd、C8g、M8g、R8g、R8gd	該当なし	65 GiB
OR1、OR2、OM2、OI2	該当なし	該当なし
Im4gn、i8ge	該当なし	65 GiB

クォータの引き上げをリクエストするには、AWS サポートまでお問い合わせください。

シャード数のクォータ

次のセクションでは、OpenSearch バージョン別の最大シャード数を一覧表示します。

エンジンバージョン	制限	注意事項
Elasticsearch 1.5 から 6.x	デフォルトの制限なし	
Elasticsearch 7.x	1,000	デフォルトの制限はクラスターを介して変更できます。max_shards_per_node 設定。
OpenSearch 1.x から 2.15	1,000	デフォルトの制限はクラスターを介して変更できます。max_shards_per_node 設定。

エンジンバージョン	制限	注意事項
OpenSearch 2.17 以降	ヒープ 16 GB ごとに 1,000、最大 4,000	デフォルトの制限は変更できません。

Java プロセスのクォータ

OpenSearch Service は、Java がメモリを処理する合計使用可能なメモリの 50% (最大 32 GiB) に制限します。32 GB の上限は、r7g および OpenSearch 最適化インスタンスには適用されません。

制限の引き上げをリクエストするには、 AWS サポートにお問い合わせください。

上級ユーザーは、フィールドデータに使用されるヒープの割合を指定することもできます。詳細については、「高度なクラスター設定」と「JVM OutOfMemoryError」を参照してください。

ドメインポリシーのクォータ

OpenSearch Service では、ドメインのアクセスポリシーが 100 KiB に制限されます。

Amazon OpenSearch Service のリザーブドインスタンス

Amazon OpenSearch Service のリザーブドインスタンス (RI) では、標準オンデマンドインスタンスと比べて大幅な割引を受けられます。インスタンス自体は同一です。RI は、アカウントでのオンデマンドインスタンスに適用される請求の割引にすぎません。使用状況が予想可能な存続時間の長いアプリケーションについては、RI により時間の経過と共に大幅なコスト削減が可能になります。

> **Note**
>
> リザーブドインスタンスのタグ付けは、Amazon OpenSearch Service ではサポートされていません。Amazon RDS などの他の AWS サービスとは異なり、OpenSearch Service リザーブドインスタンスにリソースタグを適用することはできません。

OpenSearch Service RI には 1 年または 3 年の期間が必要で、割引レートに影響を与える 3 つのお支払い方法があります。

- 前払いなし – 前払いはありません。期間内の時間ごとに割引された時間料金を支払います。
- 一部前払い – 料金の一部を前払いし、期間内の時間ごとに割引された時間料金を支払います。

- 全前払い – 料金を全額前払いします。期間内に時間料金は支払いません。

一般的に、前払い料金が多いほど割引率がなくなります。リザーブドインスタンスをキャンセルすることはできず (予約するときに、全期間に対して支払う契約を結びます)、前払い料金は払い戻しできません。

RI は柔軟ではなく、予約したインスタンスタイプにのみ適用されます。例えば、8 個の `c5.2xlarge.search` インスタンスのための予約は、16 個の `c5.xlarge.search` インスタンスまたは 4 個の `c5.4xlarge.search` インスタンスには適用されません。ただし、 の組織の一部であるリンクアカウント AWS Organizations は、インスタンスタイプ、リージョン、ファミリー、サイズが一致する限り、RI の所有アカウントから未使用の割引アプリケーションを利用できます。詳細については、「Amazon OpenSearch Service の料金」と「よくある質問」を参照してください。

リザーブドインスタンスの購入 (コンソール)

コンソールでは、既存のリザーブドインスタンスを表示したり、新しいリザーブドインスタンスを購入したりできます。

予約を購入するには

1. https://aws.amazon.com にアクセスし、[コンソールにサインイン] を選択します。
2. [分析] の下で、[Amazon OpenSearch Service] を選択します。
3. ナビゲーションペインで [リザーブドインスタンスのリース] を選択します。

   このページでは、既存の予約を見ることができます。多くの予約がある場合、フィルタリングして特定の予約を識別して見やすくすることができます。

   > **Tip**
   >
   > [リザーブドインスタンスのリース] リンクが表示されない場合は、 AWS リージョンでドメインを作成します。

4. [リザーブドインスタンスの注文] を選択します。
5. 一意のわかりやすい名前を指定します。
6. インスタンスタイプ、サイズ、およびインスタンスの数を選択します。ガイダンスについては、「the section called “ドメインのサイジング”」を参照してください。
7. 期間の長さと支払いオプションを選択します。支払いの詳細を慎重に確認します。

8. [次へ] を選択します。
9. 購入概要を慎重に確認します。リザーブドインスタンスの購入は払い戻しできません。
10. [注文] を選択します。

リザーブドインスタンスの購入 (AWS CLI)

AWS CLI には、サービスの表示、予約の購入、予約の表示を行うためのコマンドがあります。次のコマンドとサンプルレスポンスは、特定の のサービスを示しています AWS リージョン。

```
aws opensearch describe-reserved-instance-offerings --region us-east-1
{
  "ReservedInstanceOfferings": [
    {
      "FixedPrice": x,
      "ReservedInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": y,
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "UsagePrice": 0.0,
      "PaymentOption": "PARTIAL_UPFRONT",
      "Duration": 31536000,
      "InstanceType": "m4.2xlarge.search",
      "CurrencyCode": "USD"
    }
  ]
}
```

各戻り値の説明については、次の表を参照してください。

フィールド	説明
FixedPrice	予約の前払いコスト。
ReservedInstanceOfferingId	提供 ID。製品を予約する場合は、この値をメモしておきます。
RecurringCharges	予約の時間料金。

フィールド	説明
UsagePrice	レガシーフィールド。OpenSearch Service では、この値は常に 0 です。
PaymentOption	前払いなし、一部前払い、または全額前払い。
Duration	期間の長さ (秒)。 • 31536000 秒が 1 年です。 • 94608000 秒が 3 年です。
InstanceType	予約のインスタンスタイプ。各インスタンスタイプに割り当てられるハードウェアリソースについては、「Amazon OpenSearch Service の料金表」を参照してください。
CurrencyCode	FixedPrice と RecurringChargeAmount の通貨。

この次の例では、予約を購入します。

```
aws opensearch purchase-reserved-instance-offering --reserved-instance-offering-
id 1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a --reservation-name my-reservation --instance-
count 3 --region us-east-1
{
  "ReservationName": "my-reservation",
  "ReservedInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a"
}
```

最後に、以下の例を使用して特定のリージョンの予約をリストできます。

```
aws opensearch describe-reserved-instances --region us-east-1
{
  "ReservedInstances": [
    {
      "FixedPrice": x,
      "ReservedInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
      "ReservationName": "my-reservation",
      "PaymentOption": "PARTIAL_UPFRONT",
```

```
      "UsagePrice": 0.0,
      "ReservedInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": y,
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "State": "payment-pending",
      "StartTime": 1522872571.229,
      "InstanceCount": 3,
      "Duration": 31536000,
      "InstanceType": "m4.2xlarge.search",
      "CurrencyCode": "USD"
    }
  ]
}
```

> **Note**
>
> `StartTime` は Unix エポック時間 (1970 年 1 月 1 日の午前 00:00 UTC から経過した秒数)。たとえば、1522872571 エポック時間は 2018 年 4 月 4 日の 20:09:31 UTC です。オンラインコンバーターを使用することができます。

前の例で使用されているコマンドの詳細については、AWS CLI コマンドリファレンスを参照してください。

リザーブドインスタンス (AWS SDKsの購入

AWS SDKs (Android および iOS SDKs を除く) は、Amazon OpenSearch Service API リファレンスで定義されている、以下を含むすべてのオペレーションをサポートします。

- `DescribeReservedInstanceOfferings`
- `PurchaseReservedInstanceOffering`
- `DescribeReservedInstances`

このサンプルスクリプトは、 の OpenSearchService 低レベル Python クライアント AWS SDK for Python (Boto3) を使用してリザーブドインスタンスを購入します。`instance_type` の値を指定する必要があります。

```
import boto3
from botocore.config import Config

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

my_config = Config(
    # Optionally lets you specify a region other than your default.
    region_name='us-east-1'
)

client = boto3.client('opensearch', config=my_config)

instance_type = '' # e.g. m4.2xlarge.search


def describe_RI_offerings(client):
    """Gets the Reserved Instance offerings for this account"""

    response = client.describe_reserved_instance_offerings()
    offerings = (response['ReservedInstanceOfferings'])
    return offerings


def check_instance(offering):
    """Returns True if instance type is the one you specified above"""

    if offering['InstanceType'] == instance_type:
        return True

    return False


def get_instance_id():
    """Iterates through the available offerings to find the ID of the one you
 specified"""

    instance_type_iterator = filter(
        check_instance, describe_RI_offerings(client))
    offering = list(instance_type_iterator)
    id = offering[0]['ReservedInstanceOfferingId']
    return id
```

```
def purchase_RI_offering(client):
    """Purchase Reserved Instances"""

    response = client.purchase_reserved_instance_offering(
        ReservedInstanceOfferingId = get_instance_id(),
        ReservationName = 'my-reservation',
        InstanceCount = 1
    )
    print('Purchased reserved instance offering of type ' + instance_type)
    print(response)


def main():
    """Purchase Reserved Instances"""
    purchase_RI_offering(client)
```

AWS SDKsAWS 「 Software Development Kits」を参照してください。

コストを確認する

Cost Explorer は、過去 13 か月間の使用量データを表示できる無料のツールです。このデータを分析すると、傾向を見分けて、RI がユースケースに合うかどうかを理解できます。既に RI がある場合、[購入オプション] によってグループ分けし、償却コストを表示してその使用量とオンデマンドインスタンスの使用量を比較できます。また、使用状況の予算を設定して、予約を活用していることを確認することもできます。詳細については、AWS Billing ユーザーガイドの「Cost Explorer によるコストの分析」を参照してください。

Amazon OpenSearch Service でサポートされている他のリソース

このトピックでは、Amazon OpenSearch Service でサポートされる他のリソースについて説明します。

bootstrap.memory_lock

> OpenSearch Service により、`opensearch.yml` の `bootstrap.memory_lock` が有効になります。これにより、JVM メモリがロックされ、オペレーティングシステムがディスクにスワップされなくなります。これは、以下を除いて、サポートされているすべてのインスタンスタイプに適用されます。

- t2.micro.search
- t2.small.search
- t2.medium.search
- t3.small.search
- t3.medium.search

スクリプト モジュール

OpenSearch Service では、Elasticsearch 5.x 以降のドメインでスクリプティングがサポートされています。このサービスでは、1.5 または 2.3 用のスクリプティングがサポートされていません。

サポートされているスクリプトオプションには、以下が含まれます。

- Painless
- Lucene Expressions
- Mustache

Elasticsearch 5.5 以降のドメインおよびすべての OpenSearch ドメインでは、OpenSearch Service で _scripts エンドポイントを使用するストアドスクリプトがサポートされています。Elasticsearch 5.3 および 5.1 ドメインでは、インラインスクリプトのみがサポートされています。

TLS トランスポート

OpenSearch Service は、ポート 80 での HTTP とポート 443 での HTTPS をサポートしていますが、TLS トランスポートはサポートしていません。

Amazon OpenSearch Service のチュートリアル

この章には、サービスへの移行、簡単な検索アプリケーションの構築、OpenSearch Dashboards での可視化の作成など、Amazon OpenSearch Service を使用するための全般的なチュートリアルがいくつか含まれています。

トピック

- チュートリアル: Amazon OpenSearch Service でドキュメントを作成および検索する
- チュートリアル: Amazon OpenSearch Service への移行
- チュートリアル: Amazon OpenSearch Service を用いて検索アプリケーションを作成する
- チュートリアル: OpenSearch Service と OpenSearch Dashboards によるカスタマーサポートへの問い合わせを可視化する

チュートリアル: Amazon OpenSearch Service でドキュメントを作成および検索する

このチュートリアルでは、Amazon OpenSearch Service でドキュメントを作成および検索する方法を説明します。データを JSON ドキュメント形式でインデックスに追加します。OpenSearch Service は、追加した最初のドキュメントに関するインデックスを作成します。

このチュートリアルでは、HTTP リクエストを実行してドキュメントを作成し、ドキュメントの ID を自動的に生成し、ドキュメントに対して基本的な検索および高度な検索を実行する方法を説明します。

> **Note**
>
> このチュートリアルでは、オープンアクセスを持つドメインを使用します。最高レベルのセキュリティを実現するため、仮想プライベートクラウド (VPC) 内にドメインを配置することをお勧めします。

前提条件

このチュートリアルには、次のような前提条件があります。

- が必要です AWS アカウント。
- アクティブな OpenSearch Service のドメインが必要です。

ドキュメントのインデックスへの追加

ドキュメントをインデックスに追加するために、Postman、cURL、または OpenSearch Dashboards コンソールなどの任意の HTTP ツールを使用できます。これらの例は、OpenSearch Dashboards でデベロッパーコンソールを使用していることを前提としています。別のツールを使用している場合は、必要に応じて完全な URL と認証情報を入力して、適宜調整してください。

ドキュメントをインデックスに追加するには

1. 使用しているドメインの OpenSearch Dashboards URL に移動します。この URL は、OpenSearch Service コンソールのドメインダッシュボードに表示されています。URL はこの形式に従います。

   ```
   domain-endpoint/_dashboards/
   ```

2. プライマリユーザー名とパスワードを使ってサインインします。
3. 左側のナビゲーションパネルを開き、[Dev Tools] (開発ツール) を選択します。
4. 新しいリソースを作成するための HTTP 動詞は PUT です。これは、新しいドキュメントとインデックスの作成に使用します。コンソールに次のコマンドを入力します。

   ```
   PUT fruit/_doc/1
   {
     "name":"strawberry",
     "color":"red"
   }
   ```

   PUT リクエストは、fruit という名前のインデックスを作成し、ID が 1 である単一のドキュメントをインデックスに追加します。次のレスポンスが生成されます。

   ```
   {
     "_index" : "fruit",
     "_type" : "_doc",
     "_id" : "1",
     "_version" : 1,
     "result" : "created",
     "_shards" : {
   ```

```
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

自動的に生成される ID の作成

OpenSearch Service は、ドキュメントの ID を自動的に生成できます。ID を生成するコマンドは、PUT リクエストの代わりに POST リクエストを使用し、(前のリクエストと比較して) ドキュメント ID を必要としません。

デベロッパーコンソールに次のリクエストを入力します。

```
POST veggies/_doc
{
  "name":"beet",
  "color":"red",
  "classification":"root"
}
```

このリクエストは、veggies という名前のインデックスを作成し、ドキュメントをインデックスに追加します。次のレスポンスが生成されます。

```
{
  "_index" : "veggies",
  "_type" : "_doc",
  "_id" : "3WgyS4IB5DLqbRIvLxtF",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

レスポンスの追加 _id フィールドに注意してください。これは、ID が自動的に作成されたことを示します。

> **Note**
>
> URL の _doc の後には何も入力しません。ここには通常 ID が挿入されます。生成される ID を使用してドキュメントを作成しようとしているため、まだ ID を入力しません。これは更新の際に行います。

POST コマンドを使用したドキュメントの更新

ドキュメントを更新するには、ID 番号を持つ HTTP POST コマンドを使用します。

最初に、ID が 42 のドキュメントを作成します。

```
POST fruits/_doc/42
{
  "name":"banana",
  "color":"yellow"
}
```

その後、その ID を使用してドキュメントを更新します。

```
POST fruits/_doc/42
{
  "name":"banana",
  "color":"yellow",
  "classification":"berries"
}
```

このコマンドは、ドキュメントを新しいフィールド classification で更新します。次のレスポンスが生成されます。

```
{
  "_index" : "fruits",
  "_type" : "_doc",
  "_id" : "42",
  "_version" : 2,
  "result" : "updated",
```

```
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 1,
  "_primary_term" : 1
}
```

> **Note**
>
> 存在しないドキュメントを更新しようとすると、OpenSearch Service はドキュメントを作成します。

一括アクションの実行

POST _bulk API オペレーションを使用して、1 回のリクエストで 1 つ以上のインデックスに対して複数のアクションを実行できます。一括アクションコマンドは、次の形式になります。

```
POST /_bulk
<action_meta>\n
<action_data>\n
<action_meta>\n
<action_data>\n
```

各アクションには 2 行の JSON が必要です。まず、アクションの説明またはメタデータを入力します。次の行で、データを入力します。各パートは改行 (\n) で区切られます。挿入のアクションの説明は次のようになります。

```
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "7" } }
```

また、データを含む次の行は次のようになります。

```
{ "name":"kale", "color":"green", "classification":"leafy-green" }
```

メタデータとデータをまとめると、一括オペレーションでの 1 つのアクションが表されます。次のように、1 つのリクエストで多くのオペレーションを実行できます。

```
POST /_bulk
{ "create" : { "_index" : "veggies", "_id" : "35" } }
{ "name":"kale", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "36" } }
{ "name":"spinach", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "37" } }
{ "name":"arugula", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "38" } }
{ "name":"endive", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "39" } }
{ "name":"lettuce", "color":"green", "classification":"leafy-green" }
{ "delete" : { "_index" : "vegetables", "_id" : "1" } }
```

最後のアクションが delete であることに注意してください。delete アクションに続くデータはありません。

ドキュメントの検索

これでクラスターにデータが存在するようになったので、そのデータを検索できます。例えば、すべての根菜類を検索したり、すべての葉菜類の数を取得したり、1 時間あたりに口グ記録されたエラーの数を調べたりすることができます。

基本的な検索

基本的な検索は次のようになります。

```
GET veggies/_search?q=name:l*
```

リクエストにより、レタスドキュメントを含む JSON レスポンスが生成されます。

高度な検索

リクエスト本文でクエリオプションを JSON として指定すると、より高度な検索を実行できます。

```
GET veggies/_search
{
  "query": {
    "term": {
      "name": "lettuce"
    }
```

```
  }
}
```

この例では、レタスドキュメントを含む JSON レスポンスも生成します。

ソート

ソートを使用すると、このタイプのクエリをさらに多く実行できます。まず、インデックスを再作成する必要があります。これは、自動フィールドマッピングが、デフォルトではソートできないタイプを選択するためです。次のリクエストを送信して、インデックスを削除して再作成します。

```
DELETE /veggies

PUT /veggies
{
   "mappings":{
      "properties":{
         "name":{
            "type":"keyword"
         },
         "color":{
            "type":"keyword"
         },
         "classification":{
            "type":"keyword"
         }
      }
   }
}
```

その後、インデックスにデータを再入力します。

```
POST /_bulk
{ "create" : { "_index" : "veggies", "_id" : "7"  } }
{ "name":"kale", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "8" } }
{ "name":"spinach", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "9" } }
{ "name":"arugula", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "10" } }
{ "name":"endive", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_id" : "11" } }
```

```
{ "name":"lettuce", "color":"green", "classification":"leafy-green" }
```

これで、ソートを使用して検索できるようになりました。このリクエストでは、分類による昇順のソートが追加されます。

```
GET veggies/_search
{
  "query" : {
    "term": { "color": "green" }
  },
  "sort" : [
      "classification"
  ]
}
```

関連リソース

詳細については、以下のリソースを参照してください。

- 開始方法
- データのインデックス作成
- データの検索

チュートリアル: Amazon OpenSearch Service への移行

インデックススナップショットは、セルフマネージド OpenSearch または従来の Elasticsearch クラスターから Amazon OpenSearch Service に移行する一般的な方法です。そのプロセスは、大きく分けて以下のステップで構成されています。

1. 既存のクラスターのスナップショットを作成し、そのスナップショットを Amazon S3 バケットにアップロードする。
2. OpenSearch Service ドメインを作成する。
3. バケットにアクセスするための許可を OpenSearch Service に付与し、スナップショットを使用するための許可があることを確認する。
4. OpenSearch Service ドメインでスナップショットを復元する。

このチュートリアルでは、より詳細な手順と代替のオプション (ある場合) について説明します。

スナップショットの作成とアップロード

repository-s3 プラグインを使用して S3 に直接スナップショットを取得できますが、プラグインをすべてのノードにインストールし、調整 opensearch.yml (または Elasticsearch クラスターelasticsearch.ymlを使用している場合) 、各ノードを再起動し、 AWS 認証情報を追加して、最後にスナップショットを作成する必要があります。このプラグインは、継続使用や大規模なクラスターの移行の場合に役立ちます。

小規模なクラスターの場合、1 回限りのアプローチとして、共有ファイルシステムのスナップショットを作成し、 を使用して S3 AWS CLI にアップロードします。すでにスナップショットを作成している場合は、手順 4 に進んでください。

スナップショットを取り、Amazon S3 にアップロードするには

1. すべてのノードで opensearch.yml (または Elasticsearch.yml) に path.repo 設定を追加して、各ノードを再起動します。

```
path.repo: ["/my/shared/directory/snapshots"]
```

2. スナップショットを作成する前に必要な スナップショットリポジトリ を登録します。リポジトリは単なる保存場所です。共有ファイルシステム、Amazon S3、File system distribuito Hadoop (HDFS) などです。この場合、共有ファイルシステム (「fs」) を使用します。

```
PUT _snapshot/my-snapshot-repo-name
{
  "type": "fs",
  "settings": {
    "location": "/my/shared/directory/snapshots"
  }
}
```

3. スナップショットを作成します。

```
PUT _snapshot/my-snapshot-repo-name/my-snapshot-name
{
  "indices": "migration-index1,migration-index2,other-indices-*",
  "include_global_state": false
}
```

4. AWS CLI をインストールし、aws configure を実行して認証情報を追加します。

5. スナップショットのディレクトリに移動します。次のコマンドを実行して新しい S3 バケットを作成し、スナップショットのディレクトリの中身をそのバケットにアップロードします。

```
aws s3 mb s3://amzn-s3-demo-bucket --region us-west-2
aws s3 sync . s3://amzn-s3-demo-bucket --sse AES256
```

スナップショットのサイズとインターネット接続の速度によっては、この操作に時間がかかる場合があります。

ドメインの作成

コンソールはドメインを作成する最も簡単な方法ですが、この場合、ターミナルはすでに開いていて、 AWS CLI がインストールされています。次のコマンドを変更して、ニーズに合わせてドメインを作成してください。

```
aws opensearch create-domain \
  --domain-name migration-domain \
  --engine-version OpenSearch_1.0 \
  --cluster-config InstanceType=c5.large.search,InstanceCount=2 \
  --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=100 \
  --node-to-node-encryption-options Enabled=true \
  --encryption-at-rest-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true,TLSSecurityPolicy=Policy-Min-
TLS-1-2-2019-07 \
  --advanced-security-options
 Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions='{MasterUserName=master-
user,MasterUserPassword=master-user-password}' \
  --access-policies '{"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal": {"AWS": "arn:aws:iam::aws-
region:user/UserName"},"Action":["es:ESHttp*"],"Resource":"arn:aws:es:aws-
region:111122223333:domain/migration-domain/*"}]}' \
  --region aws-region
```

このコマンドでは、それぞれ 100 GiB のストレージを持つ 2 つのデータノードがあるインターネットにアクセス可能なドメインが作成されます。また、HTTP Basic 認証とすべての暗号化の設定により、きめ細かなアクセスコントロールが可能になります。VPC などのアドバンストセキュリティ設定が必要な場合は、OpenSearch Service コンソールを使用してください。

このコマンドを発行する前に、ドメイン名、マスターユーザーの認証情報、アカウント番号を変更します。S3 バケット AWS リージョン に使用したものと同じ と、スナップショットと互換性のある OpenSearch/Elasticsearch バージョンを指定します。

> **Important**
>
> スナップショットには上位互換性のみがあり、その対象は 1 つのメジャーバージョンのみです。例えば、Elasticsearch 7.x クラスター上の OpenSearch 1 x クラスターからスナップショットを復元することはできず、OpenSearch 1.x または 2.x クラスターのみです。また、マイナーバージョンも同様です。セルフマネージド 5.3.3 クラスターのスナップショットを 5.3.2 の OpenSearch Service ドメインに復元することはできません。OpenSearch または Elasticsearch は、スナップショットがサポートしている最新バージョンを選択することをお勧めします。互換性のあるバージョンのテーブルについては、「the section called “スナップショットを使用してデータを移行する”」を参照してください。

S3 バケットへの許可を提供します。

AWS Identity and Access Management (IAM) コンソールで、次のアクセス許可と信頼関係を持つロールを作成します。 https://docs.aws.amazon.com/IAM/latest/UserGuide/roles-managingrole-editing-console.html#roles-managingrole_edit-trust-policyロールの作成時に、AWS サービスとして S3 を選択します。このロールには、分かりやすいように `OpenSearchSnapshotRole` と名前を付けます。

アクセス許可

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    },
```

```
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

信頼関係

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

次に、個人的な IAM ロールに、OpenSearchSnapshotRole を引き受ける許可を付与します。以下のポリシーを作成して、アイデンティティにアタッチします。

アクセス許可

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [{
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/OpenSearchSnapshotRole"
    }
  ]
}
```

OpenSearch Dashboards にスナップショットロールをマッピングします (きめ細かなアクセスコントロールを使用している場合)

きめ細かなアクセスコントロールを有効にした場合は、HTTP 基本認証をその他すべての目的で使用する場合でも、manage_snapshots ロールを IAM ロールにマップして、スナップショットを使用できるようにする必要があります。

スナップショットを使用するためのアイデンティティの許可を提供するには

1. OpenSearch Service ドメインの作成時に指定したマスターユーザーの認証情報を使用して Dashboards にログインします。Dashboards URL は、OpenSearch Service コンソールに表示されます。`https://domain-endpoint/_dashboards/` の形式です。
2. メインメニューから [セキュリティ]、[ロール] を選択し、[manage_snapshots] ロールを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. 適切なフィールドに、個人的な IAM ロールのドメイン ARN を追加します。ARN は次のいずれかの形式となります。

   ```
   arn:aws:iam::123456789123:user/user-name
   ```

   ```
   arn:aws:iam::123456789123:role/role-name
   ```

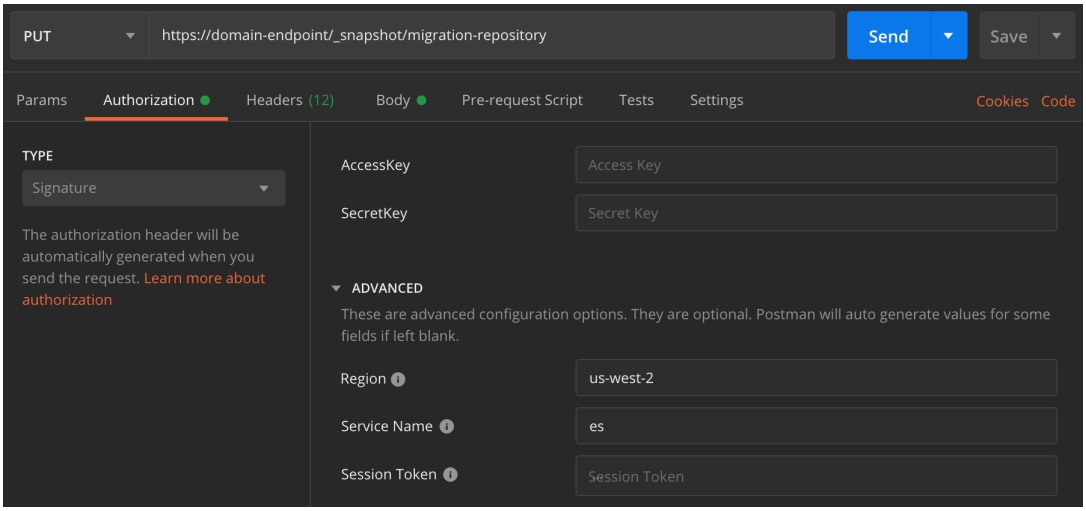
5. マップを選択し、ロールがマッピングされたユーザーに表示されていることを確認します。

スナップショットを復元する

この時点で、OpenSearch Service ドメインにアクセスするには、マスターユーザー認証情報を使用した HTTP 基本認証または IAM 認証情報を使用した AWS 認証の 2 つの方法があります。スナップ

ショットはマスターユーザーの概念がない Amazon S3 を使用するため、IAM 認証情報を使用してスナップショットのリポジトリを OpenSearch Service ドメインに登録する必要があります。

ほとんどのプログラミング言語にはリクエストの署名に役立つライブラリがありますが、Postman などのツールを使用して IAM 認証情報を [認可] セクションに入力するのが簡単な方法です。



スナップショットを復元するには

1. リクエストにどのような方法で署名するかにかかわらず、まずリポジトリを登録します。

```
PUT _snapshot/my-snapshot-repo-name
{
  "type": "s3",
  "settings": {
    "bucket": "amzn-s3-demo-bucket",
    "region": "us-west-2",
    "role_arn": "arn:aws:iam::123456789012:role/OpenSearchSnapshotRole",
    "base_path": "snapshots/my-snapshot-path"
  }
}
```

> **Note**
>
> base_path パラメータは、スナップショットが保存される S3 バケット内のパスを指定します。がないとbase_path、OpenSearch Service はアップロードされたスナップショットを S3 バケット内で見つけることができません。

2. 次に、リポジトリ内のスナップショットを一覧表示し、復元するスナップショットを見つけます。この時点で、Postman を続けて使用するか、curl などのツールに切り替えることができます。

   短縮構文

   ```
   GET _snapshot/my-snapshot-repo-name/_all
   ```

   curl

   ```
   curl -XGET -u 'master-user:master-user-password' https://domain-endpoint/
   _snapshot/my-snapshot-repo-name/_all
   ```

3. スナップショットを復元します。

   短縮構文

   ```
   POST _snapshot/my-snapshot-repo-name/my-snapshot-name/_restore
   {
     "indices": "migration-index1,migration-index2,other-indices-*",
     "include_global_state": false
   }
   ```

   curl

   ```
   curl -XPOST -u 'master-user:master-user-password' https://domain-endpoint/
   _snapshot/my-snapshot-repo-name/my-snapshot-name/_restore \
     -H 'Content-Type: application/json' \
     -d '{"indices":"migration-index1,migration-index2,other-indices-
   *","include_global_state":false}'
   ```

4. 最後に、インデックスが正常に復元されていることを検証します。

   短縮構文

   ```
   GET _cat/indices?v
   ```

   curl

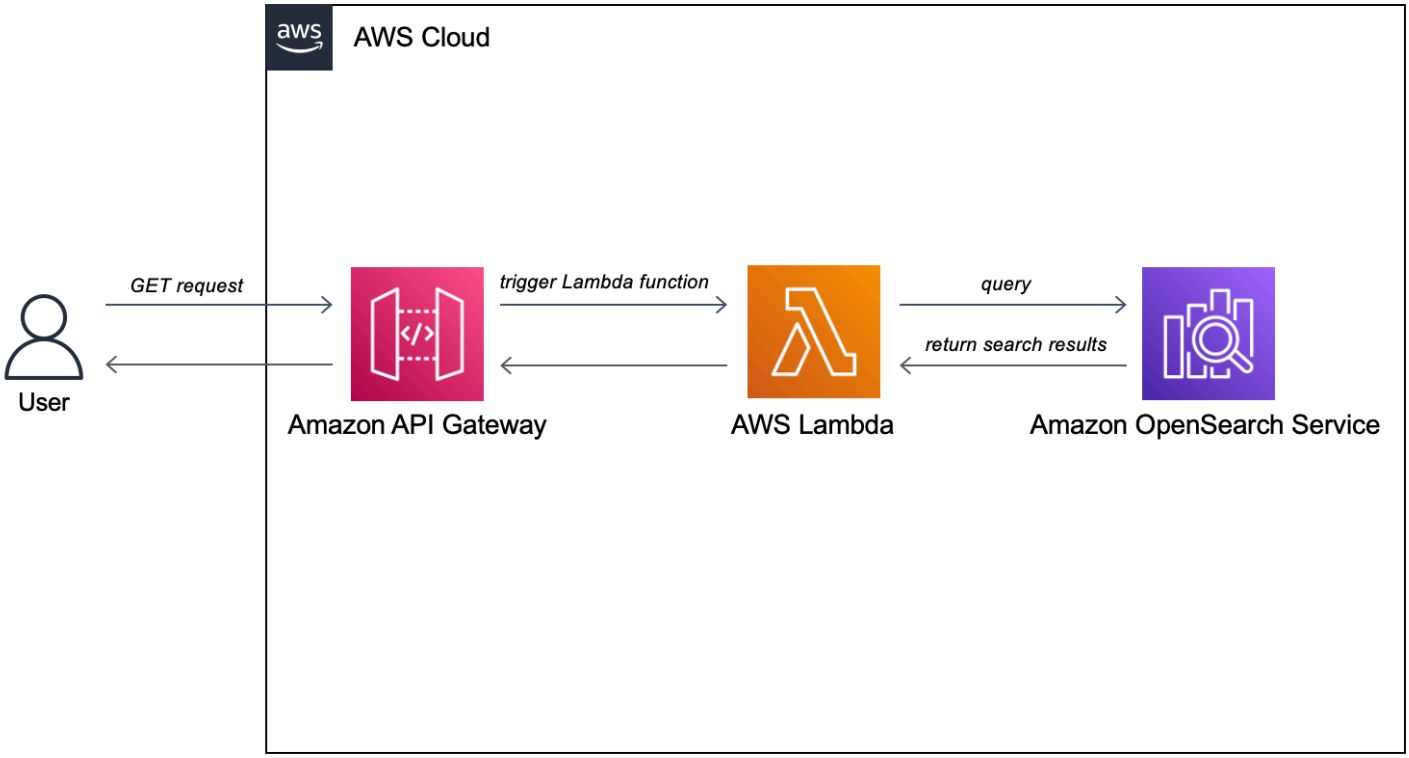
```
curl -XGET -u 'master-user:master-user-password' https://domain-endpoint/_cat/
indices?v
```

これで、移行は完了です。続いて、新しい OpenSearch Service エンドポイントを使用するようにクライアントを設定したり、ワークロードに合わせてドメインのサイズを変更したり、インデックスのシャード数を確認したり、IAM マスターユーザーに切り替えたり、OpenSearch Dashboards での可視化の構築を開始したりすることができます。

チュートリアル: Amazon OpenSearch Service を用いて検索アプリケーションを作成する

Amazon OpenSearch Service を用いて検索アプリケーションを作成する一般的な方法は、サーバーへのユーザークエリを送信するウェブフォームを使用することです。次に、OpenSearch API を直接呼び出すようサーバーを承認し、サーバーが OpenSearch Service にリクエストを送信するようにします。ただし、サーバーに依存しないクライアント側のコードを記述する場合は、セキュリティとパフォーマンスのリスクを補正する必要があります。OpenSearch API への、署名されていないパブリックアクセスを許可することはお勧めしません。ユーザーは、保護されていないエンドポイントにアクセスしたり、過度に広範なクエリ (または多すぎるクエリ) によりクラスターのパフォーマンスに影響を及ぼす可能性があります。

この章では、Amazon API Gateway を使用して OpenSearch API APIsし、API Gateway から OpenSearch Service へのリクエストに署名 AWS Lambda するソリューションについて説明します。



> **Note**
>
> スタンダードの API Gateway および Lambda 料金表が適用されますが、このチュートリアル内での使用は限定されているため、費用はごくわずかです。

前提条件

このチュートリアルの前提条件は、OpenSearch Service ドメインです。まだドメインを持っていない場合は、「OpenSearch Service ドメインを作成する」のステップに従って、ドメインを作成します。

ステップ 1: サンプルデータのインデックス作成

sample-movies.zip をダウンロードして解凍します。その後 _bulk API オペレーションを使用して、`movies` インデックスに 5,000 個のドキュメントを追加します。

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/_bulk
{ "index": { "_index": "movies", "_id": "tt1979320" } }
```

```
{"directors":["Ron
 Howard"],"release_date":"2013-09-02T00:00:00Z","rating":8.3,"genres":
["Action","Biography","Drama","Sport"],"image_url":"http://ia.media-imdb.com/images/
M/MV5BMTQyMDE0MTY0OV5BMl5BanBnXkFtZTcwMjI2OTI0OQ@@._V1_SX400_.jpg","plot":"A re-
creation of the merciless 1970s rivalry between Formula One rivals James Hunt and
 Niki Lauda.","title":"Rush","rank":2,"running_time_secs":7380,"actors":["Daniel
 Brühl","Chris Hemsworth","Olivia Wilde"],"year":2013,"id":"tt1979320","type":"add"}
{ "index": { "_index": "movies", "_id": "tt1951264" } }
{"directors":["Francis Lawrence"],"release_date":"2013-11-11T00:00:00Z","genres":
["Action","Adventure","Sci-Fi","Thriller"],"image_url":"http://ia.media-imdb.com/
images/M/
MV5BMTAyMjQ3OTAxMzNeQTJeQWpwZ15BbWU4MDU0NzA1MzAx._V1_SX400_.jpg","plot":"Katniss
 Everdeen and Peeta Mellark become targets of the Capitol after
 their victory in the 74th Hunger Games sparks a rebellion in
 the Districts of Panem.","title":"The Hunger Games: Catching
 Fire","rank":4,"running_time_secs":8760,"actors":["Jennifer Lawrence","Josh
 Hutcherson","Liam Hemsworth"],"year":2013,"id":"tt1951264","type":"add"}
...
```

上記は、利用可能なデータの、ごく一部のコマンド例であることにご注意ください。_bulk オペレーションを実行するには、sample-movies ファイルのコンテンツ全体をコピーして貼り付ける必要があります。詳細な手順については、「the section called “オプション 2: 複数のドキュメントをアップロードする”」を参照してください。

次のように curl コマンドを使用しても同じ結果が得られます。

```
curl -XPOST -u 'master-user:master-user-password' 'domain-endpoint/_bulk' --data-binary
 @bulk_movies.json -H 'Content-Type: application/json'
```

ステップ 2: Lambda 関数を作成してデプロイする

API Gateway で API を作成する前に、リクエストを渡す Lambda 関数を作成します。

Lambda 関数を作成する

このソリューションでは、API Gateway は Lambda 関数にリクエストを渡します。この関数は OpenSearch Service にクエリを送り、結果を返します。このサンプル関数は外部ライブラリを使用するため、デプロイパッケージを作成して Lambda にアップロードする必要があります。

デプロイパッケージを作成するには

1. コマンドプロンプトを開き、my-opensearch-function プロジェクトディレクトリを作成します。例えば、macOS では次のようになります。

```
mkdir my-opensearch-function
```

2. my-sourcecode-function プロジェクトディレクトリに移動します。

```
cd my-opensearch-function
```

3. 次のサンプル Python コードの内容をコピーし、opensearch-lambda.py という名前の新しいファイルに保存します。リージョンとホストエンドポイントをファイルに追加します。

```
import boto3
import json
import requests
from requests_aws4auth import AWS4Auth

region = '' # For example, us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # The OpenSearch domain endpoint with https:// and without a trailing
 slash
index = 'movies'
url = host + '/' + index + '/_search'

# Lambda execution starts here
def lambda_handler(event, context):

    # Put the user query into the query DSL for more accurate search results.
    # Note that certain fields are boosted (^).
    query = {
        "size": 25,
        "query": {
            "multi_match": {
                "query": event['queryStringParameters']['q'],
                "fields": ["title^4", "plot^2", "actors", "directors"]
            }
        }
```

```
    }

    # Elasticsearch 6.x requires an explicit Content-Type header
    headers = { "Content-Type": "application/json" }

    # Make the signed HTTP request
    r = requests.get(url, auth=awsauth, headers=headers, data=json.dumps(query))

    # Create the response and add some extra content to support CORS
    response = {
        "statusCode": 200,
        "headers": {
            "Access-Control-Allow-Origin": '*'
        },
        "isBase64Encoded": False
    }

    # Add the search results to the response
    response['body'] = r.text
    return response
```

4. 新しい package ディレクトリに外部のライブラリをインストールします。

```
pip3 install --target ./package boto3
pip3 install --target ./package requests
pip3 install --target ./package requests_aws4auth
```

5. ルートにインストール済みライブラリを含むデプロイパッケージを作成します。次のコマンドを使ってプロジェクトディレクトリに my-deployment-package.zip ファイルを生成します。

```
cd package
zip -r ../my-deployment-package.zip .
```

6. zip ファイルのルートに opensearch-lambda.py ファイルを追加します。

```
cd ..
zip my-deployment-package.zip opensearch-lambda.py
```

Lambda 関数とデプロイパッケージの作成の詳細については、AWS Lambda デベロッパーガイドの「.zip ファイルアーカイブで Python Lambda 関数をデプロイする」と本ガイドの「the section called “Lambda デプロイパッケージを作成する”」を参照してください。

Lambda コンソールを使って関数を作成するには

1. Lambda コンソール (https://console.aws.amazon.com/lambda/home) に進みます。ナビゲーションペインで、[Functions] (関数) を選択します。
2. [Create Function] (関数を作成) を選択します。
3. 以下のフィールドを設定します。

   - 関数名: opensearch-function
   - ランタイム: Python 3.9
   - アーキテクチャ: x86_64

   他のオプションはすべてデフォルトのままにして、[Create function] を選択します。
4. 関数の概要ページの[Code source] (コードソース) セクションで、ドロップダウンから [Upload from] (アップロード元) を選択し、[.zip file] (.zip ファイル) を選択します。`my-deployment-package.zip` 作成したファイルを見つけて、[Save] (保存) を選択します。
5. ハンドラーとは、イベントを処理する関数コード内のメソッドです。[Runtime settings] (ランタイム設定) の下で [Edit] (編集) を選択し、Lambda 関数が配置されているデプロイパッケージ内の、ファイルの名前に従ってハンドラー名を変更します。例えば、ファイルの名前が `opensearch-lambda.py` の場合、ハンドラー名を *`opensearch-lambda`*`.lambda_handler` に変更します。詳細については、「Python の Lambda 関数ハンドラー」を参照してください。

ステップ 3: API Gateway で API を作成する

API Gateway を使用してさらに制限された API を作成することで、OpenSearch `_search` API とのやり取りのプロセスを簡素化することができます。API Gateway により、Amazon Cognito 認証やリクエストスロットリングなどのセキュリティ機能を有効にすることもできます。API を作成してデプロイするには、以下のステップを実施します。

API の作成と設定

API Gateway コンソールを使用して API を作成するには

1. API Gateway コンソール (https://console.aws.amazon.com/apigateway/home) に移動します。ナビゲーションペインで、[API] を選択します。
2. [REST API] (プライベートではない) を見つけ、[構築] を選択します。

3. 次のページの [Create new API] (新しい API を作成) セクションで、[New API] (新規 API) が選択されていることを確認します。
4. 以下のフィールドを設定します。
   - API 名: opensearch-api
   - 説明: Amazon OpenSearch Service ドメイン検索用のパブリック API
   - エンドポイントタイプ: リージョン別
5. [API の作成] を選択します。
6. [アクション] および [メソッドの作成] を選択します。
7. ドロップダウンで [GET] を選択し、チェックマークをクリックして確定します。
8. 以下の設定を行い、[保存] を選択します。

設定	値
統合タイプ	Lambda function
Lambda プロキシ統合の使用	はい
Lambda のリージョン	*us-west-1*
Lambda function	opensearch-lambda
デフォルトタイムアウトの使用	はい

メソッドリクエストの設定

[メソッドリクエスト] を選択して、以下の設定を行います。

設定	値
Authorization	なし
リクエストの検証	クエリ文字列パラメータ、およびヘッダーの検証
API キーが必要です	false

[URL Query String Parameters] (URL クエリ文字列パラメータ) の下で [Add query string] (クエリ文字列を追加) を選択し、次のパラメータを設定します。

設定	値
名前	q
必須	はい

API のデプロイとステージの設定

API Gateway コンソールでは、デプロイを作成して新規または既存のステージに関連付けることで API をデプロイできます。

1. [アクション] および [API のデプロイ] を選択します。
2. [デプロイステージ] では、[新しいステージ] を選択し、ステージ `opensearch-api-test` に名前を付けます。
3. [デプロイ] をクリックします。
4. ステージエディタで以下の設定を行い、[変更の保存] を選択します。

設定	値
スロットリングの有効化	はい
Rate	1,000
バースト	500

これらの設定は、エンドポイントのルート (`https://`*`some-id`*`.execute-api.`*`us-west-1`*`.amazonaws.com/search-es-api-test`) への GET リクエストという 1 つのメソッドのみを持つ API を設定します。リクエストに必要なのは、検索するクエリ文字列という 1 つのパラメータ (q) です。呼び出されると、メソッドは `opensearch-lambda` 関数を実行する Lambda にリクエストを渡します。詳細については、「Amazon API Gateway での API の作成」および「Amazon API Gateway での REST API のデプロイ」を参照してください。

ステップ 4 (オプション): ドメインアクセスポリシーを変更する

OpenSearch Service ドメインで、Lambda 関数が `movies` インデックスへの `GET` リクエストを行うことを許可する必要があります。きめ細かなアクセスコントロールが有効になっているドメインで未処理のアクセスポリシーがある場合は、そのまま使用できます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-west-1:123456789012:domain/domain-name/*"
    }
  ]
}
```

または、所持しているドメインアクセスポリシーをよりきめ細かにすることもできます。例えば、次の最小限のポリシーは、`movies` インデックスへの `opensearch-lambda-role` (Lambda を通じて作成された) 読み取りアクセスを提供します。Lambda が自動的に作成するロールの正確な名前を取得するには、 AWS Identity and Access Management (IAM) コンソールに移動し、ロールを選択し、「lambda」を検索します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/service-role/opensearch-lambda-
role-1abcdefg"
      },
```

```
      "Action": "es:ESHttpGet",
      "Resource": "arn:aws:es:us-west-1:123456789012:domain/domain-name/movies/
_search"
    }
  ]
}
```

> **Important**
>
> きめ細かなアクセスコントロールがドメインに対して有効になっている場合は、OpenSearch Dashboards でロールをユーザーにマッピングする必要もあります。マッピングを行わないと、アクセス許可エラーが表示されます。

Lambda 実行ロールのアクセス許可の設定

ドメインアクセスポリシーの設定に加えて、Lambda 実行ロールに OpenSearch Service ドメインにアクセスするために必要な IAM アクセス許可があることを確認する必要があります。Lambda 関数には、マネージドドメインと OpenSearch Service Serverless コレクションのどちらを使用しているかに応じて、特定のアクセス許可が必要です。

マネージド OpenSearch Service ドメインの場合:

以下の IAM ポリシーを Lambda 実行ロールにアタッチして、OpenSearch Service ドメインへのリクエストを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttpGet",
        "es:ESHttpPost"
      ],
      "Resource": "arn:aws:es:us-west-1:123456789012:domain/domain-name/*"
    }
```

```
  ]
}
```

OpenSearch Service Serverless コレクションの場合:

OpenSearch Service Serverless を使用している場合は、以下の IAM ポリシーを Lambda 実行ロールにアタッチします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "aoss:*",
      "Resource": "arn:aws:aoss:us-west-1:123456789012:collection/collection-id"
    }
  ]
}
```

Lambda 実行ロールにこれらのポリシーをアタッチするには:

1. IAM コンソール (https://console.aws.amazon.com/iam/) に移動します。
2. [ロール] を選択し、Lambda 実行ロールを検索します (通常は `opensearch-lambda-role-xxxxxxxx` という名前です)。
3. [アクセス許可の追加] を選択し、次いで [インラインポリシーを作成] を選択します。
4. [JSON] タブを選択し、上記から適切なポリシーを貼り付けて、プレースホルダー値を実際のリソース ARN に置き換えます。
5. [ポリシーの確認] を選択し、`OpenSearchAccess` のような名前を入力して、[ポリシーの作成] を選択します。

> **Note**
>
> 上記の IAM アクセス許可がない場合、ドメインアクセスポリシーでリクエストが許可されている場合でも、Lambda 関数が OpenSearch Service ドメインにクエリしようとすると「アクセスが拒否されました」のエラーが発生します。

アクセスポリシーの詳細については、「the section called “アクセスポリシーの設定”」を参照してください。

Lambda ロールをマッピングする (きめ細かなアクセスコントロールを使用している場合)

きめ細かなアクセスコントロールでは、アプリケーションをテストする前に追加のステップが導入されます。HTTP 基本認証を他のすべての目的で使用する場合でも、ユーザーに Lambda ロールをマッピングする必要があります。マッピングを行わないと、アクセス許可エラーが表示されます。

1. ドメインの OpenSearch Dashboards URL に移動します。
2. メインメニューから [Security] (セキュリティ)、[Roles] (ロール) の順に選択し、Lambda ロールをマッピングするロールである `all_access` へのリンクを選択します。
3. [マッピングされたユーザー]、[マッピングの管理] を選択します。
4. [Backend roles] (バックエンドロール) で、Lambda ロールの Amazon リソースネーム (ARN) を追加します。ARN は `arn:aws:iam::123456789123:role/service-role/opensearch-lambda-role-1abcdefg` の形式にします。
5. [マップ] を選択し、ユーザーまたはロールが [マッピングされたユーザー] の下に表示されていることを確認します。

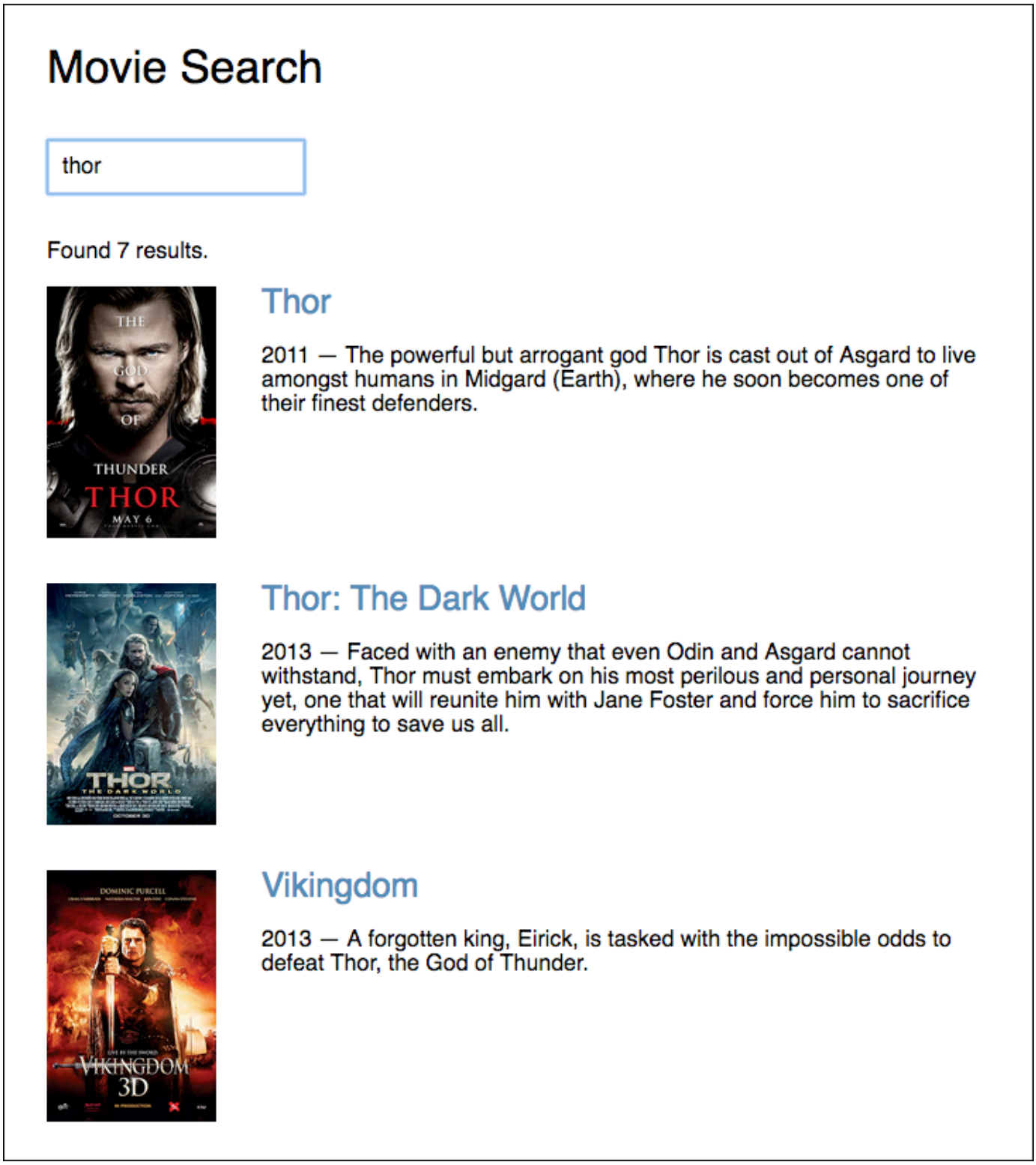
ステップ 5: ウェブアプリケーションをテストする

ウェブアプリケーションをテストするには

1. sample-site.zip をダウンロードして解凍し、お気に入りのテキストエディタで `scripts/search.js` を開きます。
2. API Gateway エンドポイントを指すように `apigatewayendpoint` 変数を更新し、所定のパスの最後にバックスラッシュを追加します。[Stages] (ステージ) と API の名前を選択することで、API ゲートウェイでエンドポイントをすばやく見つけることが

できます。apigatewayendpoint 変数は https://*some-id*.execute-api.*us-west-1*.amazonaws.com/opensearch-api-test/ の形式にします。

3. index.html を開き、thor、house、および他のいくつかの用語の検索を実行してみてください。



CORS エラーのトラブルシューティング

Lambda 関数で CORS をサポートするための応答にコンテンツが含まれていても、次のエラーが表示される場合があります。

```
Access to XMLHttpRequest at '<api-gateway-endpoint>' from origin 'null' has been
 blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present in the
 requested resource.
```

このような場合は、以下のことを試してみます。

1. GET リソースで CORS を有効化します。[Advanced] (アドバンスト) の下で、[Access-Control-Allow-Credentials] を `'true'` にセットします。
2. API ゲートウェイ ([Actions] (アクション)、[Deploy API] (API のデプロイ)) で API を再デプロイします。
3. Lambda 関数トリガーを削除して再追加します。[re-add] (再追加) を追加し、[Add trigger] (トリガーを追加) を選択して、関数を呼び出す HTTP エンドポイントを作成します。トリガーには、次の設定がある必要があります。

トリガー	API	デプロイされるステージ	セキュリティ
API ゲートウェイ	opensearch-api	opensearch-api-test	開く

次の手順

この章は、概念を示すための出発点にすぎません。例えば、次のような変更が考えられます。

- OpenSearch Service ドメインに独自のデータを追加します。
- API にメソッドを追加します。
- Lambda 関数では、検索クエリを変更するか、異なるフィールドを追加します。
- 結果のスタイルを変更するか、`search.js` を変更してユーザーに異なるフィールドを表示します。

チュートリアル: OpenSearch Service と OpenSearch Dashboards によるカスタマーサポートへの問い合わせを可視化する

この章は、次のような状況の完全なチュートリアルです。ビジネスである程度の数のカスタマーサポートコールを受けており、それらを分析したいと考えています。各問い合わせの件名は何でしょうか? 肯定的なやり取りの数はいくつでしょうか? 否定的なやり取りの数はいくつでしょうか? マネージャーはこれらの問い合わせのトランスクリプトをどのように検索または確認することができますか?

手動ワークフローでは、従業員が通話記録を聴き、各問い合わせの件名をメモし、顧客とのやり取りが肯定的であったかどうかを判断することが考えられます。

このようなプロセスには非常に大きな労力がかかります。1 回の問い合わせの平均時間が 10 分とすると、各従業員が 1 日あたり聴くことができる問い合わせの数は 48 件にすぎません。人間の先入観や偏見を除外することで、生成されるデータは非常に正確なものとなる一方で、データの量は最小限になります。つまり、問い合わせの件名と、顧客が満足したかどうかのブール値のみとなります。完全なトランスクリプトなど、それ以上の内容を伴う場合、非常に長い時間がかかる可能性があります。

Amazon S3、Amazon Transcribe、Amazon Comprehend、および Amazon OpenSearch Service を使用することで、ごくわずかなコードで同様のプロセスを自動化し、はるかに多くのデータを得ることができます。例えば、問い合わせの完全なトランスクリプト、トランスクリプトからのキーワード、および問い合わせの全体的な「センチメント」(肯定的、否定的、中立、混在) を取得できます。次に、OpenSearch と OpenSearch Dashboards を使用して、データを検索して可視化できます。

このチュートリアルはそのとおりに使用できますが、その意図は、OpenSearch Service でインデックス化する前に、JSON ドキュメントを強化する方法に関するアイデアを生み出すことです。

推定コスト

一般的に、このチュートリアルのステップを実行するコストは 2 USD 未満です。このチュートリアルでは、以下のリソースを使用します。

- 転送および保存されるデータが 100 MB 未満である S3 バケット

  詳細については、「Amazon S3 料金表」を参照してください。
- 1 つの `t2.medium` インスタンスを持つ OpenSearch Service ドメインと、数時間に対応する 10 GiB の EBS ストレージ

  詳細については、「Amazon OpenSearch Service 料金表」を参照してください。

- Amazon Transcribe への数回の呼び出し

  詳細については、「Amazon Transcribe 料金表」を参照してください。
- Amazon Comprehend への数回の自然言語処理の呼び出し

  詳細については、「Amazon Comprehend 料金表」を参照してください。

トピック

- ステップ 1: 前提条件を設定する
- ステップ 2: サンプルコードをコピーする
- (オプション) ステップ 3: サンプルデータのインデックスを作成する
- ステップ 4: データを分析し、可視化する
- ステップ 5: リソースのクリーンアップと次のステップ

ステップ 1: 前提条件を設定する

続行する前に、以下のリソースが必要です。

前提条件	説明
Amazon S3 バケット	詳細については、Amazon Simple Storage Service コンソールユーザーガイドの「バケットの作成」を参照してください。
OpenSearch Service ドメイン	データのコピー先。詳細については、「OpenSearch Service ドメインの作成」を参照してください。

これらのリソースがない場合は、次の AWS CLI コマンドを使用して作成できます。

```
aws s3 mb s3://my-transcribe-test --region us-west-2
```

```
aws opensearch create-domain --domain-name my-transcribe-test --engine-version
 OpenSearch_1.0 --cluster-config  InstanceType=t2.medium.search,InstanceCount=1
 --ebs-options EBSEnabled=true,VolumeType=standard,VolumeSize=10 --access-policies
 '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::123456789012:root"},"Action":"es:*","Resource":"arn:aws:es:us-
west-2:123456789012:domain/my-transcribe-test/*"}]}' --region us-west-2
```

> **Note**
>
> これらのコマンドでは us-west-2 リージョンを使用しますが、Amazon Comprehend がサポートしている任意のリージョンを使用することもできます。詳細については、AWS 全般のリファレンス を参照してください。

ステップ 2: サンプルコードをコピーする

1. 次の Python 3 サンプルコードをコピーし、call-center.py という新しいファイルに貼り付けます。

```
import boto3
import datetime
import json
import requests
from requests_aws4auth import AWS4Auth
import time
import urllib.request

# Variables to update
audio_file_name = '' # For example, 000001.mp3
bucket_name = '' # For example, my-transcribe-test
domain = '' # For example, https://search-my-transcribe-test-12345.us-
west-2.es.amazonaws.com
index = 'support-calls'
type = '_doc'
region = 'us-west-2'

# Upload audio file to S3.
s3_client = boto3.client('s3')

audio_file = open(audio_file_name, 'rb')

print('Uploading ' + audio_file_name + '...')
response = s3_client.put_object(
    Body=audio_file,
    Bucket=bucket_name,
    Key=audio_file_name
)

# # Build the URL to the audio file on S3.
```

```
# # Only for the us-east-1 region.
# mp3_uri = 'https://' + bucket_name + '.s3.amazonaws.com/' + audio_file_name

# Get the necessary details and build the URL to the audio file on S3.
# For all other regions.
response = s3_client.get_bucket_location(
    Bucket=bucket_name
)
bucket_region = response['LocationConstraint']
mp3_uri = 'https://' + bucket_name + '.s3-' + bucket_region + '.amazonaws.com/' +
 audio_file_name

# Start transcription job.
transcribe_client = boto3.client('transcribe')

print('Starting transcription job...')
response = transcribe_client.start_transcription_job(
    TranscriptionJobName=audio_file_name,
    LanguageCode='en-US',
    MediaFormat='mp3',
    Media={
        'MediaFileUri': mp3_uri
    },
    Settings={
        'ShowSpeakerLabels': True,
        'MaxSpeakerLabels': 2 # assumes two people on a phone call
    }
)

# Wait for the transcription job to finish.
print('Waiting for job to complete...')
while True:
    response =
 transcribe_client.get_transcription_job(TranscriptionJobName=audio_file_name)
    if response['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    else:
        print('Still waiting...')
    time.sleep(10)

transcript_uri = response['TranscriptionJob']['Transcript']['TranscriptFileUri']

# Open the JSON file, read it, and get the transcript.
```

```
response = urllib.request.urlopen(transcript_uri)
raw_json = response.read()
loaded_json = json.loads(raw_json)
transcript = loaded_json['results']['transcripts'][0]['transcript']

# Send transcript to Comprehend for key phrases and sentiment.
comprehend_client = boto3.client('comprehend')

# If necessary, trim the transcript.
# If the transcript is more than 5 KB, the Comprehend calls fail.
if len(transcript) > 5000:
    trimmed_transcript = transcript[:5000]
else:
    trimmed_transcript = transcript

print('Detecting key phrases...')
response = comprehend_client.detect_key_phrases(
    Text=trimmed_transcript,
    LanguageCode='en'
)

keywords = []
for keyword in response['KeyPhrases']:
    keywords.append(keyword['Text'])

print('Detecting sentiment...')
response = comprehend_client.detect_sentiment(
    Text=trimmed_transcript,
    LanguageCode='en'
)

sentiment = response['Sentiment']

# Build the Amazon OpenSearch Service URL.
id = audio_file_name.strip('.mp3')
url = domain + '/' + index + '/' + type + '/' + id

# Create the JSON document.
json_document = {'transcript': transcript, 'keywords': keywords, 'sentiment':
 sentiment, 'timestamp': datetime.datetime.now().isoformat()}

# Provide all details necessary to sign the indexing request.
credentials = boto3.Session().get_credentials()
```

```
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region,
 'opensearchservice', session_token=credentials.token)

# Index the document.
print('Indexing document...')
response = requests.put(url, auth=awsauth, json=json_document, headers=headers)

print(response)
print(response.json())
```

2. 最初の 6 つの変数を更新します。
3. 次のコマンドを使用して必要なパッケージをインストールします。

```
pip install boto3
pip install requests
pip install requests_aws4auth
```

4. call-center.py と同じディレクトリに MP3 を配置し、スクリプトを実行します。サンプル出力を次に示します。

```
$ python call-center.py
Uploading 000001.mp3...
Starting transcription job...
Waiting for job to complete...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Detecting key phrases...
Detecting sentiment...
Indexing document...
<Response [201]>
{u'_type': u'call', u'_seq_no': 0, u'_shards': {u'successful': 1, u'failed': 0,
 u'total': 2}, u'_index': u'support-calls4', u'_version': 1, u'_primary_term': 1,
 u'result': u'created', u'_id': u'000001'}
```

call-center.py は多数のオペレーションを実行します。

1. このスクリプトは、音声ファイル (この例では MP3 ですが、Amazon Transcribe は複数の形式をサポートします) を S3 バケットにアップロードします。
2. オーディオファイルの URL を Amazon Transcribe に送信し、書き起こしジョブの完了を待機します。

   書き起こしジョブが完了するまでの時間は、オーディオファイルの長さによって異なります。数秒ではなく、数分であると想定してください。

   > **Tip**
   >
   > 書き起こしの品質を向上させるため、Amazon Transcribe 用のカスタム語彙を設定できます。

3. 書き起こしジョブが完了すると、スクリプトはトランスクリプトを抽出し、それを 5,000 文字に切り捨て、キーワードとセンチメントの分析のため Amazon Comprehend に送信します。
4. 最後に、スクリプトは完全なトランスクリプト、キーワード、センチメント、および現在のタイムスタンプを JSON ドキュメントに追加し、そのインデックスを OpenSearch Service に作成します。

> **Tip**
>
> LibriVox には、テストに使用できるパブリックドメインオーディオブックがあります。

(オプション) ステップ 3: サンプルデータのインデックスを作成する

多くの通話記録が手元にない場合 (これは一般的です)、sample-calls.zip にサンプルドキュメントのインデックスを作成できます。これは `call-center.py` で生成されるものと同等です。

1. `bulk-helper.py` という名前のファイルを作成します :

```
import boto3
from opensearchpy import OpenSearch, RequestsHttpConnection
import json
from requests_aws4auth import AWS4Auth

host = '' # For example, my-test-domain.us-west-2.es.amazonaws.com
region = '' # For example, us-west-2
```

```
service = 'es'

bulk_file = open('sample-calls.bulk', 'r').read()

credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

search = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)

response = search.bulk(bulk_file)
print(json.dumps(response, indent=2, sort_keys=True))
```

2. host および region の最初の 2 つの変数を更新します。
3. 次のコマンドを使用して必要なパッケージをインストールします。

```
pip install opensearch-py
```

4. sample-calls.zip をダウンロードして解凍します。
5. sample-calls.bulk と同じディレクトリに bulk-helper.py を配置し、ヘルパーを実行します。サンプル出力を次に示します。

```
$ python bulk-helper.py
{
  "errors": false,
  "items": [
    {
      "index": {
        "_id": "1",
        "_index": "support-calls",
        "_primary_term": 1,
        "_seq_no": 42,
        "_shards": {
          "failed": 0,
          "successful": 1,
          "total": 2
```

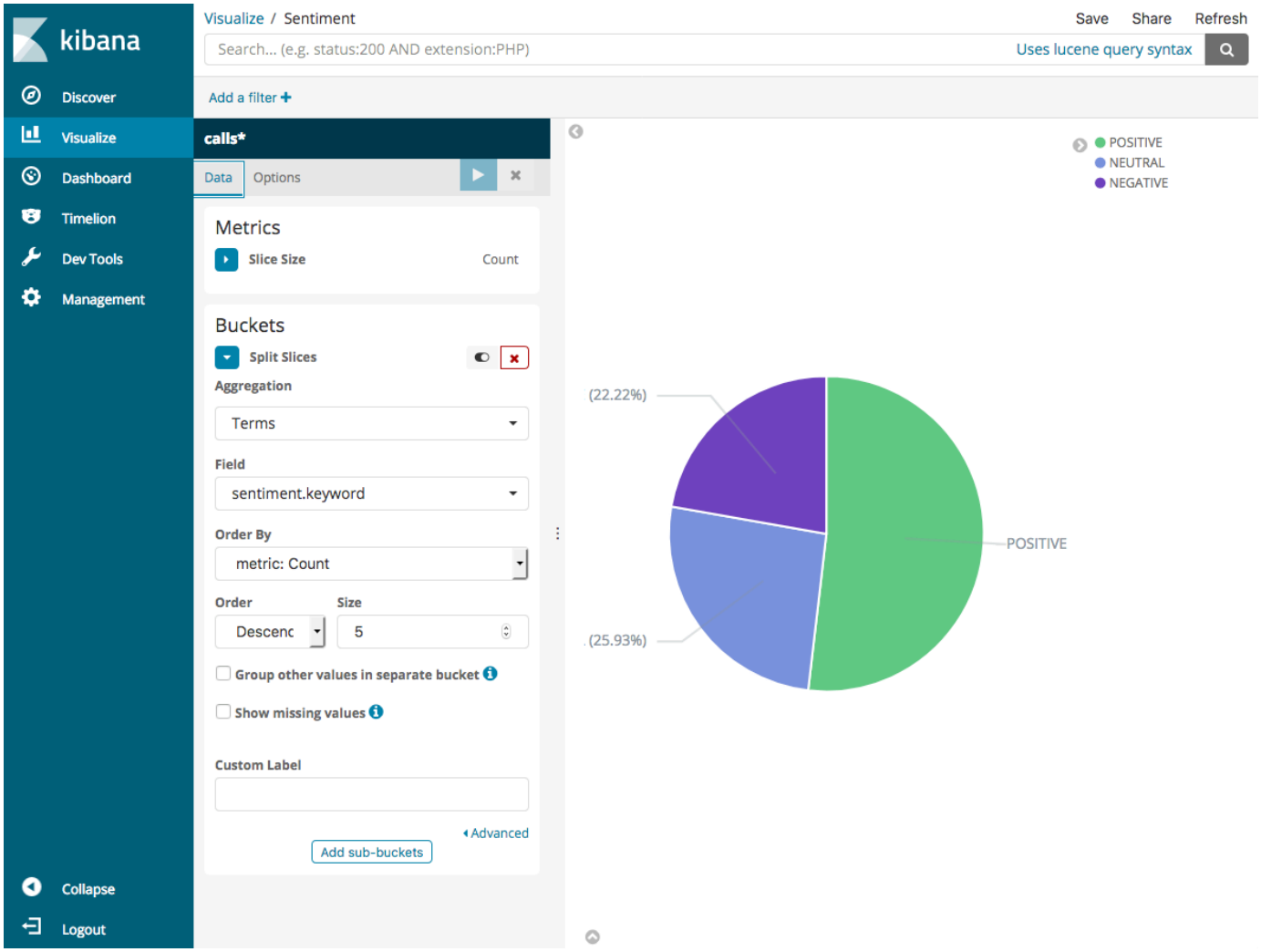
```
        },
        "_type": "_doc",
        "_version": 9,
        "result": "updated",
        "status": 200
      }
    },
    ...
  ],
  "took": 27
}
```

ステップ 4: データを分析し、可視化する

OpenSearch Service にデータを配置したので、OpenSearch Dashboards を使用してそのデータを可視化することができます。

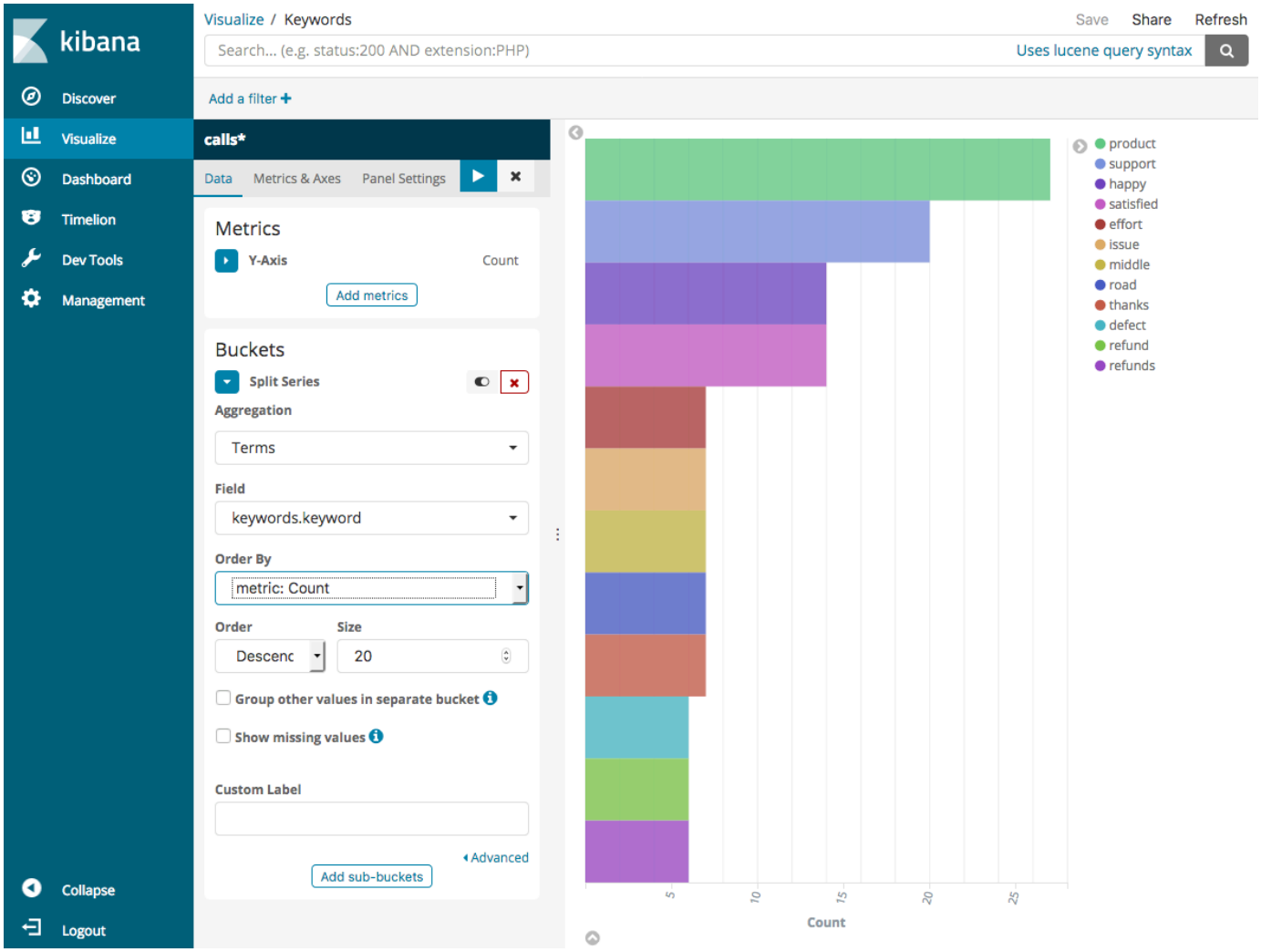
1. `https://search-`*`domain`*`.`*`region`*`.es.amazonaws.com/_dashboards` に移動します。
2. OpenSearch Dashboards を使用するには、インデックスパターンが必要です。Dashboards はインデックスパターンを使用して、分析を 1 つまたは複数のインデックスに絞り込みます。`call-center.py` が作成した `support-calls` インデックスと一致させるには、[スタックの管理]、[インデックスパターン] に移動し、`support*` のインデックスパターンを定義してから、[次のステップ] を選択します。
3. [タイムフィルターフィールド名] で、[タイムスタンプ] を選択します。
4. これで、可視化の作成を開始できます。[可視化] を選択し、新しい視覚化を追加します。
5. 円グラフと `support*` インデックスパターンを選択します。
6. デフォルトの可視化は基本的です。そこで、より魅力的な視覚化を作成するため、[分割スライス] を選択します。

   [集約] で、[用語] を選択します。[フィールド] で、[sentiment.keyword] を選択します。次に、[変更の適用] を選択し、[保存] を選択します。

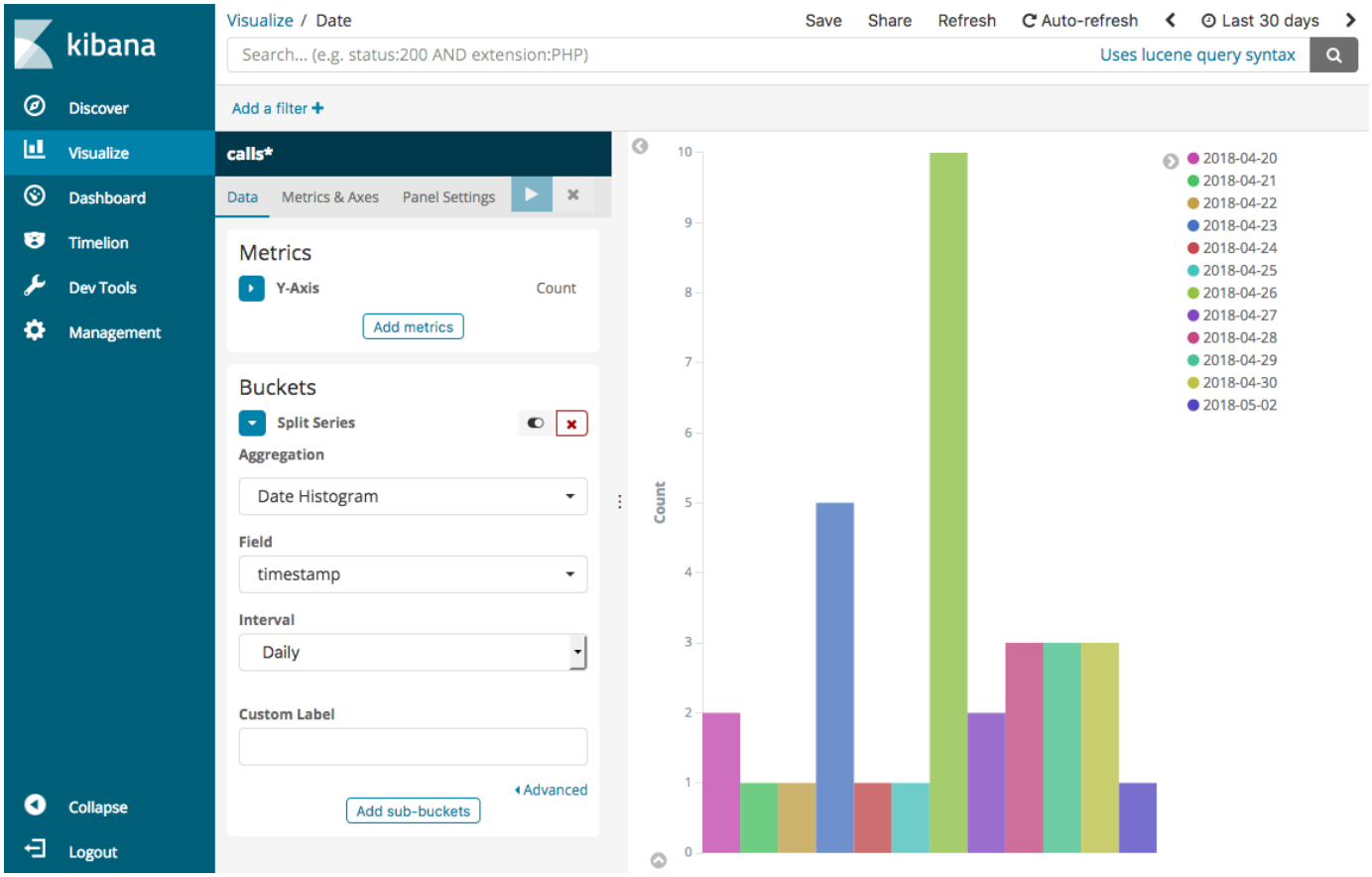


7. [可視化] ページに戻り、別の可視化を追加します。今回は、水平棒グラフを選択します。
8. [シリーズの分割] を選択します。

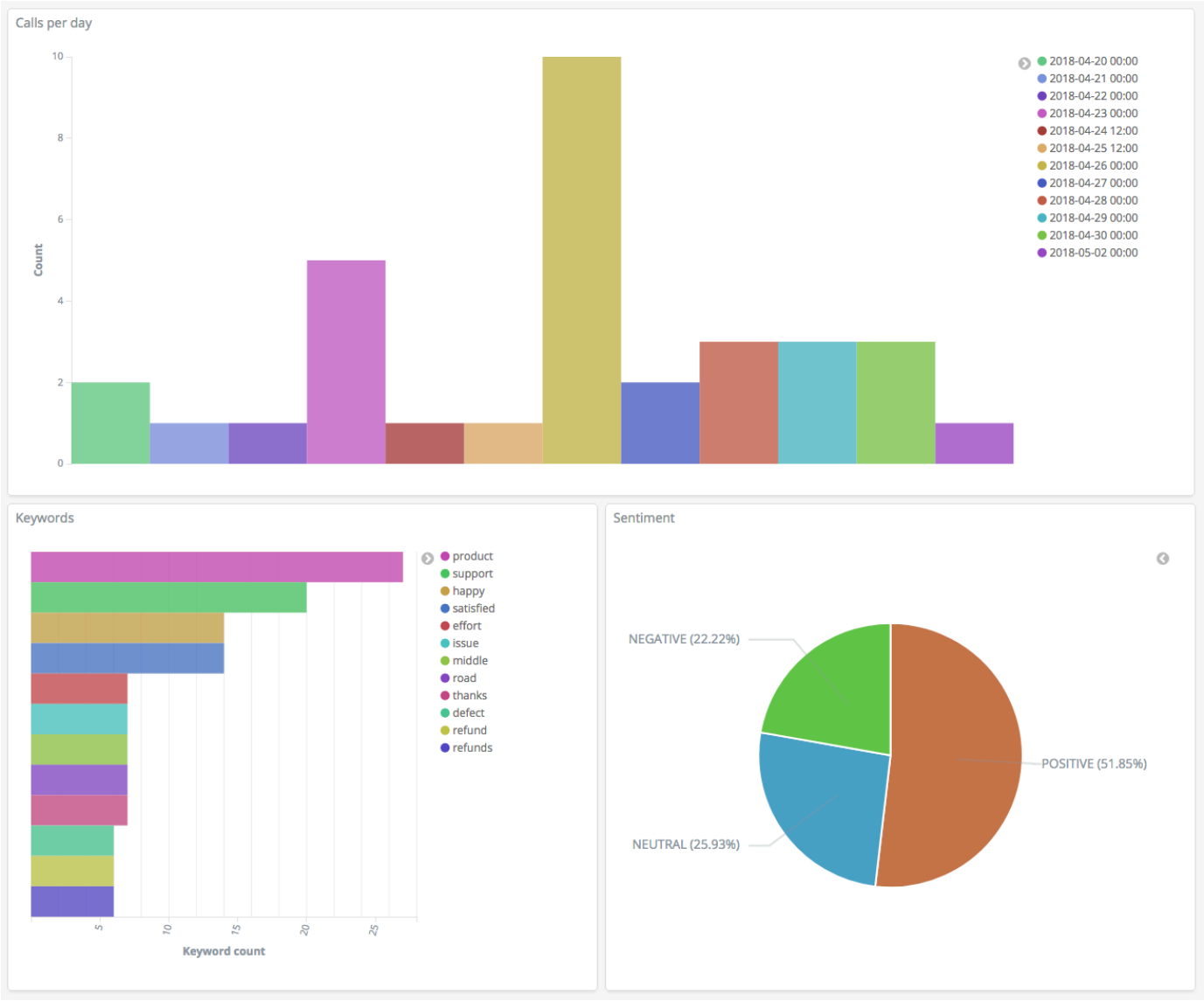
   [集約] で、[用語] を選択します。[フィールド] で、[keywords.keyword] を選択し、[サイズ] を 20 に変更します。次に、[変更の適用] を選択し、[保存] を選択します。



9. [可視化] ページに戻り、最終的な 1 つの視覚化である垂直棒グラフを追加します。
10. [シリーズの分割] を選択します。[集約] で、[日付ヒストグラム] を選択します。[フィールド] の [タイムスタンプ] を選択し、[間隔] を [日別] に設定します。
11. [メトリクス & 軸] を選択し、[モード] を [法線] に変更します。
12. [変更の適用] を選択し、[保存] を選択します。



13. これで 3 つの可視化を作成したので、Dashboards ダッシュボードに追加することができます。[ダッシュボード] を選択し、ダッシュボードを作成して、可視化を追加します。



ステップ 5: リソースのクリーンアップと次のステップ

不要な料金が発生しないようにするため、S3 バケットおよび OpenSearch Service ドメインを削除します。詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの削除」およびこのガイドの「OpenSearch Service ドメインを削除する」を参照してください。

トランスクリプトは、MP3 ファイルよりもはるかに少ないディスク容量で済みます。MP3 の保持期間を短くする (例えば、通話記録の保持期間を 3 か月から 1 か月にする) ことも、数年間のトランスクリプトを保持することもでき、いずれの場合もストレージコストを低減できます。

また、 AWS Step Functions と Lambda を使用して文字起こしプロセスを自動化したり、インデックス作成前にメタデータを追加したり、正確なユースケースに合わせてより複雑な視覚化を作成したりできます。

Amazon OpenSearch Service 名称変更 - 変更の概要

2021 年 9 月 8 日に、検索および分析スイートの名称が Amazon OpenSearch Service に変更されました。OpenSearch Service は、OpenSearch だけでなく、従来の Elasticsearch もサポートします。以下のセクションでは、名称変更によって変更されたサービスのさまざまな部分と、ドメインが正常に機能し続けるために実行する必要があるアクションについて説明します。

これらの変更の一部は、ドメインを Elasticsearch から OpenSearch にアップグレードするときにのみ適用されます。請求およびコストマネジメントコンソールなど、エクスペリエンスは速やかに変更されます。

これはすべてを網羅したリストではないことに注意してください。製品の他の部分も変更されましたが、これらの更新が最も関連性があります。

API の新バージョン

OpenSearch Service 設定API の新バージョン (2021-01-01) は、従来の Elasticsearch OSS と同様に OpenSearch で動作します。21 の API オペレーションは、より簡潔でエンジンに依存しない名前に置き換えられました (例えば、`CreateElasticsearchDomain` は `CreateDomain` に変更されました) が、OpenSearch Service は引き続き両方の API バージョンをサポートしています。

今後は、新しい API オペレーションを使用して、ドメインを作成および管理することをお勧めします。新しい API オペレーションを使用してドメインを作成するときは、バージョン番号だけではなく、形式 `Elasticsearch_X.Y` または `OpenSearch_X.Y` で `EngineVersion` パラメータを指定する必要があります。バージョンを指定しない場合、デフォルトは OpenSearch の最新バージョンになります。

を使用してドメインを作成および管理するには、 `aws opensearch ...`を AWS CLI バージョン 1.20.40 以降にアップグレードします。新しい CLI 形式については、「OpenSearch CLI リファレンス」を参照してください。

名称変更されたインスタンスタイプ

Amazon OpenSearch Service のインスタンスタイプは形式 `<type>.<size>.search` になりました (例えば、`m6g.large.`*`elasticsearch`* ではなく`m6g.large.`*`search`*)。ご自身では特に何もする必要はありません。既存のドメインは、API 内と、請求およびコストマネジメントコンソール内で新しいインスタンスタイプを自動的に参照し始めます。

リザーブドインスタンス (RI)がある場合、契約は変更の影響を受けません。古い設定 API バージョンはまだ古い名前付け形式と互換性がありますが、新しい API バージョンを使用する場合は、新しい形式を使用する必要があります。

アクセスポリシーの変更

次のセクションでは、アクセスポリシーを更新するために必要なアクションについて説明します。

IAM ポリシー

名称変更された API オペレーションを使用するように IAM ポリシーを更新することをお勧めします。ただし、OpenSearch Service は、古い API 許可を内部で複製することで、既存のポリシーを引き続き優先します。例えば、`CreateElasticsearchDomain` オペレーションを実行するための許可を現在持っている場合、`CreateElasticsearchDomain` (古い API オペレーション) と`CreateDomain` (新しい API オペレーション) 両方への呼び出しを行えるようになりました。同じことが明示的拒否に適用されます。更新された API オペレーションのリストについては、「ポリシーエレメントリファレンス」を参照してください。

SCP ポリシー

サービスコントロールポリシー (SCP)では、標準の IAM に比べてさらに複雑な追加のレイヤーが導入されています。SCP ポリシーを中断しないようにするには、古いおよび新しい API オペレーションの両方を各 SCP ポリシーに追加する必要があります。例えば、ユーザーが現在 `CreateElasticsearchDomain` の許可権限を持っている場合、彼らがドメインを作成する能力を保持できるように、`CreateDomain` の許可権限も彼らに付与する必要があります。同じことが明示的拒否に適用されます。

例:

```
"Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:CreateElasticsearchDomain",
        "es:CreateDomain"
          ...
      ],
    },
      "Effect": "Deny",
```

```
      "Action:" [
        "es:DeleteElasticsearchDomain",
        "es:DeleteDomain"
         ...
```

新しいリソースタイプ

OpenSearch Service では、次の新しいリソースタイプが導入されました。

[リソース]	説明
`AWS::OpenSearchService::Domain`	Amazon OpenSearch Service ドメインを表します。このリソースはサービスレベルに存在し、ドメインで実行されているソフトウェアに固有のものではありません。これは、CloudFormation および AWS Resource Groups のようなサービスに適用されます。そこで、サービス全体のリソースが作成および管理されます。 CloudFormation 内で定義されたドメインを Elasticsearch から OpenSearch にアップグレードする手順については、CloudFormation ユーザーガイドの解説を参照してください。
`AWS::OpenSearch::Domain`	ドメインで実行されている OpenSearch/ElasticSearch ソフトウェアを表します。このリソースは AWS CloudTrail および AWS Config のようなサービスに適用されます。これは、OpenSearch Service 全体ではなくドメイン上で実行しているソフトウェアを参照しています。これらのサービスには、OpenSearch を実行するドメイン (`AWS::OpenSearch::Domain`) に対して Elasticsearch を実行するドメイン (`AWS::Elasticsearch::Domain`) 用の個別のリソースタイプが含まれるようになりました。

> **Note**
>
> AWS Config で、1 つ以上のドメインを OpenSearch にアップグレードした場合でも、数週間は既存の `AWS::Elasticsearch::Domain` リソースタイプの下で引き続きデータを確認できます。

Kibana は OpenSearch Dashboards に名称変更されました

Kibana AWS に代わる OpenSearch Dashboards は、OpenSearch と連携するように設計されたオープンソースの視覚化ツールです。ドメインを Elasticsearch から OpenSearch にアップグレードすると、`/_plugin/kibana` エンドポイントは `/_dashboards` に変わります。OpenSearch Service はすべてのリクエストを新しいエンドポイントにリダイレクトしますが、いずれかの IAM ポリシーで Kibana エンドポイントを使用している場合は、新しい `/_dashboards` エンドポイントも含めるように、それらのポリシーを更新します。

the section called “OpenSearch Dashboards の SAML 認証” を使用している場合は、ドメインを OpenSearch にアップグレードする前に、ID プロバイダー (IdP) によって設定されたすべての Kibana URL を `/_plugin/kibana` から `/_dashboards` に変更する必要があります。最も一般的な URL は、Assertion Consumer Service (ACS) URL と受信者 URL です。

デフォルトの OpenSearch Dashboards の `kibana_read_only` ロールが `opensearch_dashboards_read_only`、`kibana_user` ロールが `opensearch_dashboards_user` に名称変更されました。この変更は、サービスソフトウェア R20211203 以降を実行して新しく作成されるすべての OpenSearch 1.x ドメインに適用されます。既存のドメインをサービスソフトウェア R20211203 にアップグレードしても、ロール名は変わりません。

名称変更された CloudWatch メトリクス

OpenSearch を実行するドメインでは、いくつかの CloudWatch メトリックスが変更されます。ドメインを OpenSearch にアップグレードすると、メトリクスが自動的に変更され、現在の CloudWatch アラームが中断されます。クラスターを Elasticsearch バージョンから OpenSearch バージョンにアップグレードする前に、新しいメトリクスを使用するように CloudWatch アラームを更新してください。

以下のメトリクスが変更されました。

元のメトリクス名	新しい名称
KibanaHealthyNodes	OpenSearchDashboardsHealthyNodes
KibanaConcurrentConnections	OpenSearchDashboardsConcurrentConnections
KibanaHeapTotal	OpenSearchDashboardsHeapTotal
KibanaHeapUsed	OpenSearchDashboardsHeapUsed
KibanaHeapUtilization	OpenSearchDashboardsHeapUtilization
KibanaOS1MinuteLoad	OpenSearchDashboardsOS1MinuteLoad
KibanaRequestTotal	OpenSearchDashboardsRequestTotal
KibanaResponseTimesMaxInMillis	OpenSearchDashboardsResponseTimesMaxInMillis
ESReportingFailedRequestSysErrCount	KibanaReportingFailedRequestSysErrCount
ESReportingRequestCount	KibanaReportingRequestCount
ESReportingFailedRequestUserErrCount	KibanaReportingFailedRequestUserErrCount
ESReportingSuccessCount	KibanaReportingSuccessCount
ElasticsearchRequests	OpenSearchRequests

OpenSearch Service が Amazon CloudWatch に送信するメトリクスの完全なリストについては、「the section called “クラスターメトリクスのモニタリング”」を参照してください。

請求およびコストマネジメントコンソールの変更

請求およびコストマネジメントコンソール内およびコストと使用状況レポート内の履歴データは引き続き古いサービス名を使用するため、データを検索するときは、[Amazon OpenSearch Service] と従来の Elasticsearch の名前の両方でフィルターの使用をスタートする必要があります。既存の保存済みレポートがある場合は、フィルターを更新して、OpenSearch Service も含まれていることを確認します。Elasticsearch の使用量が減少し、OpenSearch の使用量が増加すると、最初はアラートを受け取ることがありますが、それは数日以内に消えます。

サービス名の変更に加えて、次のフィールドは、すべてのレポート、請求書、および価格表の API オペレーションについて変更されます。

フィールド	古い形式	新しい形式
インスタンスタイプ	`m5.large.elasticsearch`	`m5.large.search`
製品ファミリー	Elasticsearch インスタンス Elasticsearch ボリューム	Amazon OpenSearch Service インスタンス Amazon OpenSearch Service ボリューム
料金の説明	c5.18xlarge.elasticsearch インスタンス時間 (または時間の端数分) あたり5.098 USD - EU	c5.18xlarge.search インスタンス時間 (または時間の端数分) あたり 5.098 USD - EU
インスタンスファミリー	`ultrawarm.elasticsearch`	`ultrawarm.search`

新しいイベント形式

OpenSearch サービスが Amazon EventBridge および Amazon CloudWatch に送信するイベントの形式が変更されました。具体的には、`detail-type` フィールドです。ソースフィールド (`aws.es`) は同じままです。各イベントタイプの完全な形式については、「the section called “イベントのモニタリング”」を参照してください。古い形式に依存する既存のイベントルールがある場合は、新しい形式に準拠するように更新してください。

同じままのものは何ですか？

以下の特徴と機能は、特に記載されていませんが、同じままです。

- サービスプリンシパル (es.amazonaws.com)
- ベンダーコード
- ドメイン ARN
- ドメインエンドポイント

使用開始: ドメインを OpenSearch 1.x にアップグレードします

OpenSearch 1.x は Elasticsearch バージョン 6.8 および 7.x からのアップグレードをサポートします。ドメインをアップグレードする手順については、「the section called “ドメインのアップグレード (コンソール)”」を参照してください。 AWS CLI または 設定 API を使用してドメインをアップグレードする場合は、 を TargetVersionとして指定する必要がありますOpenSearch_1.x。

OpenSearch 1.x では、[互換モードを有効にする] という追加のドメイン設定を導入しています。特定の Elasticsearch OSS クライアントおよびプラグインが、接続する前にクラスターのバージョンをチェックするので、互換モードでは OpenSearch がそのバージョンを 7.10 とレポートするように設定され、これらのクライアントが引き続き動作するようにします。

OpenSearch ドメインを初めて作成するとき、またはElasticsearch バージョンから OpenSearch バージョンにアップグレードするときに、互換モードを有効にできます。それが設定されていない場合、パラメータはデフォルトで、ドメインの作成時は false に、ドメインのアップグレード時は true になります。

設定 API を使用して互換モードを有効にするには、override_main_response_version を true に設定します。

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/upgradeDomain
{
  "DomainName": "domain-name",
  "TargetVersion": "OpenSearch_1.0",
  "AdvancedOptions": {
    "override_main_response_version": "true"
   }
}
```

既存の OpenSearch ドメインで互換モードを有効または無効にするには、OpenSearch _cluster/設定 API オペレーションを使用する必要があります。

```
PUT /_cluster/settings
{
  "persistent" : {
    "compatibility.override_main_response_version" : true
  }
}
```

Amazon OpenSearch Service のトラブルシューティング

このトピックでは、一般的な Amazon OpenSearch Service の問題の特定と解決方法について説明します。AWS サポートに問い合わせる前に、このセクションの情報を参照してください。

OpenSearch Dashboards にアクセスできない

OpenSearch Dashboards エンドポイントは、署名付きリクエストをサポートしていません。ドメインのアクセスコントロールポリシーで、特定の IAM ロールにのみアクセス権が付与されており、Amazon Cognito 認証を設定していない場合は、Dashboards にアクセスしようとするときに次のエラーが発生する場合があります。

```
"User: anonymous is not authorized to perform: es:ESHttpGet"
```

OpenSearch Service ドメインで VPC アクセスを使用している場合、このエラーを受け取らない可能性がありますが、リクエストがタイムアウトする可能性があります。この問題の修正、および利用できるさまざまな設定オプションの詳細については、「the section called “Dashboards へのアクセスの制御”」、「the section called “VPC ドメインのアクセスポリシーについて”」、および「the section called “Identity and Access Management”」を参照してください。

VPC ドメインにアクセスできない

「the section called “VPC ドメインのアクセスポリシーについて”」および「the section called “VPC ドメインをテストする”」を参照してください。

VPC ドメインからのエグレストラフィック

ドメインで VPC 経由で Egress を有効にし、アウトバウンド統合が失敗し始めた場合は、以下を確認してください。

- Egress を有効にすると、Egress 統合は失敗します。送信先への VPC ルーティングと VPC リゾルバーのヘルスを確認します。「the section called “トラブルシューティング”」を参照してください。
- ホスト名の解決が失敗します。VPC DNS 設定、プライベートホストゾーン、Route 53 Resolver ルールを確認します。

- サブネットに十分な IP アドレスがありません。サブネットを展開するか、専用サブネットを使用します。「the section called “前提条件”」を参照してください。
- サービスにリンクされたロールにアクセス許可がありません。サービスにリンクされたロールを再作成するか、更新されたポリシーをアタッチします。「the section called “サービスにリンクされたロールの使用”」を参照してください。

読み取り専用状態のクラスター

以前の Elasticsearch のバージョンと比較すると、OpenSearch と Elasticsearch 7.x は、クラスターの調整に別のシステムを使用します。この新しいシステムでは、クラスターがクォーラムを失うと、アクションを実行するまでクラスターは使用できなくなります。クォーラム損失には 2 つの形式があります。

- クラスターが専用マスターノードを使用している場合は、半分以上が利用できないときにクォーラム損失が発生します。
- クラスターで専用マスターノードを使用していない場合は、データノードの半分以上が利用できないときにクォーラム損失が発生します。

クォーラム損失が発生し、クラスターに複数のノードがある場合、OpenSearch Service はクォーラムを復元し、クラスターを読み取り専用状態にします。これには 2 つのオプションがあります。

- 読み取り専用状態を削除し、クラスターをそのまま使用します。
- スナップショットからクラスターまたは個々のインデックスを復元します。

クラスターをそのまま使用する場合は、次のリクエストを使用してクラスターの状態が緑色であることを確認します。

```
GET _cat/health?v
```

クラスターの状態が赤の場合、スナップショットからクラスターを復元することをお勧めします。トラブルシューティングのステップについては、the section called “赤のクラスター状態” も合わせてお読みください。クラスターの状態が緑色の場合は、次のリクエストを使用して、予想されるすべてのインデックスが存在することを確認します。

```
GET _cat/indices?v
```

次に、いくつかの検索を実行して、予想されるデータが存在することを確認します。存在する場合は、次のリクエストを使用して読み取り専用状態を削除できます。

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.blocks.read_only": false
  }
}
```

クォーラム損失が発生し、クラスターにノードが 1 つしかない場合、OpenSearch Service はノードを置き換え、クラスターを読み取り専用状態にしません。それ以外の場合、オプションは同じです。つまり、クラスターをそのまま使用するか、スナップショットから復元します。

どちらの場合も、OpenSearch Service は、Health Dashboard に 2 つのイベントを送信します。最初のステップでは、クォーラムの損失が通知されます。2 番目のステップは、OpenSearch Service でクォーラムが正常に復元された場合に発生します。の使用の詳細については Health Dashboard、AWS Health 「 ユーザーガイド」を参照してください。

赤のクラスター状態

赤のクラスター状態は、少なくとも 1 つのプライマリシャードとそのレプリカがノードに割り当てられていないことを意味します。OpenSearch Service は、インデックスの状態にかかわらず、すべてのインデックスの自動スナップショットを取得しようとしますが、赤色のクラスター状態が維持されている間、スナップショットは失敗します。

赤のクラスター状態の最も一般的な原因は、クラスターノードの障害と、継続的な高負荷による OpenSearch プロセスのクラッシュです。

> **Note**
>
> OpenSearch Service は、クラスター状態にかかわらず 14 日間自動スナップショットを保存します。したがって、赤のクラスター状態が 2 週間を超えて続くと、最後に正常な自動スナップショットが削除され、クラスターのデータが完全に失われることになります。OpenSearch Service ドメインが赤色のクラスターステータスになった場合は、お客様自身 サポート で問題に対処するか、サポートチームにサポートを依頼するかを問い合わせることができます。CloudWatch アラームを設定して、赤のクラスター状態が発生したときに通知を受けることもできます。

最終的に、赤のシャードにより赤のクラスターが発生し、赤のインデックスにより赤のシャードが発生します。赤のクラスター状態の原因となっているインデックスを識別するため、OpenSearch にはいくつかの有用な API が用意されています。

- GET /_cluster/allocation/explain は、見つかった最初の割り当てられていないシャードを選択し、そのシャードをノードに割り当てることができない理由について説明します。

```
{
    "index": "test4",
    "shard": 0,
    "primary": true,
    "current_state": "unassigned",
    "can_allocate": "no",
    "allocate_explanation": "cannot allocate because allocation is not permitted to
 any of the nodes"
}
```

- GET /_cat/indices?v はヘルス状態、ドキュメントの数、および各インデックスのディスク使用量を表示します。

```
health status index            uuid                   pri rep docs.count docs.deleted
 store.size pri.store.size
green  open   test1            30h1EiMvS5uAFr2t5CEVoQ   5   0        820            0
       14mb           14mb
green  open   test2            sdIxs_WDT56afFGu5KPbFQ   1   0          0            0
       233b           233b
green  open   test3            GGRZp_TBRZuSaZpAGk2pmw   1   1          2            0
     14.7kb          7.3kb
red    open   test4            BJxfAErbTtu5HBjIXJV_7A   1   0
green  open   test5            _8C6MIXOSxCqVYicH3jsEA   1   0          7            0
     24.3kb         24.3kb
```

赤のインデックスを削除することが、赤のクラスター状態を修正するための最速の方法です。赤のクラスター状態の理由によっては、OpenSearch Service ドメインをスケールし、より大きなインスタンスタイプ、より多くのインスタンス、またはより EBS ベースのストレージを使用して、問題のあるインデックスの再作成を試みることができます。

問題のあるインデックスを削除することが可能でない場合、スナップショットを復元する、インデックスからドキュメントを削除する、インデックスの設定を変更する、レプリカの数を減らす、または他のインデックスを削除してディスク領域を解放することができます。重要なステップ

は、OpenSearch Service ドメインを再設定する前に、赤のクラスター状態を解決することです。赤のクラスター状態のドメインを再設定すると、問題が複雑化し、状態を解決するまで、設定状態が [処理中] のままドメインがスタックする可能性があります。

赤いクラスターの自動修復

クラスターステータスが赤になって連続して 1 時間以上経過した場合、OpenSearch Service は、未割り当てのシャードを再ルーティングするか、過去のスナップショットから復元することによってクラスターを自動的に修正しようとします。

1 つ以上の赤いインデックスの修正に失敗し、クラスターステータスが合計で 14 日間赤のままのとき、OpenSearch Service は、クラスターで以下の条件のうち少なくとも 1 つが満たされている場合にのみ追加のアクションを実行します。

- アベイラビリティーゾーンが 1 つだけである
- 専用マスターノードがある
- バースト可能なインスタンスタイプ (T2 または T3) が含まれる

この時点で、これらの条件のうちの 1 つがクラスターで満たされている場合、OpenSearch Service は、次の 7 日間にわたって、これらのインデックスを修正しないとすべての未割り当てのシャードが削除されることを説明する通知を毎日送信します。21 日が経過してもクラスターステータスが赤である場合、OpenSearch Service はすべての赤いインデックスの未割り当てシャード (ストレージとコンピューティング) を削除します。通知は、これらの各イベントの OpenSearch Service コンソールの [Notifications] (通知) パネルに表示されます。詳細については、「the section called “クラスターヘルスイベント”」を参照してください。

処理の継続的な高負荷からの復旧

赤のクラスター状態がデータノードの処理の継続的な高負荷によるものであるかどうかを判断するには、次のクラスターメトリクスをモニタリングします。

関連するメトリクス	説明	復旧
JVMMemoryPressure	クラスター内のすべてのデータノードで使用する Java ヒープのパーセンテージを指定します。このメトリクスの [最大] の統計を表示し、Java ガベージコレクターが十分なメモリ	JVM のメモリ サーキットブレーカーを設定します。詳細については、「the section called “JVM OutOfMemo

関連するメトリクス	説明	復旧
	の回収に失敗したことで生じる少量ずつのメモリプレッシャーを検出します。このパターンは通常、複雑なクエリまたは大きいデータフィールドが原因です。 x86 インスタンスタイプは、コンカレントマークスイープ (CMS) ガベージコレクターを使用します。このガベージコレクターは、アプリケーションスレッドとともに一時停止を短くします。通常の収集処理中に CMS が十分なメモリを再利用できない場合、完全なガベージコレクションがトリガーされ、アプリケーションが長時間一時停止し、クラスターの安定性に影響する可能性があります。 ARM ベースの Graviton インスタンスタイプは、Garbage-First (G1) ガベージコレクターを使用します。G1 ガベージコレクターは CMS に似ていますが、追加の短い一時停止とヒープの最適化を使用して、完全なガベージコレクションの必要性をさらに減らします。 いずれの場合でも、完全なガベージコレクション中にガベージコレクターが再利用できる範囲を超えてメモリ使用量が増加し続けると、メモリ不足エラーが発生して OpenSearch がクラッシュします。すべてのインスタンスタイプで使用率を 80% 以下にすることをお勧めします。	ryError”」を参照してください。 問題が解決しない場合は、不要なインデックスを削除する、ドメインへのリクエストの数または複雑性を減少する、インスタンスを追加する、あるいはより大きなインスタンスタイプを使用します。

関連するメトリクス	説明	復旧
	_nodes/stats/jvm API は、JVM 統計の有用な要約、メモリプールの使用量、およびガベージコレクションの情報を提供します。 `GET domain-endpoint /_nodes/stats/jvm?pretty`	
CPUUtilization	クラスター内のデータノードで使用する CPU リソースのパーセンテージを指定します。このメトリクスの [最大] の統計を表示し、使用量の多い継続的なパターンを探します。	データノードを追加するか、既存のデータノードのインスタンスタイプのサイズを大きくします。
ノード	クラスターのノード数の指定。このメトリクスの [最小] の統計を表示します。サービスがクラスターの新しいインスタンス群をデプロイすると、この値が変動します。	データノードを追加します。

黄色のクラスター状態

黄色のクラスター状態は、すべてのインデックスのプライマリシャードがクラスター内のノードに割り当てられ、少なくとも 1 つのインデックスのレプリカシャードは割り当てられていないことを意味します。単一ノードクラスターでは、OpenSearch Service がレプリカを割り当てることができる他のノードがないため、常に黄色のクラスター状態で初期化されることに注意してください。緑のクラスター状態を確保するには、ノード数を増やします。詳細については、「the section called “ドメインのサイジング”」を参照してください。

マルチノードクラスターは、新しいインデックスを作成した後、またはノード障害の後に、一時的に黄色のクラスター状態になることがあります。OpenSearch がクラスター全体でデータをレプリケートすると、この状態は自己解決されます。ディスク容量の不足も黄色のクラスター状態を引き起こす可能性があります。クラスターは、ノードにレプリカシャードを収容するディスク領域がある場合のみ、レプリカシャードを配布できます。

ClusterBlockException

以下の理由により、ClusterBlockException エラーを受け取る場合があります。

使用可能なストレージ領域の不足

クラスター内の 1 つまたは複数のノードのストレージ容量が、最小値である 1) 使用可能なストレージ容量 20%、または 2) ストレージ容量 20 GiB を下回った場合、ドキュメントの追加やインデックスの作成などの基本的な書き込みオペレーションが失敗する可能性があります。「the section called "ストレージ要件の計算"」では、OpenSearch Service によるディスク容量の使用の概要について説明します。

問題を回避するには、OpenSearch Service コンソールの FreeStorageSpace メトリクスをモニタリングし、CloudWatch アラームを作成して、FreeStorageSpace が特定のしきい値を下回ったときにトリガーします。GET /_cat/allocation?v は、シャードの割り当てとディスク使用量の有用な概要も提供します。ストレージ容量の不足に関連する問題を解決するには、OpenSearch Service ドメインをスケールし、より大きなインスタンスタイプ、より多くのインスタンス、またはより EBS ベースのストレージを使用します。

JVM メモリ負荷が高い

JVMMemoryPressure メトリクスが 30 分の間 92% を超えると、クラスターが赤の状態に到達しないように、OpenSearch Service は保護メカニズムをトリガーし、すべての書き込み操作をブロックします。保護が有効な状態では、書き込みオペレーションは ClusterBlockException エラーで失敗し、新しいインデックスは作成できず IndexCreateBlockException エラーがスローされます。

[JVMMemoryPressure] メトリクスが 88% 以下に戻った状態が 5 分間続くと、保護が無効になり、クラスターへの書き込み操作はブロックされません。

高い JVM のメモリ負荷は、クラスターに対するリクエスト数の急増、ノード全体でのシャード割り当ての不均衡、クラスター内の過度に多いシャード、大量のフィールドデータまたはインデックスマッピング、または入ってくる負荷を処理できないインスタンスタイプによって引き起こされることがあります。また、集計やワイルドカードを使用したり、クエリで広い時間範囲を使用したりすることによって発生することもあります。

クラスターへのトラフィックを減らし、JVM のメモリ負荷が高い問題を解決するには、次の 1 つ以上を試してください。

- ノードあたりの最大ヒープサイズが 32 GB になるようにドメインをスケールする。
- 古いインデックスや未使用のインデックスを削除して、シャードの数を減らす。
- `POST index-name/_cache/clear?fielddata=true` API オペレーションでデータキャッシュをクリアする。キャッシュをクリアすると、進行中のクエリが中断される可能性があることに注意してください。

一般的に、将来 JVM メモリの負荷が高くなるのを避けるために、次のベストプラクティスに従ってください。

- テキストフィールドでの集計を避けるか、インデックスのマッピングタイプを `keyword` に変更します。
- 適切な数のシャードを選択して、検索とインデックス作成のリクエストを最適化します。
- 定期的に未使用のインデックスを削除するように Index State Management (ISM) ポリシーを設定します。

Multi-AZ with Standby への移行中にエラーが発生した

既存のドメインを Multi-AZ with Standby に移行する際に、次の問題が発生する場合があります。

スタンバイのないドメインからスタンバイのあるドメインへの移行している最中に、インデックス、インデックステンプレート、ISM ポリシーのいずれかを作成する

スタンバイのないマルチ AZ からスタンバイのあるマルチ AZ に移行する際にインデックスを作成し、そのインデックスのテンプレートまたは ISM ポリシーが、推奨されているデータコピーのガイドラインに従っていない場合、データの不一致が生じて移行に失敗する可能性があります。この状況を回避するには、3 の倍数のデータコピー数 (プライマリノードとレプリカの両方を含む) で新しいインデックスを作成します。`DescribeDomainChangeProgress` API を使って移行の進行状況を確認できます。レプリカ数のエラーが発生した場合は、エラーを修正してからAWS サポート に連絡し、移行を再試行します。

データコピーの数が間違っている

ドメインに適切な数のデータコピーがないと、Multi-AZ with Standby への移行は失敗します。

JVM OutOfMemoryError

JVM の `OutOfMemoryError` は、一般的に次のいずれかの JVM サーキットブレーカーに到達したことを意味します。

サーキットブレーカー	説明	クラスター設定プロパティ
親ブレーカー	すべてのサーキットブレーカーで JVM ヒープメモリのパーセンテージの合計に許可されます。デフォルト値は 95% です。	`indices.breaker.total.limit`
フィールド データ ブレーカー	JVM ヒープメモリのパーセンテージは、メモリに単一のデータフィールドをロードすることを許可します。デフォルト値は 40% です。大きいフィールドを用いてデータをアップロードする場合は、この上限を引き上げる必要があります。	`indices.breaker.fielddata.limit`
リクエストブレーカー	JVM ヒープメモリのパーセンテージは、サービスリクエストに応答するために使用されるデータ構造に許可されます。デフォルト値は 60% です。サービスリクエストが集計の計算を含む場合は、この上限を引き上げる必要がある場合があります。	`indices.breaker.request.limit`

障害が発生したクラスターノード

Amazon EC2 インスタンスでは、予期しない終了と再起動が発生する場合があります。通常、OpenSearch Service はノードを再起動します。ただし、OpenSearch クラスターの 1 つ以上のノードが失敗した状態のままであることがあります。

この状態を確認するには、OpenSearch Service コンソールでドメインダッシュボードを開きます。[クラスターのヘルス] タブに移動し、[ノード合計数] メトリクスを見つけます。報告されるノード数がクラスターに設定した数より小さいかどうかを調べます。メトリクスが、1 つ以上のノードが 1 日以上ダウンしていることを示している場合は、AWS サポートまでお問い合わせください。

CloudWatch アラームを設定して、この問題が発生したときに通知を受けることもできます。

> **Note**
>
> [合計ノード] メトリクスは、クラスター設定の変更中およびサービスの定期的なメンテナンス中は、正確ではありません。この動作は想定されるものです。メトリクスは、すぐに正しい数のクラスターノードを報告します。詳細についてはthe section called “設定変更”を参照してください。

クラスターを予期しないノードの終了と再起動から保護するには、OpenSearch Service ドメインで各インデックスに少なくとも 1 つのレプリカを作成します。

シャードの最大制限を超えました

OpenSearch および 7.x バージョンのElasticsearch のデフォルト設定は、ノードあたり 1,000 シャード以下です。OpenSearch/ElasticSearch は、新しいインデックスの作成などのリクエストが原因でこの制限を超えると、エラーをスローします。このエラーが発生した場合は、いくつかのオプションがあります。

- さらにデータノードをクラスターに追加します。
- `_cluster/settings/cluster.max_shards_per_node` 設定を増加します。
- _shrink API を使用して、ノードのシャード数を減らします。

ドメインが処理状態でスタックしている

OpenSearch Service ドメインは、設定変更中に、「処理中」状態になります。設定変更を開始すると、ドメインの状態は 「処理中」に変わり、その間、OpenSearch Service によって新しい環境が作成されます。新しい環境では、OpenSearch Service は、適用可能なノード (データ、マスター、UltraWarm など) の新しいセットを起動します。移行が完了すると、古いノードは終了します。

次のいずれかの状況が発生した場合、クラスターは「処理中」状態でスタックすることがあります。

- 新しいデータノードセットの起動は失敗します。

- 新しいデータノードセットへのシャード移行は失敗します。
- 検証チェックがエラーで失敗しました。

これらの各状況における詳細な解決ステップについては、「Amazon OpenSearch Service ドメインが「処理中」状態のまま変わらないのはなぜですか。」を参照してください。

低 EBS バーストバランス

OpenSearch Service は、いずれかの汎用 (SSD) ボリュームの EBS バーストバランスが 70% を下回ったときにコンソール通知を送信し、バランスが 20% を下回った場合はフォローアップ通知を送信します。この問題を解決するには、クラスターをスケールアップするか、読み取り/書き込み IOPS を減らしてバーストバランスをクレジットすることができます。gp3 ボリュームタイプを使用するドメインと、ボリュームサイズが 1,000 GiB を超える gp2 ボリュームを使用するドメインのバーストバランスは 0 のままになります。詳細については、「汎用 SSD ボリューム (gp2)」を参照してください。`BurstBalance` CloudWatch メトリクスを使用して、EBS バーストバランスをモニタリングできます。

ボリュームサイズ変更中の EBS メトリクスの増加

Amazon Elastic Block Store ボリュームサイズを変更すると、`Write Throughput`、`Write Throughput Micro bursting`、`Disk Queue Depth`、IOPS などのさまざまな EBS メトリクスが一時的に増加することがあります。これは、サイズ変更操作中に予想される動作であり、通常は数秒かかります。ただし、この期間は、変更するボリュームサイズによって異なる場合があります。

レイテンシーの問題やリクエストの拒否を回避するには、クラスターが正常でネットワークトラフィックが少ない場合にのみ EBS ボリュームのサイズ変更を実行します。

監査ログを有効にできない

OpenSearch Service コンソールを使用して監査ログの公開を有効にしようとすると、次のエラーが発生する場合があります。

CloudWatch Logs グループで指定されているリソースアクセスポリシーは、Amazon OpenSearch Service がログストリームを作成するための十分な許可を付与していません。リソースアクセスポリシーを確認してください。

このエラーが発生した場合は、ポリシーの `resource` 要素に正しいロググループ ARN が含まれていることを確認してください。その場合は、次のステップを実行します。

1. 数分間待ちます。
2. ウェブブラウザでページを更新します。
3. [既存のグループを選択] を選択します。
4. [既存のロググループ] では、エラーメッセージを受け取る前に作成したロググループを選択します。
5. [アクセスポリシー]セクションで、[既存のポリシーを選択] を選択します。
6. [既存のポリシー] では、エラーメッセージを受け取る前に作成したポリシーを選択します。
7. [有効] を選択します。

プロセスを数回繰り返してもエラーが続く場合は、AWS サポートに連絡してください。

インデックスが閉じない

OpenSearch Service は、OpenSearch および Elasticsearch バージョン 7.4 以降では、`_close` API のみをサポートします。古いバージョンを使用していて、スナップショットからインデックスを復元している場合は、既存のインデックスを削除することができます (それの再インデックスの前または後)。

クライアントライセンスのチェック

Logstash と Beats のデフォルトのディストリビューションには、独自のライセンスチェックが含まれており、オープンソースバージョンの OpenSearch への接続に失敗します。OpenSearch Service では、これらのクライアントの Apache 2.0 (OSS) ディストリビューションを使用してください。

リクエストのスロットリング

永続的に `403 Request throttled due to too many requests` または `429 Too Many Requests` エラーを受け取る場合は、垂直スケーリングを検討してください。Amazon OpenSearch Service は、ペイロードによってメモリ使用量が Java ヒープの最大サイズを超える場合、リクエストをスロットリングします。

ノードに SSH 接続できない

OpenSearch クラスターのノードへのアクセスに SSH 接続が使用できず、`opensearch.yml` を直接変更できません。代わりに、 コンソール AWS CLI、、または SDKsを使用してドメインを設定します。OpenSearch REST API を使用して、いくつかのクラスターレベル設定を指定することもできます。詳細については、「Amazon OpenSearch Service API リファレンス」と the section called “サポートされているオペレーション” を参照してください。

クラスターのパフォーマンスをより詳細に把握する必要がある場合は、CloudWatch にエラーログとスローログを公開できます。

「オブジェクトのストレージクラスで有効ではない」スナップショットエラー

OpenSearch Service のスナップショットは、Amazon Glacier ストレージクラスをサポートしていません。このエラーは、オブジェクトを Amazon Glacier ストレージクラスに移行するライフサイクルルールが S3 バケットに含まれている場合に、スナップショットのリストを取得しようとすると発生する場合があります。

バケットからスナップショットを復元する必要がある場合は、Amazon Glacier からオブジェクトを復元し、オブジェクトを新しいバケットにコピーして、スナップショットリポジトリとして新しいバケットを登録します。

無効なホストヘッダー

OpenSearch Service では、クライアントがリクエストヘッダーで Host を指定する必要があります。有効な Host 値は、次のような、https:// のないドメインエンドポイントです。

```
Host: search-my-sample-domain-ih2lhn2ew2scurji.us-west-2.es.amazonaws.com
```

リクエスト作成時に Invalid Host Header エラーを受け取った場合、クライアントまたはプロキシが Host ヘッダーに OpenSearch Service ドメインエンドポイント (IP アドレスなどではなく) を含めているかどうかを確認してください。

無効な M3 インスタンスタイプ

OpenSearch Service では、OpenSearch または Elasticsearch バージョン 6.7 以降を実行している既存のドメインへの M3 インスタンスの追加または変更がサポートされていません。Elasticsearch 6.5 以前では、M3 インスタンスを引き続き使用できます。

新しいインスタンスタイプの選択をお勧めします。OpenSearch または Elasticsearch 6.7 以降を実行するドメインには、次の制限が適用されます。

- 既存のドメインで M3 インスタンスを使用していない場合、今後これらに変更することはできません。
- 既存のドメインを M3 インスタンスタイプから別のインスタンスタイプに変更した場合、元に戻すことはできません。

ホットクエリが UltraWarm を有効にした後に動作を停止する

ドメインで UltraWarm を有効にしたときに、search.max_buckets 設定への既存の上書きがない場合、OpenSearch Service は自動的に値を 10000 に設定して、メモリ負荷の高いクエリがウォームノードを飽和させないようにします。ホットクエリが 10,000 を超えるバケットを使用している場合、UltraWarm を有効にしたときにそれらが動作を停止する場合があります。

Amazon OpenSearch Service の管理された性質により、この設定を変更できないため、サポートケースを開いて制限を増やす必要があります。制限の増加には、プレミアムサポートサブスクリプションは必要ありません。

アップグレード後にダウングレードできない

インプレースアップグレードは元に戻すことができませんが、AWS サポートに連絡すれば、新しいドメインで自動的なアップグレード前のスナップショットを復元する支援が得られます。たとえば、ドメインを Elasticsearch 5.6 から 6.4 にアップグレードする場合、 AWS Support はアップグレード前のスナップショットを新しい Elasticsearch 5.6 ドメインに復元するのに役立ちます。元のドメインの手動スナップショットを作成した場合は、このステップを自分で実行することができます。

すべての のドメインの概要が必要 AWS リージョン

次のスクリプトは、Amazon EC2 describe-regions AWS CLI コマンドを使用して、OpenSearch Service を使用できるすべてのリージョンのリストを作成します。次に、リージョンごとに list-domain-names を呼び出します。

```
for region in `aws ec2 describe-regions --output text | cut -f4`
do
    echo "\nListing domains in region '$region':"
    aws opensearch list-domain-names --region $region --query 'DomainNames'
done
```

リージョンごとに次の出力が表示されます。

```
Listing domains in region:'us-west-2'...
[
  {
    "DomainName": "sample-domain"
  }
]
```

リージョンで OpenSearch Service が利用できない場合は、「エンドポイント URL に接続できませんでした」というメッセージが返されます。

OpenSearch Dashboards を使用する場合のブラウザエラー

Dashboards を使用して、OpenSearch Service ドメインのデータを表示する場合、ブラウザは HTTP レスポンスオブジェクトのサービスエラーメッセージをラップします。原因となるサービスエラーを表示し、デバッグを支援するため、Chrome の開発者モードなどのウェブブラウザで一般的に利用されている開発ツールを使用できます。

Chrome でサービスエラーを表示する

1. Chrome のトップメニューバーから、[表示]、[デベロッパー]、[デベロッパーツール] の順に選択します。
2. [ネットワーク] タブを選択します。
3. [状態] 列で、状態が 500 の任意の HTTP セッションを選択します。

Firefox でサービスエラーを表示する

1. メニューで、[ツール]、[ウェブデベロッパー]、[ネットワーク] の順に選択します。
2. 状態が 500 の任意の HTTP セッションを選択します。
3. [レスポンス] タブを選択して、サービス応答を表示します。

ノードシャードとストレージスキュー

shard skew のノードは、クラスター内の 1 つ以上のノードに、他のノードに比べて非常に多くのシャードがある場合に発生します。storage skew のノードは、クラスター内の 1 つ以上のノードに、他のノードに比べて非常に多くのストレージ (`disk.indices`) がある場合に発生します。ドメインがノードを置き換え、シャードをそのノードに引き続き割り当てているときなど、これらの条件は両方とも一時的に発生する可能性がありますが、それらが持続する場合は対処する必要があります。

両方のタイプのスキューを識別するには、_cat/allocation API オペレーションを実行し、レスポンスの `shards` と `disk.indices` のエントリを比較します。

```
shards    | disk.indices  | disk.used  | disk.avail   | disk.total   | disk.percent |
host      | ip       | node
   264    |      465.3mb  |   229.9mb  |      1.4tb   |      1.5tb   |            0 |
 x.x.x.x  | x.x.x.x  | node1
   115    |        7.9mb  |    83.7mb  |     49.1gb   |     49.2gb   |            0 |
x.x.x.x  | x.x.x.x  | node2
   264    |      465.3mb  |   235.3mb  |      1.4tb   |      1.5tb   |            0 |
 x.x.x.x  | x.x.x.x  | node3
   116    |        7.9mb  |    82.8mb  |     49.1gb   |     49.2gb   |            0 |
x.x.x.x  | x.x.x.x  | node4
   115    |        8.4mb  |      85mb  |     49.1gb   |     49.2gb   |            0 |
x.x.x.x  | x.x.x.x  | node5
```

一部のストレージスキューは正常ですが、平均から 10% を超えるものは重要です。シャードの分散にスキューがあると、CPU、ネットワーク、およびディスク帯域幅の使用量にもスキューが生じる可能性があります。通常、データが多いほどインデックス作成と検索オペレーションが増大するため、最も重いノードはリソースに最も負荷のかかるノードである傾向があり、軽いノードは十分に活用されていないキャパシティーを表します。

修正: データノード数の倍数であるシャード数を使用して、各インデックスがデータノード全体で均等に分散されるようにします。

インデックスシャードとストレージスキュー

shard skew のインデックスは、1 つ以上のノードが、他のノードよりも多くのインデックスのシャードを保持する場合に発生します。storage skew のインデックスは、1 つ以上のノードが、不均衡に大量のインデックスの合計ストレージを保持する場合に発生します。

_cat/shards API 出力を操作する必要があるため、インデックススキューはノードスキューよりも識別が困難です。クラスターまたはノードメトリクスに何らかのスキューの兆候がある場合は、インデックスのスキューを調査します。インデックススキューの一般的な兆候は次のとおりです。

- データノードのサブセットで発生する HTTP 429 エラー
- データノード全体における不均等なインデックスまたは検索オペレーションのキューイング
- データノード全体における不均等な JVM ヒープおよび/または CPU 使用率

修正: データノード数の倍数であるシャード数を使用して、各インデックスがデータノード全体で均等に分散されるようにします。それでもインデックスストレージまたはシャードスキューが見られる場合は、シャードの再割り当てを強制する必要がある可能性があります。これは、OpenSearch Service ドメインのすべてのブルー/グリーンデプロイで発生します。

VPC アクセス選択後の許可されていないオペレーション

OpenSearch Service コンソールを使用して新しいドメインを作成する場合、VPC またはパブリックアクセスを選択するオプションがあります。[VPC アクセス] を選択した場合、OpenSearch Service は VPC 情報をクエリし、適切なアクセス許可がない場合は失敗します。

```
You are not authorized to perform this operation. (Service: AmazonEC2; Status Code:
 403; Error Code: UnauthorizedOperation
```

このクエリを有効にするには、ec2:DescribeVpcs、ec2:DescribeSubnets、および ec2:DescribeSecurityGroups オペレーションへのアクセス権を持っている必要があります。この要件は、コンソールのみを対象としています。 AWS CLI を使用して VPC エンドポイントでドメインを作成および設定する場合、これらのオペレーションにアクセスする必要はありません。

VPC ドメイン作成後、読み込みでスタックする

VPC アクセスを使用した新規のドメインを作成後、ドメインの [設定の状態] が [読み込み中] から先に進行しない場合があります。この問題が発生した場合は、リージョンで AWS Security Token Service (AWS STS) が無効になっている可能性があります。

VPC に VPC エンドポイントを追加するには、OpenSearch Service が AWSServiceRoleForAmazonOpenSearchService ロールを引き受ける必要があります。したがって、特定のリージョンで VPC アクセスを使用する新しいドメインを作成するには、 を有効にする AWS STS 必要があります。の有効化と無効化の詳細については AWS STS、IAM ユーザーガイドを参照してください。

OpenSearch API への拒否されたリクエスト

OpenSearch API のタグベースのアクセス制御の導入により、以前は見られなかったアクセス拒否エラーが表示されるようになる可能性があります。これは、1 つ以上のアクセスポリシーに ResourceTag 条件を使用する Deny が含まれており、それらの条件が現在尊重されていることが原因である可能性があります。

例えば、次のポリシーは、ドメインにタグ environment=production がある場合にのみ、設定 API からの CreateDomain アクションへのアクセスを拒否するために使用されます。アクションリストには ESHttpPut も含まれていますが、拒否ステートメントはそのアクションまたはその他の ESHttp* アクションには適用されませんでした。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Action": [
      "es:CreateDomain",
      "es:ESHttpPut"
```

```
    ],
    "Effect": "Deny",
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "aws:ResourceTag/environment": [
          "production"
        ]
      }
    }
  }]
}
```

OpenSearch HTTP メソッドのタグのサポートが追加されたため、上記のような IAM ID ベースのポリシーにより、接続されたユーザーは `ESHttpPut` アクションへのアクセスを拒否されます。以前は、タグの検証がない場合でも、添付されたユーザーは PUT リクエストを送信できました。

ドメインをサービスソフトウェア R20220323 以降に更新した後にアクセス拒否エラーが表示され始めた場合は、ID ベースのアクセスポリシーをチェックして、これが当てはまるかどうかを確認し、必要に応じてアクセスを許可するように更新してください。

Alpine Linux から接続できない

Alpine Linux では、DNS レスポンスのサイズが 512 バイトに制限されています。Alpine Linux バージョン 3.18.0 以前から OpenSearch Service ドメインに接続しようとした場合、ドメインが VPC 内にあり、ノードが 20 を超えていると、DNS 解決が失敗することがあります。Alpine Linux バージョン 3.18.0 以降を使用している場合は、20 を超えるホストを解決できます。詳細については、Alpine Linux 3.18.0 のリリースノートを参照してください。

ドメインが VPC 内にある場合は、他の Linux ディストリビューション (Debian、Ubuntu、CentOS、Red Hat Enterprise Linux、Amazon Linux 2 など) を使用して接続することをお勧めします。

Search Backpressure のリクエストが多すぎる

CPU ベースのアドミッションコントロールは、トラフィックの自然な増加と急増の両方について、現在のキャパシティに基づいてノードに対するリクエストの数をプロアクティブに制限するゲートキーピングメカニズムです。多すぎるリクエストは、拒否される際に HTTP 429「リクエストが多す

ぎます」ステータスコードを返します。このエラーは、クラスターリソースの不足、リソースを大量に消費する検索リクエスト、またはワークロードの意図しない急増のいずれかを示します。

Search Backpressure は拒否の理由を提供し、リソースを大量に使用する検索リクエストを微調整するのに役立ちます。トラフィックが急増した場合は、エクスポネンシャルバックオフとジッターを使用してクライアント側で再試行することをお勧めします。

SDK を使用する場合の証明書のエラー

AWS SDKsコンピュータの CA 証明書を使用するため、 AWS サーバー上の証明書を変更すると、SDK を使用しようとすると接続が失敗する可能性があります。エラーメッセージはさまざまですが、通常は次のテキストが含まれています。

```
Failed to query OpenSearch
...
SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

このようなエラーは、コンピュータ上の CA 証明書とオペレーティングシステムを最新の状態にしておくことで回避できます。ユーザーが自分のコンピュータを管理していない企業環境でこの問題が発生した場合は、必要に応じて管理者から支援を得て更新プロセスを行う必要があります。

以下のリストは、オペレーティングシステムと Java の最小バージョンを示しています。

- 2005 年 1 月以降の更新プログラムがインストールされた Microsoft Windows バージョンでは、必要な CA が信頼リストに 1 つ以上含まれています。
- Mac OS X 10.4 with Java for Mac OS X 10.4 Release 5 (2007 年 2 月)、Mac OS X 10.5 (2007 年 10 月)、および以降のバージョンでは、必要な CA が信頼リストに 1 つ以上含まれています。
- Red Hat Enterprise Linux 5 (2007 年 3 月)、6、7、および CentOS 5、6、および 7 では、必要な CA がデフォルトの CA 信頼リストに 1 つ以上含まれています。
- Java 1.4.2_12 (2006 年 5 月)、5 Update 2 (2005 年 3 月)、および以降のすべてのバージョン (Java 6 (2006 年 12 月)、7、8 を含む) では、必要な CA がデフォルトの CA 信頼リストに 1 つ以上含まれています。

以下に示す 3 つの証明機関があります。

- Amazon Root CA 1
- Starfield Services Root Certificate Authority - G2

- Starfield Class 2 Certification Authority

最初の 2 つの機関からのルート証明書は Amazon Trust Services から入手できますが、もっと簡単なソリューションは、コンピュータを最新の状態にしておくことです。ACM から提供される証明書の詳細については、「AWS Certificate Manager に関するよくある質問」を参照してください。

> **Note**
>
> 現時点では、us-east-1 リージョン内の OpenSearch Service ドメインには別の機関からの証明書が使用されます。近い将来、これらの新しい証明機関が使用されるように、リージョンを更新する予定です。

バージョンの互換性の問題によりカスタムプラグインのインストールが失敗する

問題: プラグインと実行中の OpenSearch インスタンスのバージョンが一致しないため、プラグインのインストールに失敗しました。このシステムは、次のエラーを返します。

```
PluginValidationFailureReason : The provided plugin could not be loaded.
```

原因: このプラグインは OpenSearch ${*MAJOR*}.${*MINOR*}.{*PATCH*} 用にコンパイルされていますが、現在の環境では OpenSearch ${*MAJOR*}.${*MINOR*}.0 が実行されています。OpenSearch には、安定性とセキュリティ上の理由から、プラグインとコア OpenSearch インストール間の正確なバージョンマッチングが必要です。

解決方法: OpenSearch バージョン ${*MAJOR*}.${*MINOR*} でプラグインをビルドして、クラスターのバージョンを一致させます。

OpenSearch のバージョンを検証して更新するには

1. クラスターの API またはダッシュボードを使用して、次のコマンドを実行します。*#########*を、ユーザー自身の情報に置き換えます。

   API リクエスト:

   ```
   curl -X GET your-opensearch-endpoint/
   ```

ダッシュボードの開発ツールコンソール:

```
GET /
```

コマンドは次の形式で情報を返します。

```
{
  "name": "node-id",
  "cluster_name": "account-id:domain-name",
  "cluster_uuid": "cluster-uuid",
  "version": {
    "distribution": "opensearch",
    "number": "2.17.0",
    "build_type": "tar",
    "build_hash": "unknown",
    "build_date": "2024-12-17T11:00:09.799828091Z",
    "build_snapshot": false,
    "lucene_version": "9.11.1",
    "minimum_wire_compatibility_version": "7.10.0",
    "minimum_index_compatibility_version": "7.0.0"
  },
  "tagline": "The OpenSearch Project: https://opensearch.org/"
}
```

2. バージョン番号が ${*MAJOR*}.${*MINOR*}.0 でない場合、以下を実行してプラグインを再構築します。

   a. プラグインの `descriptor.properties` を更新してバージョン ${*MAJOR*}.${*MINOR*}.0 を指定します。
   b. プロジェクトタイプに応じたコマンドを使用してプラグインを再構築します。
   c. 新しく構築された `.zip` ファイルを使用して update-package コマンドを実行します。
   d. associate-package コマンドを実行して、前のステップで `update-package` コマンドを実行したときに作成された最新のプラグインバージョンを関連付けます。

Amazon OpenSearch Service のドキュメント履歴

このトピックでは、Amazon OpenSearch Service の重要な変更点を説明します。サービスソフトウェアの更新により、新機能、セキュリティパッチ、バグ修正、その他の機能強化のサポートが追加されます。新機能を使用するには、ドメインでのサービスソフトウェアの更新が必要になる場合があります。詳細については、「the section called “サービスソフトウェア更新”」を参照してください。

サービス機能は、サービス AWS リージョン が利用可能な に段階的にロールアウトされます。このドキュメントは、最初のリリースのためにのみ更新されています。リージョンの可用性に関する情報を提供したり、その後のリージョンのロールアウトを発表したりすることはありません。サービス機能のリージョンの可用性、および更新に関する通知のサブスクライブについては、「 の最新情報」を参照してください AWS。

更新に関する通知については、RSS フィードにサブスクライブできます。

> **Note**
>
> パッチリリース: 末尾が「-P」と数字のサービスソフトウェアバージョン (例: R20211203-P4) がパッチリリースです。パッチには、パフォーマンスの改善、マイナーなバグ修正、セキュリティ修正または体制の改善が含まれる可能性があります。パッチには新機能や重大な変更は含まれないため、一般的にユーザーやドキュメントに直接的な影響はありません。そのため、各パッチの詳細はこのドキュメントの履歴に含まれていません。

変更	説明	日付
単一 OpenSearch Dashboards 3.5 のサポート	Amazon OpenSearch Service は、単一の OpenSearch Dashboards 3.5 のサポートを追加します。単一の OpenSearch Dashboards 3.5 では、OpenSearch Dashboards 3.5 をフロントエンドとして使用し、OpenSearch バージョン 2.19、3.1、3.3、3.5 に接続できます。単一の OpenSearc	2026 年 5 月 14 日

	h Dashboards 3.5 を使用するには、サービスソフトウェア R20260428 以降を使用している必要があります。	
新機能: VPC 経由でドメイン出力トラフィックをルーティングする	`true` で を `EgressEnabled` に設定することで、VPC ドメインの出力トラフィックを独自の VPC 経由でルーティングできるようになりました`VPCOptions` 。詳細については、「VPC を介したドメイン出力トラフィックのルーティング」を参照してください。	2022 年 5 月 6 日
OpenSearch 3.5 のサポート	Amazon OpenSearch Service は OpenSearch バージョン 3.4 および 3.5 をサポートしています。詳細については、OpenSearch 3.4 リリースノート」およびOpenSearch 3.5 リリースノート」を参照してください。OpenSearch 3.4 および 3.5 オペレーションの詳細については、OpenSearch REST API リファレンスまたは特定のプラグインの API リファレンスを参照してください。重大な変更の完全なリストについては、https://docs.opensearch.org/latest/breaking-changes/OpenSearch ドキュメント」を参照してください。	2026 年 3 月 17 日

新機能: ベクトルインデックスの GPU アクセラレーション、自動最適化、ベクトル取り込み	Amazon OpenSearch Service でベクトルインデックスの GPU アクセラレーションがサポートされるようになり、ベクトル検索オペレーションが高速化されました。Amazon OpenSearch Service では、自動最適化を導入して、収集パフォーマンスとベクトル取り込みを自動的に調整し、データロードを簡素化しています。詳細については、「ベクトルインデックスの GPU アクセラレーション」、「自動最適化」、「ベクトル取り込み」を参照してください。	2025 年 12 月 2 日
OpenSearch 3.3 のサポート	Amazon OpenSearch Service は OpenSearch バージョン 3.2 および 3.3 をサポートしています。詳細については、OpenSearch 3.2 リリースノート」および OpenSearch 3.3 リリースノート」を参照してください。OpenSearch 3.2 および 3.3 オペレーションの詳細については、OpenSearch REST API リファレンスまたは特定のプラグインの API リファレンスを参照してください。	2025 年 11 月 19 日

Graviton4 ベースのインスタンスファミリー (c8g、m8g、r8g r8gd) のサポート	Amazon OpenSearch Service は、Graviton4 ベースのインスタンスファミリー (c8g、m8g、r8g、r8gd) をサポートするようになりました。詳細については、「Amazon OpenSearch Service でサポートされるインスタンスタイプ」を参照してください。	2025 年 10 月 17 日
新機能: OpenSearch Ingestion の機械学習オフラインバッチ推論	Amazon OpenSearch Ingestion (OSI) パイプラインは、機械学習 (ML) オフラインバッチ推論処理をサポートし、大量のデータを低コストで効率的に強化します。非同期で処理できる大規模なデータセットがある場合は、常にオフラインバッチ推論を使用してください。オフラインバッチ推論は、Amazon Bedrock モデルと SageMaker モデルで機能します。この機能は、OpenSearch Service 2.17 以降のドメインで OpenSearch Ingestion AWS リージョン をサポートするすべての で使用できます。詳細については、「機械学習オフラインバッチ推論での OpenSearch Ingestion パイプラインの使用」を参照してください。	2025 年 10 月 2 日

新機能: セマンティックハイライト用のリモート推論を設定するための CloudFormation テンプレート	セマンティックハイライトは、キーワードの完全一致のみに依存するのではなく、クエリの意味とコンテキストを分析することで結果の関連性を高める高度な検索機能です。この機能は、機械学習モデルを使用して検索クエリとドキュメントコンテンツ間のセマンティック類似性を評価することで、ドキュメント内で特にコンテキスト関連性が高い文章やフレーズを抽出して強調表示します。Amazon OpenSearch Service に、セマンティックハイライト用に Amazon SageMaker モデルを自動的に設定する CloudFormation テンプレートが追加されました。詳細については、「セマンティックハイライトテンプレートに対するリモート推論」を参照してください。	2025 年 9 月 26 日
AmazonOpenSearchIngestionServiceRolePolicy マネージドポリシーは更新されました	この更新により、OpenSearch Ingestion に対して OpenSearch が作成した VPC エンドポイントを変更するアクセス許可が付与され、VPC 間でパイプラインを共有することができるようになります。	2025 年 9 月 19 日

新機能: クロスアカウント取り込み用の OpenSearch Ingestion パイプラインの設定	HTTP や OTel などのプッシュベースのソースの場合、Amazon OpenSearch Ingestion を使用すると、仮想プライベートクラウド (VPC) AWS アカウント から別の VPC 内のパイプラインエンドポイントにパイプラインを共有できます。詳細については、「クロスアカウント取り込み用の OpenSearch Ingestion パイプラインの設定」を参照してください。	2025 年 9 月 19 日
Amazon OpenSearch Serverless でディスクベースのベクトル検索のサポートを開始	ディスクベースのベクトル検索は、低メモリ環境のベクトルワークロードの運用コストを大幅に削減します。詳細については、「Disk-based vector search」を参照してください。	2025 年 9 月 18 日

OpenSearch 3.1 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 3.1 がサポートされるようになりました。アップグレードの互換性の詳細については、「サポートされているアップグレードパス」を参照してください。新しいバージョンの変更の詳細については、OpenSearch 3.0 および 3.1 のドキュメントを参照してください。OpenSearch 3.1 のオペレーションについては、「OpenSearch REST API リファレンス」、または特定のプラグインの API リファレンスを参照してください。	2025 年 9 月 9 日
Amazon OpenSearch Serverless supports Federal Information Processing Standards (FIPS) 140-2	FIPS 140-2 は、機密情報を保護する暗号モジュールのセキュリティ要件を規定する米国およびカナダ政府の標準です。Amazon OpenSearch Serverless FIPS エンドポイントは、FIPS がサポートされている AWS リージョン で使用できます。これらのエンドポイントは、すべての通信に TLS 1.2 以降と FIPS 検証済み暗号化アルゴリズムを使用します。	2025 年 9 月 5 日

新機能: SAML 属性ベースのきめ細かなアクセスコントロール	Amazon OpenSearch Service は、SAML によるきめ細かなアクセスコントロールをサポートし、ID プロバイダーのユーザーとグループを OpenSearch のきめ細かなアクセスコントロールのユーザーとロールに動的にマッピングします。これらのロールを特定の OpenSearch ドメインとサーバーレスコレクションにスコープし、インデックスレベルのアクセス許可とドキュメントレベルのセキュリティを定義できます。詳細については、「SAML 属性ベースの詳細なアクセス制御を設定する」を参照してください。	2025 年 8 月 8 日

新しいトピック: OpenSearch UI エンドポイントとクォータ	Amazon OpenSearch UI は、モダナイズされた運用分析エクスペリエンスを、アジアパシフィック (ハイデラバード) AWS リージョン、アジアパシフィック (大阪) 、アジアパシフィック (ソウル) 、欧州 (ミラノ) 、欧州 (チューリッヒ) 、欧州 (スペイン) 、米国西部 (北カリフォルニア) などの新しい に拡張し、単一のエンドポイントからマネージドドメインとサーバーレスコレクションにまたがるデータ全体のインサイトを得ることができます。OpenSearch UI は、バージョン 1.3 以降の OpenSearch ドメインと OpenSearch サーバーレスコレクションに接続できます。現在、22 の AWS リージョンで利用可能です。	2025 年 8 月 7 日

新機能: 自動セマンティックエンリッチメント	自動セマンティックエンリッチメントは、Amazon OpenSearch Service でのセマンティック検索の実装と機能を簡素化します。セマンティック検索は、キーワードマッチングだけでなく、ユーザーの検索のインテントとコンテキストの意味を組み込んだクエリ結果を返します。詳細については、「自動セマンティックエンリッチメントについて」を参照してください。 自動セマンティックエンリッチメントには以下の利点があります。 実装の簡素化 機械学習 (ML) の専門知識や複雑な統合は必要ありません。 自動化プロセス セマンティックエンリッチメントは、データ取り込み中に自動的に行われます。 検索の関連性の向上 セマンティックエンリッチメントは、検索結果の品質とコンテキストの精度を向上させます。 スケーラビリティ	2025 年 8 月 6 日

セマンティックエンリッチメントは、手動操作なしでセマンティック検索を大規模なデータセットに適用します。

新機能: Amazon OpenSearch Serverless での機械学習	OpenSearch Serverless は、RAG (検索拡張生成) やセマンティック検索などの AI ワークフローの機械学習機能をサポートするようになりました。コネクタを使用して、サードパーティープラットフォームまたは AWS のサービスでホストされている ML アルゴリズムとリモートモデルにアクセスします。詳細については、「Amazon OpenSearch Serverless で機械学習を設定する」を参照してください。 OpenSearch Serverless での機械学習には、次の機能が含まれています。 ニューラル検索とハイブリッド検索 より関連性の高い結果を得るために、クエリのコンテキストと意味を理解するセマンティック検索オペレーションを実行します。キーワード検索とセマンティック検索をハイブリッド検索と組み合わせて関連性を向上させます。詳細については、「OpenSearch Serverless でニューラル検索とハイブリッド検索を設	2025 年 8 月 6 日

定する」を参照してください。

コネクタとモデル

サードパーティーの ML プラットフォームおよび AWS のサービスでホストされているモデルに接続します。モデルでコネクタを設定し、それらをデプロイして予測や AI ワークフローを実行します。

ワークフロー

複数の API コールを連鎖させることで、複雑な ML セットアップや前処理タスクを自動化できます。ワークフローは、OpenSearch での AI アプリケーションの構築を簡素化します。詳細については、「Amazon OpenSearch Serverless でのワークフロー設定」を参照してください。

新機能: OpenSearch Serverless スナップショット	スナップショットは、ディザスタリカバリ機能を提供する Amazon OpenSearch Serverless コレクションのポイントインタイムバックアップです。OpenSearch Serverless は、コレクションのスナップショットを自動的に作成および管理し、ビジネス継続性とデータ保護を確保します。 主な利点 • 手動設定を必要としない時間単位の自動バックアップ • メンテナンスオーバーヘッドがゼロに • 追加のストレージコストなし • 偶発的なデータ損失からの迅速な復旧 • スナップショットから特定のインデックスを復元可能 詳細については、「スナップショットを使用したコレクションのバックアップ」を参照してください。	2025 年 7 月 30 日
OpenSearch Ingestion が 2 つの新しいソースをサポート	OpenSearch Ingestion は、次のソースをサポートするようになりました。 • Amazon Aurora • Amazon RDS	2025 年 7 月 17 日

新機能: OpenSearch Service と Amazon S3 Vectors の統合	この統合により、OpenSearch Service でベクトル検索を大幅に強化する以下の機能が利用化可能になります。 • Amazon S3 Vectors は、ベクトルの保存とクエリをネイティブにサポートする史上初のクラウドオブジェクトストアを提供します。S3 Vectors は、意味的意味や類似性に基づいてクエリを実行できる、費用対効果に優れ、伸縮性自在で耐久性の高いベクトルストレージを提供します。1 秒未満のクエリ応答時間を提供し、ベクトルのアップロード、保存、クエリのコストを最大 90% 削減します。 • Amazon OpenSearch Service では、ベクトルインデックスのベクトルエンジンとして Amazon S3 を使用できるようになりました。この機能を使用すると、ベクトルデータを Amazon S3 にオフロードしつつ、1 秒未満のベクトル検索性能を低コストで維持できます。 詳細については、OpenSearch Service での S3 ベクトルの使用」を参照してください。	2025 年 7 月 15 日

新しいトピック: Atlassian と OpenSearch Ingestion の統合	Amazon OpenSearch Ingestion パイプラインを使用して、Atlassian Jira と Confluence サービスのデータを処理できるようになりました。この統合により、Jira プロジェクト全体と Confluence スペースを OpenSearch に同期させることで、統一された検索可能なナレッジベースが構築されます。この統合は、更新の継続的なモニタリングと自動同期を通じてリアルタイムの関連性を維持しながら、特定のプロジェクト、問題タイプ、コンテンツに対して柔軟なフィルタリングオプションを提供します。詳細については、「Atlassian サービスでの OpenSearch Ingestion パイプラインの使用」を参照してください。	2025 年 6 月 5 日

を使用したカスタムプラグイン管理 AWS CLI	AWS CLIを使用して Amazon OpenSearch Service のカスタムプラグインを管理できるようになりました。この機能を使用すると、効率的なコマンドラインワークフローを通じてカスタムプラグインをインストール、更新、保護、アンインストールできます。OpenSearch を新しいバージョンに更新する際に、プラグインを AWS KMS キーで保護し、プラグインのアップグレードを管理することもできます。詳細については、「 を使用したカスタムプラグインの管理 AWS CLI」および「Amazon OpenSearch Service カスタムパッケージ AWS KMS 統合」を参照してください。	2025 年 5 月 27 日
OpenSearch 2.19 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.19 がサポートされるようになりました。詳細については、OpenSearch 2.19 のリリースノートを参照してください。OpenSearch 2.19 のオペレーションについては、「OpenSearch REST API リファレンス」、または特定のプラグインの API リファレンスを参照してください。	2025 年 4 月 30 日

OpenSearch Ingestion でのパイプラインロール指定の新しいサポート	Amazon OpenSearch Ingestion でパイプラインを作成または更新する際、YAML 形式のパイプライン設定でパイプラインロールを指定する必要はなくなりました。代わりに、コンソールで、新しい [パイプラインロール] フィールドにロールを指定します。を使用している場合は AWS CLI、新しい`--pipeline-role-arn` パラメータを使用します。詳細については、以下の各トピックを参照してください。 • Amazon OpenSearch Ingestion でのロールとユーザーの設定 • Amazon OpenSearch Ingestion パイプラインの作成 • チュートリアル: Amazon OpenSearch Ingestion を使用してドメインにデータを取り込む • チュートリアル: Amazon OpenSearch Ingestion を使用してコレクションにデータを取り込む • CreatePipeline • UpdatePipeline	2025 年 4 月 22 日

新しいトピック	トラブルシューティングのトピック「Custom plugin installation fails due to version compatibility」では、エラー `PluginValidationFailureReason : The provided plugin could not be loaded` に遭遇した際に取るべき手順について説明しています。このエラーは通常、プラグインと実行中の OpenSearch インスタンスのバージョンが一致しないために発生します。	2025 年 4 月 21 日
新機能: OpenSearch UI とクラスター間検索の互換性	OpenSearch UI がクラスター間検索と互換性を持つようになりました。これにより、1 つの で OpenSearch UI を使用して、別のリージョンのクラスター AWS リージョン にアクセスできるようになります。これを行うには、アクセスしたいクラスターを、同じリージョン内のクラスターに接続されたリモートクラスターとして設定します。詳細については、「クロスクラスター検索によるクロスリージョンおよびクロスアカウントデータアクセス」を参照してください。	2025 年 4 月 16 日

改訂内容:「Using OpenSearch UI in Amazon OpenSearch Service」	「Using OpenSearch UI in Amazon OpenSearch Service」に関する章を見直し、拡張しました。これには、拡張された手順、追加のセットアップの詳細、OpenSearch UI に対する Amazon OpenSearch Service サポートの新しいリリース履歴が含まれます。	2025 年 4 月 15 日
Amazon OpenSearch Service 用 Amazon Q Developer	Amazon Q と Amazon OpenSearch Service の統合により、次の生成機能が利用可能になりました。 • 自然言語を使用した視覚化の生成 • View alert summaries and insights • View Amazon Q-generated query result summaries on the Discover page • View recommended anomaly detectors • Access Amazon Q chat for OpenSearch-related questions	2025 年 3 月 31 日
AmazonOpenSearchServiceRolePolicy マネージドポリシーは更新されました	このポリシーにより、OpenSearch は OpenSearch によってのみ管理される AWS IAM アイデンティティセンター アプリケーションのアクセス範囲を更新できます。	2025 年 3 月 28 日

Amazon OpenSearch Service のための OR2 および OM2 インスタンス	Amazon OpenSearch Service は、OR2 および OM2 インスタンスをサポートするようになりました。社内ベンチマークによれば、OR2 はインデックス作成スループットが OR1 に比べて最大 26%、R7 に比べて最大 70% 向上しています。OM2 は OR1 に比べて最大 15%、M7g に比べて 66% スループットが向上しています。	2025 年 3 月 25 日
CloudWatch Logs と Security Lake への直接クエリ	Amazon OpenSearch Service で、CloudWatch Logs および Security Lake 内のデータをクエリするダイレクトクエリがサポートされるようになりました。	2024 年 12 月 1 日
k-NN インデックス	OpenSearch バージョン 2.17 以降では、k-NN インデックスを UltraWarm およびコールドストレージ階層に移動できます。	2024 年 11 月 13 日
OpenSearch 2.17 のサポート	Amazon OpenSearch Service が OpenSearch バージョン 2.17 をサポートするようになりました。このバージョンには、バージョン 2.16 および 2.17 の一部であったすべての機能が含まれています。詳細については、2.16 および 2.17 リリースノートを参照してください。	2024 年 11 月 13 日

OpenSearch 2.15 のサポート	Amazon OpenSearch Service が OpenSearch バージョン 2.15 をサポートするようになりました。このバージョンには、バージョン 2.14 および 2.15 の一部であったすべての機能が含まれています。詳細については、2.14 および 2.15 リリースノートを参照してください。	2024 年 10 月 11 日
新しいサービスにリンクされたロール	Amazon OpenSearch Service は、 というサービスにリンクされたロールを追加します。これにより`AWSServiceRoleForOpensearchIngestionSelfManagedVpce` 、Amazon OpenSearch Ingestion は、セルフマネージド VPC エンドポイントを持つパイプライン Amazon CloudWatch のメトリクスデータを に送信できます。	2024 年 6 月 12 日
ダイレクトクエリ	Amazon OpenSearch Service では、OpenSearch インデックスにデータを取り込むことなく、Amazon S3 に保存されているデータに対してクエリを直接実行できるようになりました。	2024 年 5 月 22 日

OpenSearch 2.13 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.13 がサポートされるようになりました。このバージョンには、バージョン 2.12 と 2.13 に含まれていた機能がすべて含まれています。詳細については、「2.12」と「2.13」のリリースノートを参照してください。	2024 年 5 月 21 日
Amazon OpenSearch Ingestion による Data Prepper バージョン 2.7 のサポート	Amazon OpenSearch Ingestion に Data Prepper バージョン 2.7 のサポートが追加されました。詳細については、「2.7 release notes」を参照してください。	2024 年 4 月 4 日
AWS のサービス OpenSearch Serverless コレクションのプライベートアクセス	Amazon Bedrock AWS のサービスなどの特定の に、ネットワークアクセスポリシー内の OpenSearch Serverless コレクションへのアクセスを許可できるようになりました。	2024 年 3 月 28 日
インプレース EBS の更新	Amazon OpenSearch Service でブルー/グリーンデプロイなしでドメインにいくつかの EBS 変更を加えることができるようになりました。	2024 年 2 月 14 日
設定変更の可視性	Amazon OpenSearch Service コンソール内で、設定 API を使用して、ドメイン設定の変更を追跡できるようになりました。	2024 年 2 月 6 日

ベクトル検索コレクションの一般提供	Amazon OpenSearch Serverless のベクトル検索コレクションの一般提供を開始しました。プレビューフェーズ中、以下の重要な改善が行われました。 • ベクトル検索コレクションは、それぞれ最大 128 次元の数十億のベクトルを使用するワークロードをサポートするようになりました。 • OpenSearch Dashboards は、ベクトル検索コレクションをサポートするようになりました。	2023 年 11 月 29 日
OR1 インスタンス	Amazon OpenSearch Service は、OR1 インスタンスタイプをサポートするようになりました。	2023 年 11 月 29 日
Amazon S3 でのダイレクトクエリ (プレビュー)	ダイレクトクエリは、トランザクションデータが Amazon S3 バケットに書き込まれてから数秒以内に Amazon OpenSearch Service で利用できるようにするためのフルマネージドソリューションを提供します。	2023 年 11 月 29 日

時系列コレクション用の 10 TiB のキャパシティ	Amazon OpenSearch Serverless は、時系列コレクションで最大 10 TiB のインデックスデータのサポートを追加します。また、このリリースは、すべてのタイプのコレクションのために 200 OCU の最大許容キャパシティと、コレクションの作成時にスタンバイレプリカを無効にする機能をサポートしています。	2023 年 11 月 29 日
OpenSearch 2.11 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.11 がサポートされるようになりました。このバージョンには、バージョン 2.10 と 2.11 に含まれていた機能がすべて含まれています。詳細については、2.10 と 2.11 のリリースノートを参照してください。	2023 年 11 月 17 日
Amazon OpenSearch Ingestion による Data Prepper バージョン 2.6 のサポート	Amazon OpenSearch Ingestion に Data Prepper バージョン 2.6 のサポートが追加されました。詳細については、「2.6 release notes」を参照してください。さらに、パイプラインソースとして Amazon DynamoDB を指定できます。詳細については、「Using an OpenSearch Ingestion pipeline with Amazon DynamoDB」を参照してください。	2023 年 11 月 17 日

Amazon OpenSearch Ingestion による Data Prepper バージョン 2.5 のサポート	Amazon OpenSearch Ingestion に Data Prepper バージョン 2.5 のサポートが追加されました。詳細については、「2.5 リリースノート」を参照してください。さらに、OpenSearch Service ドメインまたは OpenSearch Serverless コレクションをパイプラインソースとして指定できるようになりました。詳細については、Data Prepper ドキュメントの「OpenSearch source plugin」を参照してください。	2023 年 11 月 17 日
リモート推論用の CloudFormation テンプレート	セマンティック検索のためのリモート推論の設定を容易にするために、Amazon OpenSearch Service はモデルプロビジョニングプロセスを自動化する AWS CloudFormation テンプレートをコンソールで提供します。	2023 年 11 月 7 日

サービスにリンクされたロールポリシーの更新	サービスにリンクされたロールポリシー AmazonOpenSearchServiceRolePolicy が IPv6 アドレスの割り当てと割り当て解除を行うために必要なアクセス許可が追加されました。廃止された Elasticsearch ポリシー AmazonElasticsearchServiceRolePolicy も下位互換性を確保するために更新されました。	2023 年 10 月 26 日
Amazon OpenSearch Serverless のライフサイクルポリシー	Amazon OpenSearch Serverless では、データ保持と削除の管理を効率化するインデックスライフサイクルポリシーが導入されています。API またはコンソールの設定インターフェイスを使用して時系列コレクションのデータ保持ポリシーを設定できるようになったため、日次インデックスを作成する必要や古いデータを削除するためのスクリプトを作成する必要がなくなりました。	2023 年 10 月 25 日

Im4gn インスタンスのサポート	Amazon OpenSearch Service が Im4gn インスタンスタイプをサポートするようになりました。IM4gn インスタンスは、大規模なデータセットを管理し、vCPU あたりの高いストレージ密度を必要とするワークロード向けに最適化されています。	2023 年 10 月 20 日
管理オプション	Amazon OpenSearch Service では、ドメインの問題をトラブルシューティングする必要がある場合に、きめ細かな制御を可能にする複数の管理オプションが提供されるようになりました。これらのオプションには、データノードで OpenSearch プロセスを再起動する機能や、データノードを再起動する機能が含まれます。	2023 年 10 月 17 日
オプションプラグイン	Amazon OpenSearch Service では、Nori (韓国語)、Sudachi (日本語)、Pinyin (中国語)、STConvert Analysis (中国語)、および Amazon Personalize Search Ranking プラグインの 4 つの新しい言語アナライザープラグインのサポートが追加されています。	2023 年 10 月 16 日

OpenSearch 2.9 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.9 がサポートされるようになりました。このバージョンには、バージョン 2.8 と 2.9 に含まれていた機能がすべて含まれています。詳細については、2.8 と 2.9 のリリースノートを参照してください。	2023 年 10 月 2 日
ML コネクタ	Amazon OpenSearch Service に、機械学習 (ML) コネクタのサポートが追加されました。コネクタは AWS のサービス、他の またはサードパーティーの機械学習 (ML) プラットフォームでホストされている ML モデルへのアクセスを容易にします。	2023 年 9 月 6 日
Amazon OpenSearch Ingestion に Data Prepper バージョン 2.4 のサポートが追加	Amazon OpenSearch Ingestion に Data Prepper バージョン 2.4 のサポートが追加されました。詳細については、「2.4 リリースノート」を参照してください。さらに、パイプラインソースとして Amazon Managed Streaming for Apache Kafka (Amazon MSK) を指定できるようになりました。	2023 年 8 月 31 日

時系列コレクションの 6 TiB 容量	Amazon OpenSearch Serverless は、時系列コレクションで最大 6 TiB のインデックスデータのサポートを追加します。このリリースでは、検索と時系列の両方のコレクションで最大 100 の OCU をサポートします。	2023 年 8 月 15 日
ベクトル検索コレクション	ベクトル埋め込みを保存して類似性検索およびセマンティック検索を強化するために使用できる、ベクトル検索コレクションを作成するオプションが Amazon OpenSearch Serverless に追加されました。	2023 年 7 月 26 日
OpenSearch 2.7 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.7 がサポートされるようになりました。このバージョンには、バージョン 2.6 と 2.7 に含まれていた機能がすべて含まれています。詳細については、2.6 と 2.7 のリリースノートを参照してください。	2023 年 7 月 10 日
Data Prepper 2.3 のサポート	Amazon OpenSearch Ingestion に Data Prepper バージョン 2.3 のサポートが追加されました。詳細については、「2.3 リリースノート」を参照してください。さらに、Amazon Security Lake をパイプラインソースとして指定できるようになりました。	2023 年 6 月 26 日

Multi-AZ with Standby	Amazon OpenSearch Service に、3 つのアベイラビリティーゾーン (AZ) にドメインをデプロイするオプションが追加されました。各 AZ にはデータの完全なコピーが含まれ、いずれかの AZ のノードがスタンバイとして機能します。Multi-AZ with Standby のデプロイオプションにより、インフラストラクチャに障害が発生した場合でも、99.99% の可用性と一貫性のあるパフォーマンスを実現します。	2023 年 5 月 3 日
新しいサービスにリンクされたロール	Amazon OpenSearch Service にサービスにリンクされたロール (`AWSServiceRoleForAmazonOpenSearchIngestionService`) が追加され、Amazon OpenSearch Ingestion は Amazon CloudWatchにメトリクスデータを送信できるようになっています。	2023 年 4 月 26 日

Amazon OpenSearch Ingestion	Amazon OpenSearch Ingestion は、OpenSearch Service ドメインと OpenSearch Serverless コレクションにリアルタイムのログとトレースデータを配信する、フルマネージドのデータコレクターです。OpenSearch Ingestion を使用すれば、Logstash や Jaeger などのサードパーティソリューションを使用してドメインやコレクションにデータを取り込む必要がなくなります。	2023 年 4 月 26 日
OpenSearch 2.5 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.5 がサポートされるようになりました。このバージョンには、バージョン 2.4 と 2.5 に含まれていた機能がすべて含まれています。詳細については、2.4 と 2.5 のリリースノートを参照してください。	2023 年 3 月 13 日

オフピークのメンテナンスウィンドウ	Amazon OpenSearch Service ではオフピークウィンドウが追加されています。1 日の中でドメインのトラフィックが一般的に少なくなる 10 時間の時間ブロックで、この間に、サービスソフトウェアの更新や、ブルー/グリーンデプロイを必要とする Auto-Tune の最適化をスケジュールできます。オフピークの時間帯に更新を行うことにより、トラフィックの多い時間帯にクラスターの専用マスターノードにかかる負荷を、最小限に抑えることができます。 2 月 16 日以降に作成された新しいドメインの場合、オフピークウィンドウは現地時間の午後 10 時から午前 8 時の間に自動的に設定されます。既存のドメインでは、このウィンドウを明示的に有効にする必要があります。	2023 年 2 月 16 日
ドメイン作成時に SAML 認証を設定する	Amazon OpenSearch Service が、ドメイン作成中の SAML 認証の設定をサポートするようになりました。これまでは、ドメインを既に作成した後で SAML オプションを設定する必要がありました。	2023 年 2 月 1 日

VPC ドメインのリモート再インデックス	Amazon OpenSearch Service に、2 つのドメイン間での VPC エンドポイント接続のオプションが追加されます。リモート再インデックスを使用して、リバースプロキシを使用することなく、1 つの VPC ドメインから別の VPC ドメインにインデックスをコピーできるようになりました。この機能を使用するには、VPC ドメインがサービスソフトウェア R20221114 以降を実行している必要があります。	2023 年 1 月 31 日

Amazon OpenSearch Serverless の一般公開	Amazon OpenSearch Serverless が一般公開されました。プレビューフェーズ中、以下の重要な改善が行われました。 • コレクションエンドポイントのトラフィックが減少した場合に、設定した最小 OCU まで容量をスケールダウンできるようになりました。 • インデックス作成と検索両方の最大許容 OCU が 20 から 50 に引き上げられました。各 OCU には、120 GiB のインデックスデータを保存するのに十分なホットエフェメラルストレージが含まれています。 • データアクセス設定を、個別のワークフローで実行するのではなく、コレクションの作成中に実行できるようになりました。	2023 年 1 月 25 日
非同期ドライラン	Amazon OpenSearch Service が非同期ドライランをサポートするようになりました。これは、設定を変更する前に検証チェックを実行することを可能にし、変更によってブルー/グリーンデプロイが実行されることになる場合は通知を行います。	2023 年 1 月 19 日

新しいサービスにリンクされたロール	Amazon OpenSearch Service には、`AWSServiceRoleForAmazonOpenSearchServerless` というサービスにリンクされたロールが追加されています。これにより、OpenSearch Serverless は Amazon CloudWatchにメトリクスデータを送信できます。	2022 年 11 月 29 日
Amazon OpenSearch Serverless のプレビュー	Amazon OpenSearch Serverless は、Amazon OpenSearch Service 用のオンデマンド、オートスケーリング、サーバーレス設定です。サーバーレスにより、OpenSearch クラスターのプロビジョニング、設定、チューニングの運用上の複雑さがなくなります。	2022 年 11 月 29 日

OpenSearch 2.3 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 2.3 がサポートされるようになりました。このバージョンには、バージョン 2.0、2.1、2.2 に含まれているすべての機能が含まれています。詳細については、2.0、2.1、2.2、2.3 のリリースノートを参照してください。バージョン 2.3 には重大な変更が含まれています。詳細については、「サポートされているアップグレードパス」を参照してください。	2022 年 11 月 15 日
通知プラグインのサポート	Amazon OpenSearch Service は、OpenSearch プラグインからのすべての通知を一元的に管理できる通知プラグインをサポートするようになりました。バージョン 2.0 以降、アラート送信先は廃止され、通知チャネルに置き換えられました。	2022 年 11 月 15 日

Kibana 7.1.1 のサポート	Elasticsearch 7.1 を実行している Amazon OpenSearch Service ドメインが Kibana 7.1.1 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。7.1 ドメインをサービスソフトウェア R20221114 に更新すると、これらのドメインは、OpenSearch Service によってこのパッチリリースに自動的にアップグレードされます。	2022 年 11 月 15 日
Kibana 6.8.13 のサポート	Elasticsearch 6.8 を実行している Amazon OpenSearch Service ドメインが Kibana 6.8.13 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。6.8 ドメインをサービスソフトウェア R20221114 に更新すると、これらのドメインは、OpenSearch Service によってこのパッチリリースに自動的にアップグレードされます。	2022 年 11 月 15 日

Kibana 6.3.2 のサポート	Elasticsearch 6.3 を実行している Amazon OpenSearch Service ドメインが Kibana 6.3.2 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。6.3 ドメインをサービスソフトウェア R20221114 に更新すると、これらのドメインは、OpenSearch Service によってこのパッチリリースに自動的にアップグレードされます。	2022 年 11 月 15 日

AWS PrivateLink	Amazon OpenSearch Service で管理される VPC エンドポイントを使用すると、インターネット経由で接続するのではなく、インターフェイス VPC エンドポイントを使用して OpenSearch Service VPC ドメインに直接接続できます。OpenSearch Service で管理される VPC エンドポイントは、エンドポイントがプロビジョニングされている VPC 内か、エンドポイントがプロビジョニングされている VPC とピア接続されている (ルートテーブルとセキュリティグループによって許可されている) VPC からのみアクセスできます。VPC ドメインがインターフェイス VPC エンドポイントに接続するには、サービスソフトウェア R20220928 以降を実行している必要があります。	2022 年 11 月 7 日
バグ修正とパフォーマンス向上	サービスソフトウェア R20220928には、バグの修正とSAML口ギングの改善を含むパフォーマンスの向上が含まれています。また、この更新により、デフォルトのテナントが `Private` から `Global` に変更されます。	2022 年 10 月 3 日

API リファレンスの改善	Amazon OpenSearch Service は、改善された包括的な設定 API リファレンスを提供します。新しいリファレンスには、使用可能なすべてのアクションとデータタイプ、サンプルリクエストとレスポンスの構文、サポートされているすべての言語の対応する SDK リファレンスへのリンクが含まれています。	2022 年 9 月 13 日
ブルー/グリーン検証	Amazon OpenSearch Service は、ブルー/グリーンデプロイの前に検証チェックを実行し、ドメインが更新の要件を満たしていない場合は検証エラーを表示するようになりました。	2022 年 8 月 16 日
OpenSearch 1.3 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 1.3 がサポートされるようになりました。詳細については、「1.3 リリースノート」を参照してください。	2022 年 7 月 27 日
ML Commons プラグインのサポート	Amazon OpenSearch Service は、トランスポートおよび REST API コールを通じて一連の一般的な機械学習アルゴリズムを提供する ML Commons プラグインのサポートを追加します。また、PPL コマンドを通じて ML Commons プラグインを操作することもできます。	2022 年 7 月 27 日

gp3 ボリュームのサポート	Amazon OpenSearch Service は、gp3 EBS 汎用 SSD ボリュームタイプのサポートを追加します。ドメインを作成または変更するときに、プロビジョンド IOPS とスループットをさらに指定できます。	2022 年 7 月 26 日
強化されたベストプラクティスに関するドキュメント	Amazon OpenSearch Service ドキュメントは、OpenSearch Service ドメインを作成および運用するための改善された運用上のベストプラクティスと一般的な推奨事項を提供します。	2022 年 7 月 6 日
Service Quotas との統合	Service Quotas コンソールから、Amazon OpenSearch Service の割り当てを表示し、割り当ての増加をリクエストできるようになりました。	2022 年 6 月 29 日
OpenSearch API のタグベースのアクセス制御	タグを使用して、OpenSearch API へのアクセスを制御できるようになりました。以前は、タグを使用して設定 API へのアクセスを制御することしかできませんでした。	2022 年 6 月 16 日
リージョン間のクロスクラスター検索	クラスター間検索は AWS リージョン 、両方のドメインが Elasticsearch バージョン 7.10 以降、または任意のバージョンの OpenSearch を実行している限り、 でサポートされるようになりました。	2022 年 6 月 14 日

単一の Kibana 5.6 のサポート	Amazon OpenSearch Service は、単一の Kibana 5.6.16 のサポートを追加します。単一の Kibana 5.6.16 では、Elasticsearch 5.1、5.3、5.5、および 5.6 に接続しながら、フロントエンドとして Kibana 5.6 を使用できます。単一の Kibana 5.6 を使用するには、サービスソフトウェア R20220323 以降が必要です。	2022 年 4 月 4 日
R20220323-P1	Amazon OpenSearch Service は最近、サービスソフトウェアの更新プログラム R20220323 をリリースしましたが、その後、問題のために更新がロールバックされました。ドメインをパッチリリース R20220323-P1 以降に更新することをお勧めします。これにより、問題が解決されます。	2022 年 4 月 4 日
OpenSearch 1.2 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 1.2 がサポートされるようになりました。詳細については、「1.2 リリースノート」を参照してください。	2022 年 4 月 4 日

可観測性	Amazon OpenSearch Service の OpenSearch Dashboards のデフォルトインストールには、データを参照およびクエリするためにパイプ処理言語 (PPL) を使用してデータ駆動型イベントを可視化できる可観測性プラグインが含まれています。プラグインには、OpenSearch 1.2 以降とサービスソフトウェア R20220323 以降が必要です。	2022 年 4 月 4 日
Kibana 7.7.1 のサポート	Elasticsearch 7.7 を実行している Amazon OpenSearch Service ドメインが Kibana 7.7 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。7.7 ドメインをサービスソフトウェア R20220323 以降に更新すると、これらのドメイは、OpenSearch Service によってこのパッチリリースに自動的にアップグレードされます。	2022 年 4 月 4 日

JVM メモリプレッシャーメトリクスの変更	Amazon OpenSearch Service では、メモリ使用率の反映精度を向上させることを目的に JVMMemoryPressure CloudWatch メトリクスのロジックが変更されました。以前は、メトリクスでは JVM ヒープの旧世代のメモリプールのみが考慮されていました。この変更により、メトリクスでは新しい世代のメモリープールも考慮されます。ドメインをサービスソフトウェア R20220323 に更新すると、JVMMemoryPressure 、MasterJVMMemoryPressure 、および WarmJVMMemoryPressure メトリクスで増加がみられることがあります。	2022 年 4 月 4 日
IK (中国語) 分析プラグインを使用したカスタム辞書	Amazon OpenSearch Service で IK (中国語) 分析プラグインでのカスタム辞書の使用がサポートされるようになりました。	2022 年 4 月 4 日

既存のドメインでのクラスター間レプリケーション	Amazon OpenSearch Service は、2020 年 6 月 3 日以降に作成されたドメインにのみクラスター間検索およびクラスター間レプリケーションを実装できるという制限を削除しました。いつ作成されたかに関係なく、これらの機能をすべてのドメインで有効化できるようになりました。両方のドメインがサービスソフトウェア R20220323 以降である必要があります。	2022 年 4 月 4 日
Blue/Green デプロイの可視性	Amazon OpenSearch Service は、Blue/Green デプロイの進行状況をより可視化するようになりました。これらの詳細は、コンソールまたは Configuration API を使用してモニタリングできます。	2022 年 1 月 27 日
既存のドメインでのきめ細かなアクセスコントロール	きめ細かなアクセスコントロールを既存のドメインに対して有効にできるようになりました。Open/IP ベースのアクセスポリシーの一時的な移行期間を有効にして、ロールの作成とマッピング中にユーザーが引き続きドメインにアクセスできるようにすることができます。きめ細かなアクセスコントロールを既存のドメインで有効にするには、サービスソフトウェア R20211203 以降が必要です。	2022 年 1 月 6 日

OpenSearch Dashboards のロール名変更	サービスソフトウェア R20211203 で、kibana_user ロールが opensearch_dashboards_user に、kibana_read_only が opensearch_dashboards_read_only に名称変更されました。この変更は、新しく作成されるすべての OpenSearch 1.x ドメインに適用されます。サービスソフトウェア R20211203 にアップグレードする既存の OpenSearch ドメインの場合、ロールは同じままです。	2022 年 1 月 4 日
OpenSearch 1.1 のサポート	Amazon OpenSearch Service で OpenSearch バージョン 1.1 がサポートされるようになりました。詳細については、「1.1 リリースノート」を参照してください。	2022 年 1 月 4 日
ISM ビジュアルエディタ	Amazon OpenSearch Service 用 OpenSearch Dashboards のデフォルトインストールで、ISM ポリシーのビジュアルエディタがサポートされるようになりました。この機能には OpenSearch 1.1 以降が必要です。	2022 年 1 月 4 日

サービス間の混乱した代理防止の更新	Amazon OpenSearch Service はIAM リソースポリシーで `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用して、混乱した代理の問題を防止します。これらの条件キーを使用するには、サービスソフトウェア R20211203 以降にある必要があります。	2022 年 1 月 4 日

Log4j パッチ	サービスソフトウェア R20211203-P2 は、CVE-2021-44228 および CVE-2021-45046 のアドバイザリーで推奨されているように、OpenSearch Service で使用される Log4j のバージョンを更新します。このパッチは、OpenSearch と Elasticsearch のすべてのバージョンを実行しているドメインに適用されます。OpenSearch Service は、引き続きさまざまな Log4j バージョンを内部的に更新しますが、これらは必ずしも最新バージョンの Log4j に限定されません。ドメインの Log4j のバージョンは、ドメインが実行しているソフトウェアのバージョンによって異なります。ただし、Log4j のバージョンにかかわらず、R20211203-P2 以降を実行している限り、ドメインには、CVE-2021-44228 および CVE-2021-45046 に対処するために必要な Log4j の更新が含まれています。	2021 年 12 月 15日

クラスター間レプリケーション	クラスター間レプリケーションを使用すると、1 つの OpenSearch Service ドメインから別のドメインにインデックス、マッピング、メタデータをレプリケートできます。クラスター間レプリケーションには、Elasticsearch 7.10 または OpenSearch 1.1 以降を実行するドメインが必要です。	2021 年 10 月 5 日
新しい AWS管理ポリシー	Amazon OpenSearch Service の起動には、新しい AWS管理ポリシーと古いポリシーの廃止が含まれます。	2021 年 9 月 8 日
Kibana 6.4.3 のサポート	従来の Elasticsearch バージョン 6.4 を実行している Amazon OpenSearch Service ドメインは、Kibana 6.4 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。OpenSearch Service は、ドメインをこのパッチリリースに自動的にアップグレードします。	2021 年 9 月 8 日

データストリーム	Amazon OpenSearch Service では、Data Streams のサポートを追加しています。これにより、時系列データの管理のプロセスを簡素化します。データストリームを使用するには、ドメインが OpenSearch 1.0 以降を実行している必要があります。	2021 年 9 月 8 日
Amazon OpenSearch Service	AWS は Amazon OpenSearch Service の名前を変更して、従来の「Elasticsearch」ブランドを削除します。Amazon OpenSearch Service は、OpenSearch と従来の Elasticsearch OSS をサポートします。クラスターを作成するときに、どの検索エンジンを使用するかのオプションがあります。OpenSearch Service は、ソフトウェアの最終的なオープンソースバージョンである Elasticsearch OSS 7.10 との幅広い互換性を提供します。	2021 年 9 月 8 日

コールドストレージ	コールドストレージは、アクセス頻度の低いデータまたは履歴データ用の新しいストレージ階層です。コールドインデックスは、S3 ストレージのみを占有し、コンピューティングはアタッチされません。コールドストレージには、Elasticsearch 7.9 以降を実行しているドメインとサービスソフトウェア R20210426 以降が必要です。	2021 年 5 月 13 日
ARM ベースの Graviton インスタンス	Amazon OpenSearch Service は、ARM ベースの Graviton インスタンスタイプ (M6G、C6G、R6G、R6GD) をサポートするようになりました。Graviton インスタンスタイプは、Elasticsearch 7.9 以降を実行している新規および既存のドメインおよびサービスソフトウェア R20210331 以降で使用できます。	2021 年 5 月 4 日

ISM テンプレート	Amazon OpenSearch Service では、ISM テンプレートのサポートを追加しています。これにより、インデックスがポリシーで定義されているパターンと一致する場合に、ISM ポリシーをインデックスに自動的にアタッチできます。ISM テンプレートには、サービスソフトウェア R20210426 以降が必要です。この更新プログラムでは、`policy_id` 設定も非推奨にしました。つまり、インデックステンプレートを使用して、新しく作成されたインデックスに ISM ポリシーを適用できなくなります。この更新では、この設定を使用して既存の CloudFormation テンプレートに重大な変更が加えられました。	2021 年 4 月 27 日
Elasticsearch 7.10 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.10 をサポートするようになりました。詳細については、7.10 リリースノートを参照してください。	2021 年 4 月 21 日

非同期検索	Amazon OpenSearch Service は、非同期検索をサポートするようになりました。これにより、検索リクエストをバックグラウンドで実行できます。非同期検索には、Elasticsearch 7.10 以降を実行しているドメインとサービスソフトウェア R20210331 以降が必要です。	2021 年 4 月 21 日
設定 API 向けのタグベースのアクセスコントロール	AWS タグを使用して Amazon ES 設定 API へのアクセスを制御できるようになりました。	2021 年 3 月 2 日
Auto-Tune	Amazon OpenSearch Service には、Auto-Tune が追加されました。本機能は、クラスターのパフォーマンスと使用状況メトリクスを使用して、ノードの JVM 設定への変更を提案します。Auto-Tune には、Elasticsearch 6.7 以降を実行しているドメインとサービスソフトウェア R20201117 以降が必要です。	2021 年 2 月 24 日

トレース分析	Amazon OpenSearch Service 用 Kibana のデフォルトのインストールには、トレース分析プラグインが追加されました。これにより、分散アプリケーションからのトレースデータをモニタリングが可能になりました。プラグインには、Elasticsearch 7.9 以降を実行しているドメインとサービスソフトウェア R20210201 以降が必要です。	2021 年 2 月 17 日
シャードのメトリクス	Amazon OpenSearch Service には、シャードステータスを追跡するための次の CloudWatch メトリクスを追加されました: `Shards.active` 、`Shards.unassigned` 、`Shards.delayedUnassigned` `Shards.activePrimary` 、`Shards.initializing` 、`Shards.relocating` 。このメトリクスは、サービスソフトウェア R20210201 以降を実行しているドメインで使用できます。	2021 年 2 月 17 日

Kibana レポート	Amazon OpenSearch Service 用 Kibana のデフォルトインストールでは、[Discover] (検出)、[Visualize] (視覚化)、[Dashboard] (ダッシュボード) の各ページのオンデマンドレポートがサポートされるようになりました。この機能には、Elasticsearch 7.9 以降およびサービスソフトウェア R20210201 以降が必要です。	2021 年 2 月 17 日
Kibana 5.6.16 のサポート	Elasticsearch 5.6 を実行している Amazon OpenSearch Service ドメインが、 Kibana 5.6 の最新のパッチリリースをサポートするようになりました。これにより、バグ修正が追加され、セキュリティが向上します。Amazon ES は、ドメインをこのパッチリリースに自動的にアップグレードします。	2021 年 2 月 17 日
既存のドメインの暗号化	Amazon OpenSearch Service は、Elasticsearch 6.7 以降を実行している既存のドメインでの保管中のデータの暗号化とノード間の暗号化の有効化をサポートするようになりました。これらの設定を有効にした後、無効にすることはできません。	2021 年 1 月 27 日

リモート再インデックス	Amazon OpenSearch Service は、リモート再インデックスをサポートするようになりました。これにより、リモートドメインからインデックスを移行できます。この機能には、サービスソフトウェア R20201117 以降が必要です。	2020 年 11 月 24 日
パイプ処理言語	Amazon OpenSearch Service は、パイプ処理言語 (PPL) をサポートするようになりました。これは、パイプ (|) 構文を使用して Elasticsearch に保存されているデータのクエリを行うクエリ言語です。この機能には、サービスソフトウェア R20201117 以降が必要です。詳細についてはを参照してください。	2020 年 11 月 24 日
Kibana ノートブック	Amazon OpenSearch Service では、Kibana ノートブックのサポートを追加しています。これにより、ライブビジュアライゼーションとナレーションテキストを 1 つのインターフェイスで組み合わせることができます。この機能には、サービスソフトウェア R20201117 以降が必要です。	2020 年 11 月 24 日

ガントチャート	Amazon OpenSearch Service 用 Kibana のデフォルトインストールでは、新しいビジュアライゼーションタイプであるガントチャートがサポートされるようになりました。この機能には、サービスソフトウェア R20201117 以降が必要です。	2020 年 11 月 24 日
Elasticsearch 7.9 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.9 をサポートするようになりました。詳細については、7.9 リリースノートを参照してください。	2020 年 11 月 24 日
異常検出の更新	Amazon OpenSearch Service の異常検出では、高カーディナリティのサポートを追加しています。これにより、IP アドレス、製品 ID、国コードなどのディメンションを用いて異常を分類できます。この機能には、サービスソフトウェア R20201117 以降が必要です。	2020 年 11 月 24 日

動的辞書の更新	Amazon OpenSearch Service では、インデックスを再作成せずに検索アナライザーを更新できるようになりました。ドメインの一部またはすべてのドメインで辞書ファイルを更新できます。また、Amazon ES がパッケージのバージョンを長期的に追跡するので、変更内容と変更時期の履歴が得られるようになります。この機能には、サービスソフトウェア R20201019 以降が必要です。	2020 年 11 月 17 日
カスタムエンドポイント	Amazon OpenSearch Service は、カスタムエンドポイントをサポートするようになりました。これにより、Amazon ES ドメインに新しい URL を指定できます。ドメインを入れ替える場合は、同じ URL を維持できます。この機能には、サービスソフトウェア R20201019 以降が必要です。	2020 年 11 月 5 日
新しい言語プラグイン	Amazon OpenSearch Service は、サービスソフトウェア R20201019 以降を用いて Elasticsearch 7.7 以降を実行しているドメインで IK (中国語) 分析、ベトナム語分析、タイ語分析プラグインをサポートするようになりました。	2020 年 10 月 28 日

Elasticsearch 7.8 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.8 をサポートするようになりました。詳細については、7.8 リリースノートを参照してください。	2020 年 10 月 28 日
Kibana の SAML 認証	Amazon OpenSearch Service は、Kibana の SAML 認証をサポートするようになりました。これにより、サードパーティーの ID プロバイダーを使用して Kibana へのログイン、詳細なアクセスコントロールの管理、データの検索、およびビジュアライゼーションの構築を行うことができます。この機能には、サービスソフトウェア R20201019 以降が必要です。	2020 年 10 月 27 日
T3 インスタンス	Amazon OpenSearch Service は、`t3.small` と `t3.medium` のインスタンスタイプがサポートするようになりました。	2020 年 9 月 23 日
監査ログ	Amazon OpenSearch Service は、データの監査ログをサポートするようになりました。これにより、ログイン試行の失敗や、インデックス、ドキュメント、およびフィールドへのユーザーアクセスなどを追跡できます。この機能には、サービスソフトウェア R20200910 以降が必要です。	2020 年 9 月 16 日

UltraWarm 更新	Amazon OpenSearch Service の UltraWarm には、新しいメトリクス、新しい設定、より大きい移行キュー、およびキャンセル API が追加されました。これらの更新には、サービスソフトウェア R20200910 以降が必要です。詳細については、「」を参照してください。	2020 年 9 月 14 日
Learning to Rank	Amazon OpenSearch Service は、オープンソースの Learning to Rank プラグインをサポートするようになりました。これにより、機械学習テクノロジーを使用して検索の関連性を改善できます。この機能には、サービスソフトウェア R20200721 以降が必要です。	2020 年 7 月 27 日
k-NN コサイン類似度	K 最近傍 (k-NN) では、ユークリッド距離に加えて、コサイン類似度によって「最近隣接」を検索できるようになりました。この機能には、サービスソフトウェア R20200721 以降が必要です。	2020 年 7 月 23 日

gzip 圧縮	Amazon OpenSearch Service は、ほとんどの HTTP レクエストとレスポンスの gzip 圧縮をサポートするようになりました。これにより、レイテンシーを低減し、帯域幅を節約できます。この機能には、サービスソフトウェア R20200721 以降が必要です。	2020 年 7 月 23 日
Elasticsearch 7.7 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.7 をサポートするようになりました。詳細については、7.7 リリースノートを参照してください。	2020 年 7 月 23 日
Kibana マップサービス	Amazon OpenSearch Service での Kibana のデフォルトインストールには、インドおよび中国リージョンのドメインを除き、WMS マップサーバーが含まれるようになりました。	2020 年 6 月 18 日
SQL の改良点	Amazon OpenSearch Service の SQL サポートは、多数の新しいオペレーション、データ探索専用の Kibana ユーザーインターフェイス、および対話型 CLI をサポートするようになりました。	2020 年 6 月 3 日
クラスター間検索	Amazon OpenSearch Service では、接続された複数のドメインでクラスター間クエリと集計を実行できます。	2020 年 6 月 3 日

異常検出	Amazon OpenSearch Service では、ほぼリアルタイムで自動的に異常を検出できます。	2020 年 6 月 3 日
UltraWarm	Amazon OpenSearch Service の UltraWarm ストレージは、パブリックプレビューを完了し、一般利用可能になりました。この機能は、より広範なバージョン および をサポートするようになりました AWS リージョン。詳細については、「」を参照してください。	2020 年 5 月 5 日
カスタム辞書	Amazon OpenSearch Service では、クラスターで使用するカスタム辞書ファイルをアップロードできます。これらのファイルは、特定の高頻度の単語を無視するか、同等の単語として扱うように Elasticsearch に指示することで、検索結果を改善します。	2020 年 4 月 21 日
Elasticsearch 7.4 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.4 をサポートするようになりました。詳細については、「サポートされるバージョン」を参照してください。	2020 年 3 月 12 日

k-NN	Amazon OpenSearch Service では、k 最近傍 (k-NN) 検索のサポートが追加されました。k-NN には、サービスソフトウェア R20200302 以降が必要です。	2020 年 3 月 3 日
インデックスステート管理	Amazon OpenSearch Service では、インデックスステート管理 (ISM) を追加しています。これにより、インデックスが一定の期間に達したときの削除などの日常的なタスクを自動化できます。この機能には、サービスソフトウェア R20200302 以降が必要です。	2020 年 3 月 3 日
Elasticsearch 5.6.16 のサポート	Amazon OpenSearch Service は、バージョン 5.6 の最新のパッチリリースをサポートするようになりました。このバージョンでは、追加でバグが修正され、セキュリティが向上しました。Amazon ES は、既存の 5.6 ドメインをこのリリースに自動的にアップグレードします。この Elasticsearch リリースでは、バージョンが 5.6.17 と誤って報告されることに注意してください。	2020 年 3 月 2 日

きめ細かなアクセスコントロール	Amazon OpenSearch Service は、きめ細かなアクセスコントロールをサポートするようになりました。クラスターに対して、インデックスレベル、ドキュメントレベル、フィールドレベルのセキュリティ、Kibana マルチテナンシー、オプションの HTTP 基本認証が使用できるようになりました。	2020 年 2 月 11 日
UltraWarm ストレージ (プレビュー)	Amazon OpenSearch Service では、UltraWarm が追加されました。これは、Amazon S3 を使用する新しいウォームストレージ階層であり、パフォーマンスを向上させるための高度なキャッシュソリューションです。アクティブに書き込みを行っていないインデックスやクエリの頻度が低いインデックスについては、UltraWarm ストレージにより、GiB あたりのコストが大幅に削減されます。	2019 年 12 月 3 日
中国リージョンの暗号化機能	保管中のデータの暗号化およびノード間暗号化が、cn-north-1　中国 (北京) リージョンおよび cn-northwest-1　中国 (寧夏) リージョンで利用可能になりました。	2019 年 11 月 20 日

HTTPS が必要	Amazon ES ドメインへのすべてのトラフィックが HTTPS 経由で到着することを要求できるようになりました。ドメインを設定するときは、[HTTPS が必要] チェックボックスをオンにします。この機能には、サービスソフトウェア R20190808 以降が必要です。	2019 年 10 月 3 日
Elasticsearch 7.1 および 6.8 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 7.1 および 6.8 をサポートするようになりました。詳細については、「サポートされるバージョン」を参照してください。	2019 年 8 月 13 日
毎時スナップショット	Amazon OpenSearch Service は、Elasticsearch 5.3 以降を実行しているドメインのスナップショットを、毎日ではなく、毎時に取得するようになりました。これにより、データを復元するためのバックアップが、より頻繁に実行されます。	2019 年 7 月 8 日
Elasticsearch 6.7 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 6.7 をサポートするようになりました。詳細については、「サポートされるバージョン」を参照してください。	2019 年 5 月 29 日

SQL のサポート	Amazon OpenSearch Service では、SQL を使用してデータのクエリを行うことができるようになりました。SQL のサポートには、サービスソフトウェア R20190418 以降が必要です。	2019 年 5 月 15 日
5 つのシリーズのインスタンスタイプ	Amazon OpenSearch Service は、M5、C5、R5 のインスタンスタイプをサポートするようになりました。旧世代のインスタンスタイプと比較して、これらの新しいタイプは低価格で優れたパフォーマンスを提供します。詳細については、「制限」を参照してください。	2019 年 4 月 24 日
Elasticsearch 6.5 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 6.5 をサポートするようになりました。	2019 年 4 月 8 日
アラート	1 つまたは複数の Amazon ES インデックスのデータが特定の条件を満たすと、Amazon OpenSearch Service のアラートにより通知されます。アラートには、サービスソフトウェア R20190221 以降が必要です。	2019 年 3 月 25 日

3 つのアベイラビリティーゾーンのサポート	Amazon OpenSearch Service は、多くのリージョンで 3 つのアベイラビリティーゾーンをサポートするようになりました。このリリースには、効率化されたコンソール機能も含まれています。このマルチ AZ には、サービスソフトウェア R20181023 以降が必要です。	2019 年 2 月 7 日
Elasticsearch 6.4 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 6.4 をサポートするようになりました。	2019 年 1 月 23 日
200 ノードクラスター	Amazon ES では、ストレージが合計 3 PB の最大 200 個のデータノードを持つクラスターを作成できるようになりました。	2019 年 1 月 22 日
サービスソフトウェア更新	Amazon ES では、ドメインのサービスソフトウェアを手動で更新して、新機能を迅速に利用したり、トラフィックの少ない時間に更新したりできるようになりました。詳細についてはを参照してください。	2018 年 11 月 20 日
新しい CloudWatch メトリクス	Amazon ES には、ノードレベルのメトリクスが追加され、新しい [クラスターヘルス] および [インスタンスヘルス] タブが Amazon ES コンソールに追加されました。	2018 年 11 月 20 日

中国 (北京) のサポート	Amazon OpenSearch Service が cn-north-1 リージョンで使用できるようになりました。ここでは、M4、C4、R4 インスタンスタイプがサポートされます。	2018 年 10 月 17 日
ノード間の暗号化	Amazon OpenSearch Service は、ノード間の暗号化をサポートするようになりました。これにより、データは Amazon ES によってクラスター全体に配信される際に、暗号化されたままになります。	2018 年 9 月 18 日
インプレースバージョンアップグレード	Amazon OpenSearch Service は、インプレースバージョンアップグレードをサポートするようになりました。	2018 年 8 月 14 日
Elasticsearch 6.3 および 5.6 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 6.3 および 5.6 をサポートするようになりました。	2018 年 8 月 14 日
エラーログ	Amazon ES では、Elasticsearch エラーログを Amazon CloudWatch に公開できるようになりました。	2018 年 7 月 31 日
中国 (寧夏) のリザーブドインスタンス	Amazon ES は、中国 (寧夏) リージョンでリザーブドインスタンスの提供を開始しました。	2018 年 5 月 29 日

リザーブドインスタンス	Amazon ES は、リザーブドインスタンスのサポートの提供を開始しました。	2018 年 5 月 7 日

以前の更新

以下のテーブルは、2018 年 5 月以前の Amazon ES の重要な変更点をまとめたものです。

変更	説明	日付
Kibana の Amazon Cognito 認証	Amazon ES で、Kibana のログインページの保護が追加されました。詳細については"the section called "OpenSearch Dashboards の Amazon Cognito 認証""を参照してください。	2018 年 4 月 2 日
Elasticsearch 6.2 のサポート	Amazon OpenSearch Service は、Elasticsearch バージョン 6.2 をサポートするようになりました。	2018 年 3 月 14 日
韓国語分析プラグイン	Amazon ES は、メモリ最適化バージョンの Seunjeon 韓国語分析プラグインをサポートするようになりました。	2018 年 3 月 13 日
アクセスコントロールの瞬時の更新	Amazon ES ドメインでのアクセスコントロールポリシーの変更は、すぐに有効化されるようになりました。	2018 年 3 月 7 日
ペタバイトスケール	Amazon ES は、I3 インスタンスタイプと合計で最大 1.5 PB のドメインストレージをサポートするようになりました。詳細については"the section called "ペタバイトスケール""を参照してください。	2017 年 12 月 19 日
保管時のデータの暗号化	Amazon ES は、保管中のデータの暗号化をサポートするようになりました。詳細については"the section called "保管中の暗号化""を参照してください。	2017 年 12 月 7 日
Elasticsearch 6.0 のサポート	Amazon ES は、Elasticsearch バージョン 6.0 をサポートするようになりました。移行に関する考慮事項と手順については、「"the section called "ドメインのアップグレード""」を参照してください。	2017 年 12 月 6 日

変更	説明	日付
VPC サポート	Amazon ES では、Amazon Virtual Private Cloud 内でドメインを起動できるようになりました。VPC サポートはセキュリティに追加のレイヤーを提供し、VPC における Amazon ES とその他のサービス間の通信を簡素化します。詳細については the section called “VPC サポート” を参照してください。	2017 年 10 月 17 日
スローログの公開	Amazon ES は、CloudWatch Logs へのスローログの公開をサポートするようになりました。詳細については the section called “ログをモニタリングする” を参照してください。	2017 年 10 月 16 日
Elasticsearch 5.5 のサポート	Amazon ES は、Elasticsearch バージョン 5.5 をサポートするようになりました。 サポート に問い合わせることなく自動スナップショットを復元し、`_scripts` API を使用してスクリプトを保存できるようになりました。	2017 年 9 月 7 日
Elasticsearch 5.3 のサポート	Amazon ES は、Elasticsearch バージョン 5.3 のサポートを追加しました。	2017 年 6 月 1 日
クラスターあたりのインスタンスと EBS キャパシティーが増加	Amazon ES は、クラスターあたり最大 100 個のノードと 150 TB の EBS 容量をサポートするようになりました。	2017 年 4 月 5 日
カナダ (中部) と欧州 (ロンドン) のサポート	Amazon ES は、カナダ (中部) ca-central-1 および欧州 (ロンドン) eu-west-2 リージョンのサポートを追加しました。	2017 年 3 月 20 日
より多くのインスタンスと拡大した EBS ボリューム	Amazon ES は、より多くのインスタンスと拡大した EBS ボリュームのサポートを追加しました。	2017 年 2 月 21 日
Elasticsearch 5.1 のサポート	Amazon ES は、Elasticsearch バージョン 5.1 のサポートを追加しました。	2017 年 1 月 30 日

変更	説明	日付
Phonetic Analysis プラグインのサポート	Amazon ES には、Phonetic Analysis プラグインとの統合が組み込まれるようになりました。これにより、データに対して「同音異義の」クエリ実行が可能になります。	2016 年 12 月 22 日
米国東部 (オハイオ) のサポート	Amazon ES は、米国東部 (オハイオ) リージョン (us-east-2) のサポートを追加しました。	2016 年 10 月 17 日
新しいパフォーマンスメトリクス	Amazon ES が新しいパフォーマンスメトリクス `ClusterUsedSpace` を追加しました。	2016 年 7 月 29 日
Elasticsearch 2.3 のサポート	Amazon ES は、Elasticsearch バージョン 2.3 のサポートを追加しました。	2016 年 7 月 27 日
アジアパシフィック (ムンバイ) のサポート	Amazon ES は、アジアパシフィック (ムンバイ) ap-south-1 リージョンのサポートを追加しました。	2016 年 6 月 27 日
クラスターあたりのさらなるインスタンス	Amazon ES は、クラスターごとのインスタンス (インスタンス数) の最大数を 10 から 20 に増やしました。	2016 年 5 月 18 日
アジアパシフィック (ソウル) のサポート	Amazon ES は、アジアパシフィック (ソウル) ap-northeast-2 リージョンのサポートを追加しました。	2016 年 1 月 28 日
Amazon ES	初回リリース。	2015 年 10 月 1 日

AWS 用語集

最新の AWS 用語については、 AWS の用語集 リファレンスのAWS 用語集を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。