ユーザーガイド AWS CodeDeploy



API バージョン 2014-10-06

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

AWS CodeDeploy: ユーザーガイド

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

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客 に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で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

What is CodeDeploy?	1
の利点 AWS CodeDeploy	2
CodeDeploy コンピューティングプラットフォームの概要	3
CodeDeploy デプロイタイプの概要	10
インプレースデプロイの概要	12
Blue/Green デプロイの概要	13
ご意見をお待ちしております	18
プライマリコンポーネント	18
アプリケーション	18
コンピューティングプラットフォーム	19
デプロイ設定	20
デプロイグループ	20
デプロイタイプ	20
デプロイタイプ	22
リビジョン	22
サービスロール	22
ターゲットリビジョン	23
他のコンポーネント	23
デプロイ	23
AWS Lambda コンピューティングプラットフォームのデプロイ	24
Amazon ECS コンピューティングプラットフォームのデプロイ	. 27
EC2/オンプレミスコンピューティングプラットフォームの Blue/Green デプロイ	39
アプリケーション仕様ファイル	46
Amazon ECS コンピューティングプラットフォーム上の AppSpec ファイル	. 47
AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル	47
EC2 オンプレミスコンピューティングプラットフォーム上の AppSpec ファイル	48
CodeDeploy エージェントが AppSpec ファイルを使用する方法	. 48
入門	50
ステップ 1: セットアップ	50
にサインアップする AWS アカウント	50
管理アクセスを持つユーザーを作成する	51
プログラマチックアクセス権を付与する	52
ステップ 2: サービスロールを作成する	54
サービスロールの作成 (コンソール)	56

サービスロールの作成 (CLI)	59
サービスロール ARN の取得 (コンソール)	62
サービスロール ARN の取得 (CLI)	62
ステップ 3: CodeDeploy ユーザーのアクセス許可を制限する	63
ステップ 4: IAM インスタンスプロファイルを作成する	66
Amazon EC2 インスタンス(CLI)の IAM インスタンスプロファイルを作成する	67
Amazon EC2 インスタンス(コンソール)の IAM インスタンスプロファイルを作成する	71
製品およびサービスの統合	75
他の AWS サービスとの統合	75
Amazon EC2 Auto Scaling	83
エラスティックロードバランシング	91
パートナーの製品とサービスとの統合	96
GitHub	101
コミュニティから統合の例	105
ブログ記事	106
チュートリアル	107
チュートリアル: WordPress を Windows 以外のインスタンスヘデプロイする	107
ステップ 1: Amazon EC2 インスタンスを起動する	108
ステップ 2: ソース コンテンツを設定する	111
ステップ 3: Amazon S3 にアプリケーションをアップロードする	116
ステップ 4: アプリケーションをデプロイする	121
ステップ 5: アプリケーションを更新して再デプロイする	128
ステップ 6: クリーンアップする	132
チュートリアル: Windows Server インスタンスへの HelloWorld アプリケーションのデプロ	
1	135
ステップ 1: Amazon EC2 インスタンスを起動する	136
ステップ 2: ソース コンテンツを設定する	138
ステップ 3: Amazon S3 にアプリケーションをアップロードする	142
手順 4: アプリケーションをデプロイする	147
ステップ 5: アプリケーションを更新して再デプロイする	152
ステップ 6: クリーンアップする	155
チュートリアル: オンプレミスインスタンスへアプリケーションをデプロイ	159
前提条件	160
ステップ 1: オンプレミスインスタンスを設定する	160
ステップ 2: サンプルのアプリケーションリビジョンを作成する	160
ステップ 3: アプリケーションリビジョンをバンドルし、Amazon S3 にアップロードする	›. 165

ステップ 4: アプリケーションリビジョンをデプロイする	166
ステップ 5: デプロイを確認する	166
ステップ 6: リソースをクリーンアップする	166
チュートリアル: Auto Scaling グループヘデプロイします。	169
前提条件	169
ステップ 1: Auto Scaling グループを作成して設定します。	170
ステップ 2: Auto Scaling グループにアプリケーションをデプロイする	177
ステップ 3: 結果の確認	187
ステップ 4: Auto Scaling グループの Amazon EC2 インスタンスの数を増やす …	189
ステップ 5: 結果を再度確認します	190
ステップ 6: クリーンアップする	193
チュートリアル: GitHub からアプリケーションをデプロイします。	195
前提条件	196
ステップ 1: GitHub アカウントを設定します。	196
ステップ 2: GitHub リポジトリを作成します。	196
ステップ 3: GitHub リポジトリにサンプルアプリケーションをアップロードしま	す。 199
ステップ 4: インスタンスをプロビジョニングします。	204
ステップ 5: アプリケーションおよびデプロイグループを作成します。	205
ステップ 6: アプリケーションをインスタンスにデプロイします。	207
ステップ 7: デプロイをモニタリングおよび確認します。	211
ステップ 8: クリーンアップする	212
チュートリアル: Amazon ECS ヘアプリケーションをデプロイする	214
前提条件	216
ステップ 1: Amazon ECS アプリケーションを更新する	217
ステップ 2: AppSpec ファイルを作成します。	218
ステップ 3: CodeDeploy コンソールを使用してアプリケーションをデプロイする	o 219
ステップ 4: クリーンアップする	224
チュートリアル: 検証テストを使用して Amazon ECS サービスをデプロイする	224
前提条件	227
ステップ 1: テストリスナーを作成する	227
ステップ 2: Amazon ECS アプリケーションを更新する	228
ステップ 3: ライフサイクルフック Lambda 関数を作成する	228
ステップ 4: AppSpec ファイルを更新する	231
ステップ 5: CodeDeploy コンソールを使用して Amazon ECS サービスをデプロ	イする 232
ステップ 6: CloudWatch Logs で Lambda フック関数の出力を表示する	234
	235

チュートリアル: SAM を使用して Lambda AWS 関数をデプロイする	236
前提条件	237
ステップ 1: インフラストラクチャをセットアップする	237
ステップ 2: Lambda 関数を更新します	252
ステップ 3: 更新された Lambda 関数をデプロイします。	255
ステップ 4: デプロイの結果を表示する	257
ステップ 5:クリーンアップ	260
CodeDeploy エージェントの使用	262
CodeDeploy エージェントで対応するオペレーティングシステム	263
対応する Amazon EC2 AMI オペレーティングシステム	263
サポートされているオンプレミスオペレーションシステム	263
CodeDeploy エージェントの通信プロトコルとポート	263
CodeDeploy エージェントのバージョン履歴	264
CodeDeploy プロセスの管理	279
アプリケーションリビジョンとログファイルのクリーンアッププ	280
CodeDeploy エージェントでインストールされるファイル	280
CodeDeploy エージェントのオペレーションの管理	284
CodeDeploy エージェントが実行されていることの確認	284
CodeDeploy エージェントのバージョンを特定します。	286
CodeDeploy エージェントをインストールする	288
CodeDeploy エージェントを更新する	300
CodeDeploy エージェントをアンインストールする	304
CodeDeploy エージェントログを CloudWatch に送信する	305
インスタンスの使用	310
Amazon EC2 インスタンスとオンプレミスインスタンスの比較	310
CodeDeploy のためのインスタンスタスク	312
CodeDeploy デプロイのインスタンスのタグ付け	313
例 1: 単一タググループ、単一タグ	314
例 2: 単一タググループ、複数タグ	315
例 3: 複数タググループ、複数タグ	317
例 4: 複数タググループ、複数タグ	319
Amazon EC2 インスタンスの使用	323
CodeDeploy のための Amazon EC2 インスタンスを作成します。	323
Amazon EC2 インスタンスを作成する (AWS CloudFormation テンプレート)	330
Amazon EC2 インスタンスの設定	341
オンプレミスインスタンスの使用	345

オンプレミスインスタンスを設定するための前提条件	346
オンプレミスインスタンスの登録	348
オンプレミスインスタンスオペレーションの管理	383
インスタンスの詳細の表示	
インスタンスの詳細の表示 (コンソール)	391
インスタンスの詳細の表示 (CLI)	392
インスタンスの状態	393
[Health status](ヘルスステータス)	393
正常なインスタンスの最小数について	395
アベイラビリティーゾーンあたりの正常なインスタンスの最小数について	399
デプロイ設定の使用	402
EC2/オンプレミスコンピューティングプラットフォームのデプロイ設定	402
事前定義されたデプロイ設定	402
Amazon ECS コンピューティングプラットフォームのデプロイ設定	407
Amazon ECS の事前定義されたデプロイ設定	407
AWS CloudFormation blue/green デプロイのためのデプロイ設定 (Amazon ECS)	408
AWS Lambda コンピューティングプラットフォームのデプロイ設定	409
Lambda の事前定義されたデプロイ設定	409
	410
デプロイ設定を作成する	410
デプロイ設定を作成する (コンソール)	411
デプロイ設定を作成する (AWS CLI)	413
デプロイ設定の詳細の表示	414
デプロイ設定の詳細の表示 (コンソール)	415
デプロイ設定の表示 (CLI)	415
デプロイ設定を削除する	416
アプリケーションを用いた作業	417
アプリケーションの作成	418
インプレースデプロイ (コンソール) 用のアプリケーションを作成	420
Blue/Green デプロイ (コンソール) のアプリケーションを作成します。	423
Amazon FCS サービスデプロイ用のアプリケーションを作成 (コンソール)	428
	400
AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール)	
AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール) アプリケーションの作成 (CLI)	
AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール) アプリケーションの作成 (CLI) アプリケーションの詳細を表示する	430 431 432
AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール) アプリケーションの作成 (CLI) アプリケーションの詳細を表示する アプリケーションの詳細を表示する (コンソール)	430 431 432 432

通知ルールの作成	433
アプリケーションの名前を変更する	436
アプリケーションの削除	436
アプリケーションの削除 (コンソール)	437
アプリケーションの削除 (AWS CLI)	437
デプロイグループの使用	438
Amazon ECS コンピューティングプラットフォームのデプロイでのデプロイグループ	438
AWS Lambda コンピューティングプラットフォームデプロイのデプロイグループ	438
EC2 オンプレミスコンピューティングプラットフォームのデプロイでのデプロイグルー	プ 438
	439
デプロイグループの作成	440
インプレースデプロイ用のデプロイグループを作成する (コンソール)	440
EC2/オンプレミス Blue/Green デプロイ用のデプロイグループを作成する (コンソーノ	ل) 444
Amazon ECS デプロイ用のデプロイグループを作成する (コンソール)	449
CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバランサーマ	をセッ
トアップする	451
CodeDeploy Amazon ECS デプロイ用のロードバランサー、ターゲットグループ、リ	ス
ナーをセットアップする	452
デプロイグループの作成 (CLI)	457
デプロイグループの詳細の表示	459
デプロイグループの詳細の表示 (コンソール)	459
デプロイグループの詳細の表示 (CLI)	460
デプロイグループの設定を変更する	460
デプロイグループの設定を変更する (コンソール)	461
デプロイグループの設定を変更する (CLI)	462
デプロイグループの詳細オプションの設定	463
デプロイグループを削除する	466
デプロイグループを削除する (コンソール)	467
デプロイグループを削除する (CLI)	467
アプリケーションリビジョンの操作	469
リビジョンの計画を立てる	469
AppSpec ファイルを追加する	470
Amazon ECS デプロイ用の AppSpec ファイルを追加する	470
AWS Lambda デプロイ用の AppSpec ファイルを追加する	474
EC2/オンプレミスデプロイに AppSpec ファイルを追加する	476
レポジトリタイプの選択	480

リビジョンをプッシュする	483
を使用してリビジョンをプッシュする AWS CLI	485
アプリケーションリビジョンの詳細の表示	487
アプリケーションリビジョンの詳細の表示 (コンソール)	488
アプリケーションリビジョンの詳細の表示 (CLI)	488
アプリケーションリビジョンの登録	489
CodeDeploy で Amazon S3 にリビジョンを登録する (CLI)	490
CodeDeploy による GitHub でのリビジョンの登録 (CLI)	491
デプロイでの作業	492
デプロイを作成する	493
デプロイの前提条件	494
Amazon ECS コンピューティングプラットフォームのデプロイの作成 (コンソール)	497
AWS Lambda コンピューティングプラットフォームのデプロイの作成 (コンソール)	499
EC2/オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)	501
Amazon ECS コンピューティングプラットフォームのデプロイの作成 (CLI)	. 506
AWS Lambda コンピューティングプラットフォームのデプロイの作成 (CLI)	507
EC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)	509
を使用して Amazon ECS ブルー/グリーンデプロイを作成する AWS CloudFormation	513
デプロイの詳細の表示	516
デプロイの詳細の表示 (コンソール)	517
デプロイの詳細の表示 (CLI)	518
デプロイのログデータの表示	518
Amazon CloudWatch コンソールでのログファイルのデータの表示	519
インスタンスでのログファイルの表示	519
デプロイの停止	522
デプロイ (コンソール) の停止	523
デプロイ (CLI) の停止	523
デプロイを使用した再デプロイおよびロールバック	524
自動ロールバック	524
手動ロールバック	525
ロールバックおよび再デプロイのワークフロー	525
既存のコンテンツでのロールバック動作	526
別の AWS アカウントにアプリケーションをデプロイする	530
ステップ 1: いずれかのアカウントで S3 バケットを作成する	530
ステップ 2: Amazon S3 バケットへのアクセス許可を本番稼働用アカウントの インスタン	/
スプロファイルに付与する	531

ステップ 3: 本番稼働用アカウントでリソースとクロスアカウントロールを作成する	532
ステップ 4: Amazon S3 バケットにアプリケーションリビジョンをアップロードする	533
ステップ 5: クロスアカウントロールを引き受け、アプリケーションをデプロイする	534
ローカルマシンのデプロイパッケージの検証	534
前提条件	535
ローカルのデプロイを作成する。	537
例	540
デプロイのモニタリング	542
自動ツール	543
手動ツール	544
Amazon CloudWatch ツールを使用したデプロイのモニタリンググ	545
CloudWatch アラームを使用したデプロイのモニタリング	545
Amazon CloudWatch Events を使用したデプロイのモニタリング	547
を使用したデプロイのモニタリング AWS CloudTrail	550
CloudTrail での CodeDeploy 情報	550
CodeDeploy ログファイルエントリの理解	551
Amazon SNS イベント通知を使用したデプロイのモニタリング	552
サービスロールへの Amazon SNS アクセス許可の付与	553
CodeDeploy イベントのトリガーを作成	554
デプロイグループのトリガーの編集	562
デプロイグループからトリガーを削除	564
トリガーの JSON データの形式	565
セキュリティ	567
データ保護	567
インターネットトラフィックのプライバシー	568
保管中の暗号化	569
転送中の暗号化	569
暗号化キーの管理	570
アイデンティティ/アクセス管理	570
対象者	570
アイデンティティを使用した認証	571
ポリシーを使用したアクセスの管理	574
が IAM と AWS CodeDeploy 連携する方法	576
AWS CodeDeploy の マネージド (事前定義) ポリシー	581
AWS マネージドポリシーへの CodeDeploy の更新	590
アイデンティティベースのポリシーの例	593

トラブルシューティング	. 601
CodeDeploy アクセス許可 リファレンス	. 602
サービス間での不分別な代理処理の防止	. 611
インシデントへの対応	. 613
CodeDeploy とのすべてのインタラクションの監査	. 613
アラートと障害管理	. 614
コンプライアンス検証	. 615
耐障害性	. 616
インフラストラクチャセキュリティ	. 616
参照資料	. 617
AppSpec ファイルのリファレンス	. 617
Amazon ECS コンピューティングプラットフォームの AppSpec ファイル	618
AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル	. 618
EC2/オンプレミスコンピューティングプラットフォームの AppSpec ファイル	. 618
AppSpec ファイル構造	. 619
AppSpec ファイルの例	. 665
AppSpec ファイルの間隔	. 671
AppSpec ファイルの検証とファイルの場所	672
エージェント設定リファレンス	. 673
関連トピック	. 678
AWS CloudFormation テンプレートリファレンス	. 679
Amazon Virtual Private Cloud で CodeDeploy を使用	. 682
可用性	. 683
CodeDeploy 用の VPC エンドポイントを作成する	. 685
CodeDeploy エージェントと IAM 許可を設定する	. 686
リソース キットのリファレンス	687
リージョン別リソースキットバケット名	. 687
リソースキットの内容	689
リソースキットのファイルのリストの表示	. 691
リソースキットのファイルのダウンロード	. 693
クォータ	. 696
トラブルシューティング	702
一般的なトラブルシューティングの問題	702
一般的なトラブルシューティングのチェックリスト	703
CodeDeploy デプロイリソースは、一部の AWS リージョンでのみ でサポートされていま	
वे	. 704

このガイドの手順が CodeDeploy コンソールと一致しない	705
必要な IAM ロールを取得できない	705
何らかのテキストエディタを使用して AppSpec ファイルとシェルスクリプトを作成する	
と、デプロイが失敗する場合がある	706
macOS の Finder を使用してアプリケーションリビジョンをバンドルすると、デプロイがタ	£
敗することがある	707
EC2/オンプレミスのデプロイに関する問題のトラブルシューティング	707
CodeDeploy プラグイン CommandPoller の認証情報不足エラー	708
「PKCS7 署名メッセージの検証に失敗しました」というメッセージでデプロイが失敗す	
る	709
同じデプロイ先のインスタンスに対する同じファイルのデプロイや再デプロイは失敗し、	
「指定したファイルはこの場所に既に存在するため、デプロイに失敗しました」というエ	
ラーが返されます。	709
ファイルパスが長いと、「そのようなファイルまたはディレクトリはありません」というコ	Ľ
ラーが発生します	711
長時間実行されているプロセスにより、デプロイが失敗することがある	712
AllowTraffic ライフサイクルイベントが失敗し、デプロイログにエラーが報告されない場合	
のトラブルシューティング	714
失敗した ApplicationStop、BeforeBlockTraffic、または AfterBlockTraffic デプロイライフサ	
イクルイベントのトラブルシューティング	715
「不明なエラー: 読み取り用に開いていません」で失敗した DownloadBundle デプロイライ	
フサイクルイベントのトラブルシューティング	716
スキップされたすべてのライフサイクルイベントのエラーをトラブルシューティングす	
る	717
Windows PowerShell スクリプトで、デフォルトで 64 ビットバージョンの Windows	
PowerShell スクリプトを使用できない	719
Amazon ECS のデプロイに関する問題のトラブルシューティング	720
置き換えタスクセットを待っている間にタイムアウトが発生します	721
通知が継続されるのを待っている間のタイムアウトの発生	721
IAM ロールに十分なアクセス許可がありません	722
ステータスコールバックを待っている間に デプロイがタイムアウトした	722
1つ以上のライフサイクルイベントの検証機能が失敗したため デプロイが失敗しました	723
「プライマリタスクセットのターゲットグループけリスナーの後ろにある水亜があります。	120
シュー、シスパンピューシン シューシル ションパン の夜つにのつの安かめりより」 というエラーのため FIR を再新できませんでした	702
Cv ジェン のため、ELD E χ 和くらるとだくした	721
Auto Scalling で区内 9 るこ 7 / ロゴル 天秋 9 るこころ めりよ 9	124

ALB のみが段階的なトラフィックルーティングをサポートしているため、デプロイグルー	
プの作成/更新時には、代わりに AllAtOnce トラフィックルーティングを使用してくださ	
	725
デプロイが成功しても、置き換えタスクセットは Elastic Load Balancing のヘルスチェック	
に失敗し、アプリケーションがタワンしています	726
1つのデプロイグループに複数のロードバランサーをアタッチできますか?	727
ロードバランサーなしで CodeDeploy のブルー/グリーンデプロイを実行できますか? 7	/27
デプロイ中に Amazon ECS サービスを新しい情報で更新する方法を教えてください。 7	/27
AWS Lambda デプロイの問題のトラブルシューティング	728
AWS Lambda ロールバックが設定されていない Lambda デプロイを手動で停止すると、デ	
プロイは失敗します	728
デプロイグループの問題のトラブルシューティング	729
デプロイグループの一部としてインスタンスにタグを付けても、アプリケーションが自動的	
に新しいインスタンスにデプロイされない	729
インスタンスの問題のトラブルシューティング	729
タグは正しく設定する必要がある	730
AWS CodeDeploy エージェントはインスタンスにインストールして実行する必要がありま	
す	730
デプロイ中にインスタンスを削除した場合、デプロイは最大1時間は失敗しません。 7	730
ログファイルの分析によるインスタンスでのデプロイの失敗の調査	731
誤って削除した場合は、新しい CodeDeploy ログファイルを作成します。	731
「InvalidSignatureException – Signature expired: [time] is now earlier than [time]」デプロイ	
エラーのトラブルシューティング	731
GitHub トークンの問題のトラブルシューティング	732
GitHub OAuth トークンが無効です	732
GitHub OAuth トークンの最大数を超えました	732
Amazon EC2 Auto Scaling の問題のトラブルシューティング	733
Amazon EC2 Auto Scaling の一般的なトラブルシューティング	733
CodeDeplovRole は、次の AWS サービスでオペレーションを実行するアクセス許可を付与	
しません: AmazonAutoScaling」エラー	734
Amazon EC2 Auto Scaling グループのインスタンスのプロビジョニングと終了が繰り返さ	-
れてリビジョンをデプロイできない	735
Amazon EC2 Auto Scaling インスタンスを終了または再起動すると、デプロイが失敗する	
場合がある	736
複数のデプロイグループを1つの Amazon EC2 Auto Scaling グループに関連付けることは	
避ける	737

Amazon EC2 Auto Scaling グループの EC2 インスタンスが起動に失敗し、「ハートビー」 のタイムアウト」というエラーが表示される	. 737
Amazon EC2 Auto Scaling ライフサイクルフックの个一致により、Amazon EC2 Auto Scaling グループへの自動デプロイが停止または失敗することがある	. 740
「デプロイグループのインスタンスが見つからないため、デプロイに失敗しました」とい	う
エラー	741
エラーコード	750
関連トピック	. 755
リソース	756
リファレンスガイドとサポートリソース	. 756
サンプル	. 756
ブログ	. 756
AWS ソフトウェア開発キットとツール	756
ドキュメント履歴	. 758
以前の更新	. 777
AWS 用語集	800
	dccci

CodeDeploy とは

CodeDeploy は、Amazon EC2 インスタンスやオンプレミスインスタンス、サーバーレス Lambda 関数、または Amazon ECS サービスに対するアプリケーションのデプロイを自動化するデプロイメ ントサービスです。

以下のような、ほぼ無制限の多様なアプリケーションコンテンツをデプロイできます。

- ・コード
- サーバーレス AWS Lambda 関数
- ウェブファイルおよび設定ファイル
- Executables
- パッケージ
- ・スクリプト
- マルチメディアファイル

CodeDeploy では、サーバーで実行され、Amazon S3 バケット、GitHub リポジトリ、また は Bitbucket リポジトリに保存されているアプリケーションコンテンツをデプロイできま す。CodeDeploy では、サーバーレス Lambda 関数をデプロイすることもできます。既存のコードを 変更することなく CodeDeploy を使用できます。

CodeDeploy を使用すると、以下を容易に行うことができます。

- 新機能の迅速なリリース。
- AWS Lambda 関数のバージョンを更新します。
- アプリケーションのデプロイメント中のダウンタイム回避。
- アプリケーションの更新に伴う繁雑さを処理。エラーの発生しやすい手動デプロイに伴うリスクの 多くを回避できます。

サービスはインフラストラクチャに合わせてスケールするため、1 つのインスタンスまたは数千のイ ンスタンスに簡単にデプロイできます。

CodeDeploy は、設定管理、ソース管理、継続的統合、継続的配信、および継続的デプロイのための 様々なシステムで動作します。詳細については、「製品の統合」を参照してください。 また、CodeDeploy コンソールでは、リポジトリ、ビルドプロジェクト、デプロイアプリケーショ ン、パイプラインなどのリソースをすばやく検索することもできます。[Go to resource (リソースに 移動)] または / キーを押して、リソースの名前を入力します。一致するものはすべてリストに表示 されます。検索では大文字と小文字が区別されません。リソースを表示する権限がある場合のみ表示 されます。詳細については、「<u>AWS CodeDeployのためのアイデンティティおよびアクセス管理</u>」 を参照してください。

トピック

- の利点 AWS CodeDeploy
- CodeDeploy コンピューティングプラットフォームの概要
- CodeDeploy デプロイタイプの概要
- ご意見をお待ちしております
- CodeDeploy プライマリコンポーネント
- <u>CodeDeploy デプロイ</u>
- CodeDeploy アプリケーション仕様 (AppSpec) ファイル

の利点 AWS CodeDeploy

CodeDeploy には、次の利点があります。

- サーバー、サーバーレス、コンテナアプリケーション。CodeDeploy を使用すると、従来のアプリ ケーションと、サーバーレス AWS Lambda 関数バージョンまたは Amazon ECS アプリケーショ ンをデプロイするアプリケーションの両方をサーバーにデプロイできます。
- 自動デプロイ CodeDeploy は、開発、テスト、本番稼働環境にまたがるアプリケーションのデプ ロイを完全に自動化します。CodeDeployは、インフラストラクチャに合わせてスケールするた め、1 つのインスタンスまたは数千のインスタンスにデプロイできます。
- ダウンタイムの最小化。アプリケーションで EC2/オンプレミスコンピューティングプラット フォームを使用している場合、CodeDeployは、アプリケーションの可用性を最大化します。イ ンプレースデプロイの場合、CodeDeployは複数の Amazon EC2 インスタンスにまたがってロー リング更新を実行します。更新のために、一度にオフラインにするインスタンスの数を指定でき ます。Blue/Green デプロイの間、最新アプリケーションのリビジョンは、置き換え先インスタン スにインストールされます。トラフィックは、選択するとすぐに、または新しい環境のテストが 完了した時点で、これらのインスタンスに再ルーティングされます。どちらのデプロイタイプで も、CodeDeployは設定したルールに従ってアプリケーションの状態を追跡します。

- 停止してロールバック。エラーが発生した場合、自動または手動でデプロイを停止してロールバックできます。
- コントロールの一元化。CodeDeploy コンソールまたは AWS CLIでデプロイを起動し、そのス テータスを追跡できます。各アプリケーションのリビジョンがいつデプロイされ、どの Amazon EC2 インスタンスがリストされているかを示すレポートを受け取ります。
- 使い方が簡単。CodeDeployは、プラットフォームに依存せず、すべてのアプリケーションで動作します。セットアップコードは簡単に再利用できます。セットアップコードを簡単に再利用できます。また、CodeDeployはソフトウェアリリースプロセスまたは継続的な配信ツールチェーンと統合できます。
- 同時デプロイ。EC2/オンプレミスコンピューティングプラットフォームを使用する複数のアプリ ケーションがある場合、CodeDeployは同じインスタンスのセットでこれらを同時にデプロイでき ます。

CodeDeploy コンピューティングプラットフォームの概要

CodeDeploy では、アプリケーションを 3 つのコンピューティングプラットフォームにデプロイできます。

EC2/オンプレミス: Amazon EC2 クラウドインスタンス、オンプレミスサーバー、またはその両方とすることができる物理サーバーのインスタンスを記述します。EC2/オンプレミスコンピューティングプラットフォームを使用して作成されたアプリケーションは、実行可能ファイル、設定ファイル、イメージなどで構成できます。

EC2/オンプレミス コンピューティングプラットフォームを使用するデプロイでは、インプレイス または Blue/Green デプロイタイプを使用して、トラフィックをインスタンスに振り分ける方法を 管理できます。詳細については、「CodeDeploy デプロイタイプの概要」を参照してください。

 AWS Lambda: 更新されたバージョンの Lambda 関数で構成されるアプリケーションをデプロイ するために使用されます。は、高可用性コンピューティング構造で構成されるサーバーレスコン ピューティング環境で Lambda 関数 AWS Lambda を管理します。コンピューティングリソース の管理はすべて、によって実行されます AWS Lambda。詳細については、「<u>サーバーレスコン</u> <u>ピューティングとアプリケーション</u>」を参照してください。AWS Lambda および Lambda 関数の 詳細については、「」を参照してくださいAWS Lambda。

Canary 設定、線形設定、または一度にすべての設定を選択して、デプロイ時に更新済み Lambda 関数バージョンにトラフィックを移行する方法を管理できます。 Amazon ECS: Amazon ECS にコンテナ化されたアプリケーションをタスクセットとしてデプロ イするために使用します。CodeDeploy は、アプリケーションの更新バージョンを新しい置き換え タスクセットとしてインストールすることで、blue/Green のデプロイを実行します。CodeDeploy は、元のアプリケーションタスクセットからの本稼働トラフィックを置き換えタスクセットに再 ルーティングします。デプロイが正常に完了すると、元のタスクセットは削除されます。Amazon ECS の詳細については、「Amazon Elastic Container Service」を参照してください。

Canary 設定、線形設定、または一度にすべての設定を選択して、デプロイ時に更新済みタスク セットにトラフィックを移行する方法を管理できます。

Note

Amazon ECS blue/Green デプロイは、CodeDeploy と AWS CloudFormationの両方でサ ポートされています。これらのデプロイの詳細については、以降のセクションで説明しま す。

次の表に、CodeDeploy コンポーネントが各コンピューティングプラットフォームで使用される様子 を示します。詳細については、以下を参照してください。

- CodeDeploy でのデプロイグループの使用
- CodeDeploy でのデプロイグループの使用
- CodeDeploy でデプロイ設定を使用する
- CodeDeploy のアプリケーションリビジョンの操作
- CodeDeploy 内でアプリケーションを用いて作業をする

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
デプロイグループ	ー連のインスタンスにリ ビジョンをデプロイしま す。	可用性の高いコンピュー ティングインフラスト ラクチャで、サーバーレ ス Lambda 関数の新しい バージョンをデプロイし ます。	タスクセッ トとして デプロイす る、コン テナ化され たアプリ ケーショ

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
			ンイプシトッ使本びンリトクテ、さリンストすンシリラよバでEビし、さリョラク用稼才のスラをィデれケのクをるグョガーびッASスまデれケンフ提さ働プテナフ再ンプた一元セ終タ、ンームロクmサをすプたーへィ供れおシスーィルグロアシのッ了イオの、、一設zzー指。ロアーの にるよョト、ッーしイプョタ ミプトアおル定の 定

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
デプロイ	アプリケーションと AppSpec ファイルから 構成される新しいリビ ジョンをデプロイします 。AppSpec は、アプリ ケーションをデプロイグ ループのインスタンスに デプロイする方法を指定 します。	Lambda 関数の 1 つの バージョンから、同じ 関数の新しいバージョン に本稼働トラフィック を移行させます。AppS pec は、デプロイする Lambda 関数のバージョ ンを指定します。	Amazon ECテれケのジ新換セし口すDはセのラをタトテま口にスはまのコ化アシ新ンいタトデしCのタト稼ィきク再ン。完元セ除ったプョバを置スとプまdの、スか働ッ換セルグデ了のッさのリンー、きク

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
デプロイ設定	デプロイの速度と、デプ ロイ中にいつでも使用で きる必要のある正常なイ ンスタンスの最小数を決 定する設定。	更新された Lambda 関数 バージョンにトラフィッ クを移行する方法を決定 する設定。	更新され た Amazon ECS タス クセットに トラフ移行す る方法を決 定。

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
リビジョン	AppSpec ファイルとアプ リケーションファイルの 組み合わせ (例: 実行ファ イル、設定ファイル)。	AppSpec ファイルは Lambda 関数と、ライフ サイクルイベントフック 中に検証テストを実行可 能な Lambda 関数のデプ ロイを指定します。	AppSpec フ下る。 デすのナさアケンむECサスACSク。 ADCン化たリシ含Amazon ECス義 最プシデさるナ 本ラクルン Amazon ECス義 最プシデさるナ 本ラクルン のケンロてン 働ィ再テさ イいティル

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
			・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

CodeDeploy コンポーネ ント	EC2/オンプレミス	AWS Lambda	Amazon ECS
アプリケーション	デプロイグループおよび リビジョンのコレクショ ン。EC2/オンプレミスア プリケーションは、EC2/ オンプレミスコンピュー ティングプラットフォー ムを使用します。	デプロイグループおよび リビジョンのコレクショ ン。AWS Lambda デプ ロイに使用されるアプリ ケーションは、サーバー レス AWS Lambda コン ピューティングプラット フォームを使用します。	デルよジレン フプリンショク。Ama スのプ用アー、ECS コグフリンショ とCS イ れリヨ ス の に に ン フ フ リンシ の の プ 用 ア ー、 Amazon に S コ ー プ ォ ー し さ プ フ リンショ の の の プ 用 ア ー、 、 Ama こ の プ に の プ の の の の の の の の の の の の の の の

CodeDeploy デプロイタイプの概要

CodeDeploy には、2 つのデプロイタイプのオプションがあります。

インプレイスデプロイ: デプロイグループの各インスタンス上のアプリケーションが停止され、最新のアプリケーションリビジョンがインストールされて、新バージョンのアプリケーションが開始され検証されます。ロードバランサーを使用し、デプロイ中はインスタンスが登録解除され、デプロイ完了後にサービスに復元されるようにできます。EC2 オンプレミスコンピューティングプラットフォームを使用するデプロイのみが、インプレイスデプロイを使用できます。インプレイスデプロイの講細については、「インプレースデプロイの概要」を参照してください。

AWS Lambda および Amazon ECS デプロイでは、インプレースデプロイタイプを使用で きません。

- Blue/Green デプロイ: デプロイの動作は、使用するコンピューティングプラットフォームにより異なります。
 - EC2 オンプレミスコンピューティングプラットフォームの Blue/Green: 以下のステップを使用して、デプロイグループのインスタンス (元の環境) がインスタンスの別のセット (置き換え先環境) に置き換えられます。
 - 置き換え先の環境のインスタンスがプロビジョニングされます。
 - 最新のアプリケーションリビジョンは、置き換え先インスタンスにインストールされます。
 - オプションの待機時間は、アプリケーションのテストやシステム検証などのアクティビティに 対して発生します。
 - ・置き換え先環境のインスタンスは、1 つまたは複数の Elastic Load Balancing ロードバラン サーに登録され、トラフィックは、それらに再ルーティングされます。元の環境のインスタン スは、登録が解除され、終了するか、他の使用のために実行することができます。

Note

EC2/オンプレミスのコンピューティングプラットフォームを使用する場合は、blue/ green デプロイが Amazon EC2 インスタンスでのみ機能することに注意してください。

- AWS Lambda または Amazon ECS コンピューティングプラットフォームの Blue/Green: トラ フィックは、Canary、線形、または all-at-onceデプロイ設定に従って増分でシフトされます。
- 経由のブルー/グリーンデプロイ AWS CloudFormation: スタック AWS CloudFormation の更新の ー環として、トラフィックは現在のリソースから更新されたリソースに移行されます。現時点で は、ECS blue/green デプロイのみがサポートされています。

ブルー/グリーンデプロイの詳細については、「<u>Blue/Green デプロイの概要</u>」を参照してください。

CodeDeploy エージェントを使用すると、アプリケーション、デプロイグループ、または AWS アカウントを必要とせずに、サインインしているインスタンスでデプロイを実行でき ます。詳細については、<u>CodeDeploy エージェントを使用してローカルマシン上のデプロイ</u> <u>パッケージを検証する</u> を参照してください。

トピック

- インプレースデプロイの概要
- Blue/Green デプロイの概要

インプレースデプロイの概要

Note

AWS Lambda および Amazon ECS デプロイでは、インプレースデプロイタイプを使用できません。

インプレイスデプロイの仕組みは次のとおりです。

- 最初に、ローカルの開発用マシンまたは同様の環境でデプロイ可能なコンテンツを作成し、アプ リケーション特定ファイル (AppSpecファイル)を追加します。AppSpec ファイルは CodeDeploy に固有です。CodeDeploy で実行するデプロイアクションを定義します。デプロイ可能なコンテ ンツおよび AppSpec ファイルをアーカイブファイルにバンドルし、Amazon S3 バケットまたは GitHub リポジトリにアップロードします。このアーカイブファイルは、アプリケーションリビ ジョン (または単にリビジョン)と呼ばれます。
- 次に、リビジョンを取得する Amazon S3 バケットまたは GitHub リポジトリ、コンテンツをデプ ロイする Amazon EC2 インスタンスのセットなど、デプロイに関する情報を CodeDeploy に提 供します。CodeDeploy は Amazon EC2 インスタンスのセットを デプロイグループ で呼び出し ます。デプロイグループには、個別にタグ付けされた Amazon EC2 インスタンス、Amazon EC2 Auto Scaling グループ内の Amazon EC2 インスタンス、またはその両方が含まれます。

デプロイグループにデプロイする新しいアプリケーションリビジョンを正常にアップロードする たびに、そのバンドルはデプロイグループのターゲットリビジョンとして設定されます。つま り、現在デプロイ対象となっているアプリケーションリビジョンがターゲットリビジョンです。 これは、自動デプロイにプルされるリビジョンでもあります。

- 3. 次に、各インスタンスの CodeDeploy エージェントが CodeDeploy をポーリングして、指定され た Amazon S3 バケットまたは GitHub リポジトリから何をいつ取得するかを決定します。
- 4. 最後に、各インスタンスの CodeDeploy エージェントは、Amazon S3 バケットや GitHub リポジ トリからターゲットリビジョンを取得し、AppSpec ファイルの手順を使用して、コンテンツをイ ンスタンスにデプロイします。

CodeDeploy は、デプロイステータス、デプロイ設定パラメータ、インスタンスの状態を取得できる ように、デプロイのレコードを保持します。

Blue/Green デプロイの概要

Blue/Green デプロイは、アプリケーションの新しいバージョンの変更による中断を最小限に抑えな がら、アプリケーションの更新に使用されます。CodeDeploy は、本番トラフィックを再ルーティン グする前に、古いバージョンと一緒に新しいアプリケーションバージョンをプロビジョニングしま す。

- AWS Lambda: トラフィックは、Lambda 関数の1つのバージョンから同じ Lambda 関数の新しい バージョンに移行されます。
- Amazon ECS: Amazon ECS サービスのタスクセットから、同じ Amazon ECS サービスの最新の 置き換えタスクセットにトラフィックが移行します。
- EC2/オンプレミス: 元の環境内の、あるインスタンスセットから、インスタンスの置き換え先の セットにトラフィックが移行します。

すべての AWS Lambda および Amazon ECS デプロイは Blue/Green です。EC2/オンプレミス デプ ロイは、インプレースまたは Blue/Green です。Blue/Green デプロイには、インプレースデプロイと 比べて多くのメリットがあります。

- トラフィックを再ルーティングするだけで、アプリケーションを新しい置き換え先環境にインストールしてテストし、本稼働環境へデプロイできます。
- EC2/オンプレミスコンピューティングプラットフォームを使用している場合、最新バージョンのアプリケーションに切り替えることで、より迅速になり、信頼性が高まります。これは、トラフィックが終了していない限り、トラフィックを元のインスタンスにルーティングできるためです。インプレースデプロイでは、以前のバージョンのアプリケーションを再デプロイすることによってバージョンをロールバックする必要があります。

- EC2/オンプレミスコンピューティングプラットフォーム を使用している場合、新しいインスタン スは Blue/Green デプロイ向けにプロビジョニングされ、最新のサーバー設定が反映されます。これにより、長時間実行するインスタンスで発生する問題を回避できます。
- AWS Lambda コンピューティングプラットフォームを使用している場合は、トラフィックを元の AWS Lambda 関数バージョンから新しい Lambda AWS 関数バージョンに移行する方法を制御し ます。
- Amazon ECS コンピューティングプラットフォームを使用している場合は、元のタスクセットから新しいタスクセットにトラフィックを移行する方法を制御します。

を使用した Blue/Green デプロイ AWS CloudFormation では、次のいずれかの方法を使用できます。

- AWS CloudFormation デプロイ用の テンプレート: AWS CloudFormation テンプレートを使用し てデプロイを設定すると、デプロイは AWS CloudFormation 更新によってトリガーされます。リ ソースを変更し、テンプレートの変更をアップロードすると、 のスタックの更新によって新しい デプロイ AWS CloudFormation が開始されます。 AWS CloudFormation テンプレートで使用でき るリソースのリストについては、「」を参照してください<u>AWS CloudFormation CodeDeploy リ</u> ファレンスの テンプレート。
- によるブルー/グリーンデプロイ AWS CloudFormation: を使用して AWS CloudFormation、スタッ クの更新を通じてブルー/グリーンデプロイを管理できます。スタックテンプレート内で、トラ フィックルーティングと安定化設定を指定するだけでなく、Blue ソースと Green リソースの両 方を定義します。次に、スタックの更新中に選択したリソースを更新すると、は必要なすべて のグリーンリソース AWS CloudFormation を生成し、指定されたトラフィックルーティングパ ラメータに基づいてトラフィックをシフトし、ブルーリソースを削除します。詳細については、 [AWS CloudFormationユーザーガイド]の [AWS CloudFormation を使用した CodeDeploy による Perform Amazon ECS Blue/Green デプロイの実行] を参照してください。

Note

Amazon ECS Blue/Green デプロイでのみサポートされます。

Blue/Green デプロイを設定する方法は、使用するコンピューティングプラットフォームによって異なります。

AWS Lambda または Amazon ECS コンピューティングプラットフォームへのブルー/ グリーンデプロイ

AWS Lambda または Amazon ECS コンピューティングプラットフォームを使用している場合は、ト ラフィックを元の AWS Lambda 関数または Amazon ECS タスクセットから新しい関数またはタス クセットに移行する方法を指定する必要があります。トラフィックを移行する方法を指定するには、 以下のいずれかのデプロイ設定を指定する必要があります。

- canary
- 線形
- 一度にすべて

Canary 設定、線形設定、または一度にすべてのデプロイ設定でトラフィックを移行する方法については、「<u>デプロイ設定</u>」を参照してください。

Lambda デプロイ設定の詳細については、「<u>AWS Lambda コンピューティングプラットフォームの</u> <u>デプロイ設定」を参照してください。</u>

Amazon ECS デプロイ設定の詳細については、「<u>Amazon ECS コンピューティングプラットフォー</u> ムのデプロイ設定」を参照してください。

EC2/オンプレミスコンピューティングプラットフォームの Blue/Green デプロイ

Note

EC2/オンプレミスコンピューティングプラットフォームでの Blue/Green デプロイに は、Amazon EC2 インスタンスを使用する必要があります。オンプレミスインスタンスは Blue/Green デプロイタイプではサポートされません。

EC2/オンプレミスコンピューティングプラットフォームを使用している場合は、次が適用されます。

Amazon EC2 タグまたは Amazon EC2 Auto Scaling グループを識別する 1 つ以上の Amazon EC2 インスタンスが必要です。インスタンスは、以下の条件を満たす必要があります。

 各 Amazon EC2 インスタンスには、適切な IAM インスタンスプロファイルが添付されている必要 があります。 • 各インスタンスで CodeDeploy エージェントをインストールして実行する必要があります。

Note

通常は、元の環境のインスタンスでアプリケーションリビジョンも実行しますが、これは Blue/Green デプロイの要件ではありません。

Blue/Green デプロイで使用されるデプロイグループを作成するときは、置き換え先環境の指定方法 を選択できます。

既存の Amazon EC2 Auto Scaling グループのコピー: Blue/Green デプロイでは、CodeDeploy は、デプロイ中に置き換え先環境用のインスタンスを作成します。このオプションを使用する と、CodeDeploy は、指定した Amazon EC2 Auto Scaling グループを置き換え先環境のテンプレー トとして使用します。これには、同じ数の実行中のインスタンスと他の多くの設定オプションが含ま れます。

手動でインスタンスを選択: Amazon EC2 インスタンスタグ、Amazon EC2 Auto Scaling グループ 名、またはその両方を使用して、置き換え先として含めるインスタンスを指定できます。このオプ ションを選択した場合、デプロイを作成するまで置き換え先環境のインスタンスを指定する必要はあ りません。

処理の流れ

- 元の環境となるインスタンスや Amazon EC2 Auto Scaling グループは既にあります。Blue/Green デプロイを初めて実行するときは、通常、インプレースデプロイで既に使用されているインスタ ンスを使用します。
- 既存の CodeDeploy アプリケーションでは、Blue/Green デプロイグループを作成し、インプレー スデプロイに必要なオプションに加えて、次を指定します。
 - ブルー/グリーンデプロイプロセス中に元の環境から置き換え先環境にトラフィックをルーティングするロードバランサー。
 - ・置き換え先環境にトラフィックを直ちに再ルーティングするか、手動で再ルーティングするま
 で待つかどうか。
 - トラフィックが置き換え先インスタンスにルーティングされるレート。
 - 置き換え元インスタンスを削除するか引き続き実行するかどうか。
- 3. このデプロイグループのデプロイを作成するときには、次の処理が行われます。

- a. Amazon EC2 Auto Scaling グループをコピーすることを選択した場合、インスタンスは置き換 え先環境にプロビジョニングされます。
- b. デプロイに指定したアプリケーションリビジョンは、置き換え先インスタンスにインストール されます。
- c. デプロイグループの設定で待機時間を指定した場合、デプロイは一時停止します。これは置き 換え先環境のテストおよび確認を実行できる時間です。待機時間が終了する前にトラフィック を手動で再ルーティングしない場合、デプロイは停止します。
- d. 置き換え先環境のインスタンスは Elastic Load Balancing ロードバランサーに登録され、これ らのインスタンスに対してトラフィックのルーティングが開始されます。
- e. 元の環境のインスタンスは、終了するか引き続き実行するか、デプロイグループの指定に従っ て登録が解除され、処理されます。

によるブルー/グリーンデプロイ AWS CloudFormation

AWS CloudFormation テンプレートを使用してリソースをモデリングすることで、CodeDeploy Blue/ Green のデプロイを管理できます。

AWS CloudFormation テンプレートを使用して Blue/Green リソースをモデル化する場合、タス クセット AWS CloudFormation を更新するスタック更新を に作成します。本稼働トラフィック は、すべて一度に、線形デプロイとベイク処理時間を使用して、または Canary デプロイを使用し てサービスの元のタスクセットから置き換えタスクセットにシフトします。スタックの更新によ り、CodeDeploy でデプロイが開始されます。CodeDeploy でデプロイのステータスと履歴を表示で きますが、それ以外の場合は、 AWS CloudFormation テンプレートの外部で CodeDeploy リソース を作成または管理しません。

Note

からブルー/グリーンデプロイでは AWS CloudFormation、CodeDeploy アプリケーションま たはデプロイグループを作成しません。

この方法では、Amazon ECS Blue/Green デプロイのみサポートされます。によるブルー/グリーンデ プロイの詳細については AWS CloudFormation、「」を参照してください<u>を使用して Amazon ECS</u> <u>ブルー/グリーンデプロイを作成する AWS CloudFormation</u>。

ご意見をお待ちしております

ご意見をお待ちしております。お問い合わせの場合には、<u>CodeDeploy フォーラム</u> をご利用くださ い。

トピック

- Primary Components
- Deployments
- Application Specification Files

CodeDeploy プライマリコンポーネント

サービスの使用をスタートする前に、CodeDeploy デプロイプロセスの主なコンポーネントを理解し ておく必要があります。

トピック

- アプリケーション
- コンピューティングプラットフォーム
- デプロイ設定
- <u>デプロイグループ</u>
- デプロイタイプ
- <u>IAM インスタンスプロファイル</u>
- リビジョン
- サービスロール
- ターゲットリビジョン
- 他のコンポーネント

アプリケーション

アプリケーションは、デプロイするアプリケーションを一意に識別する名前です。CodeDeploy はこ の名前をコンテナとして使用して、デプロイ中にリビジョン、デプロイ設定、およびデプロイグルー プの正しい組み合わせが参照されるようにします。

コンピューティングプラットフォーム

コンピューティングプラットフォームは、CodeDeploy がアプリケーションをデプロイするプラット フォームです。コンピューティングプラットフォームは 3 つあります。

EC2/オンプレミス: Amazon EC2 クラウドインスタンス、オンプレミスサーバー、またはその両方とすることができる物理サーバーのインスタンスを記述します。EC2/オンプレミスコンピューティングプラットフォームを使用して作成されたアプリケーションは、実行可能ファイル、設定ファイル、イメージなどで構成できます。

EC2/オンプレミス コンピューティングプラットフォームを使用するデプロイでは、インプレイス または Blue/Green デプロイタイプを使用して、トラフィックをインスタンスに振り分ける方法を 管理できます。詳細については、「CodeDeploy デプロイタイプの概要」を参照してください。

 AWS Lambda: 更新されたバージョンの Lambda 関数で構成されるアプリケーションをデプロイ するために使用されます。は、高可用性コンピューティング構造で構成されるサーバーレスコン ピューティング環境で Lambda 関数 AWS Lambda を管理します。コンピューティングリソース の管理はすべて、によって実行されます AWS Lambda。詳細については、「<u>サーバーレスコン</u> <u>ピューティングとアプリケーション</u>」を参照してください。AWS Lambda および Lambda 関数の 詳細については、「」を参照してくださいAWS Lambda。

Canary 設定、線形設定、または一度にすべての設定を選択して、デプロイ時に更新済み Lambda 関数バージョンにトラフィックを移行する方法を管理できます。

 Amazon ECS: Amazon ECS にコンテナ化されたアプリケーションをタスクセットとしてデプロ イするために使用します。CodeDeploy は、アプリケーションの更新バージョンを新しい置き換え タスクセットとしてインストールすることで、blue/Green のデプロイを実行します。CodeDeploy は、元のアプリケーションタスクセットからの本稼働トラフィックを置き換えタスクセットに再 ルーティングします。デプロイが正常に完了すると、元のタスクセットは削除されます。Amazon ECS の詳細については、「Amazon Elastic Container Service」を参照してください。

Canary 設定、線形設定、または一度にすべての設定を選択して、デプロイ時に更新済みタスク セットにトラフィックを移行する方法を管理できます。

Note

Amazon ECS Blue/Green デプロイは、CodeDeploy と AWS CloudFormationの両方でサポー トされています。これらのデプロイの詳細については、以降のセクションで説明します。

デプロイ設定

デプロイ設定は、デプロイ中に CodeDeploy で使用されるデプロイルールとデプロイの成功条件と 失敗条件のセットです。デプロイで EC2/オンプレミスコンピューティングプラットフォームを使用 している場合、デプロイの正常なインスタンスの最小数を指定できます。デプロイで AWS Lambda または Amazon ECS コンピューティングプラットフォームを使用している場合は、更新された Lambda 関数または ECS タスクセットにトラフィックをルーティングする方法を指定できます。

EC2/オンプレミスコンピューティングプラットフォームを使用したデプロイに対する正常なホスト の最小数の指定については、「正常なインスタンスの最小数について」を参照してください。

以下のデプロイ設定では、Lambda または ECS コンピューティングプラットフォームを使用するデ プロイの間にトラフィックをルーティングする方法を指定します。

- Canary: トラフィックは2つの増分で移行されます。事前定義された canary オプションから選択できます。これらのオプションでは、更新された Lambda 関数または ECS タスクセットに、2回目の増分で移行される前に最初の増分および間隔 (分単位) で移行される、トラフィックの割合(%) が指定されています。
- Linear: トラフィックは等しい増分で移行され、増分間の間隔 (分) も同じです。増分ごとに移行するトラフィックの割合 (%) と、増分間の間隔 (分) を指定する、事前定義済み線形オプションから選択できます。
- 一度にすべて: すべてのトラフィックを元の Lambda 関数または ECS タスクセットから更新され た関数またはタスクセットに同時に移行します。

デプロイグループ

デプロイグループは、個々のインスタンスのセットです。デプロイグループには、個別にタグ付けさ れた Amazon EC2 インスタンス、Amazon EC2 Auto Scaling グループ内の Amazon EC2 インスタン ス、またはその両方が含まれます。Amazon EC2 インスタンスタグの詳細については、「<u>コンソー</u> <u>ルでのタグの操作</u>」を参照してください。オンプレミスインスタンスの詳細については、「<u>Working</u> <u>with On-Premises Instances</u>」を参照してください。Amazon EC2 Auto Scaling の情報に関しては、 「<u>CodeDeploy と Amazon EC2 Auto Scaling の統合</u>」を参照してください。

デプロイタイプ

デプロイタイプは、デプロイグループ内のインスタンスで最新のアプリケーションリビジョンを使用 できるようにするために使用される方法です。次の 2 種類のデプロイタイプがあります。

- インプレイスデプロイ: デプロイグループの各インスタンス上のアプリケーションが停止され、最新のアプリケーションリビジョンがインストールされて、新バージョンのアプリケーションが開始され検証されます。ロードバランサーを使用し、デプロイ中はインスタンスが登録解除され、デプロイ完了後にサービスに復元されるようにできます。EC2 オンプレミスコンピューティングプラットフォームを使用するデプロイのみが、インプレイスデプロイを使用できます。インプレイスデプロイの講細については、「インプレースデプロイの概要」を参照してください。
- Blue/Green デプロイ: デプロイの動作は、使用するコンピューティングプラットフォームにより異なります。
 - EC2 オンプレミスコンピューティングプラットフォームの Blue/Green: 以下のステップを使用して、デプロイグループのインスタンス (元の環境) がインスタンスの別のセット (置き換え先環境) に置き換えられます。
 - 置き換え先の環境のインスタンスがプロビジョニングされます。
 - 最新のアプリケーションリビジョンは、置き換え先インスタンスにインストールされます。
 - オプションの待機時間は、アプリケーションのテストやシステム検証などのアクティビティに 対して発生します。
 - ・置き換え先環境のインスタンスは、1 つまたは複数の Elastic Load Balancing ロードバラン サーに登録され、トラフィックは、それらに再ルーティングされます。元の環境のインスタン スは、登録が解除され、終了するか、他の使用のために実行することができます。

EC2/オンプレミスのコンピューティングプラットフォームを使用する場合は、blue/ green デプロイが Amazon EC2 インスタンスでのみ機能することに注意してください。

- AWS Lambda または Amazon ECS コンピューティングプラットフォームの Blue/Green: トラ フィックは、Canary、線形、または all-at-onceデプロイ設定に従って増分でシフトされます。
- 経由のブルー/グリーンデプロイ AWS CloudFormation: スタック AWS CloudFormation の更新の 一環として、トラフィックが現在のリソースから更新されたリソースに移行されます。現時点で は、ECS blue/green デプロイのみがサポートされています。

ブルー/グリーンデプロイの詳細については、「<u>Blue/Green デプロイの概要</u>」を参照してください。

Amazon ECS blue/Green デプロイは、CodeDeploy と AWS CloudFormationの両方でサポー トされています。これらのデプロイの詳細については、以降のセクションで説明します。

IAM インスタンスプロファイル

IAM インスタンスプロファイルは、Amazon EC2 インスタンス にアタッチする IAM ロールです。こ のプロファイルには、アプリケーションが保存される Amazon S3 バケットまたは GitHub リポジト リへのアクセスに必要な権限が含まれます。詳細については、「<u>ステップ 4: Amazon EC2 インスタ</u> ンス用の IAM インスタンスプロファイルを作成する」を参照してください。

リビジョン

リビジョンは、アプリケーションのバージョンです。 AWS Lambda デプロイリビジョンは、デプロ イする Lambda 関数に関する情報を指定する YAML 形式または JSON 形式のファイルです。EC2/オ ンプレミスデプロイリビジョンは、ソースコンテンツ (ソースコード、ウェブページ、実行可能ファ イル、デプロイスクリプト) とアプリケーション仕様ファイル (AppSpec ファイル) を含むアーカイ ブファイルです。 AWS Lambda リビジョンは Amazon S3 バケットに保存できます。EC2/オンプレ ミスのリビジョンは Amazon S3 バケットまたは GitHub リポジトリに格納されます。Amazon S3 で は、リビジョンの Amazon S3 オブジェクトキー、ETag、バージョン、またはその両方により、リ ビジョンが一意に識別されます。GitHub では、コミット ID により、リビジョンが一意に識別されま す。

サービスロール

サービスロールは、 サービスが AWS リソースにアクセスできるように AWS サービスにアク セス許可を付与する IAM ロールです。サービスロールにアタッチするポリシーによって、サー ビスがアクセスできる AWS リソースと、それらのリソースで実行できるアクションが決まりま す。CodeDeploy では、サービスロールは、以下の目的で使用されます。

- インスタンスに適用されているタグやインスタンスに関連付けられている Amazon EC2 Auto Scaling グループ名を読み取ることができます。これにより、CodeDeploy はアプリケーションを デプロイできるインスタンスを識別できます。
- インスタンス、Amazon EC2 Auto Scaling グループ、および Elastic Load Balancing ロードバラン サーでオペレーションを実行するには。
- 指定したデプロイまたはインスタンスイベントが発生したときに通知を送信できるよう に、Amazon SNS トピックに情報を公開すること。
- CloudWatch アラームに関する情報を取得して、デプロイのアラームモニタリングを設定します。

詳細については、「ステップ 2: CodeDeployのサービスのロールを作成する」を参照してください。

ターゲットリビジョン

ターゲットリビジョンは、リポジトリにアップロードし、デプロイグループ内のインスタンスにデプ ロイしたいアプリケーションリビジョンの最新バージョンです。つまり、現在デプロイの対象として いるアプリケーションリビジョン。これは、自動デプロイにプルされるリビジョンでもあります。

他のコンポーネント

CodeDeploy ワークフローの他のコンポーネントの詳細については、次のトピックを参照してください。

- CodeDeploy リポジトリタイプを選択する
- Deployments
- Application Specification Files
- Instance Health
- CodeDeploy エージェントの使用
- Working with On-Premises Instances

CodeDeploy デプロイ

このトピックでは、CodeDeploy のデプロイのコンポーネントおよびワークフローについて説明しま す。デプロイプロセスは、デプロイに使用するコンピューティングプラットフォームまたはデプロイ 方法 (Lambda、Amazon ECS、EC2/オンプレミス、または 経由 AWS CloudFormation) によって異 なります。

トピック

- AWS Lambda コンピューティングプラットフォームのデプロイ
- Amazon ECS コンピューティングプラットフォームのデプロイ
- EC2/オンプレミスコンピューティングプラットフォームの Blue/Green デプロイ

AWS Lambda コンピューティングプラットフォームのデプロイ

このトピックでは、 AWS Lambda コンピューティングプラットフォームを使用する CodeDeploy デ プロイのコンポーネントとワークフローについて説明します。

トピック

- AWS Lambda コンピューティングプラットフォームのデプロイワークフロー
- アプリケーションリビジョンのアップロード
- アプリケーションとデプロイグループの作成
- アプリケーションリビジョンのデプロイ
- アプリケーションの更新
- 停止、失敗したデプロイ
- デプロイと再デプロイのロールバック

AWS Lambda コンピューティングプラットフォームのデプロイワークフロー

次の図は、新規および更新された AWS Lambda 関数のデプロイの主要なステップを示しています。



ステップには以下が含まれます。

- アプリケーションを作成し、デプロイするアプリケーションリビジョンを一意に識別する名前を 付けます。Lambda 関数をデプロイするには、AWS Lambda コンピューティングプラットフォー ムは、アプリケーションを作成するときに使用します。CodeDeploy は、この名前を使用して、デ プロイグループ、デプロイ設定、アプリケーションリビジョンなど、正しいデプロイコンポーネ ントを参照していることを確認します。詳細については、「<u>CodeDeploy でアプリケーションを作</u> 成する」を参照してください。
- 2. デプロイグループを設定するには、デプロイグループの名前を指定します。
- 3. デプロイ設定を選択して、トラフィックを元の AWS Lambda 関数バージョンから新しい Lambda 関数バージョンに移行する方法を指定します。詳細については、「<u>View Deployment</u> Configuration Details」を参照してください。
- アプリケーション仕様ファイル(AppSpec ファイル) を Amazon S3 にアップロードしま す。AppSpec ファイルは、デプロイを検証するために使用される Lambda 関数のバージョンと Lambda 関数を指定します。AppSpec ファイルを作成しない場合は、YAML または JSON を使用 して、コンソールで直接 Lambda 関数バージョンと Lambda デプロイ検証用の関数を指定できま す。詳細については、「<u>CodeDeploy のアプリケーションリビジョンの操作</u>」を参照してくださ い。
- 5. アプリケーションリビジョンをデプロイグループにデプロイします。 は、指定した Lambda 関数 リビジョンを AWS CodeDeploy デプロイします。トラフィックを Lambda 関数リビジョンに移行 するには、アプリケーション作成時に選択した AppSpec ファイルのデプロイを使用します。詳細 については、「CodeDeploy でデプロイを作成する」を参照してください。
- 6. デプロイの結果を確認します。詳細については、「<u>CodeDeploy でのデプロイモニタリング</u>」を参 照してください。

アプリケーションリビジョンのアップロード

Amazon S3 に AppSpec ファイルを配置するか、コンソールまたは AWS CLIに直接入力します。詳 細については、「Application Specification Files」を参照してください。

アプリケーションとデプロイグループの作成

AWS Lambda コンピューティングプラットフォーム上の CodeDeploy デプロイグループは、1 つ以 上の AppSpec ファイルのコレクションを識別します。AppSpec ファイルごとに 1 つの Lambda 関 数バージョンをデプロイできます。デプロイグループは、今後のデプロイのために、アラームおよび ロールバックの設定などの設定オプションのセットを定義します。

アプリケーションリビジョンのデプロイ

これで、AppSpec ファイル で指定した関数リビジョンをデプロイグループにデプロイする準備がで きました。CodeDeploy コンソールまたは <u>create-deployment</u> コマンドを使用できます。デプロイを 制御するために指定できるパラメータ (リビジョン、デプロイグループ、デプロイ設定など) があり ます。

アプリケーションの更新

アプリケーションを更新し、CodeDeploy コンソールを使用するか、<u>create-deployment</u> コマンドを 呼び出してリビジョンをプッシュできます。

停止、失敗したデプロイ

デプロイを停止するには、CodeDeploy コンソールまたは<u>stop-deployment</u>コマンドを使用できま す。デプロイを停止しようとする場合、次の 3 つのうち 1 つのことが発生します。

- デプロイは停止し、オペレーションは成功というステータスを返す。この場合、停止したデプロイ に対してそれ以上デプロイライフサイクルイベントは実行されません。
- デプロイは即時に停止せず、オペレーションは保留中というステータスを返す。この場合、一部の デプロイライフサイクルイベントは、デプロイグループでまだ実行中である可能性があります。保 留中のオペレーションが完了すると、デプロイを停止するためのそれ以降の呼び出しは、成功とい うステータスを返します。
- デプロイは停止できず、オペレーションはエラーを返す。詳細については、 AWS CodeDeploy API リファレンスのErrorInformation」と「一般的なエラー」を参照してください。

失敗したデプロイでは、停止されたデプロイのように、一部のデプロイライフサイクルイベントが実 行済みになる場合があります。デプロイが失敗した理由を調べるには、CodeDeploy コンソールを使 用するか、失敗したデプロイのログファイルデータを分析することができます。詳細については、<u>ア</u> <u>プリケーションリビジョンとログファイルのクリーンアップ</u>および<u>CodeDeploy EC2/オンプレミスデ</u> プロイのログデータの表示を参照してください。

デプロイと再デプロイのロールバック

CodeDeploy は新しいデプロイとして、以前にデプロイされたリビジョンを再デプロイすることに よって、ロールバックを実装します。 デプロイが失敗した、アラームのモニタリングしきい値に一致したなど、特定の条件が満たされた場合に、自動的にデプロイをロールバックするようグループデプロイを設定できます。個別のデプロイで、デプロイグループに指定されたロールバック設定をオーバーライドすることもできます。

以前のデプロイされたバージョンを手動で再デプロイして、失敗したデプロイをロールバックするこ ともできます。

いずれの場合でも、新しいデプロイまたはロールバックされたデプロイには独自のデプロイ ID が割 り当てられます。CodeDeploy コンソールで表示できるデプロイの一覧には、どれが自動デプロイの 結果であるかが示されます。

詳細については、「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照して ください。

Amazon ECS コンピューティングプラットフォームのデプロイ

このトピックでは、Amazon ECS コンピューティングプラットフォームを使用する CodeDeploy の デプロイのコンポーネントおよびワークフローについて説明します。

トピック

- Amazon ECS デプロイを開始する前に
- Amazon ECS コンピューティングプラットフォームのデプロイワークフロー (高レベル)
- Amazon ECS デプロイ中の処理で起こっていること
- アプリケーションリビジョンのアップロード
- アプリケーションとデプロイグループの作成
- アプリケーションリビジョンのデプロイ
- アプリケーションの更新
- 停止、失敗したデプロイ
- デプロイと再デプロイのロールバック
- AWS CloudFormationを通じた Amazon ECS blue/green デプロイのデプロイ

Amazon ECS デプロイを開始する前に

Amazon ECS アプリケーションのデプロイを開始する前に、次の準備が完了している必要がありま す。デプロイグループを作成するときに指定される要件と、AppSpec ファイルで指定される要件が あります。

要件	指定される場所
Amazon ECS クラスター	デプロイグループ
Amazon ECS サービス	デプロイグループ
Application Load Balancer および Network Load Balancer	デプロイグループ
本稼働リスナー	デプロイグループ
テストリスナー (オプション)	デプロイグループ
2 つのターゲットグループ	デプロイグループ
Amazon ECS タスク定義	AppSpec ファイル
コンテナ名	AppSpec ファイル
コンテナポート	AppSpec ファイル

Amazon ECS クラスター

Amazon ECSクラスターは、タスクまたはサービスの論理グループです。CodeDeploy アプリ ケーションのデプロイグループを作成するときに、Amazon ECS サービスを含む Amazon ECS クラスターを指定します。詳細については、「<u>Amazon Elastic Container Service ユーザーガイ</u> <u>ド</u>」の「Amazon ECS のクラスター」を参照してください。

Amazon ECS サービス

Amazon ECS サービスは、Amazon ECS クラスター内のタスク定義で指定されたインスタンス を維持し、実行します。Amazon ECS サービスが CodeDeploy で有効になっている必要があり ます。デフォルトでは、Amazon ECS サービスは、Amazon ECS デプロイで有効になっていま す。デプロイグループを作成するときは、Amazon ECS クラスター内の Amazon ECS サービス をデプロイすることを選択します。詳細については、「Amazon Elastic Container Service ユー ザーガイド」の「Amazon ECS のクラスター」を参照してください。

Application Load Balancer および Network Load Balancer

Elastic Load Balancing は、Amazon ECS デプロイで更新する Amazon ECS サービスで使用す る必要があります。Application Load Balancer または Network Load Balancer を作成できます。 動的ポートマッピング、パスベースのルーティング、優先ルールなどの機能を利用できるよう に、Application Load Balancer をお勧めします。CodeDeploy アプリケーションのデプロイグ ループを作成するときに、ロードバランサーを指定します。詳細については、「Amazon Elastic Container Service ユーザーガイド」の <u>CodeDeploy Amazon ECS デプロイ用のロードバラン</u> <u>サー、ターゲットグループ、リスナーをセットアップする</u> と「<u>ロードバランサーの作成</u>」を参照 してください。

1 つまたは 2 つのリスナー

ロードバランサーは、リスナーを使用してターゲットグループにトラフィックをルーティング します。本稼働リスナーが1つ必要です。検証テストの実行中、置き換えタスクセットにトラ フィックをルーティングする、2番目のオプションのテストリスナーを指定できます。デプロイ グループを作成するときに、一方または両方のリスナーを指定します。Amazon ECS コンソール を使用して Amazon ECS サービスを作成すると、リスナーが作成されます。詳細については、 「Elastic Load Balancing ユーザーガイド」の「<u>Application Load Balancer のリスナー</u>」そして 「Amazon Elastic Container Service ユーザーガイド」の「<u>サービスの作成</u>」を参照してくださ い。

2 つの Amazon ECS ターゲットグループ

ターゲットグループは、登録済みターゲットにトラフィックをルーティングするために使用され ます。Amazon ECS デプロイには 2 つのターゲットグループが必要です。1 つは Amazon ECS アプリケーションの元のタスクセット用、もう 1 つはその置き換えタスクセット用です。デプロ イ中、CodeDeploy は置き換えタスクセットを作成し、元のタスクセットから新しいタスクセッ トにトラフィックを再ルーティングします。CodeDeploy アプリケーションのデプロイグループ を作成するときに、ターゲットグループを指定します。

デプロイ中、CodeDeploy は、ステータス PRIMARY (これが元のタスクセット)を持つ、Amazon ECS サービスのタスクセットに関連付けられているターゲットグループを決定し、それに一方 のターゲットグループを関連付けます。さらに、もう一方のターゲットグループを置き換えタス クセットと関連付けます。別のデプロイを行う場合、現在のデプロイの元のタスクセットに関 連付けられているターゲットグループは、次のデプロイの置き換えタスクセットに関連付けられ ています。詳細については、「Elastic Load Balancingのユーザーガイド」の「<u>Application Load</u> Balancer のターゲットグループ」を参照してください。

Amazon ECS タスク定義

Amazon ECS アプリケーションを含んだ Docker コンテナを実行するには、タスク定義 が必要で す。CodeDeploy アプリケーションの AppSpec ファイルで、タスク定義の ARN を指定します。 詳細については、「Amazon Elastic Container Service ユーザーガイド」と Amazon ECS デプロ <u>イ用の AppSpec の「resources」セクション</u> の「<u>Amazon ECS のタスクロール</u>」を参照してく ださい。

Amazon ECS アプリケーション用のコンテナ

Docker コンテナは、コードとその依存関係をパッケージ化してアプリケーションを実行できる ようにするソフトウェアのユニットです。コンテナはアプリケーションを分離して、さまざまな コンピューティング環境で実行できるようにします。ロードバランサーは、Amazon ECS アプ リケーションのタスクセットのコンテナにトラフィックをルーティングします。CodeDeploy ア プリケーションの AppSpec ファイルで、コンテナの名前を指定します。AppSpec ファイル で指 定したコンテナは、Amazon ECS タスク定義で指定したコンテナのいずれかである必要がありま す。詳細については、「Amazon Elastic Container Service ユーザーガイド」と <u>Amazon ECS デ</u> <u>プロイ用の AppSpec の「resources」セクション</u>の「<u>Amazon Elastic Container Service とは</u>」 を参照してください。

置き換えタスクセット用のポート

Amazon ECS デプロイ中に、ロードバランサーが CodeDeploy アプリケーションの AppSpec ファイルで指定されたコンテナ上の ポート にトラフィックを指定します。CodeDeploy アプリ ケーションの AppSpec ファイル でポートを指定します。詳細については、「<u>Amazon ECS デプ</u> ロイ用の AppSpec の「resources」セクション」を参照してください。

Amazon ECS コンピューティングプラットフォームのデプロイワークフロー (高レベル)

次の図表は、更新された Amazon ECS サービスのデプロイの主要なステップを示しています。



ステップには以下が含まれます。

- デプロイするものを一意に表す名前を指定して、AWS CodeDeploy アプリケーションを作成します。Amazon ECS アプリケーションをデプロイするには、AWS CodeDeploy アプリケーションで Amazon ECS コンピューティングプラットフォームを選択します。CodeDeploy は、デプロイ中にアプリケーションを使用して、デプロイグループ、ターゲットグループ、リスナー、トラフィックの再ルーティング動作、およびアプリケーションリビジョンなどの正しいデプロイコンポーネント参照します。詳細については、「CodeDeploy でアプリケーションを作成する」を参照してください。
- 2. デプロイグループをセットアップするには、以下を指定します。
 - デプロイグループ名。
 - お客様の Amazon ECS クラスターとサービス名 Amazon ECS サービスのデプロイコントロー ラーは、CodeDeploy に設定する必要があります。
 - 本稼働リスナー、オプションのテストリスナー、およびターゲットグループは、デプロイ中に 使用されます。

- 本稼働トラフィックを Amazon ECS サービスの置き換え Amazon ECS タスクセットに再ルー ティングするタイミングや、Amazon ECS サービスの元の Amazon ECS タスクセットを終了 するタイミングなどのデプロイ設定。
- ・トリガー、アラーム、ロールバック動作などのオプション設定。
- アプリケーション仕様ファイル (AppSpec ファイル) を指定します。Amazon S3 にアップロード したり、YAML または JSON 形式でコンソールに入力したり、 AWS CLI または SDK で指定した りできます。AppSpec ファイル は、デプロイの Amazon ECS タスク定義、トラフィックをルー ティングするコンテナ名とポートマッピング、およびデプロイのライフサイクルフックの後で実 行される Lambda 関数を指定するために使用されます。コンテナ名は、Amazon ECS タスク定義 内のコンテナである必要があります。詳細については、「<u>CodeDeploy のアプリケーションリビ</u> ジョンの操作」を参照してください。
- アプリケーションリビジョンをデプロイします。AWS CodeDeploy rerout は、Amazon ECS サー ビスのタスクセットの元のバージョンから新しい置き換えタスクセットにトラフィックをデプロ イします。デプロイグループで指定されたターゲットグループは、元のタスクセットと置き換え タスクセットにトラフィックを提供するために使用されます。デプロイが完了すると、元のタス クセットは削除されます。トラフィックが再ルーティングされる前に、テストトラフィックを置 き換えバージョンに提供するためのオプションのテストリスナーを指定できます。詳細について は、「CodeDeploy でデプロイを作成する」を参照してください。
- 5. デプロイの結果を確認します。詳細については、「<u>CodeDeploy でのデプロイモニタリング</u>」を参 照してください。

Amazon ECS デプロイ中の処理で起こっていること

テストリスナーを使用して Amazon ECS デプロイを開始する前に、そのコンポーネントを設定す る必要があります。詳細については、「<u>Amazon ECS デプロイを開始する前に</u>」を参照してくださ い。

次の図表は、Amazon ECS デプロイを開始する準備ができたときのこれらのコンポーネント間の関 係を示しています。



デプロイが開始されたら、デプロイライフサイクルイベントが一度に1つずつ実行され始めます。 ライフサイクルイベントの中には、AppSpecファイル で指定されている Lambda 関数のみを実行す るフックがあります。次の表のデプロイのライフサイクルイベントは、実行された順序で一覧表示 されています。詳細については、「<u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクショ</u> <u>ン</u>」を参照してください。

ライフサイクルイベント	ライフサイクルイベントアクション
BeforeInstall (Lambda 関数のフック)	Lambda 関数を実行します。
インストール	代替タスクの設定を行います。
AfterInstall (Lambda 関数のフック)	Lambda 関数を実行します。
AllowTestTraffic	テストリスナーからターゲットグループ 2 にト ラフィックをルーティングします。
AfterAllowTestTraffic (Lambda 関数 のフック)	Lambda 関数を実行します。
BeforeAllowTraffic (Lambda 関数の フック)	Lambda 関数を実行します。

ライフサイクルイベント	ライフサイクルイベントアクション
AllowTraffic	本稼働リスナーからターゲットグループ 2 にト ラフィックをルーティングします。
AfterAllowTraffic	Lambda 関数を実行します。

Note

フックの Lambda 関数はオプションです。

1.

AppSpec ファイルの BeforeInstall フックで指定された Lambda 関数を実行します。

2.

Install ライフサイクルイベント中:

- a. 代替タスクセットが Amazon ECS サービスで作成されます。
- b. 更新後のコンテナ化されたアプリケーションは、置き換えタスクセットにインストールされ ます。
- c. 2番目のターゲットグループは置き換えタスクセットに関連付けられています。

この図は、新しい置き換えタスクセットを含むデプロイコンポーネントを示しています。コンテ ナ化されたアプリケーションはこのタスクセット内にあります。タスクセットは3つのタスク で構成されています。(アプリケーションには任意の数のタスクを含めることができます。)2番 目のターゲットグループが置き換えタスクセットに関連付けられました。



3.

AppSpec ファイルの AfterInstall フックで指定された Lambda 関数を実行します。

4.

AllowTestTraffic イベントが呼び出されます。このライフサイクルイベントの間、テストリ スナーは、更新されたコンテナ化アプリケーションにトラフィックをルーティングします。



5.

AppSpec ファイルの AfterAllowTestTraffic フックで指定された Lambda 関数を実行 します。Lambda 関数は、テストトラフィックを使用してデプロイを検証します。たとえ ば、Lambda 関数はテストリスナーにトラフィックを送信し、置き換えタスクセットのメトリク スを追跡できます。ロールバックが設定されている場合は、Lambda 関数内の検証テストが失敗 したときにロールバックをトリガーする CloudWatch アラームを設定できます。

検証テストが完了したら、次のいずれかが発生します。

- 検証が失敗し、ロールバックが設定されている場合、デプロイステータスは Failed とマークされ、コンポーネントはデプロイが開始されたときの状態に戻ります。
- 検証が失敗し、ロールバックが設定されていない場合、デプロイステータスは Failed と マークされ、コンポーネントは現在の状態のまま変わりません。
- 検証が正常に完了すると、デプロイは引き続き BeforeAllowTraffic に進みます。

詳細については<u>CodeDeploy での CloudWatch アラームを使用したデプロイのモニタリング、自</u> 動ロールバック、およびデプロイグループの詳細オプションの設定を参照してください。

6.

AppSpec ファイルの BeforeAllowTraffic フックで指定された Lambda 関数を実行します。

7.

AllowTraffic イベントが呼び出されます。本稼働トラフィックは、元のタスクセットから置 き換えタスクセットに再ルーティングされます。次の図は、本稼働トラフィックを受信している 代替タスクセットを示しています。



8.

AppSpec ファイルの AfterAllowTraffic フックで指定された Lambda 関数を実行します。

9.

すべてのイベントが正常に完了したら、デプロイステータスは Succeeded になり、元のタスク セットは削除されます。



アプリケーションリビジョンのアップロード

Amazon S3 に AppSpec ファイルを配置するか、コンソールまたは AWS CLIに直接入力します。詳 細については、「Application Specification Files」を参照してください。

アプリケーションとデプロイグループの作成

Amazon ECS コンピューティングプラットフォーム上の CodeDeploy デプロイグループは、更新さ れた Amazon ECS アプリケーションへのトラフィックを提供するリスナー、およびデプロイ中に 使用される 2 つのターゲットグループを識別します。デプロイグループは、アラームおよびロール バックの設定などの設定オプションのセットも定義します。

アプリケーションリビジョンのデプロイ

これで、デプロイグループで指定された、更新された Amazon ECS サービスをデプロイする準備が 整いました。CodeDeploy コンソールまたは <u>create-deployment</u> コマンドを使用できます。デプロイ を制御するために指定できるパラメータ (リビジョン、デプロイグループなど) があります。

アプリケーションの更新

アプリケーションを更新し、CodeDeploy コンソールを使用するか、<u>create-deployment</u> コマンドを 呼び出してリビジョンをプッシュできます。

停止、失敗したデプロイ

デプロイを停止するには、CodeDeploy コンソールまたは<u>stop-deployment</u>コマンドを使用できま す。デプロイを停止しようとする場合、次の 3 つのうち 1 つのことが発生します。

- デプロイは停止し、オペレーションは成功というステータスを返す。この場合、停止したデプロイ
 に対してそれ以上デプロイライフサイクルイベントは実行されません。
- デプロイは即時に停止せず、オペレーションは保留中というステータスを返す。この場合、一部の デプロイライフサイクルイベントは、デプロイグループでまだ実行中である可能性があります。保 留中のオペレーションが完了すると、デプロイを停止するためのそれ以降の呼び出しは、成功とい うステータスを返します。
- ・デプロイは停止できず、オペレーションはエラーを返す。詳細については、 AWS CodeDeploy API リファレンスの「エラー情報」と「一般的なエラー」を参照してください。

デプロイと再デプロイのロールバック

CodeDeploy は、置き換えタスクセットから元のタスクセットにトラフィックを再ルーティングする ことにより、ロールバックを実装します。

デプロイが失敗した、アラームのモニタリングしきい値に一致したなど、特定の条件が満たされた場合に、自動的にデプロイをロールバックするようグループデプロイを設定できます。個別のデプロイで、デプロイグループに指定されたロールバック設定をオーバーライドすることもできます。

以前のデプロイされたバージョンを手動で再デプロイして、失敗したデプロイをロールバックするこ ともできます。

いずれの場合でも、新しいデプロイまたはロールバックされたデプロイには独自のデプロイ ID が割 り当てられます。CodeDeploy コンソールには、自動デプロイの結果であるデプロイの一覧が表示さ れます。

デプロイする場合、現在のデプロイの元のタスクセットに関連付けられているターゲットグループ は、デプロイの置き換えタスクセットに関連付けられています。

詳細については、「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照して ください。

AWS CloudFormationを通じた Amazon ECS blue/green デプロイのデプロイ

を使用して AWS CloudFormation 、CodeDeploy を通じて Amazon ECS ブルー/グリーンデプロイを 管理できます。詳細については、「<u>を使用して Amazon ECS ブルー/グリーンデプロイを作成する</u> AWS CloudFormation」を参照してください。

Note

を使用した Amazon ECS ブルー/グリーンデプロイの管理 AWS CloudFormation は、アジア パシフィック (大阪) リージョンでは利用できません。

EC2/オンプレミスコンピューティングプラットフォームの Blue/Green デ プロイ

このトピックでは、EC2 オンプレミスのコンピューティングプラットフォームを使用する CodeDeploy のデプロイのコンポーネントおよびワークフローについて説明します。Blue/Green デ プロイの詳細については、「Blue/Green デプロイの概要」を参照してください。

トピック

- EC2/オンプレミスコンピューティングプラットフォームのデプロイコンポーネント
- EC2/オンプレミスコンピューティングプラットフォームのデプロイワークフロー
- <u>インスタンスの設定</u>
- アプリケーションリビジョンのアップロード
- アプリケーションとデプロイグループの作成
- アプリケーションリビジョンのデプロイ
- アプリケーションの更新
- <u>停止、失敗したデプロイ</u>
- デプロイと再デプロイのロールバック

EC2/オンプレミスコンピューティングプラットフォームのデプロイコンポーネント

以下の図表は、EC2/オンプレミスコンピューティングプラットフォームの CodeDeploy デプロイの コンポーネントを示します。



EC2/オンプレミスコンピューティングプラットフォームのデプロイワークフロー

次の図は、アプリケーションリビジョンのデプロイの主要なステップを示しています。



ステップには以下が含まれます。

- アプリケーションを作成し、デプロイするアプリケーションリビジョンとアプリケーションのコンピューティングプラットフォームを一意に識別する名前を付けます。CodeDeploy は、この名前を使用して、デプロイグループ、デプロイ設定、アプリケーションリビジョンなど、正しいデプロイコンポーネントを参照していることを確認します。詳細については、「CodeDeploy でアプリケーションを作成する」を参照してください。
- アプリケーションリビジョンをデプロイするデプロイタイプとインスタンスを指定して、デプロ イグループをセットアップします。インプレースデプロイでは、最新のアプリケーションリビ ジョンでインスタンスを更新します。Blue/Green デプロイはロードバランサーでデプロイグルー プ用の代替セットを登録し、元のインスタンスを登録解除します。

インスタンス、Amazon EC2 Auto Scaling グループ名、または両方に適用するタグを指定できま す。

デプロイグループのタグのグループを指定すると、CodeDeploy は指定されたタグの少なくと も 1 つが適用されたインスタンスにデプロイします。2 つ以上のタググループを指定した場 合、CodeDeploy はそれぞれのタググループの条件を満たすインスタンスにのみデプロイします。 詳細については、「<u>Tagging Instances for Deployments</u>」を参照してください。 いずれの場合も、インスタンスはデプロイで使用するよう設定されていること (つまり、タグが付いているか、Amazon EC2 Auto Scaling グループに所属していること)、および CodeDeploy エージェントをインストールして実行していることが必要です。

Amazon Linux または Windows Server に基づいて Amazon EC2 インスタンスをすばやくセット アップするために使用できる AWS CloudFormation テンプレートが用意されています。また、ス タンドアロン CodeDeploy エージェントも提供しています。これにより、Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL)、または Windows Server インスタンスにインストール できます。詳細については、「<u>CodeDeploy でデプロイグループを作成する</u>」を参照してくださ い。

また、以下のオプションを指定することもできます。

- Amazon SNS の通知 成功イベントや失敗イベントなど、指定されたイベントがデプロイとイン スタンスで発生したときに、Amazon SNS トピックの受信者に通知を送信するトリガーを作成 します。詳細については、「<u>Monitoring Deployments with Amazon SNS Event Notifications</u>」を 参照してください。
- アラームベースのデプロイ管理。CloudWatch で設定したしきい値を超えるか下回ったときに、 デプロイを停止する Amazon CloudWatch アラームモニタリングを実装します。
- ・ 自動デプロイロールバック。デプロイが失敗するか、アラームのしきい値に一致したときに、 以前の既知の正常なリビジョンに自動的にロールバックするようデプロイを設定します。
- アプリケーションのリビジョンを同時にデプロイする必要があるインスタンスの数と、デプロ イの成功と失敗の条件を示すために、デプロイ構成を指定します。詳細については、「<u>View</u> <u>Deployment Configuration Details</u>」を参照してください。
- アプリケーションリビジョンを Amazon S3 または GitHub にアップロードします。デプロイす るファイルおよびデプロイ中に実行するスクリプトに加えて、アプリケーション 仕様ファイル (AppSpec ファイル) を含める必要があります。このファイルには、ファイルを各インスタンスに コピーする場所や、デプロイスクリプトを実行するタイミングなど、デプロイの手順が含まれて います。詳細については、「<u>CodeDeploy のアプリケーションリビジョンの操作</u>」を参照してくだ さい。
- デプロイグループにアプリケーションリビジョンをデプロイします。デプロイグループの各イン スタンスの CodeDeploy エージェントは、Amazon S3 または GitHub からインスタンスにアプリ ケーションリビジョンをコピーします。次に、CodeDeploy エージェントは、リビジョンをバンド ル解除し、AppSpec ファイル を使用してファイルを指定された場所にコピーして、デプロイスク リプトを実行します。詳細については、「CodeDeploy でデプロイを作成する」を参照してくださ い。

- 6. デプロイの結果を確認します。詳細については、「<u>CodeDeploy でのデプロイモニタリング</u>」を参 照してください。
- ワビジョンをデプロイします。ソースコンテンツのバグを修正する、別の順序でデプロイスクリ プトを実行する、または失敗したデプロイに対応する必要がある場合に、この作業を行います。 これを行うには、変更したソースコンテンツ、デプロイスクリプト、および AppSpec ファイル を新しいリビジョンを再バンドルし、このリビジョンを Amazon S3 バケットまたは GitHub リポ ジトリにアップロードします。次に、新しいリビジョンで同じデプロイグループに新しいデプロ イを実行します。詳細については、「CodeDeploy でデプロイを作成する」を参照してください。

インスタンスの設定

アプリケーションリビジョンを初めてデプロイする前に、インスタンスを設定する必要があります。 アプリケーションリビジョンで 3 つの本番稼働用サーバーと 2 つのバックアップサーバーが必要な 場合、5 つのインスタンスを起動または使用します。

インスタンスを手動でプロビジョニングするには:

- 1. CodeDeploy エージェントをインスタンスにインストールする CodeDeploy エージェント は、Amazon Linux、Ubuntu サーバー、RHEL、および Windows Server インスタンスにインス トールできます。
- タグを使用してデプロイグループのインスタンスを識別する場合は、タグ付けを有効にしま す。CodeDeploy は、CodeDeploy デプロイグループにインスタンスを識別し、グループ化するタ グを使用します。入門チュートリアルでは両方を使用しましたが、キーまたは値を使用して、デ プロイグループのタグを定義できます。
- IAM インスタンスプロファイルをアタッチして、Amazon EC2 インスタンスを起動しま す。CodeDeploy エージェントでインスタンスの ID を検証するには、Amazon EC2 インスタンス を起動する際に IAM インスタンスプロファイルをアタッチする必要があります。
- 4. サービスロールを作成します。CodeDeploy が AWS アカウントのタグを拡張できるように、サー ビスアクセスを提供します。

最初のデプロイでは、 AWS CloudFormation テンプレートがこれをすべて実行します。これにより、CodeDeploy エージェントがインストール済みの Amazon Linux または Windows Serverをベー スに、一つの新規Amazon EC2 インスタンスを作成および設定することができます。詳細について は、「<u>CodeDeploy のためにインスタンスを用いた操作</u>」を参照してください。

Note

Blue/Green デプロイでは、置き換え先環境用の既存のインスタンスを使用するか、デプロイ プロセスの一部として CodeDeploy で新しいインスタンスをプロビジョニングするか選択で きます。

アプリケーションリビジョンのアップロード

アプリケーションのソースコンテンツフォルダ構造で、ルートフォルダの下に AppSpec ファイル を 配置します。詳細については、「Application Specification Files」を参照してください。

zip、tar、または圧縮された tar などのアーカイブファイル形式にアプリケーションのソースコンテ ンツフォルダ構造をバンドルします。アーカイブファイル (リビジョン) を Amazon S3 バケットまた は GitHub リポジトリにアップロードします。

Note

tar および圧縮 tar アーカイブファイル形式(.tar および .tar.gz)は、Windows Server インス タンスではサポートされていません。

アプリケーションとデプロイグループの作成

CodeDeploy デプロイグループはタグ、Amazon EC2 Auto Scaling グループ名、または両方に基づい てインスタンスのコレクションを識別します。複数のアプリケーションリビジョンを同じインスタン スにデプロイできます。1 つのアプリケーションリビジョンを複数のインスタンスにデプロイできま す

たとえば、3 つの本番稼働用サーバーに「Prod」というタグを追加し、2 つのバックアップサーバー に「Backup」というタグを追加できます。これら 2 つのタグを使用して、CodeDeploy アプリケー ションで 2 つの異なるデプロイグループを作成し、デプロイにどちらのサーバーのセットを参加さ せるか (または両方を参加させるか) 選択することができます。

デプロイグループの複数のタググループを使用して、デプロイするインスタンスのセットを減らすことができます。詳細については、Tagging Instances for Deployments を参照してください。

アプリケーションリビジョンのデプロイ

これで、Amazon S3 または GitHub からデプロイグループにアプリケーションリビジョンをデプロ イできる状態になりました。CodeDeploy コンソールまたは <u>create-deployment</u> コマンドを使用でき ます。デプロイを制御するために指定できるパラメータ (リビジョン、デプロイグループ、デプロイ 設定など) があります。

アプリケーションの更新

アプリケーションを更新し、CodeDeploy コンソールを使用するか、<u>create-deployment</u> コマンドを 呼び出してリビジョンをプッシュできます。

停止、失敗したデプロイ

デプロイを停止するには、CodeDeploy コンソールまたは<u>stop-deployment</u>コマンドを使用できま す。デプロイを停止しようとする場合、次の 3 つのうち 1 つのことが発生します。

- デプロイは停止し、オペレーションは成功というステータスを返す。この場合、停止したデプロイ に対してそれ以上デプロイライフサイクルイベントは実行されません。デプロイグループで一部の ファイルは既にコピーされ、一部のスクリプトは実行され、1つ以上のインスタンスが実行されて いる可能性があります。
- デプロイは即時に停止せず、オペレーションは保留中というステータスを返す。この場合、一部の デプロイライフサイクルイベントは、デプロイグループでまだ実行中である可能性があります。デ プロイグループで一部のファイルは既にコピーされ、一部のスクリプトは実行され、1つ以上のイ ンスタンスが実行されている可能性があります。保留中のオペレーションが完了すると、デプロイ を停止するためのそれ以降の呼び出しは、成功というステータスを返します。
- デプロイは停止できず、オペレーションはエラーを返す。詳細については、AWS CodeDeploy
 API リファレンスのErrorInformation」と「一般的なエラー」を参照してください。

失敗したデプロイでは、停止されたデプロイのように、デプロイグループの1つ以上のインスタン スで一部のデプロイライフサイクルイベントが実行済みになる場合があります。デプロイが失敗した 理由を調べるには、CodeDeploy コンソールを使用して、get-deployment-instance コマンドを呼び 出すか、失敗したデプロイのログファイルデータを分析することができます。詳細については、<u>アプ</u> リケーションリビジョンとログファイルのクリーンアップおよび<u>CodeDeploy EC2/オンプレミスデプ</u> ロイのログデータの表示を参照してください。

デプロイと再デプロイのロールバック

CodeDeploy は新しいデプロイとして、以前にデプロイされたリビジョンを再デプロイすることに よって、ロールバックを実装します。

デプロイが失敗した、アラームのモニタリングしきい値に一致したなど、特定の条件が満たされた場 合に、自動的にデプロイをロールバックするようグループデプロイを設定できます。個別のデプロイ で、デプロイグループに指定されたロールバック設定をオーバーライドすることもできます。

以前のデプロイされたバージョンを手動で再デプロイして、失敗したデプロイをロールバックするこ ともできます。

いずれの場合でも、新しいデプロイまたはロールバックされたデプロイには独自のデプロイ ID が割 り当てられます。CodeDeploy コンソールで表示できるデプロイの一覧には、どれが自動デプロイの 結果であるかが示されます。

詳細については、「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照して ください。

CodeDeploy アプリケーション仕様 (AppSpec) ファイル

アプリケーション仕様ファイル (AppSpec ファイル) は、<u>YAML</u> 形式、または <u>JSON</u> 形式のファイル であり、CodeDeploy に固有のものです。AppSpec ファイルは、ファイルで定義された一連のライ フサイクルイベントフックとして、各デプロイを管理するために使用されます。

正しい形式のAppSpec ファイルを作成する方法については、「<u>CodeDeploy AppSpec ファイルのリ</u>ファレンス」を参照してください。

トピック

- Amazon ECS コンピューティングプラットフォーム上の AppSpec ファイル
- AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル
- EC2 オンプレミスコンピューティングプラットフォーム上の AppSpec ファイル
- CodeDeploy エージェントが AppSpec ファイルを使用する方法

Amazon ECS コンピューティングプラットフォーム上の AppSpec ファイル

アプリケーションで Amazon ECS コンピューティングプラットフォームを使用している場合 は、AppSpec ファイルを YAML または JSON 形式にすることができます。また、コンソールのエ ディタに直接入力することもできます。AppSpec ファイルは、以下を指定するために使用されま す。

- Amazon ECS サービスの名称と、新しいタスクセットにトラフィックを送信するために使用されるコンテナの名称およびポート。
- 検証テストとして使用される関数。

Lambda 関数は、デプロイライフサイクルイベント後に検証を実行できます。詳細について は<u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション</u>、<u>Amazon ECS デプロイ向け</u> <u>の AppSpec ファイル構造</u>、および<u>Amazon ECS デプロイの AppSpec ファイルの例</u>を参照してく ださい。

AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル

アプリケーションで AWS Lambda コンピューティングプラットフォームを使用している場 合、AppSpec ファイルは YAML または JSON のいずれかでフォーマットできます。また、コンソー ルのエディタに直接入力することもできます。AppSpec ファイルは、以下を指定するために使用さ れます。

- ・ デプロイする AWS Lambda 関数のバージョン。
- 検証テストとして使用される関数。

Lambda 関数は、デプロイライフサイクルイベント後に検証を実行できます。詳細については、 「AWS の Lambda デプロイ向けの AppSpec の「hooks」セクション」を参照してください。

EC2 オンプレミスコンピューティングプラットフォーム上の AppSpec ファイル

アプリケーションで EC2 オンプレミスコンピューティングプラットフォームを使用している場合、AppSpec ファイルは必ず YAML 形式になります。AppSpec ファイルは、以下を目的として使用 されます。

- アプリケーションリビジョンのソースファイルを、インスタンスの宛先にマッピングします。
- デプロイされたファイルのカスタムアクセス権限を指定する。
- デプロイプロセスのさまざまなフェーズにおいて、各インスタンスで実行するスクリプトを指定する。

個別のデプロイライフサイクルイベントのうちの多くは、発生後にインスタンス上でスクリプトの 実行が可能となっています。CodeDeploy はファイルで指定されているスクリプトのみを実行します が、これらのスクリプトは、インスタンスの他のスクリプトを呼び出すことができます。インスタン スで実行しているオペレーティングシステムでサポートされている限り、どのタイプのスクリプトで も実行できます。詳細については、「EC2/オンプレミスのデプロイ向けの AppSpec の「hooks」セ クション」を参照してください。

CodeDeploy エージェントが AppSpec ファイルを使用する方法

デプロイ中、CodeDeploy エージェントは、現在のイベントの名前を AppSpec ファイル内の フック セクションで検索できます。イベントが見つからない場合、CodeDeploy エージェントは次のステッ プに進みます。イベントが見つかった場合、CodeDeploy エージェントは実行するスクリプトのリス トを取得します。スクリプトはファイルに表示された順序に沿って実行されます。各スクリプトのス テータスはインスタンス上の CodeDeploy エージェントログファイルに記録されます。

スクリプトが正常に実行すると、終了コード0(ゼロ)を返します。

Note

CodeDeploy エージェントは、 AWS Lambda または Amazon ECS デプロイでは使用されま せん。

Install イベント中、CodeDeploy エージェントは AppSpec ファイルの files セクション内に定義され ているマッピングを使用して、リビジョンからインスタンスにコピーするフォルダまたはファイルを 判断します。 オペレーティングシステムにインストールされている CodeDeploy エージェントが AppSpec ファイ ルに記載されているものと一致しない場合、デプロイは失敗します。

CodeDeploy エージェントログファイルの詳細については、<u>CodeDeploy エージェントの使用</u> を参照 してください。

CodeDeploy の開始方法

トピック

- <u>ステップ 1: セットアップ</u>
- ステップ 2: CodeDeployのサービスのロールを作成する
- ステップ 3: CodeDeploy ユーザーのアクセス許可を制限する
- ステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成する

ステップ 1: セットアップ

AWS CodeDeploy を初めて使用する場合は、セットアップ手順を完了する必要があります。ステッ プでは、 AWS アカウント (まだ持っていない場合) と、プログラムによるアクセス権を持つ管理 ユーザーを作成します。

このガイドでは、管理ユーザーは CodeDeploy 管理ユーザーと呼ばれます。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

- 1. https://portal.aws.amazon.com/billing/signup を開きます。
- 2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力 するように求められます。

にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。 ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があ ります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルー トユーザーのみを使用して<u>ルートユーザーアクセスが必要なタスク</u>を実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。<u>https://</u> <u>aws.amazon.com/</u> の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビ ティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 のセキュリティを確保し AWS IAM Identity Center、 を有効に して管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

 ルートユーザーを選択し、AWS アカウントEメールアドレスを入力して、アカウント所有 者<u>AWS Management Console</u>として にサインインします。次のページでパスワードを入力しま す。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイ ドのルートユーザーとしてサインインするを参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAM <u>ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮</u> 想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「<u>AWS IAM Identity Centerの</u> 有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリア ルについては、 AWS IAM Identity Center ユーザーガイドの<u>「デフォルトを使用してユーザーア</u> クセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

 IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティ センターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン 「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

管理アクセスを持つユーザーを作成する

追加のユーザーにアクセス権を割り当てる

 IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラク ティスに従ったアクセス許可セットを作成します。

手順については、「AWS IAM Identity Center ユーザーガイド」の「<u>権限設定を作成する</u>」を参 照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てま す。

手順については、「AWS IAM Identity Center ユーザーガイド」の「<u>グループの結合</u>」を参照し てください。

これで、CodeDeploy 管理ユーザーとして作成してサインインできました。

プログラマチックアクセス権を付与する

ユーザーが の AWS 外部とやり取りする場合は、プログラムによるアクセスが必要です AWS Management Console。プログラムによるアクセスを許可する方法は、 がアクセスするユーザーの タイプによって異なります AWS。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選択しま す。

プログラマチックアクセス権 を必要とするユーザー	目的	方法
ワークフォースアイデンティ ティ (IAM アイデンティティセン ターで管理されているユー ザー)	ー時的な認証情報を使用 して、 AWS CLI、 AWS SDKs、または AWS APIs。	使用するインターフェイスの 指示に従ってください。 ・ については AWS CLI、AWS Command Line Interface 「ユーザーガイド」の「を 使用する AWS CLI ように AWS IAM Identity Centerを 設定する」を参照してくだ
		さい。

プログラマチックアクセス権 を必要とするユーザー	目的	方法
		・AWS SDKs、ツール、API については、SDK および AWS APIs <u>「IAM Identity</u> <u>Center 認証</u> 」を参照してく ださい。AWS SDKs
IAM	ー時的な認証情報を使用 して、 AWS CLI、 AWS SDKs、または AWS APIs。	「IAM <u>ユーザーガイド」の</u> <u>「 AWS リソースでの一時的</u> <u>な認証情報</u> の使用」の手順に 従います。
IAM	(非推奨) 長期認証情報を使用して、 AWS CLI、AWS SDKs、また は AWS APIs。	使用するインターフェイスの 指示に従ってください。 ・ については AWS CLI、 「AWS Command Line Interface ユーザーガイド」 の「IAM ユーザー認証情報 を使用した認証」を参照し てください。 ・ AWS SDKs「SDK とツー ルリファレンスガイド」の 「長期的な認証情報を使用 した認証」を参照してくだ さい。AWS SDKs ・ API AWS APIs「IAM ユー ザーガイド」の「IAM ユー ザーガイド」の「IAM ユー

▲ Important

CodeDeploy 管理ユーザーをワークフォース ID (IAM Identity Center で管理されるユーザー) として AWS CLIで設定することを強くお勧めします。このガイドの手順の多くは、 を使用 して設定を実行することを前提 AWS CLI としています。

▲ Important

を設定すると AWS CLI、 AWS リージョンを指定するように求められる場合があります。 「AWS 全般のリファレンス」の「<u>リージョンエンドポイント</u>」に記載されているサポート されているリージョンから一つを選びます。

ステップ 2: CodeDeployのサービスのロールを作成する

では AWS、サービスロールを使用して AWS サービスにアクセス許可を付与し、 AWS リソースに アクセスできるようにします。サービスロールにアタッチするポリシーによって、どの リソースに サービスがアクセスできるか、およびそれらのリソースで何ができるかが決まります。

CodeDeploy に作成するサービスロールで、コンピューティングプラットフォームに必要なアクセス 許可がを付与する必要があります。 複数のコンピューティングプラットフォームにデプロイした場 合、それぞれにサービスロールを1つずつ作成します。アクセス許可を追加するには、次の AWS 指 定されたポリシーを1つ以上アタッチします。

EC2/オンプレミスのデプロイには、AWSCodeDeployRole ポリシーをアタッチします。これは、 サービスロールで以下を実行するためのアクセス許可を提供します。

- ・インスタンスのタグを読み取る、または Amazon EC2 Auto Scaling グループ名により Amazon EC2 インスタンスを識別します。
- Amazon EC2 Auto Scaling グループ、ライフサイクルフック、スケーリングポリシーの読み取り、作成、更新、削除を行います。
- Amazon SNS トピックに情報を公開します。
- CloudWatch アラームに関する情報を取得します。
- Elastic Load Balancing を読み、更新します。

Note

起動テンプレートを使用して Auto Scaling グループを作成する場合は、次のアクセス許可 を追加する必要があります。

- ec2:RunInstances
- ec2:CreateTags
- iam:PassRole

詳細については、「<u>Amazon EC2 Auto Scaling ユーザーガイド</u>」、「<u>Auto Scaling グルー</u> <u>プの起動テンプレートの作成</u>」、および「起動テンプレートサポート」にはの「<u>ステップ</u> <u>2: サービスロールを作成する</u>」を参照してください。

Amazon ECS のデプロイの場合、サポートサービスへのフルアクセスが必要な場合は、

AWSCodeDeployRoleForECS ポリシーをアタッチします。これは、サービスロールで以下を実行 するためのアクセス許可を提供します。

- Amazon ECS タスクセットを読んで、更新、削除します。
- Elastic Load Balancing ターゲットグループ、リスナー、ルールを更新します。
- AWS Lambda 関数を呼び出します。
- Amazon S3 バケットのリビジョンファイルにアクセスします。
- CloudWatch アラームに関する情報を取得します。
- Amazon SNS トピックに情報を公開します。

Amazon ECS のデプロイの場合、サポートサービスへの制限付きのアクセスが必要な場合 は、AWSCodeDeployRoleForECSLimited ポリシーをアタッチします。これは、サービスロール で以下を実行するためのアクセス許可を提供します。

- Amazon ECS タスクセットを読んで、更新、削除します。
- CloudWatch アラームに関する情報を取得します。
- Amazon SNS トピックに情報を公開します。

AWS Lambda デプロイの場合、Amazon SNS への発行を許可するには、 AWSCodeDeployRoleForLambdaポリシーをアタッチします。これは、サービスロールで以下を実 行するためのアクセス許可を提供します。

- AWS Lambda 関数とエイリアスの読み取り、更新、呼び出しを行います。
- Amazon S3 バケットのリビジョンファイルにアクセスします。
- CloudWatch アラームに関する情報を取得します。
- Amazon SNS トピックに情報を公開します。

AWS Lambda デプロイでは、Amazon SNS へのアクセスを制限する場合は、 AWSCodeDeployRoleForLambdaLimitedポリシーをアタッチします。これは、サービスロールで 以下を実行するためのアクセス許可を提供します。

- AWS Lambda 関数とエイリアスの読み取り、更新、呼び出しを行います。
- Amazon S3 バケットのリビジョンファイルにアクセスします。
- CloudWatch アラームに関する情報を取得します。

また、サービスロールの設定の一環として、アクセス許可を付与するエンドポイントを指定するため に信頼関係を更新します。

サービスロールは、IAM コンソール、、 AWS CLIまたは IAM APIs を使用して作成できます。

トピック

- サービスロールの作成 (コンソール)
- ・ サービスロールの作成 (CLI)
- ・ <u>サービスロール ARN の取得 (コンソール)</u>
- ・ <u>サービスロール ARN の取得 (CLI)</u>

サービスロールの作成 (コンソール)

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/iam/</u>:// www.com」で IAM コンソールを開きます。
- 2. ナビゲーションペインで ロール を選択してから、ロールを作成する を選択します。
- 3. AWS サービスを選択し、「ユースケース」のドロップダウンリストから [CodeDeploy] を選択 します。
- 4. ユースケースを選択します。
 - ・ EC2/オンプレミスのデプロイ CodeDeploy を選択

- ・ AWS Lambda デプロイの場合は、CodeDeploy for Lambda を選択します。
- ・ Amazon ECS デプロイの場合は、CodeDeploy-ECS を選択
- 5. [Next (次へ)] を選択します。
- 「アクセス許可を追加」ページに、そのユースケースに適したアクセス許可ポリシーが表示され ます。[Next (次へ)]を選択します。
- [名前、確認、および作成] ページで、[ロール名] にサービスロールの名前 (例えば、CodeDeployServiceRole) を入力し、[ロールを作成] を選択します。

このサービスロールの説明を、[Role description] ボックスに入力することもできます。

 現在サポートされているすべてのエンドポイントへのアクセス許可をサービスロールに付与する 場合は、この手順を終了します。

このサービスロールから一部のエンドポイントへのアクセスを制限するには、この手順の残りの ステップに進みます。

- 9. ロールの一覧で、作成したロール (CodeDeployServiceRole) を検索し、選択します。
- 10. [信頼関係] タブを選択します。
- 11. [信頼ポリシーを編集]を選択します。

次のポリシーでは、サポートされているすべてのエンドポイントにアクセスする権限をサービス ロールに付与します。

サポートされているエンドポイントの一部にのみアクセスする権限をサービスロールに付与する には、信頼ポリシーテキストボックスの内容を以下のポリシーに置き換えます。アクセスを除外 するエンドポイントの行を削除し、[ポリシーの更新] を選択します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ٦
```
```
},
    "Action": "sts:AssumeRole"
    }
]
}
```

サービスロールの作成の詳細については、IAM ユーザーガイドの<u>「AWS サービスにアクセス許可を</u> <u>委任するロールの作成</u>」を参照してください。

サービスロールの作成 (CLI)

 開発マシンで、たとえば、CodeDeployDemo-Trust.json という名前のテキストファイルを 作成します。このファイルは、CodeDeploy がユーザーの代理操作の実行を許可するのに使用さ れます。

次のいずれかを行います:

 サポートされているすべての AWS リージョンへのアクセスを許可するには、次のコンテンツ を ファイルに保存してください。

サポートされているリージョンの一部にのみアクセスする権限を付与するには、ファイルに次の内容を入力し、アクセスを除外するリージョンの行を削除します。

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

{

```
"Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Note

リストにある最後のエンドポイントの後にコンマを使用しないでください。

 同じディレクトリから、create-role コマンドを呼び出し、先ほど作成したテキストファイルの 情報に基づく CodeDeployServiceRole という名前のサービスロールを作成します。

aws iam create-role --role-name CodeDeployServiceRole --assume-role-policy-document
file://CodeDeployDemo-Trust.json

A Important

ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

コマンドの出力で、Arn オブジェクトの下にある Role エントリの値を書きとめておきます。 これは、後でデプロイグループを作成する際に必要になります。値を忘れた場合は、<u>サービス</u> ロール ARN の取得 (CLI) の手順に従います。

- 3. 使用する管理ポリシーは、コンピューティングプラットフォームによって異なります。
 - EC2/オンプレミスコンピューティングプラットフォームにデプロイする場合は、以下を行います。

attach-role-policy コマンドを呼び出し、**CodeDeployServiceRole** という名前のサービス ロールに対して、**AWSCodeDeployRole** という名前の IAM マネージドポリシーに基づくアク セス許可を付与します。以下に例を示します。

aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn arn:aws:iam::aws:policy/service-role/AWSCodeDeployRole

・ デプロイ先が AWS Lambda コンピューティングプラットフォームの場合:

attach-role-policy コマンドを呼び出し、CodeDeployServiceRole という 名前のサービスロールに対して、AWSCodeDeployRoleForLambda または AWSCodeDeployRoleForLambdaLimited という名前の IAM マネージドポリシーに基づく アクセス許可を付与します。例: aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn arn:aws:iam::aws:policy/service-role/AWSCodeDeployRoleForLambda

• Amazon ECS コンピューティングプラットフォームへのデプロイの場合。

attach-role-policy コマンドを呼び出し、CodeDeployServiceRole とい う名前のサービスロールに対して、AWSCodeDeployRoleForECS または AWSCodeDeployRoleForECSLimited という名前の IAM マネージドポリシーに基づくアク セス許可を付与します。例:

aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn arn:aws:iam::aws:policy/AWSCodeDeployRoleForECS

サービスロールの作成の詳細については、IAM <u>ユーザーガイドの「 AWS サービスのロールの作成</u>」 を参照してください。

サービスロール ARN の取得 (コンソール)

IAM コンソールを使用してサービスロールの ARN を取得するには:

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/iam/</u>:// www.com」で IAM コンソールを開きます。
- 2. ナビゲーションペインで Roles (ロール) を選択します。
- [フィルタ] テキストボックスに、「CodeDeployServiceRole」と入力して、Enter キーを押します。
- 4. [CodeDeployServiceRole]を選択します。
- 5. [ロールの ARN] フィールドの値を書き留めます。

サービスロール ARN の取得 (CLI)

を使用してサービスロールの ARN AWS CLI を取得するには、 という名前のサービスロールに対し て get-role コマンドを呼び出しますCodeDeployServiceRole。

aws iam get-role --role-name CodeDeployServiceRole --query "Role.Arn" --output text

返される値はサービスロールの ARN です。

ステップ 3: CodeDeploy ユーザーのアクセス許可を制限する

セキュリティ上の理由から、<u>ステップ 1: セットアップ</u> で作成した管理ユーザーのアクセス許可 を、CodeDeploy でのデプロイの作成と管理に必要なアクセス許可のみに制限することをお勧めしま す。

以下の一連の手順を使用して、CodeDeploy 管理ユーザーのアクセス許可を制限します。

[開始する前に]

 <u>ステップ 1: セットアップ</u>の手順に従って IAM Identity Center で CodeDeploy 管理ユーザーを作 成したことを確認します。

アクセス権限セットを作成するには

このアクセス許可セットは後で CodeDeploy 管理ユーザーに割り当てます。

- 1. にサインイン AWS Management Console し、 AWS IAM Identity Center コンソールを <u>https://</u> console.aws.amazon.com/singlesignon/://www.com で開きます。
- ナビゲーションペインで [アクセス許可セット] を選択し、[アクセス許可セットの作成] を選択し ます。
- 3. [カスタム許可セット]を選択します。
- 4. [Next (次へ)] を選択します。
- 5. [インラインポリシー]を選択します。
- 6. サンプルコードを削除します。
- 7. 以下のポリシーを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeDeployAccessPolicy",
            "Effect": "Allow",
            "Action": [
            "autoscaling:*",
            "codedeploy:*",
            "ec2:*",
            "lambda:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "ecs:*",
            "Statement": [
            "Sid": "2012-10-17",
            "Statement": [
            "Statement": [
            "Statement": [
            "autoscaling:*",
            "ecs:*",
            "lambda:*",
            "ecs:*",
            "ecs:*",
```

```
"elasticloadbalancing:*",
        "iam:AddRoleToInstanceProfile",
        "iam:AttachRolePolicy",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
        "s3:*",
        "ssm:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeDeployRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID:role/CodeDeployServiceRole"
    }
  ]
}
```

このポリシーで、arn:aws:iam::account-ID:role/CodeDeployServiceRole を <u>ステッ</u> <u>プ 2: CodeDeployのサービスのロールを作成する</u> で作成した CodeDeploy サービスロールの ARN 値を置換します。ARN 値は、IAM コンソールのサービスロールの詳細ページにあります。

前述のポリシーにより AWS Lambda コンピューティングプラットフォーム、EC2/オンプレミ スコンピューティングプラットフォーム、および Amazon ECS コンピューティングプラット フォームにアプリケーションをデプロイできます。

このドキュメントで提供されている AWS CloudFormation テンプレートを使用し て、CodeDeploy と互換性のある Amazon EC2 インスタンスを起動できます。 AWS CloudFormation テンプレートを使用してアプリケーション、デプロイグループ、またはデ プロイ設定を作成するには、次のように CodeDeploy 管理ユーザーのアクセス許可ポリシー に cloudformation:* アクセス許可を追加することで、 AWS CloudFormationおよび AWS CloudFormation が依存する AWS サービスとアクションへのアクセスを提供する必要がありま す。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                ...
            "cloudformation:*"
        ],
        "Resource": "*"
        }
    ]
}
```

- 8. [Next (次へ)] を選択します。
- 9. 「アクセス許可セット名」に、次のように入力します。

CodeDeployUserPermissionSet

- 10. [Next (次へ)] を選択します。
- 11. [確認と作成] ページで情報を確認し、[グループの作成] を選択します。

許可セットを CodeDeploy 管理ユーザーに割り当てるには

- ナビゲーションペインで を選択しAWS アカウント、現在サインイン AWS アカウント しているの横にあるチェックボックスをオンにします。
- 2. [ユーザーまたはグループの割り当て] ボタンを選択します。
- 3. [ユーザー] タブを選択します。
- 4. CodeDeploy 管理ユーザーの横にあるチェックボックスをオンにします。
- 5. [Next (次へ)] を選択します。
- 6. [CodeDeployUserPermissionSet] のチェックボックスをオンにします。
- 7. [Next (次へ)] を選択します。

8. 情報を確認し、[送信] を選択します。

これで、CodeDeploy 管理ユーザー と を CodeDeployUserPermissionSetに割り当て AWS アカウント、それらをバインドしました。

CodeDeploy 管理ユーザーとしてサインアウトしてサインインし直すには

 サインアウトする前に、CodeDeploy 管理ユーザーの AWS アクセスポータル URL とユーザー 名、ワンタイムパスワードがあることを確認してください。

Note

この情報がない場合は、IAM Identity Center の CodeDeploy 管理ユーザー詳細ページに 移動し、[パスワードをリセットする]、[ワンタイムパスワードの生成 [...]] を選択して、 もう一度 [パスワードをリセットする] と、画面に情報が表示されます。

- 2. からサインアウトします AWS。
- 3. AWS アクセスポータル URL をブラウザのアドレスバーに貼り付けます。
- 4. CodeDeploy 管理ユーザーとしてサインインする。

画面に AWS アカウント ボックスが表示されます。

- 5. AWS アカウント を選択し、CodeDeploy 管理ユーザーと許可セットを割り当てた AWS アカウ ント の名前を選択します。
- 6. CodeDeployUserPermissionSetの横にある[管理コンソール]を選択します。

AWS Management Console が表示されます。これで、制限付きのアクセス許可を持つ CodeDeploy 管理ユーザーとしてサインインしました。このユーザーとして CodeDeploy 関連の 操作のみを実行できるようになりました。

ステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロ ファイルを作成する

Note

Amazon ECS または AWS Lambda コンピューティングプラットフォーム を使用している場 合は、このステップをスキップします。

ステップ 4: IAM インスタンスプロファイルを作成する

Amazon EC2 インスタンスには、アプリケーションが保存される Amazon S3 バケットまたは GitHub レポジトリへのアクセス許可が必要です。CodeDeploy と互換性のある Amazon EC2 インス タンスを起動するには、追加の IAM ロールである インスタンスプロファイル を作成する必要があり ます。以下の手順では、Amazon EC2 インスタンスにアタッチする IAM インスタンスプロファイル を作成する方法を示します。このロールでは、アプリケーションが保存される Amazon S3 バケット または GitHub リポジトリへのアクセス許可が CodeDeploy エージェントに付与されます。

IAM インスタンスプロファイルは AWS CLI、、IAM コンソール、または IAM APIs を使用して作成 できます。

Note

IAM インスタンスプロファイルは、起動時の Amazon EC2 インスタンスまたは以前に起動し たインスタンスにアタッチできます。詳細については、「<u>インスタンスプロファイル</u>」を参 照してください。

トピック

- Amazon EC2 インスタンス(CLI)の IAM インスタンスプロファイルを作成する
- Amazon EC2 インスタンス(コンソール)の IAM インスタンスプロファイルを作成する

Amazon EC2 インスタンス(CLI)の IAM インスタンスプロファイルを作成する

以下のステップでは、「<u>CodeDeploy の開始方法</u>」にある最初の 3 つの手順の指示に従っていること を前提としています。

 開発マシンで、CodeDeployDemo-EC2-Trust.json という名前のテキストファイルを作成し ます。Amazon EC2 によるユーザーの代理操作の実行を許可するには、次の内容を貼り付けま す。

}

```
"Service": "ec2.amazonaws.com"
},
"Action": "sts:AssumeRole"
}
]
```

 同じディレクトリで、CodeDeployDemo-EC2-Permissions.json という名前のテキスト ファイルを作成します。以下の内容を貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
               "s3:Get*",
               "s3:List*"
        ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Note

このポリシーを、Amazon EC2 インスタンスがアクセスする必要のある Amazon S3 バ ケットにのみ制限することをお勧めします。CodeDeploy エージェントを含む Amazon S3 バケットへのアクセスを必ず許可してください。そうしない場合、CodeDeploy エー ジェントがインスタンス上にインストールされる、または更新されるときに、エラーが 発生する可能性があります。Amazon S3 中の CodeDeploy リソースキットバケットのみ への IAM インスタンスアクセスは許すが、アクセスを防止するバケットのための行を削 除するには

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
            "s3:Get*",
            "s3:Get*",
            "Statement": [
            "Statement": [
            "s3:Get*",
            "Statement": [
            "Statement: [
```



Note

CodeDeploy で <u>IAM 認可</u>または Amazon Virtual Private Cloud (VPC) エンドポイントを 使用する場合は、アクセス許可を追加する必要があります。詳細については、[<u>Amazon</u> Virtual Private Cloud で CodeDeploy を使う] を参照してください。

3. 同じディレクトリから、create-role コマンドを呼び出して、最初のファイルの情報に基づいて CodeDeployDemo-EC2-Instance-Profile という名前の IAM ロールを作成します。

A Important

ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

aws iam create-role --role-name CodeDeployDemo-EC2-Instance-Profile --assume-rolepolicy-document file://CodeDeployDemo-EC2-Trust.json

同じディレクトリから、put-role-policy コマンドを呼び出して、2 番目のファイルの情報に基づいて CodeDeployDemo-EC2-Instance-Profile という名前のロールアクセス許可を付与します。

A Important

ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

aws iam put-role-policy --role-name CodeDeployDemo-EC2-Instance-Profile --policyname CodeDeployDemo-EC2-Permissions --policy-document file://CodeDeployDemo-EC2-Permissions.json

5. attach-role-policy を呼び出して、SSM が CodeDeploy エージェントをインストールできるよう に、ロールに Amazon EC2 Systems Manager とアクセス許可を付与します。コマンドラインを 使用してパブリック Amazon S3 バケットからエージェントをインストールする場合、このポリ シーは必要ありません。「<u>CodeDeploy エージェントのインストール</u>」の詳細を確認してくださ い。 aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ AmazonSSMManagedInstanceCore --role-name CodeDeployDemo-EC2-Instance-Profile

6. create-instance-profile コマンドに続いて add-role-to-instance-profile コマンドを呼び出し

て、CodeDeployDemo-EC2-Instance-Profile という名前の IAM インスタンスプロファイ ルを作成します。インスタンスプロファイルにより、Amazon EC2 は最初に起動されたときに CodeDeployDemo-EC2-Instance-Profile という名前の IAM ロールを Amazon EC2 イン スタンスに渡します。

```
aws iam create-instance-profile --instance-profile-name CodeDeployDemo-EC2-
Instance-Profile
aws iam add-role-to-instance-profile --instance-profile-name CodeDeployDemo-EC2-
Instance-Profile --role-name CodeDeployDemo-EC2-Instance-Profile
```

IAM インスタンスプロファイルの名前を取得する必要がある場合は、<u>AWS CLI リファレンス</u>の IAM セクションで「list-instance-profiles-for-role」を参照してください。

これで、IAM インスタンスにアタッチする Amazon EC2 インスタンスプロファイルを作成しまし た。詳細については、<u>Amazon EC2 ユーザーガイド</u> の「Amazon EC2 の IAM ロール」を参照してく ださい。

Amazon EC2 インスタンス(コンソール)の IAM インスタンスプロファイル を作成する

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/iam/</u>:// www.com」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、[Policies]、[Create policy] の順に選択します。
- 3. [アクセス許可の指定]ページで、[JSON]を選択します。
- 4. JSON サンプルコードを削除します。
- 5. 次のコードを貼り付けます。

}

```
"s3:List*"
],
"Effect": "Allow",
"Resource": "*"
}
]
```

Note

このポリシーを、Amazon EC2 インスタンスがアクセスする必要のある Amazon S3 バ ケットにのみ制限することをお勧めします。CodeDeploy エージェントを含む Amazon S3 バケットへのアクセスを必ず許可してください。そうしない場合、CodeDeploy エー ジェントがインスタンス上にインストールされる、または更新されるときに、エラーが 発生する可能性があります。Amazon S3 中の CodeDeploy リソースキットバケットのみ への IAM インスタンスアクセスは許すが、アクセスを防止するバケットのための行を削 除するには

4
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"s3:Get*",
"s3:List*"
],
"Resource": [
"arn:aws:s3:::amzn-s3-demo-bucket/*",
"arn:aws:s3:::aws-codedeploy-us-east-2/*",
"arn:aws:s3:::aws-codedeploy-us-east-1/*",
"arn:aws:s3:::aws-codedeploy-us-west-1/*",
"arn:aws:s3:::aws-codedeploy-us-west-2/*",
"arn:aws:s3:::aws-codedeploy-ca-central-1/*",
"arn:aws:s3:::aws-codedeploy-eu-west-1/*",
"arn:aws:s3:::aws-codedeploy-eu-west-2/*",
"arn:aws:s3:::aws-codedeploy-eu-west-3/*",
"arn:aws:s3:::aws-codedeploy-eu-central-1/*",
"arn:aws:s3:::aws-codedeploy-eu-central-2/*",
"arn:aws:s3:::aws-codedeploy-eu-north-1/*",
"arn:aws:s3:::aws-codedeploy-eu-south-1/*",
"arn:aws:s3:::aws-codedeploy-eu-south-2/*",



Note

CodeDeploy で <u>IAM 認可</u>または Amazon Virtual Private Cloud (VPC) エンドポイントを 使用する場合は、アクセス許可を追加する必要があります。詳細については、[<u>Amazon</u> Virtual Private Cloud で CodeDeploy を使う] を参照してください。

- 6. [Next (次へ)] を選択します。
- [確認および作成] ページで、[ポリシー名] ボックスに「CodeDeployDemo-EC2-Permissions」と入力します。
- 8. (オプション) [説明] に、ポリシーの説明を入力します。
- 9. [Create policy] を選択します。
- 10. ナビゲーションペインで [Roles] を選択し、続いて [Create role] を選択します。
- 11. [ユースケース] で、[EC2] ユースケースを選択します。
- 12. [Next (次へ)] を選択します。
- ポリシーのリストで、作成したポリシー (CodeDeployDemo-EC2-Permissions) の横にある
 チェックボックスをオンにします。必要に応じて、検索ボックスを使用してポリシーを見つけます。

- 14. Systems Manager を使用して CodeDeploy エージェントをインストールまたは設定するには、 [AmazonSSMManagedInstanceCore] の横にあるボックスを選択します。この AWS 管理ポリ シーにより、インスタンスは Systems Manager サービスコア機能を使用できます。必要に応 じて、検索ボックスを使用してポリシーを見つけます。コマンドラインを使用してパブリック Amazon S3 バケットからエージェントをインストールする場合、このポリシーは必要ありませ ん。「CodeDeploy エージェントのインストール」の詳細を確認してください。
- 15. [Next (次へ)] を選択します。
- 16. [名前、確認、および作成] ページで、[ロール名] にサービスロールの名前 (例え ば、CodeDeployDemo-EC2-Instance-Profile) を入力し、[ロールを作成] を選択します。

このサービスロールの説明を、[Role description] ボックスに入力することもできます。

これで、IAM インスタンスにアタッチする Amazon EC2 インスタンスプロファイルを作成しました。詳細については、<u>Amazon EC2 ユーザーガイド</u> の「Amazon EC2 の IAM ロール」を参照してく ださい。

CodeDeploy との製品とサービスの統合

デフォルトでは、CodeDeploy は多数の AWS サービスおよびパートナー製品およびサービスと統合 されます。以下の情報は、使用する製品やサービスを統合するための CodeDeploy の設定に役立ち ます。

- 他の AWS サービスとの統合
- パートナーの製品とサービスとの統合
- <u>コミュニティから統合の例</u>

他の AWS サービスとの統合

CodeDeploy は、次の AWS サービスと統合されています。

Amazon CloudWatch

Amazon CloudWatch は、 AWS クラウドリ ソースと、 AWSで実行するアプリケーショ ンのモニタリングサービスです。Amazon CloudWatch を使用してメトリクスの収集と追 跡、ログファイルの収集とモニタリング、アラ ームの設定ができます。CodeDeploy は、次の CloudWatch ツールをサポートしています。

 CloudWatch alarms は、デプロイをモニタリ ングし、指定したモニタリングメトリクスが CloudWatch アラームルールで指定したしき い値を超えるか下回った場合にデプロイを停 止します。アラームモニタリングを使用する には、まず CloudWatch でアラームを設定し ます。次に、CodeDeploy でアラームをアプ リケーションまたはデプロイグループに追加 し、アラームがアクティブになったときに停 止させます。

詳細はこちら:

• CloudWatch Logs アラームの作成

 Amazon CloudWatch Events では、CodeDe ploy オペレーションでのインスタンスやデプ ロイの状態の変更を検出し、対応できます。 次に、作成したルールに基づいて CloudWatc h Events はデプロイまたはインスタンスが、 ルールで指定した状態になると、1 つ以上の ターゲットアクションを呼び出します。

詳細はこちら:

- <u>Amazon CloudWatch Events を使用したデ</u> プロイのモニタリング
- Amazon CloudWatch Logs では、インスタンスに個別にサインインせずに CodeDeploy エージェントによって作成された3種類のログをモニタリングします。

詳細はこちら:

 <u>CloudWatch Logs コンソールで</u> CodeDeploy ログを表示する

Amazon EC2 Auto Scaling

CodeDeploy は、<u>Amazon EC2 Auto Scaling</u> (Amazon EC2 オートスケーリング) をサポート します。この AWS サービスは、指定した条件 に基づいて Amazon EC2 インスタンスを自動 的に起動できます。次に例を示します。

- 指定された CPU 使用率の制限を超えた。
- ディスクの読み書き。
- 指定された期間のインバウンドまたはアウト バウンドのネットワークトラフィック。

必要に応じて Amazon EC2 インスタンスグ ループをスケールアウトできます。さらに CodeDeploy を使用してアプリケーションリビ ジョンを自動的にデプロイできます。Amazon EC2 Auto Scaling は、Amazon EC2 インスタ ンスが不要になったときに、それを終了しま す。

詳細はこちら:

- <u>CodeDeploy</u>と Amazon EC2 Auto Scaling の 統合
- <u>チュートリアル: CodeDeploy を使用し</u>
 <u>て、Auto Scaling グループにアプリケーショ</u>
 ンをデプロイします。
- ・ <u>内部構造: CodeDeploy と Auto Scaling の統</u> 合

Amazon Elastic Container Service

CodeDeploy を使用して、Amazon ECSにコン テナ化されたアプリケーションをタスクセッ トとしてデプロイできます。CodeDeploy は、 アプリケーションの更新バージョンを新しい 置き換えタスクセットとしてインストールす ることで、Blue/Green のデプロイを実行しま す。CodeDeploy は、元のアプリケーションタ スクセットからの本稼働トラフィックを置き換 えタスクセットに再ルーティングします。デプ ロイが正常に完了すると、元のタスクセットは 削除されます。Amazon ECS の詳細について は、「<u>Amazon Elastic Container Service</u>」を参 照してください。

Canary 設定、線形設定、または一度にすべて の設定を選択して、デプロイ時に更新済みタス クセットにトラフィックを移行する方法を管理 できます。Amazon ECS デプロイの詳細につ いては、「<u>Amazon ECS コンピューティング</u> <u>プラットフォームでのデプロイ</u>」を参照してく ださい。

AWS CloudTrail

CodeDeploy は <u>AWS CloudTrail</u> と統合しま す。このサービスでは、 AWS アカウントで CodeDeploy によって、または CodeDeploy の 代わりに行われた API コールをキャプチャし、 指定した Amazon S3 バケットにログファイル を配信します。CloudTrail は、CodeDeploy コ ンソール、 AWS CLIを通した CodeDeploy コ マンド、または CodeDeploy API からの API 呼 び出しを直接キャプチャします。CloudTrail に よって収集されたデータを使用して、以下の情 報を判断できます。

- CodeDeploy に行われたリクエスト。
- ・ リクエストが行われたソース IP アドレス。
- 誰がリクエストを行ったか。
- 行われた時刻。

詳細はこちら:

Monitoring Deployments

AWS Cloud9

AWS Cloud9 は、インターネットに接続された マシンからブラウザのみを使用してコードを記 述、実行、デバッグ、デプロイするために使用 できるクラウドベースのオンライン統合開発 環境 (IDE) です。 AWS Cloud9 には、コード エディタ、デバッガー、ターミナル、および AWS CLI や Git などの重要なツールが含まれ ています。

- IDE を使用して、GitHub AWS Cloud9 リポ ジトリにあるコードを実行、デバッグ、構 築できます。IDE Environment ウィンドウお よびエディタのタブを使用して、コードの 表示、変更、保存を行うことができます。
 準備ができたら、AWS Cloud9 ターミナル セッションで Git を使用してコード変更を GitHub リポジトリにプッシュし、を使用し て更新 AWS CodeDeploy をデプロイできま す。GitHub AWS Cloud9 でを使用する方法 の詳細については、「<u>の GitHub サンプル</u> AWS Cloud9」を参照してください。
- IDE AWS Cloud9 を使用して AWS Lambda 関数を更新できます。その後、 を使用して AWS CodeDeploy 、 AWS Lambda 関数の 新しいバージョンにトラフィックを移行す るデプロイを作成できます。詳細について は、<u>AWS Cloud9</u>「統合開発環境 (IDE) での <u>AWS Lambda 関数の使用</u>」を参照してくだ さい。

詳細については AWS Cloud9、<u>「とは AWS</u> <u>Cloud9</u>」および<u>「の開始方法 AWS Cloud9</u>」 を参照してください。

AWS CodePipeline

AWS CodePipeline は、継続的な配信プロセ スでソフトウェアをリリースするために必要 な手順のモデル化、可視化、および自動化に 使用できる継続的な配信サービスです。AWS CodePipeline を使用して、コード変更のたび にサービスが構築、テスト、デプロイを行う よう独自のプロセスを定義できます。例えば、 ベータ、ガンマ、本番の3つのアプリケーショ ンのデプロイグループがあるとします。ソース コードが変更されるたびに、各デプロイグルー プに1つずつ更新をデプロイするようにパイプ ラインを設定できます。

CodeDeploy を使用してデプロイ AWS CodePipeline するように を設定できます。

- Amazon EC2 インスタンス、オンプレミスイ ンスタンス、または両方へのコード。
- Serverless AWS Lambda 関数のバージョン。

パイプラインを作成する前または [パイプライ ンの作成] ウィザードの段階のデプロイアク ションで使用できるように、CodeDeploy アプ リケーション、デプロイおよびデプロイグルー プを作成できます。

詳細はこちら:

- <u>DevOps 用AWS 入門ガイド</u> CodePipeline を CodeDeploy で使用して、CodeCommit リ ポジトリのソースコードを Amazon EC2 イ ンスタンスに継続的に配信およびデプロイす る方法について説明します。
- シンプルなパイプラインのチュートリアル (Amazon S3 バケット)

	 シンプルなパイプラインのチュートリアル (CodeCommit リポジトリ) 4 段階パイプラインのチュートリアル
AWS サーバーレスアプリケーションモデル	AWS サーバーレスアプリケーションモデル (AWS SAM) は、サーバーレスアプリケー ションを定義するモデルです。を拡張 AWS CloudFormation して、サーバーレスアプ リケーションに必要な関数、Amazon API Gateway APIs、Amazon DynamoDB テーブル を簡単に AWS Lambda 定義できます。AWS SAM を既に使用している場合は、デプロイ設 定を追加して CodeDeploy の使用を開始し、 AWS Lambda アプリケーションのデプロイ中 にトラフィックを移行する方法を管理できま す。
エラスティックロードバランシング	CodeDeploy は、 <u>Elastic Load Balancing</u> をサ ポートしています。このサービスでは、着信ア プリケーションを複数の Amazon EC2 インス タンス間に分散します。 また、CodeDeploy のデプロイ中、インスタン スが準備完了ではない、現在デプロイ中、また は環境の一部として不要になった場合は、ロー ドバランサーはそのインスタンスへのインター ネットトラフィックのルーティングを防止しま
	す。 詳細はこちら: • <u>Integrating CodeDeploy with Elastic Load</u> <u>Balancing</u>

トピック

- ・ CodeDeploy と Amazon EC2 Auto Scaling の統合
- CodeDeploy と Elastic Load Balancing の統合

CodeDeploy と Amazon EC2 Auto Scaling の統合

CodeDeploy は、定義した条件に従って Amazon EC2 インスタンスを自動的に起動する AWS サー ビスである Amazon EC2 Auto Scaling をサポートしています。これらの条件には、指定された時間 間隔にわたる CPU 使用率やディスクの読み取りまたは書き込み、インバウンド/アウトバウンドの ネットワークトラフィックの制限の超過が含まれます。Amazon EC2 Auto Scaling は、インスタン スが不要になったときに、それを終了します。詳細については、「<u>Amazon EC2 Auto Scaling ユー</u> <u>ザーガイド</u>」の「Amazon EC2 Auto Scaling とは」を参照してください。

Amazon EC2 Auto Scaling グループの一部として新しい Amazon EC2 インスタンスが起動される と、CodeDeploy ではリビジョンを新しいインスタンスに自動的にデプロイできます。また、Elastic Load Balancing ロードバランサーに登録された Amazon EC2 Auto Scaling インスタンスを使用 して、CodeDeploy でのデプロイを調整できます。詳細については、<u>Integrating CodeDeploy with</u> <u>Elastic Load Balancing</u>および<u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロー</u> <u>ドバランサーをセットアップする</u>を参照してください。

1 Note

複数のデプロイグループを1つの Amazon EC2 Auto Scaling グループに関連付けると、問題が発生する可能性があります。例えば、1つのデプロイが失敗すると、インスタンスはシャットダウンを開始しますが、実行中の他のデプロイはタイムアウトに1時間かかる可能性があります。詳細については、「複数のデプロイグループを1つの Amazon EC2 Auto Scaling グループに関連付けることは避ける」そして「内部構造: CodeDeploy と Amazon EC2 Auto Scaling の統合」を参照してください。

トピック

- <u>CodeDeploy アプリケーションを Amazon EC2 Auto Scaling グループにデプロイする</u>
- Auto Scaling スケールインイベント中の終了デプロイの有効化
- ・ Amazon EC2 Auto Scaling と CodeDeploy の連携
- ・ CodeDeploy と Amazon EC2 Auto Scaling でのカスタム AMI の使用

CodeDeploy アプリケーションを Amazon EC2 Auto Scaling グループにデプロイする

Amazon EC2 Auto Scaling グループに CodeDeploy アプリケーションリビジョンをデプロイするに は、次の手順を実行します。

 Amazon EC2 Auto Scaling グループが Amazon S3 での作業を許可する IAM インスタンスプロ ファイルを作成または検索します。詳細については、「<u>ステップ 4: Amazon EC2 インスタンス</u> 用の IAM インスタンスプロファイルを作成する」を参照してください。

Note

また、CodeDeploy を使用して GitHub リポジトリから Amazon EC2 Auto Scaling グ ループにリビジョンをデプロイすることもできます。Amazon EC2 インスタンスで依然 として IAM インスタンスプロファイルが必要な場合でも、GitHub リポジトリからデプ ロイするときは、プロファイルに追加のアクセス許可は不要です。

- Amazon EC2 Auto Scaling グループを作成または使用し、起動設定またはテンプレートで、IAM インスタンスプロファイルを指定します。詳細については、「<u>Amazon EC2 インスタンスで実</u> 行されるアプリケーションに対する IAM ロール」を参照してください。
- 3. Amazon EC2 Auto Scaling グループを含むデプロイグループの作成を CodeDeploy に許可する サービスロールを作成または検索します。
- CodeDeploy でデプロイグループを作成し、Amazon EC2 Auto Scaling グループ名、サービス ロール、その他いくつかのオプションを指定します。詳細については、<u>インプレースデプロイ用</u> <u>のデプロイグループを作成する (コンソール)</u> または <u>インプレースデプロイ用のデプロイグルー</u> <u>プを作成する (コンソール)</u> を参照してください。
- 5. CodeDeploy を使用して、Amazon EC2 Auto Scaling グループを含むデプロイグループにリビ ジョンをデプロイします。

詳細については、「<u>チュートリアル: CodeDeploy を使用して、Auto Scaling グループにアプリケー</u> ションをデプロイします。」を参照してください。

Auto Scaling スケールインイベント中の終了デプロイの有効化

終了デプロイは、CodeDeploy デプロイの一種で、Auto Scaling <u>スケールインイベント</u>が発生する と、自動的にアクティブ化されます。CodeDeploy は、Auto Scaling サービスがインスタンスを終了 する直前に終了デプロイを実行します。終了デプロイ中は、CodeDeploy は何もデプロイしません。 代わりに、ライフサイクルイベントを生成します。これを独自のスクリプトにフックしてカスタムの シャットダウン機能を有効にすることができます。例えば、ApplicationStop ライフサイクルイ ベントを、インスタンスの終了前にアプリケーションを正常にシャットダウンするスクリプトにフッ クできます。

CodeDeploy が終了デプロイ中に生成するライフサイクルイベントのリストについては、「<u>ライフサ</u> イクルイベントフックの可用性」を参照してください。

終了デプロイが何らかの理由で失敗した場合、CodeDeploy はインスタンスの終了を続行できるよう にします。つまり、CodeDeploy がライフサイクルイベントの完全なセット (または一部) を最後まで 実行しなかった場合でも、インスタンスはシャットダウンされます。

終了デプロイを有効にしない場合でも、スケールインイベントが発生すると、Auto Scaling サービス は Amazon EC2 インスタンスを終了しますが、CodeDeploy はライフサイクルイベントを生成しま せん。

Note

終了デプロイを有効にするかどうかにかかわらず、CodeDeploy のデプロイ中に Auto Scaling サービスが Amazon EC2 インスタンスを終了すると、Auto Scaling サービスと CodeDeploy サービスによって生成されたライフサイクルイベント間に競合状態が発生する 可能性があります。例えば、Auto Scaling サービスで生成した Terminating ライフサイク ルイベントは、CodeDeploy デプロイで生成した ApplicationStart イベントをオーバー ライドする場合があります。このシナリオでは、Amazon EC2 インスタンスの終了または CodeDeploy のデプロイのいずれかで障害が発生する可能性があります。

CodeDeploy で終了デプロイを実行できるようにするには

 デプロイグループを作成または更新するときに、[Auto Scaling グループに終了フックを追加]
 チェックボックスをオンにします。手順については、「<u>インプレースデプロイ用のデプロイグ</u> <u>ループを作成する (コンソール)</u>」または「<u>EC2/オンプレミス Blue/Green デプロイ用のデプロイ</u> <u>グループを作成する (コンソール)</u>」を参照してください。

このチェックボックスを有効にすると、CodeDeploy は、CodeDeploy デプロイグループの作成 時または更新時に指定した Auto Scaling グループに <u>Auto Scaling ライフサイクルフック</u>をイン ストールします。このフックは終了フックと呼ばれ、終了デプロイを有効にします。

終了フックをインストールすると、次のようにスケールイン(終了)イベントが展開されます。

- 1. Auto Scaling サービス (または単に Auto Scaling) は、スケールアウトイベントが発生する必要が あると判断すると、EC2 サービスに連絡して EC2 インスタンスを終了します。
- 2. EC2 サービスが EC2 インスタンスの終了を開始します。インスタンスは、Terminating 状態 へ、その後 Terminating:Wait 状態へと移行します。
- Terminating:Wait 中、Auto Scaling は、CodeDeploy によってインストールされた終了フック を含め、Auto Scaling グループにアタッチされたすべてのライフサイクルフックを実行します。
- 4. 終了フックは、CodeDeploy がポーリングする Amazon SQS キューに通知を送信します。
- 5. 通知を受信すると、CodeDeploy はメッセージを解析し、いくつかの検証を実行して、<u>終了デプロ</u> イを実行します。
- 終了デプロイの実行中、CodeDeploy は 5 分ごとに Auto Scaling にハートビートを送信し、イン スタンスがまだ処理中であることを知らせます。
- 7. これまでのところ、EC2 インスタンスは Terminating:Wait 状態 (<u>Auto Scaling グループの</u> ウォームプールを有効にしている場合は、Warmed:Pending:Wait 状態) のままです。
- 8. デプロイが完了すると、デプロイの成否にかかわらず、CodeDeploy は Auto Scaling に対して EC2 終了プロセスを CONTINUE するよう指示します。

Amazon EC2 Auto Scaling と CodeDeploy の連携

Auto Scaling グループを含むように CodeDeploy デプロイグループを作成または更新する

と、CodeDeploy は CodeDeploy サービスロールを使用して Auto Scaling グループにアクセス

し、Auto Scaling ライフサイクルフックを Auto Scaling グループにインストールします。

③ Note Auto Scaling ライフサイクルフックは、このガイドの「<u>AppSpec の「hooks」セクション</u>」 で説明している、CodeDeploy によって生成されるライフサイクルイベント (ライフサイクル イベントフックとも呼ばれます)とは異なります。

CodeDeploy がインストールする Auto Scaling ライフサイクルフックは次のとおりです。

 ・ 起動フック — このフックは、Auto Scaling <u>スケールアウトイベント</u>が進行中であり、起動デプロ イを開始する必要があることを CodeDeploy に通知します。

起動デプロイ中、CodeDeploy は、次のことを行います。

アプリケーションのリビジョンを、スケールアウトされたインスタンスにデプロイします。

デプロイの進行状況を示すライフサイクルイベントを生成します。これらのライフサイクルイベントを独自のスクリプトにフックして、カスタム起動機能を有効にすることができます。詳細については、「ライフサイクルイベントフックの可用性」の表を参照してください。

起動フックおよび関連する起動デプロイは常に有効になっており、無効にすることはできません。 ・ 終了フック — このオプションのフックは、Auto Scaling <u>スケールインイベント</u>が進行中であり、 終了デプロイを開始する必要があることを CodeDeploy に通知します。

終了デプロイ中、CodeDeploy はインスタンスのシャットダウンの進行状況を示すライフサイクル イベントを生成します。詳細については、「<u>Auto Scaling スケールインイベント中の終了デプロイ</u> の有効化」を参照してください。

トピック

- CodeDeploy でインストールしたライフサイクルフックは、どのように使用されますか?
- CodeDeploy による Amazon EC2 Auto Scaling グループの名前の設定方法
- ・ カスタムライフサイクルフックイベントの実行順序
- デプロイ中のスケールアップイベント
- デプロイ中のスケールインイベント
- cfn-init AWS CloudFormation スクリプトのイベントの順序

CodeDeploy でインストールしたライフサイクルフックは、どのように使用されますか?

起動ライフサイクルフックと終了ライフサイクルフックをインストールすると、CodeDeploy の Auto Scaling グループのスケールアウトイベントとスケールインイベントでそれぞれ使用されます。

スケールアウト (起動) イベントは次のように展開されます。

- 1. Auto Scaling サービス (または単に Auto Scaling) は、スケールアウトイベントが発生する必要が あると判断し、EC2 サービスに連絡して新しい EC2 インスタンスを起動します。
- EC2 サービスが、新しい EC2 インスタンスを起動します。インスタンスは、Pending 状態へ、 その後 Pending:Wait 状態へと移行します。
- Pending:Wait 中である場合、Auto Scaling は、Auto Scaling グループにアタッチされたすべて のライフサイクルフック (CodeDeploy によってインストールされた起動フックを含む) を実行し ます。
- 4. 起動フックは、CodeDeploy がポーリングする Amazon SQS キューに通知を送信します。

- 5. 通知を受信すると、CodeDeploy はメッセージを解析し、いくつかの検証を実行して、<u>起動デプロ</u> イを開始します。
- 6. 起動デプロイの実行中、CodeDeploy は 5 分ごとに Auto Scaling にハートビートを送信し、イン スタンスがまだ処理中であることを知らせます。
- 7. 今までのところ、EC2 インスタンスはなお Pending:Wait の状態です。
- 8. デプロイが完了すると、デプロイが成功したか失敗したかによって、CodeDeploy は Auto Scaling に EC2 起動プロセスを CONTINUE または ABANDON するよう指示を出します。
 - CodeDeploy が CONTINUE を指示する場合、Auto Scaling は他のフックの完了を待機するか、 インスタンスを Pending: Proceed 状態に、その後 InService 状態にします。
 - CodeDeploy が ABANDON を指示する場合、Auto Scaling は EC2 インスタンスを終了し、また 必要な数のインスタンスを満たす必要がある場合には、[Auto Scaling の 希望容量 の設定] で定 義されているように、起動手順を再開します。

スケールイン (終了) イベントは次のように展開されます。

「Auto Scaling スケールインイベント中の終了デプロイの有効化」を参照してください。

CodeDeploy による Amazon EC2 Auto Scaling グループの名前の設定方法

EC2/オンプレミスコンピューティングプラットフォームでの Blue/Green のデプロイ中に代替 (Green) 環境にインスタンスを追加するには 2 つの方法があります。

- 既存のインスタンス、または手動で作成したインスタンスを使用します。
- 指定した Amazon EC2 Auto Scaling グループの設定を使用して、新しい Amazon EC2 Auto Scaling グループにインスタンスを定義および作成します。

2 番目のオプションを選択した場合は、CodeDeploy によって Amazon EC2 Auto Scaling グループが プロビジョンされます。以下の規則を使用して、グループに名前を付けます。

CodeDeploy_deployment_group_name_deployment_id

例えば、ID が「10」のデプロイによって「alpha-deployments」というデプロイグ ループがデプロイされる場合、 プロビジョンされる Amazon EC2 Auto Scaling グループ の名前は CodeDeploy_alpha-deployments_10 になります。詳細については、<u>EC2/</u> <u>オンプレミス Blue/Green デプロイ用のデプロイグループを作成する (コンソール)</u> および

「<u>GreenFleetProvisioningOption</u>」を参照してください。

カスタムライフサイクルフックイベントの実行順序

CodeDeploy がデプロイする先の Amazon EC2 Auto Scaling グループに独自のライフサイクル フックを追加できます。ただし、カスタムライフサイクルフックイベントが実行される順序は、デ フォルトの CodeDeploy デプロイライフサイクルイベントに対して事前に決定できません。例え ば、Amazon EC2 Auto Scaling グループに ReadyForSoftwareInstall というカスタムライフサ イクルフックを追加すると、それが実行されるのが CodeDeploy のデフォルトデプロイライフサイ クルの最初のイベントの前であるか、最後のイベントの後であるか、事前に知ることはできません。

Amazon EC2 Auto Scaling グループにカスタムライフサイクルフックを追加する方法については、 [<u>Amazon EC2 Auto Scaling ユーザーガイド</u>] の「ライフサイクルフックの追加」を参照してくださ い。

デプロイ中のスケールアップイベント

デプロイ中に Auto Scaling スケールアウトイベントが発生すると、新しいインスタンスが更新され て、最新のアプリケーションリビジョンではなく、前回デプロイしたアプリケーションリビジョンが 反映されます。デプロイが成功すると、古いインスタンスと新しくスケールアウトされたインスタン スは異なるアプリケーションリビジョンを反映します。古いリビジョンを持つインスタンスを最新の 状態にするために、CodeDeploy は (最初のデプロイの直後に) 追加のデプロイを自動的に開始し、古 いインスタンスを更新します。このデフォルトの動作を変更して、古い EC2 インスタンスが古いリ ビジョンに残るようにする場合は、「<u>Automatic updates to outdated instances</u>」を参照してくださ い。

デプロイの実行中に Amazon EC2 Auto Scaling スケールアウトプロセスを中断する場合 は、CodeDeploy でのロードバランシングに使用される common_functions.sh スクリプトの設定 を通して行うことができます。HANDLE_PROCS=true の場合、デプロイ中は以下の Auto Scaling イ ベントが自動的に中断されます。

- AZRebalance
- AlarmNotification
- ScheduledActions
- ReplaceUnhealthy

A Important

この機能をサポートするのは、CodeDeployDefault.OneAtATime のデプロイ設定のみです。

HANDLE_PROCS=true を使用して Amazon EC2 Auto Scaling 使用時のデプロイ問題を避ける方法 の詳細については、GitHub の「<u>aws-codedeploy</u>」で「<u>Important notice about handling AutoScaling</u> processes」を参照してください。

デプロイ中のスケールインイベント

Auto Scaling グループで CodeDeploy デプロイの進行中に Auto Scaling グループがスケールインを 開始すると、終了プロセス (CodeDeploy 終了デプロイライフサイクルイベントを含む) と終了イン スタンスの他の CodeDeploy ライフサイクルイベントとの間で競合状態が発生する可能性がありま す。すべての CodeDeploy ライフサイクルイベントが完了する前にインスタンスを終了すると、こ の特定のインスタンスへのデプロイが失敗する場合があります。また、デプロイ設定で [正常なホス トの最小数] をどのように設定したかによって、CodeDeploy デプロイ全体が失敗する場合と失敗し ない場合があります。

cfn-init AWS CloudFormation スクリプトのイベントの順序

cfn-init (または cloud-init) を使用して新しくプロビジョニングした Linux ベースのインスタ ンスでスクリプトを実行する場合、インスタンスの開始後に発生するイベントの順序を厳密に制御し ないと、デプロイは失敗することがあります。

次の順序に従う必要があります。

- 1. 新しくプロビジョニングしたインスタンスが開始する。
- 2. すべての cfn-init ブートストラップスクリプトが最後まで実行する。
- 3. CodeDeploy エージェントが開始します。
- 4. 最新のアプリケーションリビジョンがインスタンスにデプロイされる。

イベントの順序を慎重に制御しないと、CodeDeploy エージェントはすべてのスクリプトの実行が終 了する前にデプロイを開始する可能性があります。

イベントの順序を制御するには、以下のベストプラクティスのいずれかを使用します。

- cfn-init スクリプトを通じて CodeDeploy エージェントをインストールし、他のすべてのスク リプトの後に配置します。
- CodeDeploy エージェントをカスタム AMI に含め、cfn-init スクリプトを使用して起動します。エージェントは、他のすべてのスクリプトの後に配置します。

cfn-init の使用方法については、<u>AWS CloudFormation ユーザーガイド</u> の「cfn-init」を参照して ください。

CodeDeploy と Amazon EC2 Auto Scaling でのカスタム AMI の使用

Amazon EC2 Auto Scaling グループで新しい Amazon EC2 インスタンスを起動するとき、基本 AMI を指定するには 2 つのオプションがあります。

- CodeDeploy エージェントがインストールされている基本カスタム AMI を 指定できます。エージェントが既にインストールされているため、このオプションはほかのオプションよりも迅速に新しい Amazon EC2 インスタンスを起動します。ただし、このオプションでは (特に CodeDeploy エージェントが古い場合)、Amazon EC2 インスタンスの初回デプロイが失敗する可能性が大きくなります。このオプションを選択した場合は、基本カスタム AMI の CodeDeploy エージェントを定期的に更新することをお勧めします。
- CodeDeploy エージェントがインストールされていない基本 AMI を指定できます。また、 Amazon EC2 Auto Scaling グループ内で新しいインスタンスが起動されるたびにエージェントを インストールすることもできます。このオプションでは、ほかのオプションよりも新しい Amazon EC2 インスタンスの起動が遅くなりますが、インスタンスの最初のデプロイが成功する可能性は 大きくなります。このオプションは、CodeDeploy エージェントの最新バージョンを使用します。

CodeDeploy と Elastic Load Balancing の統合

CodeDeploy のデプロイ中、インスタンスが準備完了ではない、現在デプロイ中、または環境の一部 として不要になった場合は、ロードバランサーはそのインスタンスへのインターネットトラフィック のルーティングを防止します。ただし、ロードバランサーの正確な役割は、Blue/Green デプロイで 使用されるかインプレースデプロイで使用されるかによって異なります。

Note

Elastic Load Balancing ロードバランサーの使用は Blue/Green デプロイでは必須、インプ レースデプロイでは任意です。

Elastic Load Balancing のタイプ

Elastic Load Balancing は、CodeDeploy デプロイで使用可能な、Classic Load Balancer、Application Load Balancer、Network Load Balancer の 3 つのタイプのロードバランサー を提供します。

Classic Load Balancer

ルーティングおよび負荷分散を、トランスポートレイヤー (TCP/SSL) またはアプリケーションレ イヤー (HTTP/HTTPS) のいずれかで行います。VPC がサポートされています。

Note

Classic Load Balancer は Amazon ECS デプロイではサポートされていません。

Application Load Balancer

ルーティングと負荷分散をアプリケーションレイヤー (HTTP/HTTPS) で行い、パスベースのルー ティングをサポートしています。Virtual Private Cloud (VPC) 内の EC2 の各インスタンスまたは コンテナインスタンスのポートにリクエストをルーティングできます。

Note

Application Load Balancer のターゲットグループには、EC2 インスタンスでのデプロイ の場合は instance、および Fargate デプロイの場合は IP のターゲットタイプがなけれ ばなりません。詳細については、「<u>ターゲットタイプ</u>」を参照してください。

Network Load Balancer

パケットのコンテンツからではなく、TCP パケットヘッダーから抽出されたアドレス情報に 基づいて、トランスポートレイヤー (TCP/UDP Layer-4) でルーティングと負荷分散を行いま す。Network Load Balancer は、ロードバランサーの有効期間中、トラフィックバーストを処理 し、クライアントの出典 IP を保持して、固定 IP を使用します。

Elastic Load Balancing ロードバランサーの詳細は、以下のトピックを参照してください。

- ・ Elastic Load Balancing とは?
- ・「Classic Load Balancer とは?」
- Application Load Balancer とは?
- Network Load Balancer とは?

Blue/Green デプロイ

Elastic Load Balancing ロードバランサーの背後でインスタンストラフィックを再ルーティングする ことは CodeDeploy Blue/Green デプロイの基本です。

Blue/Green デプロイの場合、ロードバランサーは、最新のアプリケーションリビジョンのデプロイ 先であるデプロイグループの新しいインスタンス (置き換え先環境) に対しては、指定したルールに 基づくトラフィックのルーティングを許可し、前回のアプリケーションリビジョンの実行元である古 いインスタンス (元の環境) からはトラフィックをブロックします。

置き換え先環境のインスタンスが1つ以上のロードバランサーに登録されると、置き換え元環境の インスタンスは登録解除され、終了可能になります。

ブルー/グリーンデプロイでは、デプロイグループで 1 つ以上の Classic Load Balancer、Application Load Balancer ターゲットグループ、または Network Load Balancer ターゲットグループを指定でき ます。CodeDeploy コンソールまたは を使用して AWS CLI、ロードバランサーをデプロイグループ に追加します。

Blue/Green デプロイにおけるロードバランサーの使用に関する詳細については、以下のトピックを 参照してください。

- <u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバランサーをセットアッ</u> プする
- Blue/Green デプロイ (コンソール) のアプリケーションを作成します。
- EC2/オンプレミス Blue/Green デプロイ用のデプロイグループを作成する (コンソール)

インプレースデプロイ

インプレースデプロイ中は、ロードバランサーにより、デプロイ先のインスタンスに対するインター ネットトラフィックのルーティングがブロックされ、そのインスタンスへのデプロイが完了した時点 でインスタンスに対するトラフィックのルーティングが再開されます。

インプレースデプロイ中にロードバランサーが使用されないと、インターネットトラフィックはデプ ロイプロセス中に依然としてインスタンスにルーティングされる場合があります。その結果、お客様 に表示されるウェブアプリケーションが破損していたり、不完全であったり、古いものであったりす る可能性があります。インプレースデプロイで Elastic Load Balancing ロードバランサーを使用する 場合、デプロイグループのインスタンスはロードバランサーから登録解除され、最新のアプリケー ションリビジョンに更新されてから、デプロイが成功した後で同じデプロイグループの一部として ロードバランサーに再登録されます。CodeDeploy は、ロードバランサーのバックグラウンドでイン スタンスが正常になるまで最大1時間待機します。待機期間中にロードバランサーによってインス タンスが正常とマークされていない場合、CodeDeploy はデプロイ設定に基づいて、次のインスタン スに移動するか、デプロイに失敗します。

インプレースデプロイでは、1 つ以上の Classic Load Balancer、Application Load Balancer ターゲッ トグループ、または Network Load Balancer ターゲットグループを指定できます。ロードバランサー をデプロイグループの設定の一部として指定できます。または CodeDeploy が提供するスクリプト を使用してロードバランサーを実装できます。

デプロイグループを使用してインプレースデプロイのロードバランサーを指定する

デプロイグループにロードバランサーを追加するには、CodeDeploy コンソールまたは を使用しま す AWS CLI。インプレースデプロイでロードバランサーをデプロイグループで指定する詳細につい ては、次のトピックを参照してください。

- インプレースデプロイ (コンソール) 用のアプリケーションを作成
- インプレースデプロイ用のデプロイグループを作成する (コンソール)
- <u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバランサーをセットアップする</u>

スクリプトを使用してインプレースデプロイのロードバランサーを指定する

次の手順のステップに従ってデプロイライフサイクルスクリプトを使用し、インプレースデプロイの ロードバランシングをセットアップします。

Note

CodeDeployDefault.OneAtATime 設定は、スクリプトを使用してインプレイスデプロイ 用のロードバランサーを設定するときのみ使用します。同時実行はサポートされておら ず、CodeDeployDefault.OneAtATime 設定によりスクリプトの直列実行が確実になります。 デプロイ設定の詳細については、<u>CodeDeployでデプロイ設定を使用する</u>を参照してくださ い。

GitHub の CodeDeploy サンプルリポジトリでは、CodeDeployElastic Load Balancing ロードバランサーの使用に対応できる手順とサンプルを提供します。これらのレポジ トリには、開始するのに必要なすべてのコードを提供する 3 つのサンプルスクリプ ト、register_with_elb.sh、deregister_from_elb.sh、および common_functions.sh
が含まれます。これらの3つのスクリプトのプレースホルダーを編集して、appspec.yml ファイル からこれらのスクリプトを参照します。

Elastic Load Balancing ロードバランサーに登録された Amazon EC2 インスタンスを使用して CodeDeploy でインプレースデプロイをセットアップするには、以下を実行します。

- 1. インプレースデプロイで使用するロードバランサーのタイプのサンプルをダウンロードします。
 - Classic Load Balancer
 - <u>Application Load Balancer または Network Load Balancer (同じスクリプトをいずれのタイプ</u> にも使用可能)
- 2. ターゲットの各 Amazon EC2 インスタンスに AWS CLI がインストールされていることを確認 します。
- ターゲットの各 Amazon EC2 インスタンスで、IAM インスタンスプロファイルに少なくとも elasticloadbalancing:* および autoscaling:* アクセス許可がアタッチされていることを確認しま す。
- アプリケーションのソースコードディレクトリにデプロイライフサイクルイベントのスクリプト (register_with_elb.sh、deregister_from_elb.sh、および common_functions.sh) を含めます。
- アプリケーションリビジョンの appspec.yml で、ApplicationStart イベントの間 register_with_elb.sh スクリプトを、および ApplicationStop イベントの間 deregister_from_elb.sh スクリプトを実行する CodeDeploy の手順を提供します。
- 6. インスタンスが Amazon EC2 Auto Scaling グループの一部である場合、このステップは省略で きます。

common_functions.sh スクリプトで:

- <u>Classic Load Balancer</u> を使用している場合、ELB_LIST="" で Elastic Load Balancing ロード バランサーの名前を指定し、ファイルの他のデプロイ設定に必要な変更を加えます。
- <u>Application Load Balancer または Network Load Balancer</u> を使用している場合 は、TARGET_GROUP_LIST=''' で Elastic Load Balancing ターゲットグループ名を指定し、 ファイルの他のデプロイ設定に必要な変更を加えます。
- アプリケーションのソースコード appspec.yml およびデプロイライフサイクルイベントのス クリプトをアプリケーションリビジョンにバンドルしてから、リビジョンをアップロードしま す。Amazon EC2 インスタンスにリビジョンをデプロイします。デプロイの間に、デプロイラ イフサイクルイベントのスクリプトは、Amazon EC2 インスタンスをロードバランサーから登

録解除して、接続がドレインするまで待機し、デプロイが完了してから Amazon EC2 インスタンスをロードバランサーに再登録します。

パートナーの製品とサービスとの統合

CodeDeploy には、次のパートナー製品とサービスとの統合が組み込まれています。

Ansible	すでに一連の Ansible プレイブックがあり、 実行する場所が必要なだけの場合、Ansible と CodeDeploy のテンプレートは、いくつかのシ ンプルなデプロイフックによって Ansible を ローカルデプロイインスタンスで使用し、プレ イブックを実行できることを示します。すでに インベントリを構築して維持するためのプロセ スが存在する場合、CodeDeploy エージェント のインストールと実行に使用できる Ansible モ ジュールもあります。 詳細はこちら: • Ansible と CodeDeploy
Atlassian - Bamboo ≿ Bitbucket	Bamboo の CodeDeploy タスクは AppSpec ファイルを含むディレクトリを.zip ファイルに 圧縮、ファイルを Amazon S3 にアップロード し、CodeDeploy アプリケーションで提供され ている設定に従ってデプロイを開始します。 Atlassian Bitbucket での CodeDeploy のサ ポートにより、Bitbucket UI からコードを直接 Amazon EC2 インスタンスにプッシュしたり、 オンデマンドで任意のデプロイグループにプッ シュしたりできます。これは、Bitbucket リポ ジトリのコードを更新した後、手動でのデプロ イプロセスを実行するために、継続的インテ グレーション (CI) プラットフォーム、または

Amazon EC2 インスタンスにサインインする必要がないことを意味します。

詳細はこちら:

- Bamboo の CodeDeploy タスクの使用
- <u>Atlassian Bitbucket による CodeDeploy のサ</u>ポートの発表

AWS には、Chef と CodeDeploy を統合するた めの 2 つのテンプレートサンプルが用意されて います。1 つは CodeDeploy エージェントをイ ンストールして起動する Chef クックブックで す。これにより CodeDeploy を使用中に Chef でのホストインフラストラクチャの管理を継続 することができます。2 番目のサンプルテンプ レートでは、CodeDeploy を使用して、各ノー ドで chef-solo を使用してクックブックとレシ ピの実行を調整する方法を示します。

詳細はこちら:

シェフと CodeDeploy

CircleCI では、自動化テスト、継続的な統合、 およびデプロイのツールセットを提供しま す。CircleCI AWS で使用する IAM ロールを で作成し、circle.yml ファイルでデプロイパラ メータを設定したら、CodeDeploy で CircleCI を使用してアプリケーションリビジョンを作成 し、Amazon S3 バケットにアップロードして から、デプロイを開始してモニタリングできま す。

詳細はこちら:

 <u>CircleCI Orb を使用してアプリケーションを</u> AWS CodeDeployにデプロイする方法

Chef

CircleCI

CloudBees	ビルド後のアクションとして、 <u>CloudBees</u> DEV@cloud で利用可能な CodeDeploy Jenkins プラグインを使用できます。例えば、継続的な 配信パイプラインの終了時に、サーバー群にア プリケーションリビジョンをデプロイするため
	に使用できます。 詳細はこちら:
	・ <u>CodeDeploy Jenkins プラグインが</u> DEV@cloud で利用可能に
Codeship	Codeship を使用して、CodeDeploy 経由でア プリケーションリビジョンをデプロイできま す。Codeship UI を使用して、ブランチのデプ ロイパイプラインに CodeDeploy を追加できま す。
	詳細はこちら:
	・ <u>CodeDeploy へのデプロイ</u> ・ <u>Codeship での CodeDeploy との統合</u>
GitHub	CodeDeploy を使用して、 <u>GitHub</u> リポジトリか らアプリケーションリビジョンをデプロイでき ます。そのリポジトリのソースコードが変更さ れるたびに、GitHub リポジトリからデプロイ をトリガーすることもできます。
	詳細はこちら:
	 <u>GitHub と CodeDeploy との統合</u> <u>チュートリアル: CodeDeploy を使用し</u> <u>て、GitHub からアプリケーションをデプロ</u> <u>イします。</u> <u>CodeDeploy を使用した GitHub からの自動</u> <u>的なデプロイ</u>

HashiCorp Consul	オープンソースの HashiCorp Consul ツールを 使用して、CodeDeploy でアプリケーションを デプロイする時のアプリケーション環境の状 態と安定性を確実にします。Consul を使用し て、デプロイ中に検出されるアプリケーション を登録し、アプリケーションおよびノードをメ ンテナンスモードでデプロイから除外し、ター ゲットのインスタンスに異常が生じた場合はデ プロイを停止します。 詳細はこちら:
Jenkins	CodeDeploy Jenkins プラグインでは、Jenkins プロジェクトのビルド後のステップを示しま す。ビルドに成功すると、ワークスペースが圧 縮されて Amazon S3 にアップロードされ、新 しいデプロイが開始されます。 詳細はこちら: • CodeDeploy Jenkins プラグイン • CodeDeploy の Jenkins プラグインのセット アップ

Puppet Labs	AWS には、 <u>Puppet</u> と CodeDeploy のサンプル テンプレートが用意されています。最初のテン プレートは CodeDeploy エージェントをインス トールして起動する Puppet モジュールです。 これにより CodeDeploy を使用中に Puppet で のホストインフラストラクチャの管理を継続 することができます。2 番目のサンプルテンプ レートでは、CodeDeploy を使用してモジュー ルとマニフェストの実行を、各ノードのマス ターレス puppet で調整する方法を示します。 詳細はこちら:
SaltStack	SaltStack のインフラストラクチャを CodeDeploy と統合できます。CodeDeploy モジュールを使用して、ミニオンズに CodeDeploy エージェントをインストールして 実行できます。または、いくつかのシンプル なデプロイフックを使用して、CodeDeployで Salt States の実行を調整できます。 詳細はこちら: • SaltStack と CodeDeploy

TeamCity	CodeDeploy Runner プラグインを使用し て、TeamCity からアプリケーションを直接 デプロイできます。このプラグインは、アプ リケーションリビジョンを準備して Amazon S3 バケットにアップロードし、CodeDeploy アプリケーションでリビジョンを登録し 、CodeDeploy のデプロイを作成します。必 要に応じて、デプロイが完了するまで待つ TeamCity のビルドステップを追加します。 詳細はこちら: • CodeDeploy ランナー (ダウンロード) • CodeDeploy ランナープラグイン (ドキュメ ント)
Travis CI	<u>Travis CI</u> を設定して、ビルドが成功した 後、CodeDeploy でデプロイメントをトリガー できます。
	詳細はこちら:
	・ <u>Travis CI と CodeDeploy デプロイ</u>

トピック

• <u>GitHub と CodeDeploy との統合</u>

GitHub と CodeDeploy との統合

CodeDeploy は、ウェブベースのコードホストおよび共有サービスである <u>GitHub</u>をサポートしま す。CodeDeploy は、GitHub リポジトリまたは Amazon S3 バケットに格納されているアプリケー ションリビジョンをインスタンスにデプロイできます。CodeDeploy では、EC2 およびオンプレミス デプロイのみ GitHub がサポートされています。

トピック

<u>GitHub からの CodeDeploy リビジョンのデプロイ</u>

• CodeDeploy を使用した GitHub 動作

GitHub からの CodeDeploy リビジョンのデプロイ

GitHub リポジトリからインスタンスへアプリケーションリビジョンをデプロイするには。

 CodeDeploy およびデプロイする Amazon EC2 インスタンスタイプと互換性があるリビジョン を作成します。

互換性のあるリビジョンを作成するには、<u>CodeDeploy のリビジョンを計画する</u>および CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追加 の手順に従います。

2. GitHub のアカウントを使用して、GitHub リポジトリにリビジョンを追加します。

GitHub のアカウントを作成するには、「<u>GitHub への参加</u>」を参照してください。GitHub リポ ジトリを作成するには、「Repo の作成」を参照してください。

 CodeDeploy コンソールのデプロイの作成ページまたは AWS CLI create-deployment コマンドを 使用して、GitHub リポジトリから CodeDeploy デプロイで使用するように設定されたターゲッ トインスタンスにリビジョンをデプロイします。

create-deployment コマンドを呼び出す場合、最初にコンソールの [デプロイの作成] ページを使 用して、指定したアプリケーションの推奨 GitHub アカウントの代わりに GitHub を操作するた めのアクセス許可を CodeDeploy に付与します。これを行う必要があるのは、アプリケーショ ンごとに 1 度だけです。

[Create deployment] ページを使用して、GitHub レポジトリからデプロイする方法については、 「CodeDeploy でデプロイを作成する」を参照してください。

create-deployment コマンドを呼び出して GitHub レポジトリからデプロイする方法について は、「<u>EC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)</u>」を参照し てください。

CodeDeploy デプロイで使用するためインスタンスを準備する方法については、「<u>CodeDeploy</u> <u>のためにインスタンスを用いた操作</u>」を参照してください。

詳細については、「<u>チュートリアル: CodeDeploy を使用して、GitHub からアプリケーションをデプ</u> <u>ロイします。</u>」を参照してください。

CodeDeploy を使用した GitHub 動作

トピック

- CodeDeploy のアプリケーションを使用した GitHub の認証
- ・ プライベートおよびパブリックな GitHub リポジトリとの CodeDeploy のやり取り
- CodeDeploy と組織マネージド型の GitHub リポジトリのやり取り
- <u>CodeDeploy</u>を使用して CodePipeline から自動的にデプロイする

CodeDeploy のアプリケーションを使用した GitHub の認証

GitHub を操作するためのアクセス許可を CodeDeploy に付与すると、GitHub アカウントとアプリ ケーションの間の関連付けが CodeDeploy に格納されます。GitHub の他のアカウントにアプリケー ションをリンクできます。また、GitHub を操作するための CodeDeploy に対するアクセス許可を取 り消すこともできます。

GitHub アカウントを CodeDeploy のアプリケーションにリンクするには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. 別の GitHub アカウントにリンクするアプリケーションを選択します。
- アプリケーションにデプロイグループがない場合は、[デプロイグループの作成] を選択してグ ループを作成します。詳細については、「<u>CodeDeploy でデプロイグループを作成する</u>」を参照 してください。次のステップで [デプロイの作成] を選択するには、デプロイグループが必要で す。
- 5. [デプロイ] から、[デプロイの作成] を選択します。

新しいデプロイを作成する必要はありません。これは現在 GitHub の他のアカウントを アプリケーションにリンクする唯一の方法です。

- 6. [デプロイ設定] の [リビジョンタイプ] で、[アプリケーションは GitHub に格納されています] を 選択します。
- 7. 次のいずれかを行います:
 - GitHub アカウントへの AWS CodeDeploy アプリケーションの接続を作成するには、別のウェ ブブラウザタブで GitHub からサインアウトします。[GitHub token name (GitHub トークン 名)] に、この接続を識別する名前を入力し、[GitHub に接続] を選択します。アプリケーショ ンの GitHub を操作することを CodeDeploy に許可するよう求めるメッセージがウェブページ に表示されます。ステップ 10 に進みます。
 - 作成済みの接続を使用するには、その名前を [GitHub token name (GitHub トークン名)] で選 択してから [GitHub に接続] を選択します。ステップ 8 に進みます。
 - 別の GitHub アカウントへの接続を作成するには、ウェブブラウザの別のタブで GitHub か らサインアウトします。[GitHub token name (GitHub トークン名)] に、接続を識別する名前 を入力し、[GitHub に接続] を選択します。アプリケーションの GitHub を操作することを CodeDeploy に許可するよう求めるメッセージがウェブページに表示されます。ステップ 10 に進みます。
- 8. GitHub にまだサインインしていない場合は、[Sign in] ページの手順に従い、アプリケーション をリンクする GitHub アカウントにサインインします。
- 9. [Authorize application] を選択します。GitHub は指定したアプリケーションの署名 GitHub アカ ウントの代わりに、CodeDeploy に GitHub を操作するアクセス許可を付与します。
- 10. デプロイを作成しない場合は、[Cancel] を選択します。

CodeDeploy が GitHub を操作するためのアクセス許可を取り消すには

- AWS CodeDeploy アクセス許可を取り消したい GitHub アカウントの認証情報を使用して、GitHub にサインインします。
- GitHub の [<u>Applications</u>] ページを開いて、承認されたアプリケーションのリストで [CodeDeploy] を見つけ、アプリケーションの承認取り消しの GitHub の手順に従います。

プライベートおよびパブリックな GitHub リポジトリとの CodeDeploy のやり取り

CodeDeploy は、プライベートおよびパブリックの GitHub リポジトリからのアプリケーションのデ プロイをサポートします。ユーザーに代わって GitHub にアクセスするアクセス許可を CodeDeploy に付与した場合、CodeDeploy は、そのユーザーの GitHub アカウントがアクセス権を持つすべ てのプライベート GitHub リポジトリに対して読み取り/書き込みアクセス権を持ちます。ただ し、CodeDeploy は、GitHub リポジトリからのみ読み込みます。プライベート GitHub のリポジトリ に書き込むことはありません。

CodeDeploy と組織マネージド型の GitHub リポジトリのやり取り

デフォルトでは、組織によって管理される GitHub リポジトリ (アカウントのプライベートまたはパ ブリックリポジトリとは対照的に) は、CodeDeploy を含め、サードパーティアプリケーションへの アクセスを許可しません。デプロイは、組織によるサードパーティアプリケーション制限が GitHub で有効化され、GitHub リポジトリからコードをデプロイしようとすると失敗します。この問題を解 決する 2 つの方法があります。

- 組織のメンバーとして、組織の所有者に CodeDeploy へのアクセスを承認するように要求できます。このアクセスをリクエストする手順は、個々のアカウントに対して、すでに CodeDeploy を認証済みかどうかによって異なります。
 - ユーザーのアカウントに CodeDeploy への認証済みアクセス権がある場合は、「<u>組織への認証</u> 済みアプリケーションの承認リクエスト」を参照してください。
 - ユーザーのアカウントに CodeDeploy への承認済みアクセス権がまだない場合は、「<u>組織への</u> サードパーティアプリケーションの承認リクエスト」を参照してください。
- 組織の所有者は、組織に対するサードパーティアプリケーションの制限を無効にできます。詳細については、「組織に対するサードパーティアプリケーションの制限の無効化」を参照してください。

詳細については、「サードパーティアプリケーションの制限について」を参照してください。

CodeDeploy を使用して CodePipeline から自動的にデプロイする

ソースコードが変わるたびに CodePipeline からデプロイをトリガーできます。詳細については、 「CodePipeline」を参照してください。

コミュニティから統合の例

以下のセクションは、ブログの投稿や記事、およびコミュニティで提供されている例へのリンクで す。

Note

これらのリンクは、情報提供のみを目的として提供されており、包括的なリストまたはコン テンツの例の内容を推奨するものではありません。 AWS は、外部コンテンツの内容または 正確性について責任を負いません。

ブログ記事

での CodeDeploy プロビジョニングの自動化 AWS CloudFormation

AWS CloudFormationを使用して、CodeDeploy でアプリケーションのデプロイをプロビジョニン グする方法について説明します。

2016 年 1 月投稿

・ AWS Toolkit for Eclipse CodeDeploy との統合 (パート 1)

AWS Toolkit for Eclipse CodeDeploy との統合 (パート 2)

AWS Toolkit for Eclipse CodeDeploy との統合 (パート 3)

Java 開発者が CodeDeploy plugin for Eclipse を使用して、Eclipse 開発環境 AWS から直接 にウェ ブアプリケーションをデプロイする方法について説明します。

2015 年 2 月投稿

• <u>CodeDeploy を使用した GitHub からの自動的なデプロイ</u>

GitHub から CodeDeploy への自動デプロイを使用して、ソース管理からテストまたは本番環境に 至るまでのエンドツーエンドのパイプラインを作成する方法をご覧ください。

2014 年 12 月投稿

CodeDeploy: チュートリアル

このセクションには、CodeDeploy の使用方法の学習に役立ついくつかのチュートリアルが含まれて います。

これらのチュートリアルの手順では、ファイルを保存する場所 (c:\temp など) と、バ ケット、サブフォルダ、またはファイルに追加する名前 (それぞれ amzn-s3-demobucket、HelloWorldApp、CodeDeployDemo-EC2-Trust.json など) について提案しますが、それらを 使用する必要はありません。手順を実行する際に、ファイルの場所と名前は必ず置き換えてくださ い。

トピック

- チュートリアル: Amazon EC2 インスタンスに WordPress をデプロイする (Amazon Linux または Red Hat エンタープライズ Linux および Linux、macOS、または Unix)
- チュートリアル: 「Hello, World!」 CodeDeploy を使用したアプリケーション (Windows Server)
- チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エンタープ ライズ Linux) を使用してオンプレミスインスタンスにアプリケーションをデプロイします。
- チュートリアル: CodeDeploy を使用して、Auto Scaling グループにアプリケーションをデプロイ します。
- ・ チュートリアル: CodeDeploy を使用して、GitHub からアプリケーションをデプロイします。
- <u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイする</u>
- ・ チュートリアル: 検証テストを使用して Amazon ECS サービスをデプロイする
- チュートリアル: CodeDeploy と AWS サーバーレスアプリケーションモデルを使用して、更新された Lambda 関数をデプロイする

チュートリアル: Amazon EC2 インスタンスに WordPress をデプ ロイする (Amazon Linux または Red Hat エンタープライズ Linux および Linux、macOS、または Unix)

このチュートリアルでは、PHP および MySQL に基づくオープンソースのブログツールおよびコン テンツ管理システムである WordPress を、Amazon Linux または Red Hat Enterprise Linux (RHEL) を実行している 1 つの Amazon EC2 インスタンスにデプロイします。

お探しのものではありませんか。

- 代わりに Windows Server を実行している Amazon EC2 インスタンスへのデプロイの演習を行う 場合は、「<u>チュートリアル:「Hello, World!」CodeDeploy を使用したアプリケーション (Windows</u> Server)」を参照してください。
- Amazon EC2 インスタンスではなくオンプレミスインスタンスへのデプロイの演習を行う場合 は、「<u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エ</u> <u>ンタープライズ Linux</u>)を使用してオンプレミスインスタンスにアプリケーションをデプロイしま す。」を参照してください。

このチュートリアルのステップは、Linux や macOS、Unix を実行しているローカルの開発用マシン の観点から説明されています。Windows を実行しているローカルマシンでこれらのステップのほと んどを完了できますが、chmod や wget などのコマンド、sed などのアプリケーション、および / tmp などのディレクトリパスを扱うステップは環境に合わせて変更する必要があります。

このチュートリアルを開始する前に、「<u>CodeDeploy の開始方法</u>」の前提条件を完了している必要が あります。これには、ユーザーの設定、 のインストールまたはアップグレード AWS CLI、IAM イン スタンスプロファイルとサービスロールの作成が含まれます。

トピック

- <u>ステップ 1: Amazon Linux または Red Hat エンタープライズ Linux Amazon EC2 インスタンスを</u> 起動して設定する
- <u>ステップ 2: Amazon Linux または Red Hat エンタープライズ Linux Amazon EC2 インスタンスに</u> デプロイされるようにソースコンテンツを設定する
- ステップ 3: Amazon S3 に WordPress アプリケーションをアップロードする
- <u>ステップ 4: WordPress アプリケーションをデプロイする</u>
- ステップ 5: WordPress アプリケーションを更新して再デプロイする
- ステップ 6: WordPress のアプリケーションと関連リソースのクリーンアップ

ステップ 1: Amazon Linux または Red Hat エンタープライズ Linux Amazon EC2 インスタンスを起動して設定する

CodeDeploy を使用して WordPress アプリケーションをデプロイするには、Amazon Linux または Red Hat Enterprise Linux (RHEL) を実行する Amazon EC2 インスタンスが必要です。Amazon EC2 インスタンスでは、HTTP 接続を許可する新しいインバウンドセキュリティルールが必要です。この ルールは、正しくデプロイした後で WordPress ページをブラウザで表示するために必要です。 「<u>CodeDeploy のための Amazon EC2 インスタンスを作成します。</u>」の手順に従います インスタン スの Amazon EC2 インスタンスタグの割り当てに関するこれらの指示の一部を実行する際には、必 ず Name のタグキーと、CodeDeployDemo のタグ値を指定します。(別のタグキーまたはタグ値を指 定した場合、<u>ステップ 4: WordPress アプリケーションをデプロイする</u>の手順で予期しない結果が生 成される場合があります)。

Amazon EC2 インスタンスを起動する手順に従った後、このページに戻り、次のセクションに進 みます。次のステップとして、<u>CodeDeploy でアプリケーションを作成する</u> には進まないでくださ い。

Amazon Linux と RHEL Amazon EC2 インスタンスに接続します

Amazon EC2 インスタンスが起動した後、手順に従ってインスタンスに接続する演習をします。

 ssh コマンド (または SSH 対応の <u>PuTTY</u> などのターミナルエミュレータ) を使用して Amazon Linux または RHEL Amazon EC2 インスタンスに接続します。Amazon EC2 インスタンスを開 始した時に使用したインスタンスのパブリック DNS アドレスとキーペアのプライベートキーが 必要になります。詳細については、「インスタンスへの接続」を参照してください。

例えば、パブリック DNS アドレスが ec2-01-234-567-890.compute-1.amazonaws.com で、SSH アクセスの Amazon EC2 インスタンスキーペアが codedeploydemo.pem という名 前の場合、以下のように入力します。

ssh -i /path/to/codedeploydemo.pem ec2user@ec2-01-234-567-890.compute-1.amazonaws.com

<u>/path/to/</u>codedeploydemo.pem を .pem ファイルのパスに置き換え、例の DNS アドレス を Amazon Linux または RHEL Amazon EC2 インスタンスのアドレスと置き換えます。

Note

キーファイルのアクセス権限がオープンすぎるというエラーを受信した場合は、現在の ユーザー (お客様) だけにアクセス権限を与えるように限定する必要があります。例え ば、chmod Linux、macOS、または UNIX の場合は、次のように入力します。

chmod 400 /path/to/codedeploydemo.pem

2. サインインしたら、Amazon EC2 インスタンスについては、AMI のバナーを参照してください。Amazon Linux の場合は、次のようになります。

3. 実行中の Amazon EC2 インスタンスからサインアウトできます。

A Warning

Amazon EC2 インスタンスを停止または削除しないでください。そうしない と、CodeDeploy がインスタンスにデプロイできません。

HTTP トラフィックを許可するインバウンドルールの Amazon Linux または RHEL Amazon EC2 インスタンスへの追加

次のステップでは、デプロイされた WordPress アプリケーションのホームページがブラウザに表示 されるように、Amazon EC2 インスタンスに開いている HTTP ポートがあることを確認します。

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- 2. インスタンスを選択後、ご自分のインスタンスを選択します。
- 3. セキュリティグループ 内の 説明 タブで、インバウンドのルールの表示 を選択します。

セキュリティグループのルールの一覧は、次のように表示されます。

Security	Groups	associat	ted	with	i- 2	1234567890abcdef0
Ports	Proto	ocol	Soi	urce		launch-wizard- <mark>N</mark>
22	tcp		0.0	0.0.0/	/0	#

 セキュリティグループでは、Amazon EC2 インスタンスのセキュリティグループを選択します。launch-wizard-N と名前が付けられる可能性があります。N は、インスタンスの作成時 にセキュリティグループに割り当てられた名前です。

[Inbound] (インバウンド) タブを選択します。次の値を持つルールが表示された場合、ご利用の インスタンスのセキュリティグループは正しく設定されています。

- [Type]: HTTP
- [Protocol]: TCP
- [Port Range]: 80
- ・ソース: 0.0.0.0/0
- 5. これらの値を持つルールが表示されない場合は、<u>セキュリティグループへのルールの追加</u>の手 順を使用して、新しいセキュリティルールに追加します。

ステップ 2: Amazon Linux または Red Hat エンタープライズ Linux Amazon EC2 インスタンスにデプロイされるようにソースコンテンツを設 定する

ここでは、アプリケーションのソースコンテンツを設定し、インスタンスにデプロイできるものを用 意します。

トピック

- ソースコードの入手
- アプリケーションを実行するスクリプトの作成
- アプリケーション仕様ファイルの追加

ソースコードの入手

このチュートリアルでは、開発マシンからターゲット Amazon EC2 インスタンスに WordPress コン テンツ発行プラットフォームをデプロイします。WordPress のソースコードを入手するには、組み 込みのコマンドラインの呼び出しを使用できます。または、開発マシンに Git をインストールしてい る場合は、代わりにそれを使用します。

これらのステップでは、WordPress ソースコードのコピーを開発マシンの /tmp ディレクトリにダ ウンロードすることを前提としています (任意のディレクトリを選択できますが、ステップで /tmp が指定されている場合は、その場所に必ず置き換えてください)。

次の 2 つのオプションのいずれかを選択して、開発マシンに WordPress ソースファイルをコピーし ます。最初のオプションでは、組み込みのコマンドラインの呼び出しを使用します。2 番目のオプ ションでは、Git を使用します。

トピック

- WordPress のソースコード (組み込みのコマンドライン呼び出し) のコピーを入手するには
- ・ WordPress ソースコード (Git) のコピーを入手するには

WordPress のソースコード (組み込みのコマンドライン呼び出し) のコピーを入手するには

 wget コマンドを呼び出して、WordPress のソースコードのコピーを .zip ファイル形式で現在の ディレクトリにダウンロードします。

wget https://github.com/WordPress/WordPress/archive/master.zip

- 2. unzip、mkdir、cp、rm コマンドを呼び出して、以下を実行します。
 - master.zip ファイルを /tmp/WordPress_Temp ディレクトリ (フォルダ) に解凍します。
 - 解凍された内容を、/tmp/WordPress 宛先フォルダにコピーします。
 - 一時的な /tmp/WordPress_Temp フォルダと master ファイルを削除します。

コマンドを一度に1つずつ実行します。

unzip master -d /tmp/WordPress_Temp

mkdir -p /tmp/WordPress

cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress

rm -rf /tmp/WordPress_Temp

rm -f master

これにより、/tmp/WordPress フォルダに WordPress ソースコードファイルのクリーンな セットが配置されます。

WordPress ソースコード (Git) のコピーを入手するには

- 1. 開発マシンで Git をダウンロードしてインストールします。
- 2. /tmp/WordPress フォルダで git init コマンドを呼び出します。

 git clone コマンドを呼び出してパブリック WordPress レポジトリのクローンを作成し、その独 自のコピーを /tmp/WordPress 宛先フォルダで作成します。

git clone https://github.com/WordPress/WordPress.git /tmp/WordPress

これにより、/tmp/WordPress フォルダに WordPress ソースコードファイルのクリーンな セットが配置されます。

アプリケーションを実行するスクリプトの作成

次に、ディレクトリでフォルダとスクリプトを作成します。CodeDeploy はこれらのスクリプトを使 用して、ターゲット Amazon EC2 インスタンスでアプリケーションリビジョンをセットアップし、 デプロイします。スクリプトの作成には任意のテキストエディタを使用できます。

1. WordPress ソースコードのコピーでスクリプトディレクトリを作成します。

mkdir -p /tmp/WordPress/scripts

 install_dependencies.shに /tmp/WordPress/scripts ファイルを作成します。 ファイルに以下の行を追加します。この install_dependencies.sh スクリプトは Apache、MySQL、および PHP をインストールします。また、PHP に MySQL のサポートを追 加します。

```
#!/bin/bash
sudo amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

 start_server.shに /tmp/WordPress/scripts ファイルを作成します。ファイルに以下 の行を追加します。この start_server.sh スクリプトは Apache および MySQL を起動しま す。

```
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

 stop_server.shに /tmp/WordPress/scripts ファイルを作成します。ファイルに以下の 行を追加します。この stop_server.sh スクリプトは Apache および MySQL を停止します。

```
#!/bin/bash
isExistApp="pgrep httpd"
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service
fi
```

5. create_test_db.sh に /tmp/WordPress/scripts ファイルを作成します。ファイル に以下の行を追加します。この create_test_db.sh スクリプトは、MySQL を使用し て、WordPress で使用する **test** データベースを作成します。

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB</pre>
```

 6. 最後に、/tmp/WordPress/scripts に change_permissions.sh スクリプトファイルを作 成します。これは Apache のフォルダのアクセス権限を変更するために使用されます。

A Important

このスクリプトにより、誰でも書き込めるように、/tmp/WordPress フォルダのアク セス権限が更新されました。これは、「<u>ステップ 5: WordPress アプリケーションを更</u> <u>新して再デプロイする</u>」で WordPress がそのデータベースに書き込みを行うために必 要です。WordPress アプリケーションがセットアップされたら、次のコマンドを実行し て、アクセス権限をよりセキュアな設定に更新します。

chmod -R 755 /var/www/html/WordPress

#!/bin/bash
chmod -R 777 /var/www/html/WordPress

7. すべてのスクリプトに実行可能権限を付与します。コマンドラインで、次のように入力します。

chmod +x /tmp/WordPress/scripts/*

アプリケーション仕様ファイルの追加

次に、アプリケーション仕様ファイル (AppSpec ファイル) <u>YAML</u> CodeDeploy によって使用される フォーマットされたファイルを追加します。

- アプリケーションリビジョンのソースファイルを、ターゲット Amazon EC2 インスタンスの宛先 にマッピングする。
- デプロイされたファイルのカスタムアクセス権限を指定する。
- デプロイ中にターゲット Amazon EC2 インスタンスで実行するスクリプトを指定する。

AppSpec のファイル名は、appspec.yml とする必要があります。アプリケーションソースコード のルートディレクトリに配置する必要があります。このチュートリアルでは、ルートディレクトリは /tmp/WordPress です。

テキストエディタで appspec.yml という名前のファイルを作成します。このファイルに次の行を 追加します。

```
version: 0.0
os: linux
files:
    - source: /
    destination: /var/www/html/WordPress
hooks:
    BeforeInstall:
        - location: scripts/install_dependencies.sh
        timeout: 300
        runas: root
```

AfterInstall:
 location: scripts/change_permissions.sh
timeout: 300
runas: root
ApplicationStart:
- location: scripts/start_server.sh
 location: scripts/create_test_db.sh
timeout: 300
runas: root
ApplicationStop:
- location: scripts/stop_server.sh
timeout: 300
runas: root

CodeDeploy はこの AppSpec ファイルを使用して、開発マシンの /tmp/WordPress フォルダのす べてのファイルを、ターゲットの Amazon EC2 インスタンスの /var/www/html/WordPress フォ ルダにコピーします。デプロイ中に CodeDeploy は、デプロイライフサイクルで root や /var/ www/html/WordPress/scripts などの中に指定されたイベントで、指定するスクリプトをター ゲットの Amazon EC2 インスタンスの **BeforeInstall、AfterInstall**のフォルダとして実行し ます。これらのスクリプトのいずれかで実行に 300 秒 (5 分) 以上かかる場合、CodeDeploy はデプ ロイを停止し、デプロイを失敗としてマークします。

これらの設定の詳細については、「<u>CodeDeploy AppSpec ファイルのリファレンス</u>」を参照してく ださい。

A Important

このファイルの項目間のスペースの場所と数は重要です。間隔が正しくない場

- 合、CodeDeploy はデバッグが困難な可能性のあるエラーを発生させます。詳細について
- は、「<u>AppSpec ファイルの間隔</u>」を参照してください。

ステップ 3: Amazon S3 に WordPress アプリケーションをアップロードす る

ここでソースコンテンツを CodeDeploy がデプロイできる場所に準備してアップロードします。次 の手順では、Amazon S3 バケットをプロビジョニングしてバケット用のアプリケーションリビジョ ンのファイルを準備し、リビジョンのファイルをバンドルしてから、そのリビジョンをバケットに プッシュする方法を示します。 Note

このチュートリアルでは説明されていませんが、CodeDeploy を使用して GitHub リポジト リからインスタンスにアプリケーションをデプロイできます。詳細については、「<u>GitHub と</u> CodeDeploy との統合」を参照してください。

トピック

- Amazon S3 バケットをプロビジョニングします
- バケットのアプリケーションファイルを準備する
- アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブファイルを プッシュする

Amazon S3 バケットをプロビジョニングします

ストレージコンテナあるいは Amazon S3 バケット を作成、または既存のバケットを使用します。バ ケットにリビジョンをアップロードできること、およびデプロイで使用する Amazon EC2 インスタ ンスがバケットからリビジョンをダウンロードできることを確認します。

AWS CLI、Amazon S3 コンソール、または Amazon S3 APIs を使用してAmazon S3バケットを作成 できます。バケットを作成したら、バケットとその AWS アクセス許可を付与します。

Note

バケット名は、すべての AWS アカウントで Amazon S3 全体で一意である必要がありま す。amzn-s3-demo-bucket を使用できない場合、amzn-s3-demo-bucket の後にダッ シュと自分の名前のイニシャル、または他の一意な識別子など別のバケット名を試してく ださい。このチュートリアル全体で、バケット名を amzn-s3-demo-bucket に置き換えま す。

Amazon S3 バケットは、ターゲット Amazon EC2 インスタンスが起動されるのと同じ AWS リージョンに作成する必要があります。例えば、バケットを米国東部 (バージニア北部) リー ジョンで作成する場合、対象の Amazon EC2 インスタンスを米国東部 (バージニア北部) リージョンで起動する必要があります。

トピック

• Amazon S3 バケット (CLI) の作成

- Amazon S3 バケット (コンソール) の作成
- Amazon S3 バケットと AWS アカウントにアクセス許可を付与する

Amazon S3 バケット (CLI) の作成

mb コマンドを呼び出して、**amzn-s3-demo-bucket** という名前の Amazon S3 バケットを作成し ます。

aws s3 mb s3://amzn-s3-demo-bucket --region region

Amazon S3 バケット (コンソール) の作成

- 1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
- 2. Amazon S3 コンソールで [バケットの作成] を選択します。
- 3. [Bucket name] ボックスで、バケットの名前を入力します。
- 4. [Region] リストで、ターゲットリージョンを選択し、[Create] を選択します。

Amazon S3 バケットと AWS アカウントにアクセス許可を付与する

Amazon S3 バケットへのアップロードには、許可が必要です。Amazon S3 バケットポリシーで、これらのアクセス許可を指定できます。たとえば、次の Amazon S3 バケットポリシーでは、ワイルド カード文字 (*) を使用すると、 AWS アカウント111122223333は という名前の Amazon S3 バケッ ト内の任意のディレクトリにファイルをアップロードできますamzn-s3-demo-bucket。

]

}

AWS アカウント ID を表示するには、AWS 「アカウント ID の検索」を参照してください。

今は、Amazon S3 バケットが参加している各 Amazon EC2 インスタンスからのダウンロードリクエ ストを許可していることを確認するのに適した時期です。Amazon S3 バケットポリシーで、これを 指定できます。例えば、次の Amazon S3 バケットポリシーでは、ワイルドカード文字 (*)を使用す ると、ARN arn:aws:iam::444455556666:role/CodeDeployDemo を含む IAM インスタンスプ ロファイルがアタッチされた Amazon EC2 インスタンスが、amzn-s3-demo-bucket という名前 の Amazon S3 バケットの任意のディレクトリからファイルをダウンロードすることを許可します。

```
{
    "Statement": [
        {
             "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
             "Principal": {
                "AWS": [
                     "arn:aws:iam::444455556666:role/CodeDeployDemo"
                1
            }
        }
    ]
}
```

Amazon S3 バケットポリシーを生成しアタッチする方法の詳細については、「<u>バケットポリシーの</u> 例」を参照してください。

IAM ポリシーを作成しアタッチする方法については、「ポリシーの使用」を参照してください。

バケットのアプリケーションファイルを準備する

WordPress アプリケーションファイル、およびスクリプトが、次のような開発用マシンで構成されることを確認します。

/tmp/

ステップ 3: Amazon S3 にアプリケーションをアップロードする

```
|--WordPress/
    |-- appspec.yml
    |-- scripts/
       |-- change_permissions.sh
        |-- create_test_db.sh
        |-- install_dependencies.sh
        |-- start_server.sh
        |-- stop_server.sh
    -- wp-admin/
        -- (various files...)
    -- wp-content/
       -- (various files...)
   |-- wp-includes/
       |-- (various files...)
   |-- index.php
   license.txt
   |-- readme.html
   |-- (various files ending with .php...)
```

アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブ ファイルをプッシュする

WordPress アプリケーションファイルと をアーカイブファイル (アプリケーション リビジョン と呼ばれる) にバンドルします。

Note

バケットにオブジェクトを保存したり、バケットの内外にアプリケーションのリビジョンを 転送したりする場合に課金されることがあります。詳細については、「<u>Amazon S3 の料金</u>」 を参照してください。

1. 開発マシンで、ファイルが保存されたフォルダに切り替えます。

cd /tmp/WordPress

Note

このフォルダに切替わらなければ、ファイルのバンドルは現在のフォルダで起動されま す。例えば、現在のフォルダが /tmp ではなく /tmp/WordPress である場合、バンド ルは、tmp サブフォルダ以上を含む可能性のある WordPress フォルダ内のファイルと サブフォルダから開始します。

create-application コマンドを呼び出して、WordPress_App という名前の新しいアプリケーションを登録します。

aws deploy create-application --application-name WordPress_App

 CodeDeploy で <u>push</u> コマンドを呼び出してファイルをまとめてバンドルし、Amazon S3 に リビジョンをアップロードし、アップロードされたリビジョンに関する情報を1つの操作で CodeDeploy に登録します。

aws deploy push \
 --application-name WordPress_App \
 --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
 --ignore-hidden-files

このコマンドは、現在のディレクトリ (隠しファイルを除く) から、WordPressApp.zip とい う名前の 1 つのアーカイブファイルにファイルをバンドルし、リビジョンを amzn-s3-demobucket バケットにアップロードし、CodeDeploy でアップロードしたリビジョンについての情 報を登録します。

ステップ 4: WordPress アプリケーションをデプロイする

ここで、Amazon S3 にアップロードしたサンプルの WordPress アプリケーションリビジョンをデプ ロイします。 AWS CLI または CodeDeploy コンソールを使用してリビジョンをデプロイし、デプロ イの進行状況をモニタリングできます。アプリケーションリビジョンが正常にデプロイされた後に、 その結果を確認します。

トピック

- CodeDeploy を使用して、アプリケーションリビジョンをデプロイします
- デプロイをモニタリングおよびトラブルシューティングします。
- デプロイの確認

CodeDeploy を使用して、アプリケーションリビジョンをデプロイします

AWS CLI または コンソールを使用して、アプリケーションリビジョンをデプロイします。

トピック

- アプリケーションリビジョン (CLI) をデプロイするには
- アプリケーションリビジョン (コンソール) のデプロイ

アプリケーションリビジョン (CLI) をデプロイするには

デプロイにはデプロイグループが必要です。ただし、デプロイグループを作成する前に、サービスロール ARN が必要です。サービスロールは、ユーザーに代わってサービスアクセス権限を付与する IAM ロールです。この場合、サービスロールは、Amazon EC2 インスタンスにアクセスして Amazon EC2 インスタンスタグを拡張 (読み込み) するためのアクセス権限を CodeDeploy に付与します。

すでに <u>サービスロールの作成 (CLI)</u> の手順に従ってサービスロールを作成している必要があり ます。サービスロールの ARN を取得するには、「<u>サービスロール ARN の取得 (CLI)</u>」を参照 してください。

 サービスロールの ARN を取得したら、create-deployment-group コマンドを呼び出し、WordPress_DepGroup という名前の Amazon EC2 タグと WordPress_App という名前の デプロイ設定を使用して、CodeDeployDemo という名前のアプリケーションと関連付けられる CodeDeployDefault.OneAtATime という名前のデプロイグループを作成します。

- --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE $\$
- --service-role-arn serviceRoleARN

Note

<u>デプロイメントグループの作成</u>コマンドは、デプロイおよびインスタンス内の指定され たイベントについて、トピックサブスクライバーに Amazon SNS 通知を送信するトリ ガーの作成に対応しています。このコマンドは、Amazon CloudWatch アラームのモニ タリングしきい値が満たされたときにデプロイを自動的にロールバックし、デプロイを 停止するアラームを設定するオプションもサポートします。このチュートリアルでは、 これらのアクションに関するコマンドは含まれていません。 デプロイを作成する前に、デプロイグループのインスタンスに CodeDeploy エージェントがインストールされている必要があります。 AWS Systems Manager で次のコマンドを使用して、コマンドラインからエージェントをインストールできます。

aws ssm create-association \setminus
name AWS-ConfigureAWSPackage \
targets Key=tag:Name,Values=CodeDeployDemo \
parameters action=Install,name=AWSCodeDeployAgent \
schedule-expression "cron(0 2 ? * SUN *)"

このコマンドは、CodeDeploy エージェントをインストールし、毎週日曜日の午前 2:00 に更 新を試行する関連付けを Systems Manager ステートマネージャーに作成します。CodeDeploy エージェントの詳細については、「<u>CodeDeploy エージェントの使用</u>」を参照してくださ い。Systems Manager のさらなる詳細については、「<u>AWS Systems Managerとは</u>」を参照し てください。

 次に、create-deployment コマンドを呼び出して、amzn-s3-demo-bucket という名前 のバケットで WordPressApp.zip という名前のアプリケーションバージョンを使用し て、WordPress_App という名前のアプリケーション、CodeDeployDefault.OneAtATime という名前のデプロイ設定と WordPress_DepGroup という名前のデプロイグループに関連付 けられるデプロイを作成します。

aws deploy create-deployment \
 --application-name WordPress_App \
 --deployment-config-name CodeDeployDefault.OneAtATime \
 --deployment-group-name WordPress_DepGroup \
 --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip

アプリケーションリビジョン (コンソール) のデプロイ

 CodeDeploy コンソールを使用してアプリケーションリビジョンをデプロイする前に、サービ スロール ARN が必要になります。サービスロールは、ユーザーに代わってサービスアクセス 権限を付与する IAM ロールです。この場合、サービスロールは、Amazon EC2 インスタンス にアクセスして Amazon EC2 インスタンスタグを拡張 (読み込み) するためのアクセス権限を CodeDeploy に付与します。

すでに <u>サービスロールの作成 (コンソール)</u> の手順に従ってサービスロールを作成している必要 があります。サービスロールの ARN を取得するには、「<u>サービスロール ARN の取得 (コンソー</u> <u>ル)</u>」を参照してください。 ARN があるので、CodeDeploy コンソールを使用して、アプリケーションリビジョンをデプロ イします。

にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 3. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 4. アプリケーションのリストで、WordPress_Appを選択します。
- 5. [デプロイグループ] タブで、[デプロイグループの作成] を選択します。
- 6. [Deployment group name] (デプロイグループ名) に「WordPress_DepGroup」と入力します。
- 7. [Deployment type] で、[In-place deployment] を選択します。
- 8. [環境設定] で、[Amazon EC2 インスタンス] を選択します。
- 9. を使用した エージェント設定 AWS Systems Managerでは、デフォルトのままにします。
- 10. [Key] (キー) に、「Name」と入力します。
- 11. [値] には「CodeDeployDemo」と入力します。

Note

CodeDeployDemo と入力すると、[1] が [Matching instances] の下に表示さ

れ、CodeDeploy が一致する Amazon EC2 インスタンスを 1 つ見つけたことを確認しま す。

- 12. [デプロイ設定] で [CodeDeployDefault.OneAtATime] を選択します。
- 13. [サービスロール ARN] でサービスロール ARN を選択し、[デプロイグループの作成] を選択しま す。
- 14. [デプロイの作成]を選択します。
- 15. [デプロイグループ] で [WordPress_DepGroup] を選択します。
- 16. [Repository type] の横の [My application is stored in Amazon S3] を選択します。[リビジョンの 場所] で、前に Amazon S3 にアップロードしたサンプルアプリケーション WordPress のリビ ジョンの場所を入力します。場所の取得
 - a. <u>https://console.aws.amazon.com/s3/</u> で Amazon S3 コンソールを開きます。

- b. バケットのリストで、amzn-s3-demo-bucket (またはアプリケーションリビジョンをアップ ロードしたバケットの名前)を選択します。
- c. オブジェクトのリストで、WordPressApp.zip を選択します。
- d. [概要] タブで、[リンク] フィールドの値をクリップボードにコピーします。

次のように表示されます。

https://s3.amazonaws.com/amzn-s3-demo-bucket/WordPressApp.zip

- e. CodeDeploy コンソールに戻り、[リビジョンの場所] に [リンク] フィールドの値を貼り付け ます。
- 17. [ファイルタイプ] リストで、ファイルの種類を検出できないというメッセージが表示される場合 は、[.zip] を選択します。
- 18. (オプション) [Deployment description] ボックスにコメントを入力します。
- 19. [Deployment group overrides (デプロイグループの上書き)] を展開し、[デプロイ設定] から [CodeDeployDefault.OneAtATime] を選択します。
- 20. デプロイの開始 を選択します。新しく作成されたデプロイに関する情報は [Deployments] ページに表示されます。

デプロイをモニタリングおよびトラブルシューティングします。

AWS CLI または コンソールを使用して、デプロイをモニタリングおよびトラブルシューティングします。

トピック

- デプロイ (CLI) をモニタリングおよびトラブルシューティングするには
- デプロイ (コンソール) をモニタリングおよびトラブルシューティングするには

デプロイ (CLI) をモニタリングおよびトラブルシューティングするには

1. WordPress_App という名前のアプリケーションと WordPress_DepGroup という名前のデプ ロイグループに対して list-deployments コマンドを呼び出して、デプロイの ID を取得します。

aws deploy list-deployments --application-name WordPress_App --deployment-groupname WordPress_DepGroup --query 'deployments' --output text

2. デプロイ ID を使用して get-deployment コマンドを呼び出します。

aws deploy get-deployment --deployment-id deploymentID --query
'deploymentInfo.status' --output text

3. コマンドはデプロイの全体ステータスを返します。成功すると、値は Succeeded になります。

全体的なステータスが Failed の場合、<u>list-deployment-instances</u> や <u>デプロイメントインスタ</u> <u>ンスの取得</u> などのコマンドを呼び出してトラブルシューティングを行います。トラブルシュー ティングの他のオプションについては、「<u>ログファイルの分析によるインスタンスでのデプロイ</u> の失敗の調査」を参照してください。

デプロイ (コンソール) をモニタリングおよびトラブルシューティングするには

CodeDeploy コンソール の [Deployments] ページで、[Status] 列でデプロイのステータスをモニタリ ングできます。

特に [Status] 列の値が [Succeeded] 以外の値である場合にデプロイに関する詳細情報を取得するに は。

- [デプロイ] テーブルで、デプロイの名前を選択します。デプロイが失敗すると、失敗の原因を説 明するメッセージが表示されます。
- [インスタンスアクティビティ] に、デプロイに関する詳細情報が表示されます。デプロイ失敗 後、デプロイが失敗した Amazon EC2 インスタンスおよびステップを特定できる場合がありま す。
- より多くのトラブルシューティングを行う場合、「<u>View Instance Details</u>」で説明されているような手法を使用できます。また、Amazon EC2 インスタンスでデプロイログファイルを分析できます。詳細については、「<u>ログファイルの分析によるインスタンスでのデプロイの失敗の調査</u>」を参照してください。

デプロイの確認

デプロイが成功したら、WordPress インストールが動作していることを確認します。Amazon EC2 インスタンスのパブリック DNS アドレスの後に /WordPress を使用して、ウェブブラウザのサイ トを表示します。(Amazon EC2 コンソールでパブリック DNS 値を取得するには、Amazon EC2 イ ンスタンスを選択して [説明] タブで [パブリック DNS] で値を探します。)

例えば、Amazon EC2 インスタンスのパブリック DNS アドレスが ec2-01-234-567-890.compute-1.amazonaws.com である場合、次の URL を使用します。

http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress

ブラウザでサイトを表示すると、次のような WordPress のウェルカムページが表示されます。

We kno	lcome to WordPress. Before getting started, we need some information on the database. You will need to ow the following items before proceeding.
1.	Database name
2.	Database username
3.	Database password
4.	Database host
5.	Table prefix (if you want to run more than one WordPress in a single database)
We	're going to use this information to create a wp-config.php file. If for any reason this automatic file
cre	ation doesn't work, don't worry. All this does is fill in the database information to a configuration file.
Yo	u may also simply open wp-config-sample.php in a text editor, fill in your information, and save it as
wp	-config.php. Need more help? <u>We got it</u> .
ln a	all likelihood, these items were supplied to you by your Web Host. If you don't have this information, then you
wil	l need to contact them before you can continue. If you're all ready
L	et's go!

Amazon EC2 インスタンスで、HTTP インバウンドルールがそのセキュリティグループに追加され ていない場合、WordPress ウェルカムページは表示されません。リモートサーバーが応答していな いというメッセージが表示された場合は、Amazon EC2 インスタンスのセキュリティグループにイ ンバウンドルールがあることを確認します。詳細については、「<u>HTTPトラフィックを許可するイ</u> <u>ンバウンドルールの Amazon Linux または RHEL Amazon EC2 インスタンスへの追加</u>」を参照して ください。

ステップ 5: WordPress アプリケーションを更新して再デプロイする

正常にアプリケーションリビジョンをデプロイしたので、開発用マシンで WordPress コードを更 新し、 CodeDeploy を使用して、サイトを再デプロイします。後で、Amazon EC2 インスタンスの コード変更を確認する必要があります。

トピック

- WordPress サイトの設定
- サイトの変更
- サイトの再デプロイ

WordPress サイトの設定

コード変更の効果を表示するには、完全に機能するインストールができるように、WordPress サイトの設定を終了します。

- ウェブブラウザにサイトの URL を入力します。URL は Amazon EC2 インスタンスと /WordPress 拡張子のパブリック DNS アドレスです。この WordPress サイト例 (Amazon EC2 インスタンスのパブリック DNS アドレスの例)では、URL は http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress です。
- サイトをまだ設定していない場合は、WordPressのデフォルトのウェルカムページが表示され ます。[始めましょう]を選択します。
- デフォルトの MySQL データベースを使用するため、データベース設定ページで、以下の値を入 力します。
 - データベース名: test
 - ユーザー名: root
 - ・パスワード:空白のままにします。
 - データーベースホスト: localhost
 - ・ テーブルプレフィックス: wp_

[Submit] を選択して、データベースをセットアップします。

 サイト設定を続行します。[Welcome] ページで任意の値を入力して [Install WordPress] を選択し ます。インストールが完了したら、ダッシュボードにサインインできます。

A Important

WordPress アプリケーションのデプロイ中に、**change_permissions.sh** スクリプトにより /tmp/WordPress フォルダのアクセス権限が更新されたため、誰でもこのフォルダに書 き込むことができます。ここで、次のコマンドを実行して、所有者のみが書き込めるように アクセス権限を制限します。

chmod -R 755 /var/www/html/WordPress

サイトの変更

WordPress サイトを変更するには、開発用マシンのアプリケーションのフォルダを参照してください。

cd /tmp/WordPress

サイトの色の一部を変更するには、wp-content/themes/twentyfifteen/style.css ファイル で、テキストエディターまたは sed を使用して、#fff を #768331 に変更します。

Linux または、GNU sed を持つほかのシステムでは、以下を使用します。

sed -i 's/#ff/#768331/g' wp-content/themes/twentyfifteen/style.css

macOS、Unix、または BSD sed を持つ他のシステムでは、以下を使用します。

sed -i '' 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css

サイトの再デプロイ

サイトのコードを変更したので、Amazon S3 および CodeDeploy を使用してサイトを再デプロイし ます。

「<u>アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブファイルを</u> <u>プッシュする</u>」の説明に従って、変更内容をバンドルして Amazon S3 にアップロードします。(こ れらの手順に従うときに、アプリケーションを作成する必要がないことに注意してください。) 新 しいリビジョンに以前と同じキーを指定します (WordPressApp.zip)。それを先に作成した同じ Amazon S3 バケットにアップロードします (例: amzn-s3-demo-bucket)。

サイトを再デプロイするには AWS CLI、、CodeDeploy コンソール、または CodeDeploy APIs を使 用します。

トピック

- サイト (CLI) に再デプロイするには
- サイト (コンソール) の再デプロイ

サイト (CLI) に再デプロイするには

create-deployment コマンドを呼び出して、新しくアップロードされたリビジョンに基づいたデ プロイを作成します。amzn-s3-demo-bucket という名前のバケットにある WordPress_App という名前のアプリケーション、CodeDeployDefault.OneAtATime という名前のデプロイ設 定、WordPress_DepGroup という名前のデプロイグループ、WordPressApp.zip という名前のリ ビジョンを使用します。

aws deploy create-deployment \setminus

- --application-name WordPress_App \
- --deployment-config-name CodeDeployDefault.OneAtATime \
- --deployment-group-name WordPress_DepGroup \
- --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip

「<u>デプロイをモニタリングおよびトラブルシューティングします。</u>」に説明されているように、デプ ロイのステータスを確認できます。

CodeDeploy がサイトを再デプロイしたら、ウェブブラウザのサイトに再度アクセスして、色が変更 されたことを確認します。(ブラウザを更新することが必要な場合があります。) 色が変更されていた 場合は、サイトは正常に変更され、再デプロイされています。

サイト (コンソール) の再デプロイ

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

1 Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。
- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. アプリケーションのリストで、WordPress_Appを選択します。
- 4. [デプロイグループ] タブで、[WordPress_DepGroup] を選択します。
- 5. [デプロイの作成]を選択します。
- 6. [Create deployment] ページの
 - a. [デプロイグループ] で [WordPress_DepGroup] を選択します。
 - b. [リポジトリタイプ] エリアで、[My application is stored in (アプリケーションは Amazon S3 に格納されています)] を選択し、リビジョンの Amazon S3 リンクを [リビジョンの場所] ボックスにコピーします。リンク値の確認
 - i. 別のブラウザタブで

にサインイン AWS Management Console し、Amazon S3 コンソールを <u>https://</u> <u>console.aws.amazon.com/s3/</u>://https//https//h

amzn-s3-demo-bucket を参照して開き、リビジョン を選択しま すWordPressApp.zip。

- ii. [Properties] ペインが Amazon S3 コンソールに表示されない場合、[Properties] ボタン を選択します。
- iii. [Properties] ペインで、[Link] フィールドの値をCodeDeploy コンソールの [Revision location] ボックスにコピーします。
- c. ファイルの種類を検出できないというメッセージが表示される場合は、[.zip] を選択しま す。
- d. [Deployment description] ボックスを空白のままにしておきます。
- e. [Deployment group overrides (デプロイグループの上書き)] を展開し、[デプロイ設定] から [CodeDeployDefault.OneAtATime] を選択します。
- f. デプロイの開始 を選択します。新しく作成されたデプロイに関する情報は [Deployments] ページに表示されます。
- g. 「<u>デプロイをモニタリングおよびトラブルシューティングします。</u>」に説明されているよう に、デプロイのステータスを確認できます。

CodeDeploy がサイトを再デプロイしたら、ウェブブラウザのサイトに再度アクセスして、 色が変更されたことを確認します。(ブラウザを更新することが必要な場合があります。) 色 が変更されていた場合は、サイトは正常に変更され、再デプロイされています。

ステップ 6: WordPress のアプリケーションと関連リソースのクリーンアッ プ

これで、WordPress コードを正常に更新し、サイトを再デプロイしました。このチュートリアル用 に作成したリソースの継続的な料金の発生を回避するため、以下を削除する必要があります。

- AWS CloudFormation スタック (または、の外部で作成した場合は Amazon EC2 インスタンスを 終了 AWS CloudFormation)。
- Amazon S3 バケットの場合。
- CodeDeploy 内の WordPress_App アプリケーションの名前です。
- CodeDeploy エージェントの AWS Systems Manager ステートマネージャーの関連付け。

クリーンアップを実行するには AWS CLI、、 AWS CloudFormation、Amazon S3、Amazon EC2、CodeDeploy コンソール、または AWS APIsを使用できます。

トピック

- リソース (CLI) をクリーンアップするには
- リソース (コンソール) をクリーンアップするには
- 次のステップ

リソース (CLI) をクリーンアップするには

 このチュートリアルで AWS CloudFormation テンプレートを使用した場合は、 という名前のス タックに対して delete-stack コマンドを呼び出しますCodeDeployDemoStack。これにより、 付随するすべての Amazon EC2 インスタンスが終了し、スタックによって作成された付随する すべての IAM ロールが削除されます。

aws cloudformation delete-stack --stack-name CodeDeployDemoStack

Amazon S3 バケットを削除するには、rm スイッチを使用して --recursive という名前のバケットに対して amzn-s3-demo-bucket コマンドを呼び出します。これにより、バケットとバケット内のすべてのオブジェクトが削除されます。

aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region

 WordPress_App アプリケーションを削除するには、delete-application コマンドを呼び出しま す。これにより、関連するすべてのデプロイグループレコードと、アプリケーションのデプロイ レコードも削除されます。

aws deploy delete-application --application-name WordPress_App

Systems Manager ステートマネージャーの関連付けを削除する場合、delete-association コマンドを呼び出します。

aws ssm delete-association --assocation-id association-id

describe-association コマンドを呼び出して、######## ID を取得することができます。

aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
Key=tag:Name,Values=CodeDeployDemo

このチュートリアルで AWS CloudFormation スタックを使用しなかった場合は、 terminateinstances コマンドを呼び出して、手動で作成した Amazon EC2 インスタンスをすべて終了します。 終了させる Amazon EC2 インスタンスの ID を指定します。

aws ec2 terminate-instances --instance-ids instanceId

リソース (コンソール) をクリーンアップするには

このチュートリアルで AWS CloudFormation テンプレートを使用した場合は、関連する AWS CloudFormation スタックを削除します。

- 1. にサインイン AWS Management Console し、 AWS CloudFormation コンソールを <u>https://</u> console.aws.amazon.com/cloudformation://www.com で開きます。
- フィルターボックスに、前に作成した AWS CloudFormation スタック名を入力します (例: CodeDeployDemoStack)。
- 3. スタック名の横のボックスをオンにします。[Actions] メニューで、[Delete Stack] を選択します。

AWS CloudFormation はスタックを削除し、付随するすべての Amazon EC2 インスタンスを終 了し、付随するすべての IAM ロールを削除します。 AWS CloudFormation スタックの外部で作成した Amazon EC2 インスタンスを終了するには:

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- 2. [INSTANCES] リストで、[Instances] を選択します。
- 3. 検索ボックスで、終了する Amazon EC2 インスタンス名 (例: **CodeDeployDemo**) を入力して Enter キーを押します。
- 4. Amazon EC2 インスタンスを選択します。
- 5. [Actions] メニューで [Instance State] をポイントし、[Terminate] を選択します。プロンプトが表示されたら、[Yes, Terminate] を選択します。

インスタンスごとにこれらの手順を繰り返します。

Amazon S3 バケットの削除

- 1. にサインイン AWS Management Console し、Amazon S3 コンソールを <u>https://</u> console.aws.amazon.com/s3/://www.com で開きます。
- バケットのリストで、前に作成した Amazon S3 バケットの名前を参照して選択します (例: amzn-s3-demo-bucket)。
- バケットを削除する前に、まず、そのコンテンツを削除する必要があります。WordPressApp.zipのようなバケット内のすべてのファイルを選択します。[Actions] メ ニューで、[Delete] を選択します。削除を確認するプロンプトが表示されたら、[OK] を選択しま す。
- バケットが空になると、バケットを削除できます。バケットのリストで、バケットの行 (バケット名ではなく)を選択します。[Delete bucket] を選択し、確認が求められたら [OK] を選択します。

CodeDeploy から WordPress_App アプリケーションの削除

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「<u>CodeDeploy の開始方法</u>」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. アプリケーションのリストで、WordPress_App を選択します。
- 4. [Application details] ページで、[Delete application] を選択します。
- 5. プロンプトが表示されたら、アプリケーションの名前を入力して削除することを確定し、[削除] を選択します。

Systems Manager ステートマネージャーの関連付けの削除

- 1. AWS Systems Manager コンソールを https://console.aws.amazon.com/systems-manager:// www.com で開きます。
- 2. ナビゲーションペインで、[ステートマネージャー] を選択してください。
- 3. 作成した関連付けを選択し、[削除]を選択します。

次のステップ

ここまでの作業で、CodeDeploy デプロイが正常に完了し、サイトのコードが更新され、再デプロイ されました。

チュートリアル: 「Hello, World!」 CodeDeploy を使用したアプリ ケーション (Windows Server)

このチュートリアルでは、ウェブサーバーとしてインターネットインフォメーションサービス (IIS) を実行している単一の Windows Server の Amazon EC2 インスタンスに 1 つのウェブページをデプ ロイします。このウェブページには、「Hello, World!」というシンプルなメッセージが表示されま す。メッセージ

お探しのものではありませんか。

- 代わりに Amazon Linux または Red Hat Enterprise Linux (RHEL) Amazon EC2 インスタンスへの デプロイを練習する場合、<u>チュートリアル: Amazon EC2 インスタンスに WordPress をデプロイ</u> <u>する (Amazon Linux または Red Hat エンタープライズ Linux および Linux、macOS、または Unix)</u> を参照してください。
- 代わりにオンプレミスインスタンスへのデプロイの演習を行う場合は、「<u>チュートリアル:</u> <u>CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エンタープライズ Linux) を</u> 使用してオンプレミスインスタンスにアプリケーションをデプロイします。」を参照してください。

このチュートリアルのステップは、Windows の使用を前提としています。これらのステップのほと んどは Linux、macOS、Unix を実行しているローカルマシンでも完了できますが、c:\temp など の Windows ベースのディレクトリパスを扱うステップでは、対応するパスを使用する必要があり ます。また、Amazon EC2 インスタンスに接続する場合は、Remote Desktop Protocol (RDP) 経由 で、Windows Server を実行する Amazon EC2 インスタンスに接続できるクライアントアプリケー ションが必要です。(Windows にはデフォルトで、RDP 接続クライアントアプリケーションが搭載 されています)。

このチュートリアルを開始する前に、ユーザーの設定<u>CodeDeploy の開始方法</u>、 のインストールま たはアップグレード、IAM インスタンスプロファイルとサービスロールの作成など AWS CLI、 の前 提条件を満たす必要があります。

トピック

- ステップ 1: Windows Server の Amazon EC2 インスタンスを起動する
- ステップ 2: Windows Server の Amazon EC2 instance インスタンスにデプロイするソースコンテ ンツを設定する
- ステップ 3: 「Hello, World!」をアップロードする Amazon S3 へのアプリケーション
- ステップ 4: Hello World アプリケーションをデプロイする
- ステップ 5:「Hello World!」を更新およびデプロイする アプリケーション
- ステップ 6: 「Hello, World!」をクリーンアップする アプリケーションと関連リソース

ステップ 1: Windows Server の Amazon EC2 インスタンスを起動する

CodeDeploy で Hello World アプリケーションをデプロイする場合、Windows Server を実行している Amazon EC2 インスタンスが必要です。

「<u>CodeDeploy のための Amazon EC2 インスタンスを作成します。</u>」の手順に従います。Amazon EC2 インスタンスタグをインスタンスに割り当てる準備ができたら、必ず Name のタグキー と、**CodeDeployDemo** のタグ値を指定します。(別のタグキーまたはタグ値を指定した場合、「<u>ス</u> <u>テップ 4: Hello World アプリケーションをデプロイする</u>」の手順で予期しない結果が生成される場合 があります)。

Amazon EC2 インスタンスを起動後、このページに戻り、次のセクションに進みます。次のステップとして、CodeDeploy でアプリケーションを作成する には進まないでください。

Amazon EC2 インスタンスに接続します。

Amazon EC2 インスタンスが起動した後、手順に従ってインスタンスに接続する演習をします。

Note

これらの手順では、Windows および Windows Desktop Connection クライアントアプリ ケーションを実行していることを前提としています。詳細については、「<u>RDP を使用して</u> <u>Windows インスタンスに接続する</u>」を参照してください。他のオペレーティングシステムま たは他の RDP 接続クライアントアプリケーションに、これらの手順を適用する必要が生じ る場合もあります。

- 1. にサインイン AWS Management Console し、「https://<u>https://console.aws.amazon.com/</u> ec2/.com」で Amazon EC2 コンソールを開きます。
- ナビゲーションペインの [Instances] (インスタンス) で、[Instances] (インスタンス) を選択します。
- 3. 一覧で Windows Server インスタンスを参照して選択します。
- 4. [接続]を選択してください。
- 5. [パスワードの取得]、[ファイルの選択]の順に選択します。
- 6. Windows Server の Amazon EC2 インスタンスと関連付けられた Amazon EC2 インスタンス キーペアファイルを参照して選択し、[オープン] を選択します。
- [Decrypt Password] (パスワードを復号化) を選択します。表示されるパスワードをメモしておき ます。これはステップ 10 で必要になります。
- 8. [Download Remote Desktop File] を選択し、ファイルを開きます。
- 9. リモート接続の発行元を特定できなくても接続を求められる場合は、続行します。
- 10. ステップ 7 でメモしておいたパスワードを入力し、次に進みます (RDP 接続クライアントアプ リケーションでユーザー名が求められた場合は、Administrator と入力します)。
- 11. リモートコンピュータの ID を確認できなくても接続を求められた場合は、続行します。
- 12. 接続後、Windows Server を実行している Amazon EC2 インスタンスのデスクトップが表示され ます。
- 13. これで、Amazon EC2 インスタンスから切断することができます。

Marning

インスタンスを停止または削除しないでください。それ以外の場合は、CodeDeploy を デプロイすることはできません。

ステップ 1: Amazon EC2 インスタンスを起動する

Windows Server の Amazon EC2 インスタンスへの HTTP トラフィックを許可するインバウンドルールを追加する

次のステップでは、Windows Server の Amazon EC2 インスタンスにデプロイされたホームページが ブラウザに表示されるように、Amazon EC2 インスタンスに開いている HTTP ポートがあることを 確認します。

- 1. にサインイン AWS Management Console し、「https://<u>https://console.aws.amazon.com/</u> ec2/.com」で Amazon EC2 コンソールを開きます。
- 2. インスタンスを選択後、ご自分のインスタンスを選択します。
- 3. セキュリティグループ内の説明タブで、インバウンドのルールの表示を選択します。

セキュリティグループのルールの一覧は、次のように表示されます。

S	Security Groups associated with i-1234567890abcdef0			
	Ports	Protocol	Source	launch-wizard- <mark>N</mark>
	22	tcp	0.0.0.0/0	#

セキュリティグループでは、Amazon EC2 インスタンスのセキュリティグループを選択します。launch-wizard-N と名前が付けられる可能性があります。N は、インスタンスの作成時にセキュリティグループに割り当てられた名前です。

[Inbound] (インバウンド) タブを選択します。次の値を持つルールが表示された場合、ご利用の インスタンスのセキュリティグループは正しく設定されています。

- [Type]: HTTP
- [Protocol]: TCP
- [Port Range]: 80
- ・ ソース: 0.0.0.0/0
- これらの値を持つルールが表示されない場合は、セキュリティグループへのルールの追加の手順を使用して、新しいセキュリティルールに追加します。

ステップ 2: Windows Server の Amazon EC2 instance インスタンスにデプ ロイするソースコンテンツを設定する

ここでは、アプリケーションのソースコンテンツを設定して、Amazon EC2 インスタンスにデプ ロイできるものを準備します。このチュートリアルでは、 Windows Server を実行する Amazon EC2 インスタンスに 1 つのウェブページをデプロイします。これはウェブサーバーとして Internet Information Services (IIS) を実行します。このウェブページには、「Hello, World!」というシンプル なメッセージが表示されます。メッセージ

トピック

- ウェブページの作成
- アプリケーションを実行するスクリプトの作成
- アプリケーション仕様ファイルの追加

ウェブページの作成

HelloWorldApp フォルダで c:\temp というサブディレクトリ (サブフォルダ) を作成し、そのフォルダに切り替えます。

mkdir c:\temp\HelloWorldApp
cd c:\temp\HelloWorldApp

```
Note
```

c:\temp という場所、または HelloWorldApp というサブフォルダ名を必ず使用する 必要はありません。別の場所またはサブフォルダ名を使用する場合は、必ずこのチュー トリアル全体で使用してください。

 テキストエディタを使用して、フォルダ内にファイルを作成します。ファイルを index.html と名付けます。

notepad index.html

3. 次の HTML コードをファイルに追加し、ファイルを保存します。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
        <title>Hello, World!</title>
        <style>
        body {
            color: #ffffff;
```

```
background-color: #0188cc;
     font-family: Arial, sans-serif;
     font-size:14px;
   }
  </style>
</head>
<body>
  <div align="center"><h1>Hello, World!</h1></div>
  <div align="center"><h2>You have successfully deployed an application using
CodeDeploy</h2></div>
  <div align="center">
    What to do next? Take a look through the <a href="https://aws.amazon.com/"
codedeploy">CodeDeploy Documentation</a>.
  </div>
</body>
</html>
```

アプリケーションを実行するスクリプトの作成

次に、ターゲットの Amazon EC2 インスタンスでウェブサーバーをセットアップするために CodeDeploy が使用するスクリプトを作成します。

index.html ファイルが保存されているのと同じサブフォルダで、テキストエディタを使用して別のファイルを作成します。ファイルを before-install.bat と名付けます。

notepad before-install.bat

2. 次のバッチスクリプトコードをファイルに追加し、ファイルを保存します。

REM Install Internet Information Server (IIS). c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Import-Module -Name ServerManager c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Install-WindowsFeature Web-Server

アプリケーション仕様ファイルの追加

次に、ウェブページとバッチスクリプトファイルに加えて、 アプリケーション指定のファイル (AppSpec ファイル) を追加します。AppSpec ファイルは、CodeDeploy によって次の方式で使用さ れる <u>YAML</u> ファイルです。

- アプリケーションリビジョンのソースファイルを、インスタンスの宛先にマッピングします。
- デプロイ中にインスタンスで実行するスクリプトを指定します。

AppSpec のファイル名は、appspec.yml とする必要があります。アプリケーションソースコード のルートフォルダに配置する必要があります。

 index.html および before-install.bat ファイルが保存されているのと同じサブフォルダ で、テキストエディタを使用して別のファイルを作成します。ファイルを appspec.yml と名 付けます。

notepad appspec.yml

2. 次の YAML コードをファイルに追加し、ファイルを保存します。

```
version: 0.0
os: windows
files:
   - source: \index.html
    destination: c:\inetpub\wwwroot
hooks:
   BeforeInstall:
        - location: \before-install.bat
        timeout: 900
```

CodeDeploy は、この AppSpec ファイルを使用してアプリケーションソースコードのルートフォ ルダにある index.html ファイルを、ターゲット Amazon EC2 インスタンスの c:\inetpub \wwwroot フォルダにコピーします。デプロイ中に、CodeDeploy は before-install.bat デプ ロイライフサイクルイベントの間にターゲット Amazon EC2 インスタンスで **BeforeInstall** バッ チスクリプトを実行します。このスクリプトの実行に 900 秒 (15 分) 以上かかる場合、CodeDeploy はデプロイを停止し、Amazon EC2 インスタンスへのデプロイを失敗とマークします。

これらの設定の詳細については、「<u>CodeDeploy AppSpec ファイルのリファレンス</u>」を参照してく ださい。

▲ Important

このファイルの項目間のスペースの場所と数は重要です。間隔が正しくない場合、 CodeDeploy はデバッグが困難な可能性のあるエラーを発生させます。詳細については、 「AppSpec ファイルの間隔」を参照してください。

ステップ 3: 「Hello, World!」をアップロードする Amazon S3 へのアプリ ケーション

ここでソースコンテンツを CodeDeploy がデプロイできる場所に準備してアップロードします。次 の手順では、Amazon S3 バケットをプロビジョニングしてバケット用のアプリケーションリビジョ ンのファイルを準備し、リビジョンのファイルをバンドルしてから、そのリビジョンをバケットに プッシュする方法を示します。

Note

このチュートリアルでは説明されていませんが、CodeDeploy を使用して GitHub リポジト リからインスタンスにアプリケーションをデプロイできます。詳細については、「<u>GitHub と</u> CodeDeploy との統合」を参照してください。

トピック

- Amazon S3 バケットをプロビジョニングします
- バケットのアプリケーションファイルを準備する
- アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブファイルを プッシュする

Amazon S3 バケットをプロビジョニングします

ストレージコンテナあるいは Amazon S3 バケット を作成、または既存のバケットを使用します。バ ケットにリビジョンをアップロードできること、およびデプロイで使用する Amazon EC2 インスタ ンスがバケットからリビジョンをダウンロードできることを確認します。

AWS CLI、Amazon S3 コンソール、または Amazon S3 APIs を使用してAmazon S3バケットを作成 できます。バケットを作成したら、バケットとその CodeDeploy ユーザーにアクセス許可を付与し ます。 Note

バケット名は、すべての AWS アカウントで Amazon S3 全体で一意である必要がありま す。amzn-s3-demo-bucket を使用できない場合、amzn-s3-demo-bucket の後にダッ シュと自分の名前のイニシャル、または他の一意な識別子など別のバケット名を試してく ださい。このチュートリアル全体で、バケット名を amzn-s3-demo-bucket に置き換えま す。

Amazon S3 バケットは、ターゲット Amazon EC2 インスタンスが起動されるのと同じ AWS リージョンに作成する必要があります。例えば、バケットを米国東部 (バージニア北部) リー ジョンで作成する場合、対象の Amazon EC2 インスタンスを米国東部 (バージニア北部) リージョンで起動する必要があります。

トピック

- Amazon S3 バケット (CLI) の作成
- Amazon S3 バケット (コンソール) の作成
- Amazon S3 バケットと AWS アカウントにアクセス許可を付与する

Amazon S3 バケット (CLI) の作成

mb コマンドを呼び出して、**amzn-s3-demo-bucket** という名前の Amazon S3 バケットを作成し ます。

aws s3 mb s3://amzn-s3-demo-bucket --region region

Amazon S3 バケット (コンソール) の作成

- 1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
- 2. Amazon S3 コンソールで [バケットの作成] を選択します。
- 3. [Bucket name] ボックスで、バケットの名前を入力します。
- 4. [Region] リストで、ターゲットリージョンを選択し、[Create] を選択します。

Amazon S3 バケットと AWS アカウントにアクセス許可を付与する

Amazon S3 バケットへのアップロードには、許可が必要です。Amazon S3 バケットポリシーで、これらのアクセス許可を指定できます。たとえば、次の Amazon S3 バケットポリシーでは、ワイルド

カード文字 (*) を使用すると、 AWS アカウント111122223333は という名前の Amazon S3 バケッ ト内の任意のディレクトリにファイルをアップロードできますamzn-s3-demo-bucket。

```
{
    "Statement": [
        {
             "Action": [
                 "s3:PutObject"
            ],
             "Effect": "Allow",
             "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
             "Principal": {
                 "AWS": [
                     "111122223333"
                 ]
            }
        }
    ]
}
```

AWS アカウント ID を表示するには、AWS 「アカウント ID の検索」を参照してください。

今は、Amazon S3 バケットが参加している各 Amazon EC2 インスタンスからのダウンロードリクエ ストを許可していることを確認するのに適した時期です。Amazon S3 バケットポリシーで、これを 指定できます。例えば、次の Amazon S3 バケットポリシーでは、ワイルドカード文字 (*)を使用す ると、ARN arn:aws:iam::444455556666:role/CodeDeployDemo を含む IAM インスタンスプ ロファイルがアタッチされた Amazon EC2 インスタンスが、amzn-s3-demo-bucket という名前 の Amazon S3 バケットの任意のディレクトリからファイルをダウンロードすることを許可します。

```
{
    "Statement": [
        {
            "Action": [
               "s3:Get*",
               "s3:List*"
        ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
               "Principal": {
                "AWS": [
                "AWS": [
                "arn:aws:iam::444455556666:role/CodeDeployDemo"
        ]
```

} }] }

Amazon S3 バケットポリシーを生成しアタッチする方法の詳細については、「<u>バケットポリシーの</u> 例」を参照してください。

ステップ 1: セットアップ で作成した CodeDeploy 管理ユーザーに Amazon S3 バケットにリビジョ ンをアップロードするアクセス許可が必要です。これを指定する 1 つの方法は、IAM ポリシーを使 用してユーザーのアクセス権限セットに追加するか、IAM ロール (ユーザーに引き受けを許可する) に追加することです。次の IAM ポリシーでは、ユーザーが amzn-s3-demo-bucket という名前の Amazon S3 バケット内の任意の場所でリビジョンをアップロードできるようにします。

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

IAM ポリシーの作成方法については、「IAM ユーザーガイド」の「<u>IAM ポリシーの作成</u>」を参照 してください。アクセス権限セットにポリシーを追加する方法については、「AWS IAM Identity Center ユーザーガイド」の「<u>アクセス権限セットを作成します。</u>」を参照してください。

バケットのアプリケーションファイルを準備する

ウェブページ、AppSpec ファイル、およびスクリプトが開発マシン上で次のように整理されている ことを確認します。

```
c:\
   |-- temp\
    |--HelloWorldApp\
    |-- appspec.yml
    |-- before-install.bat
    |-- index.html
```

アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブ ファイルをプッシュする

ファイルをアーカイブファイル (アプリケーションリビジョンとも呼ばれる) にバンドルします。

Note

バケットにオブジェクトを保存したり、バケットの内外にアプリケーションのリビジョンを 転送したりする場合に課金されることがあります。詳細については、「<u>Amazon S3 の料金</u>」 を参照してください。

1. 開発マシンで、ファイルが保存されたフォルダに切り替えます。

cd c:\temp\HelloWorldApp

Note

このフォルダに切替わらなければ、ファイルのバンドルは現在のフォルダで起動されま す。例えば、現在のフォルダが c:\temp ではなく c:\temp\HelloWorldApp である 場合、バンドルは、c:\temp サブフォルダ以上を含む可能性のある HelloWorldApp フォルダ内のファイルとサブフォルダから開始します。

 create-application コマンドを呼び出して、HelloWorld_App という名前の新しいアプリケー ションを CodeDeploy に登録します。

aws deploy create-application --application-name HelloWorld_App

 CodeDeploy び <u>push</u> コマンドを呼び出してファイルをまとめてバンドルし、Amazon S3 に リビジョンをアップロードし、アップロードされたリビジョンに関する情報を1つの操作で CodeDeploy に登録します。

aws deploy push --application-name HelloWorld_App --s3-location s3://amzn-s3-demobucket/HelloWorld_App.zip --ignore-hidden-files

このコマンドは、現在のディレクトリ (隠しファイルを除く) から、HelloWorld_App.zip という名前の 1 つのアーカイブファイルにファイルをバンドルし、リビジョンを amzn-s3**demo-bucket** バケットにアップロードし、CodeDeploy でアップロードしたリビジョンについ ての情報を登録します。

ステップ 4: Hello World アプリケーションをデプロイする

ここで、Amazon S3 にアップロードした Hello World サンプルアプリケーションのリビジョンをデ プロイします。 AWS CLI または CodeDeploy ンソールを使用して、リビジョンをデプロイし、デプ ロイのの進行状況をモニタリングします。アプリケーションリビジョンが正常にデプロイされた後 に、その結果を確認します。

トピック

- CodeDeploy を使用して、アプリケーションリビジョンをデプロイします。
- デプロイをモニタリングおよびトラブルシューティングします。
- デプロイの確認

CodeDeploy を使用して、アプリケーションリビジョンをデプロイします。

アプリケーションをデプロイするには、CLI またはコンソールを使用できます。

- トピック
- アプリケーションリビジョン (CLI) をデプロイするには
- アプリケーションリビジョン (コンソール) のデプロイ

アプリケーションリビジョン (CLI) をデプロイするには

 まず、デプロイにはデプロイグループが必要です。ただし、デプロイグループを作成する前に、 サービスロール ARN が必要です。サービスロールは、ユーザーに代わってサービスアクセス 権限を付与する IAM ロールです。この場合、サービスロールは、Amazon EC2 インスタンス にアクセスして Amazon EC2 インスタンスタグを拡張 (読み込み) するためのアクセス権限を CodeDeploy に付与します。

すでに <u>サービスロールの作成 (CLI)</u> の手順に従ってサービスロールを作成している必要があり ます。サービスロールの ARN を取得するには、「<u>サービスロール ARN の取得 (CLI)</u>」を参照 してください。

ARN を取得したら、create-deployment-group コマンドを呼び出し
 て、HelloWorld_DepGroup という名前のアプリケーションと関連付けられる

HelloWorld_App という名前のデプロイグループを作成し、CodeDeployDemo とい う名前の Amazon EC2 インスタンスタグと、サービスロールARNと関連付けられる CodeDeployDefault.OneAtATime という名前のデプロイ設定を使用します。

aws deploy create-deployment-group --application-name HelloWorld_App
--deployment-group-name HelloWorld_DepGroup --deploymentconfig-name CodeDeployDefault.OneAtATime --ec2-tag-filters
Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE --service-role-arn serviceRoleARN

Note

-<u>デプロイメントグループの作成</u>コマンドは、デプロイおよびインスタンス内の指定さ れたイベントについて、トピックサブスクライバーに Amazon SNS 通知を送信するト リガーの作成に対応しています。このコマンドは、Amazon CloudWatch アラームのモ ニタリングしきい値が満たされたときにデプロイを自動的にロールバックし、デプロ イを停止するアラームを設定するオプションもサポートします。このチュートリアルで は、これらのアクションに関するコマンドは含まれていません。

デプロイを作成する前に、デプロイグループのインスタンスに CodeDeploy エージェントがインストールされている必要があります。 AWS Systems Manager で次のコマンドを使用して、コマンドラインからエージェントをインストールできます。

aws ssm create-association --name AWS-ConfigureAWSPackage
 --targets Key=tag:Name,Values=CodeDeployDemo --parameters
 action=Install,name=AWSCodeDeployAgent --schedule-expression "cron(0 2 ? * SUN
 *)"

このコマンドは、CodeDeploy エージェントをインストールし、毎週日曜日の午前 2:00 に更 新を試行する関連付けを Systems Manager ステートマネージャーに作成します。CodeDeploy エージェントの詳細については、「<u>CodeDeploy エージェントの使用</u>」を参照してくださ い。Systems Manager のさらなる詳細については、「<u>AWS Systems Managerとは</u>」を参照し てください。

 次に、create-deployment コマンドを呼び出して、amzn-s3-demo-bucket という名前の バケットで HelloWorld_App.zip という名前のアプリケーションバージョンを使用し て、HelloWorld_App という名前のアプリケーション、CodeDeployDefault.OneAtATime という名前のデプロイ設定と HelloWorld_DepGroup という名前のデプロイグループに関連付 けられるデプロイを作成します。 aws deploy create-deployment --application-name HelloWorld_App --deployment-configname CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip

アプリケーションリビジョン (コンソール) のデプロイ

 CodeDeploy コンソールを使用してアプリケーションリビジョンをデプロイする前に、サービ スロール ARN が必要になります。サービスロールは、ユーザーに代わってサービスアクセス 権限を付与する IAM ロールです。この場合、サービスロールは、Amazon EC2 インスタンス にアクセスして Amazon EC2 インスタンスタグを拡張 (読み込み) するためのアクセス権限を CodeDeploy に付与します。

すでに <u>サービスロールの作成 (コンソール)</u> の手順に従ってサービスロールを作成している必要 があります。サービスロールの ARN を取得するには、「<u>サービスロール ARN の取得 (コンソー</u> ル)」を参照してください。

2. ARN があるため、CodeDeploy コンソールを使用して、アプリケーションリビジョンをデプロ イできます。

にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「<u>CodeDeploy の開始方法</u>」で設定したのと同じユーザーでサインインします。

- 3. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 4. [HelloWorld_App]を選択します。
- 5. [デプロイグループ] タブで、[デプロイグループの作成] を選択します。
- 6. [Deployment group name] (デプロイグループ名) に「HelloWorld_DepGroup」と入力しま す。
- 7. [サービスロール] で、サービスロールの名前を選択します。
- 8. [デプロイタイプ] で、[インプレース] を選択します。
- 9. [環境設定] で、[Amazon EC2 インスタンス] を選択します。
- 10. を使用した エージェント設定 AWS Systems Managerでは、デフォルトのままにします。
- 11. [Key] (キー) に、「Name」と入力します。

- 12. [値] には「CodeDeployDemo」と入力します。
- 13. [デプロイ設定] で [CodeDeployDefault.OneAtATime] を選択します。
- 14. [ロードバランサー] で、[Enable load balancing (ロードバランシングの有効化)] をオフにしま す。
- 15. デプロイグループの作成を選択します。
- 16. [デプロイの作成]を選択します。
- 17. [Deployment group] で、[HelloWorld_DepGroup] を選択します。
- 18. [リビジョンタイプ] では [アプリケーションは Amazon S3 に格納されています] を選択し、[リビ ジョンの場所] では以前に Amazon S3 にアップロードしたサンプルの Hello World アプリケー ションリビジョンの場所を入力します。場所の取得
 - a. https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
 - b. バケットのリストで、amzn-s3-demo-bucket (またはアプリケーションリビジョンをアップ ロードしたバケットの名前)を選択します。
 - c. オブジェクトのリストで、HelloWorld_App.zip を選択します。
 - d. [概要] タブで、[パスのコピー] を選択します。
 - e. CodeDeploy コンソールに戻り、[リビジョンの場所] に [リンク] フィールドの値を貼り付け ます。
- 19. [リビジョンファイルの種類] で、[.zip] を選択します。
- 20. (オプション) [デプロイの説明] にコメントを入力します。
- 21. [デプロイの作成] を選択します。新しく作成されたデプロイに関する情報は [Deployments] ペー ジに表示されます。
- デプロイをモニタリングおよびトラブルシューティングします。

AWS CLI または コンソールを使用して、デプロイをモニタリングおよびトラブルシューティングします。

トピック

- デプロイ (CLI) をモニタリングおよびトラブルシューティングするには
- デプロイ (コンソール) をモニタリングおよびトラブルシューティングするには

デプロイ (CLI) をモニタリングおよびトラブルシューティングするには

 HelloWorld_App という名前のアプリケーションと HelloWorld_DepGroup という名前の デプロイグループに対して list-deployments コマンドを呼び出して、デプロイの ID を取得しま す。

aws deploy list-deployments --application-name HelloWorld_App --deployment-groupname HelloWorld_DepGroup --query "deployments" --output text

2. デプロイ ID を使用して get-deployment コマンドを呼び出します。

aws deploy get-deployment --deployment-id deploymentID --query
"deploymentInfo.status" --output text

3. コマンドはデプロイの全体ステータスを返します。成功すると、値は Succeeded になります。

全体的なステータスが Failed の場合、<u>list-deployment-instances</u> や <u>デプロイメントインスタ</u> <u>ンスの取得</u> などのコマンドを呼び出してトラブルシューティングを行います。トラブルシュー ティングの他のオプションについては、「<u>ログファイルの分析によるインスタンスでのデプロイ</u> の失敗の調査」を参照してください。

デプロイ (コンソール) をモニタリングおよびトラブルシューティングするには

CodeDeploy コンソール の [Deployments] ページで、[Status] 列でデプロイのステータスをモニタリ ングできます。

特に [Status] 列の値が [Succeeded] 以外の値である場合にデプロイに関する詳細情報を取得するに は。

- [デプロイ] テーブルで、デプロイ ID を選択します。デプロイが失敗したら、失敗の原因を説明 するメッセージがデプロイの詳細ページに表示されます。
- 2. インスタンスのデプロイに関する詳細情報が表示されます。デプロイ失敗後、デプロイが失敗した Amazon EC2 インスタンスおよびステップを特定できる場合があります。
- より多くのトラブルシューティングを行う場合、<u>View Instance Details</u>のような手法を使用でき ます。また、Amazon EC2 インスタンスでデプロイログファイルを分析できます。詳細につい ては、「<u>ログファイルの分析によるインスタンスでのデプロイの失敗の調査</u>」を参照してくださ い。

デプロイの確認

デプロイが成功したら、インストールが動作していることを確認します。Amazon EC2 インスタン スのパブリック DNS アドレスを使用して、ウェブブラウザのウェブページを表示します。(Amazon EC2 コンソールでパブリック DNS 値を取得するには、 Amazon EC2 インスタンスを選択して [説 明] タブで [パブリック DNS] で値を探します。)

例えば、Amazon EC2 インスタンスのパブリック DNS アドレスが ec2-01-234-567-890.compute-1.amazonaws.com である場合、次の URL を使用します。

http://ec2-01-234-567-890.compute-1.amazonaws.com

成功すると、Hello World ウェブページが表示されます。

ステップ 5:「Hello World!」を更新およびデプロイする アプリケーション

アプリケーションリビジョンを正常にデプロイしたので、開発マシンでウェブページのコードを更新 し、CodeDeploy を使用してサイトを再デプロイします。デプロイ後、Amazon EC2 インスタンスで 変更を確認できます。

トピック

- ウェブページの変更
- サイトの再デプロイ

ウェブページの変更

 c:\temp\HelloWorldApp サブフォルダに移動して、テキストエディタを使用して index.html ファイルを変更します。

cd c:\temp\HelloWorldApp
notepad index.html

index.html ファイルのコンテンツを改訂するために、ウェブページの背景色と一部のテキストを変更し、ファイルを保存します。

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/ TR/html4/loose.dtd"> <html>

```
ユーザーガイド
```

```
<head>
  <title>Hello Again, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #66cc00;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello Again, World!</h1></div>
  <div align="center"><h2>You have successfully deployed a revision of an
application using CodeDeploy</h2></div>
  <div align="center">
    What to do next? Take a look through the <a href="https://aws.amazon.com/">https://aws.amazon.com/</a>
codedeploy">CodeDeploy Documentation</a>.
  </div>
</body>
</html>
```

サイトの再デプロイ

コードを変更したので、Amazon S3 と CodeDeploy を使用して、ウェブページを再デプロイしま す。

「<u>アプリケーションのファイルを1つのアーカイブファイルにバンドルし、アーカイブファイルを</u> <u>プッシュする</u>」の説明に従って、変更内容をバンドルして Amazon S3 にアップロードします。(こ れらの手順に従うときに、新しいアプリケーションを作成する必要はありません。) このリビジョン に以前と同じキーを指定します (**HelloWorld_App.zip**)。それを先に作成した同じ Amazon S3 バ ケットにアップロードします (例: amzn-s3-demo-bucket)。

AWS CLI または CodeDeploy コンソールを使用してサイトを再デプロイします。

トピック

- サイト (CLI) に再デプロイするには
- サイト (コンソール)の再デプロイ

サイト (CLI) に再デプロイするには

create-deployment コマンドを呼び出して、amzn-s3-demo-bucket という名前のバケットにあ る、HelloWorld_App という名前のアプリケーション、CodeDeployDefault.OneAtATime という名前のデプロイ設定、HelloWorld_DepGroup という名前のデプロイグループ、および HelloWorld_App.zip という名前のリビジョンをそれぞれ再度使用して、アップロードしたリビ ジョンに基づくデプロイを作成します。

aws deploy create-deployment --application-name HelloWorld_App --deployment-configname CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip

「<u>デプロイをモニタリングおよびトラブルシューティングします。</u>」に説明されているように、新し いデプロイのステータスを確認できます。

CodeDeploy でサイトを再デプロイしたら、ウェブブラウザでサイトに再度アクセスし、ウェブペー ジの背景色とテキストが変更されていることを確認します。(ブラウザを更新することが必要な場合 があります。) 背景色とテキストが変更されていれば、これで完了です。サイトは変更され、再デプ ロイされています。

サイト (コンソール) の再デプロイ

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、[アプリケーション] を選択します。
- 3. [アプリケーション] リストで、[HelloWorld_App] を選択します。
- 4. [デプロイ] タブで、[デプロイの作成] を選択します。
 - a. [Deployment group] リストで、[HelloWorld_DepGroup] を選択します。
 - b. [リビジョンの場所] に、リビジョンの Amazon S3 リンクを入力します。

リンク値の確認

i. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/s3/</u>:// www.com」で Amazon S3 コンソールを開きます。 Amazon S33 コンソールで amzn-s3-demo-bucket を参照して開きHelloWorld_App.zip、リビジョン を選択します。

- ii. [Properties] ペインが Amazon S3 コンソールに表示されない場合、[Properties] ボタン を選択します。
- iii. [プロパティ] ペインで、[リンク] フィールドの値をコピーします
- iv. CodeDeploy コンソールに戻り、[リビジョンの場所] にリンクを貼り付けます。

v.

- c. [リビジョンファイルの種類] で、ファイルの種類を検出できないというメッセージが表示された場合は、[.zip] を選択します。
- d. [デプロイメントの説明]は空白のままにしておきます。
- e. [Deployment group overrides (デプロイグループのオーバーライド)] を展開し、[デプロイ設 定] リストで、[CodeDeployDefault.OneAtATime]、[デプロイの作成] の順に選択します。

「<u>デプロイをモニタリングおよびトラブルシューティングします。</u>」に説明されているよう に、デプロイのステータスを確認できます。

CodeDeploy でサイトを再デプロイしたら、ウェブブラウザでサイトに再度アクセスし、 ウェブページの背景色とテキストが変更されていることを確認します。(ブラウザを更新す ることが必要な場合があります。) 背景色とテキストが変更されていれば、これで完了で す。サイトは変更され、再デプロイされています。

ステップ 6: 「Hello, World!」をクリーンアップする アプリケーションと関 連リソース

これで「Hello, World!」コードを正常に更新しました。コードを記述し、サイトを再デプロイしま す。このチュートリアルを完了するために作成したリソースの継続的な料金の発生を回避するため、 以下を削除する必要があります。

- AWS CloudFormation スタック (または、の外部で作成した場合は Amazon EC2 インスタンスを 終了 AWS CloudFormation)。
- Amazon S3 バケットの場合。
- CodeDeploy 内の HelloWorld_App アプリケーションの名前です。
- CodeDeploy エージェントの AWS Systems Manager ステートマネージャーの関連付け。

クリーンアップを実行するには AWS CLI、、 AWS CloudFormation、Amazon S3、Amazon EC2、CodeDeploy コンソール、または AWS APIsを使用できます。

トピック

- クリーンアップリソース (CLI) の使用
- リソース (コンソール) をクリーンアップするには
- 次のステップ

クリーンアップリソース (CLI) の使用

 このチュートリアルで AWS CloudFormation スタックを使用した場合は、 という 名前のスタックに対して delete-stack コマンドを呼び出してスタックを削除しま すCodeDeployDemoStack。これにより、すべての付随する Amazon EC2 インスタンスが削除 され、スタックによって最初に作成されたすべての付随する IAM ロールが削除されます。

aws cloudformation delete-stack --stack-name CodeDeployDemoStack

Amazon S3 バケットを削除するには、rm スイッチを使用して --recursive という名前のバケットに対して amzn-s3-demo-bucket コマンドを呼び出します。これにより、バケットとバケット内のすべてのオブジェクトが削除されます。

aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region

 CodeDeploy から HelloWorld_App アプリケーションを削除する場合、delete-application コマ ンドを呼び出します。これにより、すべての関連するデプロイグループレコードと、アプリケー ションのデプロイレコードが削除されます。

aws deploy delete-application --application-name HelloWorld_App

4. Systems Manager ステートマネージャーの関連付けを削除する場合、delete-association コマンドを呼び出します。

aws ssm delete-association --assocation-id association-id

describe-association コマンドを呼び出して、######## ID を取得することができます。

aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
Key=tag:Name,Values=CodeDeployDemo

5. このチュートリアルで AWS CloudFormation スタックを使用しなかった場合は、 terminateinstances コマンドを呼び出して、手動で作成した Amazon EC2 インスタンスを終了します。終 了する Amazon EC2 インスタンスの ID を指定します。

aws ec2 terminate-instances --instance-ids instanceId

リソース (コンソール) をクリーンアップするには

このチュートリアルで AWS CloudFormation テンプレートを使用した場合は、関連する AWS CloudFormation スタックを削除します。

- 1. にサインイン AWS Management Console し、 AWS CloudFormation コンソールを <u>https://</u> console.aws.amazon.com/cloudformation://www.com で開きます。
- 検索ボックスに、AWS CloudFormation スタック名(などCodeDeployDemoStack)を入力し ます。
- 3. スタック名の横のチェックボックスをオンにします。
- 4. [Actions] メニューで、[Delete Stack] を選択します。これにより、スタックが削除され、すべての付随する Amazon EC2 インスタンスとすべての付随する IAM ロールも削除されます。

AWS CloudFormation スタックの外部で作成した Amazon EC2 インスタンスを終了するには:

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- 2. [Instances] エリアで、[Instances] を選択します。
- 3. 検索ボックスで、削除する Amazon EC2 インスタンスの名前を入力し、[Enter] キーを押します。
- 4. Amazon EC2 インスタンスを選択します。
- 5. [Actions] を選択して [Instance State] をポイントし、[Terminate] を選択します。プロンプトが表示されたら、[Yes, Terminate] を選択します。追加の Amazon EC2 インスタンスに対して、これらのステップを繰り返します。

Amazon S3 バケットの削除

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/s3/</u>:// www.com」で Amazon S3 コンソールを開きます。
- バケットのリストで、Amazon S3 バケットの名前 (amzn-s3-demo-bucket など)を参照して 選択します。
- バケットを削除する前に、まず、そのコンテンツを削除する必要がありま す。HelloWorld_App.zipのようなバケット内のすべてのファイルを選択します。[Actions] メニューで、[Delete]を選択します。削除を確認するプロンプトが表示されたら、[OK]を選択し ます。
- バケットが空になると、バケットを削除できます。バケットのリストで、バケットの行 (バケット名ではなく)を選択します。[Delete bucket] を選択し、確認が求められたら [OK] を選択します。

CodeDeploy から HelloWorld_App アプリケーションの削除

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. HelloWorld_App を選択します。
- 4. [アプリケーションを削除]を選択します。
- 5. 確認を求めるメッセージが表示されたら、Deleteと入力し、[削除]を選択してください。

Systems Manager ステートマネージャーの関連付けの削除。

- 1. AWS Systems Manager コンソールを https://console.aws.amazon.com/systems-manager:// www.com で開きます。
- 2. ナビゲーションペインで、[ステートマネージャー] を選択してください。
- 3. 作成した関連付けを選択し、[削除]を選択します。

次のステップ

ここまでの作業で、CodeDeployを使って正常にデプロイを完了しました。お疲れ様でした。

チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サー バー、または Red Hat エンタープライズ Linux) を使用してオンプ レミスインスタンスにアプリケーションをデプロイします。

このチュートリアルでは、Windows サーバー、Ubuntu サーバー、または Red Hat エンタープライ ズ Linux (RHEL) を実行する単一のオンプレミスインスタンス (つまりは、Amazon EC2 インスタン スではない物理デバイス) へのサンプルアプリケーションリビジョンのデプロイをガイドすること で、CodeDeploy の経験を得ることができます。オンプレミスインスタンスについての情報、および CodeDeploy の使用方法については、Working with On-Premises Instances を参照してください。

お探しのものではありませんか。

- Amazon Linux または RHEL を実行する Amazon EC2 インスタンスへのデプロイの演習を行う には、<u>チュートリアル: Amazon EC2 インスタンスに WordPress をデプロイする (Amazon Linux</u> <u>または Red Hat エンタープライズ Linux および Linux、macOS、または Unix</u>) を参照してくださ い。
- Windows サーバーを実行する Amazon EC2 インスタンスへのデプロイの演習を行うには、<u>チュートリアル:「Hello, World!」 CodeDeploy を使用したアプリケーション (Windows Server)</u>を参照してください。

トピック

- 前提条件
- ステップ 1: オンプレミスインスタンスを設定する
- ステップ 2: サンプルのアプリケーションリビジョンを作成する
- ステップ 3: アプリケーションリビジョンをバンドルし、Amazon S3 にアップロードする
- ステップ 4: アプリケーションリビジョンをデプロイする
- ステップ 5: デプロイを確認する
- ステップ 6: リソースをクリーンアップする

前提条件

このチュートリアルを開始する前に、ユーザーの設定<u>CodeDeploy の開始方法</u>、 のインストールま たはアップグレード、サービスロールの作成など AWS CLI、 の前提条件を満たす必要があります。 前提条件で説明したように、IAM インスタンスプロファイルを作成する必要はありません。オンプ レミスインスタンスは、IAM インスタンスプロファイルを使用しません。

オンプレミスインスタンスとして設定する物理デバイスでは、「<u>CodeDeploy エージェントで対応す</u> <u>るオペレーティングシステム</u>」に示したいずれかのオペレーティングシステムを実行している必要が あります。

ステップ 1: オンプレミスインスタンスを設定する

オンプレミスインスタンスにデプロイする前に、設定を行う必要があります。「<u>Working with On-</u> Premises Instances」の指示に従ってから、このページに戻ります。

CodeDeploy エージェントをインストールします。

オンプレミスインスタンスを設定したら、<u>CodeDeploy エージェントのインストール</u> のオンプレミ スインスタンスのステップに従い、このページに戻ります。

ステップ 2: サンプルのアプリケーションリビジョンを作成する

このステップでは、オンプレミスインスタンスにデプロイするサンプルのアプリケーションリビジョ ンを作成します。

オンプレミスインスタンス上に既にインストールされている、または組織のポリシーによってイン ストールが許可されているソフトウェアと機能を知るのは難しいため、オンプレミスインスタンス の場所にテキストファイルを書き込むために、ここで提供するサンプルアプリケーションリビジョ ンでは、バッチスクリプト (Windows Server の場合) またはシェルスクリプト (Ubuntu Server およ び RHEL の場合) を使用します。Install、 AfterInstall、 ApplicationStart および ValidateService を 含む、複数の CodeDeploy デプロイライフサイクルイベントごとに 1 つのファイルが書き込まれま す。BeforeInstall デプロイライフサイクルイベント中にスクリプトが実行され、このサンプルの前の デプロイ中に書き込まれた古いファイルを削除し、新しいファイルを書き込むオンプレミスインスタ ンス上に場所を作成します。

Note

以下のいずれも該当しない場合、このサンプルのアプリケーションリビジョンはデプロイに 失敗することがあります。

- オンプレミスインスタンスで CodeDeploy エージェントを起動するユーザーには、スクリ プトを実行する権限がありません。
- ユーザーに、スクリプトにリストされている場所でフォルダを作成または削除する権限が ない。
- ユーザーに、スクリプトにリストされている場所でテキストファイルを作成する権限がない。

Note

Windows Server インスタンスを設定し、別のサンプルをデプロイする場合は、<u>ステップ 2:</u> Windows Server の Amazon EC2 instance インスタンスにデプロイするソースコンテンツを 設定する チュートリアルの「チュートリアル: 「Hello, World!」 CodeDeploy を使用したア プリケーション (Windows Server)」のサンプルを使用することをお勧めします。 RHEL インスタンスを設定し、別のサンプルをデプロイする場合は、<u>ステップ 2: Amazon</u> Linux または Red Hat エンタープライズ Linux Amazon EC2 インスタンスにデプロイされる ようにソースコンテンツを設定する のチュートリアルの チュートリアル: Amazon EC2 イ ンスタンスに WordPress をデプロイする (Amazon Linux または Red Hat エンタープライズ Linux および Linux、macOS、または Unix) のサンプルを使用することをお勧めします。 現在、Ubuntu サーバー用の代替サンプルはありません。

 開発マシンで、サンプルのアプリケーションリビジョンのファイルを保存す る、CodeDeployDemo-OnPremという名前のサブディレクトリ (サブフォルダ)を作成し、 そのサブフォルダに切り替えます。この例では、c:\tempのフォルダを Windows サーバーの ルートフォルダとして使用するか、/tmpのフォルダを Ubuntu サーバーおよび RHEL のルート フォルダとして使用することを前提としています。別のフォルダを使用する場合は、このチュー トリアル全体でそのフォルダに置き換えてください。

Windows の場合:

mkdir c:\temp\CodeDeployDemo-OnPrem
cd c:\temp\CodeDeployDemo-OnPrem

Linux、macOS、Unix の場合:

```
mkdir /tmp/CodeDeployDemo-OnPrem
cd /tmp/CodeDeployDemo-OnPrem
```

2. CodeDeployDemo-OnPrem サブフォルダのルートで、テキストエディタを使用して appspec.yml および install.txt という 2 つのファイルを作成します。

```
Windows サーバーのための appspec.yml
```

```
version: 0.0
os: windows
files:
  - source: .\install.txt
    destination: c:\temp\CodeDeployExample
hooks:
  BeforeInstall:
    - location: .\scripts\before-install.bat
      timeout: 900
  AfterInstall:
    - location: .\scripts\after-install.bat
      timeout: 900
  ApplicationStart:
    - location: .\scripts\application-start.bat
      timeout: 900
  ValidateService:
    - location: .\scripts\validate-service.bat
      timeout: 900
```

Ubuntu ServerとRHELのための appspec.yml:

```
version: 0.0
os: linux
files:
    - source: ./install.txt
    destination: /tmp/CodeDeployExample
hooks:
    BeforeInstall:
        - location: ./scripts/before-install.sh
        timeout: 900
AfterInstall:
        - location: ./scripts/after-install.sh
        timeout: 900
```

```
ApplicationStart:
    - location: ./scripts/application-start.sh
    timeout: 900
ValidateService:
    - location: ./scripts/validate-service.sh
    timeout: 900
```

AppSpec ファイルの詳細については、「<u>CodeDeploy 用のアプリケーション仕様ファイルをリ</u> <u>ビジョンに追加</u>」および「<u>CodeDeploy AppSpec ファイルのリファレンス</u>」を参照してくださ い。

install.txt:

The Install deployment lifecycle event successfully completed.

 CodeDeployDemo-OnPrem サブフォルダのルートの下に、scripts サブフォルダを作成し、 そのサブフォルダに切り替えます。

Windows の場合:

mkdir c:\temp\CodeDeployDemo-OnPrem\scripts
cd c:\temp\CodeDeployDemo-OnPrem\scripts

Linux、macOS、Unix の場合:

mkdir -p /tmp/CodeDeployDemo-OnPrem/scripts
cd /tmp/CodeDeployDemo-OnPrem/scripts

 scripts のサブフォルダのルートで、テキストエディタを使用して、 Windows サーバー の場合は before-install.bat、after-install.bat、application-start.bat、 validate-service.bat、あるいは Ubuntu サーバーおよび RHEL の場合は、beforeinstall.sh、after-install.sh、application-start.sh、validate-service.sh という名前の4つのファイルを作成します。

Windows サーバーの場合

before-install.bat:

set FOLDER=%HOMEDRIVE%\temp\CodeDeployExample

```
if exist %FOLDER% (
  rd /s /q "%FOLDER%"
)
mkdir %FOLDER%
```

after-install.bat:

cd %HOMEDRIVE%\temp\CodeDeployExample

```
echo The AfterInstall deployment lifecycle event successfully completed. > after-
install.txt
```

application-start.bat:

cd %HOMEDRIVE%\temp\CodeDeployExample

```
echo The ApplicationStart deployment lifecycle event successfully completed. >
    application-start.txt
```

validate-service.bat:

```
cd %HOMEDRIVE%\temp\CodeDeployExample
```

```
echo The ValidateService deployment lifecycle event successfully completed. >
validate-service.txt
```

Ubuntu Server と RHEL の場合:

before-install.sh:

```
#!/bin/bash
export FOLDER=/tmp/CodeDeployExample
if [ -d $FOLDER ]
then
  rm -rf $FOLDER
fi
mkdir -p $FOLDER
```

after-install.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample
```

```
echo "The AfterInstall deployment lifecycle event successfully completed." > after-
install.txt
```

application-start.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample
```

```
echo "The ApplicationStart deployment lifecycle event successfully completed." >
    application-start.txt
```

validate-service.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample
```

echo "The ValidateService deployment lifecycle event successfully completed." >
 validate-service.txt

unset FOLDER

5. Ubuntu サーバーおよび RHEL の場合のみ、4 つのシェルスクリプトに実行権限があることを確認します。

chmod +x ./scripts/*

ステップ 3: アプリケーションリビジョンをバンドルし、Amazon S3 に アップロードする

アプリケーションリビジョンをデプロイする前に、ファイルをバンドルし、ファイルバンドルを Amazon S3 バケットにアップロードしておく必要があります。「<u>CodeDeploy でアプリケーション</u> <u>を作成する」</u>および「<u>Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オンプレミス</u> <u>のデプロイのみ)</u>」の手順に従います。(アプリケーションとデプロイグループには任意の名前を付け ることができますが、アプリケーション名に「CodeDeploy-OnPrem-App」、デプロイグループ名 に「CodeDeploy-OnPrem-DG」を使用することをお勧めします)。これらの手順を実行したら、こ のページに戻ります。

Note

または、GitHub リポジトリにファイルバンドルをアップロードし、そこからデプロイできま す。詳細については、「GitHub と CodeDeploy との統合」を参照してください。

ステップ 4: アプリケーションリビジョンをデプロイする

アプリケーションリビジョンを Amazon S3 バケットにアップロードしたら、オンプレミスインスタ ンスへのデプロイを試します。「<u>CodeDeploy でデプロイを作成する</u>」の指示に従ってから、この ページに戻ります。

ステップ 5: デプロイを確認する

デプロイが成功したことを確認するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」の手順に従 い、このページに戻ります。

デプロイが成功している場合は、c:\temp\CodeDeployExample のフォルダ (Windows の場合) または /tmp/CodeDeployExample (Ubuntu サーバーおよび RHEL の場合) に、 4 つのテキストフォ ルダが見つかります。

デプロイが失敗した場合は、「<u>View Instance Details</u>」および「<u>インスタンスの問題のトラブル</u> <u>シューティング</u>」のトラブルシューティングステップに従ってください。必要な修正を行い、アプリ ケーションリビジョンを再バンドルしてアップロードしてから、デプロイを再試行します。

ステップ 6: リソースをクリーンアップする

このチュートリアル用に作成したリソースの料金が継続的に発生しないようにするため、それ以上使 用しない場合は Amazon S3 バケットを削除します。CodeDeploy とオンプレミスインスタンス中の アプリケーションおよびデプロイグループレコードのような関連リソースを、クリーンアップするこ ともできます。

AWS CLI または CodeDeploy コンソールと Amazon S3 コンソールの組み合わせと を使用して AWS CLI 、リソースをクリーンアップできます。
リソースのクリーンアップ (CLI)

Amazon S3 バケットを削除するには

バケット (例: --recursive) に対して、amzn-s3-demo-bucket のスイッチを用いて rm コマンドを呼び出します。バケットとバケット内のすべてのオブジェクトが削除されます。

aws s3 rm s3://your-bucket-name --recursive --region region

CodeDeploy 中のアプリケーションとデプロイグループレコードを削除するには

 アプリケーションに対して <u>delete-application</u> コマンドを呼び出します (例: CodeDeploy-OnPrem-App)。デプロイおよびデプロイグループのレコードが削除されます。

aws deploy delete-application --application-name your-application-name

オンプレミスインスタンスを登録解除し、IAM ユーザーを削除するには

・ オンプレミスインスタンスとリージョンに対して deregister コマンドを呼び出します。

aws deploy deregister --instance-name *your-instance-name* --delete-iam-user -- region *your-region*

Note

このオンプレミスインスタンスに関連付けられた IAM ユーザーを削除しない場合は、代わりに --no-delete-iam-user のオプションを使用します。

CodeDeploy エージェントをアンインストールし、オンプレミスインスタンスから設定ファイルを削除するには

・ オンプレミスインスタンスから uninstall コマンドを呼び出します。

aws deploy uninstall

これで、このチュートリアルで使用したリソースをクリーンアップするすべてのステップが完了しま した。

リソースのクリーンアップ (コンソール)

Amazon S3 バケットを削除するには

- 1. にサインイン AWS Management Console し、Amazon S3 コンソールを <u>https://</u> console.aws.amazon.com/s3/://www.com で開きます。
- 2. 削除するバケットの横にあるアイコン (例: amzn-s3-demo-bucket)を選択します。ただし、 バケット自体を選択しないでください。
- 3. [アクション]を選択し、[削除]を選択します。
- 4. バケットを削除するように求められたら、[OK] を選択します。

CodeDeploy 中のアプリケーションとデプロイグループレコードを削除するには

 にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

1 Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、[アプリケーション]を選択します。
- 3. 削除するアプリケーションの名前 (CodeDeploy-OnPrem-App など)を選択し、[アプリケー ションの削除] を選択します。
- プロンプトが表示されたら、アプリケーションの名前を入力して削除することを確定し、[削除]
 を選択します。

AWS CodeDeploy コンソールを使用してオンプレミスインスタンスの登録を解除した り、CodeDeploy エージェントをアンインストールしたりすることはできません。「<u>オンプレミスイ</u> <u>ンスタンスを登録解除し、IAM ユーザーを削除するには</u>」の手順に従います。

チュートリアル: CodeDeploy を使用して、Auto Scaling グループ にアプリケーションをデプロイします。

このチュートリアルでは、CodeDeploy を使用して Auto Scaling グループにアプリケーション リビジョンをデプロイします。Amazon EC2 Auto Scaling は、事前定義された条件を使用して Amazon EC2 インスタンスを起動した後に、不要になった Amazon EC2 インスタンスを終了し ます。Amazon EC2 Auto Scaling は、デプロイメントの負荷を処理するために常に適切な数の Amazon EC2 インスタンスを利用できるようにすることで、CodeDeploy のスケーリングに役立ちま す。CodeDeploy による Amazon EC2 Auto Scaling 統合の詳細については、<u>CodeDeploy と Amazon</u> EC2 Auto Scaling の統合 を参照してください。

トピック

- 前提条件
- ステップ 1: Auto Scaling グループを作成して設定します。
- ステップ 2: Auto Scaling グループにアプリケーションをデプロイする
- ステップ 3: 結果の確認
- ステップ 4: Auto Scaling グループの Amazon EC2 インスタンスの数を増やす
- ステップ 5: 結果を再度確認します
- ステップ 6: クリーンアップする

前提条件

このチュートリアルを実行するには:

- のセットアップと設定CodeDeployの開始方法、IAM インスタンスプロファイル (CodeDeployDemo-EC2-Instance-Profile) AWS CLI とサービスロール ()の作成など、のす べてのステップを完了しますCodeDeployDemo。サービスロール は、ユーザーに代わってサービ スアクセス権限を付与する、特別なタイプの IAM ロールです。
- ・ 起動テンプレートを使用して Auto Scaling グループを作成する場合は、次の権限を追加する必要 があります。
 - ec2:RunInstances
 - ec2:CreateTags
 - iam:PassRole

チュートリアル: Auto Scaling グループヘデプロイします。

さらなる詳細については、<u>ステップ 2: サービスロールを作成する</u>、<u>Creating a launch template</u> <u>for an Auto Scaling group</u>、および Amazon EC2 Auto Scaling User Guide 中の <u>Launch template</u> support を参照してください。

- Ubuntu サーバーインスタンスおよび CodeDeploy と互換性のあるリビジョンを作成して使用します。リビジョンでは、次のいずれかを実行できます。
 - 「<u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エン</u> <u>タープライズ Linux) を使用してオンプレミスインスタンスにアプリケーションをデプロイしま</u> <u>す。</u>」チュートリアルの <u>ステップ 2: サンプルのアプリケーションリビジョンを作成する</u> のサン プルリビジョンを作成して使用します。
 - リビジョンを独自に作成するには、「<u>CodeDeploy のアプリケーションリビジョンの操作</u>」を参照してください。
- Inbound rule を用いて、CodeDeployDemo-AS-SG という名前のセキュリティグループを作成します。
 - Type: HTTP
 - ・ ソース:どこでも

これは、アプリケーションを表示し、デプロイメントの成功を確認するために必要です。セキュリ ティグループの作成方法については、Amazon EC2 user guide 中の <u>Creating a security group</u> を参 照してください。

ステップ 1: Auto Scaling グループを作成して設定します。

このステップでは、単一の Amazon Linux、RHEL、または Windows サーバーの Amazon EC2 イン スタンスを含む Auto Scaling グループを作成します。後のステップで、もう 1 つの Amazon EC2 イ ンスタンスを追加するように Amazon EC2 Auto Scaling に指示し、CodeDeploy はそれにリビジョ ンをデプロイします。

トピック

- Auto Scaling グループ (CLI) を作成して設定するには
- Auto Scaling グループ (コンソール) を作成して設定するには

Auto Scaling グループ (CLI) を作成して設定するには

1. create-launch-template コマンドを呼び出して、Amazon EC2 起動テンプレートを作成します。

このコマンドを呼び出す前に、プレースホルダー *image-id* で表される、このチュートリアル で使用する AMI の ID が必要です。プレースホルダー *key-name* で表される、 Amazon EC2 イ ンスタンスへのアクセスを有効にする Amazon EC2 インスタンスのキーペアの名前も必要で す。

このチュートリアルで使用する AMI の ID を取得するには。

- a. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
- b. ナビゲーションペインで、[Instances] の下にある、[Instances] を選択して、[Launch Instance] を選択します。
- c. Choose an Amazon Machine Image ページの Quick Start タブ上で、Amazon Linux 2 AMI、 Red Hat Enterprise Linux 7.1、Ubuntu Server 14.04 LTS、あるいは Microsoft Windows Server 2012 R2 の横にある AMI の ID をメモします。

Note

CodeDeploy と互換性がある AMI のカスタムバージョンの場合、クイックス タート タブを参照する代わりに、ここでそれを選択します。CodeDeploy および Amazon EC2 Auto Scaling でカスタム AMI を使用することについての詳細について は、<u>CodeDeploy と Amazon EC2 Auto Scaling でのカスタム AMI の使用</u> を参照し てください。

Amazon EC2 インスタンスのキーペアについては、Amazon EC2 インスタンスのキーペアの名 前を使用します。

create-launch-template コマンドを呼び出します。

ローカル Linux、macOS、あるいは Unix マシンについて

aws ec2 create-launch-template \
 --launch-template-name CodeDeployDemo-AS-Launch-Template \
 --launch-template-data file://config.json

config.json ファイルのコンテンツ:

"InstanceType":"t1.micro",

{

```
"ImageId":"image-id",
"IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
},
"KeyName":"key-name"
}
```

ローカル Windows マシンの場合

aws ec2 create-launch-template --launch-template-name CodeDeployDemo-AS-Launch-Template --launch-template-data file://config.json

config.json ファイルのコンテンツ:

```
{
   "InstanceType":"t1.micro",
   "ImageId":"image-id",
   "IamInstanceProfile":{
        "Name":"CodeDeployDemo-EC2-Instance-Profile"
    },
        "KeyName":"key-name"
}
```

これらのコマンドは、config.json ファイルとともに、Auto Scaling グループ用の CodeDeployDemo-as-Launch-Template という名前の Amazon EC2 起動テンプレートを作成し ます。このテンプレートは、t1.micro Amazon EC2 インスタンスタイプに基づいて次のステップ で作成されます。ImageId、IamInstanceProfile、KeyName への入力に基づいて、起動テ ンプレートでは AMI ID、起動時にインスタンスに渡す IAM ロールに関連付けられたインスタン スプロファイルの名前、およびインスタンスへの接続時に使用する Amazon EC2 キーペアも指 定します。

create-auto-scaling-group のコマンドをを呼び出して、Auto Scaling グループを作成します。AWS 全般のリファレンス の<u>リージョンとエンドポイント</u>に一覧表示されているリージョンの1つで、プレースホルダー availability-zone で表される、1つのアベイラビリティーゾーンの名前が必要になります。

Note

リージョンでアベイラビリティーゾーンのリストを表示するには、以下を呼び出しま す。 aws ec2 describe-availability-zones --region region-name

例えば、米国西部 (オレゴン) リージョンのアベイラビリティーゾーンのリストを表示す るには、次のように呼び出します。

aws ec2 describe-availability-zones --region us-west-2

リージョン名識別子のリストについては、「<u>リージョン別リソースキットバケット名</u>」 を参照してください。

ローカル Linux、macOS、Unix マシンについて

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name CodeDeployDemo-AS-Group \
    --launch-template CodeDeployDemo-AS-Launch-Template,Version='$Latest' \
    --min-size 1 \
    --max-size 1 \
    --desired-capacity 1 \
    --availability-zones availability-zone \
    --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

ローカル Windows マシンの場合

aws autoscaling create-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --launch-template LaunchTemplateName=CodeDeployDemo-AS-Launch-Template,Version="\$Latest" --min-size 1 --max-size 1 -desired-capacity 1 --availability-zones availability-zone --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true

これらのコマンドは、**CodeDeployDemo-AS-Group** という名前の Amazon EC2 起動テンプ レートに基づいた **CodeDeployDemo-AS-Launch-Template** という名前の Auto Scaling グ ループを作成します。この Auto Scaling グループには Amazon EC2 インスタンスが 1 つだけあ り、指定したアベイラビリティーゾーンに作成されます。この Auto Scaling グループ内の各イ ンスタンスには、タグ Name=CodeDeployDemo があります。このタグは、後で CodeDeploy エージェントをインストールするときに使用されます。 3. CodeDeployDemo-AS-Group に対して describe-auto-scaling-groups コマンドを呼び出しま す。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
LifecycleState]" --output text
```

戻り値に、Healthy および InService と表示されるまで続行しないでください。

Auto Scaling グループ中のインスタンスには、CodeDeploy デプロイで使用されるためにインストールされている CodeDeploy エージェントがある必要があります。Auto Scaling グループの作成時に追加されたタグ AWS Systems Manager を使用して、から create-association コマンドを呼び出して CodeDeploy エージェントをインストールします。

```
aws ssm create-association \
    --name AWS-ConfigureAWSPackage \
    --targets Key=tag:Name,Values=CodeDeployDemo \
    --parameters action=Install, name=AWSCodeDeployAgent \
    --schedule-expression "cron(0 2 ? * SUN *)"
```

このコマンドは、Systems Manager ステートマネージャーで関連付けを作成します。これ は、Auto Scaling グループのすべてのインスタンスに CodeDeploy エージェントをインストー ルし、毎週日曜日の午前 2 時に更新を試みます。CodeDeploy エージェントの詳細については、 「<u>CodeDeploy エージェントの使用</u>」を参照してください。Systems Manager のさらなる詳細 については、「<u>AWS Systems Managerとは</u>」を参照してください。

Auto Scaling グループ (コンソール) を作成して設定するには

- 1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
- グローバルナビゲーションバーで、AWS 全般のリファレンス の<u>リージョンとエンドポイント</u>で 一覧表示されているリージョンのいずれかが選択されていることを確認してください。Amazon EC2 Auto Scaling リソースは、指定したリージョンに関連付けられ、CodeDeploy は選択された リージョンでのみサポートされます。
- 3. ナビゲーションペインで、[インスタンス]の[テンプレートの起動]を選択します。
- 4. [起動テンプレートの作成]を選択してください。

- Launch template name and description ダイアログで、Launch template name に CodeDeployDemo-AS-Launch-Template を入力します。他のフィールドはデフォルトのた めのデフォルトをそのままにします。
- 6. Amazon machine image (AMI) ダイアログで、AMI の下にあるドロップダウンをクリックし、こ のチュートリアルで使用する AMI を選択します。
 - AMI ドロップダウンの Quick Start 上で、次のどれかを選択します: Amazon Linux 2 AMI、Red Hat Enterprise Linux 7.1、Ubuntu Server 14.04 LTS、またはMicrosoft Windows Server 2012 R2

Note

CodeDeploy と互換性がある AMI のカスタムバージョンの場合、クイックスタート タブを参照する代わりに、ここでそれを選択します。CodeDeploy および Amazon EC2 Auto Scaling でカスタム AMI を使用する方法についての詳細は、<u>CodeDeploy</u> と Amazon EC2 Auto Scaling でのカスタム AMI の使用 を参照してください。

- Instance type で、ドロップダウンを選択し、t1.micro を選択します。検索バーを使用すると、 よりすばやく検索できます。
- 8. Key pair (login) ダイアログボックスで、[Choose an existing key pair を選択します。[キーペアの選択] のドロップダウンリストで、前のステップで作成した Amazon EC2 インスタンスのキーペアを選択します。
- 9. Network settings ダイアログボックスで、[Virtual Public Cloud (VPC) を選択します。

Security groupsドロップダウンで、<u>tutorial's prerequisites section</u> (**CodeDeployDemo-AS-SG**) で作成したセキュリティグループを選択します。

10. Advanced details ダイアログボックスを展開します。IAM instance profile ドロップダウン で、IAM instance profile の下で以前の (**CodeDeployDemo-EC2-Instance-Profile**) を作成 した IAM ロールを選択します。

デフォルトの残りはそのままにしておきます。

- 11. [起動テンプレートの作成] を選択してください。
- 12. Next steps ダイアログボックスで、[Create Auto Scaling group を選択します。
- 13. Choose launch template or configuration (起動テンプレートまたは設定の選択) ページで、Auto Scaling group name を CodeDeployDemo-AS-Group に入力します。

- 14. 起動テンプレートダイアログボックスで、起動テンプレート (CodeDeployDemo-AS-Launch-Template) を入力する必要があります。そうでない場合は、ドロップダウンメニューから、そ れを選択します。デフォルトのままにして、Next を選択します。
- 15. [ネットワーク] セクションの下にある [インスタンス起動オプションを選択] ページで、[VPC] は デフォルト VPC を選択します。次に、アベイラビリティーゾーンとサブネットにはデフォル トサブネットを選択します。デフォルトを選択できない場合は、VPC を作成する必要がありま す。Amazon S3 の詳細については、Amazon VPC の開始方法 を参照してください。
- [Instance type requirements] (インスタンスタイプの要件) セクションでは、このステップを簡略 化するためにデフォルト設定を使用します。(起動テンプレートを上書きしないでください。) こ のチュートリアルでは、起動テンプレートで指定されたインスタンスタイプを使用して、オンデ マンドインスタンスのみを起動します。
- 17. Next を選択して Configure advanced options ページに行きます。
- 18. デフォルト値をそのまま保ち、Nextを選択します。
- 19. Configure group size and scaling policies ページで、1 のデフォルトの Group size の値をそのま まにします。[Next (次へ)] を選択します。
- 20. 通知の設定の手順をスキップし、Nextを選択します。
- 21. Add tags ページ上で、後で CodeDeploy エージェントをインストールするときに使用するタグ を追加します。[タグを追加] を選択します。
 - a. [Key] (キー) に、「Name」と入力します。
 - b. [値] には「CodeDeployDemo」と入力します。

[Next (次へ)] を選択します。

- 22. Review ページで上の Auto Scaling グループの情報を確認し、Create Auto Scaling group を選択 します。
- ナビゲーションバーで、選択された Auto Scaling Groups を用いて CodeDeployDemo-AS-Group を選択し、Instance Management タブを選びます。[Lifecycle] 列に、[InService] の値が 表示されて、[Health Status] 列に、[Healthy] の値が表示されるまで進まないでください。
- 24. CodeDeploy エージェントをインストールするには、「<u>CodeDeploy エージェントのインストー</u> <u>ル</u>」のステップに従い、Name=CodeDeployDemo インスタンスタグを使用します。

ステップ 2: Auto Scaling グループにアプリケーションをデプロイする

このステップでは、Auto Scaling グループの単一 Amazon EC2 インスタンスにリビジョンをデプロ イします。

トピック

- デプロイを作成するには (CLI)
- ・ デプロイを作成するには (コンソール)

デプロイを作成するには (CLI)

 create-application コマンドを呼び出して、SimpleDemoApp と言う名前のアプリケーションを 作成します。

aws deploy create-application --application-name SimpleDemoApp

- ステップ 2: CodeDeployのサービスのロールを作成する の手順に従ってサービスロールを作成 している必要があります。サービスロールは、Amazon EC2 インスタンスにアクセスしてタグ を拡張 (読み込み) する許可を CodeDeploy に付与します。サービスロール ARN が必要になりま す。サービスロール ARN を取得するには、サービスロール ARN の取得 (CLI) の手順に従いま す。
- これで、指定したサービスロール ARN で、create-deployment-group コマンドを呼び出して SimpleDemoDG という名前のデプロイグループを作成し、SimpleDemoApp という名前のアプ リケーションと関連付け、CodeDeployDemo-AS-Group と言う名前の Auto Scaling グループ と CodeDeployDefault.OneAtATime と言う名前のデプロイ設定を使用するサービスロール ARN が作成されました。

Note

create-deployment-group コマンドは、デプロイおよびインスタンス内の指定されたイベ ントについて、トピックサブスクライバーに Amazon SNS 通知を送信するトリガーの 作成をサポートします。このコマンドは、Amazon CloudWatch アラームのモニタリン グしきい値が満たされたときにデプロイを自動的にロールバックし、デプロイを停止す るアラームを設定するオプションもサポートします。このチュートリアルでは、これら のアクションのためのコマンドは含まれていません。 ローカル Linux、macOS、Unix マシンについて

aws deploy create-deployment-group \

- --application-name SimpleDemoApp \setminus
- --auto-scaling-groups CodeDeployDemo-AS-Group \
- --deployment-group-name SimpleDemoDG \setminus
- --deployment-config-name CodeDeployDefault.OneAtATime \setminus
- --service-role-arn service-role-arn

ローカル Windows マシンの場合

aws deploy create-deployment-group --application-name SimpleDemoApp --auto-scalinggroups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG --deploymentconfig-name CodeDeployDefault.OneAtATime --service-role-arn *service-role-arn*

 指定された場所のリビジョンを使用して、SimpleDemoApp という名前のアプリケーション と関連付けられたデプロイ、CodeDeployDefault.OneAtATime という名前のデプロイ設 定、SimpleDemoDG という名前のデプロイグループを作成する create-deployment コマンドを 呼び出します。

Amazon Linux および RHEL の Amazon EC2インスタンスの場合、ローカルの Linux、macOS、 または Unixマシンから呼び出します

```
aws deploy create-deployment \
    --application-name SimpleDemoApp \
    --deployment-config-name CodeDeployDefault.OneAtATime \
    --deployment-group-name SimpleDemoDG \
    --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
SampleApp_Linux.zip
```

bucket-name は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を awscodedeploy-us-east-2 に置き換えます。バケット名のリストについては、<u>リージョン別リ</u> <u>ソースキットバケット名</u> を参照してください。

ローカル Windows マシンから呼び出した Amazon Linux および RHEL Amazon EC2 instances の場合 aws deploy create-deployment --application-name SimpleDemoApp --deployment-configname CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3location bucket=bucket-name, bundleType=zip, key=samples/latest/SampleApp_Linux.zip

bucket-name は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を awscodedeploy-us-east-2 に置き換えます。バケット名のリストについては、<u>リージョン別リ</u> ソースキットバケット名 を参照してください。

ローカルの Linux、macOS、または Unix マシンから呼び出した Windows サーバー Amazon Linux および RHEL の Amazon EC2インスタンスの場合

aws deploy create-deployment \
 --application-name SimpleDemoApp \
 --deployment-config-name CodeDeployDefault.OneAtATime \
 --deployment-group-name SimpleDemoDG \
 --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
SampleApp_Windows.zip

bucket-name は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を awscodedeploy-us-east-2 に置き換えます。バケット名のリストについては、<u>リージョン別リ</u> ソースキットバケット名 を参照してください。

ローカル Windows マシンから呼び出した Windows サーバー Amazon EC2 インスタンスの場合

aws deploy create-deployment --application-name SimpleDemoApp --deployment-configname CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Windows.zip

bucket-name は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を awscodedeploy-us-east-2 に置き換えます。バケット名のリストについては、<u>リージョン別リ</u> ソースキットバケット名 を参照してください。 Note

現在のところ、CodeDeploy は Ubuntu サーバーの Amazon EC2 インスタンスにデプ ロイするサンプルリビジョンを提供していません。リビジョンを独自に作成するに は、CodeDeploy のアプリケーションリビジョンの操作 を参照してください。

5. get-deployment コマンドを呼び出して、デプロイが成功したことを確認します。

このコマンドを呼び出す前に、create-deployment コマンドの呼び出しで返された、デプロイの ID が必要になります。デプロイ ID を再度取得することが必要な場合には、SimpleDemoApp と いう名前のアプリケーションと SimpleDemoDG と言う名前のデプロイグループに対して、listdeployments コマンドを呼び出します。

aws deploy list-deployments --application-name SimpleDemoApp --deployment-groupname SimpleDemoDG --query "deployments" --output text

次に、デプロイ ID を使用して get-deployment コマンドを呼び出します。

aws deploy get-deployment --deployment-id deployment-id --query
"deploymentInfo.status" --output text

Succeeded の値が返されるまで続けないでください。

デプロイを作成するには (コンソール)

- ステップ 2: CodeDeployのサービスのロールを作成する の手順に従ってサービスロールを作成 している必要があります。サービスロールは、インスタンスにアクセスしてタグを拡張 (読み込 み) する許可を CodeDeploy に付与します。CodeDeploy コンソールを使用してアプリケーショ ンリビジョンをデプロイする前に、サービスロール ARN が必要になります。サービスロール ARN を取得するには、サービスロール ARN の取得 (コンソール)の手順に従います。
- 2. サービスロール ARN があるので、CodeDeploy コンソールを使用して、アプリケーションリビ ジョンをデプロイできます。

にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。 Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 3. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 4. [Create application] を選択します。
- 5. [カスタムアプリケーション]を選択します。
- 6. [アプリケーション名] に、「SimpleDemoApp」と入力します。
- 7. [コンピューティングプラットフォーム] で [EC2/オンプレミス] を選択します。
- 8. [Create application] を選択します。
- 9. [デプロイグループ] タブで、[デプロイグループの作成] を選択します。
- 10. [Deployment group name] (デプロイグループ名) に「SimpleDemoDG」と入力します。
- 11. [サービスロール] で、 サービスロールの名前を選択します。
- 12. [デプロイタイプ] で、[インプレース] を選択します。
- 13. [環境設定] で、[Auto Scaling グループ]、[CodeDeployDemo-AS-Group] の順に選択します。
- 14. [デプロイ設定] で [CodeDeployDefault.OneAtATime] を選択します。
- 15. [Enable load balancing (ロードバランシングの有効化)] のチェックを外します。
- 16. デプロイグループの作成を選択します。
- 17. デプロイグループページで、[デプロイの作成]を選択します。
- 18. [Revision type (リビジョンのタイプ)] の横の [My application is stored in Amazon S3 (Amazon S3 に保存されているアプリケーション)] を選択します。
- 19. [リビジョンの場所] に、オペレーティングシステムとリージョンのサンプルアプリケーションの 場所を入力します。

Amazon Linux、RHEL Amazon EC2 インスタンスの場合

リージョン	サンプルアプリケーションの場所
米国東部 (オハイオ) リージョン	<pre>http://s3-us-east-2.amazona ws.com/aws-codedeploy-us-ea st-2/samples/latest/SampleA pp_Linux.zip</pre>

リージョン	サンプルアプリケーションの場所
米国東部(バージニア州北部) リージョン	<pre>http://s3.amazonaws.com/aws- codedeploy-us-east-1/samples/ latest/SampleApp_Linux.zip</pre>
US West (N. California) Region	<pre>http://s3-us-west-1.amazona ws.com/aws-codedeploy-us-we st-1/samples/latest/SampleA pp_Linux.zip</pre>
米国西部 (オレゴン) リージョン	<pre>http://s3-us-west-2.amazona ws.com/aws-codedeploy-us-we st-2/samples/latest/SampleA pp_Linux.zip</pre>
カナダ (中部) リージョン	<pre>http://s3-ca-central-1.amaz onaws.com/aws-codedeploy-ca -central-1/samples/latest/S ampleApp_Linux.zip</pre>
欧州 (アイルランド) リージョン	<pre>http://s3-eu-west-1.amazona ws.com/aws-codedeploy-eu-we st-1/samples/latest/SampleA pp_Linux.zip</pre>
欧州 (ロンドン) リージョン	<pre>http://s3-eu-west-2.amazona ws.com/aws-codedeploy-eu-we st-2/samples/latest/SampleA pp_Linux.zip</pre>
欧州(パリ)リージョン	<pre>http://s3-eu-west-3.amazona ws.com/aws-codedeploy-eu-we st-3/samples/latest/SampleA pp_Linux.zip</pre>

リージョン	サンプルアプリケーションの場所
欧州(フランクフルト)リージョン	<pre>http://s3-eu-central-1.amaz onaws.com/aws-codedeploy-eu -central-1/samples/latest/S ampleApp_Linux.zip</pre>
イスラエル (テルアビブ) リージョン	<pre>https://aws-codedeploy-il-c entral-1.s3.il-central-1.am azonaws.com/samples/latest/ SampleApp_Linux.zip</pre>
アジアパシフィック (香港) リージョン	<pre>https://aws-codedeploy-ap-e ast-1.s3.ap-east-1.amazonaw s.com/samples/latest/Sample App_Linux.zip</pre>
Asia Pacific (Tokyo) Region	<pre>http://s3-ap-northeast-1.am azonaws.com/aws-codedeploy- ap-northeast-1/samples/late st/SampleApp_Linux.zip</pre>
Asia Pacific (Seoul) Region	<pre>http://s3-ap-northeast-2.am azonaws.com/aws-codedeploy- ap-northeast-2/samples/late st/SampleApp_Linux.zip</pre>
アジアパシフィック (シンガポール) リー ジョン	<pre>http://s3-ap-southeast-1.am azonaws.com/aws-codedeploy- ap-southeast-1/samples/late st/SampleApp_Linux.zip</pre>
アジアパシフィック (シドニー) リージョン	<pre>http://s3-ap-southeast-2.am azonaws.com/aws-codedeploy- ap-southeast-2/samples/late st/SampleApp_Linux.zip</pre>

リージョン	サンプルアプリケーションの場所
アジアパシフィック (メルボルン) リージョ ン	<pre>https://aws-codedeploy-ap-s outheast-4.s3.ap-southeast- 4.amazonaws.com/samples/lat est/SampleApp_Linux.zip</pre>
アジアパシフィック (ムンバイ) リージョン	<pre>http://s3-ap-south-1.amazon aws.com/aws-codedeploy-ap-s outh-1/samples/latest/Sampl eApp_Linux.zip</pre>
南米 (サンパウロ) リージョン	<pre>http://s3-sa-east-1.amazona ws.com/aws-codedeploy-sa-ea st-1/samples/latest/SampleA pp_Linux.zip</pre>

Windows Server Amazon EC2 インスタンスの場合

リージョン	サンプルアプリケーションの場所
米国東部 (オハイオ) リージョン	<pre>http://s3-us-east-2.amazona ws.com/aws-codedeploy-us-ea st-2/samples/latest/SampleA pp_Windows.zip</pre>
米国東部(バージニア州北部) リージョン	<pre>http://s3.amazonaws.com/aws- codedeploy-us-east-1/samples/ latest/SampleApp_Windows.zip</pre>
US West (N. California) Region	<pre>http://s3-us-west-1.amazona ws.com/aws-codedeploy-us-we st-1/samples/latest/SampleA pp_Windows.zip</pre>

リージョン	サンプルアプリケーションの場所
米国西部 (オレゴン) リージョン	<pre>http://s3-us-west-2.amazona ws.com/aws-codedeploy-us-we st-2/samples/latest/SampleA pp_Windows.zip</pre>
カナダ (中部) リージョン	<pre>http://s3-ca-central-1.amaz onaws.com/aws-codedeploy-ca -central-1/samples/latest/S ampleApp_Windows.zip</pre>
欧州 (アイルランド) リージョン	<pre>http://s3-eu-west-1.amazona ws.com/aws-codedeploy-eu-we st-1/samples/latest/SampleA pp_Windows.zip</pre>
欧州 (ロンドン) リージョン	<pre>http://s3-eu-west-2.amazona ws.com/aws-codedeploy-eu-we st-2/samples/latest/SampleA pp_Windows.zip</pre>
欧州(パリ)リージョン	<pre>http://s3-eu-west-3.amazona ws.com/aws-codedeploy-eu-we st-3/samples/latest/SampleA pp_Windows.zip</pre>
欧州(フランクフルト)リージョン	<pre>http://s3-eu-central-1.amaz onaws.com/aws-codedeploy-eu -central-1/samples/latest/S ampleApp_Windows.zip</pre>
イスラエル (テルアビブ) リージョン	<pre>https://aws-codedeploy-il-c entral-1.s3.il-central-1.am azonaws.com/samples/latest/ SampleApp_Windows.zip</pre>

リージョン	サンプルアプリケーションの場所
アジアパシフィック (香港) リージョン	<pre>https://aws-codedeploy-ap-e ast-1.s3.ap-east-1.amazonaw s.com/samples/latest/Sample App_Windows.zip</pre>
Asia Pacific (Seoul) Region	<pre>http://s3-ap-northeast-2.am azonaws.com/aws-codedeploy- ap-northeast-2/samples/late st/SampleApp_Windows.zip</pre>
アジアパシフィック (シンガポール) リー ジョン	<pre>http://s3-ap-southeast-1.am azonaws.com/aws-codedeploy- ap-southeast-1/samples/late st/SampleApp_Windows.zip</pre>
アジアパシフィック (シドニー) リージョン	<pre>http://s3-ap-southeast-2.am azonaws.com/aws-codedeploy- ap-southeast-2/samples/late st/SampleApp_Windows.zip</pre>
アジアパシフィック (メルボルン) リージョ ン	<pre>https://aws-codedeploy-ap-s outheast-4.s3.ap-southeast- 4.amazonaws.com/samples/lat est/SampleApp_Windows.zip</pre>
アジアパシフィック (ムンバイ) リージョン	<pre>http://s3-ap-south-1.amazon aws.com/aws-codedeploy-ap-s outh-1/samples/latest/Sampl eApp_Windows.zip</pre>
南米 (サンパウロ) リージョン	<pre>http://s3-sa-east-1.amazona ws.com/aws-codedeploy-sa-ea st-1/samples/latest/SampleA pp_Windows.zip</pre>

Ubuntu Server Amazon EC2 インスタンスの場合

Amazon S3 に格納されるカスタムアプリケーションリビジョンの場所を入力します。

- 20. [デプロイメントの説明]は空白のままにしておきます。
- 21. [Advanced] を展開します。
- 22. [デプロイの作成]を選択します。

Note

Succeeded の代わりに Failed が表示された場合、<u>デプロイをモニタリングおよび</u> トラブルシューティングします。</u>にある手法の一部を試してみることもできます (SimpleDemoApp のアプリケーション名、および SimpleDemoDG のデプロイグループ 名を使用して)。

ステップ 3: 結果の確認

このステップでは、CodeDeploy が Auto Scaling グループの単一 Amazon EC2 インスタンスについ ての SimpleDemoApp のリビジョンをインストールしたかどうかを確認します。

トピック

- 結果を確認するには (CLI)
- 結果を確認するには (コンソール)

結果を確認するには (CLI)

まず、Amazon EC2 インスタンスのパブリック DNS が必要です。

を使用して AWS CLI 、 describe-instances コマンドを呼び出して Auto Scaling グループ内の Amazon EC2 インスタンスのパブリック DNS を取得します。

このコマンドを呼び出す前に、Amazon EC2 インスタンスの ID が必要です。この ID を取得するに は、以前に行ったように、**CodeDep1oyDemo-AS-Group** に対して describe-auto-scaling-groups を 呼び出します。 aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --output text

次に describe-instances コマンドを呼び出します。

aws ec2 describe-instances --instance-id instance-id --query
"Reservations[0].Instances[0].PublicDnsName" --output text

返される値は Amazon EC2 インスタンスのパブリック DNS です。

ウェブブラウザを使用して、次のような URL を使用して Amazon EC2 インスタンスにデプロイした [SimpleDemoApp] リビジョンを表示します。

http://ec2-01-234-567-890.compute-1.amazonaws.com

成功のページが表示されると、リビジョンは Auto Scaling グループの単一の Amazon EC2 インスタ ンスに CodeDeploy を使用して正しくデプロイされました。

次に、Amazon EC2 インスタンスを Auto Scaling グループに追加します。Amazon EC2 Auto Scaling が Amazon EC2 インスタンスを追加すると、CodeDeploy は新しいインスタンスにリビジョ ンをデプロイします。

結果を確認するには (コンソール)

まず、Amazon EC2 インスタンスのパブリック DNS が必要です。

Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

Amazon EC2 ナビゲーションペインの Auto Scaling の下で、Auto Scaling グループ を選択 し、**CodeDeployDemo-AS-Group** のエントリを選択します。

インスタンス タブで、リスト内の Amazon EC2 インスタンス ID を選択します。

[Instances] ページの、[Description] タブで、[Public DNS] 値をメモします。次のように表示されま す。ec2-01-234-567-890.compute-1.amazonaws.com

ウェブブラウザを使用して、次のような URL を使用して Amazon EC2 インスタンスにデプロイした [SimpleDemoApp] リビジョンを表示します。

http://ec2-01-234-567-890.compute-1.amazonaws.com

成功のページが表示されると、リビジョンは Auto Scaling グループの単一の Amazon EC2 インスタ ンスに CodeDeploy を使用して正しくデプロイされました。

次に、Auto Scaling グループに Amazon EC2 インスタンスを追加します。Amazon EC2 Auto Scaling が Amazon EC2 インスタンスを追加した後、CodeDeploy は新しい Amazon EC2 インスタ ンスにリビジョンをデプロイします。

ステップ 4: Auto Scaling グループの Amazon EC2 インスタンスの数を増や す

このステップでは、追加の Amazon EC2 インスタンスを作成するように Auto Scaling グループに指 示します。Amazon EC2 Auto Scaling がインスタンスを作成すると、CodeDeploy によってリビジョ ンがデプロイされます。

トピック

- Auto Scaling グループ (CLI) の Amazon EC2 インスタンスの数をスケールアウトする
- ・ <u>デプロイグループ (コンソール) で Amazon EC2 インスタンスの数をスケールアップするには</u>

Auto Scaling グループ (CLI) の Amazon EC2 インスタンスの数をスケールアウトする

1. update-auto-scaling-group のコマンドを呼び出し、**CodeDeployDemo-AS-Group** という名前の Auto Scaling グループの Amazon EC2 インスタンスを、1 から 2 に増やします。

ローカル Linux、macOS、Unix マシンについて:

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name CodeDeployDemo-AS-Group \
    --min-size 2 \
    --max-size 2 \
    --desired-capacity 2
```

ローカル Windows マシンの場合

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-
AS-Group --min-size 2 --max-size 2 --desired-capacity 2
```

Auto Scaling グループに 2 つの Amazon EC2 インスタンスが存在することを確認します。CodeDep1oyDemo-AS-Group に対して describe-auto-scaling-groups コマンドを呼び出します。

aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus, LifecycleState]" --output text

戻り値に、Healthy と InService の両方が表示されるまで続行しないでください。

デプロイグループ (コンソール) で Amazon EC2 インスタンスの数をスケールアップ するには

- Amazon EC2 ナビゲーションバーで、Auto Scaling の下で、Auto Scaling Groups を選択し、次 に CodeDeployDemo-AS-Group を選択します。
- 2. [Actions] (アクション)を選択して、[Edit] (編集)を選択します。
- 3. [詳細] タブで、[必要]、[最小] および [最大] ボックスに 2 と入力し、[保存] を選択します。
- [Instances] タブを選択します。新しい Amazon EC2 インスタンスが一覧に表示されます。(イン スタンスが表示されない場合、[Refresh] ボタンを数回選択する必要が生じる場合があります)。
 [Lifecycle] 列に、[InService] の値が表示されて、[Health Status] 列に、[Healthy] の値が表示され るまで進まないでください。

ステップ 5: 結果を再度確認します

このステップでは、CodeDeploy が Auto Scaling グループの新しいインスタンスの [SimpleDemoApp] リビジョンをインストールしたかどうかを確認します。

トピック

- 自動デプロイの結果を確認するには (CLI)
- 自動デプロイの結果を確認するには (コンソール)

自動デプロイの結果を確認するには (CLI)

get-deployment コマンドを確認する前に、自動デプロイの ID が必要です。ID を取得するには、SimpleDemoApp という名前のアプリケーションと SimpleDemoDG と言う名前のデプロイグループに対して list-deployments コマンドを呼び出します。

aws deploy list-deployments --application-name SimpleDemoApp --deployment-groupname SimpleDemoDG --query "deployments" --output text

2 つのデプロイ ID があるはずです。get-deployment コマンドの呼び出しでは、まだ使用してい ない ID を使用してください。

aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text

デプロイのステータスに加えて、autoScaling がコマンド出力に表示されます (autoScaling は Amazon EC2 Auto Scaling が作成したデプロイ)。

デプロイのステータスに Succeeded が表示されるまで進まないでください。

 describe-instances のコマンドを呼び出す前に、新しい Amazon EC2 インスタンスの ID が必要 です。この ID を取得するには、CodeDeployDemo-AS-Group に対して describe-auto-scalinggroups コマンドをもう一度呼び出します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --
output text
```

次に describe-instances コマンドを呼び出します。

aws ec2 describe-instances --instance-id instance-id --query
"Reservations[0].Instances[0].PublicDnsName" --output text

describe-instances のコマンドの出力で、新しい Amazon EC2 インスタンスの パブリック DNS をメモします。

3. ウェブブラウザを使用して、次のような URL を使用して Amazon EC2 インスタンスにデプロイ した SimpleDemoApp のリビジョンを表示します。

http://ec2-01-234-567-890.compute-1.amazonaws.com

成功のページが表示されると、CodeDeploy によってリビジョンが Auto Scaling グループのス ケールアップされた Amazon EC2 インスタンスに正常にデプロイされています。

自動デプロイの結果を確認するには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、Deploy を展開し、Deployments を選択します。
- 3. Amazon EC2 Auto Scaling が作成したデプロイのデプロイ ID を選択します。
- 4. [デプロイ] ページに、デプロイに関する情報が表示されます。通常の場合は、ユーザーがデプ ロイを作成しますが、Amazon EC2 Auto Scaling はユーザーに代わってリビジョンを新しい Amazon EC2 インスタンスにデプロイします。
- 5. ページ上部に [成功] と表示されたら、インスタンスで結果を確認します。最初に、インスタン スのパブリック DNS を取得する必要があります。
- Amazon EC2 ナビゲーションペインの Auto Scaling の下で、Auto Scaling グループ を選択し、CodeDeployDemo-AS-Group のエントリを選択します。
- 7. Instances タブで、新しい Amazon EC2 インスタンスの ID を選択します。
- 8. [Instances] ページの、[Description] タブで、[Public DNS] 値をメモします。次のように表示され ます。ec2-01-234-567-890.compute-1.amazonaws.com

インスタンスにデプロイされた SimpleDemoApp リビジョンを、次のような URL を使用して表示し ます。

http://ec2-01-234-567-890.compute-1.amazonaws.com

成功のページが表示されると、CodeDeploy によってリビジョンが Auto Scaling グループのスケール アップされた Amazon EC2 インスタンスに正常にデプロイされています。

ステップ 6: クリーンアップする

このステップでは、このチュートリアルで使用したリソースの料金が継続的に発生するのを避ける ために Auto Scaling グループを削除します。必要に応じて、Auto Scaling 設定、および CodeDeploy デプロイコンポーネントレコードを削除できます。

トピック

- リソース (CLI) をクリーンアップするには
- ・ リソース (コンソール) をクリーンアップするには

リソース (CLI) をクリーンアップするには

 CodeDeployDemo-AS-Group に対して delete-auto-scaling-group コマンドを呼び出すことに よって、Auto Scaling グループを削除し ます。これにより、Amazon EC2 インスタンスも終了 します。

aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete

 必要に応じて、CodeDeployDemo-AS-Launch-Template という名前の起動設定に対し て、delete-launch-template のコマンドを呼び出し、Auto Scaling 起動テンプレートを削除しま す。

aws ec2 delete-launch-template --launch-template-name CodeDeployDemo-AS-Launch-Template

 必要に応じて、delete-application という名前のアプリケーションに対して、SimpleDemoApp のコマンドを呼び出し、CodeDeploy からのアプリケーションを削除します。これにより、関連 するすべてのデプロイ、デプロイグループ、およびリビジョンレコードも削除されます。

aws deploy delete-application --application-name SimpleDemoApp

4. Systems Manager ステートマネージャーの関連付けを削除するには、delete-association のコマンドを呼び出します。

```
aws ssm delete-association --assocation-id association-id
```

describe-association のコマンドを呼び出すことで、association-id を取得できます。

aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
Key=tag:Name,Values=CodeDeployDemo

リソース (コンソール) をクリーンアップするには

Amazon EC2 インスタンスを終了する Auto Scaling グループを削除するには

1.

にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。

- Amazon EC2 ナビゲーションペインで、Auto Scaling の下で、Auto Scaling Groups を選択して から、CodeDeployDemo-AS-Group のエントリを選択します。
- 3. [Actions] を選択して、[Delete] を選択し、次に [Yes, Delete] を選択します。

(オプション) 起動テンプレートを削除するには

- ナビゲーションバーで、Auto Scaling の下で、Launch Configurations を選択し、CodeDeployDemo-AS-Launch-Template を選択します。
- 2. [Actions] を選択して、[Delete launch configuration] を選択し、[Yes, Delete] を選択します。
- 必要に応じて、CodeDeploy からアプリケーションを削除します。これにより、関連するす べてのデプロイ、デプロイグループ、およびリビジョンレコードも削除されます。<u>https://</u> console.aws.amazon.com/codedeploy で、CodeDeploy コンソールを開きます。
- 2. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。

- 3. アプリケーションのリストで、SimpleDemoApp を選択します。
- 4. [Application details] ページで、[Delete application] を選択します。

5. 確認を求めるメッセージが表示されたら、Deleteと入力し、[削除]を選択してください。

Systems Manager ステートマネージャーの関連付けの削除。

- 1. AWS Systems Manager コンソールを https://console.aws.amazon.com/systems-manager.com で開きます。
- 2. ナビゲーションペインで、[ステートマネージャー] を選択してください。
- 3. 作成した関連付けを選択し、[削除]を選択します。

チュートリアル: CodeDeploy を使用して、GitHub からアプリケー ションをデプロイします。

このチュートリアルでは、CodeDeploy を使用して GitHub からサンプルアプリケーションリビ ジョンを、Amazon Linux を実行している単一の Amazon EC2 インスタンス、単一の Red Hat Enterprise Linux (RHEL) インスタンス、または単一の Windows サーバーインスタンスにデプロイし ます。GitHub と CodeDeploy の統合については、<u>GitHub と CodeDeploy との統合</u> を参照してくだ さい。

Note

CodeDeploy を使用して、アプリケーションリビジョンを GitHub から Ubuntu サーバーイ ンスタンスにデプロイすることもできます。<u>ステップ 2: サンプルのアプリケーションリビ</u> <u>ジョンを作成する</u>の <u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、 または Red Hat エンタープライズ Linux) を使用してオンプレミスインスタンスにアプリケー ションをデプロイします。</u>の中に記述されているサンプルリビジョンを使用するか、Ubuntu サーバーインスタンスおよび CodeDeploy と互換性があるリビジョンを作成できます。独自 のリビジョンを作成するには、<u>CodeDeploy のリビジョンを計画する</u> と <u>CodeDeploy 用のア</u> <u>プリケーション仕様ファイルをリビジョンに追加</u> を参照してください。

トピック

- 前提条件
- ステップ 1: GitHub アカウントを設定します。
- ステップ 2: GitHub リポジトリを作成します。
- ステップ 3: GitHub リポジトリにサンプルアプリケーションをアップロードします。

- ステップ 4: インスタンスをプロビジョニングします。
- ステップ 5: アプリケーションおよびデプロイグループを作成します。
- ステップ 6: アプリケーションをインスタンスにデプロイします。
- ステップ 7: デプロイをモニタリングおよび確認します。
- ステップ 8: クリーンアップする

前提条件

このチュートリアルを開始する前に、以下を実行します。

- ローカルマシンで Git をインストールします。Git をインストールするには、Git downloads を参照 してください。
- AWS CLIのインストールと設定を含む、「<u>CodeDeploy の開始方法</u>」の手順を完了します。これ は、 を使用して GitHub からインスタンス AWS CLI にリビジョンをデプロイする場合に特に重要 です。

ステップ1: GitHub アカウントを設定します。

リビジョンが保存される GitHub リポジトリを作成するには、GitHub アカウントが必要です。すで に GitHub アカウントをお持ちの場合は、<u>ステップ 2: GitHub リポジトリを作成します。</u> に進んでく ださい。

- 1. https://github.com/join にアクセスします。
- 2. ユーザー名、Eメールアドレス、パスワードを入力します。
- 3. [Sign up for GitHub] を選択し、指示に従います。

ステップ 2: GitHub リポジトリを作成します。

リビジョンを保存するには、GitHub リポジトリが必要です。

すでに GitHub リポジトリがある場合は、このチュートリアル全体で **CodeDeployGitHubDemo** の 名前を必ず置き換えて、「<u>ステップ 3: GitHub リポジトリにサンプルアプリケーションをアップロー</u> ドします。」に進んでください。

1. GitHub home page で、次のいずれかの操作を実行します。

- [Your repositories] で、[New repository] を選択します。
- ・ナビゲーションバーで、[Create new (+)] を選択し、[New repository] を選択します。
- 2. [Create a new repository] ページで、次の操作を実行します。
 - [リポジトリ名] ボックスに「CodeDeployGitHubDemo」と入力します。
 - [Public] を選択します。

Note

デフォルトの [Public] オプションを選択すると、誰でもこのリポジトリを表示できま す。[プライベート] オプションを選択して、リポジトリを表示してコミットできる ユーザーを制限できます。

- [Initialize this repository with a README] チェックボックスをオフにします。代わりに、次の ステップでは、README.md ファイルを手動で作成します。
- [Create repository (リポジトリの作成)] を選択します。
- 3. ローカルマシンタイプ別の手順に従い、コマンドラインを使用してリポジトリを作成します。

Note

GitHub で 2 要素認証を有効にした場合、パスワードの入力を求められたら、GitHub の ログインパスワードの代わりに必ず個人アクセストークンを入力するようにしてくださ い。詳細については、<u>Providing your 2FA authentication code</u>を参照してください。

ローカル Linux、macOS、Unix マシンについて

1. ターミナルから、次のコマンドを一度に1つずつ実行します。*user-name*は GitHub のユー ザー名です。

mkdir /tmp/CodeDeployGitHubDemo

cd /tmp/CodeDeployGitHubDemo

touch README.md

git init

git add README.md

git commit -m "My first commit"

git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git

git push -u origin master

- 2. ターミナルを /tmp/CodeDeployGitHubDemo の場所で開いたままにします。
- ローカル Windows マシンの場合
- 1. 管理者として実行するコマンドプロンプトから、次のコマンドを一度に1つずつ実行します。

mkdir c:\temp\CodeDeployGitHubDemo

cd c:\temp\CodeDeployGitHubDemo

notepad README.md

 Notepad に README.md ファイルを保存します。Notepad を閉じます。次のコマンドを一度に1 つずつ実行します。user-name は GitHub のユーザー名です。

git init

git add README.md

git commit -m "My first commit"

git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git

ステップ 2: GitHub リポジトリを作成します。

git push -u origin master

3. コマンドプロンプトを c:\temp\CodeDeployGitHubDemo の場所で開いたままにします。

ステップ 3: GitHub リポジトリにサンプルアプリケーションをアップロー ドします。

このステップでは、パブリックの Amazon S3 バケットから GitHub リポジトリにサンプルリビジョ ンをコピーします。(分かりやすいように、このチュートリアルに用意してあるサンプルリビジョン は単一のウェブページです。)

Note

サンプルリビジョンの代わりに自身のリビジョンの 1 つを使用する場合は、以下が必要で す。

- <u>CodeDeploy のリビジョンを計画する</u> と <u>CodeDeploy 用のアプリケーション仕様ファイル</u> をリビジョンに追加 のガイドラインに従う。
- 対応するインスタンスタイプを使用する。
- GitHub のダッシュボードからアクセス可能である。

リビジョンがこれらの要件を満たしている場合は、「<u>ステップ 5: アプリケーションおよびデ</u> <u>プロイグループを作成します。</u>」に進んでください。

Ubuntu インスタンスにデプロイする場合は、Ubuntu サーバーインスタンスおよび CodeDeploy と互換性があるリビジョンを GitHub リポジトリにアップロードする必要があり ます。詳細については、<u>CodeDeploy のリビジョンを計画する</u>および<u>CodeDeploy 用のアプ</u> <u>リケーション仕様ファイルをリビジョンに追加</u>を参照してください。

トピック

- ローカル Linux、macOS、あるいは Unix マシンからサンプルリビジョンをプッシュします
- ローカル Windows マシンからサンプルリビジョンをプッシュする

ローカル Linux、macOS、あるいは Unix マシンからサンプルリビジョンをプッシュします

ターミナルを /tmp/CodeDeployGitHubDemo などの場所で開いたままにして、以下のコマンドを 一度に 1 つずつ実行します。

Note

デプロイ先を Windows 1 サーバーインスタンスにする場合は、コマンドで SampleApp_Windows.zip の代わりに SampleApp_Linux.zip を使用します。

(Amazon S3 copy command)

unzip SampleApp_Linux.zip

rm SampleApp_Linux.zip

git add .

git commit -m "Added sample app"

git push

(Amazon S3 copy command)は次のいずれかです。

- aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/
 SampleApp_Linux.zip . --region us-east-2(米国東部(オハイオ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/
 SampleApp_Linux.zip . --region us-east-1(米国東部(バージニア北部)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/ SampleApp_Linux.zip . --region us-west-1(米国西部(北カリフォルニア)リージョンの 場合)

- aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/
 SampleApp_Linux.zip . --region us-west-2(米国西部(オレゴン)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/
 SampleApp_Linux.zip . --region ca-central-1(カナダ(中部)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/
 SampleApp_Linux.zip . --region eu-west-1(欧州(アイルランド)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/
 SampleApp_Linux.zip . --region eu-west-2(欧州(ロンドン)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/
 SampleApp_Linux.zip . --region eu-west-3(欧州(パリ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/
 SampleApp_Linux.zip . --region eu-central-1(欧州(フランクフルト)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/
 SampleApp_Linux.zip . --region il-central-1(イスラエル(テルアビブ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/
 SampleApp_Linux.zip . --region ap-east-1(アジアパシフィック(香港)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/
 SampleApp_Linux.zip . --region ap-northeast-1(アジアパシフィック(東京)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/
 SampleApp_Linux.zip . --region ap-northeast-2(アジアパシフィック(ソウル)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/
 SampleApp_Linux.zip . --region ap-southeast-1(アジアパシフィック(シンガポール)
 リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/
 SampleApp_Linux.zip . --region ap-southeast-2(アジアパシフィック(シドニー)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/
 SampleApp_Linux.zip . --region ap-southeast-4(アジアパシフィック(メルボルン)
 リージョンの場合)

- aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/
 SampleApp_Linux.zip . --region ap-south-1(アジアパシフィック(ムンバイ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/
 SampleApp_Linux.zip . --region sa-east-1(南米(サンパウロ)リージョンの場合)

ローカル Windows マシンからサンプルリビジョンをプッシュする

コマンドプロンプトを c:\temp\CodeDeployGitHubDemo などの場所で開いたままにして、以下 のコマンドを一度に 1 つずつ実行します。

Note

デプロイ先を Amazon Linux または RHEL インスタンスにデプロイする予定がある場合は、 コマンドで SampleApp_Linux.zip の代わりに SampleApp_Windows.zip を使用しま す。

(Amazon S3 copy command)

the ZIP ファイルの内容の解凍先を、新しいサブディレクトリではなく、直接ローカルディレクトリ (c:\temp\CodeDeployGitHubDemo など) にします。

git add .

git commit -m "Added sample app"

git push

(Amazon S3 copy command) は次のいずれかです。

- aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/
 SampleApp_Windows.zip . --region us-east-2(米国東部(オハイオ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/
 SampleApp_Windows.zip . --region us-east-1(米国東部(バージニア北部)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/
 SampleApp_Windows.zip . --region us-west-1(米国西部(北カリフォルニア)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/
 SampleApp_Windows.zip . --region us-west-2(米国西部(オレゴン)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/
 SampleApp_Windows.zip . --region ca-central-1(カナダ(中部)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/
 SampleApp_Windows.zip . --region eu-west-1(欧州(アイルランド)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/
 SampleApp_Windows.zip . --region eu-west-2(欧州(ロンドン)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/
 SampleApp_Windows.zip . --region eu-west-3(欧州(パリ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/
 SampleApp_Windows.zip . --region eu-central-1(欧州(フランクフルト)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/
 SampleApp_Windows.zip . --region il-central-1(イスラエル(テルアビブ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/
 SampleApp_Windows.zip . --region ap-east-1(アジアパシフィック(香港)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/
 SampleApp_Windows.zip . --region ap-northeast-1(アジアパシフィック(東京)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/
 SampleApp_Windows.zip . --region ap-northeast-2(アジアパシフィック(ソウル)
 リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/
 SampleApp_Windows.zip . --region ap-southeast-1(アジアパシフィック(シンガポール)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/
 SampleApp_Windows.zip . --region ap-southeast-2(アジアパシフィック(シドニー)
 リージョンの場合)

- aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/
 SampleApp_Windows.zip . --region ap-southeast-4(アジアパシフィック(メルボルン)
 リージョンの場合)
- aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/
 SampleApp_Windows.zip . --region ap-south-1(アジアパシフィック(ムンバイ)リージョンの場合)
- aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/
 SampleApp_Windows.zip . --region sa-east-1(南米(サンパウロ)リージョンの場合)

独自のリビジョンを Ubuntu サーバーインスタンスにプッシュするには、リビジョンをローカルリポ ジトリにコピーしてから、次のコマンドを呼び出します。

git add . git commit -m "Added Ubuntu app" git push

ステップ 4: インスタンスをプロビジョニングします。

このステップでは、サンプルアプリケーションのデプロイ先であるインスタンスを作成または設定します。CodeDeploy でサポートされているオペレーティングシステムのいずれかを実行して いる Amazon EC2 インスタンスまたはオンプレミスインスタンスにデプロイできます。詳細については、CodeDeploy エージェントで対応するオペレーティングシステム (CodeDeploy デプロイで使用するインスタンスが設定済みである場合は、次のステップまでスキッ プします。)

インスタンスをプロビジョニングするには

- 1. <u>Amazon EC2 インスタンス (コンソール) を起動します。</u>の指示に従って、インスタンスをプロ ビジョニングします。
- インスタンスを起動するときは、タグの追加 ページで必ずタグを指定してください。タグを指 定する方法の詳細については、<u>Amazon EC2 インスタンス (コンソール) を起動します。</u>を参照 してください。

CodeDeploy エージェントがインスタンスで実行されていることを確認するには、

• <u>CodeDeploy エージェントが実行されていることの確認</u>の指示に従って、エージェントが実行 されていることを確認します。 インスタンスを正常にプロビジョンし、CodeDeploy エージェントが実行されていることを確認した ら、次のステップに進みます。

ステップ 5: アプリケーションおよびデプロイグループを作成します。

このステップでは、CodeDeploy コンソールまたは AWS CLI を使用して、GitHub リポジトリからサ ンプルリビジョンをデプロイするために使用するアプリケーションとデプロイグループを作成しま す。

アプリケーションおよびデプロイグループの作成 (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで Deploy を展開し、Applications を選択します。
- 3. [アプリケーションの作成]、[カスタムアプリケーション]の順に選択します。
- 4. [アプリケーション名] に、「CodeDeployGitHubDemo-App」と入力します。
- 5. [コンピューティングプラットフォーム] で [EC2/オンプレミス] を選択します。
- 6. [Create application] を選択します。
- 7. [デプロイグループ] タブで、[デプロイグループの作成] を選択します。
- 8. [Deployment group name] (デプロイグループ名) に「**CodeDeployGitHubDemo-DepGrp**」と入 力します。
- Service role で、<u>CodeDeploy のサービスロールを作成</u>で作成した CodeDeploy サービスロールの名前を選択します。
- 10. [デプロイタイプ] で、[インプレース] を選択します。
- Environment configuration で、使用するインスタンスのタイプに応じて、Amazon EC2 instances あるいは On-premises instances を選択します。[キー] と [値] に、ステップ 4: インス <u>タンスをプロビジョニングします。</u>の一部としてインスタンスに適用されたインスタンスタグ のキーと値を入力します。
- 12. [デプロイ設定] で [CodeDeployDefault.AllatOnce] を選択します。

- 13. [ロードバランサー] で、[Enable load balancing (ロードバランシングの有効化)] をオフにしま す。
- 14. [Advanced] を展開します。
- 15. [アラーム] で [アラーム設定を無視する] を選択します。
- 16. [デプロイグループの作成]を選択し、次のステップに進みます。

アプリケーションおよびデプロイグループの作成 (CLI)

 create-application のコマンドを呼び出して、CodeDeploy で CodeDeployGitHubDemo-App と 言う名前のアプリケーションを作成します。

aws deploy create-application --application-name CodeDeployGitHubDemo-App

- create-deployment-group コマンドを呼び出して CodeDeployGitHubDemo-DepGrp と言う名前のデプロイグループを作成します。
 - Amazon EC2 インスタンスにデプロイする場合、*ec2-tag-key*は、ステップ 4: インスタン スをプロビジョニングします。 EC2 インスタンスのタグキーです。
 - Amazon EC2 インスタンスにデプロイする場合、*ec2-tag-value* は、<u>ステップ 4: インス タンスをプロビジョニングします。</u>の一部として Amazon EC2 インスタンスに適用した Amazon EC2 インスタンスのタグ値です。
 - オンプレミスインスタンスにデプロイする場合、on-premises-tag-key は、「ステップ 4: インスタンスをプロビジョニングします。」の一部としてオンプレミスインスタンスに適用したオンプレミスインスタンスのタグキーです。
 - オンプレミスインスタンスにデプロイする場合、on-premises-tag-valueは、「ステップ <u>4: インスタンスをプロビジョニングします。</u>」の一部としてオンプレミスインスタンスに適用したオンプレミスインスタンスのタグ値です。
 - service-role-arn は、CodeDeployのサービスロールを作成で作成したサービスロールのサービスロール ARN です。(サービスロール ARNの取得 (CLI)の手順に従って、サービスロール ARN を見つけます)。

aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-App --ec2-tag-filters Key=ec2-tag-key,Type=KEY_AND_VALUE,Value=ec2-tag-value --onpremises-tag-filters Key=on-premises-tag-key,Type=KEY_AND_VALUE,Value=on-premises-

```
tag-value --deployment-group-name CodeDeployGitHubDemo-DepGrp --service-role-
arn service-role-arn
```

Note

<u>create-deployment-group</u> コマンドは、デプロイおよびインスタンス内の指定されたイベ ントについて、トピックサブスクライバーに Amazon SNS 通知を送信するトリガーの 作成をサポートします。このコマンドは、Amazon CloudWatch アラームのモニタリン グしきい値が満たされたときにデプロイを自動的にロールバックし、デプロイを停止す るアラームを設定するオプションもサポートします。このチュートリアルでは、これら のアクションコマンドは含まれていません。

ステップ 6: アプリケーションをインスタンスにデプロイします。

このステップでは、CodeDeploy コンソールまたは AWS CLI を使用して、GitHub リポジトリからイ ンスタンスにサンプルリビジョンをデプロイします。

リビジョンをデプロイするには (コンソール)

- 1. [デプロイグループの詳細]ページで、[デプロイの作成]を選択します。
- 2. [デプロイグループ] で [CodeDeployGitHubDemo-DepGrp] を選択します。
- 3. [リビジョンタイプ] で [GitHub] を選択します。
- 4. [Connect to GitHub] で、次のいずれかを実行します。
 - GitHub アカウントに対する CodeDeploy アプリケーションの接続を作成するには、ウェブブ ラウザの別のタブで GitHub からサインアウトします。[GitHub アカウント] に、この接続を識 別する名前を入力し、[GitHub に接続] を選択します。CodeDeployGitHubDemo-App という 名前のアプリケーションを GitHub で操作することを CodeDeploy に許可するよう求めるメッ セージがウェブページに表示されます。ステップ 5 に進みます。
 - 作成済みの接続を使用するには、その名前を [GitHub account] で選択してから [Connect to GitHub] を選択します。ステップ 7 に進みます。
 - 別の GitHub アカウントへの接続を作成するには、ウェブブラウザの別のタブで GitHub から サインアウトします。[Connect to a different GitHub account] を選択し、[Connect to GitHub] を選択します。ステップ5に進みます。
- 5. [Sign in] ページの手順に従って、GitHub アカウントにサインインします。

- 6. [Authorize application] ページで、[Authorize application] を選択します。
- CodeDeploy の Create deployment ページ上で、Repository name に、サインインに使用した GitHub のユーザー名に続いてスラッシュ (/)、アプリケーションリビジョンをプッシュしたリポ ジトリの名前 (例えば my-github-user-name/CodeDeployGitHubDemo など) を順に入力し ます。

入力する値が不確実な場合、または異なるリポジトリを指定する場合:

- a. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
- b. [Your repositories] で、ターゲットリポジトリの名前の上にマウスを置きます。GitHub ユー ザーまたは組織の名前、スラッシュ (/)、リポジトリの名前の順序でツールヒントが表示さ れます。この値を [Repository name (リポジトリ名)] に入力します。

Note

[Your repositories (自分のリポジトリ)] にターゲットリポジトリの名前が表示されな い場合は、[Search GitHub (GitHub の検索)] ボックスを使用して、ターゲットリポ ジトリと GitHub ユーザーまたは組織の名前を検索します。

8. [Commit ID (コミット ID)] ボックスに、アプリケーションリビジョンの GitHub へのプッシュに 関連付けられているコミットの ID を入力します。

入力する値が不確実な場合:

- a. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
- b. Your repositories で、CodeDeployGitHubDemo を選択します。
- c. コミットのリストで、アプリケーションリビジョンの GitHub へのプッシュに関連付けられ ているコミット ID を検索してコピーします。通常、この ID は 40 文字で、文字と数字の両 方で構成されます (コミット ID の短いバージョンを使用しないでください。通常は、長い バージョンの最初の 10 文字です)。
- d. [Commit ID] ボックスにコミット ID を貼り付けます。
- 9. [Deploy] を選択して、次のステップに進みます。

リビジョンをデプロイするには (CLI)

GitHub とやり取りする AWS CLI コマンド (次に呼び出す create-deployment コマンドなど) を呼び 出す前に、GitHub ユーザーアカウントを使用してCodeDeployGitHubDemo-Appアプリケーション の GitHub とやり取りするアクセス許可を CodeDeploy に付与する必要があります。現在、この操作 を行うには、CodeDeploy コンソールを使用する必要があります。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで Deploy を展開し、Applications を選択します。
- 3. CodeDeployGitHubDemo-App を選択します。
- 4. [デプロイ] タブで、[デプロイの作成] を選択します。

Note

新しいデプロイは作成されません。これは現在、GitHub ユーザーアカウントに代わって GitHub を操作するための権限を CodeDeploy に付与する唯一の方法です。

- 5. CodeDeployGitHubDemo-App から、CodeDeployGitHubDemo-DepGrp を選択します。
- 6. [リビジョンタイプ] で [GitHub] を選択します。
- 7. [Connect to GitHub] で、次のいずれかを実行します。
 - GitHub アカウントに対する CodeDeploy アプリケーションの接続を作成するには、ウェブブ ラウザの別のタブで GitHub からサインアウトします。[GitHub account] に、この接続を識別 する名前を入力し、[Connect to GitHub] を選択します。CodeDeployGitHubDemo-App と いう名前のアプリケーションの GitHub を操作することを CodeDeployに許可するよう求める メッセージがウェブページに表示されます。ステップ 8 に進みます。
 - 作成済みの接続を使用するには、その名前を [GitHub account] で選択してから [Connect to GitHub] を選択します。ステップ 10 に進みます。
 - 別の GitHub アカウントへの接続を作成するには、ウェブブラウザの別のタブで GitHub から サインアウトします。[Connect to a different GitHub account] を選択し、[Connect to GitHub] を選択します。ステップ 8 に進みます。
- 8. [Sign in] ページの手順に従って、GitHub のユーザー名、または E メールとパスワードでサイン インします。
- 9. [Authorize application] ページで、[Authorize application] を選択します。

- 10. CodeDeploy の Create deployment ページで、Cancel を選択します。
- 11. create-deployment コマンドを呼び出して、リビジョンを GitHub リポジトリからインスタンス にデプロイします。
 - *repository*は、GitHubアカウント名です。スラッシュ(/)、リポジトリの名前が後に続き ます (CodeDeployGitHubDemo)。例: MyGitHubUserName/CodeDeployGitHubDemo。

使用する値が不確実な場合、または異なるリポジトリを指定する場合:

- 1. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
- [Your repositories] で、ターゲットリポジトリの名前の上にマウスを置きます。GitHub ユーザーまたは組織の名前、スラッシュ (/)、リポジトリの名前の順序でツールヒントが 表示されます。これが使用する値です。

1 Note

[Your repositories] にターゲットリポジトリの名前が表示されない場合、[Search GitHub] ボックスを使用して、ターゲットリポジトリと、対応する GitHub ユー ザーまたは組織の名前を検索します。

 commit-id は、リポジトリにプッシュしたアプリケーションリビジョンのバージョンに関連 付けられているコミットです (例: f835159a...528eb76f)。

使用する値が不確実な場合:

- 1. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
- 2. Your repositories で、CodeDeployGitHubDemo を選択します。
- コミットのリストで、アプリケーションリビジョンの GitHub へのプッシュに関連付けら れているコミット ID を検索します。通常、この ID は 40 文字で、文字と数字の両方で構 成されます (コミット ID の短いバージョンを使用しないでください。通常は、長いバー ジョンの最初の 10 文字です)。この値を使用します。

ローカル Linux、macOS、あるいは Unix マシンを使用している場合

aws deploy create-deployment \

--application-name CodeDeployGitHubDemo-App \

--deployment-config-name CodeDeployDefault.OneAtATime \

--deployment-group-name CodeDeployGitHubDemo-DepGrp \

```
--description "My GitHub deployment demo" \
--github-location repository=repository,commitId=commit-id
```

ローカル Windows マシンを使用している場合

aws deploy create-deployment --application-name CodeDeployGitHubDemo-App -deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name CodeDeployGitHubDemo-DepGrp --description "My GitHub deployment demo" --githublocation repository=repository, commitId=commit-id

ステップ 7: デプロイをモニタリングおよび確認します。

このステップでは、CodeDeploy コンソールまたは AWS CLI を使用して、デプロイの成功を確認し ます。作成または設定したインスタンスにデプロイしたウェブページを表示するには、ウェブブラウ ザを使用します。

Note

Ubuntu インスタンスにデプロイする場合、独自のテスト戦略を使用して、デプロイされたリ ビジョンがインスタンスで予期したとおりに機能するかどうかを確認し、次のステップに進 みます。

デプロイをモニタリングおよび確認するには (コンソール)

- 1. ナビゲーションペインで Deploy を展開し、Deployments を選択します。
- デプロイのリストで、CodeDeployGitHubDemo-App の Application の値、および CodeDeployGitHubDemo-DepGrp の CodeDeployGitHubDemo-DepGrp の値が [-DepGrp] であ る行を探します。Succeeded または Failed が、Status の列に 表示されない場合は、定期的に Refresh のボタンを選択します。
- 3. Status の列に Failed が表示される場合は、<u>インスタンスの詳細の表示 (コンソール)</u>の指示に 従って、デプロイのトラブルシューティングを行います。
- Status 列で Succeeded が表示される場合は、ウェブブラウザを通してデプロイを確認できます。このサンプルリビジョンでは、インスタンスに単一のウェブページをデプロイします。Amazon EC2 インスタンスにデプロイする場合は、ウェブブラウ

ザで、インスタンスの http://*public-dns* にアクセスします (例えば、http://ec2-01-234-567-890.compute-1.amazonaws.com など)。

5. ウェブページを表示できれば成功です。 AWS CodeDeploy を使用して GitHub からリビジョン が正しくデプロイされました。これで、「<u>ステップ 8: クリーンアップする</u>」に進むことができ ます。

デプロイをモニタリングおよび確認するには (CLI)

1. list-deployments コマンドを呼び出して、CodeDeployGitHubDemo-App という名前のアプリ ケーションと CodeDeployGitHubDemo-DepGrp という名前のデプロイグループのデプロイ ID を取得します。

aws deploy list-deployments --application-name CodeDeployGitHubDemo-App -deployment-group-name CodeDeployGitHubDemo-DepGrp --query "deployments" --output text

get-deployment コマンドを呼び出して、list-deployments コマンドからの出力にデプロイメントの ID を指定します。

aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text

- [Failed] が返されたら、<u>インスタンスの詳細の表示 (コンソール)</u>の指示に従って、デプロイのト ラブルシューティングを行います。
- [Succeeded] が返されたら、ウェブブラウザでデプロイを確認できます。このサンプルリビジョンは、インスタンスにデプロイされた単一のウェブページです。Amazon EC2 インスタンスにデプロイする場合は、Amazon EC2 インスタンスのための http://public-dnsにアクセスすることで、このページをウェブブラウザで表示できます (たとえば、http://ec2-01-234-567-890.compute-1.amazonaws.com など)。
- 5. ウェブページを表示できれば成功です。 AWS CodeDeploy を使用して GitHub リポジトリから 正常にデプロイされました。

ステップ 8: クリーンアップする

このチュートリアルの間に使用したリソースに対する追加料金を防ぐため、Amazon EC2 インスタ ンスと関連リソースを終了する必要があります。必要に応じて、このチュートリアルに関連付けられ ている CodeDeploy デプロイコンポーネントレコードを削除できます。このチュートリアル専用に GitHub リポジトリを使用した場合は、今すぐ削除することもできます。

AWS CloudFormation スタックを削除するには(AWS CloudFormation テンプレート を使用して Amazon EC2 インスタンスを作成した場合)

- 1. にサインイン AWS Management Console し、 AWS CloudFormation コンソールを <u>https://</u> console.aws.amazon.com/cloudformation://www.com で開きます。
- 2. [スタック] 列で、CodeDeploySampleStack で始まるスタックを選択します。
- 3. [削除]を選択します。
- プロンプトが表示されたら、[スタックの削除] を選択します。Amazon EC2 インスタンスおよび
 関連する IAM インスタンスプロファイルとサービスロールが削除されます。

手動で登録を解除およびオンプレミスインスタンスをクリーンアップするには (オン プレミスインスタンスをプロビジョニングした場合)

を使用して AWS CLI、ここで your-instance-name で表されるオンプレミスインスタンスと、your-region で関連付けられたリージョンに対して登録解除コマンドを呼び出します。

aws deploy deregister --instance-name *your-instance-name* --no-delete-iam-user -- region *your-region*

2. オンプレミスインスタンスから、uninstall コマンドを呼び出します:

aws deploy uninstall

Amazon EC2 インスタンスを手動で終了するには (手動で Amazon EC2 インスタンス を起動した場合)

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- ナビゲーションペインの [Instances] (インスタンス) で、[Instances] (インスタンス) を選択します。
- 3. 終了した Amazon EC2 インスタンスの横にあるボックスを選択します。[Actions] メニューで [Instance State] をポイントし、[Terminate] を選択します。

4. プロンプトが表示されたら、[Yes, Terminate] を選択します。

CodeDeploy デプロイコンポーネントレコードを削除するには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで Deploy を展開し、Applications を選択します。
- 3. CodeDeployGitHubDemo-App を選択します。
- 4. [アプリケーションを削除]を選択します。
- 5. 確認を求めるメッセージが表示されたら、Deleteと入力し、[削除]を選択してください。

GitHub リポジトリを削除するには

<u>GitHub help</u>の <u>Deleting a repository</u> を参照してください。

チュートリアル: Amazon ECS ヘアプリケーションをデプロイする

このチュートリアルでは、CodeDeploy を使用して アプリケーションを Amazon ECS にデプロイ する方法について学びます。すでに作成し、Amazon ECS にデプロイ済みのアプリケーションに よって開始します。最初のステップは、タスク定義ファイルを新しいタグで変更してアプリケー ションを更新することです。次に、CodeDeploy を使用して、更新をデプロイします。デプロイ中 に、CodeDeploy は新しい置き換えタスクセットに更新をインストールします。そして、本稼働トラ フィックは、元のタスクセットにある Amazon ECS アプリケーションの元のバージョンから、置き 換えタスクセットの更新されたバージョンに移行します。

Amazon ECS デプロイ中、CodeDeploy は 2 つのターゲットグループと 1 つの本稼働トラフィック リスナーで構成されたロードバランサーを使用します。次の図表は、デプロイが始まるの前に、ロー ドバランサー、本稼働、リスナー、ターゲットグループ、および Amazon ECS アプリケーションが どのように関連しているかを示しています。このチュートリアルでは、Application Load Balancer を 使用します。Network Load Balancer を使用することもできます。



デプロイが成功すると、本稼働トラフィックリスナーは新しい置き換えタスクセットにトラフィック を提供し、元のタスクセットは終了します。次の図は、デプロイが成功した後にリソースがどのよう に関連しているかを示しています。詳細については、「<u>Amazon ECS デプロイ中の処理で起こって</u> <u>いること</u>」を参照してください。



を使用してアプリケーションを Amazon ECS に AWS CLI デプロイする方法については、<u>「チュートリアル: Blue/Green デプロイを使用したサービス</u>の作成」を参照してください。CodePipeline を 使用して、CodeDeploy を用いての Amazon ECS サービスへの deploy の変更を検出し、自動的に デプロイする方法については、<u>チュートリアル: Create a pipeline with an Amazon ECR source and</u> ECS-to-CodeDeploy deployment を参照してください。

このチュートリアルを完了したら、作成した CodeDeploy アプリケーションとデプロイグループを 使用して、<u>チュートリアル: 検証テストを使用して Amazon ECS サービスをデプロイする</u> 中にデプ ロイ検証テストを追加できます。

トピック

- 前提条件
- ステップ 1: Amazon ECS アプリケーションを更新する
- ステップ 2: AppSpec ファイルを作成します。
- ステップ 3: CodeDeploy コンソールを使用してアプリケーションをデプロイする
- ステップ 4: クリーンアップする

前提条件

このチュートリアルを完了するには、まず以下を行う必要があります。

- 「CodeDeploy の開始方法」のステップ 2 と 3 を完了します。
- 2 つのターゲットグループと 1 つのリスナーを用いて設定した Application Load Balancer を作成します。コンソールを使用してロードバランサーを作成する方法については、「<u>CodeDeploy</u> <u>Amazon ECS デプロイ用のロードバランサー、ターゲットグループ、リスナーをセットアッ</u> <u>プする</u>」を参照してください。を使用してロードバランサーを作成する方法については AWS CLI、「Amazon Elastic Container Service ユーザーガイド」の「ステップ 1: Application Load <u>Balancer を作成する</u>」を参照してください。ロードバランサーを作成するときは、このチュート リアルで以下の点に注意してください。
 - ロードバランサーの名前。
 - ターゲットグループの名前。
 - ロードバランサーのリスナーが使用するポート。
- Amazon ECS クラスターとサービスを作成します。さらなる詳細については、ステップ 2、3、 および Amazon Elastic Container サービスユーザーガイド の チュートリアル: Creating a service

<u>using a blue/green deployment</u> の ステップ 4 を参照してください。このチュートリアルでは、以 下の点に注意してください。

- Amazon ECS クラスターの名前。
- Amazon ECS サービスによって使用されるタスク定義の ARN
- Amazon ECS サービスによって使用されるコンテナの名前
- AppSpec ファイル用の Amazon S3 バケットを作成します。

ステップ 1: Amazon ECS アプリケーションを更新する

このセクションでは、タスク定義の新しいリビジョンで Amazon ECS アプリケーションを更新しま す。更新されたリビジョンは、新しいキーとタグのペアを追加します。 <u>ステップ 3: CodeDeploy コ</u> <u>ンソールを使用してアプリケーションをデプロイする</u> では、Amazon ECS アプリケーションの更新 バージョンをデプロイします。

タスク定義を更新するには

- 1. コンソール (https://console.aws.amazon.com/ecs/v2) を開きます。
- 2. ナビゲーションペインで、[タスク定義]を選択します。
- 3. Amazon ECS サービスによって使用されるタスク定義を選択します。
- 4. タスク定義リビジョンを選択し、[新しいリビジョンを作成]、[新しいリビジョンを作成] を選択 します。
- 5. このチュートリアルでは、タグを追加してタスク定義を少し更新します。ページの下部にある [タグ] で、新しいキーと値のペアを入力して新しいタグを作成します。
- 6. [Create] (作成)を選択します。

タスク定義のリビジョン番号が1つずつ増加します。

- 7. [JSON] タブを選択します。この情報は次のステップで必要なため、以下の点に注意してください。
 - taskDefinitionArnの値。形式は arn:aws:ecs:aws-region:account-id:taskdefinition/task-definition-family:task-definition-revisionです。これ は、更新されたタスク定義の ARN です。
 - containerDefinitions 要素の、name の値。これはコンテナの名前です。
 - portMappings 要素の、containerPort の値。これはコンテナのポートです。

ステップ 2: AppSpec ファイルを作成します。

このセクションでは、AppSpec ファイルを作成し、<u>前提条件</u>のセクションで作成した Amazon S3 バケットにアップロードします。Amazon ECS デプロイのための AppSpec ファイル は、タスク定 義、コンテナ名、およびコンテナポートを指定します。詳細については、<u>Amazon ECS デプロイの</u> <u>AppSpec ファイルの例</u>および<u>Amazon ECS デプロイ用の AppSpec の「resources」セクション</u>を 参照してください。

AppSpec ファイルを作成するには

- [YAML] を使用して AppSpec ファイルを作成したい場合は、appspec.yml という名前のファ イルを作成します。JSON を使用して AppSpec ファイルを作成したい場合は、appspec.json という名前のファイルを作成します。
- AppSpec ファイルのために YAML または JSON を使用するかどうかに応じて、適切 なタブを選択し、そのコンテンツを、先ほど作成した AppSpec ファイルにコピーしま す。TaskDefinition プロパティには、「<u>ステップ 1: Amazon ECS アプリケーションを更新</u> <u>する</u>」セクションで書き留めたタスク定義 ARN を使用します。

JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    ſ
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "your-container-name",
            "ContainerPort": your-container-port
          }
        }
      }
    }
  ]
}
```

```
version: 0.0
Resources:
    TargetService:
    Type: AWS::ECS::Service
    Properties:
    TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number"
    LoadBalancerInfo:
        ContainerName: "your-container-name"
        ContainerPort: your-container-port
```

Note

置き換えタスクセットは、元のタスクセットからサブネット、セキュリティグルー プ、プラットフォームバージョン、割り当てられたパブリック IP 値を継承しま す。AppSpec ファイル中でオプションのプロパティを設定することで、置き換えタスク セットのこれらの値を上書きできます。詳細については、<u>Amazon ECS デプロイ用の</u> <u>AppSpec の「resources」セクション</u>および<u>Amazon ECS デプロイの AppSpec ファイ</u> ルの例 を参照してください。

3. このチュートリアルの前提条件として、作成済みの S3 バケットに AppSpec ファイルをアップ ロードします。

ステップ 3: CodeDeploy コンソールを使用してアプリケーションをデプロ イする

このセクションでは、CodeDeploy アプリケーションとデプロイグループを作成して、更新されたア プリケーションを Amazon ECS にデプロイします。デプロイ中、CodeDeploy はアプリケーション の本稼働トラフィックを、新しい置き換えタスクセットの新しいバージョンに移行します。このス テップを完了するには、以下の項目が必要です。

- Amazon ECS クラスターの名前。
- Amazon ECS サービスの名前。
- ・ Application Load Balancer の名前

- 本稼働リスナーポート。
- ターゲットグループ名。
- 作成した S3 バケットの名前。

CodeDeploy でアプリケーションを作成するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy/</u>:// www.com」で CodeDeploy コンソールを開きます。
- 2. [Create application] を選択します。
- 3. [アプリケーション名] に、「ecs-demo-codedeploy-app」と入力します。
- 4. [コンピューティングプラットフォーム] で [Amazon ECS] を選択します。
- 5. [Create application] を選択します。

CodeDeploy デプロイ グループを作成するには

- アプリケーションのページの [デプロイグループ] タブで、[デプロイグループの作成] を選択します。
- 2. [Deployment group name] (デプロイグループ名) に「ecs-demo-dg」と入力します。
- サービスロール で、CodeDeploy に Amazon ECS へのアクセスを許可するサービスロールを選 択します。詳細については、「<u>AWS CodeDeployのためのアイデンティティおよびアクセス管</u> 理」を参照してください。
- 4. 環境設定 で、Amazon ECS クラスターの名前とサービス名を選択します。
- 5. Load balancers から、Amazon ECSサービスにトラフィックを提供するロードバランサーの名 前を選択します。
- Production listener port (本稼働リスナーポート)から、Amazon ECSサービスへの本稼働トラ フィックを提供するリスナーのポートとプロトコルを選択します(例: HTTP: 80)。このチュー トリアルにはオプションのテストリスナーが含まれていないため、[Test listener port (リスナー ポートをテスト)]からポートを選択しないでください。
- [Target group 1 name (ターゲットグループ 1 の名前)] および [Target group 2 name (ターゲットグループ 2 の名前)] から、デプロイ時にトラフィックをルーティングする 2 つの異なるターゲットグループを選択します。これらが、ロードバランサー用に作成したターゲットグループであることを確認します。どちらがターゲットグループ 1 に使用され、どちらがターゲットグループ 2 に使用されるかは関係ありません。
- 8. [Reroute traffic immediately (すぐにトラフィックを再ルーティングする)]を選択します。

9. [Original revision termination (元のリビジョンの終了)] で、0 日、0 時間、5 分を選択します。こ れにより、デフォルトの(1 時間)を使用するよりも迅速にデプロイが完了したことがわかりま す。

Choose an ECS cluster name	1								
ecs-tutorial-cluster	1					•			
	1								
Choose an ECS service name	+					_			
ecs-demo-service						•			
t and the law same									
Load balancers									
Choose a load balancer	1								
ecs-demo-alb						•			
Production listener port	-					_			
HTTP: 80	_					•			
Target group 1 name	1					-			
ecs-demo-tg-1						•			
Target group 2 name	í								
Target group 2 name ecs-demo-tg-2	<u>[</u>					•			
Target group 2 name ecs-demo-tg-2 Deployment settings]					•			
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t	tt e replace	ement environ	ment immediat	ely or waits for y	you to s	▼	rerouting pi	rocess	
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t Reroute traffic immediately	tt e replace	ement environ	ment immediat	ely or waits for y	you to s	▼ tart the r	rerouting pr	ocess	
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t Reroute traffic immediately Specify when to reroute traffic	the replace y affic	ement environ	ment immediat	ely or waits for y	/ou to s	▼ tart the r	rerouting pr	rocess	
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t Reroute traffic immediately Specify when to reroute traffic Deployment Configuration	the replace y affic	ement environ	ment immediat	ely or waits for y	/ou to s	▼ tart the r	erouting p	rocess	
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t Reroute traffic immediately Specify when to reroute traffic Deployment Configuration CodeDeployDefault.ECSALLAto	the replace y affic Once	ement environ	ment immediat	ely or waits for y	/ou to s	▼ tart the r	erouting p	rocess	
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t • Reroute traffic immediately • Specify when to reroute traffic of the configuration CodeDeployDefault.ECSALLAtto Original revision termination Specify how long CodeDeploy waits automatically	ti e replace y affic Once s before it	ement environ	ment immediat	ely or waits for y	you to s	▼ tart the r	rerouting pl	ocess	y or
Target group 2 name ecs-demo-tg-2 Deployment settings Traffic rerouting Choose whether traffic reroutes to t Reroute traffic immediately Specify when to reroute traffic Deployment Configuration CodeDeployDefault.ECSALLAto Original revision termination Specify how long CodeDeploy waits automatically Days	the replace y affic Once s before it	ement environ terminates the Hours	ment immediat	ely or waits for y et. After termina	you to s ation sta	The second se	rerouting pr	ocess	y or

10. デプロイグループの作成を選択します。

Amazon ECS アプリケーションをデプロイするには

- 1. デプロイグループのコンソールページで、[デプロイの作成]を選択します。
- 2. [デプロイグループ] で、[ecs-demo-dg] を選択します。
- [Revision type (リビジョンのタイプ)]の横の [My application is stored in Amazon S3 (Amazon S3 に保存されているアプリケーション)]を選択します。[リビジョンの場所] に、S3 バケットの名前を入力します。
- 4. [リビジョンファイルタイプ] で、必要に応じて[.json] または [.yaml] を選択します。
- 5. (オプション)[デプロイの説明] に、デプロイの説明を入力します。
- 6. [デプロイの作成]を選択します。
- [Deployment status (デプロイのステータス)] で、デプロイをモニタリングできます。本稼働ト ラフィックの 100% が置き換えタスクセットにルーティングされた後、5 分の待機時間が期限 切れになる前に、[Terminate original task set (元のタスクセットの終了)] を選択して元のタスク セットをすぐに終了できます。[Terminate original task set (元のタスクセットの終了)] を選択し ない場合、指定した5 分の待機時間が経過すると元のタスクセットが終了します。

d-MVGEP9PSM	Stop deployment	Stop a	and roll back deployment	Terminate original task set
Deployment status			Traffic shifting prog	gress
Step 1: Deploying replacement task	Completed		Original	Replacement
Step 2:	Suc	ceeded	0%	100%
Rerouting production traffic t replacement task set	o 100% traffic shifted Suc	ceeded	Original task set not serving traffic	Replacement task set serving traffic
Step 3: Wait 5 minutes 0 seconds	Waiting	rooress		
Step 4: Terminate original task set	Not started	in ogi Caa		
	ln p	orogress		

ステップ 4: クリーンアップする

次のチュートリアル、<u>チュートリアル:検証テストを使用して Amazon ECS サービスをデプロイす</u> <u>る</u> で、このチュートリアルを基盤として構築し、作成した CodeDeploy アプリケーションとデプロ イグループを使用します。そのチュートリアルのステップを実行する場合は、このステップをスキッ プし、作成したリソースを削除しないでください。

Note

AWS アカウントでは、作成した CodeDeploy リソースに対して料金は発生しません。

これらのステップのリソース名は、このチュートリアルで提案されている名前です(例え ば、CodeDeploy アプリケーションの名前のための ecs-demo-codedeploy-app など)。別の名 前を使用した場合は、クリーンアップ時にそれらを使用してください。

1. [デプロイグループの削除]コマンドを使用して、CodeDeploy デプロイグループを削除します。

aws deploy delete-deployment-group --application-name *ecs-demo-codedeploy-app* -- deployment-group-name *ecs-demo-dg* --region *aws-region-id*

2. [アプリケーションの削除]コマンドを使用して、 CodeDeploy アプリケーションを削除します。

aws deploy delete-application --application-name ecs-demo-codedeploy-app -region aws-region-id

チュートリアル: 検証テストを使用して Amazon ECS サービスを デプロイする

このチュートリアルでは、Lambda 関数を使用して、更新された Amazon ECS アプリケーションの デプロイの一部を検証する方法について学びます。このチュートリアルでは、CodeDeploy アプリ ケーション、CodeDeploy デプロイグループ、および <u>チュートリアル: Amazon ECS へアプリケー</u> <u>ションをデプロイする</u> で使用した Amazon ECS アプリケーションを使用します。そのチュートリア ルを完了してからこのチュートリアルを開始してください。

検証テストを追加するには、最初に Lambda 関数でテストを実装します。次に、デプロイ AppSpec ファイルで、テストしたい ライフサイクルフックのために Lambda 関数を指定します。検証テスト が失敗した場合、デプロイは停止し、ロールバックされて、失敗とマークされます。テストが成功す ると、デプロイは次のデプロイライフサイクルイベントまたはフックに進みます。

検証テストを使用した Amazon ECS デプロイ中、CodeDeploy は 2 つのターゲットグループ (1 つ の本稼働トラフィックリスナーと 1 つのテストトラフィックリスナー) で構成されたロードバラン サーを使用します。次の図表は、デプロイの前に、ロードバランサー、本稼働およびテストリス ナー、ターゲットグループ、および Amazon ECS アプリケーションがどのように関連しているかを 示しています。このチュートリアルでは、Application Load Balancer を使用します。Network Load Balancer を使用することもできます。



Amazon ECS デプロイ中、テスト用の5つのライフサイクルフックがあります。このチュートリア ルでは、3番目のライフサイクルデプロイフック、AfterAllowTestTraffic 中に1つのテストを 実装します。詳細については、「<u>Amazon ECS のデプロイ向けのライフサイクルイベントフックの</u> <u>リスト</u>」を参照してください。デプロイが成功すると、本稼働トラフィックリスナーは新しい置き換 えタスクセットにトラフィックを提供し、元のタスクセットは終了します。次の図は、デプロイが成 功した後にリソースがどのように関連しているかを示しています。詳細については、「<u>Amazon ECS</u> デプロイ中の処理で起こっていること」を参照してください。



Note

このチュートリアルを完了すると、 AWS アカウントに料金が発生する可能性があります。 これには、CodeDeploy、 AWS Lambda、CloudWatch の料金が含まれます。さらなる詳細 については、<u>AWS CodeDeploy の料金</u>、<u>AWS Lambda の料金</u>、および <u>Amazon CloudWatch</u> の料金 を参照してください。

トピック

- 前提条件
- ステップ 1: テストリスナーを作成する
- ・ ステップ 2: Amazon ECS アプリケーションを更新する
- ステップ 3: ライフサイクルフック Lambda 関数を作成する
- ステップ 4: AppSpec ファイルを更新する
- <u>ステップ 5: CodeDeploy コンソールを使用して Amazon ECS サービスをデプロイする</u>
- ステップ 6: CloudWatch Logs で Lambda フック関数の出力を表示する
- <u>ステップ 7: クリーンアップする</u>

前提条件

このチュートリアルを正常に完了するには、まず以下を行う必要があります。

- <u>前提条件</u>にある <u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイする</u>の前提条件を 完了します。
- ・「<u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイする</u>」のステップを完了します。 以下を書き留めます。
 - ロードバランサーの名前。
 - ターゲットグループの名前。
 - ロードバランサーのリスナーが使用するポート。
 - ・ ロードバランサーの ARN。これを使用して新しいリスナーを作成します。
 - いずれかのターゲットグループの ARN。これを使用して新しいリスナーを作成します。
 - ・作成する CodeDeploy アプリケーションとデプロイグループ。
 - CodeDeploy デプロイで使用されることによって作成する AppSpec ファイル このチュートリア ルでは、このファイルを編集します。

ステップ 1: テストリスナーを作成する

検証テストを使用する Amazon ECS デプロイには、2 番目のリスナーが必要です。このリスナー は、置き換えタスクセット中の更新された Amazon ECS アプリケーションにテストトラフィックを 提供するために使用されます。検証テストは、テストトラフィックに対して実行されます。

テストトラフィックのリスナーは、いずれのターゲットグループも使用できます。<u>create-listener</u> AWS CLI コマンドを使用して、テストトラフィックをポート 8080 に転送するデフォルトルールを 持つ 2 番目のリスナーを作成します。ロードバランサーの ARN といずれかのターゲットグループの ARN を使用します。

```
aws elbv2 create-listener --load-balancer-arn your-load-balancer-arn \
--protocol HTTP --port 8080 \
--default-actions Type=forward,TargetGroupArn=your-target-group-arn --region your-aws-
region
```

ステップ 2: Amazon ECS アプリケーションを更新する

このセクションでは、タスク定義の新しいリビジョンを使用するために Amazon ECS アプリケー ションを更新します。新しいリビジョンを作成し、タグを追加することでマイナー更新を追加しま す。

タスク定義を更新するには

- 1. Amazon ECS クラシックコンソール (https://console.aws.amazon.com/ecs/) を開きます。
- 2. ナビゲーションペインで、[タスク定義]を選択します。
- 3. Amazon ECS サービスで使用されるタスク定義のチェックボックスを選択します。
- 4. [Create new revision (新しいリビジョンを作成)] を選択します。
- 5. タグを追加して、タスク定義を少し更新します。ページの下部にある [タグ] で、新しいキーと 値のペアを入力して新しいタグを作成します。
- 6. [Create] (作成) を選択します。タスク定義のリビジョン番号が1増えたことがわかります。
- [JSON] タブを選択します。[taskDefinitionArn] の値を書き留めておきます。形式は arn:aws:ecs:aws-region: account-id:task-definition/task-definitionfamily: task-definition-revision です。これは、更新されたタスク定義の ARN で す。

ステップ 3: ライフサイクルフック Lambda 関数を作成する

このセクションでは、Amazon ECS デプロイの AfterAllowTestTraffic のフック用に 1 つの Lambda 関数を実装します。Lambda 関数は、更新された Amazon ECS アプリケーションがインス トールされる前に検証テストを実行します。このチュートリアルでは、Lambda 関数は Succeeded を返します。実際のデプロイ中、検証テストの結果に応じて、検証テストは Succeeded また は Failed を返します。また、実際のデプロイ中に、他の Amazon ECS デプロイライフサイ クルイベントフック (BeforeInstall、AfterInstall、BeforeAllowTraffic、および AfterAllowTraffic) に 1 つ以上の Lambda テスト関数を実装することもできます。詳細につい ては、「<u>Amazon ECS のデプロイ向けのライフサイクルイベントフックのリスト</u>」を参照してくだ さい。

Lambda 関数を作成するには、IAM ロールが必要です。ロールは、CloudWatch Logs に書き込む Lambda 関数を許可し、CodeDeploy ライフサイクルフックのステータスを設定します。

IAM ロールを作成するには

- 1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
- 2. ナビゲーションペインで [ロール] を選択し、[ロールを作成] を選択します。
- 3. 次のプロパティでロールを作成します。
 - ・ 信頼されたエンティティ]: AWS Lambda。
 - [アクセス許可]: [AWSLambdaBasicExecutionRole]。これにより、CloudWatch Logs に書き込むアクセス許可が Lambda 関数に付与されます。
 - ・ ロール名]: lambda-cli-hook-role。

詳細については、AWS Lambda 「実行ロールの作成」を参照してください。

 作成したロールにアクセス許可 codedeploy:PutLifecycleEventHookExecutionStatus をアタッチします。これにより、デプロイ中に CodeDeploy ライフサイクルフックのステータ スを設定する Lambda 関数の許可が付与されます。詳細については、AWS Identity and Access Management ユーザーガイド の Adding IAM identity permissions、および CodeDeploy API Reference の PutLifecycleEventHookExecutionStatus を参照してください。

AfterAllowTestTrafficのフック Lambda 関数を作成するには

1. 次の内容で、AfterAllowTestTraffic.jsという名前のファイルを作成します。

```
'use strict';
const AWS = require('aws-sdk');
const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
exports.handler = (event, context, callback) => {
  console.log("Entering AfterAllowTestTraffic hook.");
  // Read the DeploymentId and LifecycleEventHookExecutionId from the event payload
  var deploymentId = event.DeploymentId;
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
  var validationTestResult = "Failed";
  // Perform AfterAllowTestTraffic validation tests here. Set the test result
  // to "Succeeded" for this tutorial.
  console.log("This is where AfterAllowTestTraffic validation tests happen.")
```

```
validationTestResult = "Succeeded";
// Complete the AfterAllowTestTraffic hook by sending CodeDeploy the validation
status
var params = {
 deploymentId: deploymentId,
 lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
 status: validationTestResult // status can be 'Succeeded' or 'Failed'
};
// Pass CodeDeploy the prepared validation test results.
codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
 if (err) {
  // Validation failed.
  console.log('AfterAllowTestTraffic validation tests failed');
  console.log(err, err.stack);
  callback("CodeDeploy Status update failed");
  } else {
  // Validation succeeded.
  console.log("AfterAllowTestTraffic validation tests succeeded");
  callback(null, "AfterAllowTestTraffic validation tests succeeded");
 }
});
}
```

2. Lambda デプロイパッケージを作成する

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

 create-functionのコマンドを使用して、AfterAllowTestTrafficのフックのために Lambda 関数を作成します。

```
aws lambda create-function --function-name AfterAllowTestTraffic \
    --zip-file fileb://AfterAllowTestTraffic.zip \
    --handler AfterAllowTestTraffic.handler \
    --runtime nodejs10.x \
    --role arn:aws:iam::aws-account-id:role/lambda-cli-hook-role
```

4. create-function のレスポンスにある Lambda 関数の ARN を書き留めます。この ARN は、 次のステップで CodeDeploy デプロイの AppSpec ファイルを更新するときに使用します。

ステップ 4: AppSpec ファイルを更新する

このセクションでは、Hooks のセクションを使用して AppSpec ファイルを更新します。Hooks の セクションで、AfterAllowTestTraffic のライフサイクルフックのための Lambda 関数を指定 します。

AppSpec ファイルを更新するには

- <u>ステップ 2: AppSpec ファイルを作成します。</u>の <u>チュートリアル: Amazon ECS ヘアプリケー</u> <u>ションをデプロイする</u>で作成した AppSpec ファイルファイルを開きます。
- TaskDefinition プロパティを、「<u>ステップ 2: Amazon ECS アプリケーションを更新する</u>」
 でメモしたタスク定義 ARN で更新します。
- コピーアンドペーストして、Hooks のセクションを AppSpec ファイルファイルに追加します。ARN を AfterAllowTestTraffic でメモした Lambda 関数の ARN を用いて <u>ステップ 3:</u> ライフサイクルフック Lambda 関数を作成する の後の ARN を更新します。

JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "sample-website",
            "ContainerPort": 80
          }
        }
      }
    }
  ],
  "Hooks": [
    {
      "AfterAllowTestTraffic": "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
    }
  ]
```

}

YAML AppSpec

```
version: 0.0
Resources:
    TargetService:
    Type: AWS::ECS::Service
    Properties:
    TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number"
    LoadBalancerInfo:
        ContainerName: "sample-website"
        ContainerPort: 80
Hooks:
        AfterAllowTestTraffic: "arn:aws:lambda:aws-region-id:aws-account-id::function:AfterAllowTestTraffic"
```

4. AppSpec ファイルを保存し、S3 バケットにアップロードします。

ステップ 5: CodeDeploy コンソールを使用して Amazon ECS サービスをデ プロイする

このセクションでは、テストリスナーのポートを指定して、デプロイグループを更新しま す。これは、「<u>ステップ 1: テストリスナーを作成する</u>」で作成したリスナーです。デプロイ 中、CodeDeploy は、テストリスナーを使用して置き換えタスクセットに提供されるテストトラ フィックを使用して、AfterAllowTestTraffic のデプロイライフサイクルフックの間に、検証 テストを実行します。検証テストによって結果 Succeeded が返されるため、デプロイは次のデプ ロイライフサイクルイベントに進みます。実際のシナリオでは、テスト関数は Succeeded または Failed を返します。

デプロイグループにテストリスナーを追加するには

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy/</u>:// www.com」で CodeDeploy コンソールを開きます。
- 2. ナビゲーションペインで、[アプリケーション] を選択します。
- 3. 「<u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイする</u>」で作成したアプリケー ションを選択します。提案された名前を使用した場合は、[ecs-demo-codedeploy-app] です。

- 4. [デプロイグループ] で、先ほど <u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイす</u> <u>る</u> で作成したデプロイグループを選択します。推奨された名前を使用した場合は [ecs-demo-dg] です。
- 5. [編集]を選択します。
- 6. [Test listener port (テストリスナーポート)] から、このチュートリアルで前に作成したテストリ スナーのポートとプロトコルを選択します。これは [HTTP:8080] である必要があります。
- 7. [Save changes] (変更の保存) をクリックします。

Amazon ECS アプリケーションをデプロイするには

- 1. デプロイグループのコンソールページで、[デプロイの作成]を選択します。
- 2. [デプロイグループ] で、[ecs-demo-dg] を選択します。
- [Revision type (リビジョンのタイプ)] の場合は、[My application is stored in Amazon S3 (Amazon S3 に保存されているアプリケーション)] を選択します。リビジョンの場所 に、S3 バ ケットの名前と AppSpec ファイル (例: s3://my-s3-bucket/appspec.json) を入力しま す。
- 4. [リビジョンファイルの種類] で、必要に応じて [.json] または [.yaml] を選択します。
- 5. (オプション)[デプロイの説明] に、デプロイの説明を入力します。
- 6. [デプロイの作成]を選択します。

[Deployment status (デプロイのステータス)] で、デプロイをモニタリングできます。本稼働トラ フィックの 100% が置き換えタスクセットにルーティングされた後、[Terminate original task set (元 のタスクセットの終了)] を選択して元のタスクセットをすぐに終了できます。[Terminate original task set (元のタスクセットの終了)] を選択しない場合、元のタスクセットはデプロイグループの作成 時に指定した期間後に終了します。

Stop deployment Stop and roll back deployment	Terminate original task	set
Deployment status	Traffic shifting pro	gress
Step 1:	Original	Replacement
Deploying replacement task Completed		
Succeeded	0%	100%
Step 2:		
Test traffic route setup Completed	Original task set not	Replacement task set
Sten 3:	serving trame	serving traffic
Rerouting production traffic to 100% traffic		
replacement task set shifted		
Stop 4:		
Wait 5 minutes 0 seconds Waiting		
 In progress 		
Step 5:		
Terminate original task set Not started		
 In progress 		

ステップ 6: CloudWatch Logs で Lambda フック関数の出力を表示する

CodeDeploy デプロイが成功すると、Lambda フック関数での検証テストも成功します。これを確認 するには、CloudWatch Logs にあるフック関数のログを確認します。

- 1. CloudWatch コンソール (<u>https://console.aws.amazon.com/cloudwatch/</u>) を開きます。
- 2. ナビゲーションペインで、[Logs (ログ)] を選択します。AppSpec ファイル で指定した Lambda フック関数の新しいロググループが 1 つ表示されます。

Filter: Log Group Name Prefix	×		≪ ≪ L	og Groups 1-1 🔉
Log Groups	Insights	Expire Events After	Metric Filters	Subscriptions
O /aws/lambda/AfterAllowTestTraffic	Explore	Never Expire	0 filters	None

- 3. 新しいロググループを選択します。これは /aws/lambda/AfterAllowTestTrafficHook である必要が あります。
- ログストリームを選択します。複数のログストリームが表示された場合は、[最後のイベント時間] で最新の日付と時刻のログストリームを選択します。
- 5. ログストリームイベントを展開し、Lambda フック関数がログに成功メッセージを書き込んだこ とを確認します。以下は、 AfterAllowTraffic の Lambda フック関数が成功したことを示し ます。

	Time (UTC +00:00)	Message
	2019-09-11	
		No older ev
•	20:11:20	START RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916 Version: \$LATEST
STA	RT RequestId: e875485b-c	db2-4e1e-b4e8-8054e7b7f916 Version: \$LATEST
-	20:11:21	2019-09-11T20:11:21.033Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO Entering AfterAllowTestTraffic hook.
201	9-09-11T20:11:21.033Z e8	75485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO Entering AfterAllowTestTraffic hook.
-	20:11:21	2019-09-11T20:11:21.034Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO This is where AfterAllowTestTraffic validation tests happen.
201	9-09-11T20:11:21.034Z e8	75485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO This is where AfterAllowTestTraffic validation tests happen.
-	20:11:21	2019-09-11T20:11:21.789Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO AfterAllowTestTraffic validation tests succeeded
201	9-09-11T20:11:21.789Z e8	75485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO AfterAllowTestTraffic validation tests succeeded
•	20:11:21	END RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916
•	20:11:21	REPORT Requestid: e875485b-cdb2-4e1e-b4e8-8054e7b7f916 Duration: 977.80 ms Billed Duration: 1000 ms Memory Size: 128 MB Max

ステップ 7: クリーンアップする

このチュートリアルが終了したら、使用していないリソースに対する料金が発生しないように、 チュートリアルに関連付けられたリソースをクリーンアップします。このステップのリソース名は、 このチュートリアルで提案されている名前です (例えば、CodeDeploy アプリケーションの名前のた めの ecs-demo-codedeploy-app など)。別の名前を使用した場合は、クリーンアップでそれらを 使用してください。

チュートリアルリソースをクリーンアップするには

1. [デプロイグループの削除]コマンドを使用して、CodeDeploy デプロイグループを削除します。

aws deploy delete-deployment-group --application-name *ecs-demo-deployment-group* -deployment-group-name *ecs-demo-dg* --region *aws-region-id*

2. [アプリケーションの削除]コマンドを使用して、 CodeDeploy アプリケーションを削除します。

aws deploy delete-application --application-name ecs-demo-deployment-group -region aws-region-id 3. delete-function コマンドを使用して、Lambda フック関数を削除します。

aws lambda delete-function --function-name AfterAllowTestTraffic

4. delete-log-group コマンドを使用して、CloudWatch log グループを削除します。

aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic

チュートリアル: CodeDeploy と AWS サーバーレスアプリケー ションモデルを使用して、更新された Lambda 関数をデプロイす る

AWS SAM は、サーバーレスアプリケーションを構築するためのオープンソースフレームワー クです。 AWS SAM テンプレート内の YAML 構文を AWS CloudFormation 構文に変換して拡張 し、Lambda 関数などのサーバーレスアプリケーションを構築します。詳細については、「AWS サーバーレスアプリケーションモデルとは何ですか。」を参照してください。

このチュートリアルでは、SAM AWS を使用して、以下を実行するソリューションを作成します。

- Lambda 関数を作成します。
- CodeDeploy アプリケーションとデプロイグループを作成します。
- CodeDeploy ライフサイクルフック中にデプロイの検証テストを実行する 2 つの Lambda 関数を 作成します。
- Lambda 関数がいつ更新されたかを検出します。Lambda 関数の更新により、CodeDeploy による デプロイがトリガーされます。これにより、本稼働トラフィックが Lambda 関数の元のバージョ ンから更新されたバージョンに段階的に移行されます。

Note

このチュートリアルでは、AWS アカウントに課金される可能性のあるリソースを作成する 必要があります。これには、CodeDeploy、Amazon CloudWatch、および の料金が含まれま す AWS Lambda。詳細については、<u>CodeDeploy pricing</u>、<u>Amazon CloudWatch pricing</u> およ び <u>AWS Lambda pricing</u> を参照してください。 トピック

• 前提条件

- ステップ 1: インフラストラクチャをセットアップする
- ステップ 2: Lambda 関数を更新します
- ステップ 3: 更新された Lambda 関数をデプロイします。
- ステップ 4: デプロイの結果を表示する
- ステップ 5: クリーンアップ

前提条件

このチュートリアルを完了するには、まず以下を行う必要があります。

- 「CodeDeployの開始方法」のステップを完了します。
- ・ CLI AWS Serverless Application Model をインストールします。詳細については、<u>「SAM AWS</u> CLI のインストール」を参照してください。
- S3 バケットを作成します。 AWS SAM は、AWS SAM テンプレートで参照されているアーティ ファクトをこのバケットにアップロードします。

ステップ 1: インフラストラクチャをセットアップする

このトピックでは、AWS SAM を使用して AWS SAM テンプレートと Lambda 関数のファイルを作 成する方法について説明します。次に、 コマンド AWS SAM packageと deploy コマンドを使用 して、インフラストラクチャ内のコンポーネントを生成します。インフラストラクチャの準備がで きたら、CodeDeploy アプリケーションとデプロイグループ、更新およびデプロイする Lambda 関 数、Lambda 関数をデプロイするときに実行される検証テストを含む 2 つの Lambda 関数がありま す。完了したら、 AWS CloudFormation を使用して Lambda コンソールまたは でコンポーネントを 表示 AWS CLI し、Lambda 関数をテストできます。

トピック

- ファイルを作成する
- SAM AWS アプリケーションをパッケージ化する
- SAM AWS アプリケーションをデプロイする
- (オプション) インフラストラクチャの検査とテスト

ファイルを作成する

インフラストラクチャを作成するには、次のファイルを作成する必要があります。

- template.yml
- myDateTimeFunction.js
- beforeAllowTraffic.js
- afterAllowTraffic.js

トピック

- AWS SAM テンプレートを作成する
- Lambda 関数のファイルを作成します
- [BeforeAllowTraffic] Lambda 関数のファイルを作成する
- [AfterAllowTraffic] Lambda 関数のファイルを作成する

AWS SAM テンプレートを作成する

インフラストラクチャ内のコンポーネントを指定する AWS SAM テンプレートファイルを作成しま す。

AWS SAM テンプレートを作成するには

- 1. SAM-Tutorial という名前のディレクトリを作成します。
- 2. SAM-Tutorial ディレクトリに template.yml という名前のファイルを作成します。
- 次の YAML コードを template.yml にコピーします。これが使用する AWS SAM テンプレートです。

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: A sample SAM template for deploying Lambda functions.
Resources:
# Details about the myDateTimeFunction Lambda function
myDateTimeFunction:
   Type: AWS::Serverless::Function
   Properties:
      Handler: myDateTimeFunction.handler
      Runtime: nodejs18.x
```
```
# Instructs your myDateTimeFunction is published to an alias named "live".
      AutoPublishAlias: live
# Grants this function permission to call lambda: InvokeFunction
      Policies:
        - Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: '*'
      DeploymentPreference:
# Specifies the deployment configuration
          Type: Linear10PercentEvery1Minute
# Specifies Lambda functions for deployment lifecycle hooks
          Hooks:
            PreTraffic: !Ref beforeAllowTraffic
            PostTraffic: !Ref afterAllowTraffic
# Specifies the BeforeAllowTraffic lifecycle hook Lambda function
  beforeAllowTraffic:
   Type: AWS::Serverless::Function
   Properties:
      Handler: beforeAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
# Grants this function permission to call
codedeploy:PutLifecycleEventHookExecutionStatus
          Statement:
          - Effect: "Allow"
            Action:

    "codedeploy:PutLifecycleEventHookExecutionStatus"

            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
# Grants this function permission to call lambda: InvokeFunction
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_beforeAllowTraffic'
```

```
DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
# Specifies the AfterAllowTraffic lifecycle hook Lambda function
  afterAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: afterAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call
codedeploy:PutLifecycleEventHookExecutionStatus
          - Effect: "Allow"
            Action:

    "codedeploy:PutLifecycleEventHookExecutionStatus"

            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call lambda: InvokeFunction
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_afterAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

このテンプレートは以下を指定します。詳細については、<u>AWS SAM template concepts (テンプレー</u> <u>トの概念)</u> を参照してください。

myDateTimeFunction と呼ばれるLambda 関数

この Lambda 関数が発行されると、テンプレート中の AutoPublishAlias の行は、それを live という名前のエイリアスにリンクします。このチュートリアルの後半で、この関数の更新 により、 によるデプロイ AWS CodeDeploy がトリガーされ、本番トラフィックが元のバージョ ンから更新されたバージョンに段階的に移行されます。

2つの Lambda デプロイ検証関数

次の Lambda 関数は、CodeDeploy ライフサイクルフック中に実行されます。関数には、更新さ れた myDateTimeFunction のデプロイを検証するコードが含まれています。検証テストの結果 は、PutLifecycleEventHookExecutionStatus のAPI メソッドを使用して CodeDeploy に 渡されます。検証テストが失敗すると、デプロイは失敗し、ロールバックされます。

- CodeDeployHook_beforeAllowTraffic は、BeforeAllowTraffic フック中に実行します。
- CodeDeployHook_afterAllowTraffic は、AfterAllowTraffic フック中に実行します。

両方の関数の名前は CodeDeployHook_ で始まります。CodeDeployRoleForLambda のロールは、このプレフィックスで始まる名前の Lambda 関数でのみ、Lambda の invoke のメソッドへの呼び出しを許可します。詳細については、<u>AWS の Lambda デ</u> <u>プロイ向けの AppSpec の「hooks」セクション</u> および CodeDeploy API Reference 中の <u>PutLifecycleEventHookExecutionStatus</u> を参照してください。

更新された Lambda 関数の自動検出

AutoPublishAlias 条件は、myDateTimeFunction 関数が変更されたときに検出し、live エイリアスを使用してデプロイするようにフレームワークに指示します。

デプロイ設定

デプロイ設定により、CodeDeploy アプリケーションが Lambda 関数の元のバージョンから新し いバージョンにトラフィックを移行するレートが決まります。このテンプレートは、事前定義さ れたデプロイ設定 Linear10PercentEvery1Minute を指定します。

Note

AWS SAM テンプレートでカスタムデプロイ設定を指定することはできません。詳細については、「<u>Create a Deployment Configuration</u>」を参照してください。

デプロイライフサイクルフック関数

Hooks セクションでは、ライフサイクルイベントフック中に実行する関数を指 定します。PreTraffic は、PostTraffic フック中に実行する関数を指定しま す。BeforeAllowTraffic は、フック中に AfterAllowTraffic が実行する関数を指定しま す。

Lambda が別の Lambda 関数を呼び出すための許可

指定されたlambda:InvokeFunctionアクセス許可は、SAM AWS アプリ ケーションで使用されるロールに Lambda 関数を呼び出すアクセス許可を 付与します。これは、CodeDeployHook_beforeAllowTraffic および CodeDeployHook_afterAllowTraffic の関数が、検証テスト中に、デプロイされた Lambda 関数を呼び出す場合に必要です。

Lambda 関数のファイルを作成します

このチュートリアルの後半で更新してデプロイする関数のファイルを作成します。

Note

Lambda 関数は、 AWS Lambdaでサポートされている任意のランタイムを使用できます。詳 細については、「AWS Lambda ランタイム」を参照してください。

Lambda 関数を作成するには

- テキストファイルを作成し、myDateTimeFunction.js という名前で SAM-Tutorial ディレ クトリに保存します。
- 2. 次の Node.js コードを myDateTimeFunction.js にコピーします。

```
'use strict';
exports.handler = function(event, context, callback) {
    if (event.body) {
        event = JSON.parse(event.body);
    }
    var sc; // Status code
```

```
var result = ""; // Response payload
  switch(event.option) {
    case "date":
      switch(event.period) {
        case "yesterday":
          result = setDateResult("yesterday");
          sc = 200;
          break;
        case "today":
          result = setDateResult();
          sc = 200;
          break;
        case "tomorrow":
          result = setDateResult("tomorrow");
          sc = 200;
          break;
        default:
          result = {
            "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
          };
          sc = 400;
          break;
      }
      break;
/*
        Later in this tutorial, you update this function by uncommenting
        this section. The framework created by AWS SAM detects the update
        and triggers a deployment by CodeDeploy. The deployment shifts
        production traffic to the updated version of this function.
        case "time":
        var d = new Date();
        var h = d.getHours();
        var mi = d.getMinutes();
        var s = d.getSeconds();
        result = {
          "hour": h,
          "minute": mi,
          "second": s
        };
        sc = 200;
        break;
```

```
*/
     default:
        result = {
          "error": "Must specify 'date' or 'time'."
        };
        sc = 400;
     break;
 }
 const response = {
    statusCode: sc,
   headers: { "Content-type": "application/json" },
   body: JSON.stringify( result )
 };
 callback(null, response);
 function setDateResult(option) {
   var d = new Date(); // Today
   var mo; // Month
   var da; // Day
   var y; // Year
    switch(option) {
     case "yesterday":
        d.setDate(d.getDate() - 1);
        break;
     case "tomorrow":
        d.setDate(d.getDate() + 1);
     default:
       break;
   }
   mo = d.getMonth() + 1; // Months are zero offset (0-11)
   da = d.getDate();
   y = d.getFullYear();
   result = {
      "month": mo,
     "day": da,
     "year": y
   };
```

```
return result;
}
};
```

Lambda 関数は、昨日、今日、または明日に対して年月日を返します。このチュートリアルの後半で は、指定した日または時間に関する情報(年月日または現在の時、分、秒など)を返す関数を更新する コードをコメント解除します。によって作成されたフレームワークは、 関数の更新バージョン AWS SAM を検出してデプロイします。

1 Note

この Lambda 関数は、 AWS Cloud9 チュートリアルでも使用されます。 はクラウドベース の統合開発環境 AWS Cloud9 です。でこの関数を作成、実行、更新、デバッグする方法につ いては AWS Cloud9、<u>AWS Lambda 「」のチュートリアル AWS Cloud9</u>を参照してくださ い。

[BeforeAllowTraffic] Lambda 関数のファイルを作成する

beforeAllowTraffic のフック Lambda 関数のファイルを作成します。

- テキストファイルを作成し、beforeAllowTraffic.js という名前で SAM-Tutorial ディレ クトリに保存します。
- 次の Node.js コードを beforeAllowTraffic.js にコピーします。この関数は、デプロイの BeforeAllowTraffic フック中に実行されます。

```
'use strict';
    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();
    exports.handler = (event, context, callback) => {
        console.log("Entering PreTraffic Hook!");
        // Read the DeploymentId and LifecycleEventHookExecutionId from the event
        payload
        var deploymentId = event.DeploymentId;
        var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
```

```
var functionToTest = process.env.NewVersion;
console.log("BeforeAllowTraffic hook tests started");
console.log("Testing new function version: " + functionToTest);
// Create parameters to pass to the updated Lambda function that
// include the newly added "time" option. If the function did not
// update, then the "time" option is invalid and function returns
// a statusCode of 400 indicating it failed.
var lambdaParams = {
 FunctionName: functionToTest,
 Payload: "{\"option\": \"time\"}",
 InvocationType: "RequestResponse"
};
var lambdaResult = "Failed";
// Invoke the updated Lambda function.
lambda.invoke(lambdaParams, function(err, data) {
 if (err){ // an error occurred
  console.log(err, err.stack);
  lambdaResult = "Failed";
 }
 else{ // successful response
  var result = JSON.parse(data.Payload);
  console.log("Result: " + JSON.stringify(result));
     console.log("statusCode: " + result.statusCode);
     // Check if the status code returned by the updated
     // function is 400. If it is, then it failed. If
     // is not, then it succeeded.
  if (result.statusCode != "400"){
       console.log("Validation succeeded");
   lambdaResult = "Succeeded";
     }
     else {
       console.log("Validation failed");
     }
  // Complete the PreTraffic Hook by sending CodeDeploy the validation status
  var params = {
   deploymentId: deploymentId,
   lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: lambdaResult // status can be 'Succeeded' or 'Failed'
  };
```

```
// Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
      if (err) {
       // Validation failed.
        console.log("CodeDeploy Status update failed");
       console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
       // Validation succeeded.
       console.log("CodeDeploy status updated successfully");
       callback(null, "CodeDeploy status updated successfully");
       }
     });
    }
   });
   }
```

[AfterAllowTraffic] Lambda 関数のファイルを作成する

afterAllowTraffic のフック Lambda 関数のファイルを作成します。

- 1. テキストファイルを作成し、afterAllowTraffic.js という名前で SAM-Tutorial ディレ クトリに保存します。
- 次の Node.js コードを afterAllowTraffic.js にコピーします。この関数は、デプロイの AfterAllowTraffic フック中に実行されます。

```
'use strict';
const AWS = require('aws-sdk');
const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
var lambda = new AWS.Lambda();
exports.handler = (event, context, callback) => {
console.log("Entering PostTraffic Hook!");
// Read the DeploymentId and LifecycleEventHookExecutionId from the event
payload
var deploymentId = event.DeploymentId;
var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
```

```
var functionToTest = process.env.NewVersion;
console.log("AfterAllowTraffic hook tests started");
console.log("Testing new function version: " + functionToTest);
// Create parameters to pass to the updated Lambda function that
// include the original "date" parameter. If the function did not
// update as expected, then the "date" option might be invalid. If
// the parameter is invalid, the function returns
// a statusCode of 400 indicating it failed.
var lambdaParams = {
 FunctionName: functionToTest,
 Payload: "{\"option\": \"date\", \"period\": \"today\"}",
 InvocationType: "RequestResponse"
};
var lambdaResult = "Failed";
// Invoke the updated Lambda function.
lambda.invoke(lambdaParams, function(err, data) {
 if (err){ // an error occurred
  console.log(err, err.stack);
  lambdaResult = "Failed";
 }
 else{ // successful response
  var result = JSON.parse(data.Payload);
  console.log("Result: " + JSON.stringify(result));
     console.log("statusCode: " + result.statusCode);
     // Check if the status code returned by the updated
     // function is 400. If it is, then it failed. If
     // is not, then it succeeded.
  if (result.statusCode != "400"){
       console.log("Validation of time parameter succeeded");
   lambdaResult = "Succeeded";
     }
     else {
       console.log("Validation failed");
     }
  // Complete the PostTraffic Hook by sending CodeDeploy the validation status
  var params = {
   deploymentId: deploymentId,
   lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: lambdaResult // status can be 'Succeeded' or 'Failed'
```



SAM AWS アプリケーションをパッケージ化する

これで、SAM-Tutorial ディレクトリに次の4つのファイルがあるはずです。

- beforeAllowTraffic.js
- afterAllowTraffic.js
- myDateTimeFunction.js
- template.yml

これで、 AWS SAM sam package コマンドを使用して、Lambda 関数と CodeDeploy アプリケー ションのアーティファクトを作成してパッケージ化する準備が整いました。アーティファクトは S3 バケットにアップロードされます。コマンドの出力は、package.yml という新しいファイルです。 このファイルは、次のステップで AWS SAM sam deploy コマンドによって使用されます。

Note

sam package のコマンドのさらなる詳細については、AWS SAM デベロッパーガイド の <u>AWS Serverless Application Model CLI command reference</u> を参照してください。

SAM-Tutorial ディレクトリで、以下を実行します。

sam package ∖

- --template-file template.yml \
- --output-template-file package.yml \
- --s3-bucket amzn-s3-demo-bucket

s3-bucket のパラメータには、このチュートリアルの前提条件として作成した Amazon S3 バケットを指定します。は、 AWS SAM sam deploy コマンドで使用される新しいファイルの名前outputtemplate-fileを指定します。

SAM AWS アプリケーションをデプロイする

package.yml ファイルで AWS SAM sam deploy コマンドを使用して、 を使用して Lambda 関数と CodeDeploy アプリケーションおよびデプロイグループを作成します AWS CloudFormation。

Note

sam deploy のコマンドのさらなる詳細については、AWS Serverless Application Model デベロッパーガイド の AWS SAM CLI command reference を参照してください。

SAM-Tutorial ディレクトリで、次のコマンドを実行します。

```
sam deploy \
    --template-file package.yml \
    --stack-name my-date-time-app \
    --capabilities CAPABILITY_IAM
```

IAM ロールの作成を --capabilities CAPABILITY_IAM に許可するには、 AWS CloudFormation のパラメータが必要です。

(オプション) インフラストラクチャの検査とテスト

このトピックでは、インフラストラクチャコンポーネントを表示し、Lambda 関数をテストする方法 を示します。

sam deployの実行後にスタックの結果を表示するには

1. AWS CloudFormation コンソールを <u>https://console.aws.amazon.com/cloudformation</u>://https:// https://https://https://https://https

- ナビゲーションペインで、[Stacks] を選択します。my-date-time-app スタックが上部に表示 されます。
- [イベント] タブを選択して、完了したイベントを確認します。スタックの作成の進行中に、イベントを表示できます。スタックの作成が完了すると、すべてのスタック作成イベントを表示できます。
- スタックを選択した状態で、[リソース] を選択します。タイプ 列に、Lambda関数、myDateTimeFunction、CodeDeployHook_beforeAllowTraffic および CodeDeployHook_afterAllowTraffic が表示されます。Lambda 関数の Physical ID の各列 には、Lambda コンソールで関数を表示するためのリンクが含まれています。

Note

myDateTimeFunction Lambda 関数の名前には AWS CloudFormation スタックの名前 が付加され、識別子が追加されているため、 のようになりますmy-date-time-appmyDateTimeFunction-123456ABCDEF。

- 5. https://console.aws.amazon.com/codedeploy/ で、CodeDeploy コンソールを開きます。
- 6. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- によって作成された新しい CodeDeploy アプリケーションが、 で始ま る名前 AWS CloudFormation で表示されますmy-date-time-app-ServerlessDeploymentApplication。このアプリケーションを選択します。
- my-date-time-app-myDateTimeFunctionDeploymentGroup で始まる名前のデプロイグ ループが表示されます。このデプロイグループを選択します。

[デプロイ設定] に、[CodeDeployDefault.LambdaLinear10PercentEvery1Minute] が表示されます。

(オプション) 関数をテストするには (コンソール)

- 1. AWS Lambda コンソールを <u>https://console.aws.amazon.com/lambda/</u>://https//https//http
- ナビゲーションペインで、my-date-time-app-myDateTimeFunction 関数を選択します。コンソールでは、名前に識別子が含まれているため、my-date-time-app-myDateTimeFunction-123456ABCDEFのようになります。
- 3. [テスト]を選択します。
- 4. [イベント名] にテストイベントの名前を入力します。

5. テストイベントに以下を入力し、[作成]を選択します。

```
{
   "option": "date",
   "period": "today"
}
```

6. [テスト]を選択します。テストイベントのリストには、テストイベントのみが表示されます。

[実行結果] に [成功] と表示されます。

7. [実行結果]で、[詳細]を展開して結果を表示します。現在の年月日が表示されます。

(オプション) 関数をテストするには (AWS CLI)

- 1. Lambda 関数の ARN を配置します。関数を表示しているときに、Lambda コンソールの上部に 表示されます。
- 2. 以下のコマンドを実行してください。[*your-function-arn*] を関数 ARN に置き換えます。

```
aws lambda invoke \
--function your-function-arn \
--cli-binary-format raw-in-base64-out \
--payload "{\"option\": \"date\", \"period\": \"today\"}" out.txt
```

3. out.txtを開き、結果に現在の年月日が含まれていることを確認します。

ステップ 2: Lambda 関数を更新します

このトピックでは、myDateTimeFunction.js ファイルを更新します。次のステップでは、この ファイルを使用して、更新された関数をデプロイします。これは、本稼働トラフィックを Lambda 関数の現在のバージョンから更新されたバージョンに移行することによってデプロイするため に、CodeDeploy をトリガーします。

Lambda 関数を更新するには

- myDateTimeFunction.jsを開きます。
- 2 つのコメントマーカー(「/*」と「*/」)および case ブロック内の time という名前の switch の開始と終了にある説明テキストを削除します。

コメントされていないコードを使用すると、新しいパラメータ time を関数に渡すことができ ます。time を更新された関数に渡すと、現在の hour、minute、および second が返されま す。

3. myDateTimeFunction.js を保存します。次のようになります。

```
'use strict';
exports.handler = function(event, context, callback) {
 if (event.body) {
   event = JSON.parse(event.body);
 }
 var sc; // Status code
 var result = ""; // Response payload
 switch(event.option) {
    case "date":
      switch(event.period) {
        case "yesterday":
          result = setDateResult("yesterday");
          sc = 200;
          break;
        case "today":
          result = setDateResult();
          sc = 200;
          break;
        case "tomorrow":
          result = setDateResult("tomorrow");
          sc = 200;
          break;
        default:
          result = {
            "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
          };
          sc = 400;
          break;
      }
      break;
      case "time":
        var d = new Date();
        var h = d.getHours();
```

```
var mi = d.getMinutes();
      var s = d.getSeconds();
      result = {
        "hour": h,
        "minute": mi,
        "second": s
      };
      sc = 200;
      break;
    default:
      result = {
        "error": "Must specify 'date' or 'time'."
     };
      sc = 400;
    break;
}
const response = {
  statusCode: sc,
 headers: { "Content-type": "application/json" },
 body: JSON.stringify( result )
};
callback(null, response);
function setDateResult(option) {
 var d = new Date(); // Today
 var mo; // Month
 var da; // Day
 var y; // Year
  switch(option) {
    case "yesterday":
      d.setDate(d.getDate() - 1);
      break;
    case "tomorrow":
      d.setDate(d.getDate() + 1);
    default:
     break;
  }
```

```
mo = d.getMonth() + 1; // Months are zero offset (0-11)
da = d.getDate();
y = d.getFullYear();
result = {
    "month": mo,
    "day": da,
    "year": y
};
return result;
}
```

ステップ 3: 更新された Lambda 関数をデプロイします。

このステップでは、更新された myDateTimeFunction.js を使用して、Lambda 関数のデプロイ を更新して開始します。CodeDeploy または AWS Lambda コンソールでデプロイの進行状況をモニ タリングできます。

AWS SAM テンプレートの AutoPublishAlias: live行により、インフラストラクチャはエ イliveリアスを使用する関数の更新を検出します。関数を更新すると、CodeDeploy によるデプロ イがトリガーされます。これにより、本稼働トラフィックが関数の元のバージョンから更新された バージョンに移行されます。

sam package と sam deploy のコマンドは、Lambda 関数のデプロイを更新およびトリガーするため に使用されます。これらのコマンドは、<u>SAM AWS アプリケーションをパッケージ化する</u>および SAM AWS アプリケーションをデプロイする で実行しました。

更新された Lambda 関数をデプロイするには

1. SAM-Tutorial ディレクトリで、次のコマンドを実行します。

```
sam package \
    --template-file template.yml \
    --output-template-file package.yml \
    --s3-bucket amzn-s3-demo-bucket
```

これにより、S3 バケットで更新された Lambda 関数を参照する新しいアーティファクトのセットが作成されます。

2. SAM-Tutorial ディレクトリで、次のコマンドを実行します。

```
sam deploy \
    --template-file package.yml \
    --stack-name my-date-time-app \
    --capabilities CAPABILITY_IAM
```

スタック名はまだ であるためmy-date-time-app、 はこれがスタックの更新であること AWS CloudFormation を認識します。更新されたスタックを表示するには、 AWS CloudFormation コ ンソールを返し、ナビゲーションペインから スタックを選択します。

(オプション)デプロイ中にトラフィックを表示するには(CodeDeploy コンソール)

- 1. https://console.aws.amazon.com/codedeploy/ で、CodeDeploy コンソールを開きます。
- 2. ナビゲーションペインで、[アプリケーション] を展開し、[my-date-time-app-ServerlessDeploymentApplication] アプリケーションを選択します。
- [デプロイグループ] で、アプリケーションのデプロイグループを選択します。ステータスは [進行中]になります。
- 4. [Deployment group history (デプロイグループ履歴)] で、進行中のデプロイを選択します。

[トラフィックの移行] の進行状況バーと、このページの [Original] ボックスと 置換] ボックスの 割合に、進行状況が表示されます。

Deployment status		Traffic shifting progress	
Step 1 Pre-deployment validation	Completed	The deployment will shift 10% of tra minute(s) until all of the traffic is rou	ffic from the current version to the replacement version every 1 ted to the new version.
Step 2 Traffic shifting	20% complete fin progress	80%	20%
Step 3 Post-deployment validation	Not started	Deployment results Info	

(オプション) デプロイ中にトラフィックを表示するには(Lambda コンソール)

1. AWS Lambda コンソールを <u>https://console.aws.amazon.com/lambda/</u>://https//https//http

- ナビゲーションペインで、my-date-time-app-myDateTimeFunction 関数を選択 します。コンソールでは、名前に識別子が含まれているため、my-date-time-appmyDateTimeFunction-123456ABCDEFのようになります。
- 3. Aliases、live の順に選択します。

元の関数バージョン(バージョン 1)と更新された関数バージョン(バージョン 2)の横の重みは、この AWS Lambda コンソールページがロードされた時点で各バージョンに提供されているトラフィック 量を示します。ページでは、時間が経過しても重みは更新されません。ページを1分に1回更新す ると、バージョン1の重みは 10% 減り、バージョン2の重みは 10% 増加します。

Aliases	
You are viewing the configuration for alias live . <u>Manage the configuration</u> for the underlying version 1 . Manage the configuration for the underlying version 2 .	
Name	
live	
Description	
Version*	Weight: 80%
You can shift traffic between two versions, based on weights (%) that you assign. Click here to learn more.	
Additional version	Weight 20 %

ステップ 4: デプロイの結果を表示する

このステップでは、デプロイの結果を表示します。デプロイが成功すると、更新された Lambda 関数が本稼働トラフィックを受信することを確認できます。デプロイが失敗した場合、CloudWatch Logs を使用して、デプロイのライフサイクルフックの間に実行される Lambda 関数の検証テストの 出力を表示できます。

トピック

- デプロイされた関数をテストする
- CloudWatch Logs でのフックイベントを表示します。

デプロイされた関数をテストする

sam deploy のコマンドは、my-date-time-app-myDateTimeFunction のLambda 関数を更新し ます。関数のバージョンが 2 に更新され、live エイリアスに追加されます。

Lambda コンソール中の更新を見るには

- 1. AWS Lambda コンソールを <u>https://console.aws.amazon.com/lambda/</u>://https://https://https://https://https://https://https://https://https://https://https://https://https//ht
- ナビゲーションペインで、my-date-time-app-myDateTimeFunction 関数を選択 します。コンソールでは、名前に識別子が含まれているため、my-date-time-appmyDateTimeFunction-123456ABCDEFのようになります。
- [Qualifiers (修飾子)]、[エイリアス] の順に選択します。デプロイが完了した後(約 10 分)、live エイリアスに [バージョン: 2] と表示されます。

Switch versions/aliases		
Filter versions/aliases		
Versions	Aliases	_
Unqualified ? Version: \$LATEST		
live Version: 2		

- 4. [関数コード]で、関数のソースコードを表示します。変更が表示されます。
- 5. (オプション)<u>ステップ 2: Lambda 関数を更新します</u>のテスト手順を使用して、更新された関数 をテストできます。次のペイロードを使用して新しいテストイベントを作成し、結果に現在の 時、分、秒が含まれていることを確認します。

{ "option": "time" }

を使用して更新された関数を AWS CLI テストするには、次のコマンドを実行し、 out.txtを 開いて、結果に現在の時間、分、秒が含まれていることを確認します。 aws lambda invoke --function your-function-arn --payload "{\"option\": \"time\"}"
out.txt

Note

デプロイが完了する前に を使用して関数を AWS CLI テストすると、予期しない結果が 表示されることがあります。これは、CodeDeploy がトラフィックの 10% を毎分更新 バージョンに段階的に移行するためです。デプロイ中、一部のトラフィックは引き続き 元のバージョンを指すため、aws lambda invoke は元のバージョンを使用する場合が あります。10 分後には、デプロイが完了し、すべてのトラフィックが関数の新しいバー ジョンを指し示します。

CloudWatch Logs でのフックイベントを表示します。

BeforeAllowTraffic のフック中に、CodeDeploy は CodeDeployHook_beforeAllowTraffic の Lambda 関数を実行します。AfterAllowTraffic のフック中に、CodeDeploy は CodeDeployHook_afterAllowTraffic の Lambda 関数を実行します。各関数は、新しい time パラメータを使用して関数の更新バージョンを呼び出す検証テストを実行します。Lambda 関数の更 新が成功した場合、time のオプションによるエラーは発生せず、検証は成功します。関数が更新さ れなかった場合、認識されないパラメータによってエラーが発生し、検証が失敗します。これらの検 証テストはデモンストレーションのみを目的としています。デプロイを検証するには独自のテストを 記述します。CloudWatch Logs コンソールを使用して、検証テストを表示できます。

CodeDeploy フックイベントを表示するには

- 1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
- 2. ナビゲーションペインで、[Logs (ログ)] を選択します。
- 3. ロググループのリストから、[/aws/lambda/CodeDeployHook_beforeAllowTraffic] または [/aws/ lambda/CodeDeployHook_afterAllowTraffic] を選択します。
- 4. ログストリームを選択します。1 つのみ表示されます。
- 5. イベントを展開して詳細を表示します。

		4
	Time (UTC +00:00)	Message
	2019-07-12	
		No older events found at the moment. Re
•	22:08:56	START RequestId: 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Version: \$LATEST
•	22:08:56	2019-07-12T22:08:56.834Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Entering PreTraffic Hook!
•	22:08:56	2019-07-12T22:08:56.834Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Testing new function version: arr:aws:lambda:ca-d
-	22:08:58	2019-07-12T22:08:58.084Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Result: {"statusCode":200, "headers": {"Content-ty
201' { }	9-07-12T22:08:58.084Z 9f1d8 "statusCode": 200, "headers": { "Content-type": "appl: }, "body": "{\"hour\":22,\"mi	8158-5acc-4618-bf5f-5c1c99d8fa49 Result: ication/json" inute\":8,\"second\":57}"
•	22:08:58	2019-07-12T22:08:58.084Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 statusCode: 200
201	9-07-12T22:08:58.084Z 9f1d8	8158-5acc-4618-bf5f-5c1c99d8fa49 statusCode: 200
-	22:08:58	2019-07-12T22:08:58.084Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Validation succeeded
201	9-07-12T22:08:58.084Z 9f1d8	3158-5acc-4618-bf5f-5clc99d8fa49 Validation succeeded
-	22:08:58	2019-07-12T22:08:58.302Z 9f1d8158-5acc-4618-bf5f-5c1c99d8fa49 Codedeploy status updated successfully
201	9-07-12T22:08:58.302Z 9f1d8	3158-5acc-4618-bf5f-5c1c99d8fa49 Codedeploy status updated successfully

ステップ 5: クリーンアップ

このチュートリアルで使用したリソースの追加料金を回避するには、 AWS SAM テンプレートに よって作成されたリソースと、Lambda 検証関数によって作成された CloudWatch ログを削除しま す。

AWS CloudFormation スタックを削除するには

- 1. にサインイン AWS Management Console し、 AWS CloudFormation コンソールを <u>https://</u> console.aws.amazon.com/cloudformation://www.com で開きます。
- 2. [スタック] 列で、my-date-time-app スタックを選択し、[削除] を選択します。
- プロンプトが表示されたら、[スタックの削除] を選択します。Lambda 関数、CodeDeploy アプ リケーションとデプロイグループ、および によって作成された IAM ロール AWS SAM は削除さ れます。

CloudWatch Logs でログを削除するには

- 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、[Logs (ログ)] を選択します。

- 3. ロググループのリストから、[/aws/lambda/CodeDeployHook_beforeAllowTraffic] の横にあるボ タンを選択します。
- 4. [アクション]から、[ロググループを削除する]を選択し、次に [はい、削除します]を選択しま す。
- 5. ロググループのリストから、[/aws/lambda/CodeDeployHook_afterAllowTraffic] の横にあるボタ ンを選択します。
- 6. [アクション]から、[ロググループを削除する]を選択し、次に [はい、削除します] を選択しま す。

CodeDeploy エージェントの使用

AWS CodeDeploy エージェントは、インスタンスにインストールして設定すると、そのインスタン スを CodeDeploy デプロイで使用できるようにするソフトウェアパッケージです。

AWS は、CodeDeploy エージェントの最新マイナーバージョンをサポートしています。現在、最新 のマイナーバージョンは 1.7.x です。

Note

CodeDeploy エージェントは、EC2/オンプレミスのコンピューティングプラットフォームに デプロイする場合にのみ必要です。Amazon ECS または AWS Lambda コンピューティング プラットフォームを使用するデプロイには、このエージェントは必要ありません。

エージェントがインストールされている場合、設定ファイルはインスタンスに配置されます。この ファイルは、エージェントの動作を指定するために使用されます。この設定ファイルでは、 がイン スタンスとやり取り AWS CodeDeploy するときに使用するディレクトリパスやその他の設定を指定 します。ファイルの一部の設定オプションは変更できます。CodeDeploy エージェント設定ファイル の使用の詳細については、「CodeDeploy エージェント設定リファレンス」を参照してください。

インストール、更新、バージョンの確認のステップなど、CodeDeploy エージェントの使用に関する 詳細については、「CodeDeploy エージェントのオペレーションの管理」を参照してください。

トピック

- CodeDeploy エージェントで対応するオペレーティングシステム
- CodeDeploy エージェントの通信プロトコルとポート
- CodeDeploy エージェントのバージョン履歴
- <u>CodeDeploy プロセスの管理</u>
- <u>アプリケーションリビジョンとログファイルのクリーンアップ</u>
- CodeDeploy エージェントでインストールされるファイル
- CodeDeploy エージェントのオペレーションの管理

CodeDeploy エージェントで対応するオペレーティングシステム

対応する Amazon EC2 AMI オペレーティングシステム

CodeDeploy エージェントは、以下の Amazon EC2 AMI オペレーティングシステムでテスト済みで す。

- Amazon Linux 2023 (ARM、x86)
- Amazon Linux 2 (ARM、x86)
- Microsoft Windows Server 2022、2019
- Red Hat Enterprise Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu Server 22.04 LTS、20.04 LTS、18.04 LTS、16.04 LTS

CodeDeploy エージェントは、ニーズに適応できるようオープンソースとして利用できます。他 の Amazon EC2 AMI オペレーションシステムでも使用できます。詳細については、GitHub の CodeDeploy エージェント リポジトリを参照してください。

サポートされているオンプレミスオペレーションシステム

CodeDeploy エージェントは次のオンプレミスオペレーティングシステムでテスト済みです。

- Microsoft Windows Server 2022、2019
- Red Hat Enterprise Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu Server 22.04 LTS、20.04 LTS

CodeDeploy エージェントは、ニーズに適応できるようオープンソースとして利用できます。他の オンプレミスインスタンスオペレーティングシステムで使用できます。詳細については、GitHub の CodeDeploy エージェント リポジトリを参照してください。

CodeDeploy エージェントの通信プロトコルとポート

CodeDeploy エージェントは、ポート 443 上の HTTPS を使用してアウトバウンド通信を行います。

CodeDeploy エージェントが EC2 インスタンスで実行されると、<u>EC2 メタデータ</u> エンドポイントを 使用して、インスタンス関連の情報を取得します。詳細については、「<u>インスタンスメタデータサー</u> ビスの制限」を参照してください。

CodeDeploy エージェントのバージョン履歴

インスタンスでは、 CodeDeploy エージェントのサポートされているバージョンが実行されている 必要があります。現在サポートされている最小バージョンは 1.7.x です。

Note

最新バージョンの CodeDeploy エージェントを使用することをお勧めします。問題が発生 した場合は、 AWS サポートに連絡する前に最新バージョンに更新してください。アップグ レード情報については、「<u>CodeDeploy エージェントを更新する</u>」を参照してください。

次の表は、CodeDeploy エージェントのすべてのリリースと、各バージョンに含まれる機能や強化された機能の一覧です。

バージョ ン	リリース日	詳細
1.7.1	2024 年 11 月 14 日	変更: セキュリティパッチの依存関係を更新しました。
1.7.0	2024 年 3 月 6 日	追加: CodeDeploy エージェント:disable_imds_v1: 設定 ファイルの設定。この設定を使用して、IMDSv2 エラーが発 生したときに IMDSv1 IMDSv2へのフォールバックを無効に します。デフォルトは です false (フォールバックを有効に します)。詳細については、 <u>CodeDeploy エージェント設定</u> <u>リファレンス</u> 」を参照してください。
		追加: Red Hat Enterprise Linux 9 (RHEL 9) オペレーティング システムのサポート。 追加: Ubuntu Server での Ruby バージョン 31 および 32 の
		サポート。
		修正済み: CodeDeploy エージェント設定ファイルがロードに 失敗すると、CodeDeploy エージェントはユーザーフレンド リーなエラーを生成するようになりました。

バージョ ン	リリース日	詳細
		変更: Windows 用 CodeDeploy エージェントで Ruby を 2.7.8-1 にアップグレードしました。
1.6.0	2023 年 3 月 30 日	追加: Ruby 3.1、3.2 に対応しました。
		追加: Amazon Linux 2023 に対応しました。
		追加: Windows Server 2022 に対応しました。
		変更: Windows Server インスタンスでは、デフォルトの verbose の設定は false になりました。Windows で引 き続きデバッグメッセージをログファイルに出力するに は、verbose を true に設定する必要があります。
		削除: Windows Server 2016 および Windows Server 2012 R2 のサポートが削除されました。
		削除: Amazon Linux 2018.03.x が削除されました。

バージョ ン	リリース日	詳細
1.5.0	2023年3月3日	追加: Ruby 3 に対応しました。
		追加: Ubuntu 22.04 に対応しました。
		修正: 起動後すぐに CodeDeploy エージェントを再起動する と、エージェントがハングアップする問題を修正しました。
		変更: CodeDeploy エージェントは、フックスクリプトの実行 中にエージェントサービスが予期せず再起動した場合、エー ジェント起動時にホストデプロイに失敗するようになりまし た。この修正により、70 分のタイムアウト時間を待ってから デプロイを再試行する必要がなくなりました。
		廃止通知: CodeDeploy エージェント 1.5.0 は Windows Server 2016 と Windows Server 2012 R2 をサポートする最後 のリリースです。
		削除: Ubuntu 14.04 LTS、Windows Server 2008 R2、および Windows Server 2008 R2 32 ビットでの CodeDeploy エー ジェントのサポートが削除されました。
1.4.1	2022 年 12 月 6 日	修正: ログ記録に関連するセキュリティの脆弱性を修正しまし た。
		強化: ホストコマンドのポーリング時のログ記録を改善しまし た。

A1A/C	CodeDoplay
AVVS	CodeDeploy

バージョ ン	リリース日	詳細
1.4.0	2022 年 8 月 31 日	追加: Red Hat Enterprise Linux 8 に対応しました。
		追加: Windows 用の CodeDeploy エージェントでの長いファ イルパスに対応しました。長いファイルパスを有効にするに は、適切な Windows レジストリキーを設定し、エージェント を再起動する必要があります。詳細については、「 <u>ファイル</u> パスが長いと、「そのようなファイルまたはディレクトリは ありません」というエラーが発生します」を参照してくださ い。
		修正: ディスクがいっぱいになったときの解凍オペレーショ ンに関する問題を修正しました。CodeDeploy エージェント は、ディスクがいっぱいであることを示す解凍の <u>終了コード</u> 50 を検出し、部分的に抽出されたファイルを削除して、例外 を発生させて失敗を CodeDeploy サーバーに送信するように なりました。このエラーメッセージはライフサイクルイベン トのエラーメッセージとして表示され、ホストレベルのデプ ロイはスタックしたりタイムアウトしたりすることなく停止 します。
		修正: エージェントが失敗する原因となる問題を修正しまし た。
		修正: エッジケースの競合状態時にフックがタイムアウトする 問題を修正しました。スクリプトのないフックは引き続き動 作するようになり、障害やタイムアウトが発生しなくなりま した。
		変更: CodeDeploy エージェントの bin ディレクトリにある update スクリプトは、使用されなくなったため削除されま した。
		変更: Windows Server 用の CodeDeploy エージェントは Ruby 2.7 がバンドルされました。

バージョ ン	リリース日	詳細
		変更: デプロイバンドルのソース (Amazon S3 または GitHub) に応じてフックスクリプトが使用する新しい環境変数が追加 されました。 詳細については、「 <u>フックの環境変数の可用性</u> 」を参照して
		▲ Important 廃止通知: CodeDeploy エージェント 1.4.0 は、32 ビット Windows Server 用のインストーラーが含まれ る最後のリリースです。 廃止通知: CodeDeploy エージェント 1.4.0 は Windows Server 2008 R2 をサポートする最後のリ リースです。 削除: Amazon Linux 2014.09、2016.03、20 16.09、2017.03 の Amazon EC2 AMI での CodeDeploy エージェントに対応しました。

バージョ ン	リリース日	詳細
1.3.2	2021年5月6日	▲ Important CodeDeploy エージェント 1.3.2 は、エージェン トを実行している Windows ホストに影響を与え る <u>CVE-2018-1000201</u> に対応しています。CVE は、CodeDeploy エージェントの依存関係である ruby-ffi を挙げています。エージェントが Amazon EC2 Systems Manager (SSM) とともにインストール され、自動的に更新するように設定されている場合、 アクションは必要ありません。それ以外の場合は、 エージェントを手動で更新するためのアクションが必 要になります。エージェントをアップグレードするに は、「Windows サーバーで CodeDeploy エージェン トを更新する」の手順に従います。
		 修正: CodeDeploy エージェントを Ubuntu 20.04 以降にインストールする際の問題。 修正: 圧縮ファイルを抽出する際に、相対パスが正しく処理されていないために発生する断続的な問題。 追加: Windows インスタンスの AWS PrivateLink および VPC エンドポイント の対応。 追加: AppSpec ファイルの改善について、以下に説明します。 ・ ローカルデプロイを作成する時に、AppSpec ファイルにカスタムファイル名を指定できるようになりました。詳細については、「ローカルのデプロイを作成する。」を参照してください。 ・ AppSpec ファイルに、yaml 拡張子をつけることができる

バージョ ン	リリース日	詳細
		 AppSpec ファイルの新しいオプションの file_exis ts_behavior 設定を使用して、デプロイされたファ イルを上書きできるようになりました。詳細については、 「<u>AppSpec の「ファイル」セクション (EC2/オンプレミス</u> デプロイのみ)」を参照してください。 アップグレード: CodeDeploy が、Ruby 3.0 用の AWS SDK を使用するようになりました。
1.3.1	2020 年 12 月 22 日	修正: オンプレミスのインスタンスが起動しない 1.3.0 の問 題。
1.3.0	2020 年 11 月 10 日	▲ Important このバージョンは非推奨です。
		修正: 使用されなくなった期限切れの証明書を削除しました。 修正: が使用するエージェントアンインストールスクリプトか らプロンプトメッセージを削除し AWS Systems Manager、 ホストまたはフリートを以前のバージョンのエージェントに ダウングレードしやすくしました。

バージョ ン	リリース日	詳細
1.2.1	2020 年 9 月 23 日	変更: v2 から v3 へ AWS SDK for Ruby 依存関係をアップグ レードしました。
		追加: IMDSv2 に対応しました。IMDsv2 http リクエストが失 敗した場合、IMDSv1 へのサイレントフォールバックを含み ます。
		変更: セキュリティパッチ用に Rake と Rubyzip の依存関係を 更新しました。
		修正: 空の PID ファイルが、No CodeDeploy Agent Running のステータスを返すことを確認し、エージェントの 起動時に PID ファイルをクリーンアップします。

バージョ ン	リリース日	詳細
1.1.2	2020年8月4日	追加: Ubuntu Server 19.10 および 20.04 に対応しました。
		注意: バージョン 19.10 は使用期限を迎えたため、Ubuntu お よび CodeDeploy のサポートは終了しました。
		追加: Linux と Ubuntu のメモリ効率を改善し、予約メモリを よりタイムリーにリリースできるようになりました。
		追加: Windows Server の [サイレントクリーンクリーンアッ プ] との互換性より、エージェントが応答しなくなることがあ りました。
		追加: デプロイ時の失敗を避けるために、クリーンアップ中に 空でないディレクトリを無視します。
		追加: ロサンゼルス (LA) の AWS ローカルゾーンのサポー ト。
		追加: インスタンスメタデータから AZ を抽出して、 AWS ローカルゾーンの互換性を提供します。
		追加: ユーザーはサブディレクトリにアーカイブを提供できる ようになり、ルートディレクトリに保存する必要はありませ ん。
		追加: Rubyzip でメモリリークが発生する可能性のある問題を 検出しました。Rubyzip を使用する前に、まずシステムにイ ンストールされている unzip ユーティリティの使用を試みる ように、unzip コマンドを更新しました。
		追加: エージェント構成設定としての :enable_a uth_policy: 。
		変更: Unzip の警告が無視されるようになり、デプロイが継続 されるようになりました。

AWS CodeDeploy

バージョ ン	リリース日	詳細
1.1.0	2020 年 6 月 30 日	変更: CodeDeploy エージェントのバージョニングが Ruby の 標準のバージョニング規約に従うようになりました。
		追加: コマンドラインから特定のエージェントバージョンをイ ンストールできる、インストールおよび更新コマンドの新し いパラメータ。
		削除: Linux および Ubuntu 用の CodeDeploy エージェント Auto Update を削除しました。CodeDeploy エージェントの自 動更新を設定するには、「 <u>を使用して CodeDeploy エージェ</u> <u>ントをインストールする AWS Systems Manager</u> 」を参照し てください。
1.0.1.1597	2018 年 11 月 15 日	機能強化: CodeDeploy は Ubuntu 18.04 を対応しました。
		機能強化: CodeDeploy は Ruby 2.5 をサポートします。
		機能強化: CodeDeploy は FIPS エンドポイントをサポートし ます。FIPS エンドポイントの詳細については、「 <u>FIPS 140-2</u> <u>の概要</u> 」を参照してください。CodeBuild で使用可能なエン ドポイントについては、「 <u>CodeDeploy のリージョンとエン</u> <u>ドポイント</u> 」を参照してください。
1.0.1.1518	2018 年 6 月 12 日	機能強化: ポールリクエストの受け入れ中に CodeDeploy エー ジェントを閉じたときにエラーが発生する問題が修正されま した。
		機能強化: デプロイの進行中に CodeDeploy エージェントを閉 じられないようにするデプロイ追跡機能を追加しました。
		機能強化: ファイルを削除する際のパフォーマンスが改善され ました。

AWS CodeDeploy

バージョ ン	リリース日	詳細
1.0.1.1458	2018 年 3 月 6 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: より多くの信頼された機関をサポートするため、証 明書の検証を改善しました。
		機能強化: ローカル CLI が BeforeInstall ライフサイクルイベ ントを含むデプロイに失敗していた問題を修正しました。
		機能強化: CodeDeploy エージェントの更新された時に、アク ティブなデプロイが失敗する可能性がある問題を修正しまし た。
1.0.1.1352	2017 年 11 月 16 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能導入: CodeDeploy エージェントがインストールされてい るローカルマシンまたはインスタンスで、EC2/オンプレミス のデプロイをテストおよびデバッグする新機能が導入されま した。
1.0.1.1106	2017 年 5 月 16 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		特徴: 前回の成功したデプロイのアプリケーションリビジョン の一部ではない、デプロイ先のコンテンツを処理する新しい サポートを導入しました。既存のコンテンツのデプロイオプ ションとして、コンテンツの保持、コンテンツの上書き、ま たはデプロイの失敗が追加されました。
		機能強化: CodeDeploy エージェントをバージョン 2.9.2 (AWS SDK for Ruby aws-sdk-core 2.9.2) と互換性を持たせま した。
AWS CodeDeploy

バージョ ン	リリース日	詳細
1.0.1.1095	2017 年 3 月 29 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: 中国 (北京) リージョンに CodeDeploy エージェン トのサポートが導入されました。
		機能強化: ライフサイクルイベントフックから呼び出されたと きに Windows Server インスタンスで Puppet が実行されるよ うになりました。
		機能強化: untar オペレーションの処理が改善されました。
1.0.1.1067	2017 年 1 月 6 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: 多くのエラーメッセージを改訂し、デプロイの失敗 に関するより具体的な原因を含めました。
		機能強化: CodeDeploy エージェントが一部のデプロイ中にデ プロイする正しいアプリケーションリビジョンを特定できな い問題を修正しました。
		機能強化: untar オペレーションの前後の pushd と popd の 使用を元に戻しました。
1.0.1.1045	2016 年 11 月 21 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: CodeDeploy エージェントを のバージョン 2.6.11 AWS SDK for Ruby (aws-sdk-core 2.6.11) と互換性を持たせ ました。

AWS CodeDeploy

バージョ ン	リリース日	詳細
1.0.1.1037	2016 年 10 月 19 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		Amazon Linux、RHEL、および Ubuntu Server インスタンス の CodeDeploy エージェントを更新して、以下の変更を反映 しました。Windows Server インスタンスの場合、最新バー ジョンは 1.0.1.998 のままです。
		機能強化: エージェントは、インスタンスにインストールされ ている Ruby のバージョンを特定し、そのバージョンを使用 して codedeploy-agent スクリプトを呼び出すことがで きるようになりました。
1.0.1.101 1.1	2016 年 8 月 17 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: シェルのサポートの問題により、バージョン 1.0.1.1011 で導入された変更を削除しました。このバージョ ンのエージェントは、2016 年 7 月 11 日にリリースされた バージョン 1.0.1.998 と機能的に同じものです。

AWS CodeDeploy

バージョ ン	リリース日	詳細
1.0.1.1011	2016 年 8 月 15 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		Amazon Linux、RHEL、および Ubuntu Server インスタンス の CodeDeploy エージェントを更新して、以下の変更を反映 しました。Windows Server インスタンスの場合、最新バー ジョンは 1.0.1.998 のままです。
		機能導入: systemd init システムが使用されているオペレー ションシステムで、Bash シェルを使用して CodeDeploy エー ジェントを呼び出すサポートが追加されました。 機能強化: CodeDeploy エージェントおよび CodeDeploy エー ジェントアップデータで Ruby 2.x のすべてのバージョンに 対するサポートが有効になりました。更新された CodeDeplo y エージェントは、Ruby 2.0 のみに依存しなくなりました (CodeDeploy エージェントインストーラの deb および rpm バージョンでは、Ruby 2.0 が引き続き必要です)。
1.0.1.998 2	2016 年 7 月 11 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能強化: CodeDeploy エージェントを ルート 以外のユー ザープロファイルで実行する場合のサポートを修正しまし た。環境変数の競合を回避するため、USER いう名前の変数 は CODEDEPLOY_USER で置き換えられました。

AWS CodeDeploy

バージョ ン	リリース日	詳細
1.0.1.966	2016 年 6 月 16 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能導入: CodeDeploy エージェントを ルート 以外のユー ザープロファイルで実行する場合のサポートを導入しまし た。
		機能強化: デプロイグループに対して CodeDeploy エージェン トでアーカイブするアプリケーションリビジョン数を指定す るサポートを修正しました。
		機能強化: CodeDeploy エージェントをバージョン 2.3 の AWS SDK for Ruby (aws-sdk-core 2.3) と互換性を持たせまし た。
		機能強化: デプロイ中の UTF-8 エンコードに関する問題が修 正されました。
		機能強化: プロセス名を確認する際の精度が向上しました。
1.0.1.950	2016 年 3 月 24 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能: インストールプロキシのサポートを追加しました。
		機能強化: 最新のバージョンがすでにインストールされている 場合は、CodeDeploy エージェントをダウンロードしないよ うに、インストールスクリプトを更新しました。
1.0.1.934	2016 年 2 月 11 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		機能導入: デプロイグループに対して CodeDeploy エージェン トでアーカイブするアプリケーションリビジョン数を指定す るサポートが導入されました。

バージョ ン	リリース日	詳細
1.0.1.880	2016 年 1 月 11 日	注: このバージョンは現在サポートされていないため、デプロ イに失敗する可能性があります。
		機能強化: CodeDeploy エージェントを バージョン 2.2 AWS SDK for Ruby (aws-sdk-core 2.2) と互換性を持たせました。 バージョン 2.1.2 は引き続きサポートされます。
1.0.1.854 2015 年 11 日	2015 年 11 月 17 日	注: このバージョンは現在サポートされていません。このバー ジョンを使用すると、デプロイに失敗することがあります。
		特徴: SHA-256 ハッシュアルゴリズムのサポートが導入され ました。
		機能: .version ファイルでバージョンの追跡サポートが導入 されました。
		特徴: 環境変数の使用を通じて、デプロイグループ ID を利用 できるようになりました。
		機能強化: <u>Amazon CloudWatch Logs</u> を使用した CodeDeploy エージェントのログを監視するためのサポートを追加しまし た。

関連情報については、以下を参照してください。

- CodeDeploy エージェントのバージョンを特定します。
- CodeDeploy エージェントをインストールする

CodeDeploy エージェントのバージョン履歴については、「<u>GitHub のリリースリポジトリ</u>」を参照 してください。

CodeDeploy プロセスの管理

CodeDeploy エージェントのすべての Linux ディストリビューション (rpm と deb) は、デフォルトで systemd を使用してエージェントプロセスを管理します。 ただし、rpm ディストリビューションと deb ディストリビューションはどちらも、/etc/init.d/ codedeploy-agent にある起動スクリプトと共に出荷されます。使用するディストリビューショ ンによっては、sudo service codedeploy-agent restart などのコマンドを使用するとき に、systemd にプロセスの管理を許可せずに、/etc/init.d でスクリプトを実行してエージェン トプロセスを起動する場合があります。/etc/init.d でスクリプトを実行することは望ましくあり ません。

この問題を防ぐため、systemd をサポートしているシステムでは、どのエージェント操作にも service コマンドではなく systemctl ユーティリティを使用することをお勧めします。

例えば、CodeDeploy エージェントを再起動するには、service ユーティリティの同等コマンドの 代わりに sudo systemctl restart codedeploy-agentを使用します。

アプリケーションリビジョンとログファイルのクリーンアップ

CodeDeploy エージェントは、インスタンス上にリビジョンとログファイルをアーカイブしま す。CodeDeploy エージェントは、ディスクスペースを節約するために、これらのアーティファクト をクリーンアップします。

アプリケーションリビジョンのデプロイログ: エージェント設定ファイルの :max_revisions: オプショ ンを使用して、正の整数を入力することで、アーカイブするアプリケーションリビジョン数を指定 できます。CodeDeploy は、これらのリビジョンのログファイルもアーカイブします。その他すべて は、最後に成功したデプロイのログファイルを除いて削除されます。失敗したデプロイの数が、保持 されているバージョンの数を超えた場合でも、そのログファイルは常に保持されます。値を指定しな い場合、CodeDeploy は現在デプロイされたリビジョンに加えて 5 つの最新のリビジョンを保持しま す。

CodeDeploy のログ: Amazon Linux、Ubuntu Server、および RHEL インスタンスの場 合、CodeDeploy エージェントは、/var/log/aws/codedeploy-agent フォルダー下でログファ イルをローテーションします。ログファイルは、毎日 00:00:00 (インスタンス時間) にローテーショ ンされます。ログファイルは 7 日を経過した時点で削除されます。ローテーションされたログファ イルの名前付けパターンは codedeploy-agent.YYYYMMDD.log です。

CodeDeploy エージェントでインストールされるファイル

CodeDeploy エージェントはリビジョン、デプロイ履歴、デプロイスクリプトをインスタンスのルートディレクトリに保存します。このディレクトリのデフォルトの名前と場所:

Amazon Linux、Ubuntu Server、および RHEL インスタンス用の '/opt/codedeploy-agent/ deployment-root'。

Windows Server インスタンス用の 'C:\ProgramData\Amazon\CodeDeploy'。

CodeDeploy エージェントの設定ファイルにある root_dir 設定を使用して、ディレクトリの名前と場 所を設定することができます。詳細については、「<u>CodeDeploy エージェント設定リファレンス</u>」を 参照してください。

次の例は、ルートディレクトリ内のファイルとディレクトリの構造を示しています。この構造は N 件のデプロイグループがあることを前提とし、各デプロイグループには N 件のデプロイが含まれて います。

```
|--deployment-root/
|-- deployment group 1 ID
     |-- deployment 1 ID
          |-- Contents and logs of the deployment's revision
     I
I
     |-- deployment 2 ID
          |-- Contents and logs of the deployment's revision
     |-- deployment N ID
L
          |-- Contents and logs of the deployment's revision
|-- deployment group 2 ID
     |-- deployment 1 ID
          l-- bundle.tar
          |-- deployment-archive
I
               | -- contents of the deployment's revision
          |-- logs
I
               | -- scripts.log
     |-- deployment 2 ID
L
I
          |-- bundle.tar
          |-- deployment-archive
I
               | -- contents of the deployment's revision
I
          |-- logs
I
               | -- scripts.log
          L
     |-- deployment N ID
          |-- bundle.tar
I
I
          |-- deployment-archive
               | -- contents of the deployment's revision
          |-- logs
I
               | -- scripts.log
|-- deployment group N ID
     I-- deployment 1 ID
          |-- Contents and logs of the deployment's revision
```

```
|-- deployment 2 ID
I
          |-- Contents and logs of the deployment's revision
I
     |-- deployment N ID
I
          |-- Contents and logs of the deployment's revision
|-- deployment-instructions
     |-- [deployment group 1 ID]_cleanup
L
    |-- [deployment group 2 ID]_cleanup
I
    |-- [deployment group N ID]_cleanup
I
     |-- [deployment group 1 ID]_install.json
I
    |-- [deployment group 2 ID]_install.json
I
    |-- [deployment group N ID]_install.json
I
    [-- [deployment group 1 ID]_last_successful_install
[-- [deployment group 2 ID]_last_successful_install
I
    [-- [deployment group N ID]_last_successful_install
I
    [-- [deployment group 1 ID]_most_recent_install
[-- [deployment group 2 ID]_most_recent_install
     [-- [deployment group N ID]_most_recent_install
L
|-- deployment-logs
     |-- codedeploy-agent-deployments.log
L
```

Deployment Group ID フォルダは各デプロイグループを示しています。デプロイグループのディレクトリ名は、その ID です (例: acde1916-9099-7caf-fd21-012345abcdef)。各デプロイグループのディレクトリには、そのデプロイグループで試みた各デプロイのサブディレクトリ1つが含まれています。

batch-get-deployments コマンドを使用してデプロイグループ ID を検索できます。

- デプロイ ID フォルダはデプロイグループの各デプロイを示します。各デプロイディレクトリの名前はその ID です。各フォルダには以下が含まれています。
 - bundle.tar はデプロイのリビジョンのコンテンツを含む圧縮ファイルです。リビジョンを表示す る場合は、zip 圧縮解除ユーティリティを使用してください。
 - ・ deployment-archive はデプロイのリビジョンのコンテンツを含むディレクトリです。
 - logs は scripts.log ファイルを含むディレクトリです。このファイルには、デプロイの AppSpec ファイルで指定されたスクリプトすべての出力が一覧表示されます。

デプロイのフォルダを検索するが、そのデプロイ ID またはデプロイグループ ID がわからない場合は、 <u>AWS CodeDeploy コンソール</u>または AWS CLI を使用して検索できます。詳細について は、「CodeDeploy デプロイの詳細を表示する 」を参照してください。 デプロイグループでアーカイブできるデプロイのデフォルト最大数は5件です。最大数に達する と、その後のデプロイがアーカイブされ、一番古いアーカイブは削除されます。CodeDeploy エー ジェントの設定ファイルで max_revisions 設定を使用すればデフォルトを変更できます。詳細につ いては、「CodeDeploy エージェント設定リファレンス」を参照してください。

Note

アーカイブしたデプロイが使用したハードディスク容量を復元するには、max_revisions 設定を1や2といった低い数値に変更してください。次のデプロイがアーカイブ済みのデ プロイを削除するので、指定した数値と同じになります。

- deployment-instructions には各デプロイグループのテキストファイル 4 件が含まれています。
 - [Deployment Group ID]-cleanup はデプロイ中に実行される各コマンドの undo バージョ ンを使うテキストファイルです。サンプルファイルの名前は acde1916-9099-7caffd21-012345abcdef-cleanup です。
 - [Deployment Group ID]-install.json は最新のデプロイ中に作成された JSON ファイルです。 これにはデプロイ中に実行するコマンドが含まれています。サンプルファイルの名前は acde1916-9099-7caf-fd21-012345abcdef-install.json です。
 - [Deployment Group ID]_last_successfull_install は、最後に成功したデプロイのアーカイブディ レクトリを示すテキストファイルです。これは CodeDeploy エージェントがデプロイアプリ ケーションのファイルすべてをインスタンスにコピーした時に作成されたファイルです。次回 のデプロイで、ApplicationStop スクリプトと BeforeInstall スクリプトのどちらを実 行するか決定するために CodeDeploy エージェントを使用します。サンプルファイルの名前は acde1916-9099-7caf-fd21-012345abcdef_last_successfull_install です。
 - [Deployment Group ID]_most_recent_install は、最新のデプロイのアーカイブディレクトリ名を リストにしたテキストファイルです。このファイルはデプロイ内のファイルが正常にダウンロー ドされた時に作成されます。ダウンロードしたファイルが最終的な場所にコピーされると、この ファイルの後に [deployment group ID]_last_successfull_install ファイルが作成されます。サンプ ルファイルの名前は acde1916-9099-7caf-fd21-012345abcdef_most_recent_install です。
- deployment-logs には次のログファイルが含まれています。
 - デプロイがある日ごとに codedeploy-agent.yyyymmdd.log ファイルが作成されます。各ログファイルには、その日のデプロイに関する情報が含まれています。アクセス権限の問題などをデバッグする場合に、こうしたログファイルが役に立ちます。初期状態のログファイル名はcodedeploy-agent.log です。翌日、デプロイの日付がファイル名に挿入されます。たとえ

ば、今日の日付が 2018 年 1 月 3 日だとします。この場合、その日のデプロイすべてに関する情報は codedeploy-agent.log で見ることができます。そして翌日の 2018 年 1 月 4 日に、ロ グファイル名は codedeploy-agent.20180103.log に変更されます。

 codedeploy-agent-deployments.log は各デプロイの scripts.log ファイルのコンテンツをコ ンパイルします。scripts.log ファイルは logs サブフォルダ (各 Deployment ID フォルダ 内) にあります。このファイル内のエントリにはデプロイ ID が付いています。たとえば、"[d-ABCDEF123]LifecycleEvent - BeforeInstall"はデプロイ中に d-ABCDEF123 の ID を 使用して書き込みを実行します。codedeploy-agent-deployments.log が最大サイズに達 した場合、CodeDeploy エージェントは古いコンテンツを削除しながら引き続き書き込みを実行 します。

CodeDeploy エージェントのオペレーションの管理

このセクションの手順では、CodeDeploy エージェントをインストール、アンインストール、再イン ストール、または更新する方法、および CodeDeploy エージェントが実行されていることを確認す る方法について説明します。

トピック

- CodeDeploy エージェントが実行されていることの確認
- CodeDeploy エージェントのバージョンを特定します。
- CodeDeploy エージェントをインストールする
- CodeDeploy エージェントを更新する
- CodeDeploy エージェントをアンインストールする
- CodeDeploy エージェントログを CloudWatch に送信する

CodeDeploy エージェントが実行されていることの確認

このセクションでは、インスタンスで CodeDeploy エージェントが実行を停止した可能性がある場 合に実行するコマンドについて説明します。

トピック

- Amazon Linux または RHEL 用の CodeDeploy エージェントが実行されていることを確認します。
- ・ Ubuntu サーバーの CodeDeploy エージェントが実行されていることを確認します。
- Windows サーバーの CodeDeploy エージェントが実行されていることを確認します。

Amazon Linux または RHEL 用の CodeDeploy エージェントが実行されていることを 確認します。

CodeDeploy エージェントがインストールされていて実行していることを確認するには、インスタン スにサインインし、次のコマンドを実行します。

systemctl status codedeploy-agent

コマンドがエラーを返す場合、CodeDeploy エージェントはインストールされていません。 「<u>Amazon Linux または RHEL 用の CodeDeploy エージェントをインストールする</u>」で説明されてい るようにインストールします。

CodeDeploy エージェントがインストールされて実行されている場合は、「The AWS CodeDeploy agent is running」のようなメッセージが表示されます。

「error: No AWS CodeDeploy agent running」のようなメッセージが表示される場合は、 サービスを起動し、次の2つのコマンドを一度に1つずつ実行します。

systemctl start codedeploy-agent

systemctl status codedeploy-agent

Ubuntu サーバーの CodeDeploy エージェントが実行されていることを確認します。

CodeDeploy エージェントがインストールされていて実行していることを確認するには、インスタン スにサインインし、次のコマンドを実行します。

systemctl status codedeploy-agent

コマンドがエラーを返す場合、CodeDeploy エージェントはインストールされていません。 「<u>Ubuntu サーバー用の CodeDeploy エージェントをインストールする</u>」で説明されているようにイ ンストールします。

CodeDeploy エージェントがインストールされて実行されている場合は、「The AWS CodeDeploy agent is running」のようなメッセージが表示されます。

「error: No AWS CodeDeploy agent running」のようなメッセージが表示される場合は、 サービスを起動し、次の2つのコマンドを一度に1つずつ実行します。 systemctl start codedeploy-agent

systemctl status codedeploy-agent

Windows サーバーの CodeDeploy エージェントが実行されていることを確認します。

CodeDeploy エージェントがインストールされていて実行していることを確認するには、インスタン スにサインインし、次のコマンドを実行します。

powershell.exe -Command Get-Service -Name codedeployagent

次のような出力が表示されます:

Status	Name	DisplayName
Running	codedeployagent	CodeDeploy Host Agent Service

コマンドがエラーを返す場合、CodeDeploy エージェントはインストールされていません。 「<u>Windows サーバー用の CodeDeploy エージェントです。</u>」で説明されているようにインストール します。

Status で Running 以外の何かが表示される場合は、次のコマンドでサービスを開始します。

powershell.exe -Command Start-Service -Name codedeployagent

次のコマンドを使ってサービスを再起動できます。

powershell.exe -Command Restart-Service -Name codedeployagent

次のコマンドを使ってサービスを停止できます。

powershell.exe -Command Stop-Service -Name codedeployagent

CodeDeploy エージェントのバージョンを特定します。

インスタンスで実行中の CodeDeploy エージェントのバージョンは、2 つの方法で特定できます。

最初に、バージョン 1.0.1.854 以降の CodeDeploy エージェントでは、インスタンスの,version ファイルでバージョン番号を確認できます。次の表に、サポートされるオペレーティングシステムご との場所とサンプルのバージョン文字列を示します。

オペレーティングシステム	ファイルの場所	サンプルの agent_version 文 字列
Amazon Linux および Red Hat Enterprise Linux(RHEL)	<pre>/opt/codedeploy-ag ent/.version</pre>	OFFICIAL_1.0.1.854_rpm
Ubuntu Server	<pre>/opt/codedeploy-ag ent/.version</pre>	OFFICIAL_1.0.1.854_deb
Windows Server	C:\ProgramData\Ama zon\CodeDeploy\.ve rsion	OFFICIAL_1.0.1.854_msi

2番目に、インスタンスでコマンドを実行して、CodeDeploy エージェントのバージョンを確認できます。

トピック

- Amazon Linux または RHEL でバージョンを確認する
- Ubuntu サーバーでバージョンを確認する
- Windows サーバーでバージョンをを確認する

Amazon Linux または RHEL でバージョンを確認する

インスタンスにサインインし、次のコマンドを実行します。

sudo yum info codedeploy-agent

Ubuntu サーバーでバージョンを確認する

インスタンスにサインインし、次のコマンドを実行します。

sudo dpkg -s codedeploy-agent

Windows サーバーでバージョンをを確認する

インスタンスにサインインし、次のコマンドを実行します。

sc qdescription codedeployagent

CodeDeploy エージェントをインストールする

EC2 インスタンスまたはオンプレミスサーバーで CodeDeploy を使用するには、まず CodeDeploy エージェントをインストールする必要があります。を使用して CodeDeploy エージェントをインス トールおよび更新することをお勧めします AWS Systems Manager。Systems Manager のさらなる 詳細については、「<u>AWS Systems Managerとは</u>」を参照してください。デプロイグループの作成時 にコンソールで Systems Manager を使用し、CodeDeploy エージェントのインストールと更新スケ ジュールを設定できます。

コマンドラインを使用して、S3 バケットから直接 CodeDeploy エージェントをインストールするこ ともできます。

インストールする推奨バージョンについては、「<u>CodeDeploy エージェントのバージョン履歴</u>」を参 照してください。

トピック

- AWS Systems Managerを使用して CodeDeploy エージェントをインストールする
- コマンドラインを使用して CodeDeploy エージェントをインストールする

AWS Systems Managerを使用して CodeDeploy エージェントをインストールする

AWS Management Console または を使用して AWS CLI、CodeDeploy エージェントを Amazon EC2 またはオンプレミスインスタンスにインストールできます AWS Systems Manager。特定の バージョンをインストールするか、常に最新バージョンのエージェントをインストールするかを選択 できます。の詳細については AWS Systems Manager、<u>「とは AWS Systems Manager</u>」を参照し てください。

CodeDeploy エージェントをインストールおよび更新するには、 を使用する AWS Systems Manager ことをお勧めします。また、Amazon S3 バケットか らCodeDeploy エージェントをインストール することもできます。Amazon S3 ダウンロードリンクの使用については、「 <u>コマンドラインを使用</u> して CodeDeploy エージェントをインストールする」を参照してください。 トピック

- 前提条件
- CodeDeploy エージェントをインストールする

前提条件

CodeDeploy の開始方法のステップに従って、IAM アクセス許可と AWS CLIを設定します。

Systems Manager を使用してオンプレミスサーバーに CodeDeploy エージェントをインストールす る場合は、オンプレミスサーバーを Amazon EC2 Systems Manager に登録する必要があります。詳 細については、<u>AWS Systems Manager ユーザーガイド</u>の ハイブリッド環境での Systems Manager のセットアップ を参照してください。

CodeDeploy エージェントをインストールする

Systems Manager を使用して CodeDeploy エージェントをインストールする前に、Systems Manager に合わせてインスタンスが適切に設定されていることを確認する必要があります。

SSM Agent のインストールまたは更新

Amazon EC2 インスタンスでは、CodeDeploy エージェントを使用するにはバージョン 2.3.274.0 以 降のインスタンスが実行されている必要があります。CodeDeploy エージェントをインストールする 前に、SSM Agent を更新するか、インスタンスにインストールしてください (まだの場合)。

SSM エージェントは、 が提供する一部の Amazon EC2 AMIs にプリインストールされていま す AWS。詳細については、「<u>SSM Agent がプリインストールされた Amazon Machine Images</u> (AMIs)」を参照してください。

Note

インスタンスのオペレーティング システムも CodeDeploy エージェントでサポートされて いることを確認します。詳細については、「<u>CodeDeploy エージェントで対応するオペレー</u> <u>ティングシステム</u>」を参照してください。

Linux を実行しているインスタンスでの SSM Agent のインストールまたは更新については、<u>AWS</u> <u>Systems Manager ユーザーガイド</u> の Linux インスタンスでの SSM Agent のインストールおよび設 定 を参照してください。 Windows Server を実行しているインスタンスでの SSM Agent のインストールまたは更新について は、<u>AWS Systems Manager ユーザーガイド</u> の Windows インスタンスでの SSM Agent のインス トールおよび設定 を参照してください。

(オプション) Systems Manager の前提条件を確認します。

Systems Manager Run Command を使用して CodeDeploy エージェントをインストールする前に、 インスタンスが Systems Manager の最低要件を満たしていることを確認してください。詳細につい ては、<u>AWS Systems Managerユーザーガイド</u>の「AWS Systems Manager のセットアップ」を参照 してください。

CodeDeploy エージェントをインストールする

SSM では、CodeDeploy を一度インストールするか、新しいバージョンをインストールするスケ ジュールを設定できます。

CodeDeploy エージェントをインストールするには、「ディストリビューターによ るAWSCodeDeployAgentパッケージのインストールまたは更新」のステップに従って、パッケージ を選択します。 <u>AWS Systems Manager</u>

コマンドラインを使用して CodeDeploy エージェントをインストールする

Note

エージェントのスケジュールされた更新を設定 AWS Systems Manager できるように、 で CodeDeploy エージェントをインストールすることをお勧めします。詳細については、 「<u>AWS Systems Managerを使用して CodeDeploy エージェントをインストールする</u>」を参 照してください。

以下のトピックに従い、コマンドラインを使用して CodeDeploy エージェントをインストールおよ び実行します。

トピック

- Amazon Linux または RHEL 用の CodeDeploy エージェントをインストールする
- Ubuntu サーバー用の CodeDeploy エージェントをインストールする
- Windows サーバー用の CodeDeploy エージェントです。

Amazon Linux または RHEL 用の CodeDeploy エージェントをインストールする

インスタンスにサインインし、次のコマンドを一度に1つずつ実行します。コマンド sudo yum update を最初に実行するのが、yum を使用してパッケージをインストールするときのベストプラク ティスと考えられていますが、すべてのパッケージを更新しない場合はこのコマンドをスキップでき ます。

sudo yum update

sudo yum install ruby

sudo yum install wget

(オプション) 以前のエージェントキャッシュ情報の AMI を消去するには、次のスクリプトを実行します。

```
#!/bin/bash
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
$CODEDEPLOY_BIN stop
yum erase codedeploy-agent -y
```

ホームディレクトリに移動します。

cd /home/ec2-user

Note

以前のコマンドで、/home/ec2-user は、Amazon Linux または RHEL Amazon EC2 イン スタンスのデフォルトのユーザー名を表しています。インスタンスがカスタム AMI を使用し て作成された場合、AMI 所有者は別のデフォルトのユーザー名を指定している可能性があり ます。

CodeDeploy エージェントのインストーラをダウンロードします。

wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install

bucket-name は、お住まいの地域用の CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。######## は、お住まいの地域の識別子です。

以下に例を示します。

https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install

バケット名とリージョン識別子のリストについては、「<u>リージョン別リソースキットバケット名</u>」を 参照してください。

install ファイルに実行権限を設定します。

chmod +x ./install

CodeDeploy エージェントの最新バージョンをインストールするには:

sudo ./install auto

CodeDeploy エージェントの特定のバージョンをインストールするには:

リージョンで使用可能なバージョンを一覧表示します。

以下のいずれかのバージョンをインストールします。

sudo ./install auto -v releases/codedeploy-agent-version.noarch.rpm

Note

AWS は、CodeDeploy エージェントの最新マイナーバージョンをサポートしています。現 在、最新のマイナーバージョンは 1.7.x です。

サービスが実行されているかどうか確認するには、次のコマンドを実行します。

systemctl status codedeploy-agent

CodeDeploy エージェントがインストールされて実行されている場合は、「The AWS CodeDeploy agent is running」のようなメッセージが表示されます。

「 error: No AWS CodeDeploy agent running 」のようなメッセージが表示される場合は、 サービスを起動し、次の2つのコマンドを一度に1つずつ実行します。

systemctl start codedeploy-agent

systemctl status codedeploy-agent

Ubuntu サーバー用の CodeDeploy エージェントをインストールする

Note

エージェントのスケジュールされた更新を設定 AWS Systems Manager できるように、 で CodeDeploy エージェントをインストールすることをお勧めします。詳細については、 「<u>AWS Systems Managerを使用して CodeDeploy エージェントをインストールする</u>」を参 照してください。

CodeDeploy エージェントを Ubuntu サーバーにインストールするには

1. インスタンスにサインインします。

2. 次のコマンドを順々に入力します。

sudo apt update

sudo apt install ruby-full

sudo apt install wget

3. 次のコマンドを入力します。

cd /home/ubuntu

<u>/home/ubuntu</u>は、Ubuntu Server インスタンスのデフォルトのユーザー名を表しています。 インスタンスがカスタム AMI を使用して作成された場合、AMI 所有者は別のデフォルトのユー ザー名を指定している可能性があります。

4. 次のコマンドを入力します。

wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install

bucket-name は、お住まいの地域用の CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。######## は、お住まいの地域の識別子です。

以下に例を示します。

https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/ install

バケット名とリージョン識別子のリストについては、「<u>リージョン別リソースキットバケット</u> 名」を参照してください。

5. 次のコマンドを入力します。

chmod +x ./install

- 6. 次のいずれかを行います:
 - ・ CodeDeploy エージェントの最新バージョンをサポートされている 20.04 以外の Ubuntu にイ ンストールするには:

sudo ./install auto

• CodeDeploy エージェントの最新バージョンを Ubuntu Server 20.04 にインストールするには:

Note

出力を一時ログファイルに書き込むことで、Ubuntu Server 20.04 の install スクリ プトにある既知のバグを回避できます。このバグは現在修正中です。

sudo ./install auto > /tmp/logfile

- ・ CodeDeploy エージェントの特定のバージョンをサポートされている 20.04 以外の Ubuntu Server にインストールするには:
 - リージョンで使用可能なバージョンを一覧表示します。

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-
identifier | grep '\.deb$'
```

• 以下のいずれかのバージョンをインストールします。

sudo ./install auto -v releases/codedeploy-agent-###.deb

Note

AWS は、CodeDeploy エージェントの最新マイナーバージョンをサポートしています。現在、最新のマイナーバージョンは 1.7.x です。

- CodeDeploy エージェントの特定のバージョンを Ubuntu Server 20.04 にインストールするには:
 - リージョンで使用可能なバージョンを一覧表示します。

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-
identifier | grep '\.deb$'
```

以下のいずれかのバージョンをインストールします。

sudo ./install auto -v releases/codedeploy-agent-###.deb > /tmp/logfile

Note

出力を一時ログファイルに書き込むことで、Ubuntu Server 20.04 の install スク リプトにある既知のバグを回避できます。このバグは現在修正中です。

Note

AWS は、CodeDeploy エージェントの最新マイナーバージョンをサポートしています。現在、最新のマイナーバージョンは 1.7.x です。

サービスが実行されていることをチェックするには

1. 次のコマンドを入力します。

systemctl status codedeploy-agent

CodeDeploy エージェントがインストールされて実行されている場合は、「The AWS CodeDeploy agent is running」のようなメッセージが表示されます。

 error: No AWS CodeDeploy agent running」のようなメッセージが表示される場合 は、サービスを起動し、次の2つのコマンドを一度に1つずつ実行します。

systemctl start codedeploy-agent

systemctl status codedeploy-agent

Windows サーバー用の CodeDeploy エージェントです。

Windows サーバーのインスタンスでは、次のいずれかの方法を使用して、CodeDeploy エージェントをダウンロードしてインストールできます。

- 使用 AWS Systems Manager (推奨)
- 一連の Windows PowerShell コマンドを実行。
- 直接ダウンロードリンクを選択。
- Amazon S3 コピーコマンドを実行してください。
 - Note

CodeDeploy エージェントがインストールされているフォルダは C:\Program Data \Amazon\CodeDeploy です。このパスにディレクトリジャンクションまたはシンボリック リンクがないことを確認します。

トピック

- 使用アイテム Systems Manager
- ・ Windows PowerShell の使用

- 直接接続の使用
- Amazon S3 コピーコマンドの使用

使用アイテム Systems Manager

「<u>AWS Systems Managerを使用して CodeDeploy エージェントをインストールする</u>」の手順に従っ て CodeDeploy エージェントをインストールします。

Windows PowerShell の使用

インスタンスにサインインし、Windows PowerShell で次のコマンドを実行します。

 インターネットからダウンロードされたすべてのスクリプトと設定ファイルが、信頼された発行 元によって署名されていることを要求します。実行ポリシーの変更を求められた場合は「Y」 と入力します。

Set-ExecutionPolicy RemoteSigned

2. をロードします AWS Tools for Windows PowerShell。

Import-Module AWSPowerShell

3. CodeDeploy エージェントのインストールファイルがダウンロードされた場所で、ディレクトリ を作成します。

New-Item -Path "c:\temp" -ItemType "directory" -Force

- Set-AWSCredential および Initialize-AWSDefaultConfiguration コマンドを使用して AWS 認証情報を設定します。詳細については、<u>AWS Tools for PowerShell ユーザーガイドに</u>ある「AWS 認証情報の使用」を参照してください。
- 5. CodeDeploy エージェントのインストールファイルをダウンロードします。

Note

AWS は、CodeDeploy エージェントの最新マイナーバージョンをサポートしています。 現在、最新のマイナーバージョンは 1.7.x です。

CodeDeploy エージェントの最新バージョンをインストールするには:

powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/ codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi

CodeDeploy エージェントの特定のバージョンをインストールするには:

powershell.exe -Command Read-S3Object -BucketName bucket-name -Key releases/ codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi

は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を awscodedeploy-us-east-2 に置き換えます。バケット名のリストについては、「<u>リージョン別</u> リソースキットバケット名」を参照してください。

6. CodeDeploy エージェントのインストールファイルを実行します。

c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt

サービスが実行されているかどうか確認するには、次のコマンドを実行します。

powershell.exe -Command Get-Service -Name codedeployagent

CodeDeploy エージェントがインストールされただけで、まだ開始されていない場合は、Get-Service コマンドを実行すると、[ステータス] に「Start...」と表示されます。

Status	Name	DisplayName
Start	codedeployagent	CodeDeploy Host Agent Service

CodeDeploy エージェントが既に実行されている場合は、Get-Service コマンドを実行すると、[ス テータス] に「Running」と表示されます。

Status	Name	DisplayName
Running	codedeployagent	CodeDeploy Host Agent Service

直接接続の使用

Windows サーバーのインスタンスのブラウザのセキュリティ設定でアクセス権限 (例え ば、https://s3.*.amazonaws.com への) が提供されている場合は、リージョンの直接接続を使 用して CodeDeploy エージェントをダウンロードし、インストーラを手動で実行できます。

リンクは以下のとおりです。

https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent.msi

…#####はアプリケーションをデプロイする AWS リージョンです。

以下に例を示します。

https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latest/codedeployagent.msi

▲ Important

CodeDeploy アプリケーションと同じリージョンから .msi ファイルを取得します。別の リージョンを選択すると、.msi ファイルを実行したときに codedeploy-agent-log ファ イルに inconsistent region 障害が発生する可能性があります。

Amazon S3 コピーコマンドの使用

AWS CLI がインスタンスにインストールされている場合は、Amazon S3 <u>cp</u> コマンドを使用して CodeDeploy エージェントをダウンロードし、インストーラを手動で実行できます。詳細について は、<u>「Microsoft Windows AWS Command Line Interface に をインストールする</u>」を参照してくださ い。

Amazon S3 コマンドは以下のとおりです。

aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent.msi codedeploy-agent.msi
 --region region

…#####はアプリケーションをデプロイする AWS リージョンです。

以下に例を示します。

aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent.msi codedeployagent.msi --region af-south-1

CodeDeploy エージェントを更新する

AWS Systems Managerを使用して、すべてのサポートされているオペレーティングシステムで CodeDeploy エージェントの自動更新スケジュールを設定できます。また、インスタンスでコマンド を実行して、サポートされているすべてのオペレーティングシステムで更新を強制することもできま す。

トピック

- Amazon Linux または RHEL で CodeDeploy エージェントを更新する
- Ubuntu サーバーで CodeDeploy エージェントを更新する
- Windows サーバーで CodeDeploy エージェントを更新する

Amazon Linux または RHEL で CodeDeploy エージェントを更新する

を使用して CodeDeploy エージェントの自動スケジュール更新を設定するには、「<u>を使用して</u> <u>CodeDeploy エージェントをインストールする AWS Systems Manager</u>」のステップ AWS Systems Managerに従います。

CodeDeploy エージェントの更新を強制する場合は、インスタンスにサインインし、次のコマンドを 実行します。

sudo /opt/codedeploy-agent/bin/install auto

Ubuntu サーバーで CodeDeploy エージェントを更新する

を使用して CodeDeploy エージェントの自動スケジュール更新を設定するには、「 <u>を使用して</u> <u>CodeDeploy エージェントをインストールする AWS Systems Manager</u>」のステップ AWS Systems Managerに従います。

CodeDeploy エージェントの更新を強制する場合は、インスタンスにサインインし、次のコマンドを 実行します。

sudo /opt/codedeploy-agent/bin/install auto

CodeDeploy エージェントを更新する

API バージョン 2014-10-06 300

Windows サーバーで CodeDeploy エージェントを更新する

を使用して CodeDeploy エージェントの自動更新を有効にできます AWS Systems Manager。Systems Manager では、Systems Manager State Manager との関連付けを作成し て、Amazon EC2 またはオンプレミスインスタンスの更新スケジュールを設定できます。現在の バージョンをアンインストールして新しいバージョンをインストールすることで、CodeDeploy エー ジェントを手動で更新することもできます。

トピック

- で CodeDeploy エージェントの自動更新を設定する AWS Systems Manager
- CodeDeploy エージェントを手動で更新する
- (非推奨) Windows サーバーアップデータを使用して CodeDeploy エージェントを更新する

で CodeDeploy エージェントの自動更新を設定する AWS Systems Manager

Systems Manager を設定し、CodeDeploy エージェントの自動更新を有効にするには、「<u>を使用し</u> て CodeDeploy エージェントをインストールする AWS Systems Manager」の手順に従います。

CodeDeploy エージェントを手動で更新する

CodeDeploy エージェントを手動で更新するには、CLI または Systems Manager を使用して最 新バージョンをインストールします。「<u>CodeDeploy エージェントをインストールする</u>」の手順 に従います。<u>CodeDeploy エージェントのアンインストール</u>の手順に従って、古いバージョンの CodeDeploy エージェントをアンインストールすることをお勧めします。

(非推奨) Windows サーバーアップデータを使用して CodeDeploy エージェントを更新する

Note

Windows サーバー用の CodeDeploy エージェントアップデータは非推奨となり、1.0.1.1597 より後のバージョンには更新されなくりました。

CodeDeploy エージェントの自動更新を有効にするには、新しいインスタンスまたは既存のインス タンスに Windows サーバー用の CodeDeploy エージェントアップデーターをインストールします。 アップデータは新しいバージョンを定期的に確認します。新しいバージョンが検出された場合、アッ プデータは、最新バージョンをインストールする前に、インストールされている場合はエージェント の現在のバージョンをアンインストールします。 アップデータが新しいバージョンが検出したときにデプロイが既に進行中の場合、デプロイは完了す るまで続行されます。更新プロセス中にデプロイの開始が試みられた場合、デプロイは失敗します。

CodeDeploy エージェントの更新を強制する場合は、「<u>Windows サーバー用の CodeDeploy エー</u> ジェントです。」の手順に従います。

Windows サーバーのインスタンスでは、CodeDeploy エージェントアップデーターをダウンロード してインストールできます。これを行うには、一連の Windows PowerShell コマンドを実行するか、 直接ダウンロードリンクを使用するか、または Amazon S3 コピーコマンドを実行します。

トピック

- Windows PowerShell の使用
- 直接接続の使用
- Amazon S3 コピーコマンドの使用

Windows PowerShell の使用

インスタンスにサインインし、Windows PowerShell で、一度に 1 つずつ次のコマンドを実行しま す。

Set-ExecutionPolicy RemoteSigned

実行ポリシーの変更を求められた場合は、「Y」を選択し、Windows PowerShell により、インター ネットからダウンロードされるすべてのスクリプトと設定ファイルが、信頼された発行元によって署 名されていることが要求されるようにします。

Import-Module AWSPowerShell

New-Item -Path "c:\temp" -ItemType "directory" -Force

powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeployagent-updater.msi -File c:\temp\codedeploy-agent-updater.msi

c:\temp\codedeploy-agent-updater.msi /quiet /l c:\temp\host-agent-updater-log.txt

powershell.exe -Command Get-Service -Name codedeployagent

は、リージョンの CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。例えば、米国東部 (オハイオ) リージョンの場合、##### を aws-codedeploy-useast-2 に置き換えます。バケット名のリストについては、「<u>リージョン別リソースキットバケット</u> 名」を参照してください。

更新プロセスのエラーをトラブルシューティングする必要がある場合は、次のコマンドを入力して CodeDeploy エージェントアップデータのログファイルを開きます。

notepad C:\ProgramData\Amazon\CodeDeployUpdater\log\codedeploy-agent.updater.log

直接接続の使用

Windows サーバーのインスタンスのブラウザのセキュリティ設定で必要なアクセス許可 (例えば、 http://s3.*.amazonaws.com への) が提供されている場合は、直接接続を使用して CodeDeploy エージェントアップデーターをダウンロードできます。

リンクは以下のとおりです。

https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agentupdater.msi

…#####はアプリケーションを更新する AWS リージョンです。

以下に例を示します。

https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latest/codedeploy-agentupdater.msi

Amazon S3 コピーコマンドの使用

AWS CLI がインスタンスにインストールされている場合は、Amazon S3 <u>cp</u> コマンドを使用して CodeDeploy エージェントアップデーターをダウンロードし、インストーラーを手動で実行できま す。詳細については、<u>「Microsoft Windows AWS Command Line Interface に をインストールする</u>」 を参照してください。

Amazon S3 コマンドは以下のとおりです。

aws s3 cp s3://aws-codedeploy-*region*/latest/codedeploy-agent-updater.msi codedeployagent-updater.msi --region *region*

…<mark>#####はアプリケーションを更新する AWS リージョンです。</mark>

以下に例を示します。

aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent-updater.msi
codedeploy-agent-updater.msi --region af-south-1

CodeDeploy エージェントをアンインストールする

不要になった場合や、新しいインストールを実行する場合は、インスタンスから CodeDeploy エー ジェントを削除できます。

Amazon Linux または RHEL から CodeDeploy エージェントをアンインストールする

CodeDeploy エージェントをアンインストールするには、インスタンスにサインインし、次のコマン ドを実行します。

sudo yum erase codedeploy-agent

Ubuntu Server から CodeDeploy エージェントをアンインストール

CodeDeploy エージェントをアンインストールするには、インスタンスにサインインし、次のコマン ドを実行します。

sudo dpkg --purge codedeploy-agent

Windows サーバーから CodeDeploy エージェントをアンインストールする

CodeDeploy エージェントをアンインストールするには、インスタンスにサインインし、次の3つの コマンドを1つずつ実行します。

wmic

product where name="CodeDeploy Host Agent" call uninstall /nointeractive

exit

または、インスタンスにサインインし、[Control Panel (コントロールパネル)] で [Programs and Features (プログラムと機能)]を開き、[CodeDeploy Host Agent (CodeDeploy ホストエージェント)]を選択してから [Uninstall (アンインストール)]を選択します。

CodeDeploy エージェントログを CloudWatch に送信する

CodeDeploy エージェントのメトリクスとログデータを CloudWatch に送信するには、<u>統合した</u> CloudWatch エージェント、またはより簡単に CloudWatch エージェントを使用します。

以下の手順に従って CloudWatch エージェントをインストールし、CodeDeploy エージェントで使用 するように設定します。

前提条件

開始する前に、以下のタスクを完了します。

- CodeDeploy エージェントをインストールし、実行されていることを確認します。詳細については、<u>CodeDeploy エージェントをインストールする</u>および<u>CodeDeploy エージェントが実行されていることの確認</u>を参照してください。
- ・ CloudWatch エージェントをインストールします。詳細については、「<u>CloudWatch エージェント</u> <u>のインストール</u>」を参照してください。
- 次のアクセス権現を CodeDeploy IAM インスタンスプロファイルに追加します。
 - CloudWatchLogsFullAccess
 - CloudWatchAgentServerPolicy

CodeDeploy インスタンスプロファイルの詳細については、<u>CodeDeploy の開始方法</u>の「<u>ステップ</u> <u>4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成する</u>」を参照してくださ い。

CodeDeploy ログを収集するための CloudWatch エージェントの設定

CloudWatch エージェントは、ウィザードを使用するか、設定ファイルを手動で作成または編集する ことで設定できます。

ウィザードを使用して CloudWatch エージェントを設定する (Linux)

- 1. 「<u>CloudWatch エージェント設定ウィザードを実行する</u>」の説明に従ってウィザードを実行しま す。
- ウィザードで、Do you want to monitor any log files?と表示されたら、1と入力します。
- 3. CodeDeploy エージェントのログファイルを次のように指定します。

- 1. Log file pathには、CodeDeploy ログファイルのパス (例: /var/log/aws/ codedeploy-agent/codedeploy-agent.log) を入力します。
- 2. Log group name には、ロググループ名 (例: codedeploy-agent-log) を入力します。
- 3. Log stream name には、ログストリーム名 (例: **{instance_id}-codedeploy-agentlog**) を入力します。
- Do you want to specify any additional log files?と表示されたら、1と入力します。
- 5. CodeDeploy エージェントのデプロイログを次のように指定します。
 - Log file pathには、CodeDeployデプロイログファイルのパス(例: /opt/ codedeploy-agent/deployment-root/deployment-logs/codedeploy-agentdeployments.log)を入力します。
 - 2. Log group name には、ロググループ名 (例: codedeploy-agent-deployment-log) を 入力します。
 - 3. Log stream name には、ログストリーム名 (例: **{instance_id}-codedeploy-agentdeployment-log**) を入力します。
- 6. Do you want to specify any additional log files?と表示されたら、1と入力しま す。
- 7. CodeDeploy エージェントのアップデータログを次のように指定します。
 - 1. Log file path には、CodeDeploy アップデータログファイルのパス (例: **/tmp/** codedeploy-agent.update.log) を入力します。
 - 2. Log group name には、ロググループ名 (例: codedeploy-agent-updater-log) を入力 します。
 - 3. Log stream name には、ログストリーム名 (例: **{instance_id}-codedeploy-agent**updater-log) を入力します。

ウィザードを使用して CloudWatch エージェントを設定する (Windows)

- 「<u>CloudWatch エージェント設定ウィザードを実行する</u>」の説明に従ってウィザードを実行します。
- ウィザードで、Do you want to monitor any customized log files?と表示されたら、1と入力します。
- 3. CodeDeploy ログファイルを次のように指定します。

- Log file pathには、パスまたは CodeDeploy エージェントのログファイル (例: C: \ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt)を入力しま す。
- 2. Log group name には、ロググループ名 (例: codedeploy-agent-log) を入力します。
- 3. Log stream name には、ログストリーム名 (例: **{instance_id}-codedeploy-agent-log**) を入力します。
- Do you want to specify any additional log files?と表示されたら、1と入力します。
- 5. CodeDeploy エージェントのデプロイログを次のように指定します。
 - Log file pathには、CodeDeployデプロイログファイルのパス(例: C:\ProgramData \Amazon\CodeDeploy\deployment-logs\codedeploy-agent-deployments.log) を入力します。
 - Log group name には、ロググループ名 (例:codedeploy-agent-deployment-log) を入 力します。
 - 3. Log stream name には、ログストリームの名前 (例: **{instance_id}-codedeploy-agent-deployment-log**) を入力します。

設定ファイルを手動で作成または編集して CloudWatch エージェントを設定する (Linux)

- 1. 「<u>CloudWatch エージェント設定ファイルを手動で作成または編集する</u>」の説明に従って CloudWatch エージェント設定ファイルを作成または編集します。
- ファイル名は /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatchagent.json で、次のコードが含まれていることを確認してください。



設定ファイルを手動で作成または編集して CloudWatch エージェントを設定する (Windows)

- 「<u>CloudWatch エージェント設定ファイルを手動で作成または編集する</u>」の説明に従って CloudWatch エージェント設定ファイルを作成または編集します。
- ファイル名は C:\ProgramData\Amazon\AmazonCloudWatchAgent\amazoncloudwatch-agent.json で、次のコードが含まれていることを確認してください。

```
. .
"logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\\log\
\codedeploy-agent-log.txt",
                        "log_group_name": "codedeploy-agent-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-log"
                    },
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\
\deployment-logs\\codedeploy-agent-deployments.log",
                        "log_group_name": "codedeploy-agent-deployment-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-
deployment-log"
```



CloudWatch エージェントを再起動する

変更を加えたら、「<u>CloudWatch エージェントを起動する</u>」の説明に従って CloudWatch エージェン トを再起動します。

CodeDeploy のためにインスタンスを用いた操作

CodeDeploy は Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL)、Windows Server を実行しているインスタンスへのデプロイをサポートします。

CodeDeploy を使って、Amazon EC2 インスタンスとオンプレミスインスタンスの両方にデプ ロイできます。オンプレミスインスタンスは、CodeDeploy エージェントを実行してパブリック AWS サービスエンドポイントに接続できる Amazon EC2 インスタンスではない物理デバイスで す。CodeDeploy を使って、クラウド中の Amazon EC2 インスタンスと、オフィスのデスクトップ PC や自身のデータセンターのサーバーに、アプリケーションを同時にデプロイできます。

Amazon EC2 インスタンスとオンプレミスインスタンスの比較

次の表は、Amazon EC2 インスタンスとオンプレミスインスタンスの比較を示しています。

件名	Amazon EC2 インスタンス	オンプレミスインスタンス
インスタンス上で実行中のオ ペレーティングシステムと互 換性がある CodeDeploy エー ジェントのバージョンをイン ストールおよび実行する必要 があります。	あり	あり
インスタンスは、CodeDeploy に接続できる必要がありま す。	あり	あり
IAM インスタンスプロファイ ルがインスタンスに添付され る必要があります。IAM イ ンスタンスプロファイルに は、CodeDeploy デプロイに 参加する許可が必要です。詳 細については、 <u>ステップ 4:</u> <u>Amazon EC2 インスタンス用</u> <u>の IAM インスタンスプロファ</u>	はい	いいえ
件名	Amazon EC2 インスタンス	オンプレミスインスタンス
--	-------------------	--------------
<u>イルを作成する</u> を参照してく ださい。		
次のいずれかの操作を行って 認証を行い、インスタンスを 登録する必要があります。	いいえ	はい
 AWS Security Token Serviceを通して生成され、 定期的に更新される一時 的な認証情報を取得する ための各インスタンス上 で、IAM ユーザーによって 推定され得る IAM ロールを 作成します。 各インスタンスの IAM ユー ザーを作成し、インスタン ス上に簡易なテキストで IAM ユーザーのアカウント 認証情報を保存します。 		
それへ実行できる前 に、CodeDeploy を用いて各 インスタンスを登録する必要 があります。	いいえ	はい
CodeDeploy がデプロイを実 行できる前に、各インスタン スにタグを付ける必要があり ます。	あり	あり
CodeDeploy デプロイの一環 として、Amazon EC2 Auto Scaling および Elastic Load Balancing のシナリオに参加で きます。	はい	いいえ

件名	Amazon EC2 インスタンス	オンプレミスインスタンス
Amazon S3 バケットと GitHub リポジトリからデプロ イされることができます。	あり	あり
指定されたイベントがデプロ イまたはインスタンスで発生 したときに、SMS または E メール通知の送信を求めるト リガーをサポートできる。	あり	あり
関連デプロイへの請求対象で ある。	いいえ	はい

CodeDeploy のためのインスタンスタスク

デプロイで使用するインスタンスを起動または設定するには、以下の手順から選択します。

新しい Amazon Linux または Windows Server の Amazon EC2 インスタンスを起動する場 合。	最小限の労力で Amazon EC2 インスタンスを 起動するには、 <u>CodeDeploy 用の Amazon EC2</u> <u>インスタンスを作成する (AWS CloudForm</u> <u>ation テンプレート)</u> を参照してください。 主に自分で Amazon EC2 インスタンスを起 動するには、 <u>CodeDeploy (AWS CLI または</u> <u>Amazon EC2 コンソール) 用の Amazon EC2</u> <u>インスタンスを作成する</u> を参照してください 。
新しい Ubuntu Server または RHEL の Amazon EC2 インスタンスを起動する場合。	「 <u>CodeDeploy (AWS CLI または Amazon EC2</u> コンソール) 用の Amazon EC2 インスタンスを <u>作成する</u> 」を参照してください。
Amazon Linux、Windows サーバー、Ubuntu サーバー、または RHEL Amazon EC2 インス タンスを構成する場合。	「 <u>CodeDeploy と共に動作するように Amazon</u> <u>EC2 インスタンスを構成します</u> 」を参照して ください。

Windows サーバー、Ubuntu サーバー、また は RHEL オンプレミスインスタンス (Amazon EC2 インスタンスではない物理デバイス) を構 成する場合。

blue/green のデプロイ中に、CodeDeploy にイ ンスタンスの置換フリートをプロビジョンする ための CodeDeploy が必要な場合。 「<u>Working with On-Premises Instances</u>」を参 照してください。

「<u>CodeDeploy でのデプロイグループの使用</u>」 を参照してください。

Amazon EC2 Auto Scaling グループ中の Amazon EC2 インスタンスを準備するには、追加のステッ プに従う必要があります。詳細については、「<u>CodeDeploy と Amazon EC2 Auto Scaling の統合</u>」 を参照してください。

トピック

- Tagging Instances for Deployments
- Working with Amazon EC2 Instances
- Working with On-Premises Instances
- View Instance Details
- Instance Health

CodeDeploy 中のデプロイグループのためのインスタンスのタグ付 け

Amazon EC2 インスタンスとオンプレミスインスタンスを管理するために、タグを使用して独自の メタデータを各リソースに割り当てることができます。タグを使用すると、インスタンスをさまざま な方法 (目的、所有者、環境など) で分類することができます。これはインスタンスが多数ある場合 に便利です。割り当てられたタグに基づいて、インスタンスやインスタンスグループを迅速に識別で きます。タグはそれぞれ、1 つのキーとオプションの 1 つの値で設定されており、どちらもお客様側 が定義します。詳細については、「<u>Amazon EC2 リソースにタグを付ける</u>」を参照してください。

どのインスタンスを CodeDeploy デプロイグループ中に含めるかを指定するには、1 つ以上の tag groups にタグを指定します。タグ条件を満たすインスタンスは、そのデプロイグループへのデプロ イが作成されたときにアプリケーションの最新のリビジョンがインストール済みのものです。 Note

また、Amazon EC2 Auto Scaling グループをデプロイグループに含めることもできますが、 これらはインスタンスに適用されたタグではなく名前によって識別されます。詳細について は、CodeDeploy と Amazon EC2 Auto Scaling の統合 を参照してください。

デプロイグループのインスタンスの条件は、単一のタググループの単一のタグと同じほど簡単です。 これは、最大 3 つのタググループそれぞれに 10 個のタグという複雑なものにもできます。

単一のタググループを使用する場合は、グループ内の少なくとも1つのタグによって識別されたイ ンスタンスがデプロイグループに含まれます。複数のタググループを使用する場合は、タググルー プそれぞれの少なくとも1つのタグによって識別されたインスタンスのみが含まれます。

次の例は、タグとタググループを使用してデプロイグループのインスタンスを選択する方法を説明し ます。

トピック

- 例 1: 単一タググループ、単一タグ
- 例 2: 単一タググループ、複数タグ
- 例 3: 複数タググループ、複数タグ
- 例 4: 複数タググループ、複数タグ

例 1: 単一タググループ、単一タグ

単一のタグを単一のタググループに指定できます。

タググループ 1

+-	值
名前	AppVersion-ABC

Name=AppVersion-ABC というタグが付いている各インスタンスは、他のタグがついていても、デ プロイグループの一部になります。

CodeDeploy コンソールのセットアップビュー。

Amazon EC2 instances		
You can add up to three groups of tags for EC2 instances to this deployment group. One tag group: Any instance identified by the tag group will be deployed to. Multiple tag groups: Only instances identified by all the tag groups will be deployed to.		
Tag group 1		
Key - optional	Value - optional	
Q Name X	Q AppVersion-ABC X	
Q	Q	Remove tag

JSON の構造:



例 2: 単一タググループ、複数タグ

単一のタグを複数のタググループに指定することもできます。

タググループ 1

+-	值
リージョン	North
リージョン	South
リージョン	East

この3つのタグのうちいずれかが付いているインスタンスは、他のタグがついていても、デプロイ グループの一部になります。たとえば、Region=West というタグが付いている他のインスタンスが ある場合、それらはデプロイグループに含まれません。

CodeDeploy コンソールのセットアップビュー。

Amazon EC2 instances			
You can add up to three groups of tags for EC2 instances to this deployment group. One tag group: Any instance identified by the tag group will be deployed to. Multiple tag groups: Only instances identified by all the tag groups will be deployed to.			
Tag group 1			
Key - optional	Value - optional		
Q Region X	Q North	×	
Q Region X	Q South	×	Remove tag
Q Region X	Q East	×	Remove tag

JSON の構造:

<pre>"ec2TagSet": {</pre>	
"ec2TagS	etList": [
{	
	"Type": "KEY_AND_VALUE",
	"Key": "Region",
	"Value": "North"
},	
	{
	"Type": "KEY_AND_VALUE",
	"Key": "Region",
	"Value": "South"
},	
	{
	"Type": "KEY_AND_VALUE",
	"Key": "Region",
	"Value": "East"
}	

]] },

例 3: 複数タググループ、複数タグ

それぞれに単一のキーと値のペアを持つタググループの複数セットを使用して、デプロイグループの インスタンスの条件を指定することもできます。デプロイグループで複数のタググループを使用する 場合は、すべてのタググループによって識別されたインスタンスのみがデプロイグループに含まれま す。

タググループ1

+-	值
名前	AppVersion-ABC
タググループ 2	
+-	值
リージョン	North
タググループ 3	
+-	值
タイプ	t2.medium

多くのリージョンにインスタンスがあり、Name=AppVersion-ABC のタグが付いたさまざま なインスタンスタイプがある場合があります。この例では、同じく Region=North および Type=t2.medium のタグが付いたインスタンスのみがデプロイグループの一部になります。

CodeDeploy コンソールのセットアップビュー。

Amazon EC2 instances			
You can add up to three groups of tags for B One tag group: Any instance identified by t Multiple tag groups: Only instances identif	EC2 instances to this deployment group. he tag group will be deployed to. ied by all the tag groups will be deployed t	.0.	
Tag group 1			
Key - optional	Value - optional		
Q Name X	Q AppVersion-ABC	×	
Add tag			
Tag group 2			
Key - optional	Value - optional		
Q Region X	Q North	×	
Add tag			
			S Remove tag group
Tag group 3			
Key - optional	Value - optional		
Q Туре Х	Q t2.medium	×	
Add tag			
			S Remove tag group

JSON の構造:

<pre>"ec2TagSet": {</pre>	
"ec2TagS	etList": [
Γ	
{	
	"Type": "KEY_AND_VALUE",
	"Key": "Name",
	"Value": "AppVersion-ABC"
}	
],	



例 4: 複数タググループ、複数タグ

複数のタグを持つ複数のタググループを1つ以上のグループで使用する場合、インスタンスはそれ ぞれのグループの少なくとも1つのタグに一致している必要があります。

タググループ 1

+-	值
環境	ベータ
環境	ステージング

タググループ 2

+-	值
リージョン	North
リージョン	South
リージョン	East

タググループ 3

+-	值
タイプ	t2.medium
タイプ	t2.large

この例では、デプロイグループに含まれるには、インスタンスが次のようにタグ付けされ ている必要があります。(1) Environment=Beta または Environment=Staging、(2) Region=North、Region=South または Region=East、(3) Type=t2.medium または Type=t2.large。

たとえば、次のタググループを持つインスタンスは、デプロイグループに含まれるもののひとつにな ることがあります。

- Environment=Beta, Region=North,Type=t2.medium
- Environment=Staging,Region=East,Type=t2.large
- Environment=Staging,Region=South,Type=t2.large

次のタググループを持つインスタンスは、デプロイグループに含まれないことがあります。強調表示 されたキー値があると、インスタンスは除外されます。

- Environment=Beta, Region=West,Type=t2.medium
- Environment=Staging,Region=East,Type=t2.micro
- Environment=Production,Region=South,Type=t2.large

CodeDeploy コンソールのセットアップビュー。

Amazon EC2 instances		
You can add up to three groups of tags for I One tag group: Any instance identified by t Multiple tag groups: Only instances identif	EC2 instances to this deployment group. The tag group will be deployed to. Tied by all the tag groups will be deployed to.	
Tag group 1		
Key - optional	Value - optional	
Q Environment X	Q Staging X	
Q Environment X	Q Beta X	Remove tag
Add tag		
Tag group 2		
Key - optional	Value - optional	
Q Region X	Q South X	
Q Region X	Q North X	Remove tag
Q Region X	Q East X	Remove tag
Add tag		
		S Remove tag group
Tag group 3		
Key - optional	Value - optional	
Q Type X	Q t2.medium	
Q Type X	Q t2.large	Remove tag
Add tag		
		S Remove tag group

JSON の構造:

```
"ec2TagSet": {
         "ec2TagSetList": [
            Ε
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Environment",
                   "Value": "Beta"
               },
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Environment",
                   "Value": "Staging"
               }
            ],
            Ε
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Region",
                   "Value": "North"
               },
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Region",
                   "Value": "South"
               },
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Region",
                   "Value": "East"
               }
            ],
            Γ
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Type",
                   "Value": "t2.medium"
               },
               {
                   "Type": "KEY_AND_VALUE",
                   "Key": "Type",
                   "Value": "t2.large"
               }
```

],] },

CodeDeploy のための Amazon EC2 インスタンスを用いた作業

Amazon EC2 インスタンスは、Amazon Elastic Compute Cloud を使用して作成および設定する仮想 コンピューティング環境です Amazon EC2 は、 AWS クラウドでスケーラブルなコンピューティン グ性能を提供します。Amazon EC2 を使用して、CodeDeploy デプロイに必要な数だけ仮想サーバー を起動できます。

Amazon EC2 の詳細については、Amazon EC2 入門ガイド を参照してください。

このセクションの説明では、CodeDeploy のデプロイで使用する Amazon EC2 インスタンスを作成 および設定する方法を示します。

トピック

- <u>CodeDeploy (AWS CLI または Amazon EC2 コンソール)</u>用の Amazon EC2 インスタンスを作成す
 <u>る</u>
- CodeDeploy 用の Amazon EC2 インスタンスを作成する (AWS CloudFormation テンプレート)
- <u>CodeDeploy と共に動作するように Amazon EC2 インスタンスを構成します</u>

CodeDeploy (AWS CLI または Amazon EC2 コンソール) 用の Amazon EC2 インスタンスを作成する

これらの説明では、CodeDeploy デプロイ用に設定した新しい Amazon EC2 インスタンスを起動す る方法を示します。

AWS CloudFormation テンプレートを使用して、Amazon Linux または Windows Server を実行して いる Amazon EC2 インスタンスを起動できます。このインスタンスはCodeDeploy デプロイで使用 するように既に設定されています。Ubuntu Server または Red Hat Enterprise Linux (RHEL) を実行 している Amazon EC2 インスタンス用の AWS CloudFormation テンプレートは提供していません。 テンプレートを使用する代わりに、「<u>CodeDeploy のためにインスタンスを用いた操作</u>」を参照して ください。

Amazon EC2 コンソール AWS CLI、または Amazon EC2 APIs を使用して Amazon EC2 インスタン スを起動できます。

Amazon EC2 インスタンス (コンソール) を起動します。

前提条件

まだ設定していない場合は、<u>CodeDeploy の開始方法</u>「」の手順に従って をセットアップおよび設 定 AWS CLI し、IAM インスタンスプロファイルを作成します。

Amazon EC2 インスタンスの起動

- 1. にサインイン AWS Management Console し、「https://<u>https://console.aws.amazon.com/</u> ec2/.com」で Amazon EC2 コンソールを開きます。
- 2. ナビゲーションペインで [インスタンス]、[インスタンスの作成] の順に選択します。
- [Step 1: Choose an Amazon Machine Image (AMI)] ページで、[Quick Start] タブから、使用す るオペレーションシステムおよびバージョンを探して、[Select] を選択します。CodeDeploy に よりサポートされている Amazon EC2 AMI オペレーティングシステムを選択する必要がありま す。詳細については、「<u>CodeDeploy エージェントで対応するオペレーティングシステム</u>」を参 照してください。
- 4. [ステップ 2: インスタンスタイプの選択] ページで、利用可能な Amazon EC2 インタンスタイプ を選択し、[次の手順: インスタンスの詳細の設定] を選択します。
- IAM role 中の Step 3: Configure Instance Details ページ上で、ステップ 4: Amazon EC2 インス タンス用の IAM インスタンスプロファイルを作成する で作成した IAM インスタンスロールを選 択します。提案されたロール名を使用している場合は、[CodeDeployDemo-EC2-Instance-Profile] を選択します。独自のロール名を作成した場合は、その名前を選択します。

Note

デフォルトの仮想プライベートクラウド (VPC) がネットワークの一覧に表示されてい ない場合は、Amazon VPC とサブネットを選択するか、作成する必要があります。 [Create new VPC (新しい VPC の作成)] または [Create new subnet (新しいサブネットの 作成)]、またはその両方を選択します。詳細については、「<u>VPC とサブネット</u>」を参照 してください。

- 6. [次の手順: ストレージの追加]を選択してください。
- 7. [ステップ 4: ストレージの追加] ページは変更せず、[次の手順: タグの追加] を選択します。
- 8. [Step 5: Add Tags] (ステップ 5: タグの追加) ページで[Add Tag] (タグの追加) を選択します。
- 9. [キー] ボックスに「Name」と入力します。[値] ボックスに「CodeDeployDemo」と入力しま す。

Important

[キー] ボックスと [値] ボックスの内容は大文字と小文字が区別されます。

- 10. [Next: Configure Security Group] (次に):セキュリティグループを設定)を選択します。
- 11. [ステップ 6: セキュリティグループの設定] ページで、[新規セキュリティグループを作成] オプ ションを選択したままにします。

デフォルトの SSH ロールは、Amazon Linux、Ubuntu サーバー、または RHEL を実行する Amazon EC2 インスタンスのために設定されます。デフォルトの RDP ロールは、Windows サーバーを実行する Amazon EC2 インスタンスのために設定されます。

HTTP ポートを開く場合は、[ルールの追加] ボタンを選択し、[タイプ] ドロップダウンリストから、[HTTP] を選択します。Custom 0.0.0.0/0 のデフォルトの Source を受け入れ、次に Review and Launch を選択します。

Note

本稼働環境では、Anywhere 0.0.0.0/0 を指定する代わりに、SSH、RDP、および HTTP ポートへのアクセスを制限することを推奨します。CodeDeploy は、無制限のポートア クセスを必要とせず、HTTP アクセスを必要としません。詳細については、「<u>Amazon</u> EC2 インスタンスの保護のヒント」を参照してください。

[汎用 (SSD) から起動する] ダイアログボックスが表示されたら、指示に従って [次へ] を選択し ます。

- 13. [Step 7: Review Instance Launch] ページは変更せず、[Launch] を選択します。
- 14. [既存のキーペアの選択または新しいキーペアの作成] ダイアログボックスで、[既存のキーペア の選択] または [新しいキーペアの作成] を選択します。すでに Amazon EC2 インスタンスキー ペアを設定している場合は、ここで選択できます。

Amazon EC2 インスタンスのキーペアがまだない場合は、[Create a new key pair (新しいキーペ アの作成)] を選択して、わかりやすい名前を付けます。お客様のコンピュータに Amazon EC2 インスタンスキーペアをダウンロードするために、Download Key Pair を選択します。

A Important

SSH または RDP を使用して、Amazon EC2 インスタンスへアクセスしたい場合は、 キーペアが必要です。

- 15. [Launch Instances] (インスタンスを起動) をクリックします。
- 16. Amazon EC2 インスタンスのための ID を選択します。インスタンスが起動され、すべての チェックが成功するまで先に進まないでください。

CodeDeploy エージェントをインストールします

CodeDeploy エージェントは、CodeDeploy デプロイ中で使用する前に、Amazon EC2 インスタンス 上にインストールする必要があります。詳細については、「<u>CodeDeploy エージェントをインストー</u> ルする」を参照してください。

Note

コンソールでデプロイグループを作成するときに、CodeDeploy エージェントの自動インストールと更新を設定できます。

Amazon EC2 インスタンス (CLI) の起動

前提条件

まだ設定していない場合は、<u>CodeDeploy の開始方法</u>「」の手順に従って をセットアップおよび設 定 AWS CLI し、IAM インスタンスプロファイルを作成します。

Amazon EC2 インスタンスの起動

For Windows Server only Windows サーバーを実行している Amazon EC2 インスタンスを作成中の場合、create-security-group と authorize-security-group-ingress のコマンドを呼び出し、実行する インスタンスを作成する場合は、RDP アクセス (デフォルトでは許可されない)、または代わりに HTTP アクセスを許可するセキュリティグループを作成します。例えば、CodeDeployDemo-Windows-Security-Group という名前のセキュリティグループを作成するには、次のコマンドを1つずつ実行します。

aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-Group -description "For launching Windows Server images for use with CodeDeploy"

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 3389
```

aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 80

Note

デモンストレーションのため、これらのコマンドは、ポート 3389 を経由して RDP に、 またはポート 80 を経由して HTTP に無制限アクセスを許可するセキュリティグルー プを作成します。ベストプラクティスとして、SSH および HTTP ポートへのアクセス を制限することをお勧めします。CodeDeploy は、無制限のポートアクセスを必要とせ ず、HTTP アクセスを必要としません。詳細については、「<u>Amazon EC2 インスタンス</u> の保護のヒント」を参照してください。

2. run-instances のコマンドを呼び出して、Amazon EC2 インスタンスを作成して起動します。

このコマンドを呼び出す前に、以下を収集する必要があります。

- インスタンスに使用する Amazon マシンイメージ (AMI) (*ami-id*) の ID。ID を取得するには、「適切な AMI の検索」を参照してください。
- 例えば t1.micro のように、作成する Amazon EC2 インスタンス (*instance-type*) のタイ プ名。リストについては、Amazon EC2 instance types を参照してください。
- リージョンのための CodeDeploy エージェントインスレーションファイルが保存されている Amazon S3 バケットへのアクセス許可を持つ IAM インスタンスプロファイルの名前。

IAM インスタンスプロファイルの作成についての情報は、<u>ステップ 4: Amazon EC2 インスタ</u>ンス用の IAM インスタンスプロファイルを作成する を参照してください。

 Amazon Linux、Ubuntu サーバー、または RHEL または RDP アクセスを実行している Amazon EC2 インスタンスへのSSH、あるいは Windows サーバーを実行している Amazon EC2 へRDP アクセスを可能とする Amazon EC2インスタンスキーの名前 (*key-name*)。

Important

キーペアのファイル拡張子ではなく、キーペア名のみを入力します。例えば、mykeypair.pem ではなく、my-keypair です。

キーペア名を見つけるには、<u>https://console.aws.amazon.com/ec2</u> でAmazon EC2 コンソール を開きます。ナビゲーションペインの [ネットワーク & セキュリティ] の下で、[キーペア] を 選択し、リストのキーペア名をメモします。

新しいキーペアを生成するには、「<u>Amazon EC2 を使用してキーペアを作成する</u>」を参照 してください。「AWS 全般のリファレンス」の「<u>リージョンエンドポイント</u>」にリストさ れているリージョンの一つの中にキーペアを作成することを確かめてください。そうしない と、CodeDeploy を用いて Amazon EC2 インスタンスのキーペアを使用できなくなります。

Amazon Linux、RHEL、および Ubuntu Server 用

Amazon Linux, Ubuntu Server、あるいは RHEL を実行する Amazon EC2 インスタンスを開始 するための run-instances のコマンドを呼び出し、<u>ステップ 4: Amazon EC2 インスタンス用の</u> IAM インスタンスプロファイルを作成する で作成した IAM インスタンスプロファイルをアタッ チする方法。例: 。

```
aws ec2 run-instances \
    --image-id ami-id \
    --key-name key-name \
    --count 1 \
    --instance-type instance-type \
    --iam-instance-profile Name=iam-instance-profile
```

Note

このコマンドは、ポート 22 を経由した SSH の無制限のアクセス、または、ポー ト 80 を経由した HTTP など複数のポートへのアクセスを許可する Amazon EC2 インスタンスのデフォルトのセキュリティグループを作成します。ベストプラク ティスとして、SSH および HTTP ポートへのアクセスを制限することをお勧めしま す。CodeDeploy は、無制限のポートアクセスを必要とせず、HTTP ポートアクセスを 必要としません。詳細については、「<u>Amazon EC2 インスタンスの保護のヒント</u>」を参 照してください。

Windows Server の場合

run-instances のコマンドを呼び出して Windows サーバーを実行する Amazon EC2 インスタン スを起動し、<u>ステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成</u> <u>する</u> で作成した IAM インスタンスプロファイルをアタッチして、ステップ 1 で作成したセキュ リティグループの名前を指定するには、例: 。

aws ec2 run-instances --image-id ami-id --key-name key-name --count 1 --instancetype instance-type --iam-instance-profile Name=iam-instance-profile --securitygroups CodeDeploy-Windows-Security-Group

これらのコマンドは、指定された AMI、キーペア、およびインスタンスタイプ、指定された IAM インスタンスプロファイルを用いて単一の Amazon EC2 インスタンスプロファイルを起動 し、起動時に指定されたスクリプトを実行します。

3. 出力の InstanceID の値を記録します。この値を忘れた場合は、Amazon EC2 インスタンスの キーペアに対する describe-instances のコマンドを呼び出すことで、後で取得できます。

aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
 "Reservations[*].Instances[*].[InstanceId]" --output text

インスタンス ID を使用して create-tags のコマンドを呼び出します。このコマンドで Amazon EC2 インスタンスにタグが付けられ、後でデプロイ中に、CodeDeploy が見つけることができ るようになります。次の例で、タグ名は **CodeDeployDemo** と名付けられていますが、希望する Amazon EC2 インスタンスタグを指定できます。

aws ec2 create-tags --resources instance-id --tags Key=Name,Value=CodeDeployDemo

複数のタグを同時にインスタンスに適用できます。例:。

aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance
Key=Region,Value=West Key=Environment,Value=Beta

CodeDeploy のための Amazon EC2 インスタンスを作成します。

Amazon EC2 インスタンスが起動され、すべてのチェックが成功したことを確認するに は、describe-instance-status のコマンドを呼び出すためのインスタンス ID を使用します。

aws ec2 describe-instance-status --instance-ids instance-id --query
"InstanceStatuses[*].InstanceStatus.[Status]" --output text

インスタンスが起動され、すべてのチェックが成功すると、ok が出力に表示されます。

CodeDeploy エージェントをインストール

CodeDeploy エージェントは、CodeDeploy デプロイ中で使用する前に、Amazon EC2 インスタンス 上にインストールする必要があります。詳細については、「<u>CodeDeploy エージェントをインストー</u> ルする」を参照してください。

Note

コンソールでデプロイグループを作成するときに、CodeDeploy エージェントの自動インス トールと更新を設定できます。

CodeDeploy 用の Amazon EC2 インスタンスを作成する (AWS CloudFormation テンプレート)

AWS CloudFormation テンプレートを使用して、Amazon Linux または Windows Server を実行する Amazon EC2 インスタンスをすばやく起動できます。テンプレートを使用してインスタンスを起動 するには AWS CLI、、CodeDeploy コンソール、または AWS APIs を使用できます。インスタンス を起動することに加えて、テンプレートでは以下を実行します。

- CodeDeploy デプロイに参加するアクセス許可をインスタンスに付与 AWS CloudFormation するようにに指示します。
- インスタンスにタグ付けして、CodeDeploy がデプロイ中に見つけられるようにします。
- インスタンス上で CodeDeploy エージェントをインストールおよび実行します。

AWS CloudFormation を使用して Amazon EC2 インスタンスをセットアップする必要はありません。代替方法については、「<u>CodeDeploy のためにインスタンスを用いた操作</u>」を参照してください。

Ubuntu Server または Red Hat Enterprise Linux (RHEL) を実行している Amazon EC2 インスタンス 用の AWS CloudFormation テンプレートは提供していません。

トピック

- [開始する前に]
- <u>AWS CloudFormation テンプレートを使用して Amazon EC2 インスタンスを起動する (コンソール)</u>
- <u>AWS CloudFormation テンプレートを使用して Amazon EC2 インスタンスを起動する (AWS CLI)</u>

[開始する前に]

AWS CloudFormation テンプレートを使用して Amazon EC2 インスタンスを起動する前に、次の手順を完了してください。

- 1. 「<u>ステップ 1: セットアップ</u>」の説明に従って管理ユーザーを作成したことを確認します。ユー ザーに次の最小限の許可があることをもう一度確認し、存在しないものを追加します。
 - cloudformation:*
 - codedeploy:*
 - ec2:*
 - iam:AddRoleToInstanceProfile
 - iam:CreateInstanceProfile
 - iam:CreateRole
 - iam:DeleteInstanceProfile
 - iam:DeleteRole
 - iam:DeleteRolePolicy
 - iam:GetRole
 - iam:DeleteRolePolicy
 - iam:PutRolePolicy
 - iam:RemoveRoleFromInstanceProfile
- 2. Amazon Linux を実行する Amazon EC2 インスタンスへの SSH アクセス、または Windows Server を実行するインスタンスへの RDP アクセスを有効にするインスタンスのキーペアがある

キーペア名を見つけるには、<u>https://console.aws.amazon.com/ec2</u> でAmazon EC2 コンソールを 開きます。ナビゲーションペインの [ネットワーク & セキュリティ] の下で、[キーペア] を選択 し、リストのキーペア名をメモします。

新しいキーペアを生成するには、「<u>Amazon EC2 を使用してキーペアを作成する</u>」を参照し てください。「AWS 全般のリファレンス」の [<u>リージョンエンドポイント</u>] にリストされて いるリージョンの一つの中に、キーペアが作成されていることを確かめます。それ以外の場 合、CodeDeploy を用いてインスタンスのキーペアを使用できなくなります。

AWS CloudFormation テンプレートを使用して Amazon EC2 インスタンスを起動する (コンソール)

1. にサインイン AWS Management Console し、 AWS CloudFormation コンソールを <u>https://</u> console.aws.amazon.com/cloudformation://www.com で開きます。

▲ Important

で使用したのと同じアカウント AWS Management Console で にサインインしま す<u>CodeDeploy の開始方法</u>。ナビゲーションバーのリージョンセレクターで、「AWS 全 般のリファレンス」の「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれか のリージョンを選択します。CodeDeploy は、これらのリージョンでのみサポートされ ています。

- 2. [Create Stack] (スタックの作成) を選択します。
- [テンプレートの選択] で、[Amazon S3 テンプレート URL の指定] を選択します。ボックスに、 リージョンの AWS CloudFormation テンプレートの場所を入力し、次へを選択します。

リージョン	AWS CloudFormation テンプレートの場所
米国東部 (オハイオ) リージョン	<pre>http://s3-us-east-2.amazona ws.com/aws-codedeploy-us-ea st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
米国東部(バージニア州北部) リージョン	<pre>http://s3.amazonaws.com/aws -codedeploy-us-east-1/templ</pre>

リージョン	AWS CloudFormation テンプレートの場所
	ates/latest/CodeDeploy_Samp leCF_Template.json
US West (N. California) Region	<pre>http://s3-us-west-1.amazona ws.com/aws-codedeploy-us-we st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
米国西部 (オレゴン) リージョン	<pre>http://s3-us-west-2.amazona ws.com/aws-codedeploy-us-we st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
カナダ (中部) リージョン	<pre>http://s3-ca-central-1.amaz onaws.com/aws-codedeploy-ca -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
欧州 (アイルランド) リージョン	<pre>http://s3-eu-west-1.amazona ws.com/aws-codedeploy-eu-we st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
欧州 (ロンドン) リージョン	<pre>http://s3-eu-west-2.amazona ws.com/aws-codedeploy-eu-we st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
欧州(パリ)リージョン	<pre>http://s3-eu-west-3.amazona ws.com/aws-codedeploy-eu-we st-3/templates/latest/CodeD eploy_SampleCF_Template.json</pre>

リージョン	AWS CloudFormation テンプレートの場所
欧州(フランクフルト)リージョン	<pre>http://s3-eu-central-1.amaz onaws.com/aws-codedeploy-eu -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
イスラエル (テルアビブ) リージョン	<pre>http://s3-il-central-1.amaz onaws.com/aws-codedeploy-il -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
アジアパシフィック (香港) リージョン	<pre>http://s3-ap-east-1.amazona ws.com/aws-codedeploy-ap-ea st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
Asia Pacific (Tokyo) Region	<pre>http://s3-ap-northeast-1.am azonaws.com/aws-codedeploy- ap-northeast-1/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
Asia Pacific (Seoul) Region	<pre>http://s3-ap-northeast-2.am azonaws.com/aws-codedeploy- ap-northeast-2/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
アジアパシフィック (シンガポール) リー ジョン	<pre>http://s3-ap-southeast-1.am azonaws.com/aws-codedeploy- ap-southeast-1/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>

リージョン	AWS CloudFormation テンプレートの場所
アジアパシフィック (シドニー) リージョン	<pre>http://s3-ap-southeast-2.am azonaws.com/aws-codedeploy- ap-southeast-2/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
アジアパシフィック (メルボルン) リージョ ン	<pre>https://aws-codedeploy-ap-s outheast-4.s3.ap-southeast- 4.amazonaws.com/templates/l atest/CodeDeploy_SampleCF_T emplate.json</pre>
アジアパシフィック (ムンバイ) リージョン	<pre>http://s3-ap-south-1.amazon aws.com/aws-codedeploy-ap-s outh-1/templates/latest/Cod eDeploy_SampleCF_Template.j son</pre>
南米 (サンパウロ) リージョン	<pre>aws-codedeploy-ap-northeast -1.s3.sa-east-1.amazonaws.c om/templates/latest/CodeDep loy_SampleCF_Template.json</pre>

- 4. [スタック名] ボックスに、スタックの名前 (CodeDeployDemoStack など) を入力します
- 5. [Parameters] に以下を入力し、[Next] を選択します。
 - [InstanceCount] には、起動するインスタンスの数を入力します (デフォルトの1のままにして おくことをお勧めします)。
 - [InstanceType] には、起動するインスタンスタイプを入力します (またはデフォルトの t1.micro のままにします)。
 - [KeyPairName] のために、インスタンスのキーペア名をタイプします。キーペアのファイル拡張子ではなく、キーペア名のみを入力します。
 - OperatingSystem ボックスにのために、Windows をタイプし、Windows サーバーを実行する インスタンスを起動します (または Linux のデフォルトのままにします)。

 [SSHLocation] には、SSH または RDP でインスタンスに接続するのに使用される IP アドレ ス範囲を入力します (またはデフォルトの 0.0.0.0/0 のままにします)。

▲ Important

0.0.0.0/0 のデフォルトは、デモンストレーションのみを目的として提供されてい ます。CodeDeploy では、Amazon EC2 インスタンスのポートへの無制限のアクセス 権限を持つ必要はありません。ベストプラクティスとして、SSH (および HTTP) ポー トへのアクセスを制限することをお勧めします。詳細については、「<u>Amazon EC2 イ</u> <u>ンスタンスの保護のヒント</u>」を参照してください。

- TagKey のために、デプロイ中に CodeDeploy がインスタンスを特定するために使うインスタンスのタグキーをタイプします (または Name のデフォルトのままに)。
- TagValue のために、CodeDeploy がデプロイ中にインスタンスを特定するために使用するインスタンスのタグ値をタイプします (または CodeDeployDemo のデフォルトのままに)。
- 6. [Options] ページで、オプションボックスは空白のまま残し、[Next] を選択します。

▲ Important

AWS CloudFormation タグは CodeDeploy tags とは異なります。 はタグ AWS CloudFormation を使用してインフラストラクチャの管理を簡素化します。CodeDeploy は、タグを使用して Amazon EC2 インスタンスを識別します。お客様は、Specify Parameters ページ上にCodeDeploy タクを指定しました。

7. レビューページの「 機能」で、IAM リソースを作成する AWS CloudFormation 可能性がある 「確認」ボックスを選択し、「作成」を選択します。

AWS CloudFormation がスタックを作成し、Amazon EC2 インスタンスを起動すると、コン ソール AWS CloudFormation で CREATE_COMPLETE がステータス列に表示されます。この処 理には数分かかることもあります。

CodeDeploy エージェントが Amazon EC2 インスタンス上で実行されていることを確認するに は、<u>CodeDeploy エージェントのオペレーションの管理</u> を参照して、<u>CodeDeploy でアプリケーショ</u> <u>ンを作成する</u> に進みます。 AWS CloudFormation テンプレートを使用して Amazon EC2 インスタンスを起動する (AWS CLI)

 create-stack コマンドの呼び出しで AWS CloudFormation テンプレートを使用します。このス タックは、インストールされた CodeDeploy エージェントを用いて新しい Amazon EC2 インス タンスを起動します。

Amazon Linux を実行する Amazon EC2 インスタンスを起動するには:

aws cloudformation create-stack \setminus
stack-name CodeDeployDemoStack \
template-url
parameters ParameterKey=InstanceCount,ParameterValue=1
ParameterKey=InstanceType,ParameterValue=t1.micro \
ParameterKey=KeyPairName,ParameterValue= <i>keyName</i>
ParameterKey=OperatingSystem,ParameterValue=Linux \
ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
ParameterKey=TagKey,ParameterValue=Name 🔪
ParameterKey=TagValue,ParameterValue=CodeDeployDemo 🔪
capabilities CAPABILITY_IAM

Windows Server を実行中の Amazon EC2 インスタンスを起動するには

aws cloudformation create-stack --stack-name CodeDeployDemoStack --templateurl template-url --parameters ParameterKey=InstanceCount,ParameterValue=1 ParameterKey=InstanceType,ParameterValue=t1.micro ParameterKey=KeyPairName,ParameterValue=keyName ParameterKey=OperatingSystem,ParameterValue=Windows ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0 ParameterKey=TagKey,ParameterValue=Name ParameterKey=TagValue,ParameterValue=CodeDeployDemo --capabilities CAPABILITY_IAM

keyName は、インスタンスのキーペア名です。キーペアのファイル拡張子ではなく、キーペア 名のみを入力します。

template-url は、リージョンの AWS CloudFormation テンプレートの場所です。

リージョン	AWS CloudFormation テンプレートの場所
米国東部 (オハイオ) リージョン	<pre>http://s3-us-east-2.amazona ws.com/aws-codedeploy-us-ea st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
米国東部(バージニア州北部) リージョン	<pre>http://s3.amazonaws.com/aws -codedeploy-us-east-1/templ ates/latest/CodeDeploy_Samp leCF_Template.json</pre>
US West (N. California) Region	<pre>http://s3-us-west-1.amazona ws.com/aws-codedeploy-us-we st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
米国西部 (オレゴン) リージョン	<pre>http://s3-us-west-2.amazona ws.com/aws-codedeploy-us-we st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
カナダ (中部) リージョン	<pre>http://s3-ca-central-1.amaz onaws.com/aws-codedeploy-ca -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
欧州 (アイルランド) リージョン	<pre>http://s3-eu-west-1.amazona ws.com/aws-codedeploy-eu-we st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
欧州 (ロンドン) リージョン	<pre>http://s3-eu-west-2.amazona ws.com/aws-codedeploy-eu-we st-2/templates/latest/CodeD eploy_SampleCF_Template.json</pre>

リージョン	AWS CloudFormation テンプレートの場所
欧州(パリ)リージョン	<pre>http://s3-eu-west-3.amazona ws.com/aws-codedeploy-eu-we st-3/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
欧州(フランクフルト)リージョン	<pre>http://s3-eu-central-1.amaz onaws.com/aws-codedeploy-eu -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
イスラエル (テルアビブ) リージョン	<pre>http://s3-il-central-1.amaz onaws.com/aws-codedeploy-il -central-1/templates/latest /CodeDeploy_SampleCF_Templa te.json</pre>
アジアパシフィック (香港) リージョン	<pre>http://s3-ap-east-1.amazona ws.com/aws-codedeploy-ap-ea st-1/templates/latest/CodeD eploy_SampleCF_Template.json</pre>
Asia Pacific (Tokyo) Region	<pre>http://s3-ap-northeast-1.am azonaws.com/aws-codedeploy- ap-northeast-1/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
Asia Pacific (Seoul) Region	<pre>http://s3-ap-northeast-2.am azonaws.com/aws-codedeploy- ap-northeast-2/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>

リージョン	AWS CloudFormation テンプレートの場所
アジアパシフィック (シンガポール) リー ジョン	<pre>http://s3-ap-southeast-1.am azonaws.com/aws-codedeploy- ap-southeast-1/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
アジアパシフィック (シドニー) リージョン	<pre>http://s3-ap-southeast-2.am azonaws.com/aws-codedeploy- ap-southeast-2/templates/la test/CodeDeploy_SampleCF_Te mplate.json</pre>
アジアパシフィック (メルボルン) リージョ ン	<pre>https://aws-codedeploy-ap-s outheast-4.s3.ap-southeast- 4.amazonaws.com/templates/l atest/CodeDeploy_SampleCF_T emplate.json</pre>
アジアパシフィック (ムンバイ) リージョン	<pre>http://s3-ap-south-1.amazon aws.com/aws-codedeploy-ap-s outh-1/templates/latest/Cod eDeploy_SampleCF_Template.j son</pre>
南米 (サンパウロ) リージョン	<pre>aws-codedeploy-ap-northeast -1.s3.sa-east-1.amazonaws.c om/templates/latest/CodeDep loy_SampleCF_Template.json</pre>

このコマンドは**CodeDeployDemoStack**、指定された Amazon S3 バケットの AWS CloudFormation テンプレートを使用して、 という名前の AWS CloudFormation スタックを作成 します。 Amazon S3 Amazon EC2 インスタンスは、[t1.micro]インスタンスタイプに基づいてい ますが、任意のタイプを使用できます。このインスタンスは、**CodeDeployDemo** の値でタグ付 けされていますが、任意の値でタグ付けできます。指定されたインスタンスのキーペアが適用さ れています。

describe-stacks コマンドを呼び出して、という名前の AWS CloudFormation スタッ クCodeDeployDemoStackが正常に作成されたことを確認します。

aws cloudformation describe-stacks --stack-name CodeDeployDemoStack --query
"Stacks[0].StackStatus" --output text

CREATE_COMPLETE の値が返されるまで進まないでください。

CodeDeploy エージェントが Amazon EC2 インスタンス上で実行されていることを確認するに は、<u>CodeDeploy エージェントのオペレーションの管理</u> を参照して、<u>CodeDeploy でアプリケーショ</u> <u>ンを作成する</u> に進みます。

CodeDeploy と共に動作するように Amazon EC2 インスタンスを構成しま す

以下の手順では、Amazon Linux、Ubuntu サーバー、Red Hat エンタープライズ Linux (RHEL)、ま たは Windows サーバーを実行して、CodeDeploy デプロイで使用する Amazon EC2 インスタンスを 構成する方法について説明します。

Note

Amazon EC2 インスタンスがない場合は、 AWS CloudFormation テンプレートを使用して、Amazon Linux または Windows Server を実行しているインスタンスを起動できます。Ubuntu Server または RHEL 用のテンプレートは提供されていません。

ステップ 1: IAM インスタンスプロファイルが Amazon EC2 インスタンスにアタッチ されていることを確認する

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- ナビゲーションペインの [Instances] (インスタンス) で、[Instances] (インスタンス) を選択します。
- 3. 一覧で Amazon EC2 インスタンスを参照して選択します。
- 4. 詳細ペインの、[説明] タブで [IAM ロール] フィールドの値を書き留め、次のセクションに進みま す。

フィールドが空の場合は、IAM インスタンスプロファイルをインスタンスにアタッチすること ができます。詳細については、<u>IAM ロールをインスタンスにアタッチする</u> を参照してくださ い。

ステップ 2: 添付されたインスタンスプロファイルが正しいアクセス権限を持っている ことを確認します。

- 1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
- 2. ナビゲーションペインで [Roles] (ロール) を選択します。
- 3. 前のセクションのステップ 4 で書き留めた IAM ロールの名前を参照し、選択します。

Note

「」の手順に従って作成したロールではなく、AWS CloudFormation テンプレートに よって生成されたサービスロールを使用する場合は<u>ステップ 2: CodeDeployのサービス</u> <u>のロールを作成する</u>、次の点に注意してください。 AWS CloudFormation テンプレートの一部のバージョンでは、生成され、Amazon EC2 インスタンスにアタッチされた IAM インスタンスプロファイルの表示名は、IAM コ ンソールの表示名と同じではありません。例えば、IAM インスタンスプロファイル は、CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX の表示名を持っているかもしれませんが、一方で IAM コンソール中の インスタンスプロファイルは CodeDeploySampleStack-expnyi6-InstanceRole C5P33V1L64EX の表示名である可能性があります。 IAM コンソール中のインスタンスプロファイルを特定するに は、CodeDeploySampleStack-expnyi6-InstanceRole のプレフィックスがどち らでも同じことを確認します。これらの表示名が異なる理由に関する詳細については、 「<u>インスタンスプロファイル</u>」を参照してください。

[Trust Relationships] タブを選択します。The identity provider(s) ec2.amazonaws.com を読み取る [信頼されたエンティティ] にエンティティがない場合、この Amazon EC2 インスタンスを使用できません。CodeDeploy のためにインスタンスを用いた操作の情報を使用して、Amazon EC2 インスタンスを停止して作成します。

The identity provider(s) ec2.amazonaws.com を読み取るエントリがあり、GitHub リポジトリの みにアプリケーションを保存する場合は、「<u>ステップ 3: Amazon EC2 インスタンスへのタグ付</u> <u>け</u>」に進みます。 The identity provider(s) ec2.amazonaws.com を読み取るエントリがある場合、およびバケット にアプリケーションを Amazon S3 に保存する場合は、Permissions タブを選択します。

- 5. [Permissions policies (アクセス権限ポリシー)] エリアにポリシーがある場合は、ポリシー名を選 択してから [Edit policy (ポリシーの編集)] を選択します。
- [JSON] タブを選択します。Amazon S3 バケット中にアプリケーションを保存している場合 は、"s3:Get*"および "s3:List*" が指定したアクションのリストにあることを確認しま す。

次のように表示されます。

```
{"Statement":[{"Resource":"*","Action":[
    ... Some actions may already be listed here ...
    "s3:Get*","s3:List*"
    ... Some more actions may already be listed here ...
],"Effect":"Allow"}]
```

または、次のように表示されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
            ... Some actions may already be listed here ...
            "s3:Get*",
            "s3:List*"
            ... Some more actions may already be listed here ...
        ],
        ...
        }
    ]
}
```

If "s3:Get*" および "s3:List*" が指定されたアクションのリストにない場合、[Edit] を選択 して、それらを追加し、[Save] を選択します。("s3:Get*" または "s3:List*" のどちらかが リストの最後のアクションである場合、必ずアクションの後にコンマを追加して、ポリシード キュメントが検証するようにします。)

Note

このポリシーを、Amazon EC2 インスタンスがアクセスする必要のある Amazon S3 バ ケットにのみ制限することをお勧めします。CodeDeploy エージェントを含む Amazon S3 バケットへのアクセスを必ず許可してください。そうしない場合、CodeDeploy エー ジェントがインスタンス上にインストールされる、または更新されるときに、エラーが 発生する可能性があります。Amazon S3 中の CodeDeploy リソースキットバケットのみ への IAM インスタンスアクセスは許すが、アクセスを防止するバケットのための行を削 除するには

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
```



ステップ 3: Amazon EC2 インスタンスへのタグ付け

Amazon EC2 インスタンスにタグをつけて、CodeDeploy がデプロイ中に見つけられるようにする方 法に関する詳細について、「<u>コンソールでタグを使用する</u>」を参照してから、このページに戻りま す。

Note

任意のキーおよび値を使用して Amazon EC2 インスタンスにタグを付けられます。インスタ ンスにデプロイするとき、必ずこのキーと値を指定してください。

ステップ 4: Amazon EC2 インスタンスに AWS CodeDeploy エージェントをインス トールする

Amazon EC2 インスタンス上に CodeDeploy エージェントをインストールし、それが実行している ことを検証する方法の説明については、<u>CodeDeploy エージェントのオペレーションの管理</u>を参照 してから CodeDeploy でアプリケーションを作成する に進みます。

CodeDeploy 用のオンプレミスインスタンスを用いての作業

オンプレミスインスタンスは、CodeDeploy エージェントを実行してパブリック AWS サービスエン ドポイントに接続できる Amazon EC2 インスタンスではない物理デバイスです。

CodeDeploy アプリケーションリビジョンをオンプレミスインスタンスにデプロイするには、2 つの 主なステップがあります。

- ステップ1-オンプレミスインスタンスを設定して CodeDeploy を用いて登録し、タグをつけます。
- ステップ2 アプリケーションリビジョンをオンプレミスインスタンスにデプロイします。

Note

サンプルアプリケーションリビジョンの作成と、正しく設定および登録されたオンプレ ミスインスタンスへのデプロイを試す場合は、「<u>チュートリアル: CodeDeploy (Windows</u> サーバー、Ubuntu サーバー、または Red Hat エンタープライズ Linux) を使用してオンプ レミスインスタンスにアプリケーションをデプロイします。」を参照してください。オン プレミスインスタンス、および CodeDeploy の使用方法の詳細については、<u>Working with</u> On-Premises Instances を参照してください。

オンプレミスインスタンスをデプロイでそれ以上使用したくない場合は、デプロイグループからオン プレミスインスタンスタグを削除できます。より強力な方法としては、インスタンスからオンプレミ スインスタンスタグを削除します。明示的にオンプレミスインスタンスを登録解除し、デプロイでそ れ以上使用されないようにすることもできます。詳細については、「<u>CodeDeploy でのオンプレミス</u> インスタンスのオペレーションの管理」を参照してください。

このセクションの手順では、デプロイで使用できるようにオンプレミスインスタンスを設定 し、CodeDeploy を用いて登録し、タグ付けする方法を学びます。また、このセクションでは、デプ ロイする計画がなくなった後に、CodeDeploy を使用して、オンプレミスインスタンスについての情 報を取得する情報や、オンプレミスインスタンスの登録を解除する方法を説明します。

トピック

- オンプレミスインスタンスを設定するための前提条件
- CodeDeploy を用いてへのオンプレミスインスタンスを登録
- CodeDeploy でのオンプレミスインスタンスのオペレーションの管理

オンプレミスインスタンスを設定するための前提条件

オンプレミスインスタンスを登録するには、次の前提条件を満たす必要があります。
▲ Important

<u>register-on-premises-instance</u> コマンドを使用していて、 AWS Security Token Service (AWS STS)で生成された一時的な認証情報を定期的に更新する場合は、他の前提条件があ ります。詳細については、IAM セッション ARN 登録前提条件 を参照してください。

デバイスの要件

CodeDeploy を用いてオンプレミスインスタンスとして準備し、登録し、タグ付けするデバイスは、 サポートされている OS を実行している必要があります。リストについては、「<u>CodeDeploy エー</u> ジェントで対応するオペレーティングシステム」を参照してください。

OS がサポートされていない場合、ニーズに適合するために CodeDeploy エージェントをオープン ソースとして利用できます。さらなる詳細については、GitHub の <u>CodeDeploy agent</u> リポジトリを 参照してください。

アウトバウンド通信

オンプレミスインスタンスは、CodeDeploy と通信するためにパブリック AWS サービスエンドポイ ントに接続できる必要があります。

CodeDeploy エージェントは、ポート 443 経由で HTTPS を使用してアウトバウンドの通信をします。

管理コントロール

オンプレミスインスタンスの設定のためにオンプレミスインスタンス上で使用するローカルまた はネットワークのアカウントは、sudo あるいは root (Ubuntu サーバーの場合)、または管理者 (Windows サーバーの場合) として実行できる必要があります。

IAM アクセス許可

オンプレミスインスタンスを登録するために使用する IAM アイデンティティは、登録を完了するため(および必要に応じて登録を削除するため)の許可を付与されている必要があります。

「<u>ステップ 3: CodeDeploy ユーザーのアクセス許可を制限する</u>」で説明されているポリシーに加え て、呼び出し元の IAM アイデンティティに以下の追加のポリシーが添付済みであることを確認しま す。

{

オンプレミスインスタンスを設定するための前提条件

```
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM ポリシーをアタッチする方法については、IAM ポリシーの管理を参照してください。

CodeDeploy を用いてへのオンプレミスインスタンスを登録

オンプレミスインスタンスを登録するには、リクエストを認証するために IAM ID を使用する必要が あります。使用する IAM ID と登録方法を以下のオプションから選択できます。

- ・ リクエストを認証するため、IAM ロール ARN を使用します。
 - register-on-premises-instance コマンドを使用し、AWS Security Token Service (AWS STS)で 生成された一時的な認証情報を定期的に更新して、ほとんどの登録オプションを手動で設定し ます。認証はタイムアウトする一時的なトークンを使用して行われ、定期的に更新する必要が あるため、このオプションは最高レベルのセキュリティを提供します。このオプションは、ど のような規模の実稼働向けデプロイにも推奨されます。詳細については、[register-on-premisesinstance] コマンド (IAM セッション ARN)を使用してオンプレミスインスタンスを登録 を参照 してください。
- (非推奨) リクエストを認証するため、IAM ユーザー ARN を使用します。
 - 最も自動化された登録プロセスのために、register コマンドを使用します。このオプションは、 セキュリティがそれほど問題にならない実稼働以外のデプロイでのみ使用してください。この オプションは認証に静的(永続的)認証情報を使用するため、安全性が低くなります。このオ プションは、単一のオンプレミスインスタンスを登録する場合に適しています。詳細について

- は、<u>オンプレミスインスタンスを登録するために登録コマンド (IAM ユーザー ARN) を使用</u> を参 照してください。
- register-on-premises-instance コマンドを使って、ほとんどの登録オプションを手動で設定します。少数のオンプレミスインスタンスを登録するのに適しています。詳細については、[register-on-premises-instance] コマンド (IAM ユーザー ARN) を使用してオンプレミスインスタンスを登録します。を参照してください。

トピック

- [register-on-premises-instance] コマンド (IAM セッション ARN) を使用してオンプレミスインスタンスを登録
- オンプレミスインスタンスを登録するために登録コマンド (IAM ユーザー ARN) を使用
- [register-on-premises-instance] コマンド (IAM ユーザー ARN) を使用してオンプレミスインスタン スを登録します。

[register-on-premises-instance] コマンド (IAM セッション ARN) を使用してオンプレ ミスインスタンスを登録

オンプレミスインスタンスの認証と登録を最大限制御するには、<u>register-on-premises-instance</u> コマ ンドと AWS Security Token Service 、 () で生成された一時的な認証情報を定期的に更新しますAWS STS。インスタンスの静的 IAM ロールは、CodeDeploy デプロイオペレーションを実行するために これらの更新された AWS STS 認証情報のロールを引き受けます。

多数のインスタンスを登録する必要がある場合は、このメソッドが最も役に立ちます。これによ り、CodeDeploy での登録処理を自動化できます。自分の ID と認証システムを使用して、オンプレ ミスインスタンスを認証でき、CodeDeploy で使用するためにインスタンス IAM セッション認証情 報をサービスからインスタンスへ配布できます。

In Note

または、すべてのオンプレミスインスタンスに分散された共有 IAM ユーザーを使用して AWS STS <u>AssumeRole</u> API を呼び出して、オンプレミスインスタンスのセッション認証情 報を取得することもできます。このメソッドは安全性が低いため、本番稼働用またはミッ ションクリティカルな環境での使用は推奨しません。 以下のトピックの情報を使用して、 で生成された一時的なセキュリティ認証情報を使用してオンプ レミスインスタンスを設定します AWS STS。

トピック

- IAM セッション ARN 登録前提条件
- ステップ 1: オンプレミスインスタンスが引き受ける IAM ロールを作成
- ステップ 2: を使用して個々のインスタンスの一時的な認証情報を生成する AWS STS
- ステップ 3: オンプレミスインスタンスに設定ファイルを追加
- ステップ 4:CodeDeploy デプロイのためオンプレミスインスタンスを準備します。
- ステップ 5: CodeDeploy でオンプレミスインスタンスを登録します。
- ステップ 6: オンプレミスインスタンスにタグ付け
- ステップ 7: アプリケーションリビジョンをオンプレミスインスタンスにデプロイ
- ステップ 8: オンプレミスインスタンスへのデプロイを追跡

IAM セッション ARN 登録前提条件

<u>オンプレミスインスタンスを設定するための前提条件</u> にリストされている前提条件に加えて、次の 追加の要件を満たす必要があります。

IAM アクセス許可

オンプレミスインスタンスを登録するのに使用する IAM 識別子には、CodeDeploy オペレーション を実行するためのアクセス権限が付与される必要があります。AWSCodeDeployFullAccess 管理ポリ シーが IAM ID に添付されていることを確認します。詳細については、 <u>IAM ユーザーガイド</u>の AWS マネージドポリシー を参照してください。

一時的な認証情報を更新するシステム

IAM セッション ARN を使用してオンプレミスインスタンスを登録する場合、一時的な認証情報を定期的に更新する適切なシステムが必要です。一時的な認証情報は 1 時間後、または認証情報が生成 されたときより短い時間が指定されていればそれより早く期限切れになります。認証情報を更新する ためのメソッドは 2 つあります。

 メソッド 1: 企業ネットワーク内で ID および認証システムを適切に使用し、CRON スクリプトを 使って ID および認証システムを定期的にポーリングし、最新のセッション認証情報をインスタ ンスヘコピーするようにします。これにより、組織で使用する認証タイプをサポートするために CodeDeploy エージェントまたはサービスを変更 AWS することなく、認証とアイデンティティ構造を と統合できます。

方法 2: インスタンスで定期的に CRON ジョブを実行して AWS STS AssumeRole アクションを呼び出し、CodeDeploy エージェントがアクセスできるファイルにセッション認証情報を書き込みます。このメソッドでは、IAM ユーザーの使用、およびオンプレミスインスタンスへの認証情報のコピーをする必要はありますが、多くのオンプレミスインスタンスで同じ IAM ユーザーおよび認証情報を使用できます。

Note

メソッド1と2のどちらを使用しているかにかかわらず、一時的なセッション認証情報が更 新された後に CodeDeploy エージェントを再起動するプロセスを設定して、新しい認証情報 が有効になるようにする必要があります。

AWS STS 認証情報の作成と操作の詳細については、<u>AWS Security Token Service 「 API リファレ</u> <u>ンス</u>」および<u>「一時的なセキュリティ認証情報を使用して AWS リソースへのアクセスをリクエス</u> <u>ト</u>する」を参照してください。

ステップ 1: オンプレミスインスタンスが引き受ける IAM ロールを作成

AWS CLI または IAM コンソールを使用して、オンプレミスインスタンスが CodeDeploy を認証およ び操作するために使用する IAM ロールを作成できます。

単一の IAM ロールのみを作成する必要があります。各オンプレミスインスタンスは、このロールに 付与されたアクセス権限を提供する一時認証情報を取得するためにこのロールを引き受けることがで きます。

作成するロールは、CodeDeploy エージェントをインストールするのに必要なファイルにアクセスす るために、次の権限が必要です。

}

```
"Effect": "Allow",
"Resource": "*"
}
]
```

このポリシーを、オンプレミスインスタンスがアクセスする必要のある Amazon S3 バケットにの み制限することをお勧めします。このポリシーを制限する場合、CodeDeploy エージェントを含む Amazon S3 バケットへのアクセスを許可することを確認します。そうしない場合、CodeDeploy エージェントがオンプレミスインスタンスにインストールされる、または更新されるたびに、エ ラーが発生する可能性があります。Amazon S3 リソースへのアクセスコントロールの詳細について は、Managing access permissions to your Amazon S3 resources を参照してください。

IAM ロールを作成するには

 <u>オプションを使用して</u> create-role--role-name コマンドを呼び出し、IAM ロールの名前 (例: CodeDeployInstanceRole) と --assume-role-policy-document オプションを指定して アクセス権限を提供します。

このインスタンスの IAM ロールを作成するときは、CodeDeployInstanceRole という名前を 付け、CodeDeployRolePolicy.json という名前のファイルに必要なアクセス権限を含めま す。

aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policydocument file://CodeDeployRolePolicy.json

2. create-role コマンドを呼び出した出力で、ARN フィールドの値をメモします。例:

arn:aws:iam::123456789012:role/CodeDeployInstanceRole

AWS STS <u>AssumeRole</u> API を使用して各インスタンスの短期認証情報を生成する場合は、ロー ル ARN が必要です。

IAM ロールの作成の詳細については、IAM ユーザーガイドの<u>「AWS サービスにアクセス許可を</u> 委任するロールの作成」を参照してください。

既存のロールにアクセス権限を割り当てる方法については、<u>AWS CLI Command Reference</u>の put-role-policy を参照してください。 ステップ 2: を使用して個々のインスタンスの一時的な認証情報を生成する AWS STS

オンプレミスインスタンスの登録に使用する一時認証情報を生成する前に、一時認証情報を生成する IAM ID (ユーザーまたはロール) を作成または選択する必要があります。sts:AssumeRole アクセス 権限は、この IAM ID のポリシーの設定に含める必要があります。

IAM ID にアクセスsts:AssumeRole許可を付与する方法については、「<u>AWS サービスにアクセス</u> 許可を委任するロールの作成」およびAssumeRole」を参照してください。

一時認証情報を生成するには、2とおりの方法があります。

• AWS CLIを用いて assume-role コマンドを使用します。例:

aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --rolesession-name session-name

コードの説明は以下のとおりです。

- ・12345ACCOUNT が組織の 12 桁のアカウント番号です。
- role-arn は、ステップ 1: オンプレミスインスタンスが引き受ける IAM ロールを作成 で生成 した引き受けるロールの ARN です。
- session-name は、現在作成しているロールセッションへ付ける名前です。

Note

ID と認証システムを定期的にポーリングし、最新のセッション認証情報をインスタンスに コピーする CRON スクリプトを使用する場合(「」で説明されている一時的な認証情報を 更新するための方法 1<u>IAM セッション ARN 登録前提条件</u>)、代わりにサポートされてい る AWS SDK を使用して AssumeRole を呼び出すことができます。

が提供するツールを使用します AWS。

aws-codedeploy-session-helper ツールは AWS STS 認証情報を生成し、インスタンスに配置する ファイルに書き込みます。このツールは、<u>IAM セッション ARN 登録前提条件</u>で説明している一時 認証情報を更新するメソッド 2 に最適です。このメソッドでは、aws-codedeploy-session-helper ツールは、各インスタンスに配置され、IAM ユーザーのアクセス権限を使用してコマンドを実行 します。各インスタンスは、このツールとともに同じ IAM ユーザーの認証情報を使用します。

詳細については、aws-codedeploy-session-helper GitHub リポジトリを参照してください。

Note

IAM セッション認証情報を作成した後、オンプレミスインスタンスの任意の場所に保存し ます。次のステップで、CodeDeploy エージェントがこの場所の認証情報にアクセスでき るように設定します。

続ける前に、定期的に一時認証情報を更新するために使用するシステムを確認します。一時認証情 報が更新されていない場合、オンプレミスインスタンスへのデプロイは失敗します。詳細について は、<u>IAM セッション ARN 登録前提条件</u>にある「一時認証情報を更新するシステム」を参照してくだ さい。

ステップ 3: オンプレミスインスタンスに設定ファイルを追加

ルートまたは管理者権限を使用して、オンプレミスインスタンスに設定ファイルを追加します。この 設定ファイルは、IAM の認証情報、および CodeDeploy のために使われる AWS のリージョンを宣言 するために使用されます。ファイルは、オンプレミスインスタンスの指定の場所に追加する必要があ ります。ファイルには、IAM 一時セッション ARN、そのシークレットキー ID とシークレットアクセ スキー、およびターゲット AWS リージョンが含まれている必要があります。

設定ファイルを追加するには

- オンプレミスインスタンスの以下の場所に、codedeploy.onpremises.yml (Ubuntu サー バーまたは RHEL オンプレミスインスタンスの場合)、または、conf.onpremises.yml (Windows サーバーオンプレミスインスタンスの場合) という名前のファイルを作成します。
 - Ubuntu サーバーの場合:/etc/codedeploy-agent/conf
 - Windows サーバーについて: C:\ProgramData\Amazon\CodeDeploy
- テキストエディタを使用して、新しく作成した codedeploy.onpremises.yml ファイル (Linux) または conf.onpremises.yml ファイル (Windows) に次の情報を追加します。

```
iam_session_arn: iam-session-arn
aws_credentials_file: credentials-file
region: supported-region
```

コードの説明は以下のとおりです。

- *iam-session-arn*は、「<u>ステップ 2</u>: を使用して個々のインスタンスの一時的な認証情報を 生成する AWS STS」でメモしておいた IAM セッション ARN です。
- credentials-fileは、ステップ 2: を使用して個々のインスタンスの一時的な認証情報を 生成する AWS STS でメモした一時セッション ARN の認証情報ファイルの場所です。
- supported-regionは、「AWS 全般のリファレンス」の「<u>リージョンエンドポイント</u>」に 一覧表示されているように、CodeDeploy がサポートするリージョンの1つです。

ステップ 4:CodeDeploy デプロイのためオンプレミスインスタンスを準備します。

のインストールと設定 AWS CLI

オンプレミスインスタンス AWS CLI に をインストールして設定します。(AWS CLI は、CodeDeploy エージェントをダウンロードしてオンプレミスインスタンスにインストールするた めに使用されます)。

1. オンプレミスインスタンス AWS CLI に をインストールするには、AWS Command Line Interface 「 ユーザーガイド」の「 のセットアップ AWS CLI」の手順に従います。

Note

オンプレミスインスタンスを使用するための CodeDeploy コマンドは、 AWS CLIのバー ジョン 1.7.19 で使用できるようになりました。のバージョンが AWS CLI 既にインストー ルされている場合は、 を呼び出してそのバージョンを確認できますaws --version。

2. オンプレミスインスタンス AWS CLI で を設定するには、AWS Command Line Interface 「 ユー ザーガイド」の「 の設定 AWS CLI」の手順に従います。

▲ Important

(aws configure コマンドを呼び出す AWS CLI などして) を設定するときは、少なくとも で説明されているアクセス許可を持つ IAM ユーザーのシークレットキー ID とシークレッ トアクセスキーを必ず指定してくださいIAM セッション ARN 登録前提条件。

AWS_REGION 環境変数を設定する (Ubuntu Server および RHEL のみ)

Ubuntu サーバーまたは RHEL をオンプレミスインスタンスで実行していない場合は、このステップ をスキップして「CodeDeploy エージェントをインストールする」へ進んでください。 Ubuntu サーバーまたは RHEL オンプレミスインスタンスに CodeDeploy エージェントをインストー ルし、新しいバージョンが使用可能になったらいつでも CodeDeploy エージェントを更新するよ うにインスタンスを有効にします。これを行うには、インスタンス上の AWS_REGION の環境変数 を、CodeDeploy がサポートしているリージョンのうちの 1 つの識別子に設定します。CodeDeploy アプリケーション、デプロイグループ、およびアプリケーションリビジョンのある (us-west-2 な ど)リージョンの値に設定することをお勧めします。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「<u>リージョンエンドポイント</u>」を参照してください。

環境変数を設定するには、端末から以下を呼び出します。

export AWS_REGION=supported-region

supported-region がリージョンの識別子である場所 (例:us-west-2)。

CodeDeploy エージェントをインストール

- Ubuntu サーバーオンプレミスインスタンスの場合は、Ubuntu サーバー用の CodeDeploy エージェントをインストールするの手順に従った後、このページに戻ります。
- RHEL オンプレミスインスタンスについては、<u>Amazon Linux または RHEL 用の CodeDeploy エー</u> ジェントをインストールするの手順に従った後、このページに戻ります。
- Windows サーバーオンプレミスインスタンスの場合は、Windows サーバー用の CodeDeploy エージェントです。の手順に従った後、このページに戻ります。

ステップ 5: CodeDeploy でオンプレミスインスタンスを登録します。

このステップの手順では、オンプレミスインスタンス自体からオンプレミスインスタンスを登録して いることを想定します。オンプレミスインスタンスは、 AWS CLI がインストールされ、設定された 別のデバイスまたはインスタンスから登録できます。

AWS CLI を使用してオンプレミスインスタンスを CodeDeploy に登録し、デプロイで使用できるようにします。

を使用する前に AWS CLI、 で作成した一時セッション認証情報の ARN が必要です<u>ステップ 3: オン</u> <u>プレミスインスタンスに設定ファイルを追加</u>。例えば、AssetTag12010298EX と指定したインス タンスの場合:

arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX

register-on-premises-instance コマンドを呼び出し、以下を指定します。

• オンプレミスインスタンスを一意に識別する名前 (--instance-name オプションで指定)。

A Important

オンプレミスインスタンスを識別するために、特にデバッグのため、オンプレミスイン スタンスの一意な特徴を示す名前 (例えば、もしあれば、STS 認証情報の session-name とシリアルナンバー、または内部アセット識別子など)を指定することを強くお勧めし ます。名前として MAC アドレスを指定した場合、MAC アドレスにはコロン (:) など CodeDeploy が許可しない文字が含まれることに注意してください。許可された文字の一 覧については、「CodeDeploy のクォータ」を参照してください。

 ・ 複数のオンプレミスインスタンスを認証するために <u>ステップ 1: オンプレミスインスタンスが引き</u>
 受ける IAM ロールを作成 で設定した IAM セッション ARN。

例:

aws deploy register-on-premises-instance --instance-name *name-of-instance* --iamsession-arn arn:aws:sts::account-id:assumed-role/role-to-assume/session-name

コードの説明は以下のとおりです。

- name-of-instance は、オンプレミスインスタンスを識別するのに使用する名前です (例:AssetTag12010298EX)。
- account-id は、組織の 12 桁のアカウント IDです (例: 111222333444)。
- *role-to-assume*は、インスタンス用に作成した IAM ロールの名前です (例: CodeDeployInstanceRole)。
- session-name は、ステップ 2: を使用して個々のインスタンスの一時的な認証情報を生成する AWS STS で指定したセッションロールの名前です。

ステップ 6: オンプレミスインスタンスにタグ付け

AWS CLI または CodeDeploy コンソールを使用して、オンプレミスインスタンスにタグを付ける ことができます。(CodeDeploy はオンプレミスインスタンスタグを使用してデプロイ中にデプロイ ターゲットを識別します。)

オンプレミスインスタンスにタグ付けするには (CLI)

• add-tags-to-on-premises-instances コマンドを呼び出し、以下を指定します。

- オンプレミスインスタンスを一意に識別する名前 (--instance-names オプションで指定)。
- 使用するオンプレミスインスタンスのタグキーの名前とタグ値 (--tags オプションで指定)。
 名前と値はいずれも指定する必要があります。CodeDeploy は値のみがあるオンプレミスインスタンスタグを許可しません。

例:

aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem

オンプレミスインスタンスにタグ付けするには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、Deploy を展開し、On-premises instances を選択します。
- 3. オンプレミスインスタンスのリストで、タグ付けするオンプレミスインスタンスの名前を選択し ます。
- タグのリストで、目的のタグキーとタグ値を選択または入力します。タグキーとタグ値を入力するたびに、別の行が表示されます。最大 10 個のタグにこれを繰り返すことができます。タグを 削除するには[削除]を選択してください。
- 5. タグを追加したら、[Update Tags] を選択します。

ステップ 7: アプリケーションリビジョンをオンプレミスインスタンスにデプロイ

登録され、タグ付けされたオンプレミスインスタンスにアプリケーションリビジョンをデプロイする 準備ができました。

Amazon EC2 インスタンスにアプリケーションリビジョンをデプロイするのと同様の方法でオ ンプレミスインスタンスにアプリケーションリビジョンをデプロイします。手順については、 「<u>CodeDeploy でデプロイを作成する</u>」を参照してください。これらの指示には、アプリケーション の作成、開発グループの作成、およびアプリケーションリビジョンの準備を含む前提条件へのリン クが含まれています。シンプルなサンプルアプリケーションリビジョンをデプロイすることが必要な 場合は、<u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エン</u> <u>タープライズ Linux) を使用してオンプレミスインスタンスにアプリケーションをデプロイします。</u> の <u>ステップ 2: サンプルのアプリケーションリビジョンを作成する</u> で説明してあるものを作成できま す。

▲ Important

オンプレミスインスタンスを対象としたデプロイグループの作成の一部とし て、CodeDeploy サービスロールを再利用する場合は、Tag:get* をサービスロールのポリ シーステートメントの Action の部分に含める必要があります。詳細については、「<u>ステッ</u> プ 2: CodeDeployのサービスのロールを作成する」を参照してください。

ステップ 8: オンプレミスインスタンスへのデプロイを追跡

登録されタグ付けされたオンプレミスインスタンスへアプリケーションリビジョンをデプロイした 後、デプロイの進行状況を追跡できます。

Amazon EC2 インスタンスへのデプロイの追跡と同様の方法でオンプレミスインスタンスへのデプ ロイの追跡をします。手順については、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してく ださい。

オンプレミスインスタンスを登録するために登録コマンド (IAM ユーザー ARN) を使 用

▲ Important

IAM ユーザーを使用してインスタンスを登録することは、認証に静的 (永続的) 認証情報を使 用するため推奨されません。セキュリティ向上のため、認証には一時的な認証情報を使用し てインスタンスを登録することをお勧めします。詳細については、「<u>[register-on-premises-</u> <u>instance] コマンド (IAM セッション ARN) を使用してオンプレミスインスタンスを登録</u>」を 参照してください。 ▲ Important

IAM ユーザーのアクセスキー (永続的認証情報) をローテーションする計画を立ててくださ い。アクセスキーのローテーションの詳細については、「<u>アクセスキーの更新</u>」を参照して ください。

このセクションでは、最低限の労力でオンプレミスインスタンスを CodeDeploy で設定してタグ付 けする方法を説明します。register コマンドは、単一の、または少数のオンプレミスインスタンスを 処理する際に最も有用です。register のコマンドは、インスタンスを認証するために IAM ユーザー ARN を使用している場合のみ使用できます。register のコマンドは、認証のための IAM セッション ARN とは共に使用することはできません。

register のコマンドを使用すると、CodeDeploy で次の操作ができます。

- コマンドで IAM ユーザーを指定しない場合は、オンプレミスインスタンス AWS Identity and Access Management の に IAM ユーザーを作成します。
- ・オンプレミスインスタンスの設定ファイルに IAM ユーザーの認証情報を保存します。
- CodeDeploy でオンプレミスインスタンスを登録します。
- コマンドの一部にタグを指定した場合、オンプレミスインスタンスにタグを追加します。
 - Note

register-on-premises-instance コマンドは、register コマンドの代わりです。CodeDeploy を 用いてオンプレミスインスタンスの設定、登録、タグ付けのほとんどを自分で行いたい場 合は、register-on-premises-instance のコマンドを使用します。また、register-on-premisesinstance のコマンドを使うと、IAM ユーザー ARN の代わりに、IAM セッション ARN を使 用してインスタンスを登録できます。このアプローチは、大量のオンプレミスインスタンス がある場合、大きな利点となります。具体的には、各オンプレミスインスタンスに 1 つず つ IAM ユーザーを作成するかわりに、単一の IAM セッション ARN を使用して複数のイン スタンスを認証できます。詳細については、[register-on-premises-instance] コマンド (IAM ユーザー ARN) を使用してオンプレミスインスタンスを登録します。 および[register-onpremises-instance] コマンド (IAM セッション ARN) を使用してオンプレミスインスタンスを 登録を参照してください。

トピック

- ステップ 1: オンプレミスインスタンス AWS CLI に をインストールして設定する
- ステップ 2: 登録コマンドを呼び出す
- ステップ 3: インストールコマンドを呼び出す
- ステップ 4: デプロイアプリケーションリビジョンをオンプレミスインスタンスにデプロイする
- ステップ 5: オンプレミスインスタンスへのデプロイを追跡

ステップ 1: オンプレミスインスタンス AWS CLI に をインストールして設定する

1. オンプレミスインスタンス AWS CLI に をインストールします。 AWS CLIユーザーガイド の Getting set up with the AWS Command Line Interface の指示に従います。

Note

オンプレミスインスタンスを操作するための CodeDeploy コマンドは、 AWS CLI バー ジョン 1.7.19 以降で使用できます。 AWS CLI が既にインストールされている場合は、 を呼び出しaws --versionてバージョンを確認します。

2. オンプレミスインスタンス AWS CLI で を設定します。 AWS CLIユーザーガイド の <u>Configuring</u> the AWS Command Line Interface の指示に従います。

▲ Important

を設定するときは AWS CLI (aws configure コマンドを呼び出すなど)、 で指定された アクセス許可に加えて、少なくとも以下の AWS アクセス許可を持つ IAM ユーザーの シークレットキー ID とシークレットアクセスキーを必ず指定してください<u>オンプレミ</u> <u>スインスタンスを設定するための前提条件</u>。これにより、オンプレミスインスタンスで CodeDeploy エージェントをダウンロードしてインストールすることができるようにな ります。アクセス権限は次のようになります。

```
{
    "Version": "2012-10-17",
    "Statement" : [
        {
         "Effect" : "Allow",
         "Action" : [
         "codedeploy:*",
         "iam:CreateAccessKey",
         "iam:CreateUser",
         "Iam:CreateUseu",
         "Iam:CreateUseu",
         "Iam:CreateUseu",
         "Iam:CreateUseu",
         "Iam:CreateUseu",
```



Note

前述の Amazon S3 バケットのいずれかにアクセスしようとしたときにアクセ ス拒否エラーが表示される場合は、バケットのリソース ARN の /* の部分 (例: arn:aws:s3:::aws-codedeploy-sa-east-1) を省略してみてください。

ステップ 2: 登録コマンドを呼び出す

このステップでは、オンプレミスインスタンス自体からオンプレミスインスタンスを登録している ことを想定します。オンプレミスインスタンスは、前のステップで説明したように AWS CLI インス トールおよび設定された別のデバイスまたはインスタンスから登録することもできます。

を使用して register コマンドを AWS CLI 呼び出し、以下を指定します。

 CodeDeploy に対してオンプレミスインスタンスを一意に識別する名前 (--instance-name のオ プションを用いて)。

▲ Important

後でオンプレミスインスタンスを識別するために、特にデバッグのため、オンプレミスイ ンスタンスの一意な特徴を示す名前 (例えば、もしあれば、シリアルナンバーや一意の内 部アセット識別子など)を使用することを強くお勧めします。名前に MAC アドレスを指 定した場合、MAC アドレスにはコロン (:) など、CodeDeploy が許可しない文字が含まれ ることに注意してください。許可された文字の一覧については、「<u>CodeDeploy のクォー</u> タ」を参照してください。

必要に応じて、このオンプレミスインスタンスと関連付ける既存の IAM ユーザーの ARN (--iam-user-arn のオプションを用いて) ユーザーの ARN を取得するには、get-user コマンドを呼び出すか、または、IAM コンソールの Users セクションで IAM ユーザー名を選択した後、Summary セクションで User ARN の値を見つけます。このオプションを指定しない場合、CodeDeploy はユーザーに代わって AWS アカウントで IAM ユーザーを作成し、オンプレミスインスタンスに関連付けます。

▲ Important

--iam-user-arn オプションを指定する場合は、「<u>ステップ 4: オンプレミスインスタン</u> <u>スに設定ファイルを追加</u>」の説明にあるとおり、オンプレミスインスタンスの設定ファイ ルを手動で作成することも必要です。

1 つのオンプレミスインスタンスのみに対し、1 人の IAM ユーザーのみを関連付けること ができます。複数のオンプレミスインスタンスに 1 人の IAM ユーザーを関連付けようとす ると、エラー、オンプレミスインスタンスへのデプロイの失敗、またはオンプレミスイン スタンスへのデプロイが無期限の保留状態のままとなります。

- 必要に応じて、デプロイ先の Amazon EC2 インスタンスのセットを識別するために CodeDeploy が使用するオンプレミスインスタンスタグのセット (--tags の オプションを用いて)。各 タグを Key=*tag-key*, Value=*tag-value* で指定します (例: Key=Name, Value=Beta Key=Name, Value=WestRegion)。このオプションを指定しない場合、タグは登録されません。 後でタグを登録するには、add-tags-to-on-premises-instances コマンドを呼び出します。
- オプションで、オンプレミスインスタンスが CodeDeploy に登録される AWS リージョン(-regionオプションを使用)。これは、「AWS 全般のリファレンス」(例: us-west-2)の「リー <u>ジョンエンドポイント</u>」にリストされているサポートされたリージョンの1つである必要があり ます。このオプションを指定しない場合、呼び出し元の IAM ユーザーに関連付けられたデフォル トの AWS リージョンが使用されます。

以下に例を示します。

aws deploy register --instance-name AssetTag12010298EX --iam-userarn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem --tags Key=Name,Value=CodeDeployDemo-OnPrem --region us-west-2

register コマンドは次のことを行います。

- 1. 既存の IAM ユーザーを指定しない場合、IAM ユーザーを作成して必要なアクセス権限を付与し、 対応するシークレットキーおよびシークレットアクセスキーを生成します。オンプレミスインス タンスは、この IAM ユーザーとアクセス権限および認証情報を使用して CodeDeploy との認証お よび操作を行います。
- 2. CodeDeploy でオンプレミスインスタンスを登録します。
- 3. 指定されている場合には、CodeDeploy 中で、--tags のオプションで指定したタグと、登録済み のオンプレミスインスタンスの名前を関連付けます。

4. IAM ユーザーが作成されている場合、register のコマンドの呼び出し元と同じディレクトリに必要 な設定ファイルも作成します。

このコマンドでエラーが発生した場合、エラーメッセージが表示され、手動で残りのステップを完了 する方法について説明します。そうでない場合は、成功メッセージが表示され、次のステップに示す とおり、install コマンドを呼び出す方法について説明します。

ステップ 3: インストールコマンドを呼び出す

オンプレミスインスタンスから、 AWS CLI を使用して<u>インストール</u>コマンドを呼び出し、以下を指 定します。

- ・ 設定ファイルへのパス (--config-file オプションで指定)。
- 必要に応じて、オンプレミスインスタンスにある既存の設定ファイルを置き換えるかどうか (-override-config オプションで指定)。指定しない場合、既存の設定ファイルは置き換えられま せん。
- オプションで、オンプレミスインスタンスが CodeDeploy に登録される AWS リージョン(-regionオプションを使用)。これは、「AWS 全般のリファレンス」(例: us-west-2)の「リー <u>ジョンエンドポイント</u>」にリストされているサポートされたリージョンの 1 つである必要があり ます。このオプションを指定しない場合、呼び出し元の IAM ユーザーに関連付けられたデフォル トの AWS リージョンが使用されます。
- 必要に応じて、CodeDeploy エージェントのインストール元のカスタムロケーション (--agentinstaller のオプションを用いて)。このオプションは、CodeDeploy が公式にはサポートしてい ない <u>CodeDeploy エージェント</u>のカスタムバージョンのインストールに使用できます (GitHub の CodeDeploy エージェント リポジトリに基づくカスタムバージョンなど)。値は、次のいずれかを 含む Amazon S3 バケットへのパスである必要があります。
 - ・ CodeDeploy エージェントのインストールスクリプト (GitHub の <u>CodeDeploy agent</u> エージェン トリポジトリにあるインストールファイルと同様、Linux または Unix ベースの OS 用)。
 - CodeDeploy エージェントのインストーラパッケージ (.msi) ファイル (Windows ベースの OS 用)。

このオプションを指定しない場合、CodeDeploy は独自の場所から、オンプレミスインスタンス上 の OS と互換性のある正式にサポートされているバージョンの CodeDeploy エージェントをイン ストールしようとします。

例:

オンプレミスインスタンスの登録

aws deploy install --override-config --config-file /tmp/codedeploy.onpremises.yml -region us-west-2 --agent-installer s3://aws-codedeploy-us-west-2/latest/codedeployagent.msi

install コマンドは次のことを行います。

- 1. オンプレミスインスタンスが Amazon EC2 インスタンスかどうかを確認します。そうである場合 は、エラーメッセージが表示されます。
- 2. オンプレミスインスタンスの設定ファイルを、インスタンスの指定された場所から CodeDeploy エージェントが見つけやすい場所へ (まだその場所にない場合) コピーします。

Ubuntu サーバーおよび Red Hat Enterprise Linux (RHEL)の場合、これは /etc/codedeployagent/conf/codedeploy.onpremises.yml になります。

Windows サーバーの場合、これは C:\ProgramData\Amazon \CodeDeploy\conf.onpremises.yml になります。

--override-config オプションを指定した場合は、ファイルを作成または上書きします。

3. オンプレミスインスタンスに CodeDeploy エージェントをインストールし、起動します。

ステップ 4: デプロイアプリケーションリビジョンをオンプレミスインスタンスにデプロイする

登録され、タグ付けされたオンプレミスインスタンスにアプリケーションリビジョンをデプロイする 準備ができました。

Amazon EC2 インスタンスにアプリケーションリビジョンをデプロイするのと同様の方法で オンプレミスインスタンスにアプリケーションリビジョンをデプロイします。手順について は、<u>CodeDeploy でデプロイを作成する</u> を参照してください。これらの指示は、アプリケーショ ンの作成、開発グループの作成、およびアプリケーションリビジョンの準備を含む前提条件と関 連しています。シンプルなサンプルアプリケーションリビジョンをデプロイすることが必要な場合 は、<u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エンター</u> <u>プライズ Linux) を使用してオンプレミスインスタンスにアプリケーションをデプロイします。</u>の<u>ス</u> <u>テップ 2: サンプルのアプリケーションリビジョンを作成する</u> で説明してあるものを作成できます。

A Important

オンプレミスインスタンスを対象としたデプロイグループの作成の一部として、既存の CodeDeploy サービスロールを再利用する場合は、Tag:get* をサービスロールのポリシー ステートメントの Action の部分に含める必要があります。詳細については、「<u>ステップ 2:</u> CodeDeployのサービスのロールを作成する」を参照してください。

ステップ 5: オンプレミスインスタンスへのデプロイを追跡

登録されタグ付けされたオンプレミスインスタンスへアプリケーションリビジョンをデプロイした 後、デプロイの進行状況を追跡できます。

Amazon EC2 インスタンスへのデプロイの追跡と同様の方法でオンプレミスインスタンスへのデプ ロイの追跡をします。手順については、<u>CodeDeploy デプロイの詳細を表示する</u> を参照してくださ い。

他のオプションについては、「<u>CodeDeploy でのオンプレミスインスタンスのオペレーションの管</u> 理」を参照してください。

[register-on-premises-instance] コマンド (IAM ユーザー ARN) を使用してオンプレミ スインスタンスを登録します。

▲ Important

IAM ユーザーを使用してインスタンスを登録することは、認証に静的 (永続的) 認証情報を使 用するため推奨されません。セキュリティ向上のため、認証には一時的な認証情報を使用し てインスタンスを登録することをお勧めします。詳細については、「<u>[register-on-premises-</u> <u>instance] コマンド (IAM セッション ARN) を使用してオンプレミスインスタンスを登録</u>」を 参照してください。

▲ Important

IAM ユーザーのアクセスキー (永続的認証情報) をローテーションする計画を立ててくださ い。アクセスキーのローテーションの詳細については、「<u>アクセスキーの更新</u>」を参照して ください。

認証に固定 IAM ユーザー認証情報を使って、CodeDeploy でオンプレミスインスタンスの設定、登 録、タグ付けをほとんど自分で行う場合は、これらの手順に従います。

トピック

オンプレミスインスタンスの登録

- ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する
- ステップ 2: ユーザーにアクセス権限を割り当てる
- ステップ 3: ユーザーの認証情報を取得する
- ステップ 4: オンプレミスインスタンスに設定ファイルを追加
- ステップ 5: をインストールして設定する AWS CLI
- ステップ 6: AWS_REGION 環境変数を設定する (Ubuntu Server および RHEL のみ)
- ・ ステップ 7: CodeDeploy エージェントのインストール
- ステップ 8: CodeDeploy でオンプレミスインスタンスを登録します。
- ステップ 9: オンプレミスインスタンスにタグ付けする
- ステップ 10: アプリケーションリビジョンをオンプレミスインスタンスにデプロイする
- ステップ 11: オンプレミスインスタンスへのデプロイを追跡する

ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する

オンプレミスインスタンスが認証や CodeDeploy とのやり取りに使用する IAM ユーザーを作成しま す。

Important

それぞれの参加しているオンプレミスインスタンスに対して個別の IAM ユーザーを作成する 必要があります。複数のオンプレミスインスタンスに対して単独の IAM ユーザーを再利用し ようとすると、CodeDeploy を用いての、オンプレミスインスタンスの登録やタグ付けがで きない場合があります。それらのオンプレミスインスタンスへのデプロイは、無期限の保留 状態のまま、または完全にエラーとなる場合があります。

IAM ユーザーに、目的を識別するために、[CodeDeployUser-OnPrem] のような名前を割り当てることをお勧めします。

AWS CLI または IAM コンソールを使用して IAM ユーザーを作成できます。詳細については、 「Creating an IAM user in your AWS account を参照してください。

A Important

AWS CLI または IAM コンソールを使用して新しい IAM ユーザーを作成するかどうかにかか わらず、そのユーザーに提供されるユーザー ARN を書き留めます。この情報は後に ステッ <u>プ 4: オンプレミスインスタンスに設定ファイルを追加</u> と <u>ステップ 8: CodeDeploy でオンプ</u> レミスインスタンスを登録します。 で必要になります。

ステップ 2: ユーザーにアクセス権限を割り当てる

オンプレミスインスタンスが Amazon S3 バケットからアプリケーションリビジョンをデプロイする 場合、IAM ユーザーがバケットとやり取りするための権限を割り当てる必要があります。 AWS CLI または IAM コンソールを使用してアクセス許可を割り当てることができます。

Note

GitHub リポジトリからのみアプリケーションリビジョンをデプロイする場合は、このステッ プを飛ばして <u>ステップ 3: ユーザーの認証情報を取得する</u> へ進んでください。(<u>ステップ 1:</u> <u>オンプレミスインスタンス用に IAM ユーザーを作成する</u> で作成した IAM ユーザーに関する 情報はまだ必要です。後のステップで使用します。)

アクセス権限 (CLI) を割り当てるには

 AWS CLIを呼び出すのに使用している Amazon EC2インスタンスまたはデバイス上に、 次のポリシーの内容を用いてファイルを作成します。ファイルに CodeDeploy-OnPrem-Permissions.json のような名前を付けて、ファイルを保存します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
               "s3:Get*",
               "s3:List*"
        ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Note

このポリシーを、オンプレミスインスタンスがアクセスする必要のある Amazon S3 バ ケットにのみ制限することをお勧めします。このポリシーを制限する場合は、 AWS CodeDeploy エージェントを含む Amazon S3 バケットへのアクセスも許可してくださ い。そうしない場合、CodeDeploy エージェントが関連するオンプレミスインスタンス にインストールされる、または更新されるたびに、エラーが発生する可能性がありま す。

```
例:
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
```



put-user-policy コマンドを呼び出し、IAM ユーザーの名前 (--user-name のオプションを用いて)、ポリシーの名前 (--policy-name のオプションを用いて)、および新しく作成したポリシードキュメントへのパス (--policy-document のオプションを用いて) を指定します。例えば、CodeDeploy-OnPrem-Permissions.json というファイルが、コマンドを呼び出しているのと同じディレクトリ (フォルダ) にあるとします。

▲ Important

ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name CodeDeploy-OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-Permissions.json

アクセス権限を割り当てるには (コンソール)

- 1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
- 2. ナビゲーションペインで、[Policies] を選択し、[Create Policy] を選択します。([Get Started] ボ タンが表示された場合は、そのボタンを選択してから、[Create Policy] を選択します)。
- 3. [独自のポリシーを作成] の横で、[選択] を選択します。
- [ポリシー名] ボックスに、このポリシーの名前を入力します (CodeDeploy-OnPrem-Permissions など)。
- ポリシードキュメント ボックスに、次のアクセス許可式を入力または貼り付けます。これにより、 AWS CodeDeploy は、ポリシーで指定された Amazon S3 バケットから IAM ユーザーに代わってオンプレミスインスタンスにアプリケーションリビジョンをデプロイできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
               "s3:Get*",
              "s3:List*"
        ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

- 6. [ポリシーの作成]を選択します。
- 7. ナビゲーションペインで [ユーザー] を選択します。
- 8. ユーザーのリストで、<u>ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する</u> で作 成した IAM ユーザーを探して選択します。
- 9. [Permissions] タブを開き、[Managed Policies] で [Attach Policy] を選択します。
- CodeDeploy-OnPrem-Permissions という名前のポリシーを選択した後、[ポリシーのアタッチ] を選択します。

ステップ 3: ユーザーの認証情報を取得する

IAM ユーザーのシークレットキー ID およびシークレットアクセスキーを取得します。これは、<u>ス</u> <u>テップ 4: オンプレミスインスタンスに設定ファイルを追加</u> で必要になります。 AWS CLI または IAM コンソールを使用して、シークレットキー ID とシークレットアクセスキーを取得できます。

Note

すでにシークレットキー ID およびシークレットアクセスキーがある場合、このステップを 飛ばして <u>ステップ 4: オンプレミスインスタンスに設定ファイルを追加</u> へ進んでください。 ユーザーが の AWS 外部とやり取りする場合は、プログラムによるアクセスが必要です AWS Management Console。プログラムによるアクセスを許可する方法は、 がアクセスす るユーザーのタイプによって異なります AWS。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選 択します。

プログラマチックアクセス 権を必要とするユーザー	目的	方法
ワークフォースアイデン ティティ (IAM アイデンティティセン ターで管理されているユー ザー)	ー時的な認証情報を使用 して AWS CLI、、 AWS SDKs、または AWS APIs。	使用するインターフェイス の指示に従ってください。 ・ については AWS CLI、AWS Command Line Interface 「 ユー ザーガイド」の「 を 使用する AWS CLI よ うに AWS IAM Identity Centerを設定する」を参 照してください。 ・ AWS SDKs、ツール、 API については、SDK および AWS APIs「IAM Identity Center 認証」 を参照してください。 AWS SDKs
IAM	ー時的な認証情報を使用 して AWS CLI、、 AWS SDKs、または AWS APIs。	「IAM <u>ユーザーガイド」の</u> <u>「AWS リソースでの一時</u> <u>的な認証情報</u> の使用」の手 順に従います。

プログラマチックアクセス 権を必要とするユーザー	目的	方法
	(非推奨) 長期認証情報を使用して、 AWS CLI、AWS SDKs、ま たは AWS APIs。	使用するインターフェイス の指示に従ってください。 ・ については AWS CLI、 「AWS Command Line Interface ユーザーガイ ド」の「IAM ユーザー認 証情報を使用した認証」 を参照してください。 ・ AWS SDKs「SDK と ツールリファレンスガイ ド <u>」の「長期的な認証情</u> 報を使用した認証」を参 照してください。AWS SDKs ・ API AWS APIs <u>「IAM</u> ユーザーガイド」の 「IAM ユーザーのアクセ スキーの管理」を参照し てください。

認証情報 (CLI) を取得するには

<u>list-access-keys</u> コマンドを呼び出し、IAM ユーザーの名前を指定し (--user-name のオプションを用いて)、アクセスキー ID のみのクエリを実行します (--query および --output のオプションを用いて)。例:

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query
"AccessKeyMetadata[*].AccessKeyId" --output text
```

2. キーが出力に表示されないか、1 つのキーについての情報だけが出力に表示されたら、<u>create-access-key</u> コマンドを呼び出し、IAM ユーザーの名前を指定します (--user-name のオプションを用いて)。

aws iam create-access-key --user-name CodeDeployUser-OnPrem

create-access-key コマンドを呼び出した出力で、AccessKeyId フィールドおよび SecretAccessKey フィールドの値をメモします。この情報は <u>ステップ 4: オンプレミスインス</u> タンスに設定ファイルを追加 で必要になります。

Important

このシークレットアクセスキーにアクセスできるのは、この時だけです。このシーク レットアクセスキーを忘れた、またはアクセスできなくなった場合、<u>ステップ 3: ユー</u> ザーの認証情報を取得する のステップに従い、新しいキーを生成する必要があります。

 2 つのアクセスキーが既にリストされている場合は、そのうちの1つを削除する必要がありま す。削除するには、delete-access-key コマンドを呼び出して、IAM ユーザーの名前(--username のオプションを用いて)と削除するアクセスキーのID(--access-key-id のオプショ ンを用いて)を指定します。それから、このステップの前のほうにある説明のとおり、createaccess-key コマンドを呼び出します。delete-access-key コマンドを呼び出す例は以下のとおり です。

aws iam delete-access-key --user-name CodeDeployUser-OnPrem --access-key-id accesskey-ID

A Important

このアクセスキーの 1 つを削除するために delete-access-key のコマンドを呼び出すと き、および <u>ステップ 4: オンプレミスインスタンスに設定ファイルを追加</u> で説明するよ うにオンプレミスインスタンスがすでにこのアクセスキーを使用しているときは、<u>ス</u> <u>テップ 4: オンプレミスインスタンスに設定ファイルを追加</u> の説明に再び従って、この IAM ユーザーに関連付けられる別のアクセスキー ID とシークレットアクセスキーを指 定する必要があります。そうしない場合、そのオンプレミスインスタンスへのデプロイ は、無期限の保留状態のまま、または完全にエラーとなる可能性があります。

認証情報を取得するには (コンソール)

1. a. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

- b. ユーザーのリストがナビゲーションペインに表示されない場合は、[Users] を選択します。
- c. ユーザーのリストで、<u>ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する</u> で作成した IAM ユーザーを探して選択します。
- [Security credentials] タブにキーが表示されていない場合、または1つしか表示されていない場合は、[Create access key] を選択します。

2 つのアクセスキーが表示されている場合は、片方を削除する必要があります。片方のアクセス キーの横にある [Delete] を選択した後、[Create access key] を選択します。

A Important

このいずれかのアクセスキーの横にある Delete を選択する場合、および、オンプレミス インスタンスが <u>ステップ 4: オンプレミスインスタンスに設定ファイルを追加</u> にある説 明のとおりにすでにそのアクセスキーを使用している場合は、<u>ステップ 4: オンプレミス インスタンスに設定ファイルを追加</u> の手順に再び従って、この IAM ユーザーと関連付 けられた異なるアクセスキー ID およびシークレットアクセスキーを指定する必要があ ります。そうしない場合、そのオンプレミスインスタンスへのデプロイは、無期限の保 留状態のまま、または完全にエラーとなる可能性があります。

 [Show] を選択し、アクセスキー ID とシークレットアクセスキーをメモします。この情報は、 次のステップで必要になります。または、[Download .csv file] を選択して、アクセスキー ID と シークレットアクセスキーのコピーを保存することができます。

▲ Important

認証情報をメモする、または、ダウンロードするのでない限り、このシークレットアク セスキーにアクセスできるのは、この時だけです。このシークレットアクセスキーを忘 れた、またはアクセスできなくなった場合、<u>ステップ 3: ユーザーの認証情報を取得する</u> のステップに従い、新しいキーを生成する必要があります。

4. [Close] を選択して [Users > IAM User Name] ページに戻ります。

ステップ 4: オンプレミスインスタンスに設定ファイルを追加

ルートまたは管理者権限を使用して、オンプレミスインスタンスに設定ファイルを追加します。こ の設定ファイルは、IAM ユーザー認証情報と CodeDeploy に使用するターゲット AWS リージョンを 宣言するために使用されます。ファイルは、オンプレミスインスタンスの指定の場所に追加する必 要があります。ファイルには、IAM ユーザーの ARN、シークレットキー ID、シークレットアクセス キー、およびターゲット AWS リージョンが含まれている必要があります。ファイルは特定の形式に 従っている必要があります。

- オンプレミスインスタンス上の以下の場所に、codedeploy.onpremises.yml (Ubuntu サーバーまたは RHEL オンプレミスインスタンスの場合)、または conf.onpremises.yml (Windows サーバーのオンプレミスインスタンスの場合) という名前のファイルを作成します。
 - Ubuntu サーバーの場合:/etc/codedeploy-agent/conf
 - Windows サーバーについて: C:\ProgramData\Amazon\CodeDeploy
- テキストエディタを使用して、新しく作成した codedeploy.onpremises.yml または conf.onpremises.yml ファイルに次の情報を追加します。

```
---
aws_access_key_id: secret-key-id
aws_secret_access_key: secret-access-key
iam_user_arn: iam-user-arn
region: supported-region
```

コードの説明は以下のとおりです。

- secret-key-id は、ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する または ステップ 3: ユーザーの認証情報を取得する でメモした、対応するIAM ユーザーのシー クレットキー ID です。
- secret-access-key は、ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成 する または ステップ 3: ユーザーの認証情報を取得する でメモした、対応する IAM ユーザー のシークレットアクセスキーです。
- *iam-user-arn*は、「ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成す る」で先にメモした IAM ユーザーの ARN です。
- supported-regionは、CodeDeployでサポートされているリージョンの識別子で、CodeDeployアプリケーション、デプロイグループ、およびアプリケーションリビジョンがある場所です (us-west-2 など)。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「<u>リージョンエンドポイント</u>」を参照してください。

▲ Important

ステップ 3: ユーザーの認証情報を取得する の片方のアクセスキーの横にある Delete を 選択し、またオンプレミスインスタンスがすでに関連するアクセスキー ID およびシー クレットアクセスキーを使用している場合は、ステップ 4: オンプレミスインスタンスに 設定ファイルを追加 の手順に再び従って、この IAM ユーザーと関連付けられた異なる アクセスキー ID およびシークレットアクセスキーを指定する必要があります。そうし ない場合、オンプレミスインスタンスへのデプロイは、無期限の保留状態のまま、また は完全にエラーとなる可能性があります。

ステップ 5: をインストールして設定する AWS CLI

オンプレミスインスタンス AWS CLI に をインストールして設定します。(AWS CLI は で使用さ れ<u>ステップ 7: CodeDeploy エージェントのインストール</u>、CodeDeploy エージェントをダウンロー ドしてオンプレミスインスタンスにインストールします。)

1. オンプレミスインスタンス AWS CLI に をインストールするには、「 AWS Command Line Interface ユーザーガイド」の「 のセットアップ AWS CLI」の手順に従います。

Note

オンプレミスインスタンスを使用するための CodeDeploy コマンドは、 AWS CLIのバー ジョン 1.7.19 で使用できるようになりました。のバージョンが AWS CLI 既にインス トールされている場合は、 を呼び出してそのバージョンを確認できますaws --version。

2. オンプレミスインスタンス AWS CLI で を設定するには、AWS Command Line Interface 「 ユー ザーガイド」の「 の設定 AWS CLI」の手順に従います。

A Important

を設定するときは AWS CLI (aws configure コマンドを呼び出すなど)、 で指定された アクセス許可に加えて、少なくとも以下の AWS アクセス許可を持つ IAM ユーザーの シークレットキー ID とシークレットアクセスキーを必ず指定してください<u>オンプレミ</u> <u>スインスタンスを設定するための前提条件</u>。これにより、オンプレミスインスタンス上 で CodeDeploy エージェントをダウンロードしてインストールすることができます。

}

```
"Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

これらのアクセス権限は、<u>ステップ 1: オンプレミスインスタンス用に IAM ユーザーを</u> 作成する で作成した IAM ユーザー、または別のユーザーのいずれかに割り当てられる ことができます。これらのアクセス権限を ユーザーに割り当てるには、<u>ステップ 1: オ</u> <u>ンプレミスインスタンス用に IAM ユーザーを作成する</u> にある手順に従いますが、手順 のアクセス権限の代わりにこれらのアクセス権限を使用します。

ステップ 6: AWS_REGION 環境変数を設定する (Ubuntu Server および RHEL のみ)

オンプレミスインスタンス上で Ubuntu サーバーまたは RHEL を実行中でなければ、このステップを スキップして直接 ステップ 7: CodeDeploy エージェントのインストール へ進んでください。

Ubuntu サーバーまたは RHEL オンプレミスインスタンス上に CodeDeploy エージェントをインス トールし、新しいバージョンが使用可能になったらいつでも CodeDeploy エージェントを更新する ようにインスタンスを有効にします。これを行うには、インスタンス上の AWS_REGION の環境変数 を CodeDeploy がサポートしているリージョンのうちの 1 つの識別子に設定します。CodeDeploy アプリケーション、デプロイグループ、およびアプリケーションリビジョンのある (us-west-2 な ど)リージョンの値に設定することをお勧めします。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「リージョンエンドポイント」を参照してください。

環境変数を設定するには、端末から以下を呼び出します。

export AWS_REGION=supported-region

supported-region がリージョンの識別子である場所 (例:us-west-2)。

ステップ 7: CodeDeploy エージェントのインストール

オンプレミスインスタンスに CodeDeploy エージェントをインストールします。

- Ubuntu サーバーオンプレミスインスタンスの場合は、Ubuntu サーバー用の CodeDeploy エー ジェントをインストールするの手順に従った後、このページに戻ります。
- RHEL オンプレミスインスタンスについては、<u>Amazon Linux または RHEL 用の CodeDeploy エー</u> ジェントをインストールするの手順に従った後、このページに戻ります。
- Windows Server オンプレミスインスタンスの場合は、<u>Windows サーバー用の CodeDeploy エー</u>ジェントです。の手順に従った後、このページに戻ります。

ステップ 8: CodeDeploy でオンプレミスインスタンスを登録します。

このステップの手順では、オンプレミスインスタンス自体からオンプレミスインスタンスを登録して いることを想定します。「」で説明されているように、 AWS CLI がインストールされ、設定された 別のデバイスまたはインスタンスからオンプレミスインスタンスを登録できます<u>ステップ 5: をイン</u> ストールして設定する AWS CLI。

AWS CLI を使用してオンプレミスインスタンスを CodeDeploy に登録し、デプロイで使用できるようにします。

 を使用する前に AWS CLI、 で作成した IAM ユーザーのユーザー ARN が必要ですステップ <u>1: オンプレミスインスタンス用に IAM ユーザーを作成する</u>。ユーザー ARN がまだない場合 は、get-user コマンドを呼び出し、IAM ユーザーの名前を指定し (--user-name のオプショ ンを用いて)、ユーザー ARN のみをクエリします (--query のオプションと --output のオプ ションを用いて)。

aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --output text

- 2. register-on-premises-instance コマンドを呼び出し、以下を指定します。
 - オンプレミスインスタンスを一意に識別する名前 (--instance-name オプションで指定)。

▲ Important

オンプレミスインスタンスを識別するために、特にデバッグのため、オンプレミス インスタンスの一意な特徴を示す名前 (例えば、もしあれば、シリアルナンバーや 内部アセット識別子など)を指定することを強くお勧めします。名前として MAC ア ドレスを指定した場合、MAC アドレスには、コロン (:) など CodeDeploy が許可し ない文字が含まれることに注意してください。許可された文字の一覧については、 「CodeDeploy のクォータ」を参照してください。

 <u>ステップ 1: オンプレミスインスタンス用に IAM ユーザーを作成する</u> で作成した IAM ユー ザーの ARN (--iam-user-arn のオプションを用いて)

例:

aws deploy register-on-premises-instance --instance-name AssetTag12010298EX -iam-user-arn arn:aws:iam::4444555566666:user/CodeDeployUser-OnPrem ステップ 9: オンプレミスインスタンスにタグ付けする

AWS CLI または CodeDeploy コンソールを使用して、オンプレミスインスタンスにタグを付ける ことができます。(CodeDeploy はオンプレミスインスタンスタグを使用してデプロイ中にデプロイ ターゲットを識別します。)

オンプレミスインスタンスにタグ付けするには (CLI)

- add-tags-to-on-premises-instances コマンドを呼び出し、以下を指定します。
 - オンプレミスインスタンスを一意に識別する名前 (--instance-names オプションで指定)。
 - 使用するオンプレミスインスタンスのタグキーの名前とタグ値 (--tags オプションで指定)。
 名前と値はいずれも指定する必要があります。CodeDeploy は値のみがあるオンプレミスインスタンスタグを許可しません。

例:

aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem

オンプレミスインスタンスにタグ付けするには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. CodeDeploy メニューから On-premises instances を選択します。
- オンプレミスインスタンスのリストで、タグ付けするオンプレミスインスタンスの横にある矢印 を選択します。
- タグのリストで、目的のタグキーとタグ値を選択または入力します。タグキー とタグ値を入力するたびに、別の行が表示されます。最大 10 個のタグにこれ を繰り返すことができます。タグを削除するには、削除アイコンを選択します (3)
- 5. タグを追加したら、[Update Tags] を選択します。

)。
ステップ 10: アプリケーションリビジョンをオンプレミスインスタンスにデプロイする

登録され、タグ付けされたオンプレミスインスタンスにアプリケーションリビジョンをデプロイする 準備ができました。

Amazon EC2 インスタンスにアプリケーションリビジョンをデプロイするのと同様の方法で オンプレミスインスタンスにアプリケーションリビジョンをデプロイします。手順について は、<u>CodeDeploy でデプロイを作成する</u>を参照してください。これらの指示には、アプリケーショ ンの作成、開発グループの作成、およびアプリケーションリビジョンの準備を含む前提条件へのリン クが含まれています。シンプルなサンプルアプリケーションリビジョンをデプロイすることが必要な 場合は、<u>チュートリアル: CodeDeploy (Windows サーバー、Ubuntu サーバー、または Red Hat エン</u> <u>タープライズ Linux</u>)を使用してオンプレミスインスタンスにアプリケーションをデプロイします。 の <u>ステップ 2: サンプルのアプリケーションリビジョンを作成する</u> で説明してあるものを作成できま す。

▲ Important

オンプレミスインスタンスを対象としたデプロイグループの作成の一部とし て、CodeDeploy サービスロールを再利用する場合は、Tag:get* をサービスロールのポリ シーステートメントの Action の部分に含める必要があります。詳細については、「<u>ステッ</u> プ 2: CodeDeployのサービスのロールを作成する」を参照してください。

ステップ 11: オンプレミスインスタンスへのデプロイを追跡する

登録されタグ付けされたオンプレミスインスタンスへアプリケーションリビジョンをデプロイした 後、デプロイの進行状況を追跡できます。

Amazon EC2 インスタンスへのデプロイの追跡と同様の方法でオンプレミスインスタンスへのデプ ロイの追跡をします。手順については、<u>CodeDeploy デプロイの詳細を表示する</u> を参照してくださ い。

CodeDeploy でのオンプレミスインスタンスのオペレーションの管理

このセクションの手順に従って、オンプレミスインスタンスに関する詳細情報の取得、タグの削除、オンプレミスインスタンスのアンインストールと登録解除など、オンプレミスインスタンスを CodeDeployに登録した後の操作を管理します。

トピック

- 単一のオンプレミスインスタンスに関する情報を取得します
- 複数のオンプレミスインスタンスに関する情報を取得します
- オンプレミスインスタンスから手動でオンプレミスインスタンスタグを削除します
- <u>CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスインスタン</u> スから削除します。
- オンプレミスインスタンスを自動的に登録解除します
- 手動でオンプレミスのインスタンスの登録を解除します

単一のオンプレミスインスタンスに関する情報を取得します

<u>CodeDeploy デプロイの詳細を表示する</u>にある手順に従って、単一のオンプレミスインスタンスに 関する情報を取得できます。 AWS CLI または CodeDeploy コンソールを使用して、単一のオンプレ ミスインスタンスに関する詳細情報を取得できます。

単一のオンプレミスインスタンスについての情報を取得するには (CLI)

 <u>get-on-premises-instance</u> コマンドを呼び出し、オンプレミスインスタンスを一意に識別する名 前を指定します (--instance-name のオプションを用いて)。

aws deploy get-on-premises-instance --instance-name AssetTag12010298EX

単一のオンプレミスインスタンスに関する情報を取得するには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeployの開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、Deploy を展開し、On-premises instances を選択します。
- オンプレミスインスタンスのリストで、詳細を表示するオンプレミスインスタンスの名前を選択します。

複数のオンプレミスインスタンスに関する情報を取得します

<u>CodeDeploy デプロイの詳細を表示する</u>にある手順に従って、オンプレミスインスタンスに関する 情報を取得できます。 AWS CLI または CodeDeploy コンソールを使用して、オンプレミスインスタ ンスに関する詳細情報を取得できます。

複数のオンプレミスインスタンスに関する情報を取得するには (CLI)

- 1. オンプレミスインスタンスの名前のリストでは、<u>list-on-premises-instances</u> コマンドを呼び出し、以下を指定します。
 - すべての登録または登録解除されたオンプレミスインスタンスに関する情報を取得するかどうか (--registration-status オプションの Registered または Deregistered でそれぞれ指定)。これを省略すると、登録されている、および登録解除されたオンプレミスインスタンスの名前が両方とも返されます。
 - 特定のオンプレミスインスタンスのタグが付けられたオンプレミスインスタンスに関する情報 だけを取得するかどうか (--tag-filters オプションで指定)。各オンプレミスインスタン スタグで、Key、Value、および Type を指定します (常に KEY_AND_VALUE である必要があ ります)。複数のオンプレミスインスタンスタグは、Key、Value、Type の間にスペースを入 れて区切ります。

例:

aws deploy list-on-premises-instances --registration-status Registered --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE Key=Name,Value=CodeDeployDemo-OnPrem-Beta,Type=KEY_AND_VALUE

2. 詳細については、オンプレミスインスタンスの名前で (--instance-names のオプションを用いて)、batch-get-on-premises-instances コマンドを呼び出します。

aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
AssetTag09920444EX

複数のオンプレミスインスタンスに関する情報を取得するには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。 Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで、Deploy を展開し、デプロイを選択し、On-premises instances を選択 します。

オンプレミスインスタンスに関する情報が表示されます。

オンプレミスインスタンスから手動でオンプレミスインスタンスタグを削除します

通常、タグが使われなくなった場合や、タグに依存するデプロイグループからオンプレミスインスタ ンスを削除する場合は、オンプレミスインスタンスからオンプレミスインスタンスタグを削除しま す。 AWS CLI または AWS CodeDeploy コンソールを使用して、オンプレミスインスタンスからオ ンプレミスインスタンスタグを削除できます。

登録を解除する前にオンプレミスインスタンスからオンプレミスインスタンスタグを削除する必要は ありません。

オンプレミスインスタンスからオンプレミスインスタンスタグを手動で削除しても、インスタンス の登録は解除されません。CodeDeploy エージェントはインスタンスからアンインストールされま せん。インスタンスから設定ファイルは削除されません。インスタンスに関連付けられた IAM ユー ザーは削除されません。

自動的にオンプレミスインスタンスの登録を解除するには、<u>オンプレミスインスタンスを自動的に登</u> 録解除します を参照してください。

オンプレミスインスタンスの登録を手動で解除するには、<u>手動でオンプレミスのインスタンスの登録</u> を解除します を参照してください。

CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスインスタンス から削除するには、CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプ レミスインスタンスから削除します。 を参照してください。

CodeDeploy エージェントだけをオンプレミスインスタンスから手動でアンインストールするに は、CodeDeploy エージェントのオペレーションの管理 を参照してください。

関連する IAM ユーザーを手動で削除するには、<u>Deleting an IAM user from your AWS account</u> を参照 してください。 オンプレミスインスタンスから手動でオンプレミスインスタンスタグを削除するには (CLI)

- remove-tags-from-on-premises-instances を呼び出し、以下を指定します。
 - オンプレミスインスタンスを一意に識別する名前 (--instance-names オプションで指定)。
 - 削除するタグの名前と値 (--tags オプションで指定)。

例:

aws deploy remove-tags-from-on-premises-instances --instance-names
AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem

オンプレミスインスタンスから手動でオンプレミスインスタンスタグを削除するには (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、Deploy を展開し、On-premises instances を選択します。
- 3. オンプレミスインスタンスのリストで、タグを削除するオンプレミスインスタンスの名前を選択 します。
- 4. [タグ] で、削除する各タグの横にある [削除] を選択します。
- 5. タグを削除したら、[Update tags] を選択します。

CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスインスタンスから削除します。

通常、今後デプロイする予定がなければ、CodeDeploy エージェントをアンインストールし、オンプ レミスインスタンスから設定ファイルを削除します。

Note

CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスイン スタンスから削除しても、オンプレミスインスタンスの登録は解除されません。オンプレミ スインスタンスに関連付けられたオンプレミスインスタンスタグの関連付けは解除されませ ん。オンプレミスインスタンスに関連付けられた IAM ユーザーは削除されません。 自動的にオンプレミスインスタンスの登録を解除するには、<u>オンプレミスインスタンスを自</u> <u>動的に登録解除します</u>を参照してください。 オンプレミスインスタンスの登録を手動で解除するには、<u>手動でオンプレミスのインスタン</u> <u>スの登録を解除します</u>を参照してください。 オンプレミスインスタンスタグの関連付けを手動で解除するには、<u>オンプレミスインスタン</u>スタン スから手動でオンプレミスインスタンスタグを削除します</u>を参照してください。 オンプレミスインスタンスから CodeDeploy エージェントを手動でアンインストールするに は、<u>CodeDeploy エージェントのオペレーションの管理</u>を参照してください。 関連する IAM ユーザーを手動で削除するには、<u>Deleting an IAM user from your AWS account</u> を参照してください。

オンプレミスインスタンスから、 AWS CLI を使用してアンインストールコマンドを呼び出します。

以下に例を示します。

aws deploy uninstall

uninstall コマンドは次のことを行います。

- 1. オンプレミスインスタンスで実行の CodeDeploy エージェントを停止します。
- 2. オンプレミスインスタンスから CodeDeploy エージェントをアンインストールします。
- オンプレミスインスタンスから設定ファイルを削除します。(Ubuntu サーバーとRHELの場合、 これは/etc/codedeploy-agent/conf/codedeploy.onpremises.yml です。Windows サー バーの場合、これはC:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml です。)

オンプレミスインスタンスを自動的に登録解除します

通常は、デプロイする計画がなくなった後、オンプレミスインスタンスの登録を解除します。オンプレミスインスタンスの登録を解除すると、オンプレミスインスタンスがデプロイグループのオンプレミスインスタンスタグの一部であっても、オンプレミスインスタンスはデプロイには含まれなくなります。を使用して、オンプレミスインスタンスの登録 AWS CLI を解除できます。

Note

CodeDeploy コンソールを使用してオンプレミスインスタンスの登録を解除することはでき ません。また、オンプレミスインスタンスの登録を解除しても、オンプレミスインスタンス に関連付けられたオンプレミスインスタンスタグは解除されません。オンプレミスインスタ ンスから CodeDeploy エージェントをアンインストールしません。オンプレミスインスタン スの設定ファイルはオンプレミスインスタンスから削除されません。 このセクションのアクティビティの一部 (ただしすべてではない)を実行するために CodeDeploy コンソールを使うには、<u>手動でオンプレミスのインスタンスの登録を解除しま</u> <u>す</u>の CodeDeploy コンソールセクションを参照してください。 オンプレミスインスタンスタグの関連付けを手動で解除するには、<u>オンプレミスインスタン</u> スから手動でオンプレミスインスタンスタグを削除します CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスイン スタンスから削除するには、<u>CodeDeploy エージェントを自動でアンインストールし、設定</u> ファイルをオンプレミスインスタンスから削除します。を参照してください。 CodeDeploy エージェントのみをオンプレミスインスタンスから手動でアンインストールす るには、<u>CodeDeploy エージェントのオペレーションの管理</u>を参照してください。

を使用して AWS CLI 登録解除コマンドを呼び出し、以下を指定します。

- CodeDeploy に対してオンプレミスインスタンスを一意に識別する名前 (--instance-name のオ プションを用いて)
- 必要に応じて、オンプレミスインスタンスに関連付けられた IIAM ユーザーを削除するかどうか デフォルトの動作では、IAM ユーザーが削除されます。オンプレミスインスタンスに関連付けられた IAM ユーザーを削除しない場合は、コマンドで --no-delete-iam-user オプションを指定します。
- オプションで、オンプレミスインスタンスが CodeDeploy に登録された AWS リージョン (-regionオプションを使用)。これは、「AWS 全般のリファレンス」(例: us-west-2)の「リー <u>ジョンエンドポイント</u>」にリストされているサポート対象のリージョンの 1 つである必要があり ます。このオプションを指定しない場合、呼び出し元の IAM ユーザーに関連付けられたデフォル トの AWS リージョンが使用されます。

インスタンスを登録解除し、ユーザーを削除する例:

aws deploy deregister --instance-name AssetTag12010298EX --region us-west-2

インスタンスを登録解除し、ユーザーを削除しない例:

aws deploy deregister --instance-name AssetTag12010298EX --no-delete-iam-user --region
 us-west-2

deregister コマンドは次のことを行います。

1. CodeDeploy でオンプレミスインスタンスの登録を解除します。

2. 指定した場合、オンプレミスインスタンスに関連付けられている IAM ユーザーを削除します。

オンプレミスインスタンスの登録解除後:

コンソールにはすぐに表示されなくなります。

• 直後に同じ名前で別のインスタンスを作成できます。

このコマンドでエラーが発生した場合、エラーメッセージが表示され、手動で残りのステップを完了 する方法について説明します。それ以外の場合は、成功メッセージが表示され、uninstall コマンドを 呼び出す方法が説明されます。

手動でオンプレミスのインスタンスの登録を解除します

通常は、デプロイする計画がなくなった後、オンプレミスインスタンスの登録を解除します。を使用 して AWS CLI 、オンプレミスインスタンスを手動で登録解除します。

手動でオンプレミスインスタンスの登録を解除しても、CodeDeploy エージェントはアンインストー ルされません。インスタンスから設定ファイルは削除されません。インスタンスに関連付けられた IAM ユーザーは削除されません。インスタンスに関連付けられるタグは削除されません。

CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプレミスインスタンス から削除するには、CodeDeploy エージェントを自動でアンインストールし、設定ファイルをオンプ レミスインスタンスから削除します。 を参照してください。

手動で CodeDeploy エージェントのみをアンインストールするには、<u>CodeDeploy エージェントのオ</u> ペレーションの管理 を参照してください。

関連する IAM ユーザーを手動で削除するには、<u>Deleting an IAM user from your AWS account</u> を参照 してください。

関連付けられたオンプレミスインスタンスタグのみを手動で削除するには、<u>オンプレミスインスタン</u> <u>スから手動でオンプレミスインスタンスタグを削除します</u> を参照してください。 <u>deregister-on-premises-instance</u> コマンドを呼び出し、オンプレミスインスタンスを一意に識別 する名前を指定します (- - instance - name のオプションを用いて)。

aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX

オンプレミスインスタンスの登録解除後:

- コンソールにはすぐに表示されなくなります。
- 直後に同じ名前で別のインスタンスを作成できます。

CodeDeploy を用いてインスタンスの詳細の表示

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、デプロイで使用される インスタンスの詳細を表示できます。

インスタンスを見るための CodeDeploy API アクションについての情報について

は、<u>GetDeploymentInstance</u>、<u>ListDeploymentInstances</u>、および <u>ListOnPremisesInstances</u> を参照し てください。

トピック

- インスタンスの詳細の表示 (コンソール)
- インスタンスの詳細の表示 (CLI)

インスタンスの詳細の表示 (コンソール)

インスタンスの詳細を表示するには:

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。

i Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しましょ う。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレンス」の 「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを選択しま す。CodeDeploy は、これらのリージョンでのみサポートされています。

- 3. デプロイの詳細を表示するには、そのインスタンスのデプロイ ID を選択します。
- デプロイのページの [インスタンスアクティビティ] セクションですべてのインスタンスを表示で きます。
- 5. インスタンスの個別のデプロイライフサイクルイベントに関する詳細を表示するには、デプロイ の詳細ページの [Events] 列で、[View events] を選択します。

In the second secon

いずれかのライフサイクルイベントに対して [Failed] が表示された場合、インスタンス の詳細ページで、[View logs]、[View in EC2]、またはその両方を選択します。トラブル シューティングのヒントについては、「<u>インスタンスの問題のトラブルシューティング</u>」 を参照してください。

6. Amazon EC2 インスタンスに関するさらなる詳細情報を確認したい場合は、Instance ID 列でイン スタンスの ID を選択します。

インスタンスの詳細の表示 (CLI)

を使用してインスタンスの詳細 AWS CLI を表示するには、 get-deployment-instance コマン ドまたは list-deployment-instances コマンドを呼び出します。

1 つのインスタンスに関する詳細を表示するには、<u>get-deployment-instance</u> コマンドを呼び出し、 以下を指定します。

- ・ 一意のデプロイ ID。デプロイ ID を取得するには、list-deployments コマンドを呼び出します。
- 一意のインスタンス ID。インスタンス ID を取得するには、<u>list-deployment-instances</u> コマンドを 呼び出します。

デプロイ中で使用されるインスタンスの ID のリストを表示するには、<u>list-deployment-instances</u> コ マンドを呼び出し、以下を指定します:

- 一意のデプロイ ID。デプロイ ID を取得するには、list-deployments コマンドを呼び出します。
- オプションで、デプロイの状態別に特定のインスタンス ID のみを含めるかどうか。(指定しない場合、一致するすべてのインスタンス ID が、デプロイの状態にかかわらず表示されます。)

CodeDeploy インスタンスのヘルス

CodeDeploy は、デプロイグループのインスタンスのヘルスステータスをモニタリングします。正常 なインスタンスの数が、デプロイ中にデプロイグループに指定された正常なインスタンスの最小数を 下回ると、デプロイが失敗します。例えば、インスタンスの 85% がデプロイ中に正常な状態を維持 する必要があり、デプロイグループに 10 個のインスタンスが含まれている場合、1 つのインスタン スへのデプロイが失敗したときであっても、デプロイ全体が失敗します。これは、1 つのインスタン スがオフラインになり、最新のアプリケーションリビジョンをインストールできるようになった時 点で、正常に使用できるインスタンス数は既に 90% に低下しているためです。失敗したインスタン スと別のオフラインインスタンスを合わせると、正常に使用できるインスタンスの 80% が正常であ り、利用できることを意味します。CodeDeploy は全体的なデプロイに失敗します。

デプロイ全体が成功するには、以下が満たされている必要があります。

- CodeDeploy は、デプロイ中の各インスタンスにデプロイできます。
- ・ 少なくとも1つのインスタンスへのデプロイに成功する。つまり、デプロイ全体が成功するには、最小限の正常なホストの値が0の場合であっても、少なくとも1つ以上のインスタンスへのデプロイに成功する必要があります(1つ以上のインスタンスが正常)。

トピック

- [Health status] (ヘルスステータス)
- 正常なインスタンスの最小数について
- <u>アベイラビリティーゾーンあたりの正常なインスタンスの最小数について</u>

[Health status] (ヘルスステータス)

CodeDeploy は、各インスタンスに 2 つのヘルスステータス値 (revision health と instance health) を 割り当てます。

インスタンスの状態

リビジョンの状態

リビジョンの状態は、現在インスタンスにインストールされているアプリケーションリビジョン に基づきます。以下のステータス値があります。

- Current: インスタンスにインストールされているリビジョンは、デプロイグループの前回成功 したデプロイのリビジョンに一致します。
- Old: インスタンスにインストールされているリビジョンは、アプリケーションの以前のバージョンと一致します。
- Unknown: アプリケーションリビジョンはインスタンスに正常にインストールされていません。

インスタンスの状態

インスタンスの状態は、インスタンスへのデプロイが成功したかどうかに基づきます。以下の値 があります。

- Healthy: インスタンスへの前回のデプロイは成功しました。
- Unhealthy: インスタンスへのリビジョンのデプロイの試みは失敗したか、リビジョンがインス タンスにデプロイされていません。

CodeDeploy は、リビジョンのヘルスとインスタンスのヘルスを使用して、次の順序でデプロイグ ループのインスタンスにデプロイをスケジュールします。

- 1. インスタンスの状態が Unhealthy。
- 2. リビジョンの状態が Unknown。
- 3. リビジョンの状態が Old。
- 4. リビジョンの状態が Current。

デプロイ全体が成功すると、リビジョンは更新され、デプロイグループのヘルスステータスの値が更 新されて、最新のデプロイを反映します。

- デプロイに成功したすべての最新のインスタンスは current のままになります。それ以外の場合は、unknown になります。
- デプロイに成功したすべての古いインスタンスまたは不明なインスタンスは current になります。
 それ以外の場合は、old または unknown のままになります。
- デプロイに成功したすべての正常なインスタンスは healthy のままになります。それ以外の場合は、unhealthy になります。

 デプロイに成功したすべての異常なインスタンスは healthy になります。それ以外の場合 は、unhealthy のままになります。

全体的なデプロイが失敗するか、停止している場合:

- CodeDeploy がアプリケーションリビジョンのデプロイを試みた各インスタンスでは、そのインス タンスのデプロイの試みが成功したか失敗したかによって、そのインスタンスの状態が healthy ま たは unhealthy に設定されています。
- CodeDeploy がアプリケーションリビジョンのデプロイを試みなかった各インスタンスは、現在の インスタンスのヘルスの値を保持します。
- デプロイグループのリビジョンに変更はありません。

正常なインスタンスの最小数について

正常なインスタンスに必要な最小数は、デプロイ設定の一部として定義されます。

▲ Important

Blue/Green デプロイ中に、デプロイ設定と最小限の正常なホストの値は、元の環境ではなく 置き換え先環境のインスタンスに適用されます。ただし、元の環境のインスタンスがロード バランサーから登録解除されている場合、1 つの元のインスタンスが正常な登録解除に失敗 したときであっても、全体的なデプロイは失敗とマークされます。

CodeDeploy は、最小限の正常なホストの値を一般的に使用した 3 つのデフォルトのデプロイ構成を 提供します。

デフォルトのデプロイ設定名	事前定義された最小限の正常なホストの値
CodeDeployDefault.OneAtATime	1
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

デフォルトのデプロイ設定の詳細は、「CodeDeploy でデプロイ設定を使用する」で参照できます。

CodeDeploy 中でカスタムデプロイ設定を作成して、独自の最小限の正常なホストの値を定義できます。これらの値は、次のオペレーションを使用して整数または割合 (%) として定義できます。

- minimum-healthy-hosts で create-deployment-config コマンドを使う時の AWS CLIとして。
- CodeDeploy APIの MinimumHealthyHosts データタイプでの Value として。
- AWS CloudFormation テンプレートで <u>AWS::CodeDeploy::DeploymentConfig</u> MinimumHealthyHostsを使用する場合と同じです。

CodeDeploy では、2 つの主な目的のために、デプロイに対して正常なインスタンスの最小数を指定 できます。

- 全体的なデプロイの成功または失敗を判断する。アプリケーションリビジョンが少なくとも最小数の正常なインスタンスに正しくデプロイされた場合、デプロイは成功します。
- デプロイが続行するためにデプロイ中に正常である必要があるインスタンス数を決定する。

デプロイグループの正常なインスタンスの最小数は、インスタンス数、または合計インスタンス数の 割合 (%) として指定できます。パーセンテージを指定する場合、CodeDeploy はデプロイの開始時に パーセンテージを同等のインスタンス数に変換し、端数がある場合は切り上げます。

CodeDeploy は、デプロイプロセス中にデプロイグループのインスタンスのヘルスステータスを追跡 し、デプロイの指定された正常なインスタンスの最小数を使用して、デプロイを続行するかどうか判 断します。基本的な原則は、デプロイによって、正常なインスタンスの数が、指定した最小数を下 回ってはならないということです。このルールの1つの例外は、デプロイグループの正常なインス タンスの数が、指定した最小数より最初から少ない場合です。その場合、デプロイプロセスによって それ以上正常なインスタンスの数が減ることはありません。

Note

CodeDeploy は現在停止状態にあるものも含めて、デプロイグループ中のすべてのインスタ ンスへのデプロイを試みます。最小限の正常なホストの計算では、停止中のインスタンス は失敗したインスタンスと同じ影響を与えます。停止中のインスタンスが多すぎるために失 敗したデプロイを解決するには、インスタンスを再起動するか、タグを変更してデプロイグ ループから除外します。

CodeDeploy は、アプリケーションリビジョンをデプロイグループの異常なインスタンスにデプロイ するよう試みて、デプロイプロセスを開始します。デプロイに成功するたびに、CodeDeploy はイン スタンスのヘルスステータスを「正常」に変更し、それをデプロイグループの正常なインスタンスに 追加します。それで、CodeDeploy は正常なインスタンスの現在の数を、正常なインスタンスの指定 された最小数と比較します。

- 正常なインスタンスの数が、正常なインスタンスの指定された最小数以下である場合、CodeDeploy はデプロイをキャンセルし、より多くのデプロイによって正常なインスタンスの数が減らないようにします。
- 正常なインスタンスの数が指定された正常なインスタンスの最小数よりも少なくとも1つ大きい 場合、CodeDeployはアプリケーションのリビジョンを正常なインスタンスの元のセットにデプロ イします。

正常なインスタンスへのデプロイが失敗した場合、CodeDeploy はそのインスタンスのヘルスステー タスを unhealthy に変更します。デプロイが進行するにつれて、CodeDeploy は正常なインスタンス の現在の数を更新し、正常なインスタンスの指定された最小数と比較します。デプロイプロセスのあ る時点で、正常なインスタンスの数が指定された最小数にまで減った場合、CodeDeploy はデプロイ を停止します。この方法により、次のデプロイが失敗し、正常なインスタンスの数が指定された最小 数を下回ることを回避できます。

Note

指定する正常なインスタンスの最小数が、デプロイグループの合計インスタンス数より少な いことを確認します。割合 (%) の値を指定する場合、切り上げられることを覚えておいてく ださい。それ以外の場合、デプロイが開始すると、正常なインスタンスの数は正常なインス タンスの指定された最小数以下となり、CodeDeploy によりデプロイ全体が直ちに失敗しま す。

また、CodeDeploy は指定された正常なインスタンスの最小数および正常なインスタンスの実際の数 を使用して、複数のインスタンスにアプリケーションリビジョンをデプロイするかどうかと、その方 法を決定します。デフォルトでは、CodeDeploy は正常なインスタンスの数が、指定された正常なイ ンスタンスの最小数を下回るリスクなく、可能な限り多くのインスタンスにアプリケーションリビ ジョンをデプロイします。

一度にデプロイする必要があるデプロイ先のインスタンス数を決定する場合、CodeDeploy は次の計 算を使用します。

[total-hosts] - [minimum-healthy-hosts] =

API バージョン 2014-10-06 397

[number-of-hosts-to-deploy-to-at-once]

以下に例を示します。

- デプロイグループに 10 個のインスタンスがあり、正常なインスタンスの最小数を 9 に設定した場合、CodeDeploy は一度に 1 つのインスタンスにデプロイします。
- デプロイグループに 10 個のインスタンスがあり、正常なインスタンスの最小数を 3 に設定した場合、CodeDeploy は最初のバッチで同時に 7 つのインスタンスにデプロイし、2 番目のバッチで残りの 3 つのインスタンスにデプロイします。
- デプロイグループに 10 個のインスタンスがあり、正常なインスタンスの最小数を 0 に設定した場合、CodeDeploy は同時に 10 個のインスタンスにデプロイします。

例

次の例では、10個のインスタンスがあるデプロイグループを前提としています。

正常なインスタンスの最小数:95%

CodeDeploy は正常なインスタンスの最小数を 10 個のインスタンスに切り上げます。これは正常なインスタンスの数と等しくなります。全体的なデプロイは、いずれのインスタンスにもリビジョンをデプロイすることなく、直ちに失敗します。

正常なインスタンスの最小数:9

CodeDeploy は一度に1個のインスタンスヘリビジョンをデプロイします。いずれかのインスタンスへのデプロイが失敗した場合、正常なインスタンスの数は正常なインスタンスの最小数に等しくなるため、CodeDeploy のデプロイ全体は直ちに失敗します。このルールの例外は、最後のインスタンスが失敗した場合でも、デプロイは引き続き成功することです。

CodeDeploy は、一度に 1 個のインスタンスずつ、デプロイが失敗するか、デプロイ全体が完了 するまで、デプロイを続行します。10 のデプロイすべてに成功した場合、デプロイグループには 10 個の正常なインスタンスが含まれます。

正常なインスタンスの最小数:8

CodeDeploy は一度に 2 個のインスタンスヘリビジョンをデプロイします。これらのうち 2 つの デプロイに失敗した場合、CodeDeploy は直ちにデプロイ全体を失敗します。このルールの例外 は、最後のインスタンスが 2 番目に失敗した場合でも、デプロイは成功することです。 正常なインスタンスの最小数:0

CodeDeploy は、一度にデプロイグループ全体ヘリビジョンをデプロイします。デプロイ全体が 成功するには、インスタンスに対して、少なくとも1つ以上のデプロイが成功する必要がありま す。正常なインスタンスが0の場合、デプロイは失敗します。これは、デプロイ全体を成功とし てマークするには、正常なインスタンスの最小値が0であっても、デプロイ全体の完了時に1つ 以上のインスタンスが正常であることが要件であるためです。

アベイラビリティーゾーンあたりの正常なインスタンスの最小数について

Note

このセクションでは、Amazon EC2 インスタンスを表す用語としてインスタンスとホストを 同じ意味で使用しています。

複数の<u>アベイラビリティーゾーン</u>のインスタンスにデプロイする場合は、オプションで<u>zonal</u> <u>configuration</u>機能を有効にすると、CodeDeploy は一度に 1 つのアベイラビリティーゾーンにデプロ イできます。

この機能を有効にすると、CodeDeploy は正常なホストの数が「ゾーンあたりの正常なホストの最小 数」および「正常なホストの最小数」の値を常に超えていることを確認します。正常なホストの数が いずれかの値を下回ると、CodeDeploy はすべてのアベイラビリティーゾーンでデプロイに失敗しま す。

ー度にデプロイする先のホストの数を計算するために、CodeDeploy は「ゾーンあたりの正常なホストの最小数」と「正常なホストの最小数」の値の両方を使用します。CodeDeploy では、[A] と [B] (以下の [A] と [B]) の計算結果の小さい方を使用します。

一度にデプロイする先のホストの数を決定すると、CodeDeploy はその数のホストに (一度に 1 つの アベイラビリティーゾーンずつ) バッチでデプロイします。オプションでゾーン間に一時停止(また は「ベイク時間」)を挟みます。

アベイラビリティーゾーンあたりの正常なインスタンスの最小数について

例

デプロイが以下のように設定されている場合:

- ・ [total-hosts] は 200
- ・ [minimum-healthy-hosts] は 160
- ・ [total-hosts-per-AZ] は 100
- [minimum-healthy-hosts-per-AZ] は 50

結果…

- [A] = 200 160 = 40
- [B] = 100 50 = 50
- 40 は 50 より少ない

したがって、CodeDeploy は一度に 40 ホストにデプロイします。

このシナリオでは、次のようにデプロイされます。

- 1. CodeDeploy は最初のアベイラビリティーゾーンにデプロイします。
 - a. CodeDeploy は最初の 40 ホストにデプロイします。
 - b. CodeDeploy は次の 40 ホストにデプロイします。
 - c. CodeDeploy は残りの 20 ホストにデプロイします。

これで、1 つ目のアベイラビリティーゾーンへのデプロイが完了しました。

- 2. (オプション) CodeDeploy は、[監視期間] または [最初のゾーン監視期間を追加] の設定で定義さ れているように、最初のゾーンへのデプロイが「ベイク」されるまで待機します。問題がなけれ ば、CodeDeploy は続行します。
- 3. CodeDeploy は 2 番目のアベイラビリティーゾーンにデプロイします。
 - a. CodeDeploy は最初の 40 ホストにデプロイします。
 - b. CodeDeploy は次の 40 ホストにデプロイします。
 - c. CodeDeploy は残りの 20 ホストにデプロイします。

これで、2番目かつ最後のアベイラビリティーゾーンへのデプロイが完了しました。

ゾーン設定機能の詳細と、アベイラビリティーゾーンあたりの正常なインスタンスの最小数を指定す る方法については、「<u>zonal configuration</u>」を参照してください。

CodeDeploy でデプロイ設定を使用する

デプロイ設定とは、デプロイ中に CodeDeploy が使用する一連のルール、成功条件、および失敗条件です。これらのルールや条件は、EC2/オンプレミスのコンピューティングプラットフォーム、 AWS Lambda コンピューティングプラットフォーム、または Amazon ECS コンピューティングプ ラットフォームのいずれかにデプロイするかによって、異なります。

EC2/オンプレミスコンピューティングプラットフォームのデプロ イ設定

EC2/オンプレミスコンピューティングプラットフォームにデプロイする場合、デプロイ設定では、 「正常なホストの最小数」とオプションの「ゾーンあたりの正常なホストの最小数」の値を使用す ることで、デプロイ中のどの時点でも使用可能でなければならないインスタンスの数または割合 (%) を指定します。

が提供する 3 つの事前定義されたデプロイ設定のいずれかを使用する AWS か、カスタム デプロイ設定を作成できます。カスタムデプロイ設定の作成の詳細については、「<u>Create a</u> <u>Deployment Configuration</u>」を参照してください。デプロイ設定を指定しない場合、CodeDeploy は、CodeDeployDefault.OneAtATime デプロイ設定を使用します。

CodeDeploy がデプロイ中にインスタンスのヘルス状態をモニタリングおよび評価する方法の詳細に ついては、「<u>Instance Health</u>」を参照してください。 AWS アカウントに既に登録されているデプロ イ設定のリストを表示するには、「」を参照してくださいView Deployment Configuration Details。

EC2/オンプレミスコンピューティングプラットフォームの事前定義された デプロイ設定

次の表は、定義済みのデプロイ設定を一覧表示します。

Note

<u>zonal configuration</u> 機能 (アベイラビリティーゾーンあたりの正常なホストの数を指定でき る機能) をサポートする定義済みのデプロイ設定はありません。この機能を使用する場合 は、独自のデプロイ設定を作成する必要があります。

デプロイ設定	説明
CodeDeployDefault.AllAtOnce	インプレースデプロイ: 一度に可能な限り多くのインスタンスへアプリ ケーションリビジョンをデプロイするよう試み ます。アプリケーションリビジョンが1つ以 上のインスタンスにデプロイされる場合、デプ ロイ全体のステータスはSucceeded として表 示されます。アプリケーションリビジョンがい ずれのインスタンスにもデプロイされない場 合、デプロイ全体のステータスは Failed とし て表示されます。9つのインスタンスの例を使 い、CodeDeployDefault.AllAtOnce は、一度に 9つのインスタンスすべてにデプロイするよう 試みます。インスタンスへのデプロイが1つで も成功すると、デプロイ全体は成功します。9 つすべてのインスタンスへのデプロイが失敗し た場合に限り失敗します。
	ブルー/グリーンデプロイ
	 ・置き換え先環境へのデプロイ:インプレース デプロイの CodeDeployDefault.AllAtOnce と 同じデプロイルールに従います。 トラフィックの再ルーティング:置き換え先 環境のすべてのインスタンスに一度にトラ フィックをルーティングします。トラフィッ クが少なくとも1つのインスタンスに正常に 再ルーティングされた場合が成功です。すべ てのインスタンスへの再ルーティングが失敗 した時点で失敗です。
CodeDeployDefault.HalfAtATime	インプレースデプロイ:
	一度に最大半分のインスタンスにデプロイし ます (端数は切り捨てられます)。デプロイ全体 は、アプリケーションリビジョンが少なくとも

デプロイ設定

説明

半分のインスタンスにデプロイされた場合は成 功です (端数は切り捨てられます)。それ以外の 場合、デプロイは失敗です。9 つのインスタン スの例では、4 つまでのインスタンスに同時に デプロイされます。デプロイ全体は5 つ以上 のインスタンスへのデプロイが成功した場合は 成功です。それ以外の場合、デプロイは失敗で す。

Note

複数の Auto Scaling グループのインス タンスにデプロイする場合、CodeDe ploy は属する Auto Scaling グループに 関係なく、一度に最大半分のインスタ ンスにデプロイします。例えば、Auto Scaling グループが ASG1 と ASG2 の 2 つあり、それぞれに 10 個のインスタ ンスがあるとします。このシナリオで は、CodeDeploy は ASG1 のみで 10 個 のインスタンスにデプロイできますが 、少なくとも半数のインスタンスにデ プロイされているため、成功と見なす ことができます。

ブルー/グリーンデプロイ

- ・置き換え先環境へのデプロイ: インプレース デプロイの CodeDeployDefault.HalfAtATime と同じデプロイルールに従います。
- トラフィックの再ルーティング:置き換え先 環境の最大半分のインスタンスに一度にトラ フィックをルーティングします。少なくとも 半分のインスタンスへの再ルーティングが成

デプロイ設定	説明
	功した場合が成功です。それ以外の場合、 は失敗します。

デプロイ設定	説明	
CodeDeployDefault.OneAtATime	インプレースデプロイ:	
	一度に 1 つのインスタンスにのみアプリケー ションリビジョンをデプロイします。	
	複数のインスタンスを含むデプロイグループの 場合。	
	 デプロイ全体はアプリケーションリビジョン がすべてのインスタンスへデプロイされた場 合、成功します。このルールの例外は、最後 のインスタンスへのデプロイが失敗した場合 に、デプロイ全体が成功することです。これ は CodeDeploy が一度に 1 つのインスタンス のみ CodeDeployDefault.OneAtATime 設定 でオフラインにするためです。 デプロイ全体はアプリケーションリビジョン が最後のインスタンス以外へのデプロイに失 敗すると、ただちに失敗します。 9 つのインスタンスを使用する例では、1 つ のインスタンスに同時にデプロイされます。 最初の 8 つのインスタンスへのデプロイが成 功すると、デプロイ全体は成功します。最初 の 8 つのインスタンスのいずれかへのデプ ロイが失敗すると、デプロイ全体は失敗しま す。 	
	1 つのインスタンスのみを含むデプロイグルー プでは、1 つのインスタンスへのデプロイが成 功した場合にのみ、デプロイは全体は成功しま す。	
	ブルー/グリーンデプロイ	

デプロイ設定	説明
	・置き換え先環境へのデプロイ: インプレース デプロイの CodeDeployDefault.OneAtATime と同じデプロイルールに従います。
	 トラフィックの再ルーティング:置き換え 先環境で一度に1つのインスタンスにトラ フィックをルーティングします。トラフィッ クがすべての置き換え先インスタンスに正常 に再ルーティングされた場合が成功です。最 初に再ルーティングが失敗した時点で失敗で す。このルールの例外は、最後のインスタン スが登録に失敗しても、デプロイ全体が成功 することです。

Amazon ECS コンピューティングプラットフォームのデプロイ設 定

Amazon ECS コンピューティングプラットフォームにデプロイする場合、デプロイ設定より、更新 された Amazon ECS タスクセットにトラフィックを移行する方法を指定します。canary、リニア、 または一度にすべてのデプロイ設定を使用してトラフィックをシフトできます。詳細については、 「デプロイ設定」を参照してください。

独自の Canary または線形のデプロイ設定を作成することもできます。詳細については、「<u>Create a</u> <u>Deployment Configuration」</u>を参照してください。

Amazon ECS コンピューティングプラットフォームの事前定義されたデプ ロイ設定

以下の表に、Amazon ECS のデプロイで利用できる事前定義された設定を一覧表示します。

Note

Network Load Balancer を使用する場合、CodeDeployDefault.ECSAllAtOnce 優先デプ ロイ設定のみがサポートされます。

デプロイ設定	説明
CodeDeployDefault.ECSLinear10Percent Every1Minutes	すべてのトラフィックが移行されるまで、毎分 トラフィックの 10 パーセントを移行します。
CodeDeployDefault.ECSLinear10Percent Every3Minutes	すべてのトラフィックが移行されるまで、3 分 ごとにトラフィックの 10 パーセントを移行し ます。
CodeDeployDefault.ECSCanary10Percent 5Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 5 分後に デプロイされます。
CodeDeployDefault.ECSCanary10Percent 15Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 15 分後 にデプロイされます。
CodeDeployDefault.ECSAllAtOnce	すべてのトラフィックを同時に更新済み Amazon ECS コンテナに移行します。

AWS CloudFormation blue/green デプロイのためのデプロイ設定 (Amazon ECS)

Blue AWS CloudFormation /Green デプロイを介して Amazon ECS コンピューティングプラット フォームにデプロイする場合、デプロイ設定は、更新された Amazon ECS コンテナにトラフィック を移行する方法を指定します。canary、リニア、または一度にすべてのデプロイ設定を使用してト ラフィックをシフトできます。詳細については、「デプロイ設定」を参照してください。

Blue AWS CloudFormation /Green デプロイでは、独自のカスタム Canary または線形デプロイ設定 を作成することはできません。 AWS CloudFormation を使用して Amazon ECS ブルー/グリーンデ プロイを管理するstep-by-stepについては、AWS CloudFormation 「 ユーザーガイド」の「 <u>を使用し</u> <u>た CodeDeploy による ECS ブルー/グリーンデプロイの自動化 AWS CloudFormation</u>」を参照してく ださい。 Note

を使用した Amazon ECS ブルー/グリーンデプロイの管理 AWS CloudFormation は、欧州 (ミラノ)、アフリカ (ケープタウン)、アジアパシフィック (大阪) リージョンでは利用でき ません。

AWS Lambda コンピューティングプラットフォームのデプロイ設 定

AWS Lambda コンピューティングプラットフォームにデプロイする場合、デプロイ設定は、 アプリケーションの新しい Lambda 関数バージョンにトラフィックを移行する方法を指定しま す。canary、リニア、または一度にすべてのデプロイ設定を使用してトラフィックをシフトできま す。詳細については、「<u>デプロイ設定</u>」を参照してください。

独自の Canary または線形のデプロイ設定を作成することもできます。詳細については、「<u>Create a</u> Deployment Configuration」を参照してください。

AWS Lambda コンピューティングプラットフォームの事前定義されたデプ ロイ設定

以下の表に、 AWS Lambda のデプロイで利用できる事前定義された設定を一覧表示します。

デプロイ設定	説明
CodeDeployDefault.LambdaCanary10Perc ent5Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 5 分後に デプロイされます。
CodeDeployDefault.LambdaCanary10Perc ent10Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 10 分後 にデプロイされます。
CodeDeployDefault.LambdaCanary10Perc ent15Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 15 分後 にデプロイされます。

デプロイ設定を作成	成する
-----------	-----

デプロイ設定	説明
CodeDeployDefault.LambdaCanary10Perc ent30Minutes	最初の増分でトラフィックの 10 パーセントを 移行します。残りの 90 パーセントは 30 分後 にデプロイされます。
CodeDeployDefault.LambdaLinear10Perc entEvery1Minute	すべてのトラフィックが移行されるまで、毎分 トラフィックの 10 パーセントを移行します。
CodeDeployDefault.LambdaLinear10Perc entEvery2Minutes	すべてのトラフィックが移行されるまで、2 分 ごとにトラフィックの 10 パーセントを移行し ます。
CodeDeployDefault.LambdaLinear10Perc entEvery3Minutes	すべてのトラフィックが移行されるまで、3 分 ごとにトラフィックの 10 パーセントを移行し ます。
CodeDeployDefault.LambdaLinear10Perc entEvery10Minutes	すべてのトラフィックが移行されるまで、10 分ごとにトラフィックの 10 パーセントを移行 します。
CodeDeployDefault.LambdaAllAtOnce	すべてのトラフィックは、更新された Lambda 関数に一度に移行します。

トピック

- <u>Create a Deployment Configuration</u>
- View Deployment Configuration Details
- Delete a Deployment Configuration

CodeDeploy を使用してデプロイ設定を作成する

CodeDeploy に用意してあるデフォルトのデプロイ設定のいずれも使用しない場合は、次の手順を実行して独自のデプロイ設定を作成できます。

CodeDeploy コンソール AWS CLI、CodeDeploy APIs、または AWS CloudFormation テンプレート を使用して、カスタムデプロイ設定を作成できます。 AWS CloudFormation テンプレートを使用してデプロイ設定を作成する方法については、「」を参照 してくださいAWS CloudFormation CodeDeploy リファレンスの テンプレート。

トピック

- デプロイ設定の作成 (コンソール)
- CodeDeploy を使用してデプロイ設定を作成する (AWS CLI)

デプロイ設定の作成 (コンソール)

AWS コンソールを使用してデプロイ設定を作成するには、次の手順を実行します。

コンソールを使用して CodeDeploy でデプロイ設定を作成するには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeployの開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ設定] を選択します。

組み込みのデプロイ設定のリストが表示されます。

- 3. [Create deployment configuration (デプロイ設定の作成)] を選択します。
- [デプロイ設定名] に、デプロイ設定の名前を入力します。例えば、my-deployment-config と指定します。
- 5. [コンピューティングプラットフォーム] で、次のいずれかを選択します。
 - ・ EC2/オンプレミス
 - AWS Lambda
 - Amazon ECS
- 6. 次のいずれかを行います:
 - EC2/オンプレミスを選択した場合:
 - 1. [正常なホストの最小数] で、デプロイ中のどの時点でも使用可能でなければならないイン スタンスの数または割合を指定します。CodeDeploy がデプロイ中にインスタンスのヘルス

状態をモニタリングおよび評価する方法の詳細については、「<u>Instance Health</u>」を参照し てください。

 (オプション) [ゾーン設定] で、[ゾーン設定を有効にする] を選択すると、CodeDeploy は AWS リージョン内で一度に1つのアベイラビリティーゾーンにアプリケーションをデプロ イします。一度に1つのアベイラビリティーゾーンにデプロイすることで、デプロイのパ フォーマンスと実行可能性に対する信頼が高まるにつれて、デプロイを徐々に多くのユー ザーに公開できます。ゾーン設定を有効にしない場合、CodeDeploy はリージョン内のラン ダムに選択されたホストにアプリケーションをデプロイします。

ゾーン設定機能を有効にする場合は、次の点に注意してください。

- ゾーン設定機能は、Amazon EC2 インスタンスへのインプレースデプロイでのみサポートされます (ブルー/グリーンデプロイやオンプレミスインスタンスはサポートされません)。インプレイスデプロイの詳細については、「デプロイタイプ」を参照してください。
- ゾーン設定機能は、<u>事前定義済みのデプロイ設定</u>ではサポートされません。ゾーン設定 を使用するには、ここで説明しているように、カスタムデプロイ設定を作成する必要が あります。
- CodeDeployは、デプロイをロールバックする必要がある場合、ランダムなホストにロー ルバック操作を実行します (CodeDeployは、想定とは異なり、一度に1つのゾーンを順 にロールバックしません)。このロールバック動作は、パフォーマンス上の理由から選択 されています。ロールバックの詳細については、「CodeDeployを使用した再デプロイお よびデプロイのロールバック」を参照してください。
- [ゾーン設定を有効にする] チェックボックスをオンにした場合は、オプションで以下のオ プションを指定します。
 - ・ (オプション) [監視期間] で、CodeDeploy が 1 つのアベイラビリティーゾーンへのデプロ イを完了してから待機する時間を秒単位で指定します。CodeDeploy は、次のアベイラビ リティーゾーンへのデプロイを開始する前に、この時間だけ待機します。デプロイが 1 つのアベイラビリティーゾーンで完了して次のゾーンに進む前に、一定時間を確保して 完了を実証 (ベイク) する場合は、監視期間を追加することを検討します。監視期間を指 定しない場合、CodeDeploy は次のアベイラビリティーゾーンへのデプロイをすぐに開始 します。[監視期間] 設定の仕組みについては、「<u>アベイラビリティーゾーンあたりの正</u> <u>常なインスタンスの最小数について</u>」を参照してください。
 - (オプション) [最初のゾーンに監視期間を追加] を選択すると、最初のアベイラビリティー ゾーンにのみ監視期間が設定されます。このオプションは、1 つ目のアベイラビリ ティーゾーンのベイク時間を確保する場合に設定できます。[最初のゾーンの監視期間を

追加] に値を指定しない場合、CodeDeploy は最初のアベイラビリティーゾーンの [監視 期間] の値を使用します。

 (オプション) [ゾーンあたりの正常なホストの最小数] で、デプロイ中に使用可能でなけれ ばならない、アベイラビリティーゾーンあたりのインスタンスの数または割合を指定し ます。FLEET_PERCENT を選択してパーセンテージを指定するか、HOST_COUNT を選 択して数値を指定します。このフィールドは [正常なホストの最小数] フィールドと連動 します。詳細については、「<u>アベイラビリティーゾーンあたりの正常なインスタンスの</u> 最小数について」を参照してください。

[ゾーンあたりの正常なホストの最小数] に値を指定しない場合、CodeDeploy はデフォル ト値の 0 パーセントを使用します。

- [AWS Lambda] または [Amazon ECS] を選択した場合:
 - 1. [タイプ]] で、[リニア] または [Canary] を選択します。
 - 2. [ステップ] フィールドと [間隔] フィールドで、次のいずれかを行います。
 - [Canary] を選択した場合は、[ステップ] に、移行するトラフィックのパーセンテージを 1~99 で入力します。この値は、最初の増分で移行されるトラフィックの割合を示しま す。残りのトラフィックは、2 回目の増分で、選択した間隔後に移行されます。

[間隔]に、1番目と2番目のトラフィック移行間の分数を入力します。

[リニア]を選択した場合は、[ステップ]に、移行するトラフィックのパーセンタージを1~99で入力します。この値は、各間隔の開始時に移行されるトラフィックの割合を示します。

[間隔]に、各増分移行間の分数を入力します。

7. [Create deployment configuration (デプロイ設定の作成)] を選択します。

これで、デプロイグループに関連付けることができるデプロイ設定が整いました。

CodeDeploy を使用してデプロイ設定を作成する (AWS CLI)

を使用してデプロイ設定 AWS CLI を作成するには、<u>create-deployment-config</u> コマンドを呼び出し ます。

次の例では、ThreeQuartersHealthy という名前の EC2/オンプレミスデプロイ設定を作成しま す。このデプロイ設定では、デプロイ中にターゲットインスタンスの 75% が常に正常であることが 要求されます。 aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy -- minimum-healthy-hosts type=FLEET_PERCENT,value=75

次の例では、300Tota150PerAZ という名前の EC2/オンプレミスデプロイ設定を作成します。こ のデプロイ設定では、デプロイあたり合計 300 個のターゲットインスタンス、およびアベイラビリ ティーゾーンあたり 50 個のターゲットインスタンスが常に正常であることが要求されます。また、 監視期間を 1 時間に設定します。

aws deploy create-deployment-config --deployment-config-name 300Total50PerAZ
 --minimum-healthy-hosts type=HOST_COUNT,value=300 --zonal-config
 '{"monitorDurationInSeconds":3600,"minimumHealthyHostsPerZone":
 {"type":"HOST_COUNT","value":50}}'

次の例では、 という名前の AWS Lambda デプロイ設定を作成しま すCanary25Percent45Minutes。この際、最初の増分でトラフィックの 25 パーセントを移行す る Canary トラフィックを使用します。残りの 75 パーセントは 45 分後に移行されます。

aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --

compute-platform Lambda

次の例では、Canary25Percent45Minutes という名前の Amazon ECS デプロイ設定を作成しま す。この際、最初の増分でトラフィックの 25 パーセントを移行する Canary トラフィックを使用し ます。残りの 75 パーセントは 45 分後に移行されます。

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
    --traffic-routing-config
```

"type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" -compute-platform ECS

CodeDeploy によるデプロイ設定の詳細の表示

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、 AWS アカウントに関連付けられたデプロイ設定の詳細を表示できます。事前定義された CodeDeploy デプロイ設定の説

明については、「<u>EC2/オンプレミスコンピューティングプラットフォームの事前定義されたデプロ</u> イ設定」を参照してください。

トピック

- デプロイ設定の詳細の表示 (コンソール)
- デプロイ設定の表示 (CLI)

デプロイ設定の詳細の表示 (コンソール)

CodeDeploy コンソールを使用して、デプロイ設定名の一覧を表示するには:

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「<u>CodeDeploy の開始方法</u>」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ] を展開し、[デプロイ設定] を選択します。

コンソールで、各デプロイ設定のデプロイ設定名および条件の一覧を表示できます。

Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しましょ う。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレンス」の 「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを選択しま す。CodeDeploy は、これらのリージョンでのみサポートされています。

デプロイ設定の表示 (CLI)

を使用してデプロイ設定の詳細 AWS CLI を表示するには、 get-deployment-config コマンドま たは list-deployment-configs コマンドを呼び出します。

1 つのデプロイ設定に関する詳細を表示するには、<u>get-deployment-config</u> コマンドを呼び出し、一意 のデプロイ設定名を指定します。

複数のデプロイ設定に関する詳細を表示するには、list-deployments コマンドを呼び出します。

CodeDeploy を使用してデプロイ設定を削除する

AWS CLI または CodeDeploy APIs を使用して、 AWS アカウントに関連付けられたカスタムデプロ イ設定を削除できま

す。CodeDeployDefault.AllAtOnce、CodeDeployDefault.HalfAtATime、CodeDeployDefault などの組み込みのデプロイ設定は削除できません。

▲ Warning

まだ使用中のカスタムデプロイ設定は削除できません。未使用のカスタムデプロイ設定を削 除すると、それを新しいデプロイおよび新規デプロイグループに関連付けることはできなく なります。このアクションを元に戻すことはできません。

を使用してデプロイ設定 AWS CLI を削除するには、<u>delete-deployment-config</u> コマンドを呼び 出し、デプロイ設定名を指定します。デプロイ設定名のリストを表示するには、<u>list-deployment-</u> configs コマンドを呼び出します。

次の例では、ThreeQuartersHealthyという名前のデプロイ設定を削除します。

aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy

CodeDeploy 内でアプリケーションを用いて作業をする

インスタンスを設定したら、リビジョンをデプロイする前に CodeDeploy でアプリケーションを作 成する必要があります。アプリケーション は、CodeDeploy によって使用される単なる名称もしく はコンテナであり、その目的は正しいリビジョン、デプロイ設定、およびデプロイグループが、デプ ロイ中に正しく参照されることにあります。

次のステップ用に、以下の表の情報を使用します。

コンピューティングプラット フォーム	シナリオ	次のステップの情報
EC2/オンプレミス	まだインスタンスを作成して いません。	「 <u>CodeDeploy のためにイン</u> <u>スタンスを用いた操作</u> 」を 参照してから、このページに 戻ってください。
EC2/オンプレミス	インスタンスは作成しました が、タグ付けが完了していま せん。	「 <u>Tagging Instances for</u> <u>Deployments</u> 」を参照してか ら、このページに戻ってくだ さい。
EC2/オンプレミス, AWS Lambda, および Amazon ECS	まだアプリケーションを作成 していません。	「 <u>CodeDeploy でアプリケー</u> <u>ションを作成する</u> 」を参照し てください。
EC2/オンプレミス, AWS Lambda, および Amazon ECS	既にアプリケーションを作成 しましたが、まだデプロイグ ループを作成していません。	「 <u>CodeDeploy でデプロイグ</u> <u>ループを作成する</u> 」を参照し てください。
EC2/オンプレミス, AWS Lambda, および Amazon ECS	既にアプリケーションとデプ ロイグループを作成しまし たが、アプリケーションリビ ジョンを作成していません。	「 <u>CodeDeploy のアプリケー</u> <u>ションリビジョンの操作</u> 」を 参照してください。
EC2/オンプレミス, AWS Lambda, および Amazon ECS	既にアプリケーションとデプ ロイグループを作成し、ア プリケーションリビジョンを	「 <u>CodeDeploy でデプロイを</u> <u>作成する</u> 」を参照してくださ い。

コンピューティングプラット フォーム	シナリオ	次のステップの情報
	アップロードしました。デプロイの進備が敷いました	

トピック

- CodeDeploy でアプリケーションを作成する
- CodeDeploy を使用してアプリケーション詳細を表示する
- 通知ルールの作成
- CodeDeploy アプリケーション名の変更
- CodeDeploy でアプリケーションを削除する

CodeDeploy でアプリケーションを作成する

アプリケーション は、CodeDeploy によって使用される単なる名称もしくはコンテナであり、そ の目的は正しいリビジョン、デプロイ設定、およびデプロイグループが、デプロイ中に正しく参 照されることにあります。CodeDeploy コンソール、 AWS CLI、CodeDeploy APIs、または AWS CloudFormation テンプレートを使用してアプリケーションを作成できます。

コードまたはアプリケーションリビジョンは、デプロイと呼ばれるプロセスを通してインスタンスに インストールされます。CodeDeploy では、2 種類のデプロイがサポートされます。

- インプレイスデプロイ: デプロイグループの各インスタンス上のアプリケーションが停止され、最新のアプリケーションリビジョンがインストールされて、新バージョンのアプリケーションが開始され検証されます。ロードバランサーを使用し、デプロイ中はインスタンスが登録解除され、デプロイ完了後にサービスに復元されるようにできます。EC2 オンプレミスコンピューティングプラットフォームを使用するデプロイのみが、インプレイスデプロイを使用できます。インプレイスデプロイの詳細については、「インプレースデプロイの概要」を参照してください。
- Blue/Green デプロイ: デプロイの動作は、使用するコンピューティングプラットフォームにより異なります。
 - EC2 オンプレミスコンピューティングプラットフォームの Blue/Green: 以下のステップを使用して、デプロイグループのインスタンス (元の環境) がインスタンスの別のセット (置き換え先環境) に置き換えられます。
 - 置き換え先の環境のインスタンスがプロビジョニングされます。
- 最新のアプリケーションリビジョンは、置き換え先インスタンスにインストールされます。
- オプションの待機時間は、アプリケーションのテストやシステム検証などのアクティビティに 対して発生します。
- ・置き換え先環境のインスタンスは、1 つまたは複数の Elastic Load Balancing ロードバラン サーに登録され、トラフィックは、それらに再ルーティングされます。元の環境のインスタン スは、登録が解除され、終了するか、他の使用のために実行することができます。

Note

EC2/オンプレミスのコンピューティングプラットフォームを使用する場合は、blue/ green デプロイが Amazon EC2 インスタンスでのみ機能することに注意してください。

- AWS Lambda または Amazon ECS コンピューティングプラットフォームの Blue/Green: トラ フィックは、Canary、線形、または all-at-onceデプロイ設定に従って増分でシフトされます。
- 経由のブルー/グリーンデプロイ AWS CloudFormation: スタック AWS CloudFormation の更新の 一環として、トラフィックが現在のリソースから更新されたリソースに移行されます。現時点で は、ECS blue/green デプロイのみがサポートされています。

ブルー/グリーンデプロイの詳細については、「<u>Blue/Green デプロイの概要</u>」を参照してください。

CodeDeploy コンソールを使用してアプリケーションを作成する場合は、最初のデプロイグループを 同時に設定します。を使用してアプリケーション AWS CLI を作成する場合は、別のステップで最初 のデプロイグループを作成します。

AWS アカウントに既に登録されているアプリケーションのリストを表示するには、「」を参照し てください<u>CodeDeploy を使用してアプリケーション詳細を表示する</u>。 AWS CloudFormation テン プレートを使用してアプリケーションを作成する方法については、「」を参照してください<u>AWS</u> CloudFormation CodeDeploy リファレンスの テンプレート。

デプロイタイプはいずれも、どの送信先にも適用されません。以下の表に、3 種類のデプロイ送信先 のデプロイを指定するデプロイタイプを示します。

デプロイ送信先	インプレース	Blue/Green
Amazon EC2	あり	あり

デプロイ送信先	インプレース	Blue/Green
オンプレミス	はい	いいえ
サーバーレス AWS Lambda 関数	いいえ	はい
Amazon ECS アプリケーショ ン	いいえ	はい

トピック

- インプレースデプロイ (コンソール) 用のアプリケーションを作成
- ・ Blue/Green デプロイ (コンソール) のアプリケーションを作成します。
- ・ Amazon ECS サービスデプロイ用のアプリケーションを作成 (コンソール)
- ・ AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール)
- ・ <u>アプリケーションの作成 (CLI)</u>

インプレースデプロイ (コンソール) 用のアプリケーションを作成

CodeDeploy コンソールを使用して、インプレースデプロイ用のアプリケーションを作成するには

▲ Warning

次の場合は、これらの手順を実行しないでください。

- インスタンスを CodeDeploy デプロイで使用する準備ができていません。インスタンスを セットアップするには、CodeDeployのためにインスタンスを用いた操作の指示に従い、 その後にこのトピックの手順に従います。
- カスタムデプロイ設定を使用するアプリケーションを作成する必要があり、まだデプロイ 設定を作成していません。Create a Deployment Configuration の指示に従った後に、この トピックの手順に従います。
- 最低限必要な信頼およびアクセス権限を持つ CodeDeploy を信頼するサービスロールがありません。必要なアクセス許可を持つサービスロールを作成し、設定するには、「ステッ

- <u>プ 2: CodeDeployのサービスのロールを作成する</u>」の手順に従って、このトピックの手順 に戻ります。
- インプレースデプロイのために Elastic Load Balancing で、Classic Load Balancer、Application Load Balancer、または Network Load Balancer を選択したいが、ま だ作成していない場合。

CodeDeploy コンソールを使用して、インプレースデプロイ用のアプリケーションを作成するには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

CodeDeployの開始方法で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、[デプロイ]、[ご利用開始にあたって]の順に選択します。
- 3. [Create application] を選択します。
- 4. [アプリケーション名] に、アプリケーションの名前を入力します。
- 5. [Compute platform (コンピューティングプラットフォーム)] で [EC2/On-Premises (EC2/オンプレミス)] を選択します。
- 6. [Create application] を選択します。
- 7. アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択しま す。
- 8. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されている設定 (デプロイグループ名、タグ、Amazon EC2 Auto Scaling グループ名、または両方、およびデプロイ設定を含む) を使用する場合は、 このページでこれらの設定を指定します。この新しいデプロイグループと既存のデプ ロイグループは名前が同じでも、それぞれが別のアプリケーションに関連付けられるた め、CodeDeploy では別のデプロイグループとして扱われます。

9. [サービスロール] で、ターゲットインスタンスへのアクセス権を CodeDeploy に付与するサービ スロールを選択します。

- 10. [デプロイタイプ] で、[インプレース] を選択します。
- 11. [環境設定] で、以下のいずれかを選択します。
 - a. Amazon EC2 Auto Scaling グループ: アプリケーションリビジョンをデプロイする Amazon EC2 Auto Scaling グループの名前を入力または選択します。Amazon EC2 Auto Scaling グループの一部として新しい Amazon EC2 インスタンスが起動されると、CodeDeploy では リビジョンを新しいインスタンスに自動的にデプロイできます。デプロイグループには最大 10 個の Amazon EC2 Auto Scaling グループを追加できます。
 - b. [Amazon EC2 インスタンス] または [オンプレミスインスタンス]: [キー] と [値] フィールド
 に、インスタンスにタグを付けるために使用したキーバリューペアの値を入力します。単一
 タググループで最大 10 個のキーと値のペアをタグ付けできます。
 - i. [値] フィールドでワイルドカードを使用して、似ている Amazon EC2 インスタンス、 コストセンター、グループ名などの特定のパターンでタグ付けされているすべてのイン スタンスを識別できます。例えば、キー フィールドに 名前 を選択し、値 フィールド に GRP-*a を入力すると、CodeDeploy は GRP-1a、GRP-2a、および GRP-XYZ-a な どそのパターンに当てはまるすべてのインスタンスを特定します。
 - ii. [値] フィールドでは、大文字と小文字が区別されます。
 - iii. リストからキーと値のペアを削除するには、[タグの削除]を選択します。

CodeDeploy で、指定されたキーバリューの各ペアまたは Amazon EC2 Auto Scaling グ ループ名に一致するインスタンスが検出されると、一致したインスタンスの数が表示されま す。インスタンスに関する詳細情報を表示するには、その数をクリックします。

インスタンスへのデプロイの条件をさらに絞り込むには、[Add tag group] を選択してタグ グループを作成します。タググループは最大3つまで作成し、それぞれに最大10個のキー と値のペアを指定できます。デプロイグループで複数のタググループを使用する場合は、 すべてのタググループによって識別されたインスタンスのみがデプロイグループに含まれま す。つまり、インスタンスがデプロイグループに含まれるには、各グループの少なくとも1 つのタグが一致する必要があります。

タググループを使用してデプロイグループを絞り込む方法ついては、「<u>Tagging Instances</u> for Deployments」を参照してください。

12. [デプロイ設定] で、アプリケーションをインスタンスをデプロイするレート (例: 一度に1つ ずつ、一度にすべて) を制御するデプロイ設定を選択します。デプロイ設定の詳細について は、CodeDeploy でデプロイ設定を使用する を参照してください。 13. (オプション) [ロードバランサー] で [ロードバランシングを有効にする] を選択し、一覧 から、CodeDeploy デプロイ中のインスタンスへのトラフィックを管理する Classic Load Balancer、Application Load Balancer のターゲットグループ、Network Load Balancer のター ゲットグループを選択します。最大 10 個の Classic Load Balancer と 10 個のターゲットグ ループとで、合計 20 個のアイテムを選択できます。デプロイする Amazon EC2 インスタンス が、選択したロードバランサー (Classic Load Balancer) またはターゲットグループ (Application Load Balancer および Network Load Balancer) に登録されていることを確認します。

デプロイ中、元のインスタンスは選択したロードバランサーとターゲットグループから登録解除 され、デプロイ中にトラフィックがこれらのインスタンスにルーティングされないようにしま す。デプロイが完了すると、各インスタンスは選択したすべての Classic Load Balancer とター ゲットグループに再登録されます。

CodeDeploy デプロイでのロードバランサーの詳細については、<u>Integrating CodeDeploy with</u> Elastic Load Balancing を参照してください。

14. (オプション) アドバンストを展開し、デプロイに含めるオプション (Amazon SNS 通知トリ ガー、Amazon CloudWatch アラーム、自動ロールバックなど) を設定します。

詳細については、「<u>デプロイグループの詳細オプションの設定</u>」を参照してください。 15. デプロイグループの作成 を選択します。

次のステップでは、アプリケーションおよびデプロイグループにデプロイするリビジョンを準備しま す。手順については、CodeDeploy のアプリケーションリビジョンの操作 を参照してください。

Blue/Green デプロイ (コンソール) のアプリケーションを作成します。

CodeDeploy コンソールを使用して、Blue/Green デプロイ用のアプリケーションを作成するには

Note

AWS Lambda コンピューティングプラットフォームへのデプロイは常にブルー/グリーンデ プロイです。デプロイタイプオプションは指定しません。

\Lambda Warning

次の場合は、これらの手順を実行しないでください。

- Blue/Green デプロイプロセス中に置き換えたい CodeDeploy エージェントがインストール されたインスタンスはありません。インスタンスをセットアップするには、<u>CodeDeploy</u> <u>のためにインスタンスを用いた操作</u>の指示に従い、その後にこのトピックの手順に従いま す。
- カスタムデプロイ設定を使用するアプリケーションを作成する必要があり、まだデプロイ 設定を作成していません。Create a Deployment Configuration の指示に従った後に、この トピックの手順に従います。
- 少なくとも、「ステップ 2: CodeDeployのサービスのロールを作成する」に記載されている信頼とアクセス権限を持つ、CodeDeployを信頼するサービスロールがない。サービスロールを作成して設定するには、ステップ 2: CodeDeployのサービスのロールを作成するの指示に従い、その後にこのトピックの手順に従います。
- ・置き換え先環境でインスタンスを登録するために、Elastic Load Balancing で Classic Load Balancer、Application Load Balancer、または Network Load Balancer を作成していませ ん。詳細については、「<u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing</u> でロードバランサーをセットアップする」を参照してください。
- にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

CodeDeploy の開始方法 で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで、[デプロイ]、[ご利用開始にあたって]の順に選択します。
- 3. [アプリケーション名] に、アプリケーションの名前を入力します。
- 4. [Compute platform (コンピューティングプラットフォーム)] で [EC2/On-Premises (EC2/オンプ レミス)] を選択します。
- 5. [Create application] を選択します。
- アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択します。
- 7. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されているのと同じ設定 (デプロイグループ名タ グ、Amazon EC2 Auto Scaling グループ名、デプロイ設定など) を使用する場合は、こ のページでこれらの設定を選択します。この新しいデプロイグループと既存のデプロ イグループは名前が同じでも、それぞれが別のアプリケーションに関連付けられるた め、CodeDeploy では別のデプロイグループとして扱われます。

- 8. サービスロール で、ターゲットインスタンスへのアクセス権を CodeDeploy に付与するサービ スロールを選択します。
- 9. [デプロイタイプ] で [Blue/Green] を選択します。
- 10. [Environment configuration] で、置き換え先環境にインスタンスを提供するために使用する方法 を選択します:
 - a. Amazon EC2 Auto Scaling グループを自動的にコピーする: CodeDeploy は、ユーザーが指定したグループをコピーして Amazon EC2 Auto Scaling グループを作成します。
 - b. [Manually provision instances]: デプロイを作成するまで置き換え先環境のインスタンスを特定しません。デプロイを開始する前に、インスタンスを作成する必要があります。代わりに、ここで置換するインスタンスを指定します。
- 11. ステップ 10 での選択内容に応じて、次のいずれかを実行します:
 - [Amazon EC2 Auto Scaling グループを自動コピーする]を選択している場合: [Amazon EC2 Auto Scaling グループ]で、置き換え先環境のインスタンス用に作成される Amazon EC2 Auto Scaling グループのテンプレートとして使用したい Amazon EC2 Auto Scaling グループの名前 を選択または入力します。選択した Amazon EC2 Auto Scaling グループ内の現在正常なイン スタンスの数が、置き換え先環境で作成されます。
 - インスタンスの手動プロビジョンを選択している場合: Amazon EC2 Auto Scaling グループと Amazon EC2 インスタンスのいずれかまたは両方を有効にして、このデプロイグループに追 加するインスタンスを指定します。元の環境のインスタンス (つまり、置換対象のインスタン スまたは現在のアプリケーションリビジョンを実行しているインスタンス) を識別するための Amazon EC2 タグ値または Amazon EC2 Auto Scaling グループ名を入力します。
- [ロードバランサー]で [ロードバランシングを有効にする]を選択し、一覧から、代わりの Amazon EC2 インスタンスを登録する Classic Load Balancer、Application Load Balancer の ターゲットグループ、Network Load Balancer のターゲットグループを選択します。各代替イン スタンスは、選択したすべての Classic Load Balancer とターゲットグループに登録されます。

最大 10 個の Classic Load Balancer と 10 個のターゲットグループとで、合計 20 個のアイテム を選択できます。

トラフィックは、選択した [トラフィック再ルーティング] と [デプロイ設定] に従って、元のイ ンスタンスから代替インスタンスに再ルーティングされます。

CodeDeploy デプロイ用のロードバランサーの詳細については、「<u>Integrating CodeDeploy with</u> Elastic Load Balancing」を参照してください。

[Deployment settings] で、置き換え先環境へトラフィックを再ルーティングするためのデフォルトのオプション、デプロイに使用するデプロイ設定、デプロイ後に元の環境のインスタンスを処理する方法を確認します。

設定を変更する場合は、次のステップに進みます。それ以外の場合は、ステップ 15 に進みま す。

14. Blue/Green デプロイのデプロイ設定を変更するには、以下のいずれかの設定を変更します。

設定	オプション
[Traffic rerouting]	 [トラフィックをすぐに再ルーティングする]: 置き換え先環境のインスタンスがプロビジョニングされ、最新のアプリケーションリビジョンがインストールされるとすぐに、指定のロードバランサーとターゲットグループに自動的に登録され、トラフィックがそれらに再ルーティングされます。 [トラフィックを再ルーティングするかどうかを選択します]: 置き換え先環境のインスタンスは、手動でトラフィックを再ルーティングしないかぎり、指定のロードバランサーとターゲットグループに登録されません。指定した待機時間中にトラフィックが再ルーティングされない場合、デプロイステータスは停止に変更されます。

設定	オプション
Deployment configuration	置き換え先環境のインスタンスがロードバラ ンサーとターゲットグループに登録されてい るレート (個別、一括など) を選択します。
	 Note トラフィックが置き換え先環境に適切にルーティングされた後、元の環境のインスタンスは、どのデプロイ設定が選択されていても一度にすべて登録解除されます。
	詳細については、「 <u>CodeDeploy でデプロイ</u> <u>設定を使用する</u> 」を参照してください。
Original instances	 [デプロイグループの元のインスタンスを 終了する]: トラフィックが置き換え先環境 に再ルーティングされた後、ロードバラン サーとターゲットグループから登録解除さ れたインスタンスは、指定した待機時間の 後に終了します。
	 [デプロイグループの元のインスタンスを 実行し続ける]: トラフィックが置き換え先 環境に再ルーティングされた後、ロードバ ランサーとターゲットグループから登録解 除されたインスタンスは実行されたままに なります。

15. (オプション) アドバンスト で、Amazon SNS 通知トリガー、Amazon CloudWatch アラーム、 自動ロールバックなど、デプロイに含めたいオプションを設定します。

デプロイグループの詳細なオプションを指定する方法の詳細については、「<u>デプロイグループの</u> 詳細オプションの設定」を参照してください。

16. デプロイグループの作成を選択します。

次のステップでは、アプリケーションおよびデプロイグループにデプロイするリビジョンを準備しま す。手順については、CodeDeploy のアプリケーションリビジョンの操作 を参照してください。

Amazon ECS サービスデプロイ用のアプリケーションを作成 (コンソール)

CodeDeploy コンソールを使用して Amazon ECS サービスデプロイ用のアプリケーションを作成で きます。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

1 Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで デプロイ を展開して、ご利用開始にあたって を選択します。
- 3. [アプリケーションの作成]ページで、[CodeDeploy の使用]を選択します。
- 4. [アプリケーション名] に、アプリケーションの名前を入力します。
- 5. コンピューティングプラットフォームから、Amazon ECS を選択します。
- 6. [Create application] を選択します。
- アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択し ます。Amazon ECS デプロイのデプロイグループを作成するために必要なものの詳細について は、「Amazon ECS デプロイを開始する前に」を参照してください。
- 8. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されているのと同じ設定 (デプロイグループ名、タグ、 グ ループ名、デプロイ設定など)を使用する場合は、このページでこれらの設定を選択し ます。この新しいグループと既存のグループは名前が同じでも、それぞれが別のアプリ ケーションに関連付けられるため、CodeDeploy では別のデプロイグループとして扱わ れます。

9. サービスロール で、CodeDeploy に Amazon ECS へのアクセスを許可するサービスロールを選 択します。詳細については、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」を参照 してください。

- 10. ロードバランサーの名前 から、Amazon ECS サービスにトラフィックを提供するロードバラン サーの名前を選択します。
- 11. [本稼働リスナーポート] から、Amazon ECS サービスへの本稼働トラフィックを提供するリス ナーのポートとプロトコルを選択します。
- 12. (オプション) テストリスナーポート から、デプロイ時に Amazon ECS サービス内の置き換 えタスクセットにトラフィックを処理するテストリスナーのポートとプロトコルを選択しま す。AfterAllowTestTraffic フック中に実行する Lambda 関数を、AppSpec ファイル内に ーつ以上指定できます。この関数は、検証テストを実行できます。検証テストが失敗すると、 デプロイのロールバックが発生します。検証テストに成功すると、デプロイのライフサイクル の次のフック BeforeAllowTraffic がトリガーされます。テストリスナーポートが指定され ていない場合、AfterAllowTestTraffic フック中は何も起こりません。詳細については、 「Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション」を参照してください。
- [Target group 1 name] および [Target group 2 name] から、デプロイ時にトラフィックをルー ティングするターゲットグループを選択します。CodeDeploy は、1 つのターゲットグループを Amazon ECS サービスの元のタスクセットにバインドし、もう一方をその置き換えタスクセッ トにバインドします。詳細については、「<u>Application Load Balancer のターゲットグループ</u>」を 参照してください。
- 14. [トラフィックをすぐに再ルーティングする] または [トラフィックを再ルーティングするタイミ ングを指定する] を選択し、更新された Amazon ECS サービスにトラフィックを再ルーティン グするタイミングを決定します。

[トラフィックをすぐに再ルーティングする] を選択すると、置き換えタスクセットがプロビジョ ニングされた後、デプロイによってトラフィックが自動的に再ルーティングされます。

[トラフィックを再ルーティングするタイミングを指定する]を選択すると、置き換えタスクセットが正常にプロビジョニングされてから待機する日数、時間、分を選択します。この待機時間の間に、AppSpec ファイルで指定された Lambda 関数の検証テストが実行されます。トラフィックが再ルーティングされる前に待機時間が終了した場合、デプロイステータスは Stopped に変更されます。

- 15. 元のリビジョンの終了 では、デプロイが成功してからAmazon ECS サービスの元のタスクセットが終了するまで待機する日数、時間、分数を選択します。
- 16. (オプション) アドバンスト で、Amazon SNS 通知トリガー、Amazon CloudWatch アラーム、 自動ロールバックなど、デプロイに含めたいオプションを設定します。

詳細については、「デプロイグループの詳細オプションの設定」を参照してください。

AWS Lambda 関数デプロイ用のアプリケーションを作成 (コンソール)

CodeDeploy コンソールを使用して、 AWS Lambda 関数のデプロイ用のアプリケーションを作成できます。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note 「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで デプロイ を展開して、ご利用開始にあたって を選択します。
- 3. [アプリケーションの作成]ページで、[CodeDeploy の使用]を選択します。
- 4. [アプリケーション名] にアプリケーションの名前を入力します。
- 5. [Compute platform (コンピューティングプラットフォーム)] で [AWS Lambda] を選択します。
- 6. [Create application] を選択します。
- 7. アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択しま す。
- 8. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されているのと同じ設定 (デプロイグループ名、タグ、 グ ループ名、デプロイ設定など)を使用する場合は、このページでこれらの設定を選択し ます。この新しいデプロイグループと既存のデプロイグループは名前が同じでも、それ ぞれが別のアプリケーションに関連付けられるため、CodeDeploy では別のデプロイグ ループとして扱われます。

- 9. サービスロールで、CodeDeploy にアクセス権を付与するサービスロールを選択します AWS Lambda。詳細については、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」を参照 してください。
- 10. 事前定義済みのデプロイ設定を使用する場合は、[デプロイ設定] から 1 つを選択し、ステップ
 12 に進みます。カスタム設定を作成するには、次のステップに進みます。

デプロイ設定の詳細については、<u>AWS Lambda コンピューティングプラットフォームのデプロ</u> イ設定 を参照してください。 11. カスタム設定を作成するには、[デプロイ設定の作成] を選択し、以下の操作を行います。

- a. [デプロイ設定名] に、設定の名前を入力します。
- b. [タイプ] で、設定タイプを選択します。[Canary] を選択すると、トラフィックは 2 回の増分 で移行されます。[Linear] を選択すると、トラフィックは毎回同じ間隔 (分) の等しい増分で 移行します。
- c. [Step] に、移行するトラフィックの割合を1~99 で入力します。設定タイプが [Canary] の 場合、この値は最初の増分で移行されるトラフィックの割合を示します。残りのトラフィッ クは、2 回目の増分で、選択した間隔後に移行されます。設定タイプが [Linear] の場合、こ の値は各間隔の開始時に移行されるトラフィックの割合を示します。
- d. [Interval (間隔)] に、時間 (分数) を入力します。設定タイプが [Canary] の場合、この値は最 初と 2 回目のトラフィック移行の時間 (分) を示します。設定タイプが [Linear (リニア)] の 場合、この値は各増分の移行間の時間 (分) を示します。

Note

AWS Lambda デプロイの最大長は2日、つまり2,880分です。したがっ て、Canary 設定の [Interval] に指定された最大値は、2,800分です。リニア設定の 最大値は、[Step] の値によって異なります。たとえば、トラフィックのリニア移 行のステップの割合が25%の場合、トラフィック移行は4回です。最大間隔値 は、2,880を4で割るか、720分になります。

- e. [Create deployment configuration (デプロイ設定の作成)] を選択します。
- 12. (オプション) アドバンストで、デプロイに含めるオプション (Amazon SNS 通知トリ ガー、Amazon CloudWatch アラーム、自動ロールバックなど) を設定します。

詳細については、「デプロイグループの詳細オプションの設定」を参照してください。

13. デプロイグループの作成を選択します。

アプリケーションの作成 (CLI)

を使用してアプリケーション AWS CLI を作成するには、<u>create-application</u> コマンドを呼び出し、ア プリケーションを一意に表す名前を指定します。(AWS アカウントの場合、CodeDeploy アプリケー ション名はリージョンごとに 1 回のみ使用できます。異なるリージョンでアプリケーション名を再 利用することができます。) を使用してアプリケーション AWS CLI を作成したら、次のステップとして、リビジョンをデプロイ するインスタンスを指定するデプロイグループを作成します。手順については、<u>CodeDeploy でデプ</u> ロイグループを作成する を参照してください。

デプロイグループを作成したら、次にアプリケーションおよびデプロイグループにデプロイするリビ ジョンを準備します。手順については、<u>CodeDeploy のアプリケーションリビジョンの操作</u> を参照 してください。

CodeDeploy を使用してアプリケーション詳細を表示する

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、 AWS アカウントに関 連付けられているすべてのアプリケーションの詳細を表示できます。

トピック

- アプリケーションの詳細を表示する (コンソール)
- アプリケーションの詳細を表示する (CLI)

アプリケーションの詳細を表示する (コンソール)

CodeDeploy コンソールを使用してアプリケーションの詳細を表示するには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開して [ご利用開始にあたって] を選択します。
- 追加のアプリケーション詳細を表示するには、リストからアプリケーションの名前を選択します。

アプリケーションの詳細を表示する (CLI)

を使用してアプリケーションの詳細 AWS CLI を表示するには、 get-application コマンド、 batchget-application コマンド、または list-applications コマンドを呼び出します。 単一のアプリケーションに関する詳細を表示するには、<u>get-application</u> コマンドを呼び出し、アプリ ケーション名を指定します。

複数のアプリケーションに関する詳細を表示するには、<u>batch-get-applications</u> コマンドを呼び出 し、複数のアプリケーションの名前を指定します。

アプリケーション名のリストを表示するには、list-applications コマンドを呼び出します。

通知ルールの作成

通知ルールを使用すると、デプロイの成功や失敗など、デプロイアプリケーションに変更があった場 合にユーザーに通知できます。通知ルールは、イベントと、通知の送信に使用される Amazon SNS トピックの両方を指定します。詳細については、「通知とは」を参照してください。

コンソールまたは を使用して AWS CLI、通知ルールを作成できます AWS CodeDeploy。

通知ルールを作成するには (コンソール)

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy/</u>:// https://www.com で CodeDeploy コンソールを開きます。
- 2. [Application (アプリケーション)]を選択し、通知を追加するアプリケーションを選択します。
- アプリケーションページで、[Notify (通知)]、[Create notification rule (通知ルールの作成)] の順に 選択します。アプリケーションの [Settings (設定)] ページに移動し、[Create notification rule (通 知ルールの作成)] を選択することもできます。
- 4. [通知名] に、ルールの名前を入力します。
- 5. Amazon EventBridge に提供された情報のみを通知に含める場合は、[Detail type (詳細タイプ)] で [Basic (基本)] を選択します。Amazon EventBridge に提供される情報に加えて、CodeDeploy または通知マネージャから提供される場合がある情報も含める場合は、[Full (完全)] を選択しま す。

詳細については、「通知の内容とセキュリティについて」を参照してください。

6. [Events that trigger notifications (通知をトリガーするイベント)] で、通知を送信するイベントを 選択します。

カテゴリ	イベント
デプロイ	失敗

カテゴリ	イベント
	成功

起動済み

7. [ターゲット] で、[SNS トピックの作成] を選択します。

Note

トピックを作成すると、トピックへのイベントの発行を CodeDeploy に許可するポリ シーが適用されます。CodeDeploy 通知専用に作成されたトピックを使用すると、この デプロイアプリケーションに関する通知を表示するトピックのサブスクリプションリス トにのみユーザーを追加することもできます。

[codestar-notifications-] プレフィックスの後にトピックの名前を入力し、[送信] を選択します。

Note

新しいトピックを作成する代わりに既存の Amazon SNS トピックを使用する場合は、 [Targets (ターゲット)] でその ARN を選択します。トピックに適切なアクセスポリシー が設定されていること、およびサブスクライバーリストにデプロイアプリケーション に関する情報の表示を許可されたユーザーのみが含まれていることを確認します。詳細 については、「<u>通知用の Amazon SNS トピックを設定する</u>」および「<u>通知の内容とセ</u> <u>キュリティについて理解する」を参照してください。</u>

- 8. ルールの作成を終了するには、[Submit (送信)]を選択します。
- 通知を受け取るには、そのルールの Amazon SNS トピックにユーザーをサブスクライブする必要があります。詳細については、「<u>ターゲットである Amazon SNS トピックへのユーザーのサブスクライブ</u>」を参照してください。チャットアプリケーションで通知と Amazon Q Developer の統合を設定して、Amazon Chime チャットルームまたは Slack チャネルに通知を送信することもできます。詳細については、「チャットアプリケーションで通知と Amazon Q Developer の統合を設定する」を参照してください。

通知ルールを作成するには (AWS CLI)

ターミナルまたはコマンドプロンプトで、create-notification rule コマンドを実行して、JSON スケルトンを生成します。

ファイルには任意の名前を付けることができます。この例では、ファイルの名前を *rule.json* とします。

 プレーンテキストエディタで JSON ファイルを開き、これを編集してルールに必要 なリソース、イベントタイプ、Amazon SNS ターゲットを含めます。次の例は、ID MyNotificationRuleが 123456789012 AWS アカウントで MyDeploymentApplication という名前のアプリケーションの という名前の通知ルールを示しています。デプロイが成功 したとき、通知は、完全な詳細タイプで codestar-##-MyNotificationTopic という名で Amazon SNS トピックに送信されます:

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codedeploy-application-deployment-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDeploymentApplication",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

ファイルを保存します。

3. 先ほど編集したファイルを使用して、ターミナルまたはコマンドラインで、create-notificationrule コマンドを再度実行し、通知ルールを作成します。 {

```
aws codestar-notifications create-notification-rule --cli-input-json
file://rule.json
```

4. 成功すると、コマンドは次のような通知ルールの ARN を返します。

"Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/ dc82df7a-EXAMPLE"
}

CodeDeploy アプリケーション名の変更

AWS CLI または CodeDeploy APIs を使用して、アプリケーションの名前を変更できます。

アプリケーション名のリストを表示するには、 AWS CLI を使用して <u>list-applications</u> コマンドを呼 び出します。

を使用してアプリケーション名 AWS CLI を変更する方法については、<u>「update-application</u>」を参照 してください。

CodeDeploy API を使用してのアプリケーション名変更の詳細については、 「API_UpdateApplication」を参照してください。

CodeDeploy でアプリケーションを削除する

CodeDeploy コンソール、 AWS CLI、または CodeDeploy API アクションを使用してア プリケーションを削除できます。CodeDeploy の API アクションの使用方法については、 「DeleteApplication」を参照してください。

▲ Warning

アプリケーションを削除すると、関連するすべてのデプロイグループの情報およびデプロ イの詳細を含む、アプリケーションに関する情報が CodeDeploy システムから削除されま す。EC2 オンプレミスのデプロイ用に作成されたアプリケーションを削除しても、インスタ ンスからアプリケーションのリビジョンが削除されたり、Amazon S3 バケットからリビジョ ンが削除されたりすることはありません。EC2 オンプレミスのデプロイ用に作成されたアプ リケーションを削除しても、Amazon EC2 インスタンスが削除されたり、オンプレミスイン スタンスが登録解除されたりすることは一切ありません。このアクションを元に戻すことは できません。

トピック

- アプリケーションの削除 (コンソール)
- ・ アプリケーションの削除 (AWS CLI)

アプリケーションの削除 (コンソール)

CodeDeploy コンソールを使用してアプリケーションを削除するには以下を実行してください。

 にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. アプリケーションの一覧で、削除するアプリケーションの名前を選択します。

アプリケーションに関する詳細を含むページが表示されます。

- 4. 右上にあるアプリケーションの削除を選択します。
- 5. プロンプトが表示されたら、delete を入力し、削除したいアプリケーションが正しいことを確認します。確認したら 削除 を選択します。

アプリケーションの削除 (AWS CLI)

を使用してアプリケーション AWS CLI を削除するには、delete<u>-application</u> コマンドを呼び出し、ア プリケーション名を指定します。アプリケーション名のリストを表示するには、<u>list-applications</u> コ マンドを呼び出します。

CodeDeploy でのデプロイグループの使用

CodeDeploy アプリケーション用に1つ以上のデプロイグループを指定できます。各アプリケーショ ンのデプロイでは、そのデプロイグループの1つを使用します。デプロイグループには、デプロ イ中に使用される設定と構成が含まれています。ほとんどのデプロイグループ設定は、アプリケー ションで使用されるコンピューティングプラットフォームによって異なります。ロールバック、トリ ガー、アラームなどの一部の設定は、どのコンピューティングプラットフォーム用のデプロイグルー プでも設定できます。

Amazon ECS コンピューティングプラットフォームのデプロイで のデプロイグループ

Amazon ECS デプロイでは、デプロイグループは Amazon ECS サービス、ロードバランサー、オプ ションのテストリスナー、2 つのターゲットグループを指定します。また、代替タスクにトラフィッ クを再ルーティングするタイミングと、デプロイが成功した後で元のタスクセットと Amazon ECS アプリケーションを終了させるタイミングを設定します。

AWS Lambda コンピューティングプラットフォームデプロイのデ プロイグループ

AWS Lambda デプロイでは、デプロイグループは、 AWS Lambda 関数の今後のデプロイ用に CodeDeploy 設定のセットを定義します。例えば、デプロイグループでは、新しいバージョンの Lambda 関数にトラフィックをルーティングする方法を指定します。また、アラームとロールバック を指定する場合もあります。 AWS Lambda デプロイグループ内の 1 つのデプロイは、1 つ以上のグ ループ設定を上書きできます。

EC2 オンプレミスコンピューティングプラットフォームのデプロ イでのデプロイグループ

EC2/オンプレミス のデプロイでは、デプロイグループはデプロイをターゲットにした個別のイン スタンスのセットです。デプロイグループには、個別にタグ付けされた Amazon EC2 インスタン ス、Amazon EC2 Auto Scaling グループ内の Amazon EC2 インスタンス、またはその両方が含まれ ます。 インプレースデプロイでは、デプロイグループのインスタンスは最新のアプリケーションリビジョン で更新されます。

ブルー/グリーンデプロイでは、1 つ以上のロードバランサーから元のインスタンスを登録解除し、 通常は最新のアプリケーションリビジョンが既にインストールされたインスタンスの代替セットを登 録して、インスタンスの 1 つのセットから別のセットにトラフィックが転送されます。

複数のデプロイグループを CodeDeploy のアプリケーションに関連付けることができます。これに より、インスタンスの別々のセットに異なるタイミングでアプリケーションリビジョンをデプロイで きます。例えば、1 つのデプロイグループを使用して、Test というタグが付けられた、コードの品 質を確認するインスタンスのセットにアプリケーションリビジョンをデプロイできます。次に、追加 の確認のため、Staging というタグが付けられたインスタンスがあるデプロイグループに、同じア プリケーションリビジョンをデプロイします。最後に、最新アプリケーションを顧客にリリースする 準備ができたら、Production というタグが付けられたインスタンスを含むデプロイグループにデ プロイします。

複数のタググループを使用して、デプロイグループに含めるインスタンスをさらに絞り込むこともで きます。詳細については、Tagging Instances for Deployments を参照してください。

CodeDeploy コンソールを使用してアプリケーションを作成する場合は、最初のデプロイグループを 同時に設定します。を使用してアプリケーション AWS CLI を作成する場合は、別のステップで最初 のデプロイグループを作成します。

AWS アカウントに関連付けられているデプロイグループのリストを表示するには、「」を参照して くださいCodeDeploy を使用したデプロイグループの詳細の表示。

Amazon EC2 インスタンスタグの詳細については、「<u>コンソールでのタグの処理</u>」を参照してくだ さい。オンプレミスインスタンスの詳細については、「<u>Working with On-Premises Instances</u>」を 参照してください。Amazon EC2 Auto Scaling の情報に関しては、「<u>CodeDeploy と Amazon EC2</u> Auto Scaling の統合」を参照してください。

トピック

- the section called "デプロイグループの作成"
- the section called "デプロイグループの詳細の表示"
- the section called "デプロイグループの設定を変更する"
- the section called "デプロイグループの詳細オプションの設定"
- the section called "デプロイグループを削除する"

CodeDeploy でデプロイグループを作成する

CodeDeploy コンソール、 AWS CLI、CodeDeploy APIs、または AWS CloudFormation テンプ レートを使用してデプロイグループを作成できます。 AWS CloudFormation テンプレートを使用 してデプロイグループを作成する方法については、「」を参照してください<u>AWS CloudFormation</u> CodeDeploy リファレンスの テンプレート。

CodeDeploy コンソールを使用してアプリケーションを作成する場合は、最初のデプロイグループを 同時に設定します。を使用してアプリケーション AWS CLI を作成する場合は、別のステップで最初 のデプロイグループを作成します。

デプロイグループ作成の一環として、サービスロールを指定する必要があります。詳細については、 「ステップ 2: CodeDeployのサービスのロールを作成する」を参照してください。

トピック

- インプレースデプロイ用のデプロイグループを作成する (コンソール)
- EC2/オンプレミス Blue/Green デプロイ用のデプロイグループを作成する (コンソール)
- Amazon ECS デプロイ用のデプロイグループを作成する (コンソール)
- <u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバランサーをセットアッ</u> プする
- <u>CodeDeploy Amazon ECS デプロイ用のロードバランサー、ターゲットグループ、リスナーを</u> セットアップする
- デプロイグループの作成 (CLI)

インプレースデプロイ用のデプロイグループを作成する (コンソール)

CodeDeploy コンソールを使用して、インプレースデプロイ用のデプロイグループを作成する方法。

▲ Warning

次の場合は、これらの手順を実行しないでください。

 アプリケーションの最初の CodeDeploy デプロイで使用するインスタンスを準備していない。インスタンスをセットアップするには、CodeDeploy のためにインスタンスを用いた 操作の指示に従い、その後にこのトピックの手順に従います。

- カスタムデプロイ設定を使用してデプロイグループを作成したいが、まだデプロイ設定を 作成していない。Create a Deployment Configuration の指示に従った後に、このトピック の手順に従います。
- 少なくとも、「ステップ 2: CodeDeployのサービスのロールを作成する」に記載されている信頼とアクセス権限を持つ、CodeDeployを信頼するサービスロールがない。サービスロールを作成して設定するには、ステップ 2: CodeDeployのサービスのロールを作成するの指示に従い、その後にこのトピックの手順に従います。
- ・インプレースデプロイのために Elastic Load Balancing で、Classic Load Balancer、Application Load Balancer、または Network Load Balancer を選択したいが、ま だ作成していない場合。
- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. [Applications] ページで、デプロイグループを作成するアプリケーションの名前を選択します。
- アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択します。
- 5. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されている設定 (デプロイグループ名、タグ、Amazon EC2 Auto Scaling グループ名、または両方、およびデプロイ設定を含む) を使用する場合は、 このページでこれらの設定を指定します。この新しいデプロイグループと既存のデプ ロイグループは名前が同じでも、それぞれが別のアプリケーションに関連付けられるた め、CodeDeploy では別のデプロイグループとして扱われます。

- 6. [サービスロール] で、ターゲットインスタンスへのアクセス権を CodeDeploy に付与するサービ スロールを選択します。
- 7. [デプロイタイプ] で、[インプレース] を選択します。

- 8. [環境設定] で、次の操作を行います。
 - a. アプリケーションを Amazon EC2 Auto Scaling グループにデプロイする場合は、[Amazon EC2 Auto Scaling グループ]を選択し、アプリケーションリビジョンをデプロイする先の Amazon EC2 Auto Scaling グループの名前を選択します。Amazon EC2 Auto Scaling グループのの名前を選択します。CodeDeploy で はリビジョンを新しいインスタンスに自動的にデプロイできます。デプロイグループに は最大 10 個の Amazon EC2 Auto Scaling グループを追加できます。詳細については、「CodeDeploy と Amazon EC2 Auto Scaling の統合」を参照してください。
 - b.

[Amazon EC2 Auto Scaling グループ] を選択した場合、オプションで [Auto Scaling グルー プに終了フックを追加] を選択すると、デプロイグループを作成または更新するときに CodeDeploy が Auto Scaling グループに終了フックをインストールします。このフックを インストールすると、CodeDeploy は終了デプロイを実行します。詳細については、「<u>Auto</u> <u>Scaling スケールインイベント中の終了デプロイの有効化</u>」を参照してください。

- c. インスタンスにタグを付ける場合は、[Amazon EC2 インスタンス] または [オンプレミスイ ンスタンス] を選択します。[キー] フィールドと [値] フィールドに、インスタンスにタグを 付けるために使用するキーと値のペアの値を入力します。単一タググループで最大 10 個の キーと値のペアをタグ付けできます。
 - i. [値] フィールドでワイルドカードを使用して、似ている Amazon EC2 インスタンス、 コストセンター、グループ名などの特定のパターンでタグ付けされているすべてのイン スタンスを識別できます。例えば、キー フィールドに 名前 を選択し、値 フィールド に GRP-*a を入力すると、CodeDeploy は GRP-1a、GRP-2a、および GRP-XYZ-a な どそのパターンに当てはまるすべてのインスタンスを特定します。
 - ii. [値] フィールドでは、大文字と小文字が区別されます。
 - iii. リストからキーと値のペアを削除するには、削除のアイコンを選択します。

CodeDeploy で、指定されたキーバリューの各ペアまたは Amazon EC2 Auto Scaling グ ループ名に一致するインスタンスが検出されると、一致したインスタンスの数が表示されま す。インスタンスに関する詳細情報を表示するには、数をクリックします。

インスタンスへのデプロイの条件をさらに絞り込むには、[Add tag group] を選択してタグ グループを作成します。それぞれ最大 10 個のキーと値のペアを持つタググループを 3 つま で作成できます。デプロイグループで複数のタググループを使用する場合は、すべてのタグ グループによって識別されたインスタンスのみがデプロイグループに含まれます。つまり、 インスタンスがデプロイグループに含まれるには、各グループの少なくとも 1 つのタグが 一致する必要があります。

タググループを使用してデプロイグループを絞り込む方法ついては、「<u>Tagging Instances</u> for Deployments」を参照してください。

- [Systems Manager を使用したエージェント設定] で、デプロイグループのインスタンスに CodeDeploy エージェントをインストールおよび更新する方法を指定します。CodeDeploy エージェントの詳細については、「CodeDeploy エージェントの使用」を参照してくださ い。Systems Manager の詳細については、「Systems Manager とは」を参照してください。
 - a. Never: Systems Manager を使用する CodeDeploy インストールの設定をスキップします。
 デプロイで使用するには、インスタンスにエージェントがインストールされている必要があります。したがって、CodeDeploy エージェントを別の方法でインストールする場合のみ、このオプションを選択します。
 - b. [1回のみ]: Systems Manager は、デプロイグループ内のすべてのインスタンスに CodeDeploy エージェントを 1 回インストールします。
 - c. [現在およびスケジュール更新]: Systems Manager は、設定したスケジュールに従って CodeDeploy エージェントをインストールするステートマネージャーとの関連付けを作成し ます。ステートマネージャーおよび関連付けの詳細については、「ステートマネージャーに ついて」を参照してください。
- 10. [デプロイ設定] で、インスタンスをデプロイするレート (一度に 1 つずつ、一度にすべて、など) を制御するデプロイ設定を選択します。デプロイ設定の詳細については、<u>CodeDeploy でデプロ</u> イ設定を使用する を参照してください。
- 11. (オプション) [ロードバランサー] で [ロードバランシングを有効にする] を選択し、一覧 から、CodeDeploy デプロイ中のインスタンスへのトラフィックを管理する Classic Load Balancer、Application Load Balancer のターゲットグループ、Network Load Balancer のター ゲットグループを選択します。最大 10 個の Classic Load Balancer と 10 個のターゲットグ ループとで、合計 20 個のアイテムを選択できます。デプロイする Amazon EC2 インスタンス が、選択したロードバランサー (Classic Load Balancer) またはターゲットグループ (Application Load Balancer および Network Load Balancer) に登録されていることを確認します。

デプロイ中、元のインスタンスは選択したロードバランサーとターゲットグループから登録解除 され、デプロイ中にトラフィックがこれらのインスタンスにルーティングされないようにしま す。デプロイが完了すると、各インスタンスは選択したすべての Classic Load Balancer とター ゲットグループに再登録されます。

インプレースデプロイ用のデプロイグループを作成する (コンソール)

CodeDeploy デプロイ用のロードバランサーの詳細については、「<u>Integrating CodeDeploy with</u> Elastic Load Balancing」を参照してください。

🛕 Warning

このデプロイグループで Auto Scaling グループと Elastic Load Balancing ロードバラン サーの両方を設定し、<u>Auto Scaling グループにロードバランサーをアタッチする場合</u> <u>は</u>、このデプロイグループから CodeDeploy デプロイを作成する前にこのアタッチメン トを完了することをお勧めします。デプロイを作成した後にアタッチメントを完了しよ うとすると、すべてのインスタンスがロードバランサーから予期せず登録解除される可 能性があります。

12. (オプション) [アドバンスト] を展開し、デプロイに含めるオプション (Amazon SNS 通知トリ ガー、Amazon CloudWatch アラーム、Auto Scaling オプション、自動ロールバックなど) を設 定します。

詳細については、「<u>デプロイグループの詳細オプションの設定</u>」を参照してください。

13. デプロイグループの作成 を選択します。

EC2/オンプレミス Blue/Green デプロイ用のデプロイグループを作成する (コンソール)

CodeDeploy コンソールを使用して、Blue/Green デプロイ用のデプロイグループを作成する方法。

🔥 Warning

次の場合は、これらの手順を実行しないでください。

- Blue/Green デプロイプロセス中に置き換えたい CodeDeploy エージェントがインストール されたインスタンスはありません。インスタンスをセットアップするには、CodeDeploy のためにインスタンスを用いた操作の指示に従い、その後にこのトピックの手順に従いま す。
- カスタムデプロイ設定を使用するアプリケーションを作成する必要があり、まだデプロイ 設定を作成していません。Create a Deployment Configuration の指示に従った後に、この トピックの手順に従います。

- ・ 少なくとも、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」に記載されてい る信頼とアクセス権限を持つ、CodeDeploy を信頼するサービスロールがない。サービス ロールを作成して設定するには、<u>ステップ 2: CodeDeployのサービスのロールを作成する</u> の指示に従い、その後にこのトピックの手順に従います。
- ・置き換え先環境でインスタンスを登録するために、Elastic Load Balancing で Classic Load Balancer または Application Load Balancer を作成していません。詳細については、 「<u>CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバランサーを</u> セットアップする」を参照してください。
- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. [Applications] ページで、デプロイグループを作成するアプリケーションの名前を選択します。
- アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択します。
- 5. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されているのと同じ設定 (デプロイグループ名、タ グ、Amazon EC2 Auto Scaling グループ名、デプロイ設定など) を使用する場合は、 このページでこれらの設定を選択します。この新しいデプロイグループと既存のデプ ロイグループは名前が同じでも、これらは別のアプリケーションに関連付けられるた め、CodeDeploy では別のデプロイグループとして扱われます。

- 6. [サービスロール] で、ターゲットインスタンスへのアクセス権を CodeDeploy に付与するサービ スロールを選択します。
- 7. [デプロイタイプ] で [Blue/Green] を選択します。
- 8. [環境設定] で、次の操作を行います。

- ・置き換え先環境にインスタンスを提供するために使用する方法を選択します。次のオプション があります。
 - Amazon EC2 Auto Scaling グループを自動的にコピーする: CodeDeploy は、ユーザーが指定したグループをコピーして Amazon EC2 Auto Scaling グループを作成します。
 - [Manually provision instances]: デプロイを作成するまで置き換え先環境のインスタンスを特定しません。デプロイを開始する前に、インスタンスを作成する必要があります。代わりに、ここで置換するインスタンスを指定します。
- [Amazon EC2 Auto Scaling グループを自動的にコピー] を選択した場合、オプションで [Auto Scaling グループに終了フックを追加] を選択すると、デプロイグループを作成または更新するときに CodeDeploy が Auto Scaling グループに終了フックをインストールします。このフックをインストールすると、CodeDeploy は終了デプロイを実行します。詳細については、「Auto Scaling スケールインイベント中の終了デプロイの有効化」を参照してください。
- [Systems Manager を使用したエージェント設定] で、デプロイグループのインスタンスに CodeDeploy エージェントをインストールおよび更新する方法を指定します。CodeDeploy エージェントの詳細については、「CodeDeploy エージェントの使用」を参照してくださ い。Systems Manager の詳細については、「Systems Manager とは」を参照してください。
 - a. Never: Systems Manager を使用する CodeDeploy インストールの設定をスキップします。 デプロイで使用するには、インスタンスにエージェントがインストールされている必要があ ります。したがって、CodeDeploy エージェントを別の方法でインストールする場合のみ、 このオプションを選択します。
 - b. [1回のみ]: Systems Manager は、デプロイグループ内のすべてのインスタンスに CodeDeploy エージェントを1回インストールします。
 - c. [現在およびスケジュール更新]: Systems Manager は、設定したスケジュールに従って CodeDeploy エージェントをインストールするステートマネージャーとの関連付けを作成し ます。ステートマネージャーおよび関連付けの詳細については、「ステートマネージャーに ついて」を参照してください。
- 10. ステップ 8 での選択内容に応じて、次のいずれかを実行します:
 - [Amazon EC2 Auto Scaling グループを自動コピーする]を選択している場合: [Amazon EC2 Auto Scaling グループ]で、置き換え先環境のインスタンス用に作成される Amazon EC2 Auto Scaling グループのテンプレートとして使用したい Amazon EC2 Auto Scaling グループの名前 を選択または入力します。選択した Amazon EC2 Auto Scaling グループ内の現在正常なイン スタンスの数が、置き換え先環境で作成されます。

- [インスタンスを手動でプロビジョニングする]を選択している場合: [Amazon EC2 Auto Scaling グループ] と [Amazon EC2 Auto Scaling インスタンス] のいずれかまたは両方を選択 して、このデプロイグループに追加するインスタンスを指定します。元の環境のインスタンス (つまり、置換対象のインスタンスまたは現在のアプリケーションリビジョンを実行している インスタンス)を識別するための Amazon EC2 Auto Scaling タグ値または Amazon EC2 Auto Scaling グループ名を入力します。
- [ロードバランサー]で [ロードバランシングを有効にする]を選択し、一覧から、代わりの Amazon EC2 インスタンスを登録する Classic Load Balancer、Application Load Balancer の ターゲットグループ、Network Load Balancer のターゲットグループを選択します。各代替イン スタンスは、選択したすべての Classic Load Balancer とターゲットグループに登録されます。 最大 10 個の Classic Load Balancer と 10 個のターゲットグループとで、合計 20 個のアイテム を選択できます。

トラフィックは、選択した [トラフィック再ルーティング] と [デプロイ設定] に従って、元のイ ンスタンスから代替インスタンスに再ルーティングされます。

CodeDeploy デプロイ用のロードバランサーの詳細については、「<u>Integrating CodeDeploy with</u> <u>Elastic Load Balancing</u>」を参照してください。

Marning

このデプロイグループで Auto Scaling グループと Elastic Load Balancing ロードバラン サーの両方を設定していて、<u>ロードバランサーを Auto Scaling グループにアタッチする</u> 場合は、このデプロイグループから CodeDeploy デプロイを作成する前にこのアタッチ メントを完了することをお勧めします。デプロイを作成した後にアタッチメントを完了 しようとすると、すべてのインスタンスがロードバランサーから予期せず登録解除され る可能性があります。

12. [Deployment settings] で、置き換え先環境へトラフィックを再ルーティングするためのデフォル トのオプション、デプロイに使用するデプロイ設定、デプロイ後に元の環境のインスタンスを処 理する方法を確認します。

設定を変更する場合は、次のステップに進みます。それ以外の場合は、ステップ 14 に進みま す。

13. Blue/Green デプロイのデプロイ設定を変更するには、以下のいずれかの設定を選択します。

設定	オプション
[Traffic rerouting]	 「トラフィックをすぐに再ルーティングする]:置き換え先環境のインスタンスがプロビジョニングされ、最新のアプリケーションリビジョンがインストールされるとすぐに、指定のロードバランサーとターゲットグループに自動的に登録され、トラフィックがそれらに再ルーティングされます。元の環境内のインスタンスは、登録解除されます。 「トラフィックを再ルーティングするかどうかを選択します]:置き換え先環境のインスタンスは、手動でトラフィックを再ルーティングしないかぎり、指定のロードバランサーとターゲットグループに登録されません。指定した待機時間中にトラフィックが再ルーティングされない場合、デプロイステータスは停止に変更されます。
Deployment configuration	置き換え先環境のインスタンスがロードバラ ンサーとターゲットグループに登録されてい るレート (個別、一括など)を選択します。

設定	オプション
Original instances	 「デプロイグループの元のインスタンスを 終了する]: トラフィックが置き換え先環境 に再ルーティングされた後、ロードバラン サーとターゲットグループから登録解除さ れたインスタンスは、指定した待機時間の 後に終了します。 「デプロイグループの元のインスタンスを 実行し続ける]: トラフィックが置き換え先 環境に再ルーティングされた後、ロードバ ランサーとターゲットグループから登録解 除されたインスタンスは実行されたままに なります。

14. (オプション) [アドバンスト] で、デプロイに含めるオプション (Amazon SNS 通知トリ ガー、Amazon CloudWatch アラーム、Auto Scaling オプション、自動ロールバックなど) を設 定します。

デプロイグループの詳細なオプションを指定する方法の詳細については、「<u>デプロイグループの</u> 詳細オプションの設定」を参照してください。

15. デプロイグループの作成を選択します。

Amazon ECS デプロイ用のデプロイグループを作成する (コンソール)

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- [Applications table (アプリケーションテーブル)] から、編集するデプロイグループに関連付けられているアプリケーションの名前を選択します。
- 4. アプリケーションのページの [デプロイグループ] で、編集するデプロイグループの名前を選択 します。

- 5. アプリケーションのページで、[デプロイグループ] タブの [デプロイグループの作成] を選択し ます。Amazon ECS デプロイのデプロイグループを作成するために必要なものの詳細について は、「Amazon ECS デプロイを開始する前に」を参照してください。
- 6. [デプロイグループ名] に、デプロイグループを表す名前を入力します。

Note

他のデプロイグループで使用されているのと同じ設定 (デプロイグループ名、タグ、 グ ループ名、デプロイ設定など) を使用する場合は、このページでこれらの設定を選択し ます。この新しいグループと既存のグループは名前が同じでも、それぞれが別のアプリ ケーションに関連付けられるため、CodeDeploy では別のデプロイグループとして扱わ れます。

- サービスロール で、CodeDeploy に Amazon ECS へのアクセスを許可するサービスロールを選 択します。詳細については、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」を参照 してください。
- ロードバランサーの名前 から、Amazon ECS サービスにトラフィックを提供するロードバラン サーの名前を選択します。
- 9. [本稼働リスナーポート] から、Amazon ECS サービスへの本稼働トラフィックを提供するリス ナーのポートとプロトコルを選択します。
- 10. (オプション) テストリスナーポート から、デプロイ時に Amazon ECS サービス内の置き換 えタスクセットにトラフィックを処理するテストリスナーのポートとプロトコルを選択しま す。AfterAllowTestTraffic フック中に実行する Lambda 関数を、AppSpec ファイル内に ーつ以上指定できます。この関数は、検証テストを実行できます。検証テストが失敗すると、 デプロイのロールバックが発生します。検証テストに成功すると、デプロイのライフサイクル の次のフック BeforeAllowTraffic がトリガーされます。テストリスナーポートが指定され ていない場合、AfterAllowTestTraffic フック中は何も起こりません。詳細については、 「Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション」を参照してください。
- 11. [Target group 1 name] および [Target group 2 name] から、デプロイ時にトラフィックをルー ティングするターゲットグループを選択します。CodeDeploy は、1 つのターゲットグループを Amazon ECS サービスの元のタスクセットにバインドし、もう一方をその置き換えタスクセッ トにバインドします。詳細については、「<u>Application Load Balancer のターゲットグループ</u>」を 参照してください。
- 12. [トラフィックをすぐに再ルーティングする] または [トラフィックを再ルーティングするタイミングを指定する] を選択し、更新された Amazon ECS サービスにトラフィックを再ルーティングするタイミングを決定します。

[トラフィックをすぐに再ルーティングする] を選択すると、置き換えタスクセットがプロビジョ ニングされた後、デプロイによってトラフィックが自動的に再ルーティングされます。

[トラフィックを再ルーティングするタイミングを指定する]を選択すると、置き換えタスクセットが正常にプロビジョニングされてから待機する日数、時間、分を選択します。この待機時間の間に、AppSpec ファイルで指定された Lambda 関数の検証テストが実行されます。トラフィックが再ルーティングされる前に待機時間が終了した場合、デプロイステータスは Stopped に変更されます。

- 13. 元のリビジョンの終了 では、デプロイが成功してからAmazon ECS サービスの元のタスクセットが終了するまで待機する日数、時間、分数を選択します。
- 14. (オプション) アドバンスト で、Amazon SNS 通知トリガー、Amazon CloudWatch アラーム、 自動ロールバックなど、デプロイに含めたいオプションを設定します。

詳細については、「デプロイグループの詳細オプションの設定」を参照してください。

CodeDeploy Amazon EC2 デプロイ用の Elastic Load Balancing でロードバ ランサーをセットアップする

ブルー/グリーンデプロイまたはデプロイグループでオプションのロードバランサーを指定するイ ンプレースデプロイを実行する前に、事前に Elastic Load Balancing で少なくとも 1 つの Classic Load Balancer、Application Load Balancer、Network Load Balancer を作成しておく必要がありま す。Blue/Green デプロイの場合は、そのロードバランサーを使用して置き換え先環境を構成するイ ンスタンスを登録します。元の環境のインスタンスは、この同じロードバランサーにオプションで登 録できます。インプレースデプロイでは、ロードバランサーを使用して CodeDeploy で処理中のイ ンスタンスを登録解除し、作業が完了したら再登録します。

CodeDeploy は、複数のロードバランサーの背後にある Amazon EC2 インスタンスへのブルー/グ リーンデプロイとインプレースデプロイをサポートします。例えば、200 個の Amazon EC2 イン スタンスがあり、そのうちの 100 個が 2 つの Classic Load Balancer に登録され、さらに 100 個が 2 つの Application Load Balancer の 4 つのターゲットグループに登録されているとします。このシ ナリオでは、CodeDeploy を使用すると、2 つの Classic Load Balancer、2 つの Application Load Balancer、および 4 つのターゲットグループに分散している場合でも、200 個のインスタンスすべて にブルー/グリーンデプロイとインプレースデプロイを行うことができます。

CodeDeploy は、最大 10 個の Classic Load Balancer と 10 個のターゲットグループ、合計 20 個の アイテムをサポートします。 1 つ以上の Classic Load Balancer を設定するには、Classic Load Balancer のユーザーガイドにある 「<u>チュートリアル: Classic Load Balancer の作成</u>」の手順に従ってください。次の点に注意してくだ さい:

- ステップ 2: ロードバランサーの定義、[Create LB Inside] で、インスタンスを作成したときに選択したのと同じ VPC を選択します。
- ステップ 5: ロードバランサーへの EC2 インスタンスの登録で、現在デプロイグループにあるイン スタンス (インプレースデプロイ)、または元の環境に存在するように指定したインスタンス (Blue/ Green デプロイ)を選択します。
- ・ステップ 7: Load Balancer の作成と検証で、ロードバランサーの DNS アドレスをメモします。

例えば、ロードバランサーの名前を my-load-balancer とした場合、DNS アドレスは myload-balancer-1234567890.us-east-2.elb.amazonaws.com のような形式で表示されま す。

1 つ以上の Application Load Balancer を設定するには、以下のトピックのいずれかの指示に従ってく ださい。

- Application Load Balancer の作成
- ・ チュートリアル: を使用して Application Load Balancer を作成する AWS CLI

1 つ以上の Network Load Balancer を設定するには、以下のトピックのいずれかの指示に従ってくだ さい。

- Network Load Balancer を作成する
- ・ チュートリアル: を使用して Network Load Balancer を作成する AWS CLI

CodeDeploy Amazon ECS デプロイ用のロードバランサー、ターゲットグ ループ、リスナーをセットアップする

Amazon ECS コンピューティングプラットフォームを使用してデプロイを実行する前に、 Application Load Balancer または Network Load Balancer、2 つのターゲットグループ、および 1 つ または 2 つのリスナーを作成する必要があります。このトピックでは、Application Load Balancer を 作成する方法を説明します。詳細については、「<u>Amazon ECS デプロイを開始する前に</u>」を参照し てください。 ターゲットグループの1つが、Amazon ECS アプリケーションの元のタスクセットにトラフィック を誘導します。もう1つのターゲットグループは、置き換えタスクセットにトラフィックを送信し ます。デプロイ中、CodeDeploy は置き換えタスクセットを作成し、元のタスクセットから新しいタ スクセットにトラフィックを再ルーティングします。CodeDeploy は、各タスクセットに使用される ターゲットグループを決定します。

ロードバランサーは、リスナーを使用してターゲットグループにトラフィックをルーティングしま す。本稼働リスナーが1つ必要です。検証テストの実行中、置き換えタスクセットにトラフィック をルーティングする、オプションのテストリスナーを指定できます。

ロードバランサーでは、2 つのパブリックサブネットを別々のアベイラビリティーゾーンに持つ VPC を使用する必要があります。以下のステップでは、デフォルト VPC を確認し、Amazon EC2 Application Load Balancer を作成し、ロードバランサーに 2 つのターゲットグループを作成する方法 を示します。詳細については、「<u>Network Load Balancer のターゲットグループ</u>」を参照してくださ い。

デフォルト VPC、パブリックサブネット、およびセキュリティグループの確認

このトピックでは、Amazon ECS のデプロイ中に使用できる Amazon EC2 Application Load Balancer、2 つのターゲットグループ、および 2 つのポートを作成する方法を示します。ポートの 1 つはオプションであり、デプロイ中に検証テスト用のテストポートにトラフィックをルーティングす る場合にのみ必要です。

- 1. にサインイン AWS Management Console し、Amazon VPC コンソールを <u>https://</u> console.aws.amazon.com/vpc/://www.com で開きます。
- 使用するデフォルト VPC を確認します。ナビゲーションペインで、[Your VPCs(お使いの VPC)]を選択します。[デフォルト VPC] 列に [はい] と表示されているVPCに注意してくださ い。これがデフォルトの VPC になります。これには、使用するデフォルトのサブネットが含ま れます。
- [サブネット]を選択します。[デフォルトサブネット] 列で [はい] と表示されている 2 つのサブ ネットのサブネット ID をメモしておきます。これらの ID は、ロードバランサーを作成すると きに使用します。
- 各サブネットを選択し、[Description (説明)] タブを選択します。使用するサブネットが、異なる アベイラビリティーゾーンにあることを確認します。
- 5. サブネットを選択後、[Route Table] タブを選択します。使用する各サブネットがパブリックサ ブネットであることを確認するには、インターネットゲートウェイへのリンクのある行がルート テーブルに含まれていることを確認します。

- 6. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- 7. ナビゲーションペインで、[Security Groups] を選択します。
- 8. 使用するセキュリティグループが使用可能であることを確認し、そのグループ ID ([sgabcd1234] など)を書き留めます。これは、ロードバランサーを作成するときに使用します。

Amazon EC2 Application Load Balancer、2 つのターゲットグループ 、およびリス ナーを作成します (コンソール)

Amazon EC2 コンソールを使用して Amazon EC2 Application Load Balancer を作成するには:

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/ec2/</u>:// www.com」で Amazon EC2 コンソールを開きます。
- 2. ナビゲーションペインで、[ロードバランサー]を選択します。
- 3. [Create Load Balancer] を選択します。
- 4. [Application Load Balancer] を選択し、[Create] を選択します。
- 5. [Name] に、ロードバランサーの名前を入力します。
- 6. [Scheme] で、[インターネット向け] を選択します。
- 7. [IP address type] で、[ipv4] を選択します。
- (オプション)ロードバランサーの2番目のリスナーポートを設定します。このポートに提供 されるテストトラフィックを使用して、デプロイの検証テストを実行できます。
 - a. [Load Balancer Protocol (ロードバランサーのプロトコル)] で、[Add listener (リスナーの追加)] を選択します。
 - b. 2番目のリスナーの [Load Balancer Protocol] で、[HTTP] を選択します。
 - c. [Load Balancer Port (ロードバランサーポート)] に [8080] を入力します。
- 9. アベイラビリティーゾーン の VPCで、デフォルトの VPC を選択し、使用する 2 つのデフォル トサブネットを選択します。
- 10. [Next: Configure Security Settings] を選択します。
- 11. [Next: Configure Security Groups] を選択します。
- 12. [Select an existing security group (既存のセキュリティグループを選択する)] を選択し、デフォ ルトのセキュリティグループを選択して、その ID を書き留めます。
- 13. [Next: Configure Routing] を選択します。
- 14. [Target group (ターゲットグループ)] で、[New target group (新しいターゲットグループ)] を選択 し、最初のターゲットグループを設定します。
 - a. [Name] に、ターゲットグループの名前 (例: target-group-1) を入力します。
 - b. [Target type] で、[IP] を選択します。
 - c. [Protocol] で、[HTTP] を選択します。[Port] に「80」と入力します。
 - d. [Next: Register Targets] を選択します。
- 15. [Next: Review]、[Create] の順に選択します。

ロードバランサーの2番目のターゲットグループを作成するには

- 1. ロードバランサーがプロビジョニングされたら、Amazon EC2 コンソールを開きます。ナビ ゲーションペインで、[ターゲットグループ] を選択します。
- 2. [ターゲットグループの作成]を選択します。
- 3. [Name] に、ターゲットグループの名前 (例: target-group-2) を入力します。
- 4. [Target type] で、[IP] を選択します。
- 5. [Protocol] で、[HTTP] を選択します。[Port] に「80」と入力します。
- 6. [VPC] で、デフォルトの VPC を選択します。
- 7. [Create] (作成)を選択します。

Note

Amazon ECS デプロイを実行するには、ロードバランサー用に 2 つのターゲットグルー プを作成する必要があります。Amazon ECS サービスを作成するときに、いずれかの ターゲットグループの ARN を使用します。詳細については、Amazon ECS ユーザーガ イド の「<u>ステップ 4: Amazon ECS サービスを作成する</u>」を参照してください。

Amazon EC2 Application Load Balancer、2 つのターゲットグループ、およびリス ナーを作成します (CLI)

AWS CLIを使用して Application Load Balancer を作成するには

[ロードバランサーの作成]コマンドを使用して、Application Load Balancer を作成します。異なるアベイラビリティーゾーンにある2つのサブネット、およびセキュリティグループを指定します。

```
aws elbv2 create-load-balancer --name bluegreen-alb \
--subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --
region us-east-1
```

出力には、次の形式でロードバランサーの Amazon リソースネーム (ARN) が含まれます。

arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/ e5ba62739c16e642

 最初のターゲットグループを作成するには、<u>create-target-group</u> コマンドを使用しま す。CodeDeploy は、このターゲットグループのトラフィックを、サービスの元のタスクセット または置き換えタスクセットにルーティングします。

aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1

出力には、以下の形式で最初のターゲットグループの ARN が含まれます。

arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/ bluegreentarget1/209a844cd01825a4

 2つ目のターゲットグループを作成するには、create-target-group コマンドを使用しま す。CodeDeployは、最初のターゲットグループによって処理されないタスクセットにターゲッ トグループのトラフィックをルーティングします。たとえば、最初のターゲットグループが元の タスクセットにトラフィックをルーティングする場合、このターゲットグループは置き換えタス クセットにトラフィックをルーティングします。

aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1

出力には、以下の形式で2番目のターゲットグループの ARN が含まれます。

arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/ bluegreentarget2/209a844cd01825a4

4. <u>create-listener</u> コマンドを使用して、本稼働トラフィックをポート 80 に転送するデフォルト ルールを持つリスナーを作成します。 aws elbv2 create-listener --load-balancer-arn
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 --region us-east-1

出力には、以下の形式でリスナーの ARN が含まれます。

arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/ e5ba62739c16e642/665750bec1b03bd4

5. (オプション)<u>create-listener</u> コマンドを使用して、テストトラフィックをポート 8080 に転送 するデフォルトルールを持つ 2 番目のリスナーを作成します。このポートで提供されるテスト トラフィックを使用して、デプロイの検証テストを実行できます。



出力には、以下の形式でリスナーの ARN が含まれます。

arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/ e5ba62739c16e642/665750bec1b03bd4

デプロイグループの作成 (CLI)

を使用してデプロイグループ AWS CLI を作成するには、<u>create-deployment-group</u> コマンドを呼び 出し、以下を指定します。

 アプリケーション名。アプリケーション名のリストを表示するには、[list-applications] コマンドを 呼び出します。

- デプロイグループの名前。指定したアプリケーションに対して、この名前でデプロイグループが作成されます。デプロイグループは、1つのアプリケーションにのみ関連付けることができます。
- デプロイグループに含めるインスタンスを識別するタグ、タググループ、または Amazon EC2 Auto Scaling グループ名に関する情報。
- CodeDeploy が他の AWS サービスとやり取りするときに AWS アカウントに代わって動作できる ようにするサービスロールの Amazon リソースネーム (ARN) 識別子。サービスロール ARN を取 得するには、「<u>サービスロール ARN の取得 (CLI)</u>」を参照してください。サービスロールの詳細 については、IAM ユーザーガイド の「ロールに関する用語と概念」を参照してください。
- デプロイグループに関連付けるデプロイのタイプ(インプレースまたは Blue/Green)についての 情報。
- (オプション) 既存のデプロイ設定の名前。デプロイ設定のリストを表示するには、<u>View</u> <u>Deployment Configuration Details</u> を参照してください。指定されない場合、CodeDeploy は、デ フォルトのデプロイ設定を使用します。
- (オプション) Amazon Simple Notification Service トピックに登録しているユーザーにデプロイと インスタンスのイベントに関する通知をプッシュするトリガーを作成するコマンド。詳細について は、「Monitoring Deployments with Amazon SNS Event Notifications」を参照してください。
- ・ (オプション) アラームで指定したメトリクスが定義済みのしきい値を下回る/上回るとアクティブ 化される、既存の CloudWatch アラームをデプロイグループに追加するコマンド。
- (オプション) デプロイが失敗した時、または CloudWatch アラームが作動した時に、既知の正常な リビジョンのデプロイにロールバックするコマンド。
- (オプション) Auto Scaling スケールインイベント中にライフサイクルイベントフックを生成する デプロイのコマンド。詳細については、「<u>Amazon EC2 Auto Scaling と CodeDeploy の連携</u>」を 参照してください。
- インプレースデプロイの場合:
 - (オプション) デプロイプロセスでインスタンスへのトラフィックを管理する、Elastic Load Balancing の Classic Load Balancer、Application Load Balancer、または Network Load Balancer の名前。
- Blue/Green デプロイの場合。
 - ・ Blue/Green デプロイプロセスの設定。
 - 置き換え先環境の新しいインスタンスをプロビジョニングする方法。
 - トラフィックを置き換え先環境にすぐに再ルーティングするか、またはトラフィックを手動で 再ルーティングするために指定された期間待機するか。
 - 元の環境内のインスタンスを終了するかどうか。

 ・置き換え先環境で登録されたインスタンスに使用する Elastic Load Balancing の Classic Load Balancer、Application Load Balancer または Network Load Balancer の名前。

▲ Warning

デプロイグループで Auto Scaling グループと Elastic Load Balancing ロードバランサーの 両方を設定していて、<u>ロードバランサーを Auto Scaling グループにアタッチする場合は</u>、 このデプロイグループから CodeDeploy デプロイを作成する前にこのアタッチメントを完 了することをお勧めします。デプロイを作成した後にアタッチメントを完了しようとする と、すべてのインスタンスがロードバランサーから予期せず登録解除される可能性があり ます。

CodeDeploy を使用したデプロイグループの詳細の表示

CodeDeploy コンソール、、または CodeDeploy APIs を使用して AWS CLI、アプリケーションに関 連付けられたすべてのデプロイグループの詳細を表示できます。

トピック

- ・ デプロイグループの詳細の表示 (コンソール)
- ・ デプロイグループの詳細の表示 (CLI)

デプロイグループの詳細の表示 (コンソール)

CodeDeploy コンソールを使用してデプロイグループの詳細を表示するには:

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. [アプリケーション] ページで、デプロイグループに関連付けられたアプリケーション名を選択し ます。

デプロイグループの詳細の表示

i Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しま す。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレンス」の 「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを選択しま す。CodeDeploy は、これらのリージョンでのみサポートされています。

 個別のデプロイグループに関する詳細を表示するには、[デプロイグループ] タブで、デプロイグ ループの名前を選択します。

デプロイグループの詳細の表示 (CLI)

を使用してデプロイグループの詳細 AWS CLI を表示するには、 get-deployment-group コマンドまたは list-deployment-groups コマンドを呼び出します。

1 つのデプロイの詳細を表示するには、コマンド <u>デプロイグループ入手</u> を呼び出して、以下を指定 します。

- デプロイグループに関連付けられたアプリケーション名。アプリケーション名を取得するには、 [リストアプリケーション] コマンドを呼び出します。
- デプロイグループ名。デプロイグループ名を取得するには、[リストデプロイグループ] コマンドを 呼び出します。

デプロイグループ名のリストを表示するには、 [<u>リストデプロイグループ</u>] コマンドを呼び出して、 デプロイグループに関連付けられたアプリケーション名を指定します。アプリケーション名を取得す るには、<u>リストアプリケーション</u> コマンドを呼び出します。

CodeDeploy を使用して、デプロイグループの設定を変更します。

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、デプロイグループの設 定を変更できます。

A Warning

デプロイグループが、まだ作成していないカスタムデプロイグループを使用する場合、これ らの手順を使用しないでください。代わりに、「Create a Deployment Configuration」の手 順に従って、このトピックに戻ります。デプロイグループが、別のまだ作成していないサー ビスロールを使用する場合、これらの手順を使用しないでください。サービスロールは、最 低でも、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」で説明されているアク セス権限を持つ CodeDeploy を信頼する必要があります。正しいアクセス許可を持つサービ スロールを作成し、設定するには、「<u>ステップ 2: CodeDeployのサービスのロールを作成す</u> <u>る</u>」の手順に従って、このトピックに戻ります。

トピック

- デプロイグループの設定を変更する (コンソール)
- ・ <u>デプロイグループの設定を変更する (CLI)</u>

デプロイグループの設定を変更する (コンソール)

CodeDeploy コンソールを使用してデプロイグループの設定を変更するには。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- 3. アプリケーションのリストで、変更するデプロイグループに関連付けられているアプリケーションの名前を選択します。

Note

エントリが表示されない場合、正しいリージョンが選択されていることを確認してくだ さい。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレンス」の 「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを選択しま す。CodeDeploy は、これらのリージョンでのみサポートされています。

- 4. [デプロイグループ] タブを選択し、変更するデプロイグループの名前を選択します。
- 5. [デプロイグループ] ページで、[編集] を選択します。
- 6. 必要に応じてデプロイグループのオプションを編集します。

デプロイグループのコンポーネントの詳細については、「<u>CodeDeploy でデプロイグループを作成</u> する」を参照してください。

7. [Save changes] (変更の保存) をクリックします。

デプロイグループの設定を変更する (CLI)

を使用して AWS CLI デプロイグループの設定を変更するには、<u>update-deployment-group</u> コマンド を呼び出し、以下を指定します。

- ・ EC2/オンプレミスおよび AWS Lambda デプロイの場合:
 - アプリケーション名。アプリケーション名のリストを表示するには、[list-applications] コマンド を呼び出します。
 - 現在のデプロイグループ名。デプロイグループ名のリストを表示するには、[list-deploymentgroups] コマンドを呼び出します。
 - (オプション)別のデプロイグループ名。
 - (オプション) CodeDeploy が他の サービスとやり取りするときに AWS アカウントに代わって 動作できるようにする AWS サービスロールに対応する別の Amazon リソースネーム (ARN)。 サービスロール ARN を取得するには、「<u>サービスロール ARN の取得 (CLI)</u>」を参照してくだ さい。サービスロールの詳細については、IAM ユーザーガイド の「<u>ロールに関する用語と概</u> 念」を参照してください。
 - (オプション) デプロイ設定の名前。デプロイ設定のリストを表示するには、<u>View Deployment</u> <u>Configuration Details</u> を参照してください。(指定されない場合、CodeDeploy は、デフォルトの デプロイ設定を使用します。)
 - (オプション) アラームで指定されたメトリクスが定義したしきい値を下回る、または、超える場合にアクティベートするデプロイグループに1つ以上の既存 CloudWatch アラームを追加する コマンド。
 - (オプション) デプロイが失敗した時、または CloudWatch アラームがアクティブ化された時に、
 既知の正常なリビジョンのデプロイにロールバックするコマンド。
 - (オプション) Auto Scaling スケールインイベント中にライフサイクルイベントフックを生成 するデプロイのコマンド。詳細については、「<u>Amazon EC2 Auto Scaling と CodeDeploy の連</u> 携」を参照してください。
 - (オプション) Amazon Simple Notification Service のトピックに発行するトリガーを作成また は更新するコマンドにより、そのトピックのサブスクライバーがこのデプロイグループのデプ

ロイおよびインスタンスイベントに関する通知を受け取ります。詳細については、<u>Monitoring</u> Deployments with Amazon SNS Event Notifications を参照してください。

- EC2/オンプレミスのデプロイのみ:
 - (オプション) デプロイグループに含まれるインスタンスを一意に識別する代替タグまたはタググ ループ。
 - ・ (オプション) デプロイグループに追加する代替の Amazon EC2 Auto Scaling グループの名前。
- Amazon ECS のデプロイのみの場合:
 - ・ デプロイする Amazon ECS サービス
 - Application Load Balancer または Network Load Balancer を含むロードバランサーの情報、Amazon ECS デプロイに必要なターゲットグループ、および本番稼働用とオプションのテストリスナー情報。

デプロイグループの詳細オプションの設定

デプロイグループを作成または更新する場合は、そのデプロイグループのデプロイをより詳細に制御 および監視できるように、数多くのオプションを設定できます。

このページの情報を使用して、次のトピックで、デプロイグループを使用するときに詳細オプション を設定できます。

- CodeDeploy でアプリケーションを作成する
- CodeDeploy でデプロイグループを作成する
- CodeDeploy を使用して、デプロイグループの設定を変更します。

Amazon SNS 通知トリガー: CodeDeploy デプロイグループにトリガーを追加すると、そのデプロイ グループでのデプロイに関連するイベントに関する通知を受信できます。これらの通知は、トリガー のアクションの一部にした Amazon SNS トピックをサブスクライブする受信者に送信されます。

このトリガーが指す Amazon SNS トピックを既に設定している必要があり、CodeDeploy には、こ のデプロイグループからトピックに発行するためのアクセス許可が必要です。これらのセットアップ 手順をまだ完了していない場合は、後でデプロイグループにトリガーを追加できます。

このアプリケーションのデプロイグループのデプロイイベントに関する通知を受信するトリガーを今 すぐ作成する場合は、[Create trigger] を選択します。 Amazon EC2 インスタンスにデプロイする場合、インスタンスの通知を作成して、関連する通知を 受け取ることができます。

詳細については、「<u>Monitoring Deployments with Amazon SNS Event Notifications</u>」を参照してくだ さい。

Amazon CloudWatch アラーム: 指定した期間にわたって単一のメトリクスを監視し、複数の期間に わたり所定の閾値に対するメトリクスの相対値に基づいて 1 つまたは複数のアクションを実行する CloudWatch アラームを作成できます。Amazon EC2 デプロイの場合は、CodeDeploy オペレーショ ンで使用しているインスタンスまたは Amazon EC2 Auto Scaling グループに対してアラームを作成 できます。 AWS Lambda および Amazon ECS デプロイでは、Lambda 関数のエラーのアラームを 作成できます。

Amazon CloudWatch アラームが、メトリクスが定義された閾値を下回ったり上回ったりすることを 検出すると、デプロイが停止するように設定することができます。

デプロイグループに追加する前に、CloudWatch でアラームを作成しておく必要があります。

- 1. デプロイグループにアラームモニタリングを追加するには、[アラーム]、[アラームの追加] の順に 選択します。
- 2. このデプロイをモニタリングするためにセットアップ済みの CloudWatch アラームの名前を入力 します。

CloudWatch で作成されたのとまったく同じ CloudWatch アラームを入力する必要があります。ア ラームのリストを表示するには、<u>https://console.aws.amazon.com/cloudwatch/</u> で CloudWatch コ ンソールを開いて、[ALARM] を選択します。

追加のオプション:

追加したアラームを想定せずにデプロイを続行する場合は、[Ignore alarm configuration]を選択します。

この選択は、後で同じアラームを再び追加しなくても、デプロイグループのアラームの監視を一時 的に非アクティブ化する場合に便利です。

 (オプション) CodeDeploy が Amazon CloudWatch からアラームステータスを取得できないときで もデプロイを続行する場合は、[アラームの状態が利用できない場合でも、デプロイを続行する] を 選択してください。 Note

このオプションは、CodeDeploy API での <u>AlarmConfiguration</u> オブジェクトの ignorePollAlarmFailure に相当します。

詳細については、「<u>CodeDeploy での CloudWatch アラームを使用したデプロイのモニタリング</u>」を 参照してください。

自動ロールバック: デプロイが失敗した場合、または監視しきい値が満たされた場合に、自動的に ロールバックするようにデプロイグループまたはデプロイを設定できます。この場合、アプリケー ションリビジョンの最後の既知の正常なバージョンがデプロイされます。コンソールを使用してアプ リケーションを作成する場合、デプロイグループを作成する場合、またはデプロイグループを更新す る場合、デプロイグループのオプション設定を設定できます。新しいデプロイを作成するとき、デプ ロイグループに指定された自動ロールバック設定をオーバーライドすることもできます。

- 次のいずれかまたは両方を選択して何か問題が発生した場合、デプロイを有効化して最新の既知の正常なリビジョンにロールバックすることができます。
 - デプロイが失敗したときにロールバックする。CodeDeployは、新しいデプロイとして、最後の既知の正常なリビジョンを再デプロイします。
 - アラームのしきい値が一致したときにロールバックする。前のステップでこのアプリケーションにアラームを追加した場合、CodeDeployは、指定された1つ以上のアラームがアクティブ化されたときに、最後の既知の正常なリビジョンを再デプロイします。

Note

ロールバック設定を一時的に無視するには、[Disable rollbacks] を選択します。この選択 は、後で再び同じ設定をセットアップせずに自動ロールバックを一時的に無効にする場 合に便利です。

詳細については、「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照 してください。 古いインスタンスの自動更新:特定の状況下では、CodeDeploy がアプリケーションの古いリビジョ ンを Amazon EC2 インスタンスにデプロイすることがあります。たとえば、CodeDeploy デプロイ の進行中に EC2 インスタンスが Auto Scaling グループ (ASG) で起動された場合、それらのインス タンスには、最新ではなく古いリビジョンのアプリケーションが表示されます。これらのインスタ ンスを最新の状態にするために、CodeDeploy は自動的に最初のインスタンスの直後にフォローオン のデプロイを開始し、古いインスタンスを更新します。古い EC2 インスタンスを古いリビジョンに 残すようにこのデフォルトの動作を変更する場合は、CodeDeploy API または AWS Command Line Interface (CLI) を使用して変更できます。

API を使用して古くなったインスタンスの自動更新を設定するに

は、outdatedInstancesStrategy または UpdateDeploymentGroup のアクションに CreateDeploymentGroup リクエストパラメータを含めます。詳細については、「AWS CodeDeploy API リファレンス」を参照してください。

を使用して自動更新を設定するには AWS CLI、次のいずれかのコマンドを使用します。

aws deploy update-deployment-group arguments --outdated-instances-strategy
UPDATE | IGNORE

または...

aws deploy create-deployment-group arguments --outdated-instances-strategy
UPDATE | IGNORE

... ここで、## は、デプロイに必要な引数に置き換えられ、UPDATE / IGNORE は自動更新を有効にす る場合は UPDATE、無効にする場合は IGNORE に置き換わります。

例:

aws deploy update-deployment-group --application-name "MyApp" --currentdeployment-group-name "MyDG" --region us-east-1 --outdated-instancesstrategy IGNORE

これらの AWS CLI 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。

CodeDeploy でデプロイグループを削除する

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、 AWS アカウントに関 連付けられたデプロイグループを削除できます。

▲ Warning

デプロイグループを削除すると、そのデプロイグループに関連付けられるすべての詳細もま た CodeDeploy から削除されます。デプロイグループで使用するインスタンスは変更されま せん。このアクションを元に戻すことはできません。

トピック

- デプロイグループを削除する (コンソール)
- デプロイグループを削除する (CLI)

デプロイグループを削除する (コンソール)

CodeDeploy コンソールを使用してデプロイグループを削除するには。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note
 「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- アプリケーションのリストで、デプロイグループに関連付けられるアプリケーションの名前を選択します。
- 4. [アプリケーションの詳細] ページの [デプロイグループ] タブで、削除するデプロイグループの名 前を選択します。

5. [デプロイの詳細] ページで、[削除] を選択します。

6. 求められたら、デプロイグループの名前を入力して削除を確認してから、[Delete] を選択します。

デプロイグループを削除する (CLI)

を使用してデプロイグループ AWS CLI を削除するには、<u>delete-deployment-group</u> コマンドを呼び 出し、以下を指定します。

- デプロイグループに関連付けられたアプリケーションの名前。アプリケーション名のリストを表示 するには、[list-applications] コマンドを呼び出します。
- アプリケーションに関連付けられたデプロイグループの名前。デプロイグループ名のリストを表示 するには、[list-deployment-groups] コマンドを呼び出します。

CodeDeploy のアプリケーションリビジョンの操作

CodeDeploy では、リビジョンには CodeDeploy がインスタンスにデプロイするソースファイルの バージョン、または CodeDeploy がインスタンスで実行するスクリプトが含まれています。

リビジョンを計画し、このリビジョンに AppSpec ファイルを追加してから、リビジョンを Amazon S3 または GitHub にプッシュします。リビジョンをプッシュしたら、デプロイできます。

トピック

- CodeDeploy のリビジョンを計画する
- CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追加
- CodeDeploy リポジトリタイプを選択する
- Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オンプレミスのデプロイのみ)
- CodeDeploy を使用したアプリケーションリビジョン詳細の表示
- CodeDeploy で Amazon S3 にアプリケーションリビジョンを登録する

CodeDeploy のリビジョンを計画する

良い計画は、リビジョンのデプロイをより簡単にします。

AWS Lambda または Amazon ECS コンピューティングプラットフォームへのデプロイの場合、 リビジョンは AppSpec ファイルと同じです。次の情報は適用されません。詳細については、 「<u>CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追加</u>」を参照してください。

EC2/オンプレミスコンピューティングプラットフォームにデプロイする場合、開発マシンで空の ルートディレクトリ (フォルダ) を作成することから始めます。これはインスタンスで実行するイン スタンスまたはスクリプトにデプロイするソースファイル (テキストファイルやバイナリファイル、 実行ファイル、パッケージなど) を保存します。

例えば、Linux、macOS、Unix の /tmp/ のルートフォルダ、または Windows の c : \temp のルート フォルダ:

	mySourceFile.rb
	myExecutableFile.exe
	myInstallerFile.msi
	myPackage.rpm
	myImageFile.png
s	cripts (subfolder)
	myShellScript.sh
	myBatchScript.bat
	myPowerShellScript.ps1
l a	

また、ルートフォルダには、以下に示すとおり、アプリケーション仕様ファイル (AppSpec ファイ ル) が含まれている必要があります。詳細については、「<u>CodeDeploy 用のアプリケーション仕様</u> <u>ファイルをリビジョンに追加</u>」を参照してください。

CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追加

このトピックでは、AppSpec ファイルをデプロイに追加する方法について説明します。また、 AWS Lambda および EC2/オンプレミスデプロイ用の AppSpec ファイルを作成するためのテンプレートも 含まれています。

トピック

- Amazon ECS デプロイ用の AppSpec ファイルを追加する
- AWS Lambda デプロイ用の AppSpec ファイルを追加する
- ・ <u>EC2/オンプレミスデプロイに AppSpec ファイルを追加する</u>

Amazon ECS デプロイ用の AppSpec ファイルを追加する

Amazon ECS コンピューティングプラットフォームへのデプロイの場合:

- AppSpec ファイルは、デプロイに使用される Amazon ECS タスク定義、トラフィックをルーティングするコンテナ名とポートマッピング、およびオプションとしてデプロイライフサイクルイベントの後で実行される Lambda 関数を指定します。
- ・ リビジョンは、AppSpec ファイルと同じです。
- AppSpec ファイルは、JSON または YAML を使用して書き込むことができます。

 AppSpec ファイルは、テキストファイルとして保存するか、デプロイ作成時にコンソールに直接 入力することができます。詳細については、「<u>Amazon ECS コンピューティングプラットフォー</u> ムのデプロイの作成 (コンソール)」を参照してください。

AppSpec ファイルを作成するには

- 1. JSON または YAML テンプレートをテキストエディタ、またはコンソールの AppSpec エディタ にコピーします。
- 2. 必要に応じてテンプレートを変更します。
- JSON または YAML バリデータを使用して、AppSpec ファイルを検証します。AppSpec エディ タを使用している場合は、[デプロイの作成] を選択すると、このファイルが検証されます。
- チキストエディタを使用している場合は、ファイルを保存します。を使用してデプロイ AWS CLI を作成する場合は、ハードドライブまたは Amazon S3 バケットにある AppSpec ファイル を参照してください。コンソールを使用している場合は、AppSpec ファイルを Amazon S3 に プッシュする必要があります。

Amazon ECS デプロイ用の手順付き YAML AppSpec ファイルテンプレート

以下に示しているのは、利用できるすべてのオプションを含む Amazon ECS デプロイ用の AppSpec ファイルの YAML テンプレートです。hooks セクションで使用するライフサイクルイベントについ ては、「<u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション</u>」を参照してください。

```
# This is an appspec.yml template file for use with an Amazon ECS deployment in
CodeDeploy.
# The lines in this template that start with the hashtag are
#
   comments that can be safely left in the file or
#
   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#
   "CodeDeploy User Guide" at
   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
#
version: 0.0
# In the Resources section, you must specify the following: the Amazon ECS service,
task definition name,
# and the name and port of the load balancer to route traffic,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - TargetService:
      Type: AWS::ECS::Service
```

Properties:

```
TaskDefinition: "" # Specify the ARN of your task definition
 (arn:aws:ecs:region:account-id:task-definition/task-definition-family-name:task-
definition-revision-number)
       LoadBalancerInfo:
          ContainerName: "" # Specify the name of your Amazon ECS application's
 container
          ContainerPort: "" # Specify the port for your container where traffic
 reroutes
# Optional properties
        PlatformVersion: "" # Specify the version of your Amazon ECS Service
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["",""] # Specify one or more comma-separated subnets in your
 Amazon ECS service
            SecurityGroups: ["",""] # Specify one or more comma-separated security
 groups in your Amazon ECS service
            AssignPublicIp: "" # Specify "ENABLED" or "DISABLED"
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event.
Hooks:
# Hooks for Amazon ECS deployments are:
    - BeforeInstall: "" # Specify a Lambda function name or ARN
    - AfterInstall: "" # Specify a Lambda function name or ARN
    - AfterAllowTestTraffic: "" # Specify a Lambda function name or ARN
    - BeforeAllowTraffic: "" # Specify a Lambda function name or ARN
    - AfterAllowTraffic: "" # Specify a Lambda function name or ARN
```

Amazon ECS デプロイテンプレート用の JSON AppSpec ファイル

以下に示しているのは、利用できるすべてのオプションを含む Amazon ECS デプロイ用の AppSpec ファイルの JSON テンプレートです。テンプレートの使用方法については、前のセクションの YAML バージョンのコメントを参照してください。hooks セクションで使用するライフサイクルイ ベントについては、「<u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション</u>」を参照し てください。

```
"LoadBalancerInfo": {
        "ContainerName": "",
        "ContainerPort":
       },
       "PlatformVersion": "",
       "NetworkConfiguration": {
        "AwsvpcConfiguration": {
         "Subnets": [
          "",
          .....
         ],
         "SecurityGroups": [
          "",
          .....
         ],
         "AssignPublicIp": ""
        }
       }
     }
   }
  }
 ],
 "Hooks": [
  {
   "BeforeInstall": ""
  },
  {
   "AfterInstall": ""
  },
  {
   "AfterAllowTestTraffic": ""
  },
  {
  "BeforeAllowTraffic": ""
  },
  {
   "AfterAllowTraffic": ""
  }
 ]
}
```

AWS Lambda デプロイ用の AppSpec ファイルを追加する

AWS Lambda コンピューティングプラットフォームへのデプロイの場合:

- AppSpec ファイルには、デプロイされる Lambda 関数に関する手順が含まれており、デプロイ検 証で使用されます。
- ・ リビジョンは、AppSpec ファイルと同じです。
- AppSpec ファイルは、JSON または YAML を使用して書き込むことができます。
- AppSpec ファイルは、テキストファイルとして保存するか、デプロイ作成時にコンソールの AppSpec エディタに直接入力することができます。詳細については、「<u>AWS Lambda コンピュー</u> <u>ティングプラットフォームのデプロイの作成 (コンソール)</u>」を参照してください。

AppSpec ファイルを作成するには

- 1. JSON または YAML テンプレートをテキストエディタ、またはコンソールの AppSpec エディタ にコピーします。
- 2. 必要に応じてテンプレートを変更します。
- JSON または YAML バリデータを使用して、AppSpec ファイルを検証します。AppSpec エディ タを使用している場合は、[デプロイの作成] を選択すると、このファイルが検証されます。
- チキストエディタを使用している場合は、ファイルを保存します。を使用してデプロイ AWS CLI を作成する場合は、ハードドライブまたは Amazon S3 バケットにある AppSpec ファイル を参照してください。コンソールを使用している場合は、AppSpec ファイルを Amazon S3 に プッシュする必要があります。

AWS Lambda デプロイ用の手順付き YAML AppSpec ファイルテンプレート

hooks セクションで使用するライフサイクルイベントについては、「<u>AWS の Lambda デプロイ向け</u> の AppSpec の「hooks」セクション」を参照してください。

This is an appspec.yml template file for use with an AWS Lambda deployment in CodeDeploy. # The lines in this template starting with the hashtag symbol are # instructional comments and can be safely left in the file or # ignored. # For help completing this file, see the "AppSpec File Reference" in the

```
#
   "CodeDeploy User Guide" at
#
   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section specify the name, alias,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - MyFunction: # Replace "MyFunction" with the name of your Lambda function
      Type: AWS::Lambda::Function
      Properties:
        Name: "" # Specify the name of your Lambda function
        Alias: "" # Specify the alias for your Lambda function
        CurrentVersion: "" # Specify the current version of your Lambda function
       TargetVersion: "" # Specify the version of your Lambda function to deploy
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event. Replace "LifeCycleEvent" with BeforeAllowTraffic
# or AfterAllowTraffic.
Hooks:
    - LifeCycleEvent: "" # Specify a Lambda validation function between double-quotes.
```

AWS Lambda デプロイテンプレートの JSON AppSpec ファイル

以下のテンプレートで、"MyFunction" を AWS Lambda 関数名に置き換えます。オプションの Hooks セクションで、ライフサイクルイベントを BeforeAllowTraffic または AfterAllowTraffic に置き換えま す。

Hooks セクションで使用するライフサイクルイベントについては、「<u>AWS の Lambda デプロイ向け</u> の AppSpec の「hooks」セクション」を参照してください。

}] }

EC2/オンプレミスデプロイに AppSpec ファイルを追加する

AppSpec ファイルがないと、CodeDeploy は、アプリケーションリビジョンのソースファイルを送 信先にマッピングしたり、EC2/オンプレミスコンピューティングプラットフォームにデプロイする ためのスクリプトを実行したりできません。

各リビジョンには、AppSpec ファイルを1つだけ含める必要があります。

AppSpec ファイルをリビジョンに追加するには:

- 1. テンプレートにテキストエディターをコピーします。
- 2. 必要に応じてテンプレートを変更します。
- 3. YAML バリデータを使用して、AppSpec ファイルの有効性をチェックします。
- 4. リビジョンのルートディレクトリに appspec.yml としてファイルを保存します。
- 5. AppSpec ファイルをルートディレクトリに配置したことを確認するには、次のいずれかのコマ ンドを実行します。
 - Linux、macOS、Unix の場合:

find /path/to/root/directory -name appspec.yml

AppSpec ファイルがその場所に見つからない場合は、何も出力されません。

• Windows の場合:

dir path\to\root\directory\appspec.yml

そこに AppSpec ファイルが保存されていない場合は、[File Not Found] エラーが表示されます。

6. リビジョンを Amazon S3 または GitHub にプッシュします。

手順については、<u>Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オンプレミス</u>のデプロイのみ)を参照してください。

EC2/オンプレミスデプロイ用の手順付き AppSpec ファイルテンプレート

Note

Windows Server インスタンスへのデプロイでは、runas 要素をサポートしていません。Windows Server インスタンスにデプロイする場合は、AppSpec ファイルに含めないでください。

```
# This is an appspec.yml template file for use with an EC2/On-Premises deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
    instructional comments and can be safely left in the file or
#
#
    ignored.
# For help completing this file, see the "AppSpec File Reference" in the
    "CodeDeploy User Guide" at
#
   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
#
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux,
    Red Hat Enterprise Linux (RHEL), or Ubuntu Server
#
#
    instances.
# Specify "os: windows" if this revision targets Windows Server instances.
# (You cannot specify both "os: linux" and "os: windows".)
os: linux
# os: windows
# During the Install deployment lifecycle event (which occurs between the
    BeforeInstall and AfterInstall events), copy the specified files
#
#
   in "source" starting from the root of the revision's file bundle
   to "destination" on the Amazon EC2 instance.
#
# Specify multiple "source" and "destination" pairs if you want to copy
   from multiple sources or to multiple destinations.
#
# If you are not copying any files to the Amazon EC2 instance, then remove the
#
   "files" section altogether. A blank or incomplete "files" section
   may cause associated deployments to fail.
#
files:
  - source:
    destination:
  - source:
    destination:
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#
   you can specify a "permissions"
#
    section here that describes special permissions to apply to the files
```

```
#
    in the "files" section as they are being copied over to
#
    the Amazon EC2 instance.
    For more information, see the documentation.
#
# If you are deploying to Windows Server instances,
#
   then remove the
    "permissions" section altogether. A blank or incomplete "permissions"
#
    section may cause associated deployments to fail.
#
permissions:
  - object:
    pattern:
    except:
    owner:
    group:
    mode:
    acls:
      _
    context:
      user:
      type:
      range:
    type:
# If you are not running any commands on the Amazon EC2 instance, then remove
#
   the "hooks" section altogether. A blank or incomplete "hooks" section
#
   may cause associated deployments to fail.
hooks:
# For each deployment lifecycle event, specify multiple "location" entries
    if you want to run multiple scripts during that event.
#
# You can specify "timeout" as the number of seconds to wait until failing the
 deployment
    if the specified scripts do not run within the specified time limit for the
#
    specified event. For example, 900 seconds is 15 minutes. If not specified,
#
   the default is 1800 seconds (30 minutes).
#
   Note that the maximum amount of time that all scripts must finish executing
#
#
   for each individual deployment lifecycle event is 3600 seconds (1 hour).
#
   Otherwise, the deployment will stop and CodeDeploy will consider the deployment
    to have failed to the Amazon EC2 instance. Make sure that the total number of
#
 seconds
#
   that are specified in "timeout" for all scripts in each individual deployment
    lifecycle event does not exceed a combined 3600 seconds (1 hour).
#
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
   you can specify "runas" in an event to
#
#
   run as the specified user. For more information, see the documentation.
#
   If you are deploying to Windows Server instances,
```

```
remove "runas" altogether.
#
# If you do not want to run any commands during a particular deployment
    lifecycle event, remove that event declaration altogether. Blank or
#
    incomplete event declarations may cause associated deployments to fail.
#
# During the ApplicationStop deployment lifecycle event, run the commands
    in the script specified in "location" starting from the root of the
#
#
    revision's file bundle.
  ApplicationStop:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the BeforeInstall deployment lifecycle event, run the commands
    in the script specified in "location".
#
  BeforeInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the AfterInstall deployment lifecycle event, run the commands
    in the script specified in "location".
#
  AfterInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ApplicationStart deployment lifecycle event, run the commands
#
    in the script specified in "location".
  ApplicationStart:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ValidateService deployment lifecycle event, run the commands
    in the script specified in "location".
#
  ValidateService:
```

- location: timeout: runas:
- location: timeout: runas:

CodeDeploy リポジトリタイプを選択する

CodeDeploy に必要なファイルのストレージの場所は リポジトリと呼ばれます。使用するリポジト リは、デプロイで使用するコンピューティングプラットフォームによって異なります。

- EC2/オンプレミス: 1 つ以上のインスタンスにアプリケーションコードをデプロイするには、コードをアーカイブファイルにバンドルし、これをデプロイ処理中に CodeDeploy がアクセスできるリポジトリに配置する必要があります。デプロイ可能なコンテンツと AppSpec ファイルをアーカイブファイルにバンドルしてから、CodeDeploy でサポートされているいずれかのレポジトリタイプにアップロードします。
- AWS Lambda と Amazon ECS: デプロイには AppSpec ファイルが必要です。これは、デプロイ中 に次のいずれかの方法でアクセスできます。
 - Amazon S3 バケットから。
 - ・コンソールの AppSpec エディタへ直接入力されたテキストから 詳細については、「<u>AWS</u> <u>Lambda コンピューティングプラットフォームのデプロイの作成 (コンソール)</u>」および 「<u>Amazon ECS コンピューティングプラットフォームのデプロイの作成 (コンソール)</u>」を参照 してください。
 - を使用する場合は AWS CLI、ハードドライブまたはネットワークドライブにある AppSpec ファ イルを参照できます。詳細については、「<u>AWS Lambda コンピューティングプラットフォーム</u> <u>のデプロイの作成 (CLI)</u>」および「<u>Amazon ECS コンピューティングプラットフォームのデプ</u> ロイの作成 (CLI)」を参照してください。
- 現在、CodeDeploy は次のリポジトリタイプをサポートしています。

リポジトリタイプ	リポジトリの詳細	サポートされているコン ピューティングプラット フォーム
Amazon S3	<u>Amazon Simple Storage</u> <u>Service</u> (Amazon S3) は、安	以下のコンピューティングプ ラットフォームを使用するデ

全でスケーラブルなオブジェ クトストレージ向けの AWS ソリューションです。Amaz on S3 は、データをオブジェ クトとして バケット に保存し ます。オブジェクトは、ファ イルと、オプションとしてそ のファイルを記述する任意の メタデータで構成されていま す。

Amazon S3 にオブジェクト を保存するには、バケットに ファイルをアップロードしま す。ファイルをアップロード する際に、オブジェクトにア クセス権限とメタデータを設 定することができます。

詳細はこちら:

- <u>Amazon S3 にバケットを作</u> 成する
- <u>Amazon S3 に CodeDeploy</u>
 <u>のリビジョンをプッシュす</u>
 <u>る (EC2/オンプレミスのデ</u>
 <u>プロイのみ)</u>
- <u>CodeDeploy を使用した</u>
 <u>Amazon S3 からの自動的な</u>
 <u>デプロイ</u>

プロイでは、Amazon S3 バ ケット にリビジョンを保存で きます。

- ・ EC2/オンプレミス
- AWS Lambda
- Amazon ECS

EC2/オンプレミスデプロイで

ジョンを保存できます。

GitHub

アプリケーションリビジョン を GitHub リポジトリに保存 のみ GitHub リポジトリにリビ できます。そのリポジトリの ソースコードが変更されるた びに、GitHub リポジトリか らデプロイをトリガーできま す。

詳細はこちら:

- GitHub と CodeDeploy との 統合
- チュートリアル: CodeDeplo yを使用して、GitHub から アプリケーションをデプロ イします。

Bitbucket	Bitbucket Pipelines の CodeDeploy パイプ を使用 すると、EC2 インスタンス のデプロイグループにコー ドをデプロイすることがで きます。Bitbucket Pipelines は、Bitbucket Deployments など、継続的インテグレー ションと継続的デリバリー (CI/CD) 機能を提供します。 CodeDeploy パイプはまず、 指定された S3 バケットに アーティファクトをプッシュ し、次にバケットからコード アーティファクトをデプロイ します。 詳細はこちら: ・Bitbucket の CodeDeploy パ	EC2/オンプレミスデプロイで のみ BitBucket リポジトリに リビジョンを保存できます。
	 Bitbucket の CodeDeploy パ イプを参照する 	

Note

AWS Lambda デプロイは Amazon S3 リポジトリでのみ機能します。

Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オ ンプレミスのデプロイのみ)

「CodeDeploy のリビジョンを計画する」で説明されているようにリビジョンを計画して、

「<u>CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追加</u>」で説明されているように AppSpec ファイルをリビジョンに追加したら、コンポーネントファイルをバンドルして、リビジョ ンを Amazon S3 にプッシュすることができます。Amazon EC2 インスタンスにデプロイする場合 は、リビジョンをプッシュしたら、CodeDeploy を使用して、Amazon S3 からインスタンスへのリ ビジョンをデプロイできます。 Note

CodeDeploy は、GitHub にプッシュされたリビジョンのデプロイにも使用できます。詳細に ついては、GitHub のドキュメントを参照してください。

「<u>CodeDeploy の開始方法</u>」で説明している AWS CLIのセットアップの指示に従っていることを前 提としています。これは、後で説明する push コマンドを呼び出す場合に特に重要です。

Amazon S3 バケットがあることを確認してください。バケットの作成の手順に従います。

Amazon EC2 インスタンスにデプロイする場合は、ターゲットの Amazon S3 バケットを作成する か、そのバケットがターゲット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。例えば、 米国東部 (バージニア北部) リージョンのインスタンスと米国西部 (オレゴン) リージョンの他のイン スタンスにリビジョンをデプロイしたい場合は、米国東部 (バージニア北部) リージョンのバケット にリヒジョンのコピーを一つ、米国西部 (オレゴン) の別のバケットに同じリビジョンの別のコピー を持っている必要があります。このシナリオでは、両方のリージョンとバケットでリビジョンが同 じであっても、米国東部 (バージニア北部) リージョンに 1 つ、米国西部 (オレゴン) リージョンに別 の、2 つの別々のデプロイを作成する必要があります。

Amazon S3 バケットへのアップロードには、許可が必要です。Amazon S3 バケットポリシーで、これらのアクセス許可を指定できます。たとえば、次の Amazon S3 バケットポリシーでは、ワイルド カード文字 (*) を使用すると、 AWS アカウント111122223333は という名前の Amazon S3 バケッ ト内の任意のディレクトリにファイルをアップロードできますamzn-s3-demo-bucket。

```
{
    "Statement": [
        {
            "Action": [
            "s3:PutObject"
        ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                 "111122223333"
            ]
        }
     }
     ]
```

}

AWS アカウント ID を表示するには、AWS 「アカウント ID の検索」を参照してください。

Amazon S3 バケットポリシーを生成しアタッチする方法については、「<u>バケットポリシーの例</u>」を 参照してください。

push コマンドを呼び出すユーザーは、少なくとも、各ターゲット Amazon S3 バケットにリビジョ ンをアップロードするアクセス権限が必要です。例えば、次のポリシーでは、ユーザーが amzns3-demo-bucket という名前の Amazon S3 バケット内の任意の場所でリビジョンをアップロード できるようにします。

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

IAM ポリシーを作成しアタッチする方法については、「ポリシーの使用」を参照してください。

を使用してリビジョンをプッシュする AWS CLI

Note

push コマンドは、アプリケーションのアーティファクトおよび AppSpec ファイルをリビ ジョンにバンドルします。このリビジョンのファイル形式は、圧縮された ZIP ファイルで す。各 は JSON 形式または YAML 形式の AppSpec ファイルであるリビジョンを想定してい るため、 AWS Lambda または Amazon ECS デプロイで コマンドを使用することはできま せん。

push コマンドを呼び出してリビジョンをバンドルしてプッシュし、デプロイします。パラメータは 次のとおりです。

- --application-name: (文字列) 必須。アプリケーションバージョンに関連付けられる CodeDeploy ア プリケーションの名前。
- --s3-location: (文字列) 必須。Amazon S3 にアップロードされるアプリケーションリビジョンの 場所に関する情報。Amazon S3 バケットとキーを指定する必要があります。キーは、リビジョ ンの名前です。CodeDeploy はコンテンツをアップロードする前に zip 形式で圧縮します。形式 s3://amzn-s3-demo-bucket/your-key.zip を使用します。
- --ignore-hidden-files または --no-ignore-hidden-files: (プール値) オプション。--no-ignore-hidden-files フラグ (デフォルト) を使用して、隠しファイルをバンドルし Amazon S3 にアップロードします。隠しファイルをバンドルして Amazon S3 へアップロードしない --ignore-hidden-files フラグを使用する。
- --source(文字列) オプション。zip 圧縮され Amazon S3 にアップロードされる開発マシン上の、 デプロイするコンテンツおよび AppSpec ファイルの場所。この場所は、現在のディレクトリに 対する相対パスとして指定されます。相対パスが指定されていない場合、または 1 つのピリオド (「.」) がパスとして使用される場合、現在のディレクトリが使用されます。
- --description(文字列) オプション。アプリケーションリビジョンを要約したコメントです。指定しない場合、デフォルトの文字列「Uploaded by AWS CLI 'time' UTC」が使用されます。ここで、「time」は協定世界時 (UTC) の現在のシステム時刻です。

を使用して AWS CLI、Amazon EC2 デプロイのリビジョンをプッシュできます。プッシュコマンド の例は、次のようになります。

Linux、macOS、または Unix の場合:

```
aws deploy push \
    --application-name WordPress_App \
    --description "This is a revision for the application WordPress_App" \
    --ignore-hidden-files \
    --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
    --source .
```

Windows の場合:

aws deploy push --application-name WordPress_App --description "This is a revision for the application WordPress_App" --ignore-hidden-files --s3-location s3://amzn-s3-demobucket/WordPressApp.zip --source .

このコマンドは次のことを行います。

- バンドルされたファイルを WordPress_App という名前のアプリケーションに関連付けます。
- リビジョンに説明をアタッチします。
- 隠しファイルを無視します。
- リビジョンに WordPressApp.zip という名前を付け、amzn-s3-demo-bucket というバケット にプッシュします。
- ルートディレクトリ内のすべてのファイルをリビジョンにバンドルします。

プッシュが成功したら、 AWS CLI または CodeDeploy コンソールを使用して Amazon S3 からリビ ジョンをデプロイできます。を使用してこのリビジョンをデプロイするには AWS CLI:

Linux、macOS、または Unix の場合:

aws deploy create-deployment \
 --application-name WordPress_App \

- --application-name wordfiess_App (
- --deployment-config-name your-deployment-config-name \
- --deployment-group-name your-deployment-group-name \
- --s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip

Windows の場合:

aws deploy create-deployment --application-name WordPress_App --deployment-configname your-deployment-config-name --deployment-group-name your-deployment-group-name -s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip

詳細については、「CodeDeploy でデプロイを作成する」を参照してください。

CodeDeploy を使用したアプリケーションリビジョン詳細の表示

CodeDeploy コンソール、、または CodeDeploy APIs を使用して AWS CLI、指定したアプリケー ションの AWS アカウントに登録されているすべてのアプリケーションリビジョンの詳細を表示でき ます。

リビジョンの登録の詳細については、「<u>CodeDeploy で Amazon S3 にアプリケーションリビジョン</u> を登録する」を参照してください。

トピック

- アプリケーションリビジョンの詳細の表示 (コンソール)
- アプリケーションリビジョンの詳細の表示 (CLI)

アプリケーションリビジョンの詳細の表示 (コンソール)

アプリケーションリビジョンの詳細を表示するには:

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。

Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しましょ う。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレンス」の 「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを選択しま す。CodeDeploy は、これらのリージョンでのみサポートされています。

- 3. 表示するリビジョンと関連するアプリケーションの名前を選択します。
- (アプリケーションの詳細)ページで [リビジョン] タブを選択し、アプリケーションに登録されているリビジョンを一覧表示します。リビジョンを選択し、[詳細を表示]を選択します。

アプリケーションリビジョンの詳細の表示 (CLI)

を使用してアプリケーションリビジョン AWS CLI を表示するには、 get-application-revision コマンドまたは list-application-revisions コマンドを呼び出します。

Note

GitHub のリファレンスは、EC2 オンプレミスデプロイに対するデプロイにのみ適用されます。 AWS Lambda デプロイのリビジョンは GitHub では機能しません。

単一のアプリケーションリビジョンに関する詳細を表示するには、<u>get-application-revision</u> コマンド を呼び出し、以下を指定します。

- アプリケーション名。アプリケーション名を取得するには、<u>list-applications</u> コマンドを呼び出します。
- GitHub に保存されたリビジョンの場合、GitHub レポジトリ名と、リポジトリにプッシュされたアプリケーションリビジョンを参照するコミットの ID。
- Amazon S3 に保存されたリビジョンの場合、リビジョンを含む Amazon S3 バケット名、アップ ロードされたアーカイブファイルの名前とファイル形式、さらにオプションで、アーカイブファ イルの Amazon S3 バージョン ID と ETag。バージョン IDかETag、またはその両方を registerapplication-revision の呼び出し中に指定した場合、ここで指定する必要があります。

複数のアプリケーションリビジョンに関する詳細を表示するには、<u>list-application-revisions</u> コマンド を呼び出し、以下を指定します。

- アプリケーション名。アプリケーション名を取得するには、<u>list-applications</u> コマンドを呼び出します。
- オプションで、Amazon S3 アプリケーションリビジョン詳細のみを表示する場合、リビジョンを 含む Amazon S3 バケット名。
- オプションで、Amazon S3 アプリケーションリビジョンの詳細のみを表示する場合、 Amazon S3 アプリケーションリビジョンの検索を制限するプレフィックス文字列 (指定されていない場 合、CodeDeploy は一致するすべての Amazon S3 アプリケーションリビジョンを表示します。)
- オプションで、各リビジョンがデプロイグループのターゲットリビジョンかどうかに基づいてリビジョンの詳細を表示するかどうか(指定されていない場合、CodeDeployは一致するすべてのリビジョンを表示します。)
- オプションで、リビジョンの詳細の一覧をソートする際の基準とする列名と順序(指定されていない場合、CodeDeployは任意の順序で結果を表示します。)

すべてのリビジョン、または Amazon S3 に保存されたリビジョンのみを表示できます。GitHub に 保存されているリビジョンのみを表示することはできません。

CodeDeploy で Amazon S3 にアプリケーションリビジョンを登録 する

既に push コマンドを呼び出してアプリケーションリビジョンを Amazon S3 にプッシュしている場合、リビジョンを登録する必要はありません。ただし、他の方法で Amazon S3 にリビジョンをアッ プロードし、リビジョンを CodeDeploy コンソールに表示するか、 AWS CLIを通じて表示する場合 は、次のステップに従って最初にリビジョンを登録します。 アプリケーションリビジョンを GitHub リポジトリにプッシュし、そのリビジョンを CodeDeploy コ ンソールまたは で表示する場合は AWS CLI、以下のステップも実行する必要があります。

Amazon S3 AWS CLI または GitHub でアプリケーションリビジョンを登録するには、 または CodeDeploy APIs のみを使用できます。 Amazon S3

トピック

- CodeDeploy で Amazon S3 にリビジョンを登録する (CLI)
- CodeDeploy による GitHub でのリビジョンの登録 (CLI)

CodeDeploy で Amazon S3 にリビジョンを登録する (CLI)

- 1. Amazon S3 にリビジョンをアップロードする
- 2. register-application-revision コマンドを呼び出し、以下を指定します。
 - アプリケーション名。アプリケーション名のリストを表示するには、[list-applications] コマンドを呼び出します。
 - ・ 登録するリビジョンに関する情報:
 - ・ リビジョンを含む Amazon S3 バケットの名前。
 - アップロードされたリビジョンの名前とファイル形式。AWS Lambda デプロイの場合、リビジョンは、JSON または YAML で書き込まれた AppSpec ファイルです。EC2/オンプレミスデプロイの場合、このリビジョンには、CodeDeploy からインスタンスにデプロイされるソースファイルか、CodeDeploy によってインスタンスで実行されるスクリプトのバージョンが含まれます。

1 Note

tar および圧縮 tar アーカイブファイル形式 (.tar および .tar.gz) は、Windows Server インスタンスではサポートされていません。

- (オプション) リビジョンの Amazon S3 バージョン識別子。(バージョン識別子を指定しない 場合、CodeDeploy は最新バージョンを使用します)。
- ・ (オプション) のリビジョン ETag。(ETag が指定されていない場合、CodeDeploy はオブ ジェクトの検証をスキップします)
- (オプション) リビジョンに関連付ける説明。
Amazon S3 のリビジョンに関する情報は、register-application-revision 呼び出しの一部としてこの構 文を使用して、コマンドラインで指定できます。(version および eTag はオプションです。)

EC2/オンプレミスのデプロイ向けリビジョンファイルの場合

--s3-location bucket=string,key=string,bundleType=tar|tgz|
zip,version=string,eTag=string

AWS Lambda デプロイのリビジョンファイルの場合:

--s3-location bucket=*string*, key=*string*, bundleType=JSON|YAML, version=*string*, eTag=*string*

CodeDeploy による GitHub でのリビジョンの登録 (CLI)

Note

AWS Lambda デプロイは GitHub では機能しません。

- 1. GitHub リポジトリにリビジョンをアップロードします。
- 2. register-application-revision コマンドを呼び出し、以下を指定します。
 - アプリケーション名。アプリケーション名のリストを表示するには、[list-applications] コマンドを呼び出します。
 - ・ 登録するリビジョンに関する情報:
 - スラッシュ (/)、リポジトリの名前が後に続く、リポジトリに割り当てられたリビジョンを 含む GitHub ユーザーまたは組織の名前。
 - リポジトリのリビジョンを参照するコミットの ID。
 - (オプション) リビジョンに関連付ける説明。

GitHub のリビジョンについての情報は、この構文を register-application-revision の一部として使用 することでコマンドラインで指定できます。

--github-location repository=*string*, commitId=*string*

CodeDeploy でのデプロイグループの使用

CodeDeploy では、デプロイはプロセスであり、プロセスには 1 つ以上のインスタンスのコンテンツ をインストールするコンポーネントが含まれます。このコンテンツはコード、ウェブ、設定ファイ ル、実行可能ファイル、パッケージ、スクリプトなどによって構成できます。CodeDeploy は、指定 した設定ルールに従って、ソースリポジトリに格納されたコンテンツをデプロイします。

EC2 オンプレミスコンピューティングプラットフォームを使用する場合、インスタンスの同じセットへの2 つのデプロイは同時に実行できます。

CodeDeploy は、2 つのデプロイタイプのオプション、インプレースデプロイおよび Blue/Green デ プロイを提供します。

- インプレイスデプロイ: デプロイグループの各インスタンス上のアプリケーションが停止され、最新のアプリケーションリビジョンがインストールされて、新バージョンのアプリケーションが開始され検証されます。ロードバランサーを使用し、デプロイ中はインスタンスが登録解除され、デプロイ完了後にサービスに復元されるようにできます。EC2 オンプレミスコンピューティングプラットフォームを使用するデプロイのみが、インプレイスデプロイを使用できます。インプレイスデプロイの講細については、「インプレースデプロイの概要」を参照してください。
- Blue/Green デプロイ: デプロイの動作は、使用するコンピューティングプラットフォームにより異なります。
 - EC2 オンプレミスコンピューティングプラットフォームの Blue/Green: 以下のステップを使用して、デプロイグループのインスタンス (元の環境) がインスタンスの別のセット (置き換え先環境) に置き換えられます。
 - ・置き換え先の環境のインスタンスがプロビジョニングされます。
 - 最新のアプリケーションリビジョンは、置き換え先インスタンスにインストールされます。
 - オプションの待機時間は、アプリケーションのテストやシステム検証などのアクティビティに 対して発生します。
 - ・置き換え先環境のインスタンスは、1 つまたは複数の Elastic Load Balancing ロードバラン サーに登録され、トラフィックは、それらに再ルーティングされます。元の環境のインスタン スは、登録が解除され、終了するか、他の使用のために実行することができます。

Note

EC2/オンプレミスのコンピューティングプラットフォームを使用する場合は、blue/ green デプロイが Amazon EC2 インスタンスでのみ機能することに注意してください。

- AWS Lambda または Amazon ECS コンピューティングプラットフォームの Blue/Green: トラ フィックは、Canary、線形、または all-at-onceデプロイ設定に従って増分でシフトされます。
- 経由のブルー/グリーンデプロイ AWS CloudFormation: スタック AWS CloudFormation の更新の 一環として、トラフィックは現在のリソースから更新されたリソースに移行されます。現時点で は、ECS blue/green デプロイのみがサポートされています。

ブルー/グリーンデプロイの詳細については、「<u>Blue/Green デプロイの概要</u>」を参照してください。

Amazon S3 からの自動的なデプロイの詳細については、「<u>CodeDeployを使用して Amazon S3 から</u> 自動的にデプロイする」を参照してください。

トピック

- CodeDeploy でデプロイを作成する
- CodeDeploy デプロイの詳細を表示する
- ・ CodeDeploy EC2/オンプレミスデプロイのログデータの表示
- <u>CodeDeploy でデプロイの停止</u>
- CodeDeploy を使用した再デプロイおよびデプロイのロールバック
- 異なる AWS アカウントでアプリケーションをデプロイする
- CodeDeploy エージェントを使用してローカルマシン上のデプロイパッケージを検証する

CodeDeploy でデプロイを作成する

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、既に Amazon S3 に プッシュしたアプリケーションリビジョンをインストールするデプロイを作成できます。デプロイが EC2/オンプレミスコンピューティングプラットフォームの場合は、デプロイグループのインスタン スに GitHub をインストールします。

デプロイを作成するプロセスは、デプロイで使用されるコンピューティングプラットフォームによっ て異なります。

トピック

- デプロイの前提条件
- Amazon ECS コンピューティングプラットフォームのデプロイの作成 (コンソール)
- AWS Lambda コンピューティングプラットフォームのデプロイの作成 (コンソール)

- ・ EC2/オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)
- Amazon ECS コンピューティングプラットフォームのデプロイの作成 (CLI)
- AWS Lambda コンピューティングプラットフォームのデプロイの作成 (CLI)
- EC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)
- を使用して Amazon ECS ブルー/グリーンデプロイを作成する AWS CloudFormation

デプロイの前提条件

デプロイを開始する前に、以下のステップが完了していることを確認します。

AWS Lambda コンピューティングプラットフォームのデプロイ前提条件

- ・ 少なくとも1つのデプロイグループを含むアプリケーションを作成します。詳細については、 「<u>CodeDeployでアプリケーションを作成する</u>」および「<u>CodeDeployでデプロイグループを作成</u> <u>する</u>」を参照してください。
- デプロイする Lambda 関数バージョンを指定するアプリケーションリビジョン (AppSpec ファイ ルとも呼ばれる)を準備します。AppSpec ファイルでは、Lambda 関数を指定してデプロイを検 証することもできます。詳細については、CodeDeployのアプリケーションリビジョンの操作 を 参照してください。
- デプロイにカスタムデプロイ設定を使用する場合、デプロイプロセスを開始する前にカスタムデプ ロイ設定を作成します。詳細については、「<u>Create a Deployment Configuration</u>」を参照してくだ さい。

EC2/オンプレミスコンピューティングプラットフォームのデプロイ前提条件

- インプレースデプロイの場合は、デプロイ先のインスタンスを作成または設定します。詳細については、「<u>CodeDeploy のためにインスタンスを用いた操作</u>」を参照してください。Blue/Greenデプロイのために、置き換え先環境のテンプレートとして使用する既存の Amazon EC2 Auto Scaling グループがあるか、元の環境として指定する 1 つ以上のインスタンスまたは Amazon EC2 Auto Scaling グループがあります。詳細については、「<u>チュートリアル: CodeDeploy を使用して、Auto Scaling グループにアプリケーションをデプロイします。</u>」および「<u>CodeDeploy と Amazon EC2 Auto Scaling の統合</u>」を参照してください。
- ・ 少なくとも1つのデプロイグループを含むアプリケーションを作成します。詳細については、 「<u>CodeDeployでアプリケーションを作成する</u>」および「<u>CodeDeployでデプロイグループを作成</u>する」を参照してください。

- デプロイグループのインスタンスにデプロイするアプリケーションリビジョンを準備します。詳細 については、「CodeDeployのアプリケーションリビジョンの操作」を参照してください。
- デプロイにカスタムデプロイ設定を使用する場合、デプロイプロセスを開始する前にカスタムデプ ロイ設定を作成します。詳細については、「<u>Create a Deployment Configuration</u>」を参照してくだ さい。
- Amazon S3 バケットからアプリケーションリビジョンをデプロイする場合、バケットはデプロイ グループのインスタンスと同じ AWS リージョンにあります。
- Amazon S3 バケットからアプリケーションのリビジョンをデプロイする場合、Amazon S3 バケットポリシーをバケットに適用済みです。このポリシーでは、アプリケーションリビジョンをダウンロードするために必要なアクセス許可をインスタンスに付与します。

例えば、次の Amazon S3 バケットポリシーは、ARN arn:aws:iam::444455556666:role/ CodeDeployDemo を含む IAM インスタンスプロファイルがアタッチされた Amazon EC2 インス タンスが、amzn-s3-demo-bucket という名前の Amazon S3 バケットの任意の場所からダウン ロードすることを許可します。

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                 "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                 "AWS": [
                     "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

次の Amazon S3 バケットポリシーは、ARN arn:aws:iam::444455556666:user/ CodeDeployUser を含む IAM ユーザーに関連付けられたオンプレミスインスタンスが、amzns3-demo-bucket という名前の Amazon S3 バケット内の任意の場所からダウンロードすること を許可します。

```
{
    "Statement": [
        {
            "Action": [
                 "s3:Get*",
                 "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                 "AWS": [
                     "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

Amazon S3 バケットポリシーを生成しアタッチする方法の詳細については、「<u>バケットポリシー</u>の例」を参照してください。

 Blue/Green デプロイを作成する場合、またはインプレースデプロイのためにデプロイグループに オプションの Classic Load Balancer、Application Load Balancer、Network Load Balancer を指定 している場合、Amazon VPC を使用して、少なくとも2つのサブネットを含む VPC を作成して いることになります。(CodeDeploy で使用する Elastic Load Balancing では、ロードバランサーグ ループ内のすべてのインスタンスが1つの VPC 内にあることが必要です)。

VPC を作成済みでない場合は、「Amazon VPC 入門ガイド」を参照してください。

 ブルー/グリーンデプロイを作成する場合、Elastic Load Balancing に、少なくとも1つの Classic Load Balancer、Application Load Balancer または Network Load Balancerを設定し、これを使用 して元の環境を構成するインスタンスを登録しています。

Note

置き換え先環境内のインスタンスは後でロードバランサーを使用して登録されます。

ロードバランサーの設定の詳細については、「<u>CodeDeploy Amazon EC2 デプロイ用の Elastic</u> Load Balancing でロードバランサーをセットアップする」および「<u>CodeDeploy Amazon ECS デ</u>

デプロイの前提条件

<u>プロイ用のロードバランサー、ターゲットグループ、リスナーをセットアップする</u>」を参照してく ださい。

を使用した Blue/Green デプロイのデプロイ前提条件 AWS CloudFormation

- テンプレートでは、CodeDeploy アプリケーションまたはデプロイグループのリソースをモデル化 する必要はありません。
- テンプレートには、少なくとも2つのサブネットを含む Amazon VPC を使用する VPC のリソー スを含める必要があります。
- テンプレートには、トラフィックをターゲットグループに誘導するために使用される Elastic Load Balancing の1つまたは複数の Classic Load Balancer、Application Load Balancer、または Network Load Balancer のリソースを含める必要があります。

Amazon ECS コンピューティングプラットフォームのデプロイの作成 (コ ンソール)

このトピックでは、コンソールを使用して Amazon ECS サービスをデプロイする方法を示します。 詳細については、「<u>チュートリアル: Amazon ECS ヘアプリケーションをデプロイする</u>」および 「<u>チュートリアル: 検証テストを使用して Amazon ECS サービスをデプロイする</u>」を参照してくだ さい。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note

「<u>CodeDeploy の開始方法</u>」で設定したのと同じユーザーでサインインします。

- 2. 次のいずれかを行います:
 - アプリケーションをデプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[ア プリケーション] を選択します。デプロイするアプリケーションの名前を選択します。アプリ ケーションの [Compute platform (コンピューティングプラットフォーム)] 列が [Amazon ECS] になっていることを確認します。
 - デプロイを再デプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[デプロイ]
 を選択します。再デプロイするデプロイを選択し、[アプリケーション] 列で、アプリケーショ

ンの名前を選択します。デプロイの [Compute platform (コンピューティングプラットフォーム)] 列が [Amazon ECS] になっていることを確認します。

3. [デプロイ] タブで、[デプロイの作成] を選択します。

Note

アプリケーションをデプロイするには、アプリケーションにデプロイグループが必要で す。アプリケーションにデプロイグループがない場合は、[デプロイグループ] タブで、 [デプロイグループの作成] を選択します。詳細については、「<u>CodeDeploy でデプロイ</u> <u>グループを作成する</u>」を参照してください。

- 4. [デプロイグループ] で、このデプロイで使用するデプロイグループを選択します。
- 5. [リビジョンの場所]の横で、リビジョンの場所を選択します。
 - [My application is stored in Amazon S3 (Amazon S3 に保存されたアプリケーション)] 詳細に 関しては、<u>Amazon S3 バケットに格納されているリビジョンについての情報を指定します</u>を 参照し、ステップ 6 に戻ります。
 - [Use AppSpec editor (AppSpec エディタの使用)] JSON または YAML を選択し、エディ タに AppSpec ファイルを入力します。AppSpec ファイルを保存するには、[テキストファ イルとして保存] を選択します。これらのステップの最後で [Deploy (デプロイ)] を選択する と、JSON または YAML が有効ではないというエラーが発生します。AppSpec ファイルの作 成の詳細については、「<u>CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追</u> 加」を参照してください。
- 6. (オプション) [デプロイの説明] に、このデプロイの説明を入力します。
- (オプション) [Rollback configuration overrides] で、該当する場合は、デプロイグループに指定されているものとは別の自動ロールバックオプションをこのデプロイに指定できます。

CodeDeploy のロールバックの詳細については、「<u>デプロイと再デプロイのロールバック</u>」および「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照してください。

次から選択します。

- [Roll back when a deployment fails (デプロイが失敗したときにロールバックする)] CodeDeploy は既知の正常なリビジョンを新しいデプロイとして再デプロイします。
- [Roll back when alarm thresholds are met (アラームのしきい値が一致したときにロールバック する)] — デプロイグループにアラームが追加された場合、1 つ以上の指定したアラームがアク ティブ化されたときに、CodeDeploy は既知の正常なリビジョンを再デプロイします。

- [ロールバックを無効にする] このデプロイのロールバックを実行しません。
- 8. [デプロイの作成]を選択します。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してくだ さい。

AWS Lambda コンピューティングプラットフォームのデプロイの作成 (コ ンソール)

このトピックでは、コンソールを使用して Lambda 関数をデプロイする方法を示します。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. 次のいずれかを行います:
 - アプリケーションをデプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[ア プリケーション] を選択します。デプロイするアプリケーションの名前を選択します。アプリ ケーションの [Compute platform (コンピューティングプラットフォーム)] 列が [AWS Lambda] になっていることを確認します。
 - デプロイを再デプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[デプロイ]
 を選択します。再デプロイするデプロイを選択し、[アプリケーション] 列で、アプリケーショ
 ンの名前を選択します。デプロイ の [Compute platform (コンピューティングプラットフォーム)] 列が [AWS Lambda] になっていることを確認します。
- 3. [デプロイ] タブで、[デプロイの作成] を選択します。

Note

アプリケーションをデプロイするには、アプリケーションにデプロイグループが必要で す。アプリケーションにデプロイグループがない場合は、[デプロイグループ] タブで、 [デプロイグループの作成] を選択します。詳細については、「<u>CodeDeploy でデプロイ</u> <u>グループを作成する</u>」を参照してください。

- 4. [デプロイグループ] で、このデプロイで使用するデプロイグループを選択します。
- 5. [リビジョンの場所]の横で、リビジョンの場所を選択します。
 - [My application is stored in Amazon S3 (Amazon S3 に保存されたアプリケーション)] 詳細に 関しては、<u>Amazon S3 バケットに格納されているリビジョンについての情報を指定します</u>を 参照し、ステップ 6 に戻ります。
 - [Use AppSpec editor (AppSpec エディタの使用)] JSON または YAML を選択し、エディ タに AppSpec ファイルを入力します。AppSpec ファイルを保存するには、[テキストファ イルとして保存] を選択します。これらのステップの最後で [Deploy (デプロイ)] を選択する と、JSON または YAML が有効ではないというエラーが発生します。AppSpec ファイルの作 成の詳細については、「<u>CodeDeploy 用のアプリケーション仕様ファイルをリビジョンに追</u> 加」を参照してください。
- 6. (オプション) [デプロイの説明] に、このデプロイの説明を入力します。
- (オプション) [Deployment group overrides (デプロイグループのオーバーライド)] を展開し、 トラフィックを Lambda 関数バージョンにシフトする方法を制御するデプロイ設定を選択しま す。この場合、デプロイグループに指定したものとは異なるデプロイ設定を選択します。

詳細については、「<u>AWS Lambda コンピューティングプラットフォームのデプロイ設定</u>」を参 照してください。

(オプション) [Rollback configuration overrides] で、該当する場合は、デプロイグループに指定されているものとは別の自動ロールバックオプションをこのデプロイに指定できます。

CodeDeploy のロールバックの詳細については、「<u>デプロイと再デプロイのロールバック</u>」およ び「CodeDeploy を使用した再デプロイおよびデプロイのロールバック」を参照してください。

次から選択します。

- [Roll back when a deployment fails (デプロイが失敗したときにロールバックする)] CodeDeploy は既知の正常なリビジョンを新しいデプロイとして再デプロイします。
- [Roll back when alarm thresholds are met (アラームのしきい値が一致したときにロールバック する)] — デプロイグループにアラームが追加された場合、1 つ以上の指定したアラームがアク ティブ化されたときに、CodeDeploy は既知の正常なリビジョンを再デプロイします。
- [ロールバックを無効にする] このデプロイのロールバックを実行しません。
- 9. [デプロイの作成]を選択します。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してくだ さい。

EC2/オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)

このトピックでは、コンソールを使用して Amazon EC2 またはオンプレミスサーバーにアプリケー ションをデプロイする方法を示します。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. 次のいずれかを行います:
 - アプリケーションをデプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[ア プリケーション] を選択します。デプロイするアプリケーションの名前を選択します。アプリ ケーションの [コンピューティングプラットフォーム] 列が [EC2/オンプレミス] になっている ことを確認します。
 - デプロイを再デプロイする場合は、ナビゲーションペインで [デプロイ] を展開し、[デプロイ]
 を選択します。再デプロイするデプロイを見つけ、[アプリケーション] 列で、アプリケーションの名前を選択します。デプロイ の [コンピューティングプラットフォーム] 列が [EC2/オンプレミス] になっていることを確認します。
- 3. [デプロイ] タブで、[デプロイの作成] を選択します。

Note アプリケーションをデプロイするには、アプリケーションにデプロイグループが必要で す。アプリケーションにデプロイグループがない場合は、[デプロイグループ] タブで、 [デプロイグループの作成] を選択します。詳細については、「<u>CodeDeploy でデプロイ</u> <u>グループを作成する</u>」を参照してください。

- 4. [デプロイグループ] で、このデプロイで使用するデプロイグループを選択します。
- 5. [リポジトリタイプ]の横で、リビジョンが保存されているリポジトリタイプを選択します。

- [My application is stored in Amazon S3 (Amazon S3 に保存されたアプリケーション)] 詳細に 関しては、<u>Amazon S3 バケットに格納されているリビジョンについての情報を指定します</u>を 参照し、ステップ 6 に戻ります。
- [My application is stored in GitHub (GitHub に保存されたアプリケーション)] 詳細について は、以下の「<u>GitHub リポジトリに格納されているリビジョンについての情報を指定します</u>」 を参照し、ステップ 6 に戻ります。
- 6. (オプション) [デプロイの説明] に、このデプロイの説明を入力します。
- (オプション) [Override deployment configuration (デプロイ設定の上書き)] を展開してデプロイ 設定を選択し、デプロイグループに指定したものとは異なる Amazon EC2 またはオンプレミス サーバーにトラフィックをシフトする方法を制御します。

詳細については、「CodeDeploy でデプロイ設定を使用する」を参照してください。

- 8. a. ApplicationStop ライフサイクルイベントが失敗した場合でもインスタンスへのデプロ イを成功させる場合は、[Don't fail the deployment if the ApplicationStop lifecycle event fails (ApplicationStop ライフサイクルイベントの障害でデプロイを失敗させない)] を選択しま す。
 - b. [Additional deployment behavior settings (追加のデプロイ動作設定)] を展開して、以前 に成功したデプロイに含まれていなかったデプロイターゲットの場所にあるファイルを CodeDeploy で処理する方法を指定します。

次から選択します。

- [Fail the deployment (デプロイの失敗)] エラーが報告され、デプロイのステータスが Failed に変更されます。
- [コンテンツの上書き] デプロイ先に同じ名前のファイルが存在する場合は、アプリケーションリビジョンのバージョンによって置き換えられます。
- [コンテンツの保持] デプロイ先に同じ名前のファイルが存在する場合は、ファイルが 保持され、アプリケーションリビジョンのバージョンはインスタンスにコピーされません。

詳細については、「既存のコンテンツでのロールバック動作」を参照してください。

9. (オプション) [Rollback configuration overrides] で、該当する場合は、デプロイグループに指定さ れているものとは別の自動ロールバックオプションをこのデプロイに指定できます。 CodeDeploy のロールバックの詳細については、「<u>デプロイと再デプロイのロールバック</u>」および「CodeDeploy を使用した再デプロイおよびデプロイのロールバック」を参照してください。

次から選択します。

- [Roll back when a deployment fails (デプロイが失敗したときにロールバックする)] CodeDeploy は既知の正常なリビジョンを新しいデプロイとして再デプロイします。
- [Roll back when alarm thresholds are met (アラームのしきい値が一致したときにロールバック する)] — デプロイグループにアラームが追加された場合、1 つ以上の指定したアラームがアク ティブ化されたときに、CodeDeploy は既知の正常なリビジョンをデプロイします。
- [ロールバックを無効にする] このデプロイのロールバックを実行しません。
- 10. デプロイの開始 を選択します。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してくだ さい。

トピック

- Amazon S3 バケットに格納されているリビジョンについての情報を指定します
- GitHub リポジトリに格納されているリビジョンについての情報を指定します

Amazon S3 バケットに格納されているリビジョンについての情報を指定します

「<u>EC2/オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)</u>」の手順を 使用している場合は、以下のステップに従って Amazon S3 バケットに保存されているアプリケー ションリビジョンの詳細を追加してください。

- 1. リビジョンの Amazon S3 リンクを [Revision location (リビジョンの場所)] ボックスにコピーし ます。リンク値の確認
 - a. 別のブラウザタブで

にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/s3/</u> で Amazon S3 コンソールを開きます。

リビジョンを参照して選択します。

b. [Properties] ペインが表示されていない場合、[Properties] ボタンを選択します。

c. [Properties] ペインで、[Link] フィールドの値をCodeDeploy コンソールの [Revision location] ボックスにコピーします。

ETag (ファイルチェックサム)をリビジョンの場所の一部として指定するには。

- [リンク] フィールド値が ?versionId=versionId で終わる場合、&etag= および ETag を [リンク] フィールド値の末尾に追加します。
- ・ [リンク] フィールド値がバージョン ID を指定していない場合、?etag= および ETag を [リンク] フィールド値の末尾に追加します。

(i) Note

[Link] フィールドの値をコピーするように簡単ではありませんが、次のいずれかの形式 でリビジョンの場所を入力することもできます。 s3://bucket-name/folders/objectName s3://bucket-name/folders/objectName?versionId=versionId s3://bucket-name/folders/objectName?etag=etag s3://bucket-name/folders/objectName?versionId=versionId&etag=etag bucket-name.s3.amazonaws.com/folders/objectName

[File type] リストに、ファイル形式を検出できないというメッセージが表示された場合は、リビジョンのファイル形式を選択します。それ以外の場合は、検出されたファイル形式を使用します。

GitHub リポジトリに格納されているリビジョンについての情報を指定します

「<u>EC2/オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)</u>」の手順を 使用している場合は、以下のステップに従って GitHub リポジトリに保存されているアプリケーショ ンリビジョンの詳細を追加してください。

- 1. [Connect to GitHub] で、次のいずれかを実行します。
 - GitHub アカウントに対する CodeDeploy アプリケーションの接続を作成するには、ウェブブ ラウザの別のタブで GitHub からサインアウトします。[GitHub アカウント] に、この接続を識 別する名前を入力し、[GitHub に接続] を選択します。アプリケーションの GitHub を操作す

ることを CodeDeploy に許可するよう求めるメッセージがウェブページに表示されます。ス テップ 2 に進みます。

- 作成済みの接続を使用するには、その名前を [GitHub account] で選択してから [Connect to GitHub] を選択します。ステップ 4 に進みます。
- 別の GitHub アカウントへの接続を作成するには、ウェブブラウザの別のタブで GitHub から サインアウトします。[Connect to a different GitHub account] を選択し、[Connect to GitHub] を選択します。ステップ2に進みます。
- 2. GitHub にサインインするよう求められたら、[Sign in] ページの手順に従います。GitHub のユー ザー名、または E メールとパスワードでサインインします。
- 3. [Authorize application] ページが表示された場合、[Authorize application] を選択します。
- [Create deployment (デプロイの作成)] ページの、[Repository name (リポジトリの名前)] ボックスに、スラッシュ (/)、リビジョンを含むリポジトリの名前が後に続く、リビジョンを含む GitHub ユーザーまたは組織の名前を入力します。入力する値が不確実な場合:
 - a. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
 - b. [Your repositories] で、ターゲットリポジトリの名前の上にマウスを置きます。GitHub ユー ザーまたは組織の名前、スラッシュ (/)、リポジトリの名前の順序でツールヒントが表示さ れます。[Repository name (リポジトリの名前)] ボックスに、この表示された値を入力しま す。

Note

[Your repositories] にターゲットリポジトリの名前が表示されない場合、[Search GitHub] ボックスを使用して、ターゲットリポジトリの名前と、GitHub ユーザーま たは組織の名前を検索します。

- 5. [Commit ID (コミットID)] で、リポジトリのリビジョンを参照するコミット ID を入力します。 入力する値が不確実な場合:
 - a. ウェブブラウザの別のタブで、GitHub dashboard にアクセスします。
 - b. [Your repositories] で、ターゲットコミットを含むリポジトリの名前を選択します。
 - c. コミットのリストで、リポジトリのリビジョンを参照するコミット ID を検索してコピーし ます。通常、この ID は 40 文字で、文字と数字の両方で構成されます (通常はコミット ID の長いバージョンの最初の 10 文字の、コミット ID の短いバージョンを使用しないでくだ さい)。
 - d. [Commit ID] ボックスにコミット ID を貼り付けます。

Amazon ECS コンピューティングプラットフォームのデプロイの作成 (CLI)

アプリケーションとリビジョンの作成後 (Amazon ECS のデプロイでは、これは AppSpec ファイル です)。

create-deployment コマンドを呼び出し、指定します。

- アプリケーション名。アプリケーション名のリストを表示するには、<u>list-applications</u> コマンドを 呼び出します。
- デプロイグループ名。デプロイグループ名のリストを表示するには、<u>list-deployment-groups</u>コマンドを呼び出します。
- デプロイするリビジョンに関する情報。

Amazon S3 に格納されているリビジョン。

- Amazon S3 バケットの名前がリビジョンに含まれています。
- アップロードされたリビジョンの名前。
- ・ (オプション) リビジョンの Amazon S3 バージョンのID。(バージョン ID を指定しない場合、CodeDeploy は最新バージョンを使用します。)
- ・ (オプション) リビジョンの ETag。(ETag が指定されていない場合、CodeDeploy はオブジェクトの検証をスキップします。)

Amazon S3 にないファイルに保存されているリビジョンの場合、ファイル名とパスが必要です。 リビジョンファイルは、YAML または JSON で書かれているため、ほとんどの場合、その拡張子 は .json または .yaml です。

・(オプション) デプロイの説明。

リビジョンファイルは Amazon S3 バケットにアップロードされるファイルまたは文字列として指定 できます。create-deployment コマンドの一部として使用する場合の各構文は次のようになります。

• Amazon S3 バケット:

versionおよびeTagはオプションです。

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

文字列:

--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'

Note

create-deployment コマンドはファイルからリビジョンをロードできます。詳細について は、「<u>ファイルからパラメータをロードする</u>」を参照してください。

AWS Lambda デプロイリビジョンテンプレートについては、「」を参照してください<u>AWS Lambda</u> <u>デプロイ用の AppSpec ファイルを追加する</u>。リビジョンの例については、「<u>AWS Lambda デプロ</u> イの AppSpec ファイルの例 」を参照してください。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してください。

AWS Lambda コンピューティングプラットフォームのデプロイの作成 (CLI)

アプリケーションとリビジョンを作成した後(AWS Lambda デプロイでは、これは AppSpec ファイ ルです)。

create-deployment コマンドを呼び出し、指定します。

- アプリケーション名。アプリケーション名のリストを表示するには、<u>list-applications</u> コマンドを 呼び出します。
- デプロイグループ名。デプロイグループ名のリストを表示するには、<u>list-deployment-groups</u>コマンドを呼び出します。
- デプロイするリビジョンに関する情報。

Amazon S3 に格納されているリビジョン。

- Amazon S3 バケットの名前がリビジョンに含まれています。
- アップロードされたリビジョンの名前。
- ・ (オプション) リビジョンの Amazon S3 バージョンのID。(バージョン ID を指定しない場合、CodeDeploy は最新バージョンを使用します。)

・ (オプション) リビジョンの ETag。(ETag が指定されていない場合、CodeDeploy はオブジェクトの検証をスキップします。)

Amazon S3 にないファイルに保存されているリビジョンの場合、ファイル名とパスが必要です。 リビジョンファイルは、YAML または JSON で書かれているため、ほとんどの場合、その拡張子 は .json または .yaml です。

- (オプション)使用するデプロイ設定の名前。デプロイ設定のリストを表示するには、<u>list-</u> <u>deployment-configs</u> コマンドを呼び出します。(指定されない場合、CodeDeploy は、特定のデ フォルトのデプロイ設定を使用します。)
- ・(オプション)デプロイの説明。

リビジョンファイルは Amazon S3 バケットにアップロードされるファイルまたは文字列として指定 できます。create-deployment コマンドの一部として使用する場合の各構文は次のようになります。

• Amazon S3 バケット:

versionおよびeTagはオプションです。

--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string

文字列:

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

Note

create-deployment コマンドはファイルからリビジョンをロードできます。詳細について は、「<u>ファイルからパラメータをロードする</u>」を参照してください。

AWS Lambda デプロイリビジョンテンプレートについては、「」を参照してください<u>AWS Lambda</u> <u>デプロイ用の AppSpec ファイルを追加する</u>。リビジョンの例については、「<u>AWS Lambda デプロ</u> イの AppSpec ファイルの例 」を参照してください。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してくださ い。

EC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)

を使用して EC2/オンプレミスコンピューティングプラットフォームにリビジョンを AWS CLI デプ ロイするには:

- 1. インスタンスを準備し、アプリケーションを作成して、リビジョンをプッシュした後、次のいず れかを実行します。
 - Amazon S3 バケットからリビジョンをデプロイする場合は、そのままステップ2に進みます。
 - GitHub リポジトリからリビジョンをデプロイする場合は、まず「CodeDeploy アプリケーションを GitHub リポジトリに接続します。」のステップを完了してからステップ 2 に戻ります。
- 2. create-deployment コマンドを呼び出し、指定します。
 - --application-name: アプリケーション名。アプリケーション名のリストを表示するには、list-applications コマンドを呼び出します。
 - --deployment-group-name: Amazon EC2 デプロイグループ名。デプロイグループ名のリストを表示するには、list-deployment-groupsコマンドを呼び出します。
 - --revision: デプロイするリビジョンに関する情報。

Amazon S3 に格納されているリビジョン。

- s3Location: リビジョンを含む Amazon S3 バケットの名前がリビジョンに含まれています。
- s3Location --> key: アップロードされたリビジョンの名前。
- s3Location --> bundleType: アップロードされたリビジョンの名前とファイル形式。

Note

tar および圧縮 tar アーカイブファイル形式 (.tar および .tar.gz) は、Windows Server インスタンスではサポートされていません。

- s3Location --> version: (オプション) リビジョンの Amazon S3 バージョン ID。(バージョン ID を指定しない場合、CodeDeploy は最新バージョンを使用します。)
- s3Location --> eTag: (オプション) リビジョンの ETag。(ETag が指定されていない場合、CodeDeploy はオブジェクトの検証をスキップします。)

GitHub で格納されたリビジョンの場合。

- gitHubLocation --> repository: スラッシュ (/)、リポジトリの名前が後に続く、リポジトリに割り当てられたリビジョンを含む GitHub ユーザーまたは組織の名前。
- gitHubLocation --> commitId: リビジョンのコミット ID。
- --deployment-config-name: (オプション) 使用するデプロイ設定の名前。デプロイ設定のリストを表示するには、<u>list-deployment-configs</u> コマンドを呼び出します。(指定されない場合、CodeDeploy は特定のデフォルトのデプロイ設定を使用します。)
- --ignore-application-stop-failures | --no-ignore-application-stopfailures:(オプション)BeforeInstall デプロイライフサイクルのイベントを続行してイ ンスタンスへのデプロイを続行するかどうか。たとえ ApplicationStop デプロイライフサイ クルイベントが失敗してもです。
- --description: (オプション) デプロイの説明。
- --file-exists-behavior: (オプション) デプロイプロセスの一環として、CodeDeploy エージェントは、前回のデプロイでインストールされたすべてのファイルを各インスタンスから削除します。前回のデプロイに含まれていないファイルがデプロイ先に表示された場合の処理を選択します。
- --target-instances: Blue/Green デプロイの場合、1 つ以上の Amazon EC2 Auto Scaling グループの名前、または、Amazon EC2 インスタンスを識別するのに使用するタグフィルタ キー、型、および値を含む、Blue/Green デプロイの置き換え先環境に属するインスタンスに関 する情報です。

Note

Amazon S3 内のリビジョンに関する情報をコマンドラインで直接指定するには、createdeployment 呼び出しの一部としてこの構文を使用します。(version および eTag 設定はオ プションです)。

--s3-location bucket=string,key=string,bundleType=tar|tgz|
zip,version=string,eTag=string

コマンドラインで、GitHub のリビジョンについての情報を直接呼び出す create-deployment の一部としてこの構文を使用します。

--github-location repository=*string*, commitId=*string*

EC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)

すでにプッシュされているリビジョンについての情報を入手するには、<u>list-application-</u> revisions コマンドを呼び出します。

デプロイの状態を追跡するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」を参照してくださ い。

create-deployment コマンドリファレンス

以下に、create-deployment コマンドのコマンド構造とオプションを示します。詳細について は、「AWS CLI コマンドリファレンス」の「create-deployment」を参照してください。

```
create-deployment
--application-name <value>
[--deployment-group-name <value>]
[--revision <value>]
[--deployment-config-name <value>]
[--description <value>]
[--description <value>]
[--target-instances <value>]
[--target-instances <value>]
[--auto-rollback-configuration <value>]
[--update-outdated-instances-only | --no-update-outdated-instances-only]
[--file-exists-behavior <value>]
[--s3-location <value>]
[--github-location <value>]
[--cli-input-json <value>]
[--cli-input-json <value>]
```

CodeDeploy アプリケーションを GitHub リポジトリに接続します。

を使用して GitHub リポジトリからアプリケーションを初めてデプロイする前に AWS CLI、まず GitHub アカウントに代わって GitHub を操作するための CodeDeploy アクセス許可を付与する必要 があります。CodeDeploy コンソールを使用して、このステップをアプリケーションごとに一度実行 する必要があります。

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. [Applications] (アプリケーション)を選択します。
- [アプリケーション] から、GitHub ユーザーアカウントにリンクするアプリケーションを選択し、[アプリケーションをデプロイする]を選択します。

Note

デプロイを作成していません。これは現在、GitHub ユーザーアカウントに代わって GitHub を操作するためにアクセス権限を CodeDeploy に付与する唯一の方法です。

- 4. [Repository type] の横の [My application revision is stored in GitHub] を選択します。
- 5. [Connect to GitHub] (GitHub に接続)を選択します。

Note

[Connect to a different GitHub account] リンクが表示されている場合:

すでに CodeDeploy が GitHub とやりとりすることを、アプリケーションの別の GitHub アカウントに代わって許可している場合があります。 CodeDeploy にリンクされているすべてのアプリケーションに対してサインインしてい

る GitHub アカウントに代わって、GitHub を操作する CodeDeploy の承認を取り消した 可能性があります。

詳細については、「<u>CodeDeploy のアプリケーションを使用した GitHub の認証</u>」を参照 してください。

- 6. GitHub にまだサインインしていない場合は、[Sign in] ページの手順に従います。
- 7. [Authorize application] ページで、[Authorize application] を選択します。
- 8. CodeDeploy にアクセス許可が付与されたら、[キャンセル] を選択して、「<u>EC2/ オンプレミス</u> <u>コンピューティングプラットフォームのデプロイ作成 (CLI)</u>」の手順に進んでください。

を使用して Amazon ECS ブルー/グリーンデプロイを作成する AWS CloudFormation

を使用して AWS CloudFormation、CodeDeploy を通じて Amazon ECS ブルー/グリーンデプ ロイを管理できます。デプロイを生成するには、Green と Blue のリソースを定義し、 AWS CloudFormationで使用するトラフィックルーティングと安定化の設定を指定します。このトピック では、CodeDeploy によって管理される Amazon ECS Blue/Green デプロイと AWS CloudFormation によって管理されるデプロイの違いについて説明します。

AWS CloudFormation を使用して Amazon ECS ブルー/グリーンデプロイを管理するstep-by-stepに ついては、 AWS CloudFormation ユーザーガイドの「 <u>を使用した CodeDeploy による ECS ブルー/</u> グリーンデプロイの自動化 AWS CloudFormation」を参照してください。

Note

を使用した Amazon ECS ブルー/グリーンデプロイの管理 AWS CloudFormation は、アジア パシフィック (大阪) リージョンでは利用できません。

CodeDeploy と を使用した Amazon ECS ブルー/グリーンデプロイの違い AWS CloudFormation

AWS CloudFormation スタックテンプレートは、Amazon ECS タスク関連のリソースとインフラ ストラクチャ、およびデプロイの設定オプションをモデル化します。したがって、標準の Amazon ECS ブルー/グリーンデプロイと、 を通じて作成されるブルー/グリーンデプロイには違いがありま す AWS CloudFormation。

標準の Amazon ECS Blue/Green デプロイとは異なり、以下のモデル作成や手動作成は行いません。

- デプロイする内容を一意に表す名前を指定しても、AWS CodeDeploy アプリケーションは作成されません。
- AWS CodeDeploy デプロイグループを作成しません。
- アプリケーション仕様ファイル (AppSpec ファイル)を指定しない。通常、AppSpec ファイルで管理される情報 (加重設定オプションやライフサイクルイベントなど) は、AWS::CodeDeploy::BlueGreen フックによって管理されます。

この表は、デプロイタイプ間の高レベルのワークフローの違いをまとめたものです。

関数	標準 Blue/Green デプロイ	によるブルー/グリーンデプロ イ AWS CloudFormation
Amazon ECS クラス ター、Amazon ECS サー ビス、Application Load Balancer またはNetwork Load Balancer、本稼働リスナー、 テストリスナー、および 2 つ のターゲットグループを指定 します。	これらのリソースを指定する CodeDeploy デプロイグルー プを作成します。	これらのリソースをモデル化 する AWS CloudFormation テ ンプレートを作成します。
デプロイする変更を指定しま す。	CodeDeploy でアプリケー ションを作成します。	コンテナイメージを指定する AWS CloudFormation テンプ レートを作成します。
Amazon ECS タスク定義、コ ンテナ名、コンテナポートを 指定します。	これらのリソースを指定する AppSpec ファイルを作成しま す。	これらのリソースをモデル化 する AWS CloudFormation テ ンプレートを作成します。
デプロイトラフィックシフト オプションとライフサイク ルイベントフックを指定しま す。	これらのオプションを指定す る AppSpec ファイルを作成し ます。	AWS::CodeDeploy::B lueGreen フックパラメー タを使用してこれらのオ プションを指定する AWS CloudFormation テンプレート を作成します。
CloudWatch アラーム。	ロールバックをトリガーする CloudWatch アラームを作成 します。	ロールバックをトリガーする CloudWatch アラームを AWS CloudFormation スタックレベ ルで設定します。
ロールバック/再デプロイ。	ロールバックおよび再デプロ イのオプションを指定しま す。	スタックの更新をキャンセ ルします AWS CloudForm ation。

による Amazon ECS ブルー/グリーンデプロイのモニタリング AWS CloudFormation

ブルー/グリーンデプロイは、 AWS CloudFormation と CodeDeploy を使用してモニタリングでき ます。によるモニタリングの詳細については AWS CloudFormation、「 AWS CloudFormation ユー ザーガイド」の<u>「 でのブルー/グリーンイベントのモニタリング AWS CloudFormation</u>」を参照して ください。

CodeDeploy で Blue/Green デプロイのデプロイステータスを表示するには

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u> で CodeDeploy コンソールを開きます。

Note 「CodeDeployの開始方法」で設定したのと同じユーザーでサインインします。

 デプロイでは、AWS CloudFormation スタックの更新によってトリガーされたデプロイが表示 されます。デプロイを選択して、[デプロイ履歴] を表示します。

De	Developer Tools > CodeDeploy > Deployments									
	Depl	oyment histor	у					C View	v details Ac	tions 1
		Deployment Id	Status	Deployment type	Compute platform	Application	Deployment group	Revision location	Initiating event	Start
	0	d- H8IKMZ571	Succeeded	Blue/green	Amazon ECS	-	-	stack/dkst	CloudForma tion stack update	Nov 1 3:41

 デプロイを選択して、トラフィックシフトステータスを表示します。アプリケーションおよびデ プロイグループは作成されないことに注意してください。

d-H8IKMZ571						
Deployment status				Traffic shifting	progress	
Step 1: Deploying replacement task set Step 2:		Completed	⊘ Succeeded	Original		Replacement
Rerouting production traffic to reposed of the second seco	placement task set 10	00% traffic shifted	⊘ Succeeded	Original task set not	serving traffic	Replacement task s
Terminate original task set		Completed	⊘ Succeeded			
Deployment details						
Application	Deployment ID d-H8IKMZ571			Status Succeeded		
Deployment configuration	Deployment group -			Initiated by CloudFormation st	ack update	
Deployment description This deployment is triggered by a west-	stack update for CloudFormat	tion stackId arn:aw	s:cloudformation:eu	u-		

- 4. デプロイのロールバックまたは停止には、次のことが適用されます。
 - 成功したデプロイは CodeDeploy に表示され、AWS CloudFormationによってデプロイが開始 されたことが示されます。
 - デプロイを停止してロールバックする場合は、スタックの更新をキャンセルする必要があります AWS CloudFormation。

CodeDeploy デプロイの詳細を表示する

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、 AWS アカウントに関 連付けられたデプロイの詳細を表示できます。

(i) Note

インスタンスの EC2 オンプレミスデプロイログは以下の場所で確認できます。

• Amazon Linux、RHEL、Ubuntu Server:/opt/codedeploy-agent/deploymentroot/deployment-logs/codedeploy-agent-deployments.log Windows Server の場合: C:\ProgramData\Amazon\CodeDeploy<DEPLOYMENT-GROUP-ID><DEPLOYMENT-ID>\logs\scripts.log

詳細については、「<u>ログファイルの分析によるインスタンスでのデプロイの失敗の調査</u>」を 参照してください。

トピック

- ・ デプロイの詳細の表示 (コンソール)
- <u>デプロイの詳細の表示 (CLI)</u>

デプロイの詳細の表示 (コンソール)

CodeDeploy コンソールを使用してデプロイの詳細を表示するには:

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。

1 Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しま しょう。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレン ス」の「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを 選択します。CodeDeploy は、これらのリージョンでのみサポートされています。

 1 つのデプロイの詳細を表示するには、[デプロイ履歴] でデプロイ ID を選択するか、デプロイ ID の横にあるボタンを選択して、[表示] を選択します。

デプロイの詳細の表示 (CLI)

を使用してデプロイの詳細 AWS CLI を表示するには、 get-deployment コマンドまたは batchget-deployments コマンドを呼び出します。list-deployments コマンドを呼び出して、getdeployment コマンドおよび batch-get-deployments コマンドへの入力として使用する一意の デプロイ ID のリストを取得できます。

1 つのデプロイの詳細を表示するには、[<u>デプロイの取得</u>] コマンドを呼び出し、一意のデプロイ ID を特定します。デプロイ ID を取得するには、[<u>リストデプロイグループ</u>] コマンドを呼び出します。

複数のデプロイの詳細を表示するには、[<u>デプロイの取得バッチ</u>] コマンドを呼び出し、複数の一意の デプロイ ID を特定します。デプロイ ID を取得するには、[<u>リストデプロイ</u>] コマンドを呼び出しま す。

デプロイ ID の一覧を表示するには、[リストデプロイ] コマンドを呼び出し、指定します。

- ・デプロイに関連付けられたアプリケーションの名前。アプリケーション名のリストを表示するには、[リストアプリケーション] コマンドを呼び出します。
- デプロイに関連付けられたデプロイグループの名前。デプロイグループ名のリストを表示するには、[リストデプロイグループ] コマンドを呼び出します。
- オプションで、デプロイの詳細をデプロイの状態別に含めるかどうか(指定しない場合、一致する すべてのデプロイが、デプロイの状態にかかわらず表示されます)。
- オプションで、デプロイの作成開始または終了時間別、あるいはその両方を基準にデプロイの詳細 を含めるかどうか(指定しない場合、一致するすべてのデプロイが、作成時間にかかわらず表示さ れます)。

CodeDeploy EC2/オンプレミスデプロイのログデータの表示

CodeDeploy デプロイで作成されたログデータを表示するには、CloudWatch コンソールに集約デー タを表示するように Amazon CloudWatch エージェントを設定するか、個別のインスタンスにログイ ンしてログファイルを確認します。

Note

ログは、 AWS Lambda または Amazon ECS デプロイではサポートされていません。EC2 オ ンプレミスデプロイのみに作成できます。

デプロイの詳細の表示 (CLI)

トピック

- Amazon CloudWatch コンソールでのログファイルのデータの表示
- インスタンスでのログファイルの表示

Amazon CloudWatch コンソールでのログファイルのデータの表示

インスタンスに Amazon CloudWatch エージェントがインストールされていると、そのインスタンス に対するすべてのデプロイのデータは、CloudWatch コンソールで表示できるようになります。わか りやすいように、インスタンス別に表示するのではなく、CloudWatch を使用してログファイルを集 中的にモニタリングすることをお勧めします。詳細については、「<u>CodeDeploy エージェントログを</u> CloudWatch に送信する」を参照してください。

インスタンスでのログファイルの表示

個別のインスタンスのデプロイログデータを表示するには、インスタンスにサインインして、エラー や他のデプロイイベントに関する情報を参照します。

トピック

- <u>Amazon Linux、RHEL、および Ubuntu サーバーインスタンスのデプロイログファイルを表示する</u> には
- Windows Server インスタンスのデプロイログファイルを表示するには

Amazon Linux、RHEL、および Ubuntu サーバーインスタンスのデプロイログファイ ルを表示するには

Amazon Linux、RHEL、および Ubuntu サーバーの各インスタンスのデプロイログは、次の場所に保存されています。

/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agentdeployments.log

Amazon Linux、RHEL、および Ubuntu サーバーの各インスタンスのデプロイログを表示または分析 するには、インスタンスにサインインし、次のコマンドを入力して CodeDeploy エージェントのロ グファイルを開きます。

less /var/log/aws/codedeploy-agent/codedeploy-agent.log

エラーメッセージのログファイルを参照するには、次のコマンドを入力します。

コマンド	結果
& ERROR	ログファイルでエラーメッセージのみを表示 します。ERROR という単語の前後に1つのス ペースを使用します。
/ ERROR	次のエラーメッセージを検索します。¹
? ERROR	前のエラーメッセージを検索します。² 単語の 前後に1つのスペースを入力します ERROR 。
G	ログファイルの末尾に移動します。
g	ログファイルの先頭に移動します。
q	ログファイルを終了します。
h	追加のコマンドについて参照します。

¹ / ERROR と入力してから、次のエラーメッセージを検索するには、n と入力します。前のエ ラーメッセージを検索するには、N と入力します。

²? ERROR と入力してから、次のエラーメッセージを検索するには n か前のエラーメッセージ を検索するには N と入力します。

また、次のコマンドを入力して CodeDeploy スクリプトのログファイルを開くこともできます。

less /opt/codedeploy-agent/deployment-root/deployment-group-ID/deployment-ID/logs/
scripts.log

エラーメッセージのログファイルを参照するには、次のコマンドを入力します。

コマンド	結果
&stderr	ログファイルでエラーメッセージのみを表示し ます。

コマンド	結果
/stderr	次のエラーメッセージを検索します。¹
?stderr	前のエラーメッセージを検索します。²
G	ログファイルの末尾に移動します。
g	ログファイルの先頭に移動します。
q	ログファイルを終了します。
h	追加のコマンドについて参照します。

¹/stderrと入力してから、次のエラーメッセージを前方に検索するにはnと入力します。前の エラーメッセージを後方に検索するには、Nと入力します。

²?stderrと入力してから、次のエラーメッセージを後方に検索するには、nと入力します。前の エラーメッセージを前方に検索するには、Nと入力します。

Windows Server インスタンスのデプロイログファイルを表示するには

CodeDeploy エージェントのログファイル: Windows サーバーインスタンスで CodeDeploy エージェ ントのログファイルは次の場所に保存されています。

C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt

Windows Serve インスタンスで CodeDeploy エージェントのログファイルを表示または分析するに は、インスタンスにサインインし、次のコマンドを入力してファイルを開きます。

notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt

ログファイルでエラーメッセージを参照するには、Ctrl+F キーを押し、ERROR [と入力してから、Enter キーを押して最初のエラーを見つけます。

CodeDeploy スクリプトログファイル: Windows サーバーインスタンスのデプロイログは次の場所に 保存されています。

C:\ProgramData\Amazon\CodeDeploy\deployment-group-id\deployment-id\logs
\scripts.log

コードの説明は以下のとおりです。

- *deployment-group-id* は examplebf3a9c7a-7c19-4657-8684-b0c68d0cd3c4 などの文 字列
- *deployment-id* は、d-12EXAMPLE などの識別子

また、次のコマンドを入力して CodeDeploy スクリプトのログファイルを開くこともできます。

notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\deployment-ID\logs
\scripts.log

ログファイルでエラーメッセージを参照するには、Ctrl+F キーを押し、**stderr** と入力してか ら、Enter キーを押して最初のエラーを見つけます。

CodeDeploy でデプロイの停止

CodeDeploy コンソール、 AWS CLI、または CodeDeploy APIs を使用して、 AWS アカウントに関 連付けられたデプロイを停止できます。

Marning

EC2 オンプレミス のデプロイを停止すると、デプロイグループのインスタンスの一部または すべてを未確定のデプロイメントの状態のまま残すことができます。詳細については、「<u>停</u> 止、失敗したデプロイ」を参照してください。

デプロイを停止することも、デプロイを停止してロールバックすることもできます。

- デプロイ (コンソール) の停止
- <u>デプロイ (CLI) の停止</u>

Note

デプロイが によるブルー/グリーンデプロイの場合 AWS CloudFormation、CodeDeploy コン ソールでこのタスクを実行することはできません。 AWS CloudFormation コンソールに移動 して、このタスクを実行します。

デプロイ (コンソール) の停止

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeployの開始方法」で設定したのと同じユーザーでサインインします。

2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。

Note

エントリが表示されない場合は、正しいリージョンが選択されていることを確認しま しょう。ナビゲーションバーのリージョンセレクターで、「AWS 全般のリファレン ス」の「<u>リージョンとエンドポイント</u>」に一覧表示されているいずれかのリージョンを 選択します。CodeDeploy は、これらのリージョンでのみサポートされています。

- 3. 停止するデプロイを選択し、以下のいずれかの操作を行います。
 - 1. [デプロイの停止] を選択し、ロールバックなしでデプロイを停止します。
 - 2. デプロイを停止してロールバックするには、[Stop and roll back deployment (デプロイを停止 してロールバック)] を選択します。

詳細については、「<u>CodeDeploy を使用した再デプロイおよびデプロイのロールバック</u>」を参照 してください。

Note

[デプロイの停止] および [Stop and roll back deployment (デプロイを停止してロールバック)] が使用不可の場合、そのデプロイは停止できないところまで進行しています。

デプロイ (CLI) の停止

デプロイ ID を指定して、[<u>停止デプロイ</u>] コマンドを呼び出します。デプロイ ID の一覧を表示するに は、[<u>リストデプロイ]</u> コマンドを呼び出します。

CodeDeploy を使用した再デプロイおよびデプロイのロールバック

CodeDeploy は、以前にデプロイされたアプリケーションのリビジョンを新しいデプロイとして再デ プロイすることによって、デプロイをロールバックします。これらのロールバックされたデプロイ は、前のデプロイのバージョンを復元するのではなく、新しいデプロイ ID を使用する技術的に新し いデプロイです。

デプロイは、自動または手動でロールバックできます。

トピック

- 自動ロールバック
- 手動ロールバック
- ロールバックおよび再デプロイのワークフロー
- 既存のコンテンツでのロールバック動作

自動ロールバック

デプロイが失敗した場合、または指定した監視しきい値に達した場合、自動的にロールバックするよ うに、デプロイグループまたはデプロイを設定できます。この場合、アプリケーションリビジョンの 最後の既知の正常なバージョンがデプロイされます。自動ロールバックは、アプリケーションを作成 するとき、またはデプロイグループを作成または更新するときに設定します。

新しいデプロイを作成するとき、デプロイグループに指定された自動ロールバック設定をオーバーラ イドすることもできます。

Note

デプロイが自動的にロールバックされるときには、Amazon Simple Notification Service を 使用して通知を受け取ることができます。詳細については、<u>Monitoring Deployments with</u> Amazon SNS Event Notifications を参照してください。

自動ロールバックの設定の詳細については、「<u>デプロイグループの詳細オプションの設定</u>」を参照し てください。

手動ロールバック

自動ロールバックをセットアップしていない場合は、以前にデプロイされたアプリケーションリビ ジョンを使用する新しいデプロイを作成し、リビジョンを再デプロイする手順に従うことによって、 デプロイを手動でロールバックすることができます。アプリケーションが不明な状態になった場合、 これを行う場合があります。トラブルシューティングに多くの時間を費やすのではなく、アプリケー ションを既知の動作状態に再デプロイすることができます。詳細については、「<u>CodeDeployでデプ</u> ロイを作成する」を参照してください。

Note

デプロイグループからインスタンスを削除する場合、CodeDeploy はそのインスタンスにす でにインストールされているものはアンインストールしません。

ロールバックおよび再デプロイのワークフロー

自動ロールバックが開始された場合、または再デプロイまたは手動ロールバックを手動で開始した場合、CodeDeploy は、まず、最後に正常にインストールされたすべてのファイルを各参加インスタン スから削除しようとします。CodeDeploy は、クリーンアップファイルをチェックすることでこれを 行います:

/opt/codedeploy-agent/deployment-root/deployment-instructions/*deployment-group-ID*-cleanup ファイル (Amazon Linux、Ubuntu Server、および RHEL インスタンス用)

C:\ProgramData\Amazon\CodeDeploy\deployment-instructions*deployment-group-ID*-cleanup ファイル (Windows Server インスタンス用)

存在する場合、CodeDeploy は、クリーンアップファイルを使用して、新しいデプロイを開始する前 にリストされたすべてのファイルをインスタンスから削除します。

例えば、最初の 2 つのテキストファイルおよび 2 つのスクリプトファイルは、Windows Server を実 行している Amazon EC2 インスタンスにデプロイ済みであり、スクリプトによってデプロイライフ サイクルイベント中にさらに 2 つのテキストファイルが作成されました :

c:\temp\a.txt (previously deployed by CodeDeploy)
c:\temp\b.txt (previously deployed by CodeDeploy)
c:\temp\c.bat (previously deployed by CodeDeploy)
c:\temp\d.bat (previously deployed by CodeDeploy)

クリーンアップファイルは、最初の2つのテキストファイルおよび2つのスクリプトファイルのみ が表示されます。

c:\temp\a.txt
c:\temp\b.txt
c:\temp\c.bat
c:\temp\d.bat

新しいデプロイの前に、CodeDeploy は最初の 2 つのテキストファイルおよび 2 つのスクリプトファ イルのみを削除し、最後の 2 つのテキストファイルはそのまま残します :

c:\temp\a.txt will be removed c:\temp\b.txt will be removed c:\temp\c.bat will be removed c:\temp\d.bat will be removed c:\temp\e.txt will remain c:\temp\f.txt will remain

このプロセスの一部として、CodeDeploy は、手動または自動ロールバックであっても、その後の 再デプロイの間に以前のデプロイのスクリプトによって実行されるアクションを元に戻したり、一 致させたりしません。例えば、c.bat および d.bat のファイルに、e.txt および f.txt のファイ ルが既に存在している場合、再作成しないロジックを含んでいる場合は、e.txt および f.txt の 古いバージョンはそのまま残り、それは CodeDeploy デプロイがその後のデプロイで c.bat およ び d.bat を実行しても残ったままです。c.bat および d.bat にロジックを追加して、新しいバー ジョンを作成する前に e.txt および f.txt の古いバージョンを常にチェックして削除することが できます。

既存のコンテンツでのロールバック動作

デプロイプロセスの一環として、CodeDeploy エージェントは、前回のデプロイでインストールされ たすべてのファイルを各インスタンスから削除します。前回のデプロイに含まれていないファイル がデプロイが、ターゲットデプロイに表示された場合は、次回のデプロイ時にこれらのファイルを CodeDeploy で処理する方法を選択できます:

• [Fail the deployment] — エラーが報告され、デプロイのステータスが「Failed (失敗)」に変更され ます。
- [コンテンツの上書き] アプリケーションリビジョンのファイルのバージョンにより、インスタンスの既存のファイルのバージョンが置き換えられます。
- [コンテンツの保持] デプロイ先のファイルは保持され、アプリケーションリビジョンのバー ジョンはインスタンスにコピーされません。

この動作は、デプロイの作成時に選択できます。コンソールでデプロイを作成する場合は、「<u>EC2/</u> <u>オンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)</u>」を参照してくださ い。を使用してデプロイを作成する場合は AWS CLI、「」を参照してください<u>EC2/ オンプレミスコ</u> ンピューティングプラットフォームのデプロイ作成 (CLI)。

ファイルを保持して次回のデプロイの一部とすることを選択すると、そのファイルはアプリケーショ ンリビジョンパッケージに追加する必要がなくなります。例えば、デプロイに必要なファイルでもア プリケーションリビジョンバンドルには追加しないで直接インスタンスにアップロードできます。ま たは、アプリケーションが既に本番稼働環境にあっても CodeDeploy を初めて使用してデプロイす る場合は、ファイルをインスタンスにアップロードできます。

ロールバックでは、デプロイの失敗が原因で前回の成功したデプロイのアプリケーションリビジョン が再デプロイされますが、その前回の成功したデプロイのコンテンツ処理オプションがロールバック デプロイに適用されます。

ただし、失敗したデプロイの設定がファイルを保持せずに上書きするようになっていた場合、ロール バックは予期しない結果になる可能性があります。特に、保持しようとしていたファイルが、失敗し たデプロイによって削除される可能性があります。ロールバックデプロイを実行したときに、ファイ ルは、ロールバック時にインスタンス上ではなくデプロイが実行されます。

次の例では 3 つのデプロイがあります。失敗したデプロイ 2 で上書き (削除) されたファイルは、デ プロイ 3 でアプリケーションリビジョン 1 が再度デプロイされたときには使用不能 (保持不能) で す。

デプロイ	アプリケーショ ンリビジョン	コンテンツ上書 きオプション	デプロイのス テータス	動作と結果
デプロイ 1	アプリケーショ ンリビジョン 1	保持	成功	CodeDeploy は、前回のデプ ロイでデプロ イされなかった ファイルをデプ ロイ先で検出し

AWS	Code	Deploy
-----	------	--------

デプロイ	アプリケーショ ンリビジョン	コンテンツ上書 きオプション	デプロイのス テータス	動作と結果
				ます。これらのファイルらのファイルロイのでするためーごうないまでしためにていまつののたいまファイルはのファイルはクロッケーで保持およす。
デプロイ 2	アプリケーショ ンリビジョン 2	上書き	失敗	デプロイプ ロセス中 に、CodeDeploy は前プロの成し たまているす ているすべて アイルを 削にてアイルを れてファイルを れてファイルを たび たいで イルも含 まただ ちたい で り し、 で り し、 で の し し ているす で つ し たま の し ているす で つ し て いるす で て いるす で で の た で で の の し て い る す で の の し て い る す で の の し て い る す で の の し て の の し て の の し て い る す で の の し て の の し て い る す で の の し て の の し て の の し て の の し て の の し て の の し て の の し て の の し て の の し て の の し こ ろ て い る す の し た の ろ し た の ろ し の た の つ つ し て の の し し た の ろ し し た の う の し し た の つ し し た の ろ し し の た つ て の て の て の て の て の ろ つ ち つ て の ろ て の ろ つ て の て つ て の ち つ こ の ち の つ つ し ち つ つ て つ つ て の つ ろ つ て つ ろ つ て つ し つ て つ つ し ち の つ つ ろ つ て つ つ つ つ し つ つ つ つ つ つ し つ つ つ つ つ つ

デプロイ	アプリケーショ ンリビジョン	コンテンツ上書 きオプション	デプロイのス テータス	動作と結果
デプロイ 3	アプリケーショ ンリビジョン 1	保持		デデプロ有る plるてプリリビプ たイるデ失除得き C れはシン加た ププに一効た oy こ いリビケジロ だ 1 フプ敗 さすま d しに アロすれる t ひで b し に アロすれる t ひで b し に アロ すれる t D o フプリ 必自 ン z コ 1 ば ス で t いりごう アンに、タき、 プサイイ いり いん いっかい b c c c c c c c c c c c c c c c c c c
				ノリケーション

デプロイ	アプリケーショ ンリビジョン	コンテンツ上書 きオプション	デプロイのス テータス	動作と結果
				リビジョンを作 成できます。

異なる AWS アカウントでアプリケーションをデプロイする

通常、組織には、さまざまな目的で使用する複数の AWS アカウントがあります (たとえば、1 つは システム管理タスク用、もう 1 つは開発、テスト、本番稼働用タスク用、または 1 つは開発および テスト環境に関連付けられ、もう 1 つは本番稼働用環境に関連付けられます)。

異なるアカウントで関連作業を実行できる場合でも、CodeDeploy デプロイグループとデプロイ先の Amazon EC2 インスタンスは、これらを作成したアカウントに厳密に関連付けられています。たと えば、1 つのアカウントで起動したインスタンスを別のデプロイグループに追加することはできませ ん。

開発 AWS アカウントと本番稼働用アカウントの 2 つのアカウントがあるとします。主に開発用アカ ウントで作業しますが、認証情報のフルセットまたは開発用アカウントからのサインアウトや本番稼 働用アカウントへのサインインなしで、本番稼働用アカウントでデプロイの開始を可能にします。

クロスアカウント設定手順に従うことで、別のアカウントの認証情報フルセットの必要なしで、組織の別のアカウントに属するデプロイを開始できます。これは、そのアカウントへの一時的アクセスを 許可する AWS Security Token Service (AWS STS) が提供する機能を利用して一部行われます。

ステップ 1: いずれかのアカウントで S3 バケットを作成する

開発用アカウントまたは本番稼働用アカウントで以下を行います。

 まだそうしていない場合は、本稼働用アカウントのアプリケーションリビジョンが保存される Amazon S3 バケットを作成します。詳細については、「Amazon S3 のバケットの作成」を参照し てください。同じファイルを開発用アカウントでテスト、確認した本稼働環境にデプロイして、同 じバケットとアプリケーションリビジョンを両方のアカウントに使用することもできます。

ステップ 2: Amazon S3 バケットへのアクセス許可を本番稼働用アカウン トの インスタンスプロファイルに付与する

ステップ 1 で作成した Amazon S3 バケットが本番稼働用アカウントにある場合、このステップは必 要ありません。後で引き受けるロールは、本番稼働用アカウントにもあるため、このバケットへのア クセス権限をすでに持っています。

開発用アカウントで Amazon S3 バケットを作成する場合は、以下を実行します。

 本番稼働用アカウントで、IAM インスタンスプロファイルを作成します。詳細については、「<u>ス</u> <u>テップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成する</u>」を参照して ください。

Note

この IAM インスタンスプロファイルの ARN を書き留めます。次に作成するクロスバケットポリシーにそれを追加する必要があります。

 開発用アカウントで作成した Amazon S3 バケットへのアクセス権限を、本番稼働用アカウントで 先ほど作成した IAM インスタンスプロファイルに付与します。詳細については、「例 2: バケット 所有者がクロスアカウントのバケットのアクセス許可を付与する」を参照してください。

クロスアカウントのバケットのアクセス許可を付与するプロセスを完了するにあたり、次の点に注 意してください。

- サンプルチュートリアルでは、アカウントAは開発用アカウントを表し、アカウントBは本番 稼働用アカウントを表します。
- アカウントA(開発用アカウント)タスクを実行する際、チュートリアルで提供されているサン プルポリシーを使用する代わりに、次のバケットポリシーを変更してクロスアカウントアクセス 許可を付与します。

```
"Action": [
    "s3:Get*",
    "s3:List*"
],
    "Resource": [
    "arn:aws:s3:::bucket-name/*"
]
}
```

account-idは、IAM インスタンスプロファイルを作成した本番稼働用アカウントのアカウン ト番号を表します。

role-nameは、作成した IAM インスタンスプロファイルの名前を表します。

bucket-name は、ステップ 1 で作成したバケットの名前を表します。バケット名の後に必ず / * が含まれるようにして、バケット内の各ファイルへのアクセスを提供します。

ステップ 3: 本番稼働用アカウントでリソースとクロスアカウントロールを 作成する

本番稼働用アカウントで以下を行います。

- このガイドの手順を使用して、CodeDeploy リソース(アプリケーション、デプロイグループ、 デプロイ設定、Amazon EC2 インスタンス、Amazon EC2 インスタンスプロファイル、サービス ロールなど)を作成します。
- 開発用アカウントのユーザーが、この本番稼働用アカウントで CodeDeploy オペレーションを実行するために追加で引き受けるクロスアカウント IAM ロールを作成します。

クロスAWS アカウントロールの作成に役立つガイドとして、チュートリアル: IAM ロールを使用 してアカウント間のアクセスを委任します。チュートリアルのサンプルアクセス許可をポリシー ドキュメントに追加する代わりに、少なくとも次の 2 つの AWS 指定されたポリシーをロールにア タッチする必要があります。

AmazonS3FullAccess: S3 バケットが開発用アカウントにある場合にのみ必要です。引き受けた本番稼働用アカウントのロールに対して、リビジョンが保存されている、開発用アカウントのAmazon S3 サービスとリソースへのフルアクセスを提供します。

AWSCodeDeployDeployerAccess: リビジョンを登録してデプロイすることをユーザーに許可します。

デプロイを開始するだけでなく、デプロイグループを作成および管理する場

合、AWSCodeDeployFullAccess ポリシーの代わりに、AWSCodeDeployDeployerAccess ポ リシーを追加します。IAM マネージドポリシーを使用して CodeDeploy タスクにアクセス許可を 付与する方法の詳細については、「<u>AWS CodeDeploy の マネージド (事前定義) ポリシー</u>」を参照 してください。

このクロスアカウントロールの使用時に、ほかの AWS サービスでタスクを実行する場合、追加の ポリシーをアタッチできます。

A Important

クロスアカウントの IAM ロールを作成する際に、本番稼働用アカウントへのアクセスを得る ために必要な詳細を書き留めておきます。 を使用してロール AWS Management Console を切り替えるには、次のいずれかを指定する

を使用してロール AWS Management Console を切り替えるには、次のいすれかを指定する 必要があります。

- 引き受けたロールの認証情報を使用して本番稼働用アカウントにアクセスするための URL。URL は [確認] ページのクロスアカウント作成プロセスの最後に表示されます。
- クロスアカウントロール名およびアカウント ID 番号またはエイリアス。

を使用してロール AWS CLI を切り替えるには、以下を指定する必要があります。

・引き受けるクロスアカウントロールの ARN。

ステップ 4: Amazon S3 バケットにアプリケーションリビジョンをアップ ロードする

Amazon S3 バケットを作成したアカウントで

 Amazon S3 バケットにアプリケーションリビジョンをアップロードします。詳細については、 「<u>Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オンプレミスのデプロイのみ)</u>」 を参照してください。

ステップ 5: クロスアカウントロールを引き受け、アプリケーションをデプ ロイする

開発アカウントでは、 AWS CLI または AWS Management Console を使用してクロスアカウント ロールを引き受け、本番稼働用アカウントでデプロイを開始できます。

を使用してロール AWS Management Console を切り替え、デプロイを開始する方法について は、<u>「ロールへの切り替え (AWS Management Console)</u>」および「」を参照してください<u>EC2/オ</u> ンプレミスコンピューティングプラットフォームのデプロイ作成 (コンソール)。

を使用してクロスアカウントロールを引き受け AWS CLI 、デプロイを開始する方法について は、<u>「IAM ロールへの切り替え (AWS Command Line Interface)</u>」および「」を参照してくださ いEC2/ オンプレミスコンピューティングプラットフォームのデプロイ作成 (CLI)。

を通じてロールを引き受ける方法の詳細については AWS STS、<u>AWS Security Token Service</u> <u>「 ユーザーガイド</u>」の<u>AssumeRole</u>」および<u>AWS CLI 「 コマンドリファレンス</u>」の<u>「 assume-</u> role」を参照してください。

関連トピック:

• CodeDeploy: 開発用アカウントから本番稼働用アカウントへのデプロイ

CodeDeploy エージェントを使用してローカルマシン上のデプロイ パッケージを検証する

CodeDeploy エージェントを使用して、ログインしているインスタンスのコンテンツをデプロイでき ます。これにより、デプロイ時に使用予定のアプリケーション仕様ファイル (AppSpec ファイル) と デプロイ予定のコンテンツの整合性を検証することができます。

アプリケーションおよびデプロイグループを作成する必要はありません。ローカルインスタンスに 保存されているコンテンツをデプロイする場合は、 AWS アカウントも必要ありません。最も簡単 な検証方法としては、AppSpec ファイル やデプロイ予定のコンテンツが含まれているディレクトリ で、オプションを指定せずに codedeploy-local コマンドを実行します。このツールには、他のテス トケースのオプションが含まれています。

ローカルマシン上のデプロイパッケージを検証することで、以下を行うことができます。

- アプリケーションリビジョンの整合性の検証。
- AppSpec ファイルのコンテンツの検証。

- ・既存のアプリケーションコードを使用した CodeDeploy の初回実行。
- コンテンツの迅速なデプロイ (インスタンスにすでにログインしている場合)。

ローカルインスタンス、またはサポートされているリモートリポジトリタイプ (Amazon S3 バケッ トまたはパブリックの GitHub リポジトリ) 上に保存されているコンテンツをデプロイできます。

前提条件

ローカルのデプロイを開始する前に、以下の手順を行います。

- CodeDeploy エージェントでサポートされているインスタンスタイプを作成または使用します。詳細については、「CodeDeploy エージェントで対応するオペレーティングシステム」を参照してください。
- バージョン 1.0.1.1352 以降の CodeDeploy エージェントをインストールします。詳細については、「CodeDeploy エージェントをインストールする」を参照してください。
- Amazon S3 バケットや GitHub リポジトリからコンテンツをデプロイする場合は、CodeDeploy で 使用するユーザーをプロビジョニングします。詳細については、「ステップ 1: セットアップ」を 参照してください。
- Amazon S3 バケットからアプリケーションリビジョンをデプロイする場合は、作業をしている リージョンで Amazon S3 バケットを作成し、このバケットに Amazon S3 バケットポリシーを適 用します。このポリシーでは、アプリケーションリビジョンをダウンロードするために必要なアク セス許可をインスタンスに付与します。

例えば、次の Amazon S3 バケットポリシーは、ARN arn:aws:iam::444455556666:role/ CodeDeployDemo を含む IAM インスタンスプロファイルがアタッチされた Amazon EC2 インス タンスが、amzn-s3-demo-bucket という名前の Amazon S3 バケットの任意の場所からダウン ロードすることを許可します。

```
{
    "Statement": [
        {
            "Action": [
               "s3:Get*",
               "s3:List*"
        ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
        }
    }
}
```

```
"AWS": [
"arn:aws:iam::444455556666:role/CodeDeployDemo"
]
}
]
]
```

次の Amazon S3 バケットポリシーは、ARN arn:aws:iam::444455556666:user/ CodeDeployUser を含む IAM ユーザーに関連付けられたオンプレミスインスタンスが、amzns3-demo-bucket という名前の Amazon S3 バケット内の任意の場所からダウンロードすること を許可します。

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                 "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                 "AWS": [
                     "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

Amazon S3 バケットポリシーを生成しアタッチする方法の詳細については、「<u>バケットポリシー</u> <u>の例</u>」を参照してください。

 Amazon S3 バケットまたは GitHub リポジトリからアプリケーションリビジョンをデプロイする 場合は、IAM インスタンスプロファイルをセットアップして、そのプロファイルをインスタンス にアタッチします。詳細については、「ステップ 4: Amazon EC2 インスタンス用の IAM インスタ ンスプロファイルを作成する」、「CodeDeploy (AWS CLI または Amazon EC2 コンソール) 用の Amazon EC2 インスタンスを作成する」、および「CodeDeploy 用の Amazon EC2 インスタンス を作成する (AWS CloudFormation テンプレート)」を参照してください。 GitHub からコンテンツをデプロイする場合は、GitHub アカウントおよびパブリックリポジト リを作成します。GitHub のアカウントを作成するには、「GitHub への参加」を参照してください。

Note

プライベートリポジトリは、現在サポートされていません。プライベートの GitHub リ ポジトリにコンテンツが保存されている場合は、インスタンスにダウンロードし、-bundle-location オプションでローカルパスを指定します。

- インスタンスにデプロイするコンテンツ (AppSpec ファイルなど)を準備し、Amazon S3 バ ケットまたは GitHub リポジトリのローカルインスタンスに配置します。詳細については、 「CodeDeploy のアプリケーションリビジョンの操作」を参照してください。
- 他の設定オプションでデフォルト以外の値を使用する場合は、設定ファイルを作成し、その設定 ファイルをインスタンスに配置します (Amazon Linux、RHEL、または Ubuntu Server インスタ ンスの場合は /etc/codedeploy-agent/conf/codedeployagent.yml、Windows Server イ ンスタンスの場合は C:\ProgramData\Amazon\CodeDeploy\conf.yml)。詳細については、 「CodeDeploy エージェント設定リファレンス」を参照してください。

Note

Amazon Linux、RHEL、または Ubuntu Server インスタンスで設定ファイルを使用する場合は、以下のいずれかを行う必要があります。

- :root_dir:変数および:log_dir:変数を使用して、デプロイルートおよびログディレクトリフォルダでデフォルト以外の場所を指定する。
- sudo を使用して、CodeDeploy エージェントのコマンドを実行する。

ローカルのデプロイを作成する。

ローカルデプロイを作成するインスタンスで、ターミナルセッション (Amazon Linux、RHEL、また は Ubuntu Server インスタンスの場合)、またはコマンドプロンプト (Windows Server の場合) を開 き、ツールコマンドを実行します。

Note

codedeploy-local コマンドは、以下の場所にインストールされます。

- ・ Amazon Linux、RHEL、または Ubuntu Server: /opt/codedeploy-agent/bin
- Windows Server: C:\ProgramData\Amazon\CodeDeploy\bin

基本的なコマンド構文

codedeploy-local [options]

概要

```
codedeploy-local
[--bundle-location <value>]
[--type <value>]
[--file-exists-behavior <value>]
[--deployment-group <value>]
[--events <comma-separated values>]
[--agent-configuration-file <value>]
[--appspec-filename <value>]
```

オプション

-I、--bundle-location

アプリケーションリビジョンバンドルの場所。場所を指定しない場合は、現在作業中のディレクトリ がデフォルトで使用されます。--bundle-location に値を指定する場合、--type の値も指定す る必要があります。

バンドルの場所を表す形式の例を以下に示します。

- ローカルのAmazon Linux、RHEL、または Ubuntu Server インスタンス: /path/to/local/ bundle.tgz
- ローカルの Windows Server インスタンス: C:/path/to/local/bundle
- Amazon S3 バケット: s3://amzn-s3-demo-bucket/bundle.tar
- GitHub リポジトリ: https://github.com/account-name/repository-name/

-t、--type

アプリケーションリビジョンバンドルの形式。サポートされるタイプには tgz、tar、zip、directory などがあります。タイプを指定しない場合は、デフォルトで directory が使用されます。--type に値を指定する場合、--bundle-location の値も指定す る必要があります。

-b、--file-exists-behavior

デプロイのターゲット場所に存在しているが、デプロイが正常に完了していないファイルを処理する 方法について説明します。オプションには DISALLOW、OVERWRITE、RETAIN などがあります。 詳細については、「<u>AWS CodeDeploy API リファレンス</u>」の「fileExistsBehavior」を参照してくだ さい。

-g、--deployment-group

デプロイされるコンテンツのターゲット場所を示すフォルダへのパス。フォルダを指定しない場合 は、default-local-deployment-group という名前のフォルダがデプロイルートディレクトリ内に作成 されます。ローカルデプロイを作成する度に、d-98761234-local のような名前のサブディレクトリ がこのフォルダ内に作成されます。

-e、--events

上書きライフサイクルのイベントフックセット。AppSpec ファイルに表示されているイベントの代わりに、順番に実行されます。フックが複数ある場合は、カンマ区切りで指定できます。このオプションは以下の場合に使用できます。

- AppSpec ファイルをアップデートせずに、異なるイベントのセットを実行する場合。
- AppSpec ファイルの例外として、単一のイベントフックを実行する場合 (例: ApplicationStop)。

上書きリストで DownloadBundle イベントや Install イベントを指定しない場合は、指定したすべて のイベントフックより前に実行されます。DownloadBundle や Install を --events オプションのリ ストに含めた場合、これらのイベントに先行するイベントは、CodeDeploy デプロイでこれらのイ ベントの前に通常実行されるイベントに限ります。詳細については、「<u>AppSpec の「hooks」セク</u> ション」を参照してください。

-c、--agent-configuration-file

デプロイに使用する設定ファイルの場所 (デフォルト以外の場所に設定ファイルを保存している場合)。設定ファイルは、デプロイ向けに別のデフォルト値および動作を指定します。

デフォルトでは、設定ファイルは /etc/codedeploy-agent/conf/codedeployagent.yml (Amazon Linux、RHEL、または Ubuntu Server インスタンス)、または C:/ProgramData/ Amazon/CodeDeploy/conf.yml (Windows Server)。詳細については、「<u>CodeDeploy エージェン</u> ト設定リファレンス」を参照してください。

-A, --appspec-filename

AppSpec ファイルの名前。ローカルデプロイの場合、許容される値は appspec.yml と appspec.yaml です。デフォルトでは、AppSpec ファイルは appspec.yml と呼ばれます。

-h、--help

ヘルプコンテンツの概要を表示します。

-v、--version

ツールのバージョン番号を表示します。

```
例
```

有効なコマンド形式の例を以下に示します。

codedeploy-local

codedeploy-local --bundle-location /path/to/local/bundle/directory

codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deploymentgroup my-deployment-group

codedeploy-local --bundle-location /path/to/local/directory --type directory -deployment-group my-deployment-group

Amazon S3 からバンドルをデプロイする。

codedeploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.tgz --type tgz

codedeploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.zip? versionId=1234&etag=47e8 --type zip --deployment-group my-deployment-group

パブリック GitHub リポジトリからバンドルをデプロイする:

codedeploy-local --bundle-location https://github.com/awslabs/aws-codedeploy-sampletomcat --type zip

codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploysample-tomcat/zipball/master --type zip

codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploysample-tomcat/zipball/HEAD --type zip

codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploysample-tomcat/zipball/1a2b3c4d --type zip

複数のライフサイクルイベントを指定するバンドルをデプロイする:

codedeploy-local --bundle-location /path/to/local/bundle.tar --type tar --applicationfolder my-deployment --events DownloadBundle,Install,ApplicationStart,HealthCheck

ApplicationStop ライフサイクルイベントを使用して、以前にデプロイされたアプリケーションを停止する:

codedeploy-local --bundle-location /path/to/local/bundle.tgz --type tgz --deploymentgroup --events ApplicationStop

特定のデプロイグループ ID を使用してデプロイする:

codedeploy-local --bundle-location C:/path/to/local/bundle/directory --deployment-group
1234abcd-5dd1-4774-89c6-30b107ac5dca

codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deploymentgroup 1234abcd-5dd1-4774-89c6-30b107ac5dca

CodeDeploy でのデプロイモニタリング

モニタリングは、CodeDeploy と 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。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグで きるように、 AWS ソリューションのすべての部分からモニタリングデータを収集する必要がありま す。ただし、CodeDeploy のモニタリングを開始する前に、以下の質問に対する回答を反映したモニ タリング計画を作成する必要があります。

- モニタリングの目的は何ですか?
- どのリソースをモニタリングしますか?
- どのくらいの頻度でこれらのリソースをモニタリングしますか?
- ・ どのモニタリングツールを利用しますか?
- 誰がモニタリングタスクを実行しますか?
- 問題が発生したときに誰が通知を受け取りますか?

次のステップでは、さまざまなタイミングと負荷条件で CodeDeploy パフォーマンスを測定するこ とにより、お客様の環境で通常の のパフォーマンスのベースラインを確定します。CodeDeploy の モニタリングでは、過去のモニタリングデータを保存し、現在のパフォーマンスデータと比較する ことで、パフォーマンスの通常パターンと異常パターンを特定し、問題に対処する方法を考案できま す。

例えば、CodeDeploy を使用すると、デプロイおよびターゲットインスタンスのステータスをモニタ リングできます。デプロイまたはインスタンスが失敗すると、アプリケーション仕様ファイルの再 設定、CodeDeploy エージェントの再インストールまたは更新、アプリケーションまたはデプロイグ ループの設定の更新、インスタンスの設定または AppSpec ファイルの変更が必要が生じることがあ ります。

ベースラインを確立するには、少なくとも次の項目をモニタリングする必要があります。

- デプロイイベントとステータス
- インスタンスイベントとステータス

自動モニタリングツール

AWS には、CodeDeploy のモニタリングに使用できるさまざまなツールが用意されています。これ らのツールの一部はモニタリングを行うように設定できますが、一部のツールは手動による介入が必 要です。モニタリングタスクをできるだけ自動化することをお勧めします。

以下の自動化されたモニタリングツールを使用して、CodeDeploy を監視し、問題が発生したときに レポートできます。

- Amazon CloudWatch アラーム 指定した期間にわたって単一のメトリクスをモニタリングし、複数の期間にわたる特定のしきい値に対するメトリクスの値に基づいて1つ以上のアクションを実行します。アクションは、Amazon Simple Notification Service (Amazon SNS)のトピックまたはAmazon EC2 Auto Scalingのポリシーに送信される通知です。CloudWatch アラームは、特定の状態にあるという理由だけでアクションを呼び出すことはありません。状態が変更され、指定された期間維持されている必要があります。詳細については、「Monitoring Deployments with Amazon CloudWatch Tools」を参照してください。
 - CloudWatch アラームモニタリングを使用するサービスロールの更新に関する詳細については、 「<u>CloudWatch アクセス権限を CodeDeploy サービスロールに付与する</u>」を参照してくださ い。CloudWatch アラームモニタリングの CodeDeploy オペレーションへの追加の詳細について は、「<u>CodeDeploy でアプリケーションを作成する</u>」、「<u>CodeDeploy でデプロイグループを作成</u> <u>する</u>」または「<u>CodeDeploy を使用して、デプロイグループの設定を変更します。</u>」を参照してく ださい。
- Amazon CloudWatch Logs AWS CloudTrail またはその他の出典のログファイルのモニタリン グ、保存、アクセスを行います。詳細については、「Amazon CloudWatch ユーザーガイド」の 「ログファイルのモニタリング」を参照してください。

CloudWatch コンソールを使用して CodeDeploy ログを表示する方法については、「<u>CloudWatch</u> Logs コンソールで CodeDeploy ログを表示する」を参照してください。。

Amazon CloudWatch Events - イベントに一致したものを1つ以上のターゲットの関数またはストリームに渡して、変更、状態の情報の収集、是正措置を行います。詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch Events とは」を参照してください。

CodeDeploy オペレーションで CloudWatch Events の使用の詳細については、「<u>Amazon</u> CloudWatch Events を使用したデプロイのモニタリング」を参照してください。

 AWS CloudTrail ログモニタリング – アカウント間でログファイルを共有し、CloudWatch Logs に 送信CloudWatch CloudTrail ログファイルをリアルタイムでモニタリングし、Java でログ処理アプ リケーションを書き込み、CloudTrail による配信後にログファイルが変更されていないことを確認 します。詳細については、「AWS CloudTrail ユーザーガイド」の「<u>CloudTrail ログファイルの使</u> 用」を参照してください。

CodeDeploy での CloudTrail の使用についての詳細は、「<u>Monitoring Deployments</u>」を参照してく ださい。

 Amazon Simple 通知サービス - イベント駆動型のトリガーを設定して、成功または失敗など、デ プロイおよびインスタンスイベントについての SMS や電子メール通知を受信します。詳細につい ては、「<u>トピックの作成</u> と <u>Amazon Simple Notification Service とは</u>」を参照してください。

アカウントレベルの Amazon SNS の CodeDeploy 通知をセットアップする詳しい方法について は、「Monitoring Deployments with Amazon SNS Event Notifications」を参照してください。

手動モニタリングツール

CodeDeploy のモニタリングでもう 1 つ重要な点は、CloudWatch のアラームの対象外の項目を手動 でモニタリングすることです。CodeDeploy、CloudWatch、およびその他の AWS コンソールダッ シュボードには、 AWS 環境の状態がat-a-glanceビューが表示されます。CodeDeploy デプロイのロ グファイルを確認することもお勧めします。

- CodeDeploy コンソールは以下で表示されます。
 - デプロイのステータス
 - リビジョンのデプロイを最後に試みた日時と、最後に成功した日時
 - デプロイの成功、失敗、スキップ、進行中のインスタンス数
 - オンプレミスインスタンスのステータス
 - オンプレミスインスタンスが登録、または登録解除された日時
- CloudWatch ホームページには、次の内容が表示されます。
 - 現在のアラームとステータス
 - アラームとリソースのグラフ
 - サービスのヘルスステータス

また、CloudWatch を使用して以下のことを行えます。

- 重視するサービスをモニタリングするためのカスタマイズしたダッシュボードを作成します
- ・メトリクスデータをグラフ化して、問題のトラブルシューティングを行い、傾向を確認する
- すべての AWS リソースメトリクスを検索して参照する

• 問題があることを通知するアラームを作成/編集する

トピック

- Monitoring Deployments with Amazon CloudWatch Tools
- Monitoring Deployments
- Monitoring Deployments with Amazon SNS Event Notifications

Amazon CloudWatch ツールを使用したデプロイのモニタリング

CodeDeploy デプロイは、Amazon CloudWatch Events、CloudWatch アラーム、およびAmazon CloudWatch Logs の CloudWatch ツールを使用して、モニタリングできます。

CodeDeploy エージェントおよびデプロイによって作成されたログを確認することにより、デプ ロイの障害の原因のトラブルシューティングを行うのに役立ちます。各インスタンスで個別に CodeDeploy ログを確認する代わりに、CloudWatch Logs を使用してすべてのログを一元的にモニタ リングできます。

CloudWatch アラームと CloudWatch Events を使用して CodeDeploy デプロイをモニタリングする 方法については、以下のトピックを参照してください。

トピック

- <u>CodeDeploy での CloudWatch アラームを使用したデプロイのモニタリング</u>
- <u>Amazon CloudWatch Events を使用したデプロイのモニタリング</u>

CodeDeploy での CloudWatch アラームを使用したデプロイのモニタリン グ

CodeDeploy オペレーションで使用するインスタンスや Amazon EC2 Auto Scaling グループの CloudWatch アラームを作成する方法を説明します。アラームは、指定期間にわたって単一のメト リクスを監視し、その値と複数期間に対するしきい値との比較結果に基づいて 1 つ以上のアクショ ンを実行します。CloudWatch アラームは、状態が変化したときにアクションを呼び出します(例え ば、OK から ALARM)。

ネイティブ CloudWatch アラーム機能では、デプロイで使用しているインスタンスが失敗したと きに Amazon SNS 通知を送信したり、インスタンスを停止、終了、再起動、復旧したりするな ど、CloudWatch でサポートされているアクションを指定できます。CodeDeploy オペレーションで は、デプロイグループに関連付けた CloudWatch アラームがアクティブ化されるたびにデプロイを停 止するようにデプロイグループを設定できます。

最大 10 個の CloudWatch アラームを CodeDeploy デプロイグループに関連付けることができます。 指定したアラームがアクティブ化した場合、デプロイは停止し、ステータスは [Stopped] に更新され ます。このオプションを使用するには、CodeDeploy サービスロールに対して CloudWatch へのアク セス許可を付与する必要があります。

CloudWatch コンソール での CloudWatch アラームの設定に関する詳細については、<u>Amazon</u> CloudWatch アラームの作成 は Amazon CloudWatch ユーザーガイド を参照してください。

CloudWatch アラームを CodeDeploy でデプロイグループに関連付ける方法については、 「<u>CodeDeploy でデプロイグループを作成する</u>」および「<u>CodeDeploy を使用して、デプロイグルー</u> <u>プの設定を変更します。</u>」を参照してください。

トピック

• CloudWatch アクセス権限を CodeDeploy サービスロールに付与する

CloudWatch アクセス権限を CodeDeploy サービスロールに付与する

デプロイで CloudWatch アラームモニタリングを使用する前に、CodeDeploy オペレーションで使用 するサービスロールに対して CloudWatch リソースへのアクセス許可を付与する必要があります。

CloudWatch アクセス権限を CloudWatch サービスロールに付与する

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/iam/</u>:// www.com」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。
- 3. AWS CodeDeploy オペレーションで使用するサービスロールの名前を選択します。
- 4. [アクセス許可] タブの [インラインポリシー] エリアで、[ロールポリシーの作成] を選択します。

-または-

[Create Role Policy] ボタンを使用できない場合は、[Inline Policies] エリアを拡張して、[click here] を選択します。

5. [Set Permissions] ページで、[Custom Policy] を選択し、次に [Select] を選択します。

- 6. [Review Policy] ページで、[Policy Name] フィールドに、このポリシーを識別するための名前 [CWA1arms] などを入力します。
- 7. [Policy Document] フィールドに以下を貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudwatch:DescribeAlarms",
            "Resource": "*"
        }
    ]
}
```

8. [ポリシーを適用]を選びます。

Amazon CloudWatch Events を使用したデプロイのモニタリング

Amazon CloudWatch Events を使用して、インスタンスの状態の変更、または CodeDeploy オペ レーションのデプロイ (「イベント」として) を検出して対処できます。次に、作成したルールに基 づいて CloudWatch Events はデプロイまたはインスタンスが、ルールで指定した状態になると、1 つ以上のターゲットアクションを呼び出します。状態変更のタイプに応じて、通知を送信、状態情報 を取得し、修正作業またはその他のアクションを取ることができます。CodeDeploy オペレーション の一部として CloudWatch Events を使用している場合は、以下のターゲットのタイプを選択できま す。

- AWS Lambda 関数
- Kinesis Streams
- ・ Amazon SQS キュー
- 組み込みターゲット (EC2 CreateSnapshot API call、EC2 RebootInstances API call、EC2 StopInstances API call および EC2 TerminateInstances API call)。
- ・ Amazon SNS トピック

次にユースケースをいくつか示します。

• Lambda 機能を使用して、デプロイが失敗するたびに Slack チャネルに通知を配信します。

- Kinesis ストリームにデプロイまたはインスタンスのデータをプッシュして、包括的でリアルタイムの状態モニタリングをサポートします。
- CloudWatch アラームアクションを使用して、指定したデプロイやインスタンスイベントが発生したときに、Amazon EC2 インスタンスを自動的に停止、終了、再起動、復旧します。

このトピックの残りの部分では、CodeDeploy の CloudWatch Events ルールを作成するための基本 的な手順について説明します。ただし CodeDeploy オペレーションで使用するイベントルールを作 成する前に、以下のことを実行する必要があります。

- CloudWatch Events の前提条件を完了します。詳細については、「<u>Amazon CloudWatch Events</u> 前提条件」を参照してください。
- CloudWatch Events のイベント、ルール、およびターゲットをしっかりと理解しておきます。詳細については、<u>Amazon CloudWatch Events とは</u>」および<u>「新しい CloudWatch Events AWS</u> リソースの変更を追跡して応答する」を参照してください。
- イベントのルールで使用するターゲットを作成します。

CodeDeploy の CloudWatch Events ルールを作成するには

- 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の Events] を選択します。
- 3. [ルールの作成] を選択してから、[イベントの選択] で [AWS CodeDeploy] を選択します。
- 4. 詳細タイプを指定します。
 - インスタンスとデプロイの両方のすべての状態変更に適用されるルールを作成するには、[Any detail type] を選択してから、ステップ6に進んでください。
 - インスタンスのみに適用するルールを作成するためには [Specific detail type] を選択してから、[CodeDeploy インスタンスの状態変更通知] を選択します。
 - デプロイのみに適用するルールを作成するためには [Specific detail type] を選択してから、
 [CodeDeploy デプロイの状態変更通知] を選択します。
- 5. ルールを適用する状態変更を指定します。
 - すべての状態変更に適用されるルールを作成するには、[Any state] を選択します。
 - いくつかの状態変更のみに適用されるルールを作成するためには、Specific state(s)を選択してから、リストから1つ以上のステータス値を選択します。次の表は、使用できるステータス値を一覧表示します。

デプロイのステータス値	インスタンスのステータス値
FAILURE	FAILURE
開始	開始
STOP	準備完了
QUEUED	SUCCESS
準備完了	
SUCCESS	

- 6. ルールが適用される CodeDeploy アプリケーションを指定します。
 - すべてのアプリケーションに適用するルールを作成するためには、[Any application] を選択し、ステップ8に進んでください。
 - 1 つのアプリケーションのみに適用するルールを作成するためには、[Specific application] を 選択してから、リストからアプリケーション名を選択します。
- 7. ルールが適用されるデプロイを指定します。
 - ・選択したアプリケーションと関連付けられたすべてのデプロイグループに適用されるルールを 作成するためには、[Any deployment group] を選択します。
 - 選択したアプリケーションに関連付けられる1つのデプロイグループのみに適用されるルー ルを作成するためには、[Specific deployment group(s)]を選択してから、リストからデプロイ グループ名を選択します。
- 8. ルール設定を確認して、イベントモニタリング要件を満たしていることを確認します。
- 9. [Targets] エリアで、[Add target*] を選択します。
- 10. Select target type リストで、このルールを使用するために準備したターゲットのタイプを選択 してから、そのタイプに必要な追加オプションを設定します。
- 11. 設定の詳細を選択します。
- 12. [Configure rule details] ページで、ルールの名前と説明を入力し、[State] ボックスを選択して、 すぐにルールを有効化します。
- 13. ルールが適切であることを確認したら、[Create rule] を選択します。

を使用したデプロイのモニタリング AWS CloudTrail

CodeDeploy は CloudTrail と統合されています。CloudTrail は、 AWS アカウントで CodeDeploy によって行われた、または CodeDeploy に代わって行われた API コールをキャプチャし、指定し た Amazon S3 バケットにログファイルを配信するサービスです。CloudTrail は、CodeDeploy コン ソール、 を介した CodeDeploy コマンド AWS CLI、または CodeDeploy APIs。CloudTrail によって 収集された情報を使用して、CodeDeploy に対してどのようなリクエストが行われたかを判断する ことができます。リクエストの作成元のソース IP アドレス、リクエストの実行者、リクエストの実 行日時などです。設定や有効化の方法など、CloudTrail の詳細については、「<u>AWS CloudTrail ユー</u> <u>ザーガイド</u>」を参照してください。

CloudTrail での CodeDeploy 情報

AWS アカウントで CloudTrail ログ記録が有効になっている場合、CodeDeploy アクションに対して 行われた API コールはログファイルで追跡されます。CodeDeploy レコードは、他の AWS サービス レコードと一緒にログファイルに書き込まれます。CloudTrail は、期間とファイルサイズに基づい て、新しいファイルをいつ作成して書き込むかを決定します。

CodeDeploy のアクションはすべてログに記録されます。このアクションについては、「<u>AWS</u> <u>CodeDeploy コマンドラインリファレンス</u>」および「<u>AWS CodeDeploy API リファレンス</u>」に記載さ れています。たとえば、デプロイを作成し、アプリケーションを削除し、アプリケーションリビジョ ンを登録する呼び出しは、CloudTrail ログファイルにエントリを生成します。

各ログエントリには、リクエストの生成者に関する情報が含まれます。ログのユーザー ID 情報は、 リクエストがルート認証情報またはユーザー認証情報を使用して行われたか、ロールまたはフェデ レーティッドユーザーの一時的なセキュリティ認証情報を使用して送信されたか、または別の AWS サービスによって送信されたかを判断するのに役立ちます。詳細については、<u>CloudTrail ログイベン</u> トリファレンスの userIdentity フィールドを参照してください。

必要な場合はログファイルを自身のバケットに保管できますが、ログファイルを自動的にアーカイブ または削除するにように Amazon S3 ライフサイクルルールを定義することもできます。デフォルト で、Amazon S3 のサーバー側の暗号化 (SSE) を使用して、ログファイルが暗号化されます。

新しいログファイルが配信されるときに、CloudTrail が Amazon SNS の通知を発行するようにでき ます。詳細については、<u>CloudTrail 用の Amazon SNS 通知の設定</u>を参照してください。

複数の AWS リージョンと複数の AWS アカウントからの CodeDeploy ログファイルを 1 つの Amazon S3 バケットに集約することもできます。詳細については、「<u>複数のリージョンから</u> <u>CloudTrail ログファイルを受けるとき</u>」を参照してください。

CodeDeploy ログファイルエントリの理解

CloudTrail ログファイルには、複数の JSON 形式イベントで構成されるひとつ以上のログエントリ を記録できます。ログエントリは任意の送信元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、任意のパラメータ、アクションの日時などに関する情報を含みます。ログエントリは、特 定の順序になるように生成されるわけではありません。つまり、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ありません。

以下の例は、CodeDeploy がデプロイグループのアクションを作成することを示す CloudTrail ログエ ントリを示しています。

```
{
 "Records": [{
  "eventVersion": "1.02",
  "userIdentity": {
  "type": "AssumedRole",
   "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
   "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
  "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
   "attributes": {
     "mfaAuthenticated": "false",
     "creationDate": "2014-11-27T03:57:36Z"
    },
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAI44QH8DHBEXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/example-role",
     "accountId": "123456789012",
     "userName": "example-role"
   }
  }
 },
  "eventTime": "2014-11-27T03:57:36Z",
  "eventSource": "codedeploy.amazonaws.com",
  "eventName": "CreateDeploymentGroup",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.11",
  "userAgent": "example-user-agent-string",
  "requestParameters": {
   "applicationName": "ExampleApplication",
```

```
"serviceRoleArn": "arn:aws:iam::123456789012:role/example-instance-group-role",
   "deploymentGroupName": "ExampleDeploymentGroup",
   "ec2TagFilters": [{
                "value": "CodeDeployDemo",
    "type": "KEY_AND_VALUE",
    "kev": "Name"
            }],
            "deploymentConfigName": "CodeDeployDefault.HalfAtATime"
  },
  "responseElements": {
   "deploymentGroupId": "7d64e680-e6f4-4c07-b10a-9e117EXAMPLE"
  },
  "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
  "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
 },
    ... additional entries ...
    1
}
```

Amazon SNS イベント通知を使用したデプロイのモニタリング

トリガーを CodeDeploy デプロイグループに追加すると、そのデプロイグループのデプロイまたは インスタンスに関連するイベントに関する通知を受信できます。これらの通知は、トリガーのアク ションの一部にした Amazon SNS トピックをサブスクライブする受信者に送信されます。

SMS メッセージまたは電子メールメッセージで、CodeDeploy イベントの通知を受信できます。 また、Amazon SQS キューへのメッセージの送信、または AWS Lambdaでの関数の呼び出しな ど、指定されたイベントが他の方法で発生したときに作成される JSON データを使用することも できます。デプロイおよびインスタンストリガーに提供される JSON データの構造については、 「CodeDeploy トリガーの JSON データ形式」を参照してください。

次の場合に、トリガーを使用して通知を受け取ることもできます。

- トラブルシューティングできるように、デプロイが失敗または停止したときに知る必要がある開発 者の場合。
- Amazon EC2 フリートの状態を監視するために、いくつのインスタンスが失敗したかを知る必要 があるシステム管理者の場合。

 デスクトップの電子メールクライアントのフォルダに様々なタイプの通知をルーティングするフィ ルタリングルールを使用して、デプロイおよびインスタンスのイベントの数を一目で把握したいマ ネージャーの場合。

次のいずれかのイベントタイプに対して、CodeDeploy デプロイグループごとに最大 10 のトリガー を作成できます。

インスタンスイベント

すべてのインスタンスイベント

• Success (成功)

失敗

起動済み

準備完了¹

デプロイイベント

- Success (成功)
- 失敗
- ・起動済み
- 停止
- ロールバック
- 準備完了1
- すべてのデプロイイベント

Blue/Green デプロイにのみ適用されます。最新のアプリケーションのリビジョンが、置き換え先 環境でインスタンスにインストールされており、元の環境からのトラフィックをロードバランサ ーの背後で再ルーティングすることができることを示します。詳細については、「<u>CodeDeploy</u> <u>でのデプロイグループの使用</u>」を参照してください。

トピック

- Amazon SNS アクセス許可を CodeDeploy サービスロールに付与する
- CodeDeploy イベントのトリガーを作成
- CodeDeploy デプロイグループのトリガーの編集
- CodeDeploy デプロイグループからトリガーを削除
- CodeDeploy トリガーの JSON データ形式

Amazon SNS アクセス許可を CodeDeploy サービスロールに付与する

トリガーが通知を生成する前に、CodeDeploy オペレーションで使用するサービスロールに は、Amazon SNS リソースへのアクセス許可を付与する必要があります。 サービスロールへの Amazon SNS アクセス許可の付与

- 1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/iam/</u>:// www.com」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。
- 3. AWS CodeDeploy オペレーションで使用するサービスロールの名前を選択します。
- 4. [アクセス許可] タブの [インラインポリシー] エリアで、[ロールポリシーの作成] を選択します。

-または-

[Create Role Policy] ボタンを使用できない場合は、[Inline Policies] エリアを拡張して、[click here] を選択します。

- 5. [Set Permissions] ページで、[Custom Policy] を選択し、次に [Select] を選択します。
- [ポリシーの確認] ページで、[ポリシー名] フィールドにポリシーを識別する名前 (SNSPublish など) を入力します。
- 7. [Policy Document] フィールドに以下を貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

8. [ポリシーを適用]を選びます。

CodeDeploy イベントのトリガーを作成

AWS CodeDeploy デプロイまたはインスタンスイベントの Amazon SNS (Amazon Simple Notification Service) トピックを発行するトリガーを作成できます。次に、そのイベントが発生した場合、関連付けられたトピックのすべてのサブスクライバーは、SMS メッセージまたは E メールメッセージなどの、トピックで指定されたエンドポイントを経由して通知を受信します。Amazon SNS では、トピックをサブスクライブするための複数の方法が用意されています。

トリガーを作成する前に、トリガーの参照先となる Amazon SNS トピックを設定する必要があ ります。詳細については、「<u>トピックの作成</u>」を参照してください。トピックを作成する際に は、Topic-group-us-west-3-deploy-fail または Topic-group-project-2-instancestop などの形式で、目的をわかりやすくする名前を付けることをお勧めします。

トリガーのために通知を送信する前に、Amazon SNS サービスロールに CodeDeploy アクセス許可 を付与する必要もあります。詳細については、<u>Amazon SNS アクセス許可を CodeDeploy サービス</u> ロールに付与する を参照してください。

トピックを作成した後、サブスクライバーを追加できます。トピックの作成、管理、およびサブスク ライブについての詳細は、「Amazon Simple Notification Service とは」を参照してください。

CodeDeploy イベントの通知を送信するトリガーの作成 (コンソール)

CodeDeploy コンソールを使用して、CodeDeploy イベント用のトリガーを作成できます。セット アッププロセスの最後に、アクセス許可およびトリガーの詳細の両方が正しくセットアップされてい ることを確認するためにテスト通知メッセージが送信されます。

CodeDeploy イベントのトリガーを作成

- 1. で AWS Management Console、 AWS CodeDeploy コンソールを開きます。
- 2. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 3. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- [アプリケーション]ページで、トリガーを追加するデプロイグループに関連付けられているアプ リケーションの名前を選択します。
- 5. [アプリケーションの詳細]ページで、トリガーを追加するデプロイグループを選択します。
- 6. [編集]を選択します。
- 7. [Advanced optional (詳細 オプション)] を展開します。
- 8. [トリガー] エリアで、[Create trigger (トリガーの作成)] を選択します。
- 9. [Create deployment trigger (デプロイトリガーの作成)] ペインで、次の操作を行います。

- a. [トリガー名] に、用途をわかりやすく示すトリガーの名前を入力します。Triggergroup-us-west-3-deploy-fail または Trigger-group-eu-central-instancestop などの形式をお勧めします。
- b. [イベント] で、通知を送信するために Amazon SNS トピックをトリガーするイベントタイ プまたはタイプを選択します。
- c. [Amazon SNS トピック] で、このトリガーの通知を送信するために作成したトピックの名 前を選択します。
- d. [Create trigger (トリガーの作成)] を選択します。CodeDeploy は、CodeDeploy および Amazon SNS トピックとの間のアクセスを正しく設定したことを確認するためのテスト通 知を送信します。トピックに対して選択したエンドポイントタイプに応じて、トピックをサ ブスクライブしている場合は、SMS メッセージまたは E メールメッセージで確認を受信し ます。
- 10. [Save changes] (変更の保存) をクリックします。

CodeDeploy イベントの通知を送信するトリガーの作成 (CLI)

CLIを使用して、デプロイグループを作成するときにトリガーを含めることも、既存のデプロイグ ループにトリガーを追加することもできます。

新しいデプロイグループの通知を送信するためのトリガーを作成するには

JSON ファイルを作成してデプロイグループを設定し、<u>デプロイメントグループの作成</u> コマンドの 使用 --cli-input-json オプションを実行します。

JSON ファイルを作成する最も簡単な方法は、--generate-cli-skeleton オプションを使用して JSON 形式のコピーを取得し、プレーンテキストエディターで必要な値を指定することです。

1. 次のコマンドを実行し、結果をプレーンテキストエディターにコピーします。

aws deploy create-deployment-group --generate-cli-skeleton

2. 既存の CodeDeploy アプリケーションの名前を追加します。

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentGroupName": "",
    "deploymentConfigName": "",
    "ec2TagFilters": [
```

```
{
             "Key": "",
             "Value": "",
             "Type": ""
        }
    ],
    "onPremisesInstanceTagFilters": [
        {
             "Key": "",
             "Value": "",
             "Type": ""
        }
    ],
    "autoScalingGroups": [
    ],
    "serviceRoleArn": "",
    "triggerConfigurations": [
        {
             "triggerName": "",
             "triggerTargetArn": "",
             "triggerEvents": [
                 ....
             ]
        }
    ]
}
```

3. 設定するパラメータの値を指定します。

<u>create-deployment-group</u> コマンドを使用する場合は、少なくとも次のパラメータ値を指定する 必要があります。

- applicationName: アカウントで既に作成されたアプリケーションの名前。
- deploymentGroupName: 作成するデプロイグループの名前。
- serviceRoleArn: アカウント内の CodeDeploy 用にセットアップする既存のサービスロー ルの ARN。詳細については、ステップ 2: CodeDeployのサービスのロールを作成する を参照 してください。

triggerConfigurations セクションで、次のパラメーターの値を指定します。

- triggerName: 簡単に識別できるように、トリガーに付与した名前。Trigger-group-uswest-3-deploy-fail または Trigger-group-eu-central-instance-stop などの形 式をお勧めします。
- triggerTargetArn: トリガーに関連付けるために作成した Amazon SNS トピックの ARN で以下の形式: arn:aws:sns:us-east-2:444455556666:NewTestTopic。
- triggerEvents: 通知をトリガーするイベントタイプまたはイベン
 ト。1つ以上のイベントタイプを指定し、複数のイベントタイプ名
 をカンマで区切ることができます(たとえば、"triggerEvents":
 ["DeploymentSuccess", "DeploymentFailure", "InstanceFailure"])。1つ以上の
 イベントタイプを追加すると、これらのすべてのタイプの通知は、それぞれの異なるトピック
 ではなく、指定したトピックに送信されます。次のイベントタイプから選択できます。
 - DeploymentStart
 - DeploymentSuccess
 - DeploymentFailure
 - DeploymentStop
 - DeploymentRollback
 - DeploymentReady (Blue/Green デプロイで置き換え先インスタンスにのみ適用します)
 - InstanceStart
 - InstanceSuccess
 - InstanceFailure
 - InstanceReady (Blue/Green デプロイで置き換え先インスタンスにのみ適用します)

以下の設定例では、dep-group-ghi-789-2 という名前のアプリケーション用の TestAppus-east-2 という名前のデプロイグループを作成し、デプロイの開始、成功、または失敗のた びに通知の送信を促すトリガーを作成します。

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "deploymentGroupName": "dep-group-ghi-789-2",
    "ec2TagFilters": [
        {
            [Key": "Name",
            "Value": "Project-ABC",
        }
}
```



 更新を JSON ファイルとして保存し、create-deployment-group コマンドを実行するときに -cli-input-json オプションを使用してそのファイルを呼び出します。

A Important ファイル名の前に必ず file: // を含めてください。このコマンドでは必須です。

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

作成プロセスの最後に、アクセス許可およびトリガーの詳細の両方が正しく設定されていること を示すテスト通知メッセージが届きます。

既存のデプロイグループの通知を送信するためのトリガーを作成するには

を使用して CodeDeploy イベントのトリガーを既存のデプロイグループ AWS CLI に追加するには、 デプロイグループを更新する JSON ファイルを作成し、 --cli-input-jsonオプションを使用し て <u>update-deployment-group</u> コマンドを実行します。

JSON ファイルを作成する最も簡単な方法は、get-deployment-group コマンドを実行し、JSON 形式 でデプロイグループの設定をコピーして、プレーンテキストエディターでパラメーター値を更新する ことです。

1. 次のコマンドを実行し、結果をプレーンテキストエディターにコピーします。

aws deploy get-deployment-group --application-name application --deployment-groupname deployment-group

- 2. 出力から次のものを削除します。
 - 出力の先頭の [{ "deploymentGroupInfo":]を削除します。
 - 出力の末尾の [}] を削除します。
 - [deploymentGroupId] を含む行を削除します。
 - [deploymentGroupName] を含む行を削除します。

テキストファイルのコンテンツは、次のようになります。

[triggerConfigurations] セクションで、[triggerEvents]、[triggerTargetArn]、および [triggerName] パラメーターのデータを追加します。トリガーの設定パラメータについての詳細は、「TriggerConfig」を参照してください。

テキストファイルのコンテンツは、次のようになります。このコードでは、デプロイの開始、成功、または失敗のたびに通知を送信するように求められます。

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
```

```
"autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Kev": "Name"
        }
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::4444555566666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

アップデートを JSON ファイルとして保存し、--cli-input-json オプションを使用して、<u>update-deployment-group</u> コマンドを実行します。必ず --current-deployment-group-name オプションを含めて、*filename* を JSON ファイルの名前に置き換えてください。

▲ Important ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

aws deploy update-deployment-group --current-deployment-group-name deploymentgroup-name --cli-input-json file://filename.json

作成プロセスの最後に、アクセス許可およびトリガーの詳細の両方が正しく設定されていること を示すテスト通知メッセージが届きます。

CodeDeploy デプロイグループのトリガーの編集

通知の要件が変更された場合は、新しいトリガーを作成するのではなく、トリガーを変更することが できます。

CodeDeploy トリガーの変更 (CLI)

を使用して、デプロイグループを更新するときに CodeDeploy イベントのトリガーの詳細 AWS CLI を変更するには、JSON ファイルを作成してデプロイグループのプロパティの変更を定義し、 -cli-input-jsonオプションを指定して update-deployment-group コマンドを実行します。

JSON ファイルを作成する最も簡単な方法は、get-deployment-group コマンドを実行して現在のデプ ロイグループの詳細を JSON 形式で取得し、プレーンテキストエディターで必要な値を編集するこ とです。

 以下のコマンドを実行します (アプリケーションおよびデプロイグループの名前を application および deployment-group に置き換えます)。

aws deploy get-deployment-group --application-name application --deployment-groupname deployment-group

- 2. コマンド結果をプレーンテキストエディターにコピーし、次のものを削除します。
 - 出力の先頭の [{ "deploymentGroupInfo":]を削除します。
 - ・ 出力の末尾の [}] を削除します。
 - [deploymentGroupId] を含む行を削除します。
 - [deploymentGroupName] を含む行を削除します。

テキストファイルのコンテンツは、次のようになります。

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "East-1-Instances",
            "Key": "Name"
```


- 3. 必要に応じてパラメーターを変更します。トリガーの設定パラメータについての詳細は、 「TriggerConfig」を参照してください。
- アップデートを JSON ファイルとして保存し、--cli-input-json オプションを使用して、<u>update-deployment-group</u> コマンドを実行します。必ず --current-deployment-group-name オプションを含めて、*filename* を JSON ファイルの名前に置き換えてください。

A Important

ファイル名の前に必ず file:// を含めてください。このコマンドでは必須です。

aws deploy update-deployment-group --current-deployment-group-name *deployment-group-name* --cli-input-json file://filename.json

作成プロセスの最後に、アクセス許可およびトリガーの詳細の両方が正しく設定されていることを示 すテスト通知メッセージが届きます。

CodeDeploy デプロイグループからトリガーを削除

デプロイグループごとに 10 個のトリガーの制限があるため、使用されなくなったトリガーを削除す ることをお勧めします。トリガーの削除は元に戻すことはできませんが、1 つを再作成することはで きます。

デプロイグループ (コンソール) からトリガーを削除

1. にサインイン AWS Management Console し、<u>https://console.aws.amazon.com/codedeploy</u>:// www.com」で CodeDeploy コンソールを開きます。

Note

「CodeDeploy の開始方法」で設定したのと同じユーザーでサインインします。

- 2. ナビゲーションペインで [デプロイ] を展開し、[アプリケーション] を選択します。
- [アプリケーション] ページで、トリガーを削除するデプロイグループに関連付けられているアプ リケーションの名前を選択します。
- 4. [アプリケーションの詳細]ページで、トリガーを削除するデプロイグループを選択します。
- 5. [編集]を選択します。
- 6. [Advanced optional (詳細 オプション)] を展開します。
- [トリガー] 領域で、削除するトリガーを選択し、[Delete trigger (トリガーを削除)] を選択します。
- 8. [Save changes] (変更の保存) をクリックします。

デプロイグループ (CLI) からトリガーを削除

CLI を使用してトリガーを削除するには、空のトリガー設定パラメータを使用して、<u>update-</u> <u>deployment-group</u> コマンドを呼び出し、次のように指定します。

- デプロイグループに関連付けられたアプリケーションの名前。アプリケーション名のリストを表示 するには、[list-applications] コマンドを呼び出します。
- アプリケーションに関連付けられたデプロイグループの名前。デプロイグループ名のリストを表示 するには、[list-deployment-groups] コマンドを呼び出します。

例:

aws deploy update-deployment-group --application-name application-name --currentdeployment-group-name deployment-group-name --trigger-configurations

CodeDeploy トリガーの JSON データ形式

デプロイまたはインスタンスのトリガーが、Amazon SQS キューへのメッセージの送信、または AWS Lambdaでの関数の呼び出しなどのカスタム通知ワークフローでアクティブ化されたときに作 成される JSON 出力を使用できます。

Note

このガイドでは、JSON を使用して通知を設定する方法については説明していませ ん。Amazon SNS を使用して Amazon SQS キューにメッセージを送信する方法の詳細 については、<u>Amazon SNS を Amazon SQS キューへメッセージ送信</u> を参照してくださ い。Amazon SNS を使用して Lambda 関数を呼び出す方法の詳細については、<u>Amazon SNS</u> デベロッパーガイドの Lambda 関数の呼び出しを参照してください。

次の例は、CodeDeploy トリガーで使用可能な JSON 出力の構造を示しています。

インスタンスベースのトリガー用 JSON 出力のサンプル

```
{
    "region": "us-east-2",
    "accountId": "111222333444",
    "eventTriggerName": "trigger-group-us-east-instance-succeeded",
    "deploymentId": "d-75I7MBT7C",
    "instanceId": "arn:aws:ec2:us-east-2:444455556666:instance/i-496589f7",
    "lastUpdatedAt": "1446744207.564",
    "instanceStatus": "Succeeded",
    "lifecycleEvents": [
        {
            "LifecycleEvent": "ApplicationStop",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744188.595",
            "EndTime": "1446744188.711"
        },
        {
            "LifecycleEvent": "BeforeInstall",
            "LifecycleEventStatus": "Succeeded",
```

```
"StartTime": "1446744189.827",
"EndTime": "1446744190.402"
}
//More lifecycle events might be listed here
]
}
```

デプロイベースのトリガー用 JSON 出力のサンプル

```
{
    "region": "us-west-1",
    "accountId": "111222333444",
    "eventTriggerName": "Trigger-group-us-west-3-deploy-failed",
    "applicationName": "ProductionApp-us-west-3",
    "deploymentId": "d-75I7MBT7C",
    "deploymentGroupName": "dep-group-def-456",
    "createTime": "1446744188.595",
    "completeTime": "1446744190.402",
    "deploymentOverview": {
        "Failed": "10",
        "InProgress": "0",
        "Pending": "0",
        "Skipped": "0",
        "Succeeded": "0"
    },
    "status": "Failed",
    "errorInformation": {
        "ErrorCode": "IAM_ROLE_MISSING",
        "ErrorMessage": "IAM Role is missing for deployment group: dep-group-def-456"
    }
}
```

セキュリティ in AWS CodeDeploy

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 お客様とお客様の間の責任共有です。<u>責任共有モデル</u>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 AWS クラウドで 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。 AWS CodeDeploy に適用されるコンプラ イアンスプログラムの詳細については、AWS「コンプライアンスプログラムによる対象範囲内の サービス」を参照してください。
- クラウド内のセキュリティ 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、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律や規制といった他の要因 についても責任を担います。

このドキュメントは、CodeDeploy を使用する際の責任共有モデルの適用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 CodeDeploy を設定する方法について説明します。また、CodeDeploy 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

- AWS CodeDeploy でのデータ保護
- AWS CodeDeployのためのアイデンティティおよびアクセス管理
- <u>CodeDeploy でのロギングとモニタリング</u>
- AWS CodeDeploy のコンプライアンス検証
- 耐障害性 in AWS CodeDeploy
- インフラストラクチャセキュリティ in AWS CodeDeploy

AWS CodeDeploy でのデータ保護

責任 AWS <u>共有モデル</u>、 AWS CodeDeploy でのデータ保護に適用されます。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、<u>データプライバシー</u> に関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリ ティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事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 Identity Center 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 が必須で、TLS 1.3 をお勧めします。
- 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、内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 AWS のサービス SDK を使用して CodeDeploy AWS CLIまたは他の 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。

インターネットトラフィックのプライバシー

CodeDeploy は、EC2 インスタンス、 Lambda 関数、Amazon ECS、オンプレミスサーバーをサ ポートするフルマネージド型のデプロイサービスです。EC2 インスタンスとオンプレミスサーバー に関しては、ホストベースのエージェントは TLS を使用して CodeDeploy と通信します。 現在、エージェントからサービスへの通信にはアウトバウンドインターネット接続が必要で、それに よってエージェントは、公開された CodeDeploy および Amazon S3 サービスエンドポイントと通信 できるようになります。仮想プライベートクラウドでは、インターネットゲートウェイ、企業ネット ワークへのサイト間 VPN 接続、または直接接続を使用してこれを実現できます。

CodeDeploy エージェントは HTTP プロキシをサポートします。

Amazon VPC エンドポイントは AWS PrivateLink、特定のリージョンで CodeDeploy で使用できま す。詳細については、「<u>Amazon Virtual Private Cloud で CodeDeploy を使用</u>」を参照してくださ い。

Note

CodeDeploy エージェントは、Amazon EC2 オンプレミスのコンピューティングプラット フォームにデプロイする場合にのみ必要です。エージェントは、Amazon ECS または AWS Lambda コンピューティングプラットフォームを使用するデプロイには必要ありません。

保管中の暗号化

カスタマーコードは CodeDeploy に保存されません。デプロイサービスとして、CodeDeploy は EC2 インスタンスまたはオンプレミスサーバーで稼働している CodeDeploy エージェントにコマン ドを送信します。次に、CodeDeploy エージェントは TLS を使用してコマンドを実行します。デプ ロイ、デプロイ設定、デプロイグループ、アプリケーション、およびアプリケーションリビジョンの サービスモデルデータは Amazon DynamoDB に保存され AWS 所有のキー、CodeDeploy が所有お よび管理する を使用して保管時に暗号化されます。詳細については、<u>AWS 所有のキー</u>を参照して ください。

転送中の暗号化

CodeDeploy エージェントは、CodeDeploy とのすべての通信をポート 443 を介して開始します。 エージェントは CodeDeploy をポーリングしてコマンドをリッスンします。CodeDeploy エージェン トはオープンソースです。すべてのサービスからサービスへの通信およびクライアントからサービス への通信は、TLS を使用して転送中に暗号化されます。これにより、CodeDeploy と Amazon S3 な どの他のサービス間で転送されるお客様データが保護されます。

暗号化キーの管理

お客様が管理する必要のある暗号化キーはありません。CodeDeploy サービスモデルデータは AWS 所有のキー、CodeDeploy が所有および管理する を使用して暗号化されます。詳細について は、AWS 所有のキー を参照してください。

AWS CodeDeployのためのアイデンティティおよびアクセス管理

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、誰を認証 (サインイン) し、誰に CodeDeploy リソースの使用を承認する (アクセス許可を付与する) かを管理します。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

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- が IAM と AWS CodeDeploy 連携する方法
- AWS CodeDeploy の マネージド (事前定義) ポリシー
- <u>AWS マネージドポリシーへの CodeDeploy の更新</u>
- ・ AWS CodeDeploy アイデンティティベースポリシーの例
- AWS CodeDeploy ID とアクセスのトラブルシューティング
- CodeDeploy アクセス許可 リファレンス
- サービス間での不分別な代理処理の防止

対象者

AWS Identity and Access Management (IAM) の使用方法は、CodeDeploy で行う作業によって異な ります。

[Service user] (サービスユーザー) – CodeDeploy サービスを使用してジョブを実行する場合は、必要なアクセス許可と認証情報を管理者が用意します。さらに多くの CodeDeploy 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。CodeDeploy の機能にアク

セスできない場合は、<u>AWS CodeDeploy ID とアクセスのトラブルシューティング</u>を参照してください。

サービス管理者 – 社内の CodeDeploy リソースを担当している場合は、通常 CodeDeploy への全 面的アクセス権があります。サービスのユーザーがどの CodeDeploy 機能やリソースにアクセスす るかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユー ザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解して ください。会社で CodeDeploy を使用して IAM を利用する方法の詳細については、<u>が IAM と AWS</u> CodeDeploy 連携する方法 を参照してください。

IAM 管理者 – 管理者は、CodeDeploy へのアクセスを管理するポリシー作成方法の詳細について確認 する場合があります。IAM で使用できる CodeDeploy アイデンティティベースのポリシーの例を表 示するには、AWS CodeDeploy アイデンティティベースポリシーの例 を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証(にサイン イン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインイ ンできます。 AWS IAM Identity Center (IAM Identity Center) ユーザー、会社のシングルサインオン 認証、Google または Facebook 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーション が設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引 き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインイ ンできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド<u>」の「 へ</u> のサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインイ ンターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。 AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分 で署名する推奨方法の使用については、「IAM ユーザーガイド」の「<u>API リクエストに対するAWS</u> Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たと えば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことを お勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「<u>多要素認証</u>」お よび「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウ ント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサイ ンインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強く お勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実 行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストに ついては、「IAM ユーザーガイド」の「<u>ルートユーザー認証情報が必要なタスク</u>」を参照してくだ さい。

ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウ ント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保 有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただ し、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキー をローテーション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「長期的 な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照 してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインイ ンすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できま す。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。 例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許 可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは1人の人または1つのアプリケーションに一意に 関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー ザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細につ いては、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

<u>IAM ロール</u>は、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは IAM ユーザーに 似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受ける には AWS Management Console、<u>ユーザーから IAM ロール (コンソール) に切り替える</u>ことができ ます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタ ム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「<u>ロー</u> ルを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス フェデレーティッド ID に許可を割り当てるには、ロール を作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID は ロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロール については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セッ トを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、 「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる 権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス 一部の は他の の機能 AWS のサービス を使用します AWS のサービ ス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプ リケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスで は、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこ れを行う場合があります。
 - 転送アクセスセッション (FAS) IAM ユーザーまたはロールを使用してアクションを実行する AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行する ことで、別のサービスの別のアクションがトリガーされることがあります。FAS は、を呼び出 すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービ ス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービス が他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け 取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必

要です。FAS リクエストを行う際のポリシーの詳細については、「<u>転送アクセスセッション</u>」 を参照してください。

- サービスロール サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができま す。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを 作成する」を参照してください。
- サービスにリンクされたロール 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。
- Amazon EC2 で実行されているアプリケーション IAM ロールを使用して、EC2 インスタンスで 実行され、AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を 管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 イン スタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするに は、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロ ファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を 取得できます。詳細については、「IAM ユーザーガイド」の「<u>Amazon EC2 インスタンスで実行</u> <u>されるアプリケーションに IAM ロールを使用して許可を付与する</u>」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。 ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を 定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッ ション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限によ り、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメ ント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、 「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

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。

デフォルトでは、ユーザーやロールに権限はありません。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。その後、管理者 はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。 IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例え ば、iam:GetRole アクションを許可するポリシーがあるとします。そのポリシーを持つユーザー は、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

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ポリシーは、IAM ユーザーグループ、ユーザーのグループ、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「<u>カスタマー管理ポリ</u> 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。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類 できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれてい ます。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロン ポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシー が含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法について は、「IAM ユーザーガイド」の「<u>管理ポリシーとインラインポリシーのいずれかを選択する</u>」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。

- アクセス許可の境界 アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。Principalフィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- 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 をグループ化して一元管理するためのサービス AWS アカウント 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 を含むメンバー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 AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「<u>サービスコントロールポリシー (SCP)</u>」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アクセス許可 を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs<u>「リソースコントロールポリ</u> シー (RCPs」を参照してください。AWS のサービス
- セッションポリシー セッションポリシーは、ロールまたはフェデレーションユーザーの一時的な セッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果として セッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポ リシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もありま す。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細について は、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解する のがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどう か AWS を決定する方法については、「IAM ユーザーガイド」の<u>「ポリシー評価ロジック</u>」を参照し てください。

が IAM と AWS CodeDeploy 連携する方法

IAM を使用して CodeDeploy へのアクセスを管理する前に、CodeDeploy で使用できる IAM 機能に ついて理解しておきましょう。詳細については、IAM ユーザーガイド の「<u>IAM と連携するAWS サー</u> <u>ビス</u>」を参照してください。

トピック

- CodeDeploy アイデンティティベースのポリシー
- CodeDeploy のリソースベースのポリシー
- CodeDeploy タグに基づく認可
- CodeDeploy IAM ロール

CodeDeploy アイデンティティベースのポリシー

IAM のアイデンティティベースポリシーでは、許可または拒否するアクションとリソース、および アクションが許可または拒否される条件を指定できます。CodeDeploy はアクション、リソース、お よび条件キーをサポートします。JSON ポリシーで使用する要素については、<u>IAM ユーザーガイド</u> 内の「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。

アクション

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。

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。

CodeDeploy のポリシーアクションは、アクションの前に codedeploy: プレフィックスを使用し ます。例えば、codedeploy:GetApplication 許可は、GetApplication オペレーションを実 行するためのアクセス許可をユーザーに付与します。ポリシーステートメントにはAction または NotAction 要素を含める必要があります。CodeDeploy は、このサービスで実行できるタスクを記 述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには次のようにコンマで区切ります。

"Action": ["codedeploy:action1", "codedeploy:action2"

ワイルドカード (*) を使用して複数アクションを指定できます。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": "ec2:Describe*"

が IAM と AWS CodeDeploy 連携する方法

CodeDeploy アクションのリストを表示するには、IAM ユーザーガイド の 「<u>AWS CodeDeployに</u> よって定義されたアクション」を参照してください。

すべての CodeDeploy API アクションとそれらが適用されるリソースの表については、<u>CodeDeploy</u> アクセス許可 リファレンス を参照してください。

リソース

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。ステートメ ントにはResource または NotResource 要素を含める必要があります。ベストプラクティスとし て、<u>アマゾン リソースネーム (ARN)</u>を使用してリソースを指定します。これは、リソースレベルの 許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限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": "*"

例えば、以下のように ARN を使用して、ステートメント内でデプロイグループ (*myDeploymentGroup*)を指定できます。

"Resource": "arn:aws:codedeploy:uswest-2:123456789012:deploymentgroup:myApplication/myDeploymentGroup"

以下のようにワイルドカード文字 (*) を使用して、特定のアカウントに属するすべてのデプロイグ ループを指定することもできます。

"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*"

すべてのリソースを指定する場合、または API アクションが ARN をサポートしていない場合は、以 下の要領で、Resource エレメント内でワイルドカード文字 (*) を使用します。

"Resource": "*"

が IAM と AWS CodeDeploy 連携する方法

一部の CodeDeploy API アクションは複数のリソースを受け入れます (例:
 BatchGetDeploymentGroups)。単一のステートメントに複数のリソースを指定するには、以下のようにコンマで ARN を区切ります。

"Resource": ["arn1", "arn2"]

CodeDeploy には、CodeDeploy リソースを操作するための一連のオペレーションが用意されていま す。使用可能なオペレーションのリストについては、「<u>CodeDeploy アクセス許可 リファレンス</u>」 を参照してください。

CodeDeploy リソースタイプとその ARN のリストを表示するには、IAM ユーザーガイド の <u>AWS</u> <u>CodeDeploy「で定義されるリソース」</u>を参照してください。各リソースの ARN を指定できるアク ションの詳細については、AWS CodeDeployで定義されたアクション を参照してください。

CodeDeploy リソースとオペレーション

CodeDeploy では、プライマリリソースはデプロイグループです。ポリシーで Amazon リソース ネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。CodeDeploy は、アプリ ケーション、デプロイ設定、インスタンスなど、デプロイグループで使用できる他のリソースをサ ポートします。これらは サブリソースと呼ばれます。これらのリソースとサブリソースには、一意 の ARN が関連付けられています。詳細については、Amazon Web Services 全般のリファレンス の 「Amazon リソースネーム (ARN)」を参照してください。

リソースタイプ	ARN 形式
デプロイグループ	<pre>arn:aws:codedeploy: region:account-id :deployme ntgroup: application-name /deployment-group-name</pre>
アプリケーション	<pre>arn:aws:codedeploy: region:account-id :applicat ion: application-name</pre>
デプロイ設定	<pre>arn:aws:codedeploy: region:account-id :deployme ntconfig: deployment-configuration-name</pre>
インスタンス	<pre>arn:aws:codedeploy: region:account-id :instance / instance-ID</pre>
すべての CodeDeploy リソース	arn:aws:codedeploy:*

リソースタイプ	ARN 形式		
特定のリージョンの特 定アカウントが所有す るすべての CodeDeploy リソース	arn:aws:codedeploy:	region:account-id	:*

Note

のほとんどのサービスは、ARN でコロン (:) またはスラッシュ (/) を同じ文字として AWS 扱います。 ARNs ただし、CodeDeploy では、リソースパターンとルールで完全一致が使用されます。イベントパターンを作成するときは、リソースの ARN 構文と一致するように正しい ARN 文字を使用してください。

条件キー

CodeDeploy にはサービス固有条件キーがありませんが、いくつかのグローバル条件キーの使用がサ ポートされています。条件キーの詳細については、<u>IAM ユーザーガイド</u>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。

例

CodeDeploy アイデンティティベースのポリシーの例については、<u>AWS CodeDeploy アイデンティ</u> ティベースポリシーの例 を参照してください。

CodeDeploy のリソースベースのポリシー

CodeDeploy では、リソースベースのポリシーはサポートされていません。詳細なリソースベースの ポリシーページの例を表示するには、<u>「リソースベースのポリシーの使用 AWS Lambda</u>」を参照し てください。

CodeDeploy タグに基づく認可

CodeDeployは、リソースのタグ付けやタグに基づいたアクセスの制御をサポートしていません。

CodeDeploy IAM ロール

IAM ロールは、特定のアクセス許可を持つ AWS アカウントのエンティティです。

CodeDeploy での一時的な認証情報の使用

ー時的な認証情報を使用して、フェデレーションでサインインする、IAM 役割を引き受ける、また はクロスアカウント役割を引き受けることができます。一時的なセキュリティ認証情報を取得するに は、AssumeRole や GetFederationToken などの AWS STS API オペレーションを呼び出します。

CodeDeploy は、一時的な認証情報の使用をサポートしています。

サービスにリンクされた役割

CodeDeploy は、サービスにリンクされたロールをサポートしていません。

サービス役割

この機能により、ユーザーに代わってサービスが<u>サービス役割</u>を引き受けることが許可されます。 この役割により、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを 完了することが許可されます。サービスロールは AWS アカウントに表示され、アカウントによって 所有されます。つまり、ユーザーは、このロールのアクセス許可を変更できます。ただし、それによ り、サービスの機能が損なわれる場合があります。

CodeDeploy はサービスロールをサポートします。

CodeDeploy での IAM ロールの選択

CodeDeploy でデプロイグループリソースを作成する場合、 CodeDeploy がユーザーの代わりに自動 的に Amazon EC2 にアクセスすることを許可するロールを選択する必要があります。サービスロー ルあるいはサービスにリンクされたロールを以前に作成している場合、CodeDeploy は選択できる ロールのリストを表示します。EC2 インスタンスの起動と停止へのアクセスを許可するロールを選 択することが重要です。

AWS CodeDeploy の マネージド (事前定義) ポリシー

AWS は、 によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多くの 一般的なユースケースに対処します AWS。これらの AWS管理ポリシーは、一般的なユースケース のアクセス許可を付与するため、どのアクセス許可が必要かを調査する必要がなくなります。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。

トピック

- CodeDeploy の AWS 管理ポリシーのリスト
- CodeDeploy のマネージドポリシーと通知

CodeDeploy の AWS 管理ポリシーのリスト

アカウントのユーザーにアタッチできる以下の AWS 管理ポリシーは、CodeDeploy に固有です。

• AWSCodeDeployFullAccess: CodeDeployへの全面的なアクセス権を付与します。

Note

AWSCodeDeployFullAccess は、Amazon EC2 や Amazon S3 などのアプリケーショ ンをデプロイするために必要な他のサービスの操作にアクセス権限を提供しませ ん。CodeDeploy に固有の操作にのみアクセス権限を提供します。

- AWSCodeDeployDeployerAccess: リビジョンを登録してデプロイする権限を許可します。
- AWSCodeDeployReadOnlyAccess: CodeDeploy への読み取り専用アクセス権を付与します。
- AWSCodeDeployRole: CodeDeploy に以下を許可します
 - ・インスタンスのタグを読み取る、または Amazon EC2 Auto Scaling グループ名により Amazon EC2 インスタンスを識別する
 - Amazon EC2 Auto Scaling グループ、ライフサイクルフック、スケーリングポリシー、ウォームプールの機能の読み取り、作成、更新、削除を行う
 - Amazon SNS トピックに情報を公開
 - Amazon CloudWatch アラームに関する基本的な情報を取得
 - Elastic Load Balancing サービスでのリソースの読み取りと更新

ポリシーには、次の規定が含まれます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
          "autoscaling:CompleteLifecycleAction",
          "autoscaling:DeleteLifecycleHook",
          "autoscaling:DescribeAutoScalingGroups",
```

"autoscaling:DescribeLifecycleHooks", "autoscaling:PutLifecycleHook", "autoscaling:RecordLifecycleActionHeartbeat", "autoscaling:CreateAutoScalingGroup", "autoscaling:CreateOrUpdateTags", "autoscaling:UpdateAutoScalingGroup", "autoscaling:EnableMetricsCollection", "autoscaling:DescribePolicies", "autoscaling:DescribeScheduledActions", "autoscaling:DescribeNotificationConfigurations", "autoscaling:SuspendProcesses", "autoscaling:ResumeProcesses", "autoscaling:AttachLoadBalancers", "autoscaling:AttachLoadBalancerTargetGroups", "autoscaling:PutScalingPolicy", "autoscaling:PutScheduledUpdateGroupAction", "autoscaling:PutNotificationConfiguration", "autoscaling:DescribeScalingActivities", "autoscaling:DeleteAutoScalingGroup", "autoscaling:PutWarmPool", "ec2:DescribeInstances", "ec2:DescribeInstanceStatus", "ec2:TerminateInstances", "tag:GetResources", "sns:Publish", "cloudwatch:DescribeAlarms", "cloudwatch:PutMetricAlarm", "elasticloadbalancing:DescribeLoadBalancers", "elasticloadbalancing:DescribeLoadBalancerAttributes", "elasticloadbalancing:DescribeInstanceHealth", "elasticloadbalancing:RegisterInstancesWithLoadBalancer", "elasticloadbalancing:DeregisterInstancesFromLoadBalancer", "elasticloadbalancing:DescribeTargetGroups", "elasticloadbalancing:DescribeTargetGroupAttributes", "elasticloadbalancing:DescribeTargetHealth", "elasticloadbalancing:RegisterTargets", "elasticloadbalancing:DeregisterTargets"], "Resource": "*" }]

}

- AWSCodeDeployRoleForLambda: CodeDeploy に、デプロイに必要な AWS Lambda およびその 他のリソースへのアクセス許可を付与します。
- AWSCodeDeployRoleForECS: CodeDeploy に Amazon ECS およびデプロイに必要なその他のリ ソースへのアクセス許可を付与します。
- AWSCodeDeployRoleForECSLimited: CodeDeploy に Amazon ECS およびデプロイに必要なその他のリソースへのアクセス許可を付与します。ただし、下記を例外とします。
 - AppSpec ファイルの hooks セクションでは、名前が CodeDeployHook_ で始まる Lambda 関数のみを使用できます。詳細については、「<u>Amazon ECS のデプロイ向けの AppSpec の</u> 「hooks」セクション」を参照してください。
 - S3 バケットへのアクセスは、UseWithCodeDeploy の値の登録タグを持つ true で S3 バケットに限定されます。詳細については、「オブジェクトのタグ付け」を参照してください。
- AmazonEC2RoleforAWSCodeDeployLimited: CodeDeploy Amazon S3 バケット内のオブジェ クトを取得および一覧表示のアクセス権限を CodeDeploy に付与します。ポリシーには、次の規 定が含まれます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*/CodeDeploy/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
```



デプロイプロセスの一部に対するアクセス許可は、CodeDeploy を代行する他の 2 つのロールタイプ に付与されます。

- IAM インスタンスプロファイルは、Amazon EC2 インスタンス にアタッチする IAM ロールです。 このプロファイルには、アプリケーションが保存される Amazon S3 バケットまたは GitHub リポ ジトリへのアクセスに必要な権限が含まれます。詳細については、「ステップ 4: Amazon EC2 イ ンスタンス用の IAM インスタンスプロファイルを作成する」を参照してください。
- サービスロールは、AWS リソースにアクセスできるように AWS サービスにアクセス許可を付与 する IAM ロールです。サービスロールにアタッチするポリシーによって、サービスがアクセスで きる AWS リソースと、それらのリソースで実行できるアクションが決まります。CodeDeploy で は、サービスロールは、以下の目的で使用されます。
 - インスタンスに適用されているタグやインスタンスに関連付けられている Amazon EC2 Auto Scaling グループ名を読み取ることができます。これにより、CodeDeploy はアプリケーション をデプロイできるインスタンスを識別できます。
 - インスタンス、Amazon EC2 Auto Scaling グループ、および Elastic Load Balancing ロードバラ ンサーでオペレーションを実行するには。
 - 指定したデプロイまたはインスタンスイベントが発生したときに通知を送信できるよう
 に、Amazon SNS トピックに情報を公開すること。
 - CloudWatch アラームに関する情報を取得して、デプロイのアラームモニタリングを設定します。

その他の詳細については、「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」を参照して ください。

カスタム IAM ポリシーを作成して、CodeDeploy アクションとリソースのアクセス許可を付与する こともできます。こうしたカスタムポリシーを IAM ロールにアタッチし、ロールをアクセス許可が 必要なユーザーまたはグループに割り当てます。

AWS CodeDeploy の マネージド (事前定義) ポリシー

API バージョン 2014-10-06 585

CodeDeploy のマネージドポリシーと通知

CodeDeploy は、デプロイへの重要な変更をユーザーに知らせるための通知をサポートしていま す。CodeDeploy のマネージドポリシーには、通知機能のポリシーステートメントが含まれます。詳 細については、通知とは を参照してください。

フルアクセスマネージドポリシー内の通知へのアクセス許可

AWSCodeDeployFullAccess マネージドポリシーには、通知へのフルアクセスを許可する次のス テートメントが含まれています。このマネージドポリシーが適用されたユーザーは、Amazon SNS 通知関連トピックの作成と管理、トピックへのユーザーの登録および登録解除、そして通知ルールの ターゲットとして選択するトピックの一覧表示を行うことができます。

```
{
       "Sid": "CodeStarNotificationsReadWriteAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:CreateNotificationRule",
           "codestar-notifications:DescribeNotificationRule",
           "codestar-notifications:UpdateNotificationRule",
           "codestar-notifications:DeleteNotificationRule",
           "codestar-notifications:Subscribe",
           "codestar-notifications:Unsubscribe"
      ],
       "Resource": "*",
       "Condition" : {
           "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"}
       }
  },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules",
           "codestar-notifications:ListTargets",
           "codestar-notifications:ListTagsforResource"
       ],
       "Resource": "*"
   },
   {
       "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
       "Effect": "Allow",
```

```
"Action": [
        "sns:CreateTopic",
        "sns:SetTopicAttributes"
    ],
    "Resource": "arn:aws:sns:*:*:codestar-notifications*"
},
{
  "Sid" : "CodeStarNotificationsChatbotAccess",
  "Effect" : "Allow",
  "Action" : [
    "chatbot:DescribeSlackChannelConfigurations"
  ],
  "Resource" : "*"
},
{
    "Sid": "SNSTopicListAccess",
    "Effect": "Allow",
    "Action": [
        "sns:ListTopics"
    ],
    "Resource": "*"
}
```

読み取り専用マネージドポリシー内の通知へのアクセス許可

AWSCodeDeployReadOnlyAccess マネージドポリシーには、通知への読み取り専用アクセスを許 可する以下のステートメントが含まれています。このマネージドポリシーが適用されたユーザーは、 リソースの通知を表示することはできますが、リソースの作成や管理、リソースへのサブスクライブ を行うことはできません。

```
{
    "Sid": "CodeStarNotificationsPowerUserAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications:DescribeNotificationRule"
    ],
    "Resource": "*",
    "Condition" : {
        "ArnLike" : {"codestar-notifications:NotificationsForResource" :
    "arn:aws:codedeploy:*:*:application:*"}
    }
    },
    {
}
```

```
"Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications:ListNotificationRules"
],
    "Resource": "*"
}
```

他の管理ポリシーの通知のアクセス許可

AWSCodeDeployDeployerAccess 管理ポリシーには、ユーザーが リソースの通知を作成、更新、 サブスクライブ、表示することを許可する以下のステートメントが含まれていますが、削除すること はできません。この管理ポリシーが適用されたユーザーは、通知の Amazon SNS トピックを作成お よび管理することもできます。

このポリシーには以下を実行するための許可が含まれます。

- codestar-notifications:CreateNotificationRule プリンシパルが通知を作成できる ようにします。
- codestar-notifications:DescribeNotificationRule プリンシパルが通知に関する情報を取得できるようにします。
- codestar-notifications:UpdateNotificationRule プリンシパルが通知を更新できる ようにします。
- codestar-notifications:Subscribe プリンシパルが通知更新をサブスクライブできるようにします。
- codestar-notifications:Unsubscribe プリンシパルが通知更新のサブスクリプションを 解除できるようにします。
- codestar-notifications:ListNotificationRules プリンシパルが通知ルールのリスト を取得できるようにします。
- codestar-notifications:ListTargets プリンシパルがターゲットのリストを取得できる ようにします。
- codestar-notifications:ListTagsforResource プリンシパルがタグのリストを取得で きるようにします。
- codestar-notifications:ListEventTypes プリンシパルがイベントタイプのリストを取 得できるようにします。
- chatbot:DescribeSlackChannelConfiguration プリンシパルが Slack チャネル設定に関 する情報を取得できるようにします。

 sns:ListTopics – プリンシパルが通知用の Amazon SNS トピックのリストを取得できるよう にします。

```
{
     "Sid" : "CodeStarNotificationsReadWriteAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:CreateNotificationRule",
       "codestar-notifications:DescribeNotificationRule",
       "codestar-notifications:UpdateNotificationRule",
       "codestar-notifications:Subscribe",
       "codestar-notifications:Unsubscribe"
     ],
     "Resource" : "*",
     "Condition" : {
       "ArnLike" : {
         "codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"
       }
     }
  },
   {
     "Sid" : "CodeStarNotificationsListAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:ListNotificationRules",
       "codestar-notifications:ListTargets",
       "codestar-notifications:ListTagsforResource",
       "codestar-notifications:ListEventTypes"
     ],
     "Resource" : "*"
  },
   {
     "Sid" : "CodeStarNotificationsChatbotAccess",
     "Effect" : "Allow",
     "Action" : [
       "chatbot:DescribeSlackChannelConfigurations"
     ],
     "Resource" : "*"
  },
   {
     "Sid" : "SNSTopicListAccess",
```

```
"Effect" : "Allow",
"Action" : [
    "sns:ListTopics"
],
    "Resource" : "*"
}
```

詳細については、「<u>AWS CodeStar Notifications の Identity and Access Management</u>」を参照してく ださい。

AWS マネージドポリシーへの CodeDeploy の更新

このサービスがこれらの変更の追跡を開始してからの CodeDeploy 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。このページへの変更に関する自動アラートを受けるに は、CodeDeploy ドキュメント履歴 の RSS フィードに登録してください。

変更	説明	日付
AWSCodeDeployDeplo yerAccess マネージドポ リシ - 既存のポリシーへの更 新	IAM ポリシー検証の変更をサ ポートするように codestar- notifications:Noti ficationsForResour ce アクションを更新しまし た。元のリソースarn:aws:c odedeploy:* がに更 新されましたarn:aws:c odedeploy:*:*:appl ication:* 。 このポリシーの詳細について は、他の管理ポリシーの通知 のアクセス許可 を参照してく ださい。	2024年12月16日
AWSCodeDeployReadO nlyAccess マネージドポ リシ - 既存のポリシーへの更 新	IAM ポリシー検証の変更をサ ポートするように codestar- notifications:Noti ficationsForResour ce アクションを更新しまし	2024 年 12 月 16 日

AWS CodeDeploy

変更	説明	日付
	た。元のリソースarn:aws:c odedeploy:* がに更 新されましたarn:aws:c odedeploy:*:*:appl ication:* 。 このポリシーの詳細について は、読み取り専用マネージド ポリシー内の通知へのアクセ ス許可 を参照してください。	
AWSCodeDeployFullA ccess マネージドポリシ- 既存のポリシーへの更新	IAM ポリシー検証の変更をサ ポートするように codestar- notifications:Noti ficationsForResour ce アクションを更新しまし た。元のリソースarn:aws:c odedeploy:* がに更 新されましたarn:aws:c odedeploy:*:*:appl ication:* 。 このポリシーの詳細について は、フルアクセスマネージド ポリシー内の通知へのアクセ ス許可 を参照してください。	2024年12月16日

AWS CodeDeploy

変更	説明	日付
AWSCodeDeployRole マ ネージドポリシ - 既存のポリ シーへの更新	Elastic Load Balancing の変更 をサポートする elasticlo adbalancing:Descri beLoadBalancerAttr ibutes および elasticlo adbalancing:Descri beTargetGroupAttri butes アクションをポリシ ーステートメントに追加しま した。 このポリシーの詳細について は、 <u>AWSCodeDeployRole</u> を 参照してください。	2023年8月16日
AWSCodeDeployFullA ccess マネージドポリシ- 既存のポリシーへの更新	通知の変更をサポートする chatbot:ListMicros oftTeamsChannelCon figurations アクション をポリシーステートメントに 追加しました。 このポリシーの詳細について は、 <u>AWSCodeDeployRole</u> を 参照してください。	2023 年 5 月 11 日
AWSCodeDeployRole マ ネージドポリシ - 既存のポリ シーへの更新	Amazon EC2 Auto Scaling 認可の変更をサポートする autoscaling:Create OrUpdateTags アクション をポリシーステートメントに 追加しました。 このポリシーの詳細について は、 <u>AWSCodeDeployRole</u> を 参照してください。	2023年2月3日

AWS CodeDeploy

変更	説明	日付
AmazonEC2RoleforAW SCodeDeployLimited マネージドポリシ - 既存のポ リシーへの更新	s3:ListBucket アクション をs3:ExistingObjectT ag/UseWithCodeDepl oy 条件を含むポリシース テートメントから削除しまし た。 このポリシーの詳細について は、 <u>AmazonEC2RoleforAW</u> <u>SCodeDeployLimited</u> を参照し てください。	2021年11月22日
AWSCodeDeployRole マ ネージドポリシ - 既存のポリ シーへの更新	ブルー/グリーンデプロイ に 関して <u>Amazon EC2 Auto</u> <u>Scaling グループへのウォー</u> <u>ムプールの追加</u> をサポートす る autoscaling:PutWar mPool アクションが追加さ れました。 不要な重複アクションを削除 しました。	2021 年 5 月 18 日
CodeDeploy が変更の追跡を スタート	CodeDeploy は AWS 、管理ポ リシーの変更の追跡を開始し ました。	2021 年 5 月 18 日

AWS CodeDeploy アイデンティティベースポリシーの例

デフォルトでは、ユーザーには、CodeDeploy リソースを作成または変更するアクセス許可はありま せん。また、 AWS Management Console、 AWS CLI、または AWS API を使用してタスクを実行す ることはできません。必要な指定されたリソースに対して API オペレーションを実行するためのア クセス許可を IAM ロールに付与する IAM ポリシーを作成する必要があります。続いて、それらのア クセス許可が必要なユーザーまたはグループにその IAM ロールをアタッチします。 これらの JSON ポリシードキュメント例を使用して IAM の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「<u>JSON タブでのポリシーの作成</u>」を参照してく ださい。

CodeDeploy 内では、アイデンティティベースのポリシーは、デプロイプロセスに関連するさまざま なリソースに対するアクセス権限を管理するために使用されます。次のリソースタイプへのアクセス をコントロールできます。

- アプリケーションおよびアプリケーションリビジョン。
- ・デプロイ。
- デプロイ設定。
- インスタンスとオンプレミスインスタンス。

リソースポリシーによって制御される機能は、次の表にあるように、リソースタイプによって異なり ます。

リソースタイプ	機能
すべて	リソースの詳細の閲覧および一覧表示
アプリケーション	リソースの作成
デプロイ設定	リソースの削除
デプロイグループ	
デプロイ	デプロイの作成
	デプロイの停止
アプリケーションリビジョン	アプリケーションリビジョンの登録
アプリケーション	更新されたリソース
デプロイグループ	
オンプレミスインスタンス	インスタンスにタグを追加する
	インスタンスからタグを削除する

リソースタイプ	機能
	インスタンスを登録する
	インスタンスを登録解除する

次の例では、us-west-2 リージョンの WordPress_App という名前のアプリケーションに関連付 けられている WordPress_DepGroup という名前のデプロイグループを削除することをユーザーに 許可するアクセス許可ポリシーを示しています。

{
"Version": "2012-10-17",
"Statement" : [
{
"Effect" : "Allow",
"Action" : [
"codedeploy:DeleteDeploymentGroup"
],
"Resource" : [
"arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
]
}
]
}

トピック

- カスタマーマネージドポリシーの例
- ポリシーに関するベストプラクティス
- CodeDeploy コンソールの使用
- 自分の権限の表示をユーザーに許可する

カスタマーマネージドポリシーの例

このセクションでは、さまざまな CodeDeploy アクションのアクセス許可を付与するポリシー例を 示しています。これらのポリシーは、CodeDeploy API、 AWS SDK、または AWS CLIを使用してい るときに機能します。コンソールで実行するアクションには、追加のアクセス許可を付与する必要が あります。コンソールアクセス許可の付与の詳細については、「<u>CodeDeploy コンソールの使用</u>」を 参照してください。

Note

例はすべて、米国西部 (オレゴン) リージョン (us-west-2) を使用し、架空のアカウント ID を使用しています。

例

- 例 1: 単一の リージョンで CodeDeploy オペレーションを実行することを許可する
- 例 2: 単一のアプリケーションへのリビジョンの登録を許可する
- 例 3: 単一のデプロイグループへのデプロイの作成を許可する

例 1: 単一の リージョンで CodeDeploy オペレーションを実行することを許可する

次の例では、CodeDeploy オペレーションを us-west-2 リージョンでのみ実行するアクセス許可を 付与します。

```
{
    "Version": "2012-10-17",
    "Statement" : [
        {
            "Effect" : "Allow",
            "Action" : [
            "codedeploy:*"
        ],
        "Resource" : [
            "arn:aws:codedeploy:us-west-2:4444555566666:*"
        ]
        }
    ]
}
```

例 2: 単一のアプリケーションへのリビジョンの登録を許可する

次の例では、us-west-2 リージョンの Test で始まるすべてのアプリケーションのアプリケーショ ンリビジョンを登録するアクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement" : [
        {
         "Effect" : "Allow",
         "Action" : [
            "codedeploy:RegisterApplicationRevision"
        ],
         "Resource" : [
            "arn:aws:codedeploy:us-west-2:4444555566666:application:Test*"
        ]
      }
    ]
}
```

例 3: 単一のデプロイグループへのデプロイの作成を許可する

次の例では、WordPress_App という名前のアプリケーションに関連付けられた WordPress_DepGroup という名前のデプロイグループ用、ThreeQuartersHealthy という名前 のカスタムデプロイ構成用、および WordPress_App という名前のアプリケーションに関連付けら れたあらゆるアプリケーションリビジョン用のデプロイを作成することを許可します。これらのリ ソースはすべて us-west-2 リージョンにあります。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      1
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
```

```
"arn:aws:codedeploy:us-
west-2:444455556666:deploymentconfig:ThreeQuartersHealthy"
]
},
{
"Effect" : "Allow",
"Action" : [
"codedeploy:GetApplicationRevision"
],
"Resource" : [
"arn:aws:codedeploy:us-west-2:444455556666:application:WordPress_App"
]
}
```

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内の CodeDeploy リソースを誰かが作成、アクセ ス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに 料金が発生する可能性があります。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「<u>AWS マネージドポリシー</u>」または「<u>ジョブ機</u> 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「<u>IAM でのポリシーとアクセス許可</u>」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。例えば、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 AWS 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 Access Analyzer 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 Access Analyzer 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「<u>IAM Access Analyzer でポリシーを</u> 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「<u>IAM でのセキュリ</u> ティのベストプラクティス」を参照してください。

CodeDeploy コンソールの使用

CodeDeploy コンソールを使用する場合は、 AWS アカウントの他の AWS リソースを記述できる最 小限のアクセス許可のセットが必要です。CodeDeploy コンソール内で CodeDeploy を利用するに は、次のサービスのアクセス許可が必要です。

- Amazon EC2 Auto Scaling
- AWS CodeDeploy
- Amazon Elastic Compute Cloud
- エラスティックロードバランシング
- AWS Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

これらの最小限必要なアクセス許可よりも制限された IAM ポリシーを作成している場合、その IAM ポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しません。このような ユーザーが CodeDeploy コンソールを使用できることを保証するには、ユーザーに割り当てられた ロール対する AWSCodeDeployReadOnlyAccess マネージドポリシーのアタッチも行ってください。こちらは「AWS CodeDeploy の マネージド (事前定義) ポリシー」に記述があります。

AWS CLI または CodeDeploy 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。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表 示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、 または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可 が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
```

]

}

AWS CodeDeploy ID とアクセスのトラブルシューティング

次の情報は、CodeDeploy と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復 に役立ちます。

トピック

- iam:PassRole を実行する権限がありません
- AWS アカウント以外のユーザーに CodeDeploy リソースへのアクセスを許可したい

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更 新して CodeDeploy にロールを渡せるようにする必要があります。

ー部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成 する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロー ルを渡す権限が必要です。

以下のエラー例は、marymajor という IAM ユーザーがコンソールを使用して CodeDeploy でアク ションを実行しようとした場合に発生したものです。ただし、このアクションをサービスが実行する には、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可 がありません。

User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担 当者が管理者です。

AWS アカウント以外のユーザーに CodeDeploy リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成 できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまた はアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用し て、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- CodeDeploy がこれらの機能をサポートするかどうかについては、「<u>が IAM と AWS CodeDeploy</u> 連携する方法」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユー ザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」 を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユー ザーガイドの<u>「サードパーティー AWS アカウント が所有する へのアクセスを提供する</u>」を参照 してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の 「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「<u>IAM でのクロスアカウントのリソースへのアクセス</u>」を参照してください。

CodeDeploy アクセス許可 リファレンス

アクセスをセットアップし、IAM アイデンティティ (アイデンティティベースのポリシー) にアタッ チできるアクセス権限ポリシーを作成する際、以下の表をリファレンスとして使用できます。この 表には、各 CodeDeploy API オペレーション、アクションを実行するためのアクセス許可を付与で きるアクション、およびアクセス許可を付与するために使用するリソース ARN の形式が示されてい ます。アクションは、ポリシーの Action フィールドで指定します。ポリシーの Resource フィー ルドでリソース値として、ワイルドカード文字 (*) を使用して、または使用せずに ARN を指定しま す。

CodeDeploy ポリシーで AWS全体の条件キーを使用して、条件を表現できます。 AWS全体のキーの 完全なリストについては、IAM ユーザーガイドの「使用可能なキー」を参照してください。

アクションを指定するには、API オペレーション名 (例えば、codedeploy: や codedeploy:GetApplication)の前に codedeploy:CreateApplication プレフィックスを 使用します。単一のステートメントに複数のアクションを指定するには、コンマで区切ります (例え ば、"Action": ["codedeploy:action1", "codedeploy:action2"])。

ワイルドカード文字の使用

ARN でワイルドカード文字 (*) を使用して、複数のアクションまたはリソースを指定できます。例え ば、codedeploy:* は、すべての CodeDeploy アクションを指定し、codedeploy:Get* は、Get という単語で始まるすべての CodeDeploy アクションを指定します。次の例では、West で始まり、 名前が Test で始まるアプリケーションに関連付けられている名前を持つすべてのデプロイグループ にアクセス権限を付与します。

arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:Test*/West*

表に表示されている次のリソースでワイルドカードを使用できます。

- application-name
- deployment-group-name
- deployment-configuration-name
- instance-ID

ワイルドカードは *region* または *account-id* では使用できません。ワイルドカードの詳細につい ては、IAM ユーザーガイドの IAM ID を参照してください。

Note

各アクションの ARN ではリソースの後にコロン (:) が続きます。また、リソースの後にス ラッシュ (/) を使用できます。詳細については、「<u>CodeDeploy の ARN の例</u>」を参照してく ださい。

CodeDeploy API オペレーションおよびコミットされたコードのアクションで必要なアクセス権限

AddTagsToOnPremisesInstances

アクション:codedeploy:AddTagsToOnPremisesInstances

1つ以上のオンプレミスインスタンスにタグを追加するために必要です。

リソース: arn:aws:codedeploy:region:account-id:instance/instance-ID

BatchGetApplicationRevisions

アクション:codedeploy:BatchGetApplicationRevisions

ユーザーに関連付けられた複数のアプリケーションリビジョンに関する情報を取得するために必 要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

BatchGetApplications

アクション:codedeploy:BatchGetApplications

ユーザーに関連付けられた複数のアプリケーションに関する情報を取得するために必要です。

リソース: arn: aws: codedeploy: region: account-id: application: *

BatchGetDeploymentGroups

アクション:codedeploy:BatchGetDeploymentGroups

ユーザーに関連付けられた複数のデプロイグループに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentgroup:application-name/deployment-group-name

BatchGetDeploymentInstances

アクション:codedeploy:BatchGetDeploymentInstances

デプロイグループの一部である1つ以上のインスタンスに関する情報を取得するために必要で す。

リソース: arn: aws: codedeploy: region: account-

id:deploymentgroup:application-name/deployment-group-name

BatchGetDeployments

アクション:codedeploy:BatchGetDeployments

ユーザーに関連付けられた複数のデプロイに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentgroup:application-name/deployment-group-name

BatchGetOnPremisesInstances

アクション:codedeploy:BatchGetOnPremisesInstances

1つ以上のオンプレミスインスタンスに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:*

ContinueDeployment

アクション:codedeploy:ContinueDeployment

ブルー/グリーンデプロイ中、Elastic Load Balancing ロードバランサーを使用して、置き換え先 環境でのインスタンス登録を開始するために必要です。

リソース: arn:aws:codedeploy:*region:account-*

id:deploymentgroup:application-name/deployment-group-name

CreateApplication

アクション:codedeploy:CreateApplication

ユーザーに関連付けられたアプリケーションを作成するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

CreateDeployment ¹

アクション:codedeploy:CreateDeployment

ユーザーに関連付けられたアプリケーションのデプロイを作成するために必要です。

リソース: arn:aws:codedeploy:*region:account-*

id:deploymentgroup:application-name/deployment-group-name

CreateDeploymentConfig

アクション:codedeploy:CreateDeploymentConfig

ユーザーに関連付けられたカスタムデプロイ設定を作成するために必要です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentconfig/deployment-configuration-name

CreateDeploymentGroup

アクション:codedeploy:CreateDeploymentGroup

ユーザーに関連付けられたアプリケーションのデプロイグループを作成するために必要です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentgroup:application-name/deployment-group-name

DeleteApplication

アクション:codedeploy:DeleteApplication

ユーザーに関連付けられたアプリケーションを削除するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

DeleteDeploymentConfig

アクション:codedeploy:DeleteDeploymentConfig

ユーザーに関連付けられたカスタムデプロイ設定を削除するために必要です。

リソース: arn:aws:codedeploy:*region:accountid*:deploymentconfig/*deployment-configuration-name*

DeleteDeploymentGroup

アクション:codedeploy:DeleteDeploymentGroup

ユーザーに関連付けられたアプリケーションのデプロイグループを削除するために必要です。

リソース: arn:aws:codedeploy:region:accountid:deploymentgroup:application-name/deployment-group-name

DeregisterOnPremisesInstance

アクション:codedeploy:DeregisterOnPremisesInstance

オンプレミスインスタンスを登録解除するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:instance/*instance-ID* GetApplication

アクション:codedeploy:GetApplication

ユーザーに関連付けられた単一のアプリケーションに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

GetApplicationRevision

アクション:codedeploy:GetApplicationRevision

ユーザーに関連付けられたアプリケーションの単一のアプリケーションのリビジョンに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

GetDeployment

アクション:codedeploy:GetDeployment

ユーザーに関連付けられたアプリケーションのデプロイグループへの単一のデプロイに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentgroup:application-name/deployment-group-name

GetDeploymentConfig

アクション:codedeploy:GetDeploymentConfig

ユーザーに関連付けられた単一のデプロイ設定に関する情報を取得するために必要です。

リソース: arn: aws: codedeploy: region: account-

id:deploymentconfig/deployment-configuration-name

GetDeploymentGroup

アクション:codedeploy:GetDeploymentGroup

ユーザーに関連付けられたアプリケーションの単一のデプロイグループに関する情報を取得する ために必要です。

リソース: arn:aws:codedeploy:*region:accountid*:deploymentgroup:*application-name/deployment-group-name* GetDeploymentInstance

アクション:codedeploy:GetDeploymentInstance

ユーザーに関連付けられたデプロイの単一のインスタンスに関する情報を取得するために必要で す。

リソース: arn: aws: codedeploy: region: account-

id:deploymentgroup:application-name/deployment-group-name

GetOnPremisesInstance

アクション:codedeploy:GetOnPremisesInstance

単一のオンプレミスインスタンスに関する情報を取得するために必要です。

リソース:arn:aws:codedeploy:*region:account-id*:instance/instance-ID

ListApplicationRevisions

アクション:codedeploy:ListApplicationRevisions

ユーザーに関連付けられたアプリケーションのすべてのアプリケーションリビジョンに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*

ListApplications

アクション:codedeploy:ListApplications

ユーザーに関連付けられたすべてのアプリケーションに関する情報を取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*

ListDeploymentConfigs

アクション:codedeploy:ListDeploymentConfigs

ユーザーに関連付けられたすべてのデプロイ設定に関する情報を取得するために必要です。

リソース:arn:aws:codedeploy:*region:account-id*:deploymentconfig/*

ListDeploymentGroups

アクション:codedeploy:ListDeploymentGroups

ユーザーに関連付けられたアプリケーションのすべてのデプロイグループに関する情報を取得す るために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:deploymentgroup:*application-name*/*

ListDeploymentInstances

アクション:codedeploy:ListDeploymentInstances

ユーザーに関連付けられたデプロイのすべてのインスタンスに関する情報を取得するために必要 です。

リソース: arn:aws:codedeploy:region:account-

id:deploymentgroup:application-name/deployment-group-name

ListDeployments

アクション:codedeploy:ListDeployments

ユーザーに関連付けられたデプロイグループへのすべてのデプロイに関する情報、またはユー ザーに関連付けられたすべてのデプロイに関する情報を取得するために必要です。

リソース: arn: aws: codedeploy: region: account-

id:deploymentgroup:application-name/deployment-group-name

ListGitHubAccountTokenNames

アクション:codedeploy:ListGitHubAccountTokenNames

GitHub アカウントへの保存された接続の名前を一覧表示するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:*

ListOnPremisesInstances

アクション:codedeploy:ListOnPremisesInstances

1つ以上のオンプレミスインスタンス名のリストを取得するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:*

RegisterApplicationRevision

アクション:codedeploy:RegisterApplicationRevision

ユーザーに関連付けられたアプリケーションのアプリケーションリビジョンに関する情報を登録 するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

RegisterOnPremisesInstance

アクション:codedeploy:RegisterOnPremisesInstance

オンプレミスインスタンスを CodeDeploy で登録するために必要です。

リソース: arn:aws:codedeploy:region:account-id:instance/instance-ID

RemoveTagsFromOnPremisesInstances

アクション:codedeploy:RemoveTagsFromOnPremisesInstances

1つ以上のオンプレミスインスタンスからタグを削除するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:instance/*instance-ID* SkipWaitTimeForInstanceTermination

アクション:codedeploy:SkipWaitTimeForInstanceTermination

指定された待機時間をオーバーライドし、ブルー/グリーンデプロイでトラフィックが正常にルー ティングした直後に元の環境でインスタンスの削除を開始するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:instance/*instance-ID* StopDeployment

アクション:codedeploy:StopDeployment

ユーザーに関連付けられたアプリケーションのデプロイグループへの進行中のデプロイを停止す るために必要です。

リソース: arn: aws: codedeploy: region: account-

id:deploymentgroup:application-name/deployment-group-name

UpdateApplication ³

アクション:codedeploy:UpdateApplication

ユーザーに関連付けられたアプリケーションに関する情報を変更するために必要です。

リソース: arn:aws:codedeploy:*region:account-id*:application:*application-name*

UpdateDeploymentGroup ³

アクション:codedeploy:UpdateDeploymentGroup

ユーザーに関連付けられたアプリケーションで単一のデプロイグループに関する情報を変更する ために必要です。

リソース: arn:aws:codedeploy:*region:account-*

id:deploymentgroup:application-name/deployment-group-name

CreateDeployment アクセス許可を指定する場合は、デプロイ設定および GetDeploymentConfig の GetApplicationRevision アクセス許可、またはアプリケーション リビジョンへの RegisterApplicationRevision アクセス許可も指定する必要があります。

ListDeployments 特定のデプロイグループを指定する場合には有効ですが、ユーザーに関連付け られたすべてのデプロイを一覧表示する場合には有効ではありません。

UpdateApplication については、古いアプリケーション名と新しいアプリケーション名の両方 に対する UpdateApplication アクセス許可が必要です。デプロイグループの名前の変更を伴う UpdateDeploymentGroup アクションの場合、古いデプロイグループ名と新しいデプロイグループ 名の両方に対する UpdateDeploymentGroup アクセス許可が必要です。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス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の実行を強制できてしまう場合に生じる、セキュリティ上の 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。これを防ぐために、 は、アカウント内のリ 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を持つすべてのサービスのデータを保護す るのに役立つツール AWS を提供します。

リソースポリシーで <u>aws:SourceArn</u> および <u>aws:SourceAccount</u> のグローバル条件コンテキスト キーを使用して、CodeDeploy が別のサービスに付与する許可をそのリソースに制限することをお 勧めします。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。クロスサービスが使用できるように、アカウント内の任意のリソースを関連付けたい場合 は、aws:SourceAccount を使用します。

EC2/オンプレミス、 AWS Lambda、および通常の Amazon ECS デプロイの場合、 の値には CodeDeploy が IAM ロールを引き受けることができる CodeDeploy デプロイグループ ARN を含め るaws:SourceArn必要があります。

<u>を使用して作成された Amazon ECS ブルー/グリーンデプロイ AWS CloudFormation</u>の場合、 の値 には CodeDeploy が IAM ロールを引き受けることができる CloudFormation スタック ARN を含め るaws : SourceArn必要があります。

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 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 スを指定する場合には、未知部分を示すためにワイルドカード文字 (*) を使用します。

たとえば、EC2/オンプレミス、 AWS Lambda、または通常の Amazon ECS デプロイで次の信頼ポ リシーを使用できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                     "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                     "aws:SourceArn": "arn:aws:codedeploy:us-
east-1:111122223333:deploymentgroup:myApplication/*"
            }
        }
    ]
```

}

<u>を使用して作成された Amazon ECS ブルー/グリーンデプロイ AWS CloudFormation</u>では、以下を使 用できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                     "aws:SourceAccount": "111122223333"
                },
                "StringLike": {
                     "aws:SourceArn": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyCloudFormationStackName/*"
                }
            }
        }
    ]
}
```

CodeDeploy でのロギングとモニタリング

このセクションでは、CodeDeploy でのモニタリング、ログ記録、およびインシデントへの対応の概 要を説明します。

CodeDeploy とのすべてのインタラクションの監査

CodeDeploy は AWS CloudTrail、 AWS アカウントで CodeDeploy によって行われた、または CodeDeploy に代わって行われた API コールをキャプチャし、指定した S3 バケットにログファイ ルを配信するサービスである と統合されています。CloudTrail は、CodeDeploy コンソール、 AWS CLIを通した CodeDeploy コマンド、または CodeDeploy API からの API コールを直接キャプチャし ます。CloudTrail で収集された情報を使用して、CodeDeploy に対するリクエスト、リクエスト元の 出典 IP アドレス、リクエスト送信者、リクエスト日時などの詳細を確認できます。CloudTrail のよ り詳しい情報については、<u>AWS CloudTrail ユーザーガイド</u> 内の 「CloudTrail ログファイルの操作」 を参照してください。

CodeDeploy のデプロイで作成されたログデータを表示するには、Amazon CloudWatch エージェン トを設定して CloudWatch コンソール内の集計データを表示するか、インスタンスにサインインして ログファイルを確認します。詳細については、「<u>CodeDeploy エージェントログを CloudWatch に送</u> 信する」を参照してください。

アラートと障害管理

Amazon CloudWatch Events を使用して、CodeDeploy オペレーション内のインスタンスまたはデプ ロイ(「イベント」)のステートにおける変化を検出して対処できます。この場合、デプロイまたは インスタンスがルール内で指定した状態になると、作成したルールに基づいて、CloudWatch Events が1つ以上のターゲットアクションを呼び出します。状態変更のタイプに応じて、通知を送信、状 態情報を取得し、修正作業またはその他のアクションを取ることができます。CloudWatch Events を CodeDeploy オペレーションの一部としてを使用する場合、次のタイプのターゲットを選択でき ます。

- AWS Lambda 関数
- Kinesis Streams
- ・ Amazon SQS SQS キュー
- ・ 組み込みターゲット (CloudWatch アラームアクション)
- ・ Amazon SNS トピック

次にユースケースをいくつか示します。

- Lambda 機能を使用して、デプロイが失敗するたびに Slack チャネルに通知を配信します。
- Kinesis ストリームにデプロイまたはインスタンスのデータをプッシュして、包括的でリアルタイムの状態モニタリングをサポートします。
- CloudWatch アラームアクションを使用することで、指定したデプロイやインスタンスイベントが 発生したときに、EC2 インスタンスを自動的に停止、終了、再起動、もしくは復旧することがで きます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「<u>Amazon CloudWatch Events とは</u>」 を参照してください。

AWS CodeDeploy のコンプライアンス検証

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、<u>AWS のサービス「コンプライアンスプログラムによる範囲内</u>」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、<u>AWS 「 Compliance</u> ProgramsAssurance」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。 は、コンプライアン スに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする 手順を示します。
- HIPAA 対応サービスのリファレンス HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービス が HIPAA の対象となるわけではありません。
- <u>AWS コンプライアンスリソース</u> このワークブックとガイドのコレクションは、お客様の業界や 地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド コンプライアンスの観点から責任共有モデルを理解 します。このガイドは、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコント ロールを保護し、そのガイダンスに AWS のサービス マッピングするためのベストプラクティス をまとめたものです。
- 「デベロッパーガイド」の「ルールによるリソースの評価」 この AWS Config サービスは、リ ソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価 します。 AWS Config
- AWS Security Hub これにより AWS のサービス、内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。
- <u>Amazon GuardDuty</u> 不審なアクティビティや悪意のあるアクティビティがないか環境をモニタ リングすることで AWS アカウント、、ワークロード、コンテナ、データに対する潜在的な脅威

AWS のサービス を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワー クで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライア ンス要件に対応できます。

<u>AWS Audit Manager</u> – これにより AWS のサービス、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

耐障害性 in AWS CodeDeploy

AWS グローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心に 構築されています。AWS リージョンは、低レイテンシー、高スループット、高冗長ネットワークで 接続された複数の物理的に分離および分離されたアベイラビリティーゾーンを提供します。アベイラ ビリティーゾーンでは、ゾーン間で中断することなく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が高く、フォールトトレラントで、 スケーラブルです。

AWS リージョンと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。

インフラストラクチャセキュリティ in AWS CodeDeploy

マネージドサービスである AWS CodeDeploy は、ホワイトペーパー<u>「Amazon Web Services: セ</u> <u>キュリティプロセスの概要</u>」に記載されている AWS グローバルネットワークセキュリティ手順で保 護されています。

AWS 公開された API コールを使用して、ネットワーク経由で CodeDeploy にアクセスします。クラ イアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。TLS 1.3 以降 が推奨されます。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポート されている必要があります。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。

リクエストは、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のアクセ スキーを使用して署名する必要があります。または、<u>AWS Security Token Service</u> (AWS STS) を使 用して、一時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。

参照資料

リファレンス

トピック

- <u>CodeDeploy AppSpec ファイルのリファレンス</u>
- ・ CodeDeploy エージェント設定リファレンス
- AWS CloudFormation CodeDeploy リファレンスの テンプレート
- ・ Amazon Virtual Private Cloud で CodeDeploy を使用
- <u>CodeDeploy リソースキットリファレンス</u>
- CodeDeploy のクォータ

CodeDeploy AppSpec ファイルのリファレンス

このセクションは参照のみを目的としています。AppSpec ファイルの概念の概要について は、Application Specification Files を参照してください。

アプリケーション仕様ファイル (AppSpec ファイル) は、CodeDeploy がデプロイを管理するために 使用する <u>YAML</u> 形式または JSON 形式のファイルです。

Note

EC2/オンプレミスデプロイの AppSpec ファイルは、ローカルデプロイを実行しない限 り、appspec.yml という名前にする必要があります。詳細については、「<u>ローカルのデプ</u> <u>ロイを作成する。</u>」を参照してください。

トピック

- Amazon ECS コンピューティングプラットフォームの AppSpec ファイル
- ・ <u>AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル</u>
- ・ EC2/オンプレミスコンピューティングプラットフォームの AppSpec ファイル
- <u>AppSpec ファイル構造</u>
- AppSpec ファイルの例
- <u>AppSpec ファイルの間隔</u>

Amazon ECS コンピューティングプラットフォームの AppSpec ファイル

Amazon ECS コンピューティングプラットフォームアプリケーションでは、AppSpec ファイルは CodeDeploy が判断するために使用されます。

- Amazon ECS タスク定義ファイル これは、AppSpec ファイルの TaskDefinition 命令の ARN で指定されます。
- ・置き換えタスクのコンテナとポートは、Application Load Balancer または Network Load Balancer がデプロイ中にトラフィックを再ルーティングする場所を設定します。これは、AppSpec ファイ ルの LoadBalancerInfo 命令で指定されます。
- サービスが実行されるプラットフォームのバージョン、そのサブネット、およびそのセキュリティ グループなどの、Amazon ECS サービスに関するオプション情報。
- Amazon ECS のデプロイ中、ライフサイクルイベントに応じたフック中に実行される、オプションの Lambda 関数。詳細については、「<u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション</u>」を参照してください。

AWS Lambda コンピューティングプラットフォーム上の AppSpec ファイル

AWS Lambda コンピューティングプラットフォームアプリケーションの場合、AppSpec ファイルは CodeDeploy が以下を判断するために使用されます。

- ・ デプロイする Lambda 関数のバージョン。
- ・検証テストとして使用する Lambda 関数。

AppSpec ファイルは、YAML 形式または JSON 形式とすることができます。デプロイの作成時 に、AppSpec ファイルの内容を直接 CodeDeploy コンソールに入力することもできます。

EC2/オンプレミスコンピューティングプラットフォームの AppSpec ファ イル

アプリケーションで EC2/オンプレミスコンピューティングプラットフォームを使用する場 合、AppSpec ファイルは、appspec.yml という名前の YAML 形式のファイルで、アプリケーショ ンのソースコードのディレクトリ構造のルートに配置する必要があります。それ以外の場合、デプロ イは失敗します。CodeDeploy を決定するために使用されます。

- Amazon S3 または GitHub のアプリケーションリビジョンからインスタンスにインストールする 必要があるもの。
- デプロイライフサイクルイベントに応じて実行するライフサイクルイベントフック。

AppSpec ファイルを完了したら、デプロイするコンテンツとともに、アーカイブファイル (zip、tar、または圧縮 tar) にバンドルします。詳細については、「<u>CodeDeploy のアプリケーション</u> リビジョンの操作」を参照してください。

Note

tar および圧縮 tar アーカイブファイル形式 (.tar および .tar.gz) は、Windows Server インス タンスではサポートされていません。

アーカイブファイル (CodeDeploy では リビジョン と呼ばれます) をバンドルしたら、Amazon S3 バ ケットまたは Git レポジトリにアップロードします。次に、CodeDeploy を使用してリビジョンをデ プロイします。手順については、CodeDeploy でデプロイを作成する を参照してください。

EC2/オンプレミスコンピューティングプラットフォームのデプロイの appspec.yml は、リビジョン のルートディレクトリに保存されます。詳細については、<u>EC2/オンプレミスデプロイに AppSpec</u> ファイルを追加するおよびCodeDeploy のリビジョンを計画するを参照してください。

AppSpec ファイル構造

次に示すのは、 AWS Lambda および EC2/オンプレミスコンピューティングプラットフォームへの デプロイに使用される AppSpec ファイルのハイレベルな構造です。

文字列である YAML 形式の AppSpec ファイルの値は、別途指定されている場合を除き、引用符 ("") で囲んではいけません。

Amazon ECS デプロイ向けの AppSpec ファイル構造

Note

この AppSpec ファイルは YAML で記述されますが、同じ構造を使用して JSON で記述する こともできます。JSON 形式の AppSpec ファイルの文字列は、常に引用符 ("") で囲みます。

```
version: 0.0
resources:
    ecs-service-specifications
hooks:
    deployment-lifecycle-event-mappings
```

この構造の説明

バージョン

このセクションでは、AppSpec ファイルのバージョンを指定します。この値を変更しないで ください。この値は必須です。現在許容されている値は、**0.0**のみです。将来の利用のために CodeDeploy で予約されます。

文字列で version を指定します。

リソース

このセクションでは、デプロイする Amazon ECS アプリケーションに関する情報を指定します。

詳細については、「<u>Amazon ECS デプロイ用の AppSpec の「resources」セクション</u>」を参照 してください。

hooks

このセクションでは、デプロイ検証のために特定のデプロイライフサイクルイベントフックで実 行する、Lambda 関数を指定します。

詳細については、「<u>Amazon ECS のデプロイ向けのライフサイクルイベントフックのリスト</u>」を 参照してください。

AWS Lambda のデプロイ向けの AppSpec ファイル構造

Note

この AppSpec ファイルは、YAML で記述されますが、同じ構造を使用して JSON で Lambda デプロイ用の AppSpec ファイルを記述することもできます。JSON 形式の AppSpec ファイルの文字列は、常に引用符 ("") で囲みます。

```
version: 0.0
resources:
   lambda-function-specifications
hooks:
   deployment-lifecycle-event-mappings
```

この構造の説明

バージョン

このセクションでは、AppSpec ファイルのバージョンを指定します。この値を変更しないで ください。この値は必須です。現在許容されている値は、**0.0**のみです。将来の利用のために CodeDeploy で予約されます。

文字列で version を指定します。

リソース

このセクションでは、デプロイする Lambda 関数に関する情報を指定します。

詳細については、「<u>AppSpec の「リソース」セクション (Amazon ECS と AWS Lambda デプロ</u> イのみ)」を参照してください。

hooks

このセクションでは、デプロイを検証するために特定のデプロイライフサイクルイベントで実行 する Lambda 関数を指定します。

詳細については、「AppSpec の「hooks」セクション」を参照してください。

EC2/オンプレミスデプロイの AppSpec ファイル構造

```
version: 0.0
os: operating-system-name
files:
    source-destination-files-mappings
permissions:
    permissions-specifications
hooks:
    deployment-lifecycle-event-mappings
```

この構造の説明

バージョン

このセクションでは、AppSpec ファイルのバージョンを指定します。この値を変更しないで ください。この値は必須です。現在許容されている値は、**0.0**のみです。将来の利用のために CodeDeploy で予約されます。

文字列で version を指定します。

os

このセクションでは、デプロイ先であるインスタンスのオペレーティングシステムの値を指定し ます。この値は必須です。次の値を指定できます。

- linux— インスタンスは Amazon Linux、Ubuntu Server、RHEL インスタンスです。
- windows— インスタンスは Windows Server インスタンスです。

文字列で os を指定します。

ファイル

このセクションでは、デプロイの Install イベント中に、インスタンスにコピーするファイル名を 指定します。

詳細については、「<u>AppSpec の「ファイル」セクション (EC2/オンプレミスデプロイのみ)</u>」を参 照してください。

アクセス権限

このセクションでは、インスタンスへのコピー中に特別なアクセス権限をファイルの files セ クションに適用する方法を指定します。このセクションは、Amazon Linux、Ubuntu Server、お よび Red Hat Enterprise Linux (RHEL) インスタンスのみに適用されます。

詳細については、「<u>AppSpec の「許可」セクション (EC2/オンプレミスデプロイのみ)</u>」を参照し てください。

hooks

このセクションでは、デプロイ中に特定のデプロイライフサイクルイベントで実行するスクリプトを指定します。

詳細については、「AppSpec の「hooks」セクション」を参照してください。

トピック

AppSpec ファイル構造

- AppSpec の「ファイル」セクション (EC2/オンプレミスデプロイのみ)
- AppSpec の「リソース」セクション (Amazon ECS と AWS Lambda デプロイのみ)
- <u>AppSpec の「許可」セクション (EC2/オンプレミスデプロイのみ)</u>
- AppSpec の「hooks」セクション

AppSpec の「ファイル」セクション (EC2/オンプレミスデプロイのみ)

デプロイの Install イベント中にインスタンスにインストールする、アプリケーションリビジョンの ファイルに関する情報を CodeDeploy に提供します。このセクションは、デプロイ中にリビジョン からインスタンス上の場所にファイルをコピーする場合のみ必要です。

このセクションの構造は次のとおりです。

files:
 - source: source-file-location-1
 destination: destination-file-location-1
file_exists_behavior: DISALLOW/OVERWRITE/RETAIN

複数の source と destination ペアを設定できます。

source は、リビジョンからインスタンスにコピーするファイルまたはディレクトリを識別します。

- source はファイルを参照し、指定されたファイルのみがインスタンスにコピーされます。
- sourceはディレクトリを参照し、そのディレクトリのすべてのファイルがインスタンスにコピー されます。
- source がシングルスラッシュ (Amazon Linux、RHEL、および Ubuntu Server のインスタンスでは "/"、Windows Server のインスタンスでは "\") である場合、リビジョンのすべてのファイルがインスタンスにコピーされます。

source で使用されているパスは appspec.yml ファイルに対する相対パスで、 ファイルはリ ビジョンのルートに存在する必要があります。リビジョンのファイル構造の詳細については、 「<u>CodeDeploy のリビジョンを計画する</u>」を参照してください。

destination は、ファイルをコピーするインスタンス上の場所を識別します。これは、/root/ destination/directory (Linux、RHEL、Ubuntuの場合) または c:\destination\folder (Windows の場合) のような完全修飾パスである必要があります。 source と destination は、それぞれ文字列で指定されます。

file_exists_behavior 命令はオプションで、CodeDeploy がデプロイターゲットの場所にすで に存在し、前回成功したデプロイの一部ではなかったファイルを処理する方法を指定します。この設 定は、以下のいずれかの値を取ることができます。

- DISALLOW: デプロイは失敗です。オプションが何も指定されていないときは、これもデフォルトの動作となります。
- OVERWRITE: 現在デプロイされているアプリケーションリビジョンのファイルのバージョンにより、インスタンスの既存のファイルのバージョンが置き換えられます。
- RETAIN: インスタンスにすでに存在するファイルのバージョンは保持され、新しいデプロイの一部として使用されます。

file_exists_behavior 設定を使用する場合、この設定を理解してください。

- は1度だけ指定でき、files: にリストされたすべてのファイルとディレクトリに適用されます。
- は、 --file-exists-behavior AWS CLI オプションと fileExistsBehavior API オプション (どちらもオプション) よりも優先されます。

以下は、Amazon Linux、Ubuntu Server、または RHEL インスタンスの files セクションの例で す。

files:
 - source: Config/config.txt
 destination: /webapps/Config
 - source: source
 destination: /webapps/myApp

この例では、次の2つのオペレーションが、Install イベント中に実行されます。

- 使用するリビジョンの Config/config.txt ファイルをインスタンスの /webapps/Config/ config.txt パスにコピーします。
- 2. リビジョンの source ディレクトリのすべてのファイルを、インスタンスの /webapps/myApp ディレクトリに再帰的にコピーします。

「files」セクションの例

次の例は、files セクションを指定する方法を示しています。これらの例は、Windows Server ファ イルとディレクトリ (フォルダ) 構造を示していますが、Amazon Linux、Ubuntu Server、および RHEL インスタンスに簡単に適用することができます。

Note

EC2/オンプレミスデプロイのみ、files セクションを使用します。 AWS Lambda デプロイ には適用されません。

次の例では、以下のファイルが source のルートのバンドルに表示されることを前提としていま す。

- appspec.yml
- my-file.txt
- my-file-2.txt
- my-file-3.txt

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
  - source: .\my-file.txt
    destination: c:\temp
#
# Result:
#
    c:\temp\my-file.txt
#
#
 -----
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
  - source: my-file-2.txt
    destination: c:\temp
  - source: my-file-3.txt
    destination: c:\temp
#
# Result:
```

```
#
    c:\temp\my-file-2.txt
    c:\temp\my-file-3.txt
#
#
#
  ------
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the appspec.yml
file) to the destination folder c:\temp.
#
files:
  - source: \
    destination: c:\temp
#
# Result:
   c:\temp\appspec.yml
#
   c:\temp\my-file.txt
#
#
   c:\temp\my-file-2.txt
#
   c:\temp\my-file-3.txt
```

次の例では、appspec.yml が source のルートのバンドルに、3 つのファイルを含む my-folder という名前のフォルダとともに表示されることを前提としています。

- appspec.yml
- my-folder\my-file.txt
- my-folder\my-file-2.txt
- my-folder\my-file-3.txt

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the
destination folder c:\temp.
#
files:
  - source: .\my-folder
   destination: c:\temp
#
# Result:
   c:\temp\my-file.txt
#
   c:\temp\my-file-2.txt
#
   c:\temp\my-file-3.txt
#
#
#
 ------
#
# 5) Copy my-folder and its 3 files to my-folder within the destination folder c:\temp.
```

AWS CodeDeploy

```
#
files:
 - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#
   c:\temp\my-folder\my-file.txt
   c:\temp\my-folder\my-file-2.txt
#
#
   c:\temp\my-folder\my-file-3.txt
#
#
 _____
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder c:
\temp.
#
files:
 - source: .\my-folder
    destination: c:\temp\other-folder
#
# Result:
   c:\temp\other-folder\my-file.txt
#
#
   c:\temp\other-folder\my-file-2.txt
#
   c:\temp\other-folder\my-file-3.txt
#
#
 ------
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destination
folder c:\temp.
#
files:
 - source: .\my-folder\my-file-2.txt
   destination: c:\temp\my-folder
 - source: .\my-folder\my-file-3.txt
    destination: c:\temp\my-folder
#
# Result:
   c:\temp\my-folder\my-file-2.txt
#
#
   c:\temp\my-folder\my-file-3.txt
#
#
  #
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the destination
folder c:\temp.
#
```

```
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\other-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\other-folder
#
# Result:
    c:\temp\other-folder\my-file-2.txt
#
#
    c:\temp\other-folder\my-file-3.txt
#
#
     #
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the
 destination folder c:\temp. If any of the files already exist on the instance,
 overwrite them.
#
files:
  - source: \
    destination: c:\temp
file_exists_behavior: OVERWRITE
#
# Result:
#
   c:\temp\appspec.yml
#
   c:\temp\my-folder\my-file.txt
#
   c:\temp\my-folder\my-file-2.txt
#
    c:\temp\my-folder\my-file-3.txt
```

AppSpec の「リソース」セクション (Amazon ECS と AWS Lambda デプロイのみ)

AppSpec ファイルの 'resources' セクションの内容は、デプロイのコンピューティングプラット フォームによって異なります。Amazon ECS デプロイの 'resources' セクションには、Amazon ECS タスク定義、最新の Amazon ECS タスクセットにトラフィックをルーティングするため のコンテナとポート、およびその他のオプション情報が含まれています。 AWS Lambda デプロ イ'resources'のセクションには、Lambda 関数の名前、エイリアス、現在のバージョン、および ターゲットバージョンが含まれています。

トピック

- AWS Lambda デプロイの AppSpec 'resources' セクション
- Amazon ECS デプロイ用の AppSpec の「resources」セクション

AWS Lambda デプロイの AppSpec 'resources' セクション

'resources' セクションでは、デプロイする Lambda 関数を指定します。その構造は次のとおり です。

YAML:

```
resources:
        - name-of-function-to-deploy:
        type: "AWS::Lambda::Function"
        properties:
            name: name-of-lambda-function-to-deploy
            alias: alias-of-lambda-function-to-deploy
            currentversion: version-of-the-lambda-function-traffic-currently-points-to
            targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

JSON:

各プロパティは文字列で指定します。

- name 必須。これはデプロイする Lambda 関数の名前です。
- alias 必須。これは Lambda 関数のエイリアスの名前です。
- currentversion 必須。これは、トラフィックが現在指している Lambda 関数のバージョンです。値は有効な正の整数である必要があります。
- targetversion 必須。これは、トラフィックの移行先の Lambda 関数のバージョンです。値は有効な正の整数である必要があります。

Amazon ECS デプロイ用の AppSpec の「resources」セクション

' resources' セクションでは、デプロイする Amazon ECS サービスを指定します。その構造は次 のとおりです。

YAML:

```
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-arn"
        LoadBalancerInfo:
          ContainerName: "ecs-container-name"
          ContainerPort: "ecs-application-port"
# Optional properties
        PlatformVersion: "ecs-service-platform-version"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["ecs-subnet-1","ecs-subnet-n"]
            SecurityGroups: ["ecs-security-group-1","ecs-security-group-n"]
            AssignPublicIp: "ENABLED | DISABLED"
        CapacityProviderStrategy:
          - Base: integer
            CapacityProvider: "capacityProviderA"
            Weight: integer
          - Base: integer
            CapacityProvider: "capacityProviderB"
            Weight: integer
```

JSON:



各プロパティは、ContainerPort 以外の文字列で指定され、数字である。

 TaskDefinition – 必須。これはデプロイする Amazon ECS サービスのタスク定義です。タス ク定義の ARN で指定します。ARN 形式は arn:aws:ecs:aws-region:account-id:taskdefinition/task-definition-family:task-definition-revision です。詳細につい ては、「Amazon リソースネーム (ARNsと AWS サービス名前空間」を参照してください。

Note

ARN の:*task-definition-revision*の割り当ては任意です。省略した場合、Amazon ECS はタスク定義の最新の ACTIVE リビジョンを使用します。

- ContainerName 必須。これは、Amazon ECS アプリケーションを含む Amazon ECS コンテナの名前です。Amazon ECS タスク定義で指定されたコンテナにする必要があります。
- ContainerPort 必須。これは、トラフィックのルーティング先となるコンテナ上のポートです。
- PlatformVersion: オプション。デプロイされた Amazon ECS サービス内の、Fargate タスクの プラットフォームのバージョン。詳細については、の「<u>AWS Fargate プラットフォームバージョ</u> ン」を参照してください。指定されない場合、デフォルトで LATEST が使用されます。
- NetworkConfiguration: オプション。AwsvpcConfiguration で、以下を指定する ことができます。詳細については、Amazon ECS Container Service API リファレンスの 「AwsVpcConfiguration」を参照してください。
 - Subnets: オプション。Amazon ECS サービス内の、1 つ以上のサブネットのカンマ区切りリスト。
 - SecurityGroups: オプション。Amazon Elastic Container Service にある、1つ以上のセキュ リティグループのカンマ区切りリスト。
 - AssignPublicIp: オプション。Amazon ECS サービスの Elastic Network Interface がパブリック IP アドレスを受け取るかどうかを指定する文字列。有効な値は ENABLED および DISABLED です。

Note

NetworkConfiguration の下にあるすべての設定を指定するか、何も指定しない 必要があります。たとえば、Subnets を指定する場合は、 SecurityGroups と AssignPublicIp も指定する必要があります。何も指定しない場合、CodeDeploy は現在 のネットワーク Amazon ECS の設定を使用します。

- CapacityProviderStrategy: オプション。デプロイに使用したい Amazon ECS キャパシティ プロバイダーのリスト。詳細については、Amazon Elastic Container Service デベロッパーガイ ドの「<u>Amazon ECS キャパシティープロバイダー</u>」を参照してください。キャパシティプロバイ ダーごとに、次の設定を指定できます。これらの設定の詳細については、AWS CloudFormation ユーザーガイドの「<u>AWS::ECS::ServiceCapacityProviderStrategyItem</u>」を参照してください。
 - Base: オプション。ベース値は、指定されたキャパシティープロバイダーで実行するタスクの最小限の数を指定します。キャパシティープロバイダー戦略では、ベースを定義できるキャパシティープロバイダーは1つだけです。値が指定されていない場合は、デフォルト値の0が使用されます。
 - CapacityProvider: オプション。容量プロバイダーの短い名前。例: capacityProviderA

• Weight: オプション。

ウエイト値は、指定したキャパシティープロバイダーを使用する起動済みタスクの総数に対する 相対的な割合を示します。weight 値は、base 値が、もし定義されている場合、満たされた後 に、考慮されます。

weight 値が指定されていない場合は、デフォルト値の0が使用されます。キャパシティープ ロバイダー戦略内で複数のキャパシティープロバイダーを指定する場合、少なくとも1つの キャパシティープロバイダーのウェイト値が0より大きい必要があり、ウェイトが0のキャパ シティープロバイダーはタスクを配置するのに使用しません。すべてのキャパシティープロバイ ダーが0のウェイトを持つ戦略で複数のキャパシティープロバイダーを指定すると、キャパシ ティープロバイダー戦略を使用する RunTask または CreateService アクションは失敗しま す。

例えば、加重を使用するシナリオが2つのキャパシティープロバイダーを含む戦略を定義し、 両方の加重が1である場合、base が満たされたとき、タスクは2つのキャパシティープロ バイダー間で均等に分割されます。同じロジックを使用して、キャパシティープロバイダー capacityProviderA に1のウエイトを指定し、capacityProviderB に4のウエイトを指定する と、capacityProviderA を使用して実行されるタスクごとに、4つのタスクが capacityProviderB を使用します。

AppSpec の「許可」セクション (EC2/オンプレミスデプロイのみ)

'permissions' セクションでは、インスタンスへのコピー後に、特別なアクセス許可を 'files' セクションのファイルおよびディレクトリ/フォルダに適用する方法を指定します。複数の object 指示を指定できます。このセクションはオプションです。Amazon Linux、Ubuntu Server、RHEL イ ンスタンスにのみ適用されます。

Note

EC2/オンプレミスデプロイにのみ、'permissions' セクションを使用します。 AWS Lambda または Amazon ECS のデプロイには使用されません。

このセクションの構造は次のとおりです。

permissions:

- object: object-specification

pattern: pattern-specification
except: exception-specification
owner: owner-account-name
group: group-name
mode: mode-specification
acls:
 - acls-specification
context:
 user: user-specification
 type: type-specification
 range: range-specification
type:
 - object-type

手順は次のとおりです。

object – 必須。これは、インスタンスへのファイルシステムオブジェクトのコピー後に、指定されたアクセス権限を適用する一連のファイルシステムオブジェクト (ファイルまたはディレクトリ/フォルダ)です。

文字列で object を指定します。

pattern - オプション。アクセス権限を適用するパターンを指定します。指定しない場合、または特殊文字 "**" で指定する場合、type に応じて一致するすべてのファイルまたはディレクトリに、指定されたアクセス権限が適用されます。

引用符 ("") 付きの文字列で pattern を指定します。

except - オプション。pattern の例外とするファイルまたはディレクトリを指定します。

角括弧で囲った文字列のカンマ区切りリストで except を指定します。

 owner - オプション。object の所有者の名前。指定しない場合、既存のすべての所有者が元の ファイルに適用されます。それ以外の場合、ディレクトリ/フォルダ構造は、コピーオペレーショ ンによって変更されません。

文字列で owner を指定します。

 group - オプション。object のグループの名前。指定しない場合、既存のすべてのグループが 元のファイルに適用されます。それ以外の場合、ディレクトリ/フォルダ構造は、コピーオペレー ションによって変更されません。

文字列で group を指定します。

AppSpec ファイル構造
mode - オプション。object に適用されるアクセス権限を指定する数値。モード設定は、Linux の chmod コマンドの構文に従います。

▲ Important

値の先頭にゼロが含まれる場合は、ダブルクォートで囲むか、先頭のゼロを削除して3桁 だけにする必要があります。

Note

u+x の設定では、mode のような記号表記はサポートされていません。

例:

- mode: "0644"は、オブジェクトの所有者に読み書きのアクセス権限(6)、グループに対する読み取り専用アクセス許可(4)、およびその他すべてのユーザーに読み取り専用アクセス権限(4) を与えます。
- mode: 644 と同じ権限を付与する mode: "0644"。
- mode: 4755 は setuid 属性を設定し (4)、所有者にフルコントロール権限を与え (7)、グループ に対する読み取りと実行の権限を付与し (5)、他のすべてのユーザーに読み取りと実行の権限を 付与します (5)。

その他の例については、Linux の chmod コマンドのドキュメントを参照してください。

モードを指定しない場合、既存のすべてのモードが元のファイルに適用されます。それ以外の場 合、フォルダ構造は、コピーオペレーションによって変更されません。

acls - オプション。1 つ以上のアクセスコントロールリスト (ACL) エントリを表す文字列のリストが、object に適用されます。たとえば、u:bob:rwは、ユーザー bob の読み取りおよび書き込みアクセス権限を表します(その他の例については、Linux の setfacl コマンドドキュメントのACL入力形式の例を参照してください)。複数のACL エントリを指定できます。acls を指定しない場合、既存のすべてのACLが元のファイルに適用されます。それ以外の場合、ディレクトリ/フォルダ構造は、コピーオペレーションによって変更されません。既存のACL は置き換えられます。

acls を指定します。ダッシュ (-) の後にスペースを続け、その後に文字列を続けます(例:u:jane:rw)。ACL が複数ある場合は、それぞれ個別の行で指定します。 Note

名前のないユーザー、名前のないグループ、またはその他の同様の ACL エントリを設定す ると、AppSpec ファイルは失敗します。これらのタイプのアクセス許可を指定するには、 代わりに mode を使用します。

- context オプション。Security-Enhanced Linux (SELinux) 対応インスタンスの場合、コピーし たオブジェクトに適用されるセキュリティ関連コンテキストラベルのリスト。ラベルは、user, type、および range を含むキーとして指定されます。(詳細については、SELinux のドキュメン トを参照してください)。各キーは文字列で入力します。指定しない場合、既存のすべてのラベ ルが元のファイルに適用されます。それ以外の場合、ディレクトリ/フォルダ構造は、コピーオペ レーションによって変更されません。
 - ・ user オプション。SELinux ユーザー。
 - type オプション。SELinux の型名。
 - range オプション。SELinux の範囲指定子。マルチレベルセキュリティ (MLS) およびマルチ カテゴリセキュリティ (MCS) がマシンで有効になっていない限り、この効果はありません。有 効になっていない場合は、range はデフォルトで s0 になります。

文字列 で context を指定します(例: user: unconfined_u)。それぞれの context は個別の行で指定されます。

type - オプション。指定された権限を適用するオブジェクトのタイプ。type は、file あるいは directory に設定できる文字列です。file を指定した場合、アクセス許可は、コピーオペレーションの後に(object 自体ではなく) object 内に直接含まれるファイルのみに適用されます。directory を指定した場合、アクセス権限は、コピーオペレーションの後に(object 自体ではなく)、object 内のいずれかの場所にあるすべてのディレクトリ/フォルダに再帰的に適用されます。

type を指定します。ダッシュ (-) の後にスペースを続け、その後に文字列を続けます(例: file)。

「permissions」セクションの例

次の例は、object、pattern、except、owner、mode、および type の手順を使用して 'permissions' セクションを指定する方法を示しています。この例は、Amazon Linux、Ubuntu Server、RHEL インスタンスにのみ適用されます。この例では、次のファイルとフォルダが、この階 層のインスタンスにコピーされることを前提としています。

/tmp
` my-app
my-file-1.txt
my-file-2.txt
my-file-3.txt
my-folder-1
my-file-4.txt
my-file-5.txt
` my-file-6.txt
` my-folder-2
my-file-7.txt
my-file-8.txt
my-file-9.txt
` my-folder-3

次の AppSpec ファイルは、コピー後にこれらのファイルとフォルダでアクセス権限を設定する方法 を示しています。

```
version: 0.0
os: linux
# Copy over all of the folders and files with the permissions they
# were originally assigned.
files:
  - source: ./my-file-1.txt
    destination: /tmp/my-app
  - source: ./my-file-2.txt
    destination: /tmp/my-app
  - source: ./my-file-3.txt
   destination: /tmp/my-app
  - source: ./my-folder-1
    destination: /tmp/my-app/my-folder-1
  - source: ./my-folder-2
    destination: /tmp/my-app/my-folder-2
# 1) For all of the files in the /tmp/my-app folder ending in -3.txt
# (for example, just my-file-3.txt), owner = adm, group = wheel, and
# mode = 464 (-r--rw-r--).
permissions:
  - object: /tmp/my-app
    pattern: "*-3.txt"
    owner: adm
    group: wheel
    mode: 464
    type:
```

ユーザーガイド

```
- file
# 2) For all of the files ending in .txt in the /tmp/my-app
 folder, but not for the file my-file-3.txt (for example,
#
# just my-file-1.txt and my-file-2.txt),
# owner = ec2-user and mode = 444 (-r--r--).
  - object: /tmp/my-app
    pattern: "*.txt"
    except: [my-file-3.txt]
    owner: ec2-user
   mode: 444
    type:
      - file
# 3) For all the files in the /tmp/my-app/my-folder-1 folder except
# for my-file-4.txt and my-file-5.txt, (for example,
# just my-file-6.txt), owner = operator and mode = 646 (-rw-r--rw-).
  - object: /tmp/my-app/my-folder-1
    pattern: "**"
    except: [my-file-4.txt, my-file-5.txt]
    owner: operator
   mode: 646
    type:
      - file
# 4) For all of the files that are immediately under
# the /tmp/my-app/my-folder-2 folder except for my-file-8.txt,
# (for example, just my-file-7.txt and
# my-file-9.txt), owner = ec2-user and mode = 777 (-rwxrwxrwx).
  - object: /tmp/my-app/my-folder-2
    pattern: "**"
    except: [my-file-8.txt]
    owner: ec2-user
   mode: 777
    type:
      - file
# 5) For all folders at any level under /tmp/my-app that contain
 the name my-folder but not
#
 /tmp/my-app/my-folder-2/my-folder-3 (for example, just
#
 /tmp/my-app/my-folder-1 and /tmp/my-app/my-folder-2),
#
# owner = ec2-user and mode = 555 (dr-xr-xr-x).
  - object: /tmp/my-app
    pattern: "*my-folder*"
    except: [tmp/my-app/my-folder-2/my-folder-3]
    owner: ec2-user
    mode: 555
    type:
```

作成されるアクセス権限は次のとおりです。

```
-r--r-- ec2-user root my-file-1.txt
-r--r-- ec2-user root my-file-2.txt
                  wheel my-file-3.txt
-r--rw-r-- adm
dr-xr-xr-x ec2-user root my-folder-1
-rw-r--r-- root root my-file-4.txt
                 root my-file-5.txt
-rw-r--r-- root
-rw-r--rw- operator root my-file-6.txt
dr-xr-xr-x ec2-user root my-folder-2
-rwxrwxrwx ec2-user root my-file-7.txt
-rw-r--r-- root
                  root my-file-8.txt
-rwxrwxrwx ec2-user root my-file-9.txt
                  wheel my-folder-3
dr-xrw-r-- root
```

次の例では、「acls」および「context」の手順を追加して、'permissions' セクションを指定 する方法を示します。この例は、Amazon Linux、Ubuntu Server、RHEL インスタンスにのみ適用さ れます。

```
permissions:
    object: /var/www/html/WordPress
    pattern: "**"
    except: [/var/www/html/WordPress/ReadMe.txt]
    owner: bob
    group: writers
    mode: 644
    acls:
        - u:mary:rw
        - u:sam:rw
        - m::rw
        context:
```

AppSpec の「hooks」セクション

AppSpec ファイルの 'hooks' セクションの内容は、デプロイのコンピューティングプラット フォームによって異なります。EC2/オンプレミスのデプロイの 'hooks' セクションには、デプ ロイライフサイクルイベントフックを 1 つ以上のスクリプトにリンクするマッピングが含まれま す。Lambda または Amazon ECS のデプロイの 'hooks' セクションは、デプロイライフサイクル イベント中に実行する Lambda 検証の関数を指定します。イベントフックが存在しない場合、その イベントに対してオペレーションは実行されません。このセクションは、デプロイの一部としてスク リプトまたは Lambda 検証の関数を実行する場合のみ必須です。

トピック

- <u>Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション</u>
- <u>AWS の Lambda デプロイ向けの AppSpec の「hooks」セクション</u>
- ・ EC2/オンプレミスのデプロイ向けの AppSpec の「hooks」セクション

Amazon ECS のデプロイ向けの AppSpec の「hooks」セクション

トピック

- Amazon ECS のデプロイ向けのライフサイクルイベントフックのリスト
- <u>Amazon ECS デプロイでフックの順序を実行します。</u>
- 「hooks」 セクションの構造
- Lambda の「フック」関数のサンプル

Amazon ECS のデプロイ向けのライフサイクルイベントフックのリスト

AWS Lambda フックは、ライフサイクルイベントの名前の後に新しい行に文字列で指定された1つの Lambda 関数です。各フックはデプロイごとに1回実行されます。以下は、Amazon ECS デプロイートにフックを実行できるライフサイクルイベントの説明です。

BeforeInstall 置き換えタスクセットが作成される前にタスクを実行するために使用します。1
 つのターゲットグループが元のタスクセットに関連付けられています。オプションのテストリス

ナーが指定されている場合、それは元のタスクセットに関連付けられます。この時点で、ロール バックはできません。

- AfterInstall 置き換えタスクセットが作成され、ターゲットグループの1つがそれに関連付け られた後、タスクを実行するために使用します。オプションのテストリスナーが指定されている場 合、それは元のタスクセットに関連付けられます。このライフサイクルイベントでのフック関数の 結果により、ロールバックをトリガーできます。
- AfterAllowTestTraffic テストリスナーが置き換えタスクセットにトラフィックを提供した 後、タスクを実行するために使用します。この時点でのフック関数の結果により、ロールバックを トリガーできます。
- BeforeAllowTraffic 2 番目のターゲットグループが置き換えタスクセットに関連付けられた 後、かつ、トラフィックが置き換えタスクセットに移行される前に、タスクを実行するために使用 します。このライフサイクルイベントでのフック関数の結果により、ロールバックをトリガーでき ます。
- AfterAllowTraffic 2 番目のターゲットグループが置き換えタスクセットにトラフィックを提供した後、タスクを実行するために使用します。このライフサイクルイベントでのフック関数の結果により、ロールバックをトリガーできます。

詳細については、<u>Amazon ECS デプロイ中の処理で起こっていること</u>および<u>チュートリアル: 検証テ</u> ストを使用して Amazon ECS サービスをデプロイ<mark>する</mark>を参照してください。

Amazon ECS デプロイでフックの順序を実行します。

Amazon ECS デプロイでは、イベントフックは次の順序で実行されます。



Note

デプロイ中の Start、Install、TestTraffic、AllowTraffic、および End イベントはスクリプト化 できないため、この図ではグレーで表示されています。

「hooks」 セクションの構造

次の例は、'hooks' セクションの構造の例を示します。

YAML の使用:

Hooks:

- BeforeInstall: "BeforeInstallHookFunctionName"
- AfterInstall: "AfterInstallHookFunctionName"
- AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
- BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
- AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"

JSON の使用:

```
"Hooks": [
  {
   "BeforeInstall": "BeforeInstallHookFunctionName"
  },
  {
   "AfterInstall": "AfterInstallHookFunctionName"
  },
  {
   "AfterAllowTestTraffic": "AfterAllowTestTrafficHookFunctionName"
  },
  {
   "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
  },
  {
   "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
  }
 ]
}
```

Lambda の「フック」関数のサンプル

'hooks' セクションを使用して、Amazon ECS のデプロイを検証する ために CodeDeploy が呼び出すことができる Lambda 関数を指定しま す。、BeforeInstall、、AfterInstallAfterAllowTestTraffic、および AfterAllowTrafficデプロイライフサイクルイベントにはBeforeAllowTraffic、同じ関数ま たは別の関数を使用できます。検証テストが完了すると、Lambda AfterAllowTraffic 関数は CodeDeploy を呼び戻し、Succeeded または Failed の結果を配信します。

A Important

1 時間以内に CodeDeploy が Lambda 検証関数から通知されない場合、デプロイは失敗した と見なされます。

Lambda フック関数を呼び出す前に、サーバーは putLifecycleEventHookExecutionStatus コマンドを使用して、デプロイ ID およびライフサイクルイベントフック実行 ID について通知され る必要があります。

次に示すのは、Node.js で記述されたサンプルの Lambda フック関数の例です。

```
'use strict';
const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});
exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;
    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
    /*
    Enter validation tests here.
    */
    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
```

```
lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
};
// Pass CodeDeploy the prepared validation test results.
codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
    if (err) {
        // Validation failed.
        callback('Validation test failed');
    } else {
        // Validation succeeded.
        callback(null, 'Validation test succeeded');
    }
};
```

AWS の Lambda デプロイ向けの AppSpec の「hooks」セクション

トピック

- AWS Lambda デプロイのライフサイクルイベントフックのリスト
- ・ Lambda 関数のバージョンのデプロイでのフックの実行順
- 「hooks」 セクションの構造
- ・ Lambda の「フック」関数のサンプル

AWS Lambda デプロイのライフサイクルイベントフックのリスト

AWS Lambda フックは、ライフサイクルイベントの名前の後に新しい行に文字列で指定された1つの Lambda 関数です。各フックはデプロイごとに1回実行されます。以下は、AppSpec ファイルに使用できるフックの説明です。

- BeforeAllowTraffic これを使用して、トラフィックがデプロイされた Lambda 関数のバージョン に移行する前にタスクを実行します。
- AfterAllowTraffic これを使用して、トラフィックがデプロイされた Lambda 関数のバージョンに 移行した後でタスクを実行します。

Lambda 関数のバージョンのデプロイでのフックの実行順

サーバーレスの Lambda 関数のバージョンのデプロイでは、イベントフックは次の順序で実行され ます。

AppSpec ファイル構造



Note

デプロイの Start、AllowTraffic、および End イベントはスクリプト化できません。このため、これらのイベントはこの図でグレーで表示されています。

「hooks」 セクションの構造

次の例は、「hooks」セクションの例を示します。

YAML の使用:

hooks:

- BeforeAllowTraffic: BeforeAllowTrafficHookFunctionName
- AfterAllowTraffic: AfterAllowTrafficHookFunctionName

JSON の使用:

```
"hooks": [{
    "BeforeAllowTrafficHookFunctionName"
    },
    {
```

}]

"AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"

Lambda の「フック」関数のサンプル

「hooks」セクションを使用して、Lambda のデプロイを検証するために CodeDeploy が呼び出すこ とができる Lambda 関数を指定します。BeforeAllowTraffic および AfterAllowTrafficデプ ロイライフサイクルイベントには、同じ関数または別の関数を使用できます。検証テストが完了す ると、Lambda 検証関数は CodeDeploy を呼び戻し、Succeeded または Failed の結果を配信しま す。

🛕 Important

1 時間以内に CodeDeploy が Lambda 検証関数から通知されない場合、デプロイは失敗した と見なされます。

Lambda フック関数を呼び出す前に、サーバーは putLifecycleEventHookExecutionStatus コマンドを使用して、デプロイ ID およびライフサイクルイベントフック実行 ID について通知され る必要があります。

次に示すのは、Node.js で記述されたサンプルの Lambda フック関数の例です。

```
'use strict';
const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});
exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;
    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
    /*
    Enter validation tests here.
    */
    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
```

```
var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };
    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

EC2/オンプレミスのデプロイ向けの AppSpec の「hooks」セクション

トピック

- ライフサイクルイベントフックのリスト
- ライフサイクルイベントフックの可用性
- デプロイでのフックの実行順
- 「hooks」 セクションの構造
- フックスクリプトでのファイルの参照
- フックの環境変数の可用性
- hooks の例

ライフサイクルイベントフックのリスト

EC2/オンプレミスのデプロイのフックは、デプロイごとに1回インスタンスに対して実行されま す。フックには実行するスクリプトを1つまたは複数指定することができます。ライフサイクルイ ベントの各フックは、文字列で個別の行に指定します。以下は、AppSpec ファイルに使用できる フックの説明です。

デプロイおよびロールバックの種類別の有効なライフサイクルフックの詳細については、「<u>ライフサ</u> イクルイベントフックの可用性」を参照してください。 ApplicationStop このデプロイライフサイクルイベントは、アプリケーションリビジョンがダウンロードされる前でも発生します。アプリケーションを適切に中止するか、現在インストールされているパッケージを削除してデプロイの準備をする場合は、このイベントのスクリプトを指定できます。このデプロイライフサイクルイベントに使用される AppSpec ファイルとスクリプトは、前回正常にデプロイされたアプリケーションリビジョンのものです。

Note

AppSpec ファイルは、デプロイする前にはインスタンスに存在しません。したがっ て、ApplicationStop フックは、初めてインスタンスにデプロイするときは実行されま せん。インスタンスに 2 回目にデプロイするときは、ApplicationStop フックを使用で きます。

最後に正常にデプロイされたアプリケーションリビジョンの場所を特定するため、CodeDeploy エージェントは *deployment-group-id*_last_successful_install ファイルにリストされ た場所を探します。このファイルは次の場所にあります。

Amazon Linux、Ubuntu Server、RHEL Amazon EC2 インスタンスの /opt/codedeployagent/deployment-root/deployment-instructions フォルダ。

Windows Server の Amazon EC2 インスタンスの C:\ProgramData\Amazon\CodeDeploy \deployment-instructions フォルダ。

ApplicationStop デプロイライフサイクルイベント中に失敗するデプロイをトラブルシュー ティングするには、「<u>失敗した ApplicationStop、BeforeBlockTraffic、または AfterBlockTraffic デ</u> <u>プロイライフサイクルイベントのトラブルシューティング</u>」を参照してください。

 DownloadBundle このデプロイライフサイクルイベント中に、CodeDeploy エージェントはアプ リケーションリビジョンファイルを一時的な場所にコピーします。

Amazon Linux、Ubuntu Server、RHEL Amazon EC2 インスタンスの /opt/codedeployagent/deployment-root/*deployment-group-id/deployment-id*/deploymentarchive フォルダ。

Windows Server の Amazon EC2 インスタンスの C:\ProgramData\Amazon\CodeDeploy \
deployment-group-id\deployment-id\deployment-archive フォルダ。

このイベントは CodeDeploy エージェント用に予約されていて、スクリプトを実行するために使用することはできません。

DownloadBundle デプロイライフサイクルイベント中に失敗するデプロイをトラブルシューティ ングするには、「<u>「不明なエラー: 読み取り用に開いていません」で失敗した DownloadBundle デ</u> プロイライフサイクルイベントのトラブルシューティング」を参照してください。

- BeforeInstall このデプロイライフサイクルイベントは、ファイルの復号や現在のバージョンのバックアップの作成などの事前インストールタスクに使用できます。
- Install このデプロイのライフサイクルイベントでは、CodeDeploy エージェントが一時的なロ ケーションからリビジョンファイルを最終的な送信先フォルダにコピーします。このイベントは CodeDeploy エージェント用に予約されていて、スクリプトを実行するために使用することはでき ません。
- AfterInstall アプリケーションの設定やファイルのアクセス許可の変更などのタスクに、この デプロイライフサイクルイベントを使用できます。
- ApplicationStart 通常、このデプロイライフサイクルイベントを使用して、ApplicationStop 中に停止されたサービスを再起動します。
- ValidateService これが最後のデプロイライフサイクルイベントです。デプロイが正常に完了 したことを確認するために使用されます。
- BeforeBlockTraffic このデプロイライフサイクルイベントを使用して、ロードバランサーから登録解除される前のインスタンスでタスクを実行できます。

BeforeBlockTraffic デプロイライフサイクルイベント中に失敗するデプロイをトラ ブルシューティングするには、「<u>失敗した ApplicationStop、BeforeBlockTraffic、または</u> <u>AfterBlockTraffic デプロイライフサイクルイベントのトラブルシューティング</u>」を参照してくださ い。

- BlockTraffic このデプロイライフサイクルイベント中は、現在トラフィックの処理中であるインスタンスに対するインターネットトラフィックのアクセスがブロックされます。このイベントは CodeDeploy エージェント用に予約されていて、スクリプトを実行するために使用することはできません。
- AfterBlockTraffic このデプロイライフサイクルイベントを使用して、それぞれのロードバランサーから登録解除された後のインスタンスでタスクを実行できます。

AfterBlockTraffic デプロイライフサイクルイベント中に失敗するデプロイをトラブルシュー ティングするには、「<u>失敗した ApplicationStop、BeforeBlockTraffic、または AfterBlockTraffic デ</u> <u>プロイライフサイクルイベントのトラブルシューティング</u>」を参照してください。

- BeforeAllowTraffic このデプロイライフサイクルイベントを使用して、ロードバランサーに 登録される前のインスタンスでタスクを実行できます。
- AllowTraffic このデプロイライフサイクルイベント中は、デプロイ後のインスタンスに対する インターネットトラフィックのアクセスが許可されます。このイベントは CodeDeploy エージェ ント用に予約されていて、スクリプトを実行するために使用することはできません。
- AfterAllowTraffic このデプロイライフサイクルイベントを使用して、ロードバランサーに登録された後のインスタンスでタスクを実行できます。

ライフサイクルイベントフックの可用性

次の表に、各デプロイおよびロールバックシナリオで使用できるライフサイクルイベントフックを示 します。

ライフサ イクルイ ベント名	Auto Scaling 起動デプ ロイ ¹	Auto Scaling 終了デプ ロイ ¹	インプ レースデ プロイ¹	Blue/ Green デ プロイ: 元のイン スタンス	Blue/ Green デ プロイ: 代替イン スタンス	Blue/ Green デ プロイの ロールバ ック: 元 のインス タンス	Blue/ Green デ プロイの ロールバ ック: 代 替インス タンス
Applicati onStop	\checkmark	\checkmark	\checkmark		\checkmark		
DownloadB undle³	\checkmark		\checkmark		\checkmark		
BeforeIns tall	\checkmark		\checkmark		\checkmark		
Install ³	\checkmark		\checkmark		\checkmark		
AfterInst all	\checkmark		\checkmark		\checkmark		
Applicati onStart	\checkmark		\checkmark		\checkmark		

AWS CodeDeploy

ライフサ イクルイ ベント名	Auto Scaling 起動デプ ロイ ¹	Auto Scaling 終了デプ ロイ ¹	インプ レースデ プロイ¹	Blue/ Green デ プロイ: 元のイン スタンス	Blue/ Green デ プロイ: 代替イン スタンス	Blue/ Green デ プロイの ロールバ ック: 元 のインス タンス	Blue/ Green デ プロイの ロールバ ック: 代 替インス タンス
ValidateS ervice	\checkmark		\checkmark		\checkmark		
BeforeBlo ckTraffic		\checkmark	\checkmark	\checkmark			\checkmark
BlockTraf fic³		\checkmark	\checkmark	\checkmark			\checkmark
AfterBloc kTraffic		\checkmark	\checkmark	\checkmark			\checkmark
BeforeAll owTraffic	\checkmark		\checkmark		\checkmark	\checkmark	
AllowTraf fic³	\checkmark		\checkmark		\checkmark	\checkmark	
AfterAllo wTraffic	\checkmark		\checkmark		\checkmark	\checkmark	

¹ Amazon EC2 Auto Scaling デプロイの詳細については、「<u>Amazon EC2 Auto Scaling と</u> <u>CodeDeploy の連携</u>」を参照してください。

²インプレースデプロイのロールバックにも適用されます。

³ CodeDeploy オペレーション用に予約されています。スクリプトの実行には使用できません。

デプロイでのフックの実行順

Auto Scaling 起動デプロイ

Auto Scaling 起動デプロイ中に、CodeDeploy は次の順序でイベントフックを実行します。

Auto Scaling 起動デプロイの詳細については、「<u>Amazon EC2 Auto Scaling と CodeDeploy の連携</u>」 を参照してください。



Note

デプロイの Start、DownloadBundle、Install、AllowTraffic、End の各イベントはスクリプ ト化できないため、この図ではグレーで表示されています。ただし、AppSpec ファイルの 'files' セクションを編集して、Install イベント中にインストールされるものを指定できます。

Auto Scaling 終了デプロイ

Auto Scaling 終了デプロイ中に、CodeDeploy は次の順序でイベントフックを実行します。

Auto Scaling 終了デプロイの詳細については、「<u>Auto Scaling スケールインイベント中の終了デプロ</u> <u>イの有効化</u>」を参照してください。



Note

デプロイの Start、BlockTraffic、End の各イベントはスクリプト化できないため、この図で はグレーで表示されています。

インプレースデプロイ

インプレースデプロイのロールバックを含むインプレースデプロイで、イベントフックは次の順序で 実行されます。

Note

インプレースデプロイの場合、トラフィックのブロックと許可に関する 6 つのフックは、 デプロイグループに Elastic Load Balancing から Classic Load Balancer、Application Load Balancer、または Network Load Balancer を指定した場合のみ適用されます。





Note

デプロイ中の Start、DownloadBundle、Install、および End イベントはスクリプト化できな いため、この図ではグレーで表示されています。ただし、AppSpec ファイルの 'files' セ クションを編集して、Install イベント中にインストールされるものを指定できます。

Blue/Green デプロイ

Blue/Green デプロイでは、イベントフックは次の順序で実行されます。



Note

デプロイ中の Start、DownloadBundle、Install、BlockTraffic、AllowTraffic、および End イベントはスクリプト化できないため、この図ではグレーで表示されています。ただ し、AppSpec ファイルの「files」セクションを編集して、Install イベント中にインストール されるものを指定できます。

「hooks」 セクションの構造

'hooks' セクションは以下の構造を持ちます。

hooks: deployment-lifecycle-event-name: - location: script-location timeout: timeout-in-seconds runas: user-name

デプロイライフサイクルイベント名の後で、次の要素を hook エントリに含めることができます。

location

必須。リビジョンのスクリプトファイルのバンドルでの位置。hooks セクションで指定するスク リプトの場所は、アプリケーションリビジョンバンドルのルートから相対的な位置です。詳細に ついては、「<u>CodeDeploy のリビジョンを計画する</u>」を参照してください。

timeout

オプション。失敗と見なされる前にスクリプトの実行を許可する秒数。デフォルト値は 3600 秒 (1 時間) です。

Note

3600 秒 (1 時間) は、各デプロイライフサイクルイベントのスクリプト実行で許可される 最大の時間です。スクリプトがこの制限を超過した場合、デプロイは停止し、インスタン スへのデプロイは失敗します。各デプロイライフサイクルイベントのすべてのスクリプト で、timeout に指定された合計秒数が、この制限を超えないようにします。

runas

オプション。スクリプトの実行時に偽装するユーザー。デフォルトでは、インスタンス上で実行 されている CodeDeploy エージェントです。CodeDeploy はパスワードを保存しないため、runas ユーザーがパスワードを必要とする場合、ユーザーになりすますことはできません。この要素 は、Amazon Linux、Ubuntu Server、RHEL インスタンスにのみ適用されます。

フックスクリプトでのファイルの参照

<u>AppSpec の「hooks」セクション</u>で説明しているように、スクリプトを CodeDeploy ライフサイク ルイベントにフックし、スクリプト内のファイル (helper.sh など) を参照する場合は、以下を使用 して helper.sh を指定する必要があります。

- ・(推奨)絶対パス。「絶対パスの使用」を参照してください。
- 相対パス。「相対パスの使用」を参照してください。

絶対パスの使用

絶対パスを使用してファイルを参照するには、次のいずれかを行うことができます。

- AppSpec ファイルの files セクションの destination プロパティに絶対パスを指定します。
 次に、フックスクリプトに同じ絶対パスを指定します。詳細については、「<u>AppSpec の「ファイ</u>ル」セクション (EC2/オンプレミスデプロイのみ)」を参照してください。
- フックスクリプトに動的絶対パスを指定します。詳細については、「デプロイアーカイブの場所」
 を参照してください。

デプロイアーカイブの場所

<u>DownloadBundle</u> ライフサイクルイベント中に、CodeDeploy エージェントはデプロイ用の<u>リビジョ</u> ンを次の形式のディレクトリに抽出します。

root-directory/deployment-group-id/deployment-id/deployment-archive

パスの *root-directory* 部分は、常に次の表に示すデフォルトに設定されるか、:root_dir 構成 設定によって制御されます。構成設定の詳細については、「<u>CodeDeploy エージェント設定リファレ</u> ンス」を参照してください。

エージェントプラットフォー ム	デフォルトのルートディレク トリ	
Linux — すべての rpm ディス トリビューション	<pre>/opt/codedeploy-ag ent/deployment-root</pre>	
Ubuntu サーバー — すべての deb ディストリビューション	<pre>/opt/codedeploy-ag ent/deployment-root</pre>	
Windows Server	%ProgramData%\Amazon \CodeDeploy	

フックスクリプトから、ルートディレクトリパスおよび環境変数 (DEPLOYMENT_ID と DEPLOYMENT_GROUP_ID) を使用して現在のデプロイアーカイブにアクセスできます。使用できる 変数の詳細については、「<u>フックの環境変数の可用性</u>」を参照してください。

例えば、Linux のリビジョンのルートにある data.json ファイルにアクセスする方法は次のとおり です。

#!/bin/bash

```
rootDirectory="/opt/codedeploy-agent/deployment-root" # note: this will be different if
you
```

```
configuration
dataFile="$rootDirectory/$DEPLOYMENT_GROUP_ID/$DEPLOYMENT_ID/deployment-archive/
data.json"
data=$(cat dataFile)
```

別の例として、Windows の Powershell を使用してリビジョンのルートにある data . j son ファイル にアクセスする方法は次のとおりです。

customize the :root_dir

相対パスの使用

相対パスを使用してファイルを参照するには、CodeDeploy エージェントの作業ディレクトリを知っ ている必要があります。ファイルパスは、このディレクトリからの相対パスです。

次の表は、CodeDeploy エージェントのサポートされているプラットフォームごとの作業ディレクト リを示しています。

エージェントプラットフォー ム	プロセス管理メソッド	ライフサイクルイベントスク リプトの作業ディレクトリ
Linux — すべての rpm ディス	systemd (デフォルト)	/
F9E1-999	init.d — <u>詳細はこちら</u>	/opt/codedeploy-ag ent
Ubuntu サーバー — すべての debian ディストリビューショ ン	すべて	/opt/codedeploy-ag ent
Windows Server	該当なし	C:\Windows\System32

フックの環境変数の可用性

各デプロイライフサイクルイベントの間、フックスクリプトは次の環境変数にアクセスできます。

APPLICATION_NAME

現在のデプロイの一部である CodeDeploy のアプリケーションの名前 (WordPress_App など)。 DEPLOYMENT_ID

CodeDeploy が、現在のデプロイに割り当てられた ID (例えば、d-AB1CDEF23)。

DEPLOYMENT_GROUP_NAME

現在のデプロイの一部である CodeDeploy のデプロイグループの名前 (WordPress_DepGroup など)。

DEPLOYMENT_GROUP_ID

現在のデプロイの一部である CodeDeploy のデプロイグループの ID (b1a2189bdd90-4ef5-8f40-4c1c5EXAMPLE など)。

LIFECYCLE_EVENT

現在のデプロイライフサイクルイベントの名前 (例: AfterInstall)。

これらの環境変数は各デプロイライフサイクルイベントにローカルです。

デプロイバンドルのソースに応じて、スクリプトをフックできる環境変数が他にもあります。

Amazon S3 からのバンドル

BUNDLE_BUCKET

デプロイバンドルがダウンロードされた Amazon S3 バケットの名前 (例: my-s3-bucket)。

BUNDLE_KEY

Amazon S3 バケット内のダウンロードされたバンドル用のオブジェクトキー (例: WordPress_App.zip)。

• BUNDLE_VERSION

バンドル用のオブジェクトバージョン (例: 3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY +MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo)。この変数は Amazon S3 バケットで<u>オブジェクトバー</u> ジョニングが有効になっている場合にのみ設定されます。

• BUNDLE_ETAG

バンドル用のオブジェクト Etag (例: b10a8db164e0754105b7a99be72e3fe5-4)。

GitHub からのバンドル

BUNDLE_COMMIT

Git によって生成されたバンドルの SHA256 コミットハッシュ (例: d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26)。

以下のスクリプトは、DEPLOYMENT_GROUP_NAME の値が Staging と等しい場合に、Apache HTTP サーバーでリッスンするポートを 80 ではなく 9090 に変更します。このスクリプトは BeforeInstall デプロイライフサイクルイベント中に呼び出される必要があります。

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
```

```
sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

次のスクリプトの例では、DEPLOYMENT_GROUP_NAME 環境変数の値が Staging に等しい場合 に、エラーログに記録されるメッセージの詳細レベルを警告からデバッグに変更します。このスクリ プトは BeforeInstall デプロイライフサイクルイベント中に呼び出される必要があります。

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

以下のスクリプトの例では、指定されたウェブページを、これらの環境変数の値を表示するテキスト で置き換えます。このスクリプトは AfterInstall デプロイライフサイクルイベント中に呼び出さ れる必要があります。

```
#!/usr/bin/python
import os
strToSearch="<h2>This application was deployed using CodeDeploy.</h2>"
strToReplace="<h2>This page for "+os.environ['APPLICATION_NAME']+"
application and "+os.environ['DEPLOYMENT_GROUP_NAME']+" deployment group with
"+os.environ['DEPLOYMENT_GROUP_ID']+" deployment group ID was generated by a
"+os.environ['LIFECYCLE_EVENT']+" script during "+os.environ['DEPLOYMENT_ID']+"
deployment.</h2>"
fp=open("/var/www/html/index.html","r")
buffer=fp.read()
fp.close()
fp.write(buffer.replace(strToSearch,strToReplace))
fp.close()
```

hooks の例

hooks エントリの例を次に示します。AfterInstall ライフサイクルイベントに 2 つのフックを指 定しています。

hooks:

AppSpec ファイル構造

```
AfterInstall:
```

- location: Scripts/RunResourceTests.sh timeout: 180
- location: Scripts/PostDeploy.sh timeout: 180

デプロイプロセスの AfterInstall ステージ中に、Scripts/RunResourceTests.sh スクリプ トが実行されます。スクリプトの実行に 180 秒 (3 分) 以上かかる場合、デプロイは成功しません。

「hooks」セクションで指定するスクリプトの場所は、アプリケーションリビジョンバンドルの ルートに相対的な位置です。前述の例では、RunResourceTests.sh という名前のファイルが Scripts という名前のディレクトリにあります。Scripts ディレクトリはバンドルのルートレベル にあります。詳細については、「CodeDeploy のリビジョンを計画する」を参照してください。

AppSpec ファイルの例

このトピックでは、 AWS Lambda および EC2/オンプレミスデプロイ用の AppSpec ファイルの例を 示します。

トピック

- <u>Amazon ECS デプロイの AppSpec ファイルの例</u>
- ・ AWS Lambda デプロイの AppSpec ファイルの例
- ・ EC2/オンプレミスデプロイの AppSpec ファイルの例

Amazon ECS デプロイの AppSpec ファイルの例

Amazon ECS サービスをデプロイするために YAML で書かれた AppSpec ファイルの例を示しま す。

```
version: 0.0
Resources:
    - TargetService:
    Type: AWS::ECS::Service
    Properties:
    TaskDefinition: "arn:aws:ecs:us-east-1:111222333444:task-definition/my-task-
definition-family-name:1"
    LoadBalancerInfo:
        ContainerName: "SampleApplicationName"
        ContainerPort: 80
# Optional properties
```

```
PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["subnet-1234abcd", "subnet-5678abcd"]
            SecurityGroups: ["sg-12345678"]
            AssignPublicIp: "ENABLED"
        CapacityProviderStrategy:
          - Base: 1
            CapacityProvider: "FARGATE_SPOT"
            Weight: 2
          - Base: 0
            CapacityProvider: "FARGATE"
            Weight: 1
Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "LambdaFunctionToValidateAfterTestTrafficStarts"
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
```

- AfterAllowTraffic: "LambdaFunctionToValidateAfterAllowingProductionTraffic"

JSON で書かれた前述の例のバージョンを示します。

```
{
    "version": 0.0,
    "Resources": [
        {
            "TargetService": {
                "Type": "AWS::ECS::Service",
                "Properties": {
                    "TaskDefinition": "arn:aws:ecs:us-east-1:111222333444:task-
definition/my-task-definition-family-name:1",
                    "LoadBalancerInfo": {
                         "ContainerName": "SampleApplicationName",
                         "ContainerPort": 80
                    },
                    "PlatformVersion": "LATEST",
                    "NetworkConfiguration": {
                         "AwsvpcConfiguration": {
                             "Subnets": [
                                 "subnet-1234abcd",
                                 "subnet-5678abcd"
                             ],
                             "SecurityGroups": [
```

```
"sg-12345678"
                             ],
                             "AssignPublicIp": "ENABLED"
                         }
                    },
                    "CapacityProviderStrategy": [
                         {
                             "Base" : 1,
                             "CapacityProvider" : "FARGATE_SPOT",
                             "Weight" : 2
                         },
                         {
                             "Base" : 0,
                             "CapacityProvider" : "FARGATE",
                             "Weight" : 1
                         }
                    ]
                }
            }
        }
    ],
    "Hooks": [
        {
            "BeforeInstall": "LambdaFunctionToValidateBeforeInstall"
        },
        {
            "AfterInstall": "LambdaFunctionToValidateAfterInstall"
        },
        {
            "AfterAllowTestTraffic": "LambdaFunctionToValidateAfterTestTrafficStarts"
        },
        {
            "BeforeAllowTraffic":
 "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
        },
        {
            "AfterAllowTraffic":
 "LambdaFunctionToValidateAfterAllowingProductionTraffic"
        }
    ]
}
```

デプロイ中のイベントのシーケンスを次に示します。

- 1. 最新の Amazon ECS アプリケーションが置き換えタスクセットにインストールされる前に、LambdaFunctionToValidateBeforeInstall と呼ばれる Lambda 関数が実行されます。
- 最新の Amazon ECS アプリケーションが置き換えタスクセットにインストールされた後、トラ フィックを受信する前に、LambdaFunctionToValidateAfterInstall と呼ばれる Lambda 関数が実行されます。
- 置き換えタスクセット上の Amazon ECS アプリケーションが、テストリスナーからのトラフィックの受信を開始した後、LambdaFunctionToValidateAfterTestTrafficStarts と呼ばれる Lambda 関数が実行されます。この関数は、デプロイが続行されるかどうかを判断するために、検証テストを実行する可能性があります。デプロイグループでテストリスナーを指定しない場合、このフックは無視されます。
- AfterAllowTestTraffic フックの検証テストがすべて完了した後、かつ、 最新の Amazon ECS アプリケーションに本稼働トラフィックが提供される前 に、LambdaFunctionToValidateBeforeAllowingProductionTraffic と呼ばれる Lambda 関数が実行されます。
- 5. 置き換えタスクセット上の最新の Amazon ECS アプリケーションに本稼働トラフィックが提供された後に、LambdaFunctionToValidateAfterAllowingProductionTraffic と呼ばれる Lambda 関数が実行されます。

フック中に実行される Lambda 関数は、検証テストを実行したり、トラフィックメトリクスを収集 したりできます。

AWS Lambda デプロイの AppSpec ファイルの例

Lambda 関数のバージョンをデプロイするために YAML で書かれた AppSpec ファイルの例を示します。

```
version: 0.0
Resources:
    - myLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
Hooks:
```

```
- BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
```

- AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"

JSON で書かれた前述の例のバージョンを示します。

```
{
  "version": 0.0,
  "Resources": [{
   "myLambdaFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "myLambdaFunction",
     "Alias": "myLambdaFunctionAlias",
     "CurrentVersion": "1",
     "TargetVersion": "2"
    }
   }
  }],
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
      },
      {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
   }
  ]
 }
```

デプロイ中のイベントのシーケンスを次に示します。

- myLambdaFunction という名前の Lambda 関数のバージョン 1 からバージョン 2 にトラフィッ クを移行する前に、デプロイでトラフィックの移行を開始する準備が整っていることを確認す る、LambdaFunctionToValidateBeforeTrafficShift という名前の Lambda 関数を実行し ます。
- LambdaFunctionToValidateBeforeTrafficShift が終了コード 0 (成功) を返した場合 は、myLambdaFunction のバージョン 2 へのトラフィックの移行を開始します。このデプロイ のデプロイ設定により、トラフィックが移行するレートが決まります。
- myLambdaFunction という名前の Lambda 関数のバージョン 1 からバージョン
 2 へのトラフィックの移行が完了したら、デプロイが正常に完了したことを確認する、LambdaFunctionToValidateAfterTrafficShift という名前の Lambda 関数を実行します。

EC2/オンプレミスデプロイの AppSpec ファイルの例

以下は、Amazon Linux、Ubuntu Server、または RHEL インスタンスへのインプレイスデプロイのた めの AppSpec ファイルの例です。

Note

Windows Server インスタンスへのデプロイメントでは、runas 要素をサポートしていません。Windows Server インスタンスにデプロイする場合は、AppSpec ファイルに含まれないでください。

```
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

Windows Server のインスタンスで、[os: linux] を [os: windows] に変更します。ま た、destination パスを完全に修飾する必要があります (例: c:\temp\webapps\Config、c: \temp\webapps\myApp)。runas エレメントは含めないでください。

デプロイ中のイベントのシーケンスを次に示します。

1. Scripts/UnzipResourceBundle.sh にあるスクリプトを実行します。
- 2. 前のスクリプトが終了コード 0 (成功) を返した場合、Scripts/UnzipDataBundle.sh にある スクリプトを実行します。
- ファイルを Config/config.txt のパスからパス /webapps/Config/config.txt にコピー します。
- 4. source ディレクトリのすべてのファイルを再帰的に /webapps/myApp ディレクトリにコピー します。
- 5. Scripts/RunResourceTests.sh にあるスクリプトを、180 秒 (3 分) のタイムアウトで実行し ます。
- 6. Scripts/RunFunctionalTests.sh にあるスクリプトを、3600 秒 (1 時間) のタイムアウトで 実行します。
- 7. Scripts/MonitorService.sh にあるスクリプトを、ユーザー codedeploy として 3600 秒 (1 時間) のタイムアウトで実行します。

AppSpec ファイルの間隔

AppSpec ファイルの間隔の正しい形式を以下に示します。角括弧に含まれた番号は、項目の間に必要なスペースの数を示します。たとえば、[4] は、項目の間に 4 つのスペースを挿入することを意味します。CodeDeploy は、AppSpec ファイルの場所とスペースの数が正しくない場合、デバッグが困難なエラーを発生させることがあります。

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
```

```
[4]type:
[6]-[1]object-type
hooks:
[2]deployment-lifecycle-event-name:
[4]-[1]location:[1]script-location
[6]timeout:[1]timeout-in-seconds
[6]runas:[1]user-name
```

正しい間隔が設定された AppSpec ファイルの例を次に示します。

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

間隔の詳細については、YAML の仕様を参照してください。

AppSpec ファイルの検証とファイルの場所

ファイル構文

AWSが提供する AppSpec Assistant スクリプトを使用して、AppSpec ファイルの内容を検証できま す。<u>GitHub</u> 上に、AppSpec ファイルテンプレートとスクリプトを見つけることができます。 代わりに、<u>YAML Lint</u> や <u>Online YAML Parser</u> などのブラウザベースのツールを使用して YAML 構文 をチェックすることもできます。

ファイルの場所

AppSpec ファイルをアプリケーションのソースコンテンツのディレクトリ構造のルートディレクト リに配置したことを確認するには、次のいずれかのコマンドを実行します。

ローカルの Linux、macOS、Unix インスタンスでは、以下を実行します。

ls path/to/root/directory/appspec.yml

AppSpec ファイルがその場所にない場合、「No such file or directory」エラーが表示されます。

ローカル Windows インスタンスの場合

dir path\to\root\directory\appspec.yml

AppSpec ファイルがその場所にない場合、「File Not Found」エラーが表示されます。

CodeDeploy エージェント設定リファレンス

CodeDeploy エージェントがインストールされている場合、設定ファイルはインスタンスに配置され ます。この設定ファイルは、インスタンスとやり取りするときに使用する CodeDeploy のディレク トリパスおよびその他の設定を指定します。ファイルの一部の設定オプションは変更できます。

Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL) インスタンスの場合、設定ファイルの名前は、codedeployagent.yml です。ファイルは、/etc/codedeploy-agent/conf ディレクトリに配置されます。

Windows Server インスタンスの場合、設定ファイルは conf.yml という名前になります。ファイル は、C:\ProgramData\Amazon\CodeDeploy ディレクトリに配置されます。

設定には以下が含まれます。

:log_aws_wire:

true に設定するため CodeDeploy エージェ ントが Amazon S3 からワイヤログをキャプ チャし、:log_dir: 設定で指定された場所にある

codedeploy-agent.wire.log という名 前のファイルに書き込みます。

▲ Warning

ワイヤログの取得に必要な時間の み、:log aws wire:をtrueに設定す る必要があります。codedeployagent.wire.log ファイルは非常に 大きなサイズになる場合があります。 このファイルのワイヤログ出力には、 この設定が true に設定されている 間に Amazon S3 との間で転送された ファイルのプレーンテキストの内容な どの重要情報が含まれている場合があ ります。ワイヤログには、CodeDepl oy デプロイに関連するアクティビティ だけでなく、この設定が AWS に設定 されている間に true アカウントに関 連付けられていたすべての Amazon S3 アクティビティに関する情報が含まれ ます。

デフォルトの設定はfalse です。

この設定は、すべてのインスタンスタイプに適 用されます。この設定を使用できるようにする には、この設定を Windows サーバーインスタ ンスに追加する必要があります。

:log_dir:	CodeDeploy エージェントオペレーションに関 連するログファイルが保存されるインスタンス のフォルダ。	
	デフォルトの設定は、Amazon Linux、Ubu ntu Server、RHEL インスタンス用の '/var/ log/aws/codedeploy-agent'、お よび Windows Server インスタンス用の C: \ProgramData\Amazon\CodeDeploy \log です。	
:pid_dir:	codedeploy-agent.pid が保存されてい るフォルダ。	
	このファイルには、CodeDeploy エージェント のプロセス ID (PID) が含まれます。デフォル トの設定は ' /opt/codedeploy-agent/ state/.pid' です。	
	この設定は、Amazon Linux、Ubuntu Server、RHEL インスタンスにのみ適用されま す。	
:program_name:	CodeDeploy エージェントのプログラム名。	
	デフォルトの設定はcodedeploy-agent で す。	
	この設定は、Amazon Linux、Ubuntu Server、RHEL インスタンスにのみ適用されま す。	

:root_dir:	関連するリビジョン、デプロイ履歴、およびイ ンスタンスのデプロイスクリプトが保存される フォルダ。	
	デフォルトの設定は、Amazon Linux、Ubuntu Server、RHEL インスタンス用の /opt/code deploy-agent/deployment-root 、 および Windows Server インスタンス用の C: \ProgramData\Amazon\CodeDeploy で す。	
:verbose:	CodeDeploy エージェントでインスタンスのデ バッグメッセージログファイルを表示する場合 は、true に設定します。	
	デフォルトの設定はfalse です。	
:wait_between_runs:	保留中のデプロイに対する CodeDeploy エー ジェントによる CodeDeploy のポーリング間隔 (秒単位)。	
	デフォルトの設定は1 です。	

:on_premises_config_file:	 オンプレミスインスタンスの場 合、codedeploy.onpremises.yml (Ubuntu Server および RHEL の場合)、または conf.onpremises.yml (Windows Server の場合)という名前の設定ファイルの別の場所へのパスです。 デフォルトでは、これらのファイルは Ubuntu Server および RHEL の場合は /etc/code deploy-agent/conf /codedeplo y.onpremises.yml 、Windows Server の 場合は C:\ProgramData\Amazon\CodeD eploy conf.onpremises.yml に保存されます。 バージョン 1.0.1.686 以降のバージョンの CodeDeploy エージェントで利用できます。
:proxy_uri:	(オプション) CodeDeploy エージェント が CodeDeploy オペレーション AWS のた めに に接続する HTTP CodeDeploy プロキ シ。https://user:password@my.pr oxy:443/path?query のような形式を使 用します。
	バージョン 1.0.1.824 以降のバージョンの CodeDeploy エージェントで利用できます。

:max_revisions:	(オプション) CodeDeploy エージェントでアー カイブするデプロイグループ用のアプリケー ションリビジョンの数。指定された数を超える リビジョンは削除されます。 正の整数を入力します。値を指定しない場 合、CodeDeploy は現在デプロイされたリビ ジョンに加えて 5 つの最新のリビジョンを保持 します。 バージョン 1.0.1.966 以降のバージョンの CodeDeploy エージェント でサポートされます 。
:enable_auth_policy:	(オプション) IAM 認可 を使用して、アクセス コントロールを設定し、CodeDeploy エージェ ントが使用している IAM ロールまたはユー ザーのアクセス許可を制限する場合に true に設定します。Amazon Virtual Private Cloud で CodeDeploy を使用 にするには、この値 は、true である必要があります。 デフォルトの設定はfalse です。
:disable_imds_v1:	この設定は、CodeDeploy エージェント 1.7.0 以降で使用できます。 IMDSv2 エラーが発生したときに IMDSv1 への フォールバックを無効にするtrueには、 に設 定します。 IMDSv2 デフォルトは です false (フォールバックを有効にします)。

関連トピック

CodeDeploy エージェントの使用

<u>CodeDeploy エージェントのオペレーションの管理</u>

AWS CloudFormation CodeDeploy リファレンスの テンプレート

このセクションでは、CodeDeploy デプロイで動作するように設計された AWS CloudFormation リ ソース、変換、フックについて説明します。CodeDeploy の AWS CloudFormation フックによって管 理されるスタック更新を作成する手順については、「」を参照してください。 <u>を使用して Amazon</u> ECS ブルー/グリーンデプロイを作成する AWS CloudFormation

Note

AWS CloudFormation フックは の AWS CloudFormation コンポーネントの一部 AWS であり、CodeDeploy ライフサイクルイベントフックとは異なります。

CodeDeploy で使用できる他のメソッドに加えて、 AWS CloudFormation テンプレートを使用して以下のタスクを実行できます。

- アプリケーションを作成します。
- デプロイグループを作成し、ターゲットリビジョンを指定します。
- ・ デプロイ設定を作成します。
- Amazon EC2 インスタンスを作成します。

AWS CloudFormation は、 テンプレートを使用して AWS リソースをモデル化およびセットアップ するのに役立つサービスです。 AWS CloudFormation テンプレートは、形式が JSON 標準に準拠し ているテキストファイルです。必要なすべての AWS リソースを記述するテンプレートを作成し、 AWS CloudFormation がそれらのリソースのプロビジョニングと設定を行います。

詳細については、<u>AWS CloudFormationユーザーガイド</u>の「<u>AWS CloudFormation とは</u>」および 「Working with AWS CloudFormation Templates」を参照してください。

組織内で CodeDeploy と互換性のある AWS CloudFormation テンプレートを使用する場合は、 管理 者として、 AWS CloudFormation が AWS CloudFormation 依存する AWS サービスとアクションへ のアクセスとアクセスを許可する必要があります。アプリケーション、デプロイグループ、デプロ イ設定を作成するアクセス許可を付与するには、作業するユーザーのアクセス許可セットに次のポリ シーを追加します AWS CloudFormation。

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

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:*"
    ],
    "Resource": "*"
    }
  ]
}
```

ポリシーの詳細については、以下のトピックを参照してください。

- Amazon EC2 インスタンスを作成する許可セットのユーザーに追加する必要があるポリシーを表示するには、「<u>CodeDeploy 用の Amazon EC2 インスタンスを作成する (AWS CloudFormation テ</u>ンプレート)」を参照してください。
- 許可セットにポリシーを追加する方法については、「IAM ユーザーガイド」の「<u>アクセス権限</u> セットを作成します。」を参照してください。
- CodeDeployのアクションとリソースの限定されたセットにユーザーを制限する方法については、 「AWS CodeDeployのマネージド (事前定義) ポリシー」を参照してください。

次の表は、 AWS CloudFormation テンプレートがユーザーに代わって実行できるアクションと、 AWS CloudFormation テンプレートに追加できる AWS リソースタイプとそのプロパティタイプに関 する詳細情報へのリンクを示しています。

アクション	AWS CloudFormation リファ レンス	参照タイプ
CodeDeploy のアプリケー ションを作成します。	<u>AWS::CodeDeploy::Applicatio</u> <u>n</u>	AWS CloudFormation リソー ス
アプリケーションリビジョン のデプロイに使用されるデ プロイグループの詳細を作成 し、指定します。1	<u>AWS::CodeDeploy::D</u> eploymentGroup	AWS CloudFormation リソー ス
CodeDeploy がデプロイ中に 使用する一連のデプロイの ルール、デプロイの成功条	AWS::CodeDeploy::D eploymentConfig	AWS CloudFormation リソー ス

AWS CodeDeploy

アクション	AWS CloudFormation リファ レンス	参照タイプ
件、デプロイの失敗条件を作 成します。		
Amazon EC2 インスタンスを 作成します。²	AWS::EC2::Instance	AWS CloudFormation リソー ス
変換とAWS::Code Deploy::BlueGreen フッ クを使用して、CodeDeplo y AWS CloudFormation	AWS::CodeDeployBlueGreen	AWS : : CodeDeployBlu eGreen 変換は、AWS CloudFormation によりホスト されるマクロです。
Aws::codebeployBlu eGreen ブルー/グリーンデプ ロイのスタック更新の管理、 リソースの作成、トラフィッ クの移行を行います。 ³	<u>AWS::CodeDeploy::BlueGreen</u>	AWS::CodeDeploy::B lueGreen フックは のHookリソースとして構 造化されています AWS CloudFormation。指定された CodeDeploy ライフサイクル イベントフックを指すことに よって、フックには、CodeDe ploy AppSpec ファイルの代 わりにパラメータが含まれま す。

アクション	AWS CloudFormation リファ レンス	参照タイプ		
¹ デプロイグループの一部としてデプロイするアプリケーショ ンリビジョンのバージョンを指定する場合、プロビジョニン グプロセスが完了するとすぐに、ターゲットリビジョンがデ プロイされます。テンプレート設定の詳細については、 <u>AWS</u> <u>CloudFormation ユーザーガイド</u> の <u>CodeDeploy Deploymen</u> <u>tGroup デプロイメントリビジョン S3Location</u> と CodeDeplo y デプロイ DeploymentGroup デプロイメントリビジョン GitHubLocation を参照してください。				
² CodeDeploy がサポートされているリージョンで、Amazon EC2 インスタンスの作成に使用できるテンプレートを提供 します。これらのテンプレートの使用の詳細については、 「 <u>CodeDeploy 用の Amazon EC2 インスタンスを作成する</u> (AWS CloudFormation テンプレート)」を参照してください。				
³ このデプロイ設定では、Amazon ECS ブルー/グリーンデプロ イのみがサポートされています。AWS CloudFormationによる Amazon ECS ブルー/グリーンデプロイのデプロイ構成の詳細に ついては、「 <u>AWS CloudFormation blue/green デプロイのため</u> <u>のデプロイ設定 (Amazon ECS)</u> 」を参照してください。による Amazon ECS ブルー/グリーンデプロイの詳細 AWS CloudForm ation と CodeDeploy でデプロイを表示する方法については、				
'」を参照してくたさい <u>を使用して Amazon ECS フルー/ク</u> リーンデプロイを作成する AWS CloudFormation。				

Amazon Virtual Private Cloud で CodeDeploy を使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC と CodeDeploy 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、CodeDeploy はパ ブリックインターネットを経由せずに、VPC のリソースと通信できます。

Amazon VPC は、定義した仮想ネットワークで AWS リソースを起動するために使用できる AWS サービスです。VPC を使用することで、IP アドレス範囲、サブネット、ルートテーブル、ネット ワークゲートウェイなどのネットワーク設定を制御できます。VPC エンドポイントでは、VPC と AWS サービス間のルーティングは AWS ネットワークによって処理され、IAM ポリシーを使用して サービスリソースへのアクセスを制御できます。

VPC を CodeDeploy に接続するには、CodeDeploy の インターフェイス VPC エンドポイント を 定義します。インターフェイスエンドポイントは、サポートされている AWS サービス宛てのト ラフィックのエントリポイントとして機能するプライベート IP アドレスを持つ Elastic Network Interface です。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、および VPN 接続を必要とせず、信頼性が高くスケーラブルな CodeDeploy へ の接続を提供します。詳細については、「Amazon VPC ユーザーガイド」の「<u>Amazon VPC とは</u>」 を参照してください。

インターフェイス VPC エンドポイントは AWS PrivateLink、プライベート IP アドレスを持つ Elastic Network Interface を使用して AWS サービス間のプライベート通信を可能にする AWS テクノ ロジーを利用しています。詳細については、「AWS PrivateLink」を参照してください。

以下の手順は、Amazon VPC のユーザー向けです。詳細については、『Amazon VPC ユーザーガイ ド』の「開始方法」を参照してください。

可用性

CodeDeploy には 2 つの VPC エンドポイントがあります。1 つは CodeDeploy エージェントオペ レーション用で、もう 1 つは CodeDeploy API オペレーション用です。次の表は、各エンドポイン トでサポートされている AWS リージョンを示しています。

リージョン名	リージョンコード	エージェントのエン ドポイント	API エンドポイント
米国東部 (バージニア 北部)	us-east-1	あり	あり
米国東部 (オハイオ)	us-east-2	あり	あり
米国西部 (北カリフォ ルニア)	us-west-1	あり	あり
米国西部 (オレゴン)	us-west-2	あり	あり
アフリカ (ケープタウ ン)	af-south-1	はい	いいえ

AWS CodeDeploy

リージョン名	リージョンコード	エージェントのエン ドポイント	API エンドポイント
アジアパシフィック (香港)	ap-east-1	あり	あり
アジアパシフィック (ハイデラバード)	ap-south-2	はい	いいえ
アジアパシフィック (ジャカルタ)	ap-southeast-3	はい	いいえ
アジアパシフィック (メルボルン)	ap-southeast-4	はい	いいえ
アジアパシフィック (ムンバイ)	ap-south-1	あり	あり
アジアパシフィック (大阪)	ap-northeast-3	はい	いいえ
アジアパシフィック (ソウル)	ap-northeast-2	あり	あり
アジアパシフィック (シンガポール)	ap-southeast-1	あり	あり
アジアパシフィック (シドニー)	ap-southeast-2	あり	あり
アジアパシフィック (東京)	ap-northeast-1	あり	あり
カナダ (中部)	ca-central-1	あり	あり
中国 (北京)	cn-north-1	はい	いいえ
中国 (寧夏)	cn-northwest-1	いいえ	いいえ
欧州 (フランクフルト)	eu-central-1	あり	あり

AWS CodeDeploy

リージョン名	リージョンコード	エージェントのエン ドポイント	API エンドポイント
欧州 (アイルランド)	eu-west-1	あり	あり
欧州 (ロンドン)	eu-west-2	あり	あり
欧州 (ミラノ)	eu-south-1	はい	いいえ
欧州 (パリ)	eu-west-3	あり	あり
欧州 (スペイン)	eu-south-2	はい	いいえ
欧州 (ストックホルム)	eu-north-1	あり	あり
欧州 (チューリッヒ)	eu-central-2	はい	いいえ
イスラエル (テルアビ ブ)	il-central-1	あり	あり
中東 (バーレーン)	me-south-1	あり	あり
中東 (UAE)	me-central-1	はい	いいえ
南米 (サンパウロ)	sa-east-1	あり	あり
AWS GovCloud (米国 東部)	us-gov-east-1	いいえ	いいえ
AWS GovCloud (米国 西部)	us-gov-west-1	いいえ	いいえ

CodeDeploy 用の VPC エンドポイントを作成する

VPC で CodeDeploy の使用を開始するには、CodeDeploy のインターフェイス VPC エンドポイント を作成します。CodeDeploy には、エージェントの Git オペレーションと CodeDeploy API オペレー ション用に別々のエンドポイントが必要です。ビジネスニーズに応じて、複数の VPC エンドポイ ントを作成する必要がある場合があります。CodeDeploy 用の VPC エンドポイントを作成するとき は、[AWS サービス] を選択し、[サービス名] で、次のオプションから選択します。

- com.amazonaws.*region*.codecommit: CodeDeploy API オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択します。例えば、ユーザーが AWS CLI、CodeDeploy API、または AWS SDKs を使用して CreateApplication、、などのオペレーションで CodeDeploy とやり取りする場合はGetDeployment、このオプションを選択しますListDeploymentGroups。
- com.amazonaws.*region*.codecommit: CodeDeploy エージェント オペレーション用の VPC エンドポイントを作成する場合は、このオプションを選択します。また、:enable_auth_policy: に true エージェント設定ファイルで、必要な権限をアタッチする必要があります 詳細については、「CodeDeploy エージェントと IAM 許可を設定する」を参照してください。

Lambda または ECS デプロイを使用している場合は、用の VPC エンドポイントを作成する必要が あります。com.amazonaws.**region**.codedeploy Amazon EC2 デプロイを使用しているお客様につ いては、com.amazonaws.**region**.codedeploy と com.amazonaws.**region**.codedeploy-commandssecure はどちらも VPC エンドポイントが必要です

CodeDeploy エージェントと IAM 許可を設定する

CodeDeploy で Amazon VPC エンドポイントを使用するには、:enable_auth_policy: に true EC2 またはオンプレミスインスタンスにあるエージェント設定ファイルに次の値を設定する必要が あります。設定ファイルの形式の詳細については、「<u>CodeDeploy エージェント設定リファレンス</u>」 を参照してください。

Amazon EC2 インスタンスプロファイル (Amazon EC2 インスタンスを使用している場合) または IAM ユーザーまたはロール (オンプレミスインスタンスを使用している場合) に次の IAM アクセス許 可を追加する必要があります。

```
{
   "Statement": [
    {
        "Action": [
            "codedeploy-commands-secure:GetDeploymentSpecification",
            "codedeploy-commands-secure:PollHostCommand",
            "codedeploy-commands-secure:PutHostCommandAcknowledgement",
            "codedeploy-commands-secure:PutHostCommandComplete"
        ],
        "Effect": "Allow",
        "Resource": "*"
    }
]
```

}

詳細については、『Amazon VPC ユーザーガイド』の「<u>インターフェイスエンドポイントの作成</u>」 を参照してください。

CodeDeploy リソースキットリファレンス

CodeDeploy が依存するファイルの多くは、公開されている AWS リージョン固有の Amazon S3 バケットに保存されます。このようなファイルには、CodeDeploy エージェントのインストール ファイル、テンプレート、サンプルアプリケーションファイルがあります。この一連のファイルを CodeDeploy リソースキットと呼びます。

トピック

- リージョン別リソースキットバケット名
- リソースキットの内容
- リソースキットのファイルのリストの表示
- リソースキットのファイルのダウンロード

リージョン別リソースキットバケット名

この表は、本ガイドの一部の手順で必要な *bucket-name* を置換する名前の一覧です。これら は、CodeDeploy リソースキットファイルが含まれている Amazon S3 バケットの名前です。

Note

アジアパシフィック (香港) リージョンの Amazon S3 バケットにアクセスするには、 AWS アカウントでリージョンを有効にする必要があります。詳細については、<u>AWS 「リージョ</u> <u>ンの管理</u>」を参照してください。

リージョン名	bucket-name 置換	リージョン識別子
米国東部 (バージニア北部)	aws-codedeploy-us-east-1	us-east-1
米国東部 (オハイオ)	aws-codedeploy-us-east-2	us-east-2
米国西部 (北カリフォルニア)	aws-codedeploy-us-west-1	us-west-1

AWS CodeDeploy

リージョン名	bucket-name 置換	リージョン識別子
米国西部 (オレゴン)	aws-codedeploy-us-west-2	us-west-2
アフリカ (ケープタウン)	aws-codedeploy-af-south-1	af-south-1
アジアパシフィック (香港)	aws-codedeploy-ap-east-1	ap-east-1
アジアパシフィック (ハイデラ バード)	aws-codedeploy-ap-south-2	ap-south-2
アジアパシフィック (ジャカル タ)	aws-codedeploy-ap-southeast -3	ap-southeast-3
アジアパシフィック (メルボル ン)	aws-codedeploy-ap-southeast -4	ap-southeast-4
アジアパシフィック (ムンバ イ)	aws-codedeploy-ap-south-1	ap-south-1
アジアパシフィック (大阪)	aws-codedeploy-ap-northeast -3	ap-northeast-3
アジアパシフィック (ソウル)	aws-codedeploy-ap-northeast -2	ap-northeast-2
アジアパシフィック (シンガ ポール)	aws-codedeploy-ap-southeast -1	ap-southeast-1
アジアパシフィック (シド ニー)	aws-codedeploy-ap-southeast -2	ap-southeast-2
アジアパシフィック (東京)	aws-codedeploy-ap-northeast -1	ap-northeast-1
カナダ (中部)	aws-codedeploy-ca-central-1	ca-central-1
欧州 (フランクフルト)	aws-codedeploy-eu-central-1	eu-central-1
欧州 (アイルランド)	aws-codedeploy-eu-west-1	eu-west-1

リージョン名	bucket-name 置換	リージョン識別子
欧州 (ロンドン)	aws-codedeploy-eu-west-2	eu-west-2
欧州 (ミラノ)	aws-codedeploy-eu-south-1	eu-south-1
欧州 (パリ)	aws-codedeploy-eu-west-3	eu-west-3
欧州 (スペイン)	aws-codedeploy-eu-south-2	eu-south-2
欧州 (ストックホルム)	aws-codedeploy-eu-north-1	eu-north-1
欧州 (チューリッヒ)	aws-codedeploy-eu-central-2	eu-central-2
イスラエル (テルアビブ)	aws-codedeploy-il-central-1	il-central-1
中東 (バーレーン)	aws-codedeploy-me-south-1	me-south-1
中東 (UAE)	aws-codedeploy-me-central-1	me-central-1
南米 (サンパウロ)	aws-codedeploy-sa-east-1	sa-east-1
AWS GovCloud (米国東部)	aws-codedeploy-us-gov-east-1	us-gov-east-1
AWS GovCloud (米国西部)	aws-codedeploy-us-gov-west- 1	us-gov-west-1

リソースキットの内容

次の表に、CodeDeploy リソースキットのファイルを一覧表示します。

ファイル	説明
LATEST_VERSION	Amazon EC2 Systems Manager などの更新メ カニズムによって使用され、CodeDeploy エー ジェントの最新バージョンを特定するファイル です。
VERSION	自動更新メカニズムが CodeDeploy エージェン トバージョン 1.1.0 で削除され、このファイル

AWS CodeDeploy

ファイル	説明
	は使用されなくなりました。インスタンスで実 行されているときに CodeDeploy エージェント が自身の更新に使用するファイルです。
codedeploy-agent.noarch.rpm	Amazon Linux および Red Hat Enterprise Linux (RHEL) 用の CodeDeploy エージェントです。 同じベースファイル名で、異なるバージョン (-1.0-0 など) の複数のファイルが存在するこ とがあります。
codedeploy-agent_all.deb	Ubuntu サーバー用の CodeDeploy エージェン トです。同じベースファイル名で、異なるバー ジョン (_1.0-0 など) の複数のファイルが存在 することがあります。
codedeploy-agent.msi	Windows サーバー用の CodeDeploy エージェ ントです。同じベースファイル名で、異なる バージョン (-1.0-0 など) の複数のファイルが 存在することがあります。
install	CodeDeploy エージェントをより簡単にインス トールするために使用できるファイルです。
CodeDeploy_SampleCF_Templat e.json	Amazon Linux または Windows Server を実 行する 1 つから 3 つの Amazon EC2 インス タンスから起動するために使用できる AWS CloudFormation テンプレート。同じベース ファイル名で、異なるバージョン (-1.0.0 な ど) の複数のファイルが存在することがありま す。

ファイル	説明
CodeDeploy_SampleCF_ELB_Int egration.json	Apache ウェブサーバーで実行されている負荷 分散されたサンプルウェブサイトを作成する ために使用できる AWS CloudFormation テン プレート。アプリケーションは、作成された リージョン内のすべてのアベイラビリティー ゾーンに分散されるように設定されます。この テンプレートは、3 つの Amazon EC2 インス タンスと IAM インスタンスプロファイルを作 成し、インスタンスに Amazon S3、Amazon EC2 Auto Scaling AWS CloudFormation、およ び Elastic Load Balancing のリソースへのアク セスを許可します。また、ロードバランサーと CodeDeploy サービスロールも作成します。
SampleApp_ELB_Integration.zip	Elastic Load Balancing ロードバランサーに登 録されている Amazon EC2 インスタンスにデ プロイできるサンプルアプリケーションリビ ジョンです。
SampleApp_Linux.zip	Amazon Linux を実行している Amazon EC2 イ ンスタンス、または Ubuntu サーバーまたは RHEL インスタンスにデプロイできるサンプル アプリケーションリビジョンです。同じベース ファイル名で、異なるバージョン (-1.0 など) の複数のファイルが存在することがあります。
SampleApp_Windows.zip	Windows Server インスタンスにデプロイでき るサンプルのアプリケーションリビジョンで す。同じベースファイル名で、異なるバージョ ン (-1.0 など) の複数のファイルが存在するこ とがあります。

リソースキットのファイルのリストの表示

ファイルの一覧を表示するには、使用しているリージョンの aws s3 ls コマンドを使用します。

(〕 Nc 各 す	ote バク 。	アツ	トのファイル	は、対応するリージョンのリソースで作業するように設計されていま
• (aws	s3	ls	recursive	s3://aws-codedeploy-us-east-2region us-east-2
• (aws	s3	ls	recursive	s3://aws-codedeploy-us-east-1region us-east-1
• (aws	s3	ls	recursive	s3://aws-codedeploy-us-west-1region us-west-1
	aws	s3	ls	recursive	s3://aws-codedeploy-us-west-2region us-west-2
• (aws	s3	ls	recursive	s3://aws-codedeploy-ca-central-1region ca-central-1
	aws	s3	ls	recursive	s3://aws-codedeploy-eu-west-1region eu-west-1
	aws	s3	ls	recursive	s3://aws-codedeploy-eu-west-2region eu-west-2
	aws	s3	ls	recursive	s3://aws-codedeploy-eu-west-3region eu-west-3
	aws	s3	ls	recursive	s3://aws-codedeploy-eu-central-1region eu-central-1
	aws	s3	ls	recursive	s3://aws-codedeploy-il-central-1region il-central-1
	aws	s3	ls	recursive	s3://aws-codedeploy-ap-east-1region ap-east-1
• (aws	s3	ls	recursive	s3://aws-codedeploy-ap-northeast-1region ap-northeast-1
	aws	s3	ls	recursive	s3://aws-codedeploy-ap-northeast-2region ap-northeast-2
	aws	s3	ls	recursive	s3://aws-codedeploy-ap-southeast-1region ap-southeast-1
•	aws	s3	ls	recursive	s3://aws-codedeploy-ap-southeast-2region ap-southeast-2

aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-4 --region ap-southeast-4

aws s3 ls --recursive s3://aws-codedeploy-ap-south-1 --region ap-south-1

aws s3 ls --recursive s3://aws-codedeploy-sa-east-1 --region sa-east-1

リソースキットのファイルのダウンロード

ファイルをダウンロードするには、使用しているリージョンの aws s3 cp コマンドを使用します。

Note

最後にピリオド (.) を使用してください。このコマンドは、現在のディレクトリにファイル をダウンロードします。

たとえば、次のコマンドは、バケットの /samples/latest/ フォルダの 1 つから SampleApp_Linux.zip という名前の 1 つのファイルをダウンロードします。

aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region
us-east-2

aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region
us-east-1

aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region
 us-west-1

aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region
us-west-2

aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . -region ca-central-1

aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region
eu-west-1

aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region
eu-west-2

aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region
eu-west-3

aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . -region eu-central-1

aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . -region il-central-1

aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region
ap-east-1

aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . -region ap-northeast-1

aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . -region ap-northeast-2

aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . -region ap-southeast-1

aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . -region ap-southeast-2

aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Linux.zip . -region ap-southeast-4

aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . -region ap-south-1

aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region
sa-east-1

すべてのファイルをダウンロードするには、リージョンで次のコマンドの1つを使用します。

aws s3 cp --recursive s3://aws-codedeploy-us-east-2 . --region us-east-2 aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1 aws s3 cp --recursive s3://aws-codedeploy-us-west-1 . --region us-west-1 aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2 aws s3 cp --recursive s3://aws-codedeploy-ca-central-1 . --region ca-central-1 aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1 aws s3 cp --recursive s3://aws-codedeploy-eu-west-2 . --region eu-west-2 aws s3 cp --recursive s3://aws-codedeploy-eu-west-3 . --region eu-west-3 aws s3 cp --recursive s3://aws-codedeploy-eu-central-1 . --region eu-central-1 aws s3 cp --recursive s3://aws-codedeploy-il-central-1 . --region il-central-1 aws s3 cp --recursive s3://aws-codedeploy-ap-east-1 . --region ap-east-1 aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-1 . --region ap-northeast-1 aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-2 . --region ap-northeast-2 aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-1 . --region ap-southeast-1 aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2 aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-4 . --region ap-southeast-4 aws s3 cp --recursive s3://aws-codedeploy-ap-south-1 . --region ap-south-1 aws s3 cp --recursive s3://aws-codedeploy-sa-east-1 . --region sa-east-1

CodeDeploy のクォータ

次の表では、CodeDeploy のクォータについて説明します。

Note

EC2/オンプレミスのインプレースデプロイ実行時間の制限はさまざまです。2023 年 6 月より前に作成されたカスタムデプロイ設定の場合、制限は 8 時間です。2023 年 6 月以降に作成されたカスタムデプロイ設定の場合、制限は 12 時間です。事前定義済みのデプロイ設定の場合、制限は 12 時間です。

名前	デフォルト	引き上げ可能	説明
AWS Lambda デプロイを数時間で実行	サポートされてい る各リージョン: 50	い え	AWS Lambda デプロイが 実行できる最大時間 (最 初のトラフィックシフト と最後のトラフィックシ フトの間の最大時間に対 して 48 時間、および 2 つのライフサイクルフッ クごとに 1 時間)
アカウントあたりの関連付けられたア プリケーション (リージョンごと)	サポートされてい る各リージョン: 1,000	<u>可</u> 能	1 つのリージョンの AWS アカウントに関連付けら れているアプリケーショ ンの最大数
デプロイグループごとの関連付けられ たアラーム	サポートされてい る各リージョン: 50	<u>可</u> 能	デプロイグループに関連 付けられるアラームの最 大数

名前	デフォルト	引き上げ可能	説明
デプロイグループ内の Auto Scaling グ ループ	サポートされてい る各リージョン: 10	<u>可</u> 能	デプロイグループの Amazon EC2 Auto Scaling グループの最大数
アカウントあたりの同時実行デプロイ	サポートされてい る各リージョン: 1,300	<u>可</u> 能	AWS アカウントに関連 付けられた同時デプロイ の最大数。Amazon EC2 Auto Scaling グループ のスケールアップされ た Amazon EC2 インス タンスへの各デプロイ は、EC2 インスタンスが 関連付けられているアプ リケーションごとに、単 ーの同時デプロイとして カウントされます。
デプロイグループあたりの同時実行デ プロイ	サポートされてい る各リージョン: 1	[い い え]	デプロイグループへの同 時デプロイの最大数。こ の制限により、同じデプ ロイグループに対して同 じアプリケーションを同 時デプロイすることを防 ぎます。
アカウントあたりのカスタムデプロイ 設定	サポートされてい る各リージョン: 50	い い え	AWS アカウントに関連付 けられたカスタムデプロ イ設定の最大数

名前	デフォルト	引き上げ可能	説明
単一のアプリケーションに関連付けら れたデプロイグループ	サポートされてい る各リージョン: 1,000	<u>可</u> 能	1 つのアプリケーション に関連付けられるデプロ イグループの最大数
EC2/オンプレミスの Blue/Green デプロ イ実行 (時間)	サポートされてい る各リージョン: 109	いえ	EC2/オンプレミスのブ ルー/グリーンデプロイ で実行できる最大時間数 (上記の 2 つの各制限に 48 時間 + 発生する可能性 のある 13 のライフサイ クルイベントに対して 1 時間)
EC2/オンプレミスのインプレースデプ ロイ実行 (時間)	サポートされてい る各リージョン: 12	い い え	EC2/オンプレミスインプ レースデプロイで実行で きる最大時間数
デプロイグループのイベント通知トリ ガー	サポートされてい る各リージョン: 10	<u>可</u> 能	デプロイグループのイベ ント通知トリガーの最大 数
アカウントあたりの GitHub 接続トーク ン	サポートされてい る各リージョン: 25	い い え	1 つの AWS アカウント の GitHub 接続トークン の最大数
EC2/オンプレミスの Blue/Green デプロ イ時に、デプロイが完了してから元の インスタンスが終了するまでの時間	サポートされてい る各リージョン: 48	い い え	EC2/オンプレミスのブ ルー/グリーンデプロイに おいて、デプロイが完了 してから元のインスタン スが終了するまでの最大 時間数

名前	デフォルト	引き上げ可能	説明
EC2/オンプレミスの Blue/Green デプロ イ時に、リビジョンのデプロイから、 トラフィックが代替インスタンスにシ フトするまでの時間	サポートされてい る各リージョン: 48	いいえ	EC2/オンプレミス のブ ルー/グリーンデプロイ において、リビジョンを デプロイしてから置き換 え先インスタンスへトラ フィックが移行されるま での最大時間数
デプロイあたりのインスタンス数	us-east-1: 2,000 他のサポートされ ている各リージョ ン: 1,000	<u>可</u> 能	1 つのデプロイ内のイン スタンスの最大数
Blue/Green デプロイがデプロイに成功 した後、元のデプロイからインスタン スを終了するまでの待機時間 (分)	サポートされてい る各リージョン: 2,800	い い え	Blue/Green デプロイがデ プロイに成功した後、元 のデプロイからインスタ ンスを終了するまでの最 大待機時間 (分)
AWS Lambda Canary または線形デプロ イ中の最初のトラフィックシフトと最 後のトラフィックシフトの間の分数	サポートされてい る各リージョン: 2,880	い い え	AWS Lambda Canary ま たは線形デプロイ中の最 初のトラフィックシフト と最後のトラフィックシ フトの間の最大分数

名前	デフォルト	引き上げ可能	説明
デプロイが失敗するまでの時間 (分) (ラ イフサイクルイベントが開始しない場 合)	サポートされてい る各リージョン: 5	いえ	ライフサイクルイベン トが (1) コンソールまた は AWS CLI create-de ployment コマンドを使用 してデプロイがトリガー された後、または (2) 以 前のライフサイクルイベ ントが完了した後にデプ ロイが失敗するまでの最 大分数。
Amazon ECS サービスに関連付けるこ とができるデプロイグループの数	サポートされてい る各リージョン: 1	[い い え]	Amazon ECS サービスに 関連付けることができる デプロイグループの最大 数
BatchGetOnPremisesInstances API ア クションへ渡すことができるインスタ ンスの数	サポートされてい る各リージョン: 100	いいえ	BatchGetOnPremises Instances API アクション に渡すことができるイン スタンスの最大数
アカウントあたりの進行中の同時実行 デプロイによって使用されたインスタ ンスの数	us-east-1: 3,000 他のサポートされ ている各リージョ ン: 1,000	<u>可</u> 能	進行中かつ1つのアカウ ントに関連付けられてい る同時デプロイで使用で きるインスタンスの最大 数
Amazon ECS デプロイ中のトラフィッ クルートのリスナー数	サポートされてい る各リージョン: 1	[い い え]	Amazon ECS デプロイ中 のトラフィックルートの リスナーの最大数

名前	デフォルト	引き上げ可能	説明
デプロイライフサイクルイベントが完 了しなかった場合に失敗するまでの時 間 (秒)	サポートされてい る各リージョン: 3,600 秒	いいえ	デプロイライフサイクル イベントが完了しなかっ た場合に失敗するまでの 最大時間 (秒)
デプロイグループ名のサイズ	サポートされてい る各リージョン: 100	い い え	デプロイグループ名の最 大文字数
タグキーのサイズ	サポートされてい る各リージョン: 128	い い え	タグキーの最大文字数
タグ値のサイズ	サポートされてい る各リージョン: 256	い い え	タグ値の最大文字数
デプロイグループのタグ	サポートされてい る各リージョン: 10	い い え	デプロイグループのタグ の最大数
AWS Lambda デプロイ中に 1 増分で移 行できるトラフィック	サポートされてい る各リージョン: 99	い い え	AWS Lambda デプロイ中 に 1 回の増分で移行でき るトラフィックの最大割 合

CodeDeploy のトラブルシューティング

このセクションのトピックは、CodeDeploy を使用するときに発生することがある問題やエラーの解 決にお使いください。

Note

多くのデプロイ失敗の原因を特定するには、デプロイプロセスで作成されたログファイル を確認できます。分かりやすいように、インスタンス別に表示するのではなく、Amazon CloudWatch Logs を使用してログファイルを集中的にモニタリングすることをお勧めしま す。詳細については、<u>Monitoring Deployments with Amazon CloudWatch Tools</u> を参照してく ださい。

トピック

- 一般的なトラブルシューティングの問題
- EC2/オンプレミスのデプロイに関する問題のトラブルシューティング
- Amazon ECS のデプロイに関する問題のトラブルシューティング
- AWS Lambda デプロイの問題のトラブルシューティング
- デプロイグループの問題のトラブルシューティング
- インスタンスの問題のトラブルシューティング
- GitHub トークンの問題のトラブルシューティング
- ・ Amazon EC2 Auto Scaling の問題のトラブルシューティング
- のエラーコード AWS CodeDeploy

一般的なトラブルシューティングの問題

トピック

- 一般的なトラブルシューティングのチェックリスト
- CodeDeploy デプロイリソースは、一部の AWS リージョンでのみ でサポートされています
- このガイドの手順が CodeDeploy コンソールと一致しない
- 必要な IAM ロールを取得できない

- 何らかのテキストエディタを使用して AppSpec ファイルとシェルスクリプトを作成すると、デプ ロイが失敗する場合がある
- macOS の Finder を使用してアプリケーションリビジョンをバンドルすると、デプロイが失敗する ことがある

一般的なトラブルシューティングのチェックリスト

次のチェックリストを使用して、失敗したデプロイをトラブルシューティングできます。

- デプロイが失敗した理由を確認するには、「<u>CodeDeploy デプロイの詳細を表示する</u>」および 「<u>View Instance Details</u>」を参照してください。原因を特定できない場合は、このチェックリスト の項目を確認します。
- 2. インスタンスが正しく設定されているかどうかを確認します。
 - インスタンスは、指定された EC2 キーペアで起動されましたか? 詳細については、「Amazon EC2 ユーザーガイド」の「EC2 キーペア」を参照してください。 Amazon EC2
 - ・ 正しい IAM インスタンスプロファイルがインスタンスにアタッチされていますか? 詳細については、CodeDeploy と共に動作するように Amazon EC2 インスタンスを構成しますおよびステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成するを参照してください。
 - インスタンスにタグが付けられていますか? 詳細については、Amazon EC2 <u>ユーザーガイド」</u>の「コンソールでのタグの使用」を参照してください。
 - CodeDeploy エージェントは、インスタンスにインストール後、更新、実行されていますか? 詳細については、「CodeDeploy エージェントのオペレーションの管理」を参照してください。インストールされているエージェントのバージョンを確認するには、「CodeDeploy エージェントのバージョンを特定します。」を参照してください。
- 3. アプリケーションとデプロイグループの設定を確認します。
 - アプリケーション設定を確認するには、「CodeDeploy を使用してアプリケーション詳細を表示 する」を参照してください。
 - デプロイグループの設定を確認するには、「<u>CodeDeploy を使用したデプロイグループの詳細の</u> 表示」を参照してください。
- 4. アプリケーションリビジョンが正しく設定されていることを確認します
 - AppSpec ファイルの形式を確認します。詳細については、<u>CodeDeploy 用のアプリケーション</u> <u>仕様ファイルをリビジョンに追加</u>および<u>CodeDeploy AppSpec ファイルのリファレンス</u>を参照 してください。

- GitHub レポジトリで Amazon S3 バケットを調べ、アプリケーションリビジョンが予期されている場所にあることを確認します。
- CodeDeploy アプリケーションリビジョンの詳細を調べ、正しく登録されていることを確認します。詳細については、CodeDeployを使用したアプリケーションリビジョン詳細の表示を参照してください。
- Amazon S3 からデプロイする場合は、Amazon S3 バケットをチェックし、アプリケーション リビジョンをダウンロードするアクセス許可が CodeDeploy に付与されていることを確認しま す。バケットポリシーの詳細については、デプロイの前提条件 を参照してください。
- GitHub からデプロイする場合、GitHub リポジトリをチェックし、アプリケーションリビジョン をダウンロードするアクセス許可が CodeDeploy に付与されていることを確認します。詳細に ついては、<u>CodeDeploy でデプロイを作成する</u>および<u>CodeDeploy のアプリケーションを使用し</u> た GitHub の認証を参照してください。
- 5. サービスロールが正しく設定されているかどうかを確認します。詳細については、<u>ステップ 2:</u> CodeDeployのサービスのロールを作成する を参照してください。
- 6. 「CodeDeployの開始方法」のステップに従って次の操作を行ったことを確認します。
 - 適切なアクセス許可を持つユーザーをプロビジョニングしました。
 - AWS CLIをインストールまたはアップグレードして設定する。
 - IAM インスタンスプロファイルとサービスロールを作成する。

詳細については、「<u>AWS CodeDeployのためのアイデンティティおよびアクセス管理</u>」を参照し てください。

7. AWS CLI バージョン 1.6.1 以降を使用していることを確認します。インストールしたバージョン を確認するには、aws --version を呼び出します。

それでも失敗したデプロイをトラブルシューティングできない場合は、このトピックの他の問題を確 認します。

CodeDeploy デプロイリソースは、一部の AWS リージョンでのみ でサ ポートされています

または AWS CLI CodeDeploy コンソールからアプリケーション、デプロイグループ、インスタン ス、またはその他のデプロイリソースが表示されないか、アクセスできない場合は、「」の AWS リージョン<u>とエンドポイント</u>にリストされているリージョンのいずれかを参照していることを確認し てくださいAWS 全般のリファレンス。 CodeDeploy デプロイで使用される EC2 インスタンスと Amazon EC2 Auto Scaling グループは、これらの AWS リージョンのいずれかで起動および作成する必要があります。

を使用している場合は AWS CLI、 から aws configure コマンドを実行します AWS CLI。その後、デ フォルトの AWS リージョンを表示して設定できます。

CodeDeploy コンソールを使用している場合は、ナビゲーションバーのリージョンセレクタから、サ ポートされている AWS リージョンのいずれかを選択します。

▲ Important

中国 (北京) リージョンまたは中国 (寧夏)でサービスを使用するには、そのリージョンのアカ ウントと認証情報が必要です。他の AWS リージョンのアカウントと認証情報は、北京およ び寧夏リージョンでは機能せず、その逆も同様です。 CodeDeploy リソースキットのバケット名や CodeDeploy エージェントのインストール手 順など、中国リージョン向けのいくつかのリソースに関する情報は、今回の「CodeDeploy ユーザーガイド」には含まれていません。 詳細については:

- 中国 (北京) リージョン AWS での の開始方法の CodeDeploy
- 「CodeDeploy User Guide for the China Regions (中国リージョン向け CodeDeploy ユー ザーガイド)」(英語版 | 中国語版)

このガイドの手順が CodeDeploy コンソールと一致しない

このガイドの手順は、新しいコンソールデザインで記述されています。古いバージョンのコンソール を使用した場合、古い概念が反映され、本ガイドの基本的な手順がそのまま適用されます。新しいコ ンソールのヘルプにアクセスするには、情報アイコンを選択します。

必要な IAM ロールを取得できない

AWS CloudFormation スタックの一部として作成された IAM インスタンスプロファイルまたはサー ビスロールに依存している場合、スタックを削除すると、すべての IAM ロールも削除されます。こ れが、IAM ロールが IAM コンソールに表示されなくなり、CodeDeploy が予期どおり動作しなくな る理由と考えられます。この問題を解決するには、削除された IAM ロールを再作成する必要があり ます。

何らかのテキストエディタを使用して AppSpec ファイルとシェルスクリプ トを作成すると、デプロイが失敗する場合がある

テキストエディタによっては、不適合で非表示の文字がファイルに含まれる場合があります。 テキストエディタを使用して AppSpec ファイルやシェルスクリプトファイルを作成または変更 し、Amazon Linux、Ubuntu Server、または RHEL インスタンスで実行する場合、これらのファイル に依存するデプロイは失敗する可能性があります。CodeDeploy がこれらのファイルをデプロイ中に 使用したときに、このような文字が存在すると、トラブルシューティングが困難な AppSpec ファイ ルの検証エラーにつながり、スクリプトの実行が失敗する可能性があります。

CodeDeploy コンソールのデプロイのイベント詳細ページで、[ログを表示する] を選択します。 (または、AWS CLI を使用して <u>get-deployment-instance</u> コマンドを呼び出します。) invalid character、command not found、file not foundのようなエラーを探します。

この問題に対処するには、次のことをお勧めします。

- ・ 改行 (^M 文字) などの非表示の文字を AppSpec ファイルとシェルスクリプトファイルに含めるテキストエディタは使用しない。
- AppSpec ファイルやシェルスクリプトファイルで改行など非表示の文字を表示するテキストエディタを使用して、対象となる可能性のある文字を見つけ、削除できるようにする。このようなタイプのテキストエディタの例については、インターネットで「改行を表示するテキストエディタ」を検索します。
- Amazon Linux、Ubuntu サーバー、または RHEL インスタンスで動作するテキストエディタを使用して、Amazon Linux、Ubuntu サーバー、または RHEL インスタンスで動作するシェルスクリプトファイルを作成してください。このようなタイプのテキストエディタの例については、インターネットで「Linux シェルスクリプトエディタ」を検索します。
- Windows または macOS でテキストエディタを使用して、Amazon Linux、Ubuntu Server、また は RHEL インスタンスで実行するシェルスクリプトファイルを作成する場合は、Windows または macOS 形式のテキストを Unix 形式に変換するプログラムまたはユーティリティを使用する。こ のようなプログラムとユーティリティの例については、インターネットで「DOS から UNIX へ」 または「Mac から UNIX へ」を検索します。必ず、ターゲットのオペレーティングシステムで、 変換されたシェルスクリプトファイルをテストします。
macOS の Finder を使用してアプリケーションリビジョンをバンドルする と、デプロイが失敗することがある

Mac の Finder グラフィカルユーザーインタフェース (GUI) アプリケーションを使用して、AppSpec ファイルおよび関連ファイルとスクリプトをアプリケーションリビジョンのアーカイブ (.zip) ファ イルにバンドル (zip) すると、デプロイが失敗する場合があります。これは、Finder が .zip ファ イルに中間の ___MACOSX フォルダを作成し、そこにコンポーネントファイルを配置するためで す。CodeDeploy はコンポーネントファイルを見つけることができないため、デプロイは失敗しま す。

この問題に対処するには、 を使用して push コマンド AWS CLI を呼び出し、コンポーネントファイ ルを期待される構造に圧縮することをお勧めします。または、GUI の代わりにターミナルを使用し てコンポーネントファイルを zip 圧縮できます。ターミナルでは、中間の __MACOSX フォルダは作 成されません。

EC2/オンプレミスのデプロイに関する問題のトラブルシューティ ング

トピック

- <u>CodeDeploy プラグイン CommandPoller の認証情報不足エラー</u>
- •「PKCS7 署名メッセージの検証に失敗しました」というメッセージでデプロイが失敗する
- ・ 同じデプロイ先のインスタンスに対する同じファイルのデプロイや再デプロイは失敗し、「指定したファイルはこの場所に既に存在するため、デプロイに失敗しました」というエラーが返されます。

 す。
- ファイルパスが長いと、「そのようなファイルまたはディレクトリはありません」というエラーが 発生します
- 長時間実行されているプロセスにより、デプロイが失敗することがある
- AllowTraffic ライフサイクルイベントが失敗し、デプロイログにエラーが報告されない場合のトラ ブルシューティング
- <u>失敗した ApplicationStop、BeforeBlockTraffic、または AfterBlockTraffic デプロイライフサイクル</u>
 イベントのトラブルシューティング
- 「不明なエラー: 読み取り用に開いていません」で失敗した DownloadBundle デプロイライフサイ クルイベントのトラブルシューティング
- スキップされたすべてのライフサイクルイベントのエラーをトラブルシューティングする

 <u>Windows PowerShell スクリプトで、デフォルトで 64 ビットバージョンの Windows PowerShell</u> スクリプトを使用できない

Note

多くのデプロイ失敗の原因は、デプロイプロセス中に作成されたログファイルを確認し て特定できます。分かりやすいように、インスタンス別に表示するのではなく、Amazon CloudWatch Logs を使用してログファイルを集中的にモニタリングすることをお勧めしま す。詳細については、「<u>CloudWatch Logs コンソールで CodeDeploy ログを見る</u>」を参照し てください。

🚺 Tip

EC2/オンプレミスデプロイに関連する多くのトラブルシューティングタスクを自動化する ランブックについては、「AWS Systems Manager オートメーションランブックリファレン ス」の「<u>AWSSupport-TroubleshootCodeDeploy</u>」を参照してください。

CodeDeploy プラグイン CommandPoller の認証情報不足エラー

InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profileのようなエラーが表示された場合は、次のいずれがが原因の可能性があります。

デプロイ先のインスタンスに、関連付けられている IAM インスタンスプロファイルがない。

• IAM インスタンスプロファイルに、適切なアクセス許可が設定されていない。

IAM インスタンスプロファイルによって、CodeDeploy と通信し、リビジョンを Amazon S3 からダ ウンロードするためのアクセス許可が CodeDeploy エージェントに付与されている。EC2 インスタ ンスの場合は、「<u>AWS CodeDeployのためのアイデンティティおよびアクセス管理</u>」を参照してく ださい。オンプレミスインスタンスの場合は、<u>Working with On-Premises Instances</u> を参照してくだ さい。

「PKCS7 署名メッセージの検証に失敗しました」というメッセージでデプ ロイが失敗する

このエラーメッセージは、SHA-1 ハッシュアルゴリズムのみをサポートするバージョンの CodeDeploy エージェントをインスタンスが実行していることを示します。SHA-2 ハッシュアルゴリ ズムのサポートは、2015 年 11 月にリリースされたバージョン 1.0.1.854 の CodeDeploy エージェン トで導入されました。2016 年 10 月 17 日から、バージョン 1.0.1.854 以前の CodeDeploy エージェ ントがインストールされている場合、デプロイは失敗します。詳細については、「<u>AWS SSL 証明書</u> <u>のハッシュアルゴリズムを SHA256 に変更すること</u>」、「<u>NOTICE: バージョン 1.0.1.85 より古い</u> <u>CodeDeploy ホストエージェントを使用停止にする</u>」、および「<u>CodeDeploy エージェントを更新す</u> る」を参照してください。

同じデプロイ先のインスタンスに対する同じファイルのデプロイや再デプ ロイは失敗し、「指定したファイルはこの場所に既に存在するため、デプ ロイに失敗しました」というエラーが返されます。

CodeDeploy でファイルをデプロイしようとして、指定したデプロイ先のインスタンスに同じファ イルが既に存在している場合、そのインスタンスへのデプロイは失敗する場合があります。「指定 したファイルはこの場所 (*location-name*) に既に存在しているため、デプロイに失敗しました」 というエラーメッセージが返される場合があります。このエラーが発生するのは、デプロイごとに CodeDeploy では最初にクリーンアップログファイルにリストされているすべてのファイルを以前の デプロイから削除するためです。このクリーンアップファイルにリストされていないファイルがデプ ロイ先のフォルダにあると、CodeDeploy エージェントでは、デフォルトでこれをエラーと解釈し、 デプロイを失敗させます。

Note

Amazon Linux、RHEL、および Ubuntu Server の各インスタンスでは、クリーンアップファ イルは /opt/codedeploy-agent/deployment-root/deployment-instructions/ にあります。Windows Server インスタンスでは、場所は C:\ProgramData\Amazon \CodeDeploy\deployment-instructions\ です。

このエラーを最も簡単に回避するには、デプロイを失敗させるデフォルト動作とは別のオプションを 指定します。デプロイごとに、デプロイを失敗させるか、クリーンアップファイルにリストされてい ないファイルを上書きするか、インスタンスの既存ファイルを保持するかを選択できます。 上書きオプションは、最後のデプロイ後に手動でインスタンスに追加した同じ名前のファイルを、次 回のアプリケーションリビジョンにも追加する場合などに便利です。

保持オプションは、次回のデプロイでインスタンスに追加するファイルを、アプリケーションリビ ジョンパッケージには追加する必要がない場合に選択できます。保持オプションは、アプリケーショ ンファイルが既に本番稼働環境にあり、これらを CodeDeploy を使用して初めてデプロイする場合 にも便利です。詳細については、<u>EC2/オンプレミスコンピューティングプラットフォームのデプロ</u> イ作成 (コンソール)および既存のコンテンツでのロールバック動作を参照してください。

The deployment failed because a specified file already exists at this location デプロイメント失敗のトラブルシューティング

CodeDeploy がデプロイ先で検出したコンテンツを上書きまたは保持するオプションを指定しない場合 (または既存のコンテンツを処理するデプロイオプションをプログラムによるコマンドで指定しない場合)、エラーのトラブルシューティングを行うことを選択できます。

次の情報は、コンテンツを保持または上書きしないことを選択した場合にのみ該当します。

同じ名前と場所のファイルの再デプロイを試みる場合、以前に使用した基になるデプロイグルー プ ID のアプリケーション名およびデプロイグループを指定すると、成功する可能性が高まりま す。CodeDeploy は、基になるデプロイグループ ID を使用して、再デプロイする前に削除するファ イルを識別します。

新しいファイルのデプロイや、インスタンスの同じ場所への同じファイルの再デプロイは、次の理由 により失敗することがあります。

- 同じリビジョンの同じインスタンスへの再デプロイのため、別のアプリケーション名を指定した。
 デプロイグループ名が同じでも、別のアプリケーション名を使用すると、基になる別のデプロイグ
 ループ ID が使用されるため、再デプロイは失敗します。
- アプリケーションのデプロイグループを削除して再作成し、同じリビジョンをデプロイグループに 対して再デプロイしようとした。デプロイグループ名が同じでも、CodeDeploy は基になるデプロ イグループ ID を参照するため、再デプロイは失敗します。
- CodeDeployのアプリケーションとデプロイグループを削除し、削除したものと同じ名前の新しい アプリケーションとデプロイグループを作成した。その後、以前のデプロイグループにデプロイ されていたリビジョンを、同じ名前の新しいデプロイグループに再デプロイしようとした。アプリ ケーション名とデプロイグループ名が同じであっても、CodeDeployは削除したデプロイグループ の ID を引き続き参照するため、再デプロイは失敗します。

- リビジョンをデプロイグループにデプロイし、同じリビジョンを同じインスタンスの別のデプロ イグループにデプロイした。CodeDeployは別の基になるデプロイグループ ID を参照するため、2 番目のデプロイは失敗します。
- リビジョンを1つのデプロイグループにデプロイし、別のリビジョンを同じインスタンスの別の デプロイグループにデプロイした。2番目のデプロイグループがデプロイを試みる同じ名前と場所 のファイルが、少なくとも1つある。CodeDeployは2番目のデプロイが開始される前に既存の ファイルを削除しないため、2番目のデプロイは失敗します。両方のデプロイは、異なるデプロイ グループ ID を参照します。
- CodeDeploy でリビジョンデプロイしたが、同じ名前と場所のファイルが少なくとも1つある。デフォルトでは、CodeDeploy はデプロイが開始される前に既存のファイルを削除しないため、デプロイは失敗します。

これらの状況に対応するには、次のいずれかを実行します。

- 以前のデプロイされた場所とインスタンスからファイルを削除してから、もう一度デプロイを試み ます。
- リビジョンの AppSpec ファイルで、ApplicationStop または BeforeInstall デプロイライフサイクル イベントのいずれかにおいて、リビジョンでインストールするファイルと一致するファイルがいず れかの場所にある場合、これらを削除するためのカスタムスクリプトを指定します。
- 以前のデプロイの一部ではない場所またはインスタンスにファイルをデプロイまたは再デプロイします。
- アプリケーションまたはデプロイグループを削除する前に、インスタンスにコピーするファイル を指定しない AppSpec ファイル を含むリビジョンをデプロイします。このデプロイの場合、削除 しようとしているものと同じ、基になるアプリケーションおよびデプロイグループ ID を使用する アプリケーション名とデプロイグループ名を指定します (get-deployment-group コマンドで、デプ ロイグループ ID を取得することができます。) CodeDeploy は、基になるデプロイグループ ID と AppSpec ファイルを使用して、前回成功したデプロイでインストールしたすべてのファイルを削 除します。

ファイルパスが長いと、「そのようなファイルまたはディレクトリはあり ません」というエラーが発生します

Windows インスタンスへのデプロイでは、appspec.yml ファイルのファイルセクションに 260 文字 を超えるファイルパスがあると、デプロイが次のようなエラーで失敗することがあります。 No such file or directory @ dir_s_mkdir - C:\your-long-file-path

このエラーは、<u>Microsoft のドキュメント</u>に詳述されているように、Windows ではデフォルトで 260 文字を超えるファイルパスが許可されていないために発生します。

CodeDeploy エージェントバージョン 1.4.0 以降では、エージェントのインストールプロセスに応じ て、次の 2 つの方法で長いファイルパスを有効にできます。

CodeDeploy エージェントがまだインストールされていない場合:

1. CodeDeploy エージェントをインストールする予定のマシンで、次のコマンドを使用して LongPathsEnabled Windows レジストリキーを有効にします。

New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem" -Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force

2. CodeDeploy エージェントをインストールします。詳細については、「<u>CodeDeploy エージェント</u> をインストールする」を参照してください。

CodeDeploy エージェントがすでにインストールされている場合:

1. CodeDeploy エージェントマシンで、次のコマンドを使用して LongPathsEnabled Windows レ ジストリキーを有効にします。

New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force

2. CodeDeploy エージェントを再起動して、レジストリキーの変更を有効にします。エージェントを 再起動するには、次のコマンドを使用します。

powershell.exe -Command Restart-Service -Name codedeployagent

長時間実行されているプロセスにより、デプロイが失敗することがある

Amazon Linux、Ubuntu Server、RHEL インスタンスへのデプロイでは、長時間実行されるプロセス を開始するデプロイスクリプトがある場合、CodeDeploy はデプロイライフサイクルイベントでの待 機に長い時間がかかり、その後デプロイに失敗することがあります。これは、そのイベントのフォア グラウンドおよびバックグラウンドで予期されるよりも長くプロセスが実行される場合、プロセスが まだ予期どおり実行されていても、CodeDeploy によってデプロイは中止され、失敗するためです。 たとえば、アプリケーションリビジョンのルートに 2 つのファイル (after-install.sh および sleep.sh) が含まれているとします。その AppSpec ファイルファイル には次の指示が含まれてい ます。

```
version: 0.0
os: linux
files:
   - source: ./sleep.sh
    destination: /tmp
hooks:
   AfterInstall:
        - location: after-install.sh
        timeout: 60
```

after-install.sh ファイルは AfterInstall アプリケーションライフサイクルイベント中に実行さ れます。そのコンテンツは次のとおりです。

#!/bin/bash
/tmp/sleep.sh

sleep.sh ファイルには以下が含まれます。これはプログラムの実行を 3 分 (180 秒) 停止し、長時 間実行されるプロセスをシミュレートします。

#!/bin/bash
sleep 180

after-install.sh が sleep.sh を呼び出すと、sleep.sh が起動し、3 分 (180 秒) 実行しま す。これは CodeDeploy が sleep.sh (および関連で after-install.sh) の実行を停止させると 予期した時間を 2 分 (120 秒) 経過した時間です。1 分 (60 秒) のタイムアウト後、sleep.sh は予期 どおり実行を継続しますが、CodeDeploy は停止し、AfterInstall アプリケーションライフサイクルイ ベントでデプロイに失敗します。次のエラーが表示されます。

Script at specified location: after-install.sh failed to complete in 60 seconds.

単純に & でアンパサンド (after-install.sh) を追加して、バックグラウンドで sleep.sh を実 行することはできません。

```
#!/bin/bash
```

Do not do this.
/tmp/sleep.sh &

この操作を行うと、デプロイは最大でデフォルトの 1 時間のデプロイライフサイクルイベントのタ イムアウト期間まで保留状態になり、その後 CodeDeploy は以前のように AfterInstall アプリケー ションライフサイクルイベントでデプロイを停止し、失敗させます。

after-install.sh で、次のように sleep.sh を呼び出します。これにより、CodeDeploy はプ ロセスの実行開始後に続行できます。

#!/bin/bash
/tmp/sleep.sh > /dev/null 2> /dev/null < /dev/null &</pre>

前の呼び出しで、sleep.sh はバックグラウンドで実行を開始し、stdout、stderr、および stdin を / dev/null にリダイレクトするプロセスの名前です。

AllowTraffic ライフサイクルイベントが失敗し、デプロイログにエラーが報 告されない場合のトラブルシューティング

AllowTraffic ライフサイクルイベント中に Blue/Green デプロイが失敗しても、デプロイログに失敗 の原因が表示されない場合があります。

通常、この障害は、デプロイグループへのトラフィック管理に使用される Classic Load Balancer、Application Load Balancer、または Network Load Balancer の Elastic Load Balancing の ヘルスチェックが間違って設定されていることが原因です。

この問題を解決するには、ロードバランサーのヘルスチェックの設定エラーを確認して修正します。

Classic Load Balancer については、「Classic Load Balancer のユーザーガイド」の「<u>Configure</u> <u>health checks for your Classic Load Balancer</u> (Classic Load Balancer のヘルスチェックの 設定)」、および「Elastic Load Balancing API リファレンスバージョン 2012-06-01」の 「ConfigureHealthCheck」を参照してください。

Application Load Balancer については、「Application Load Balancer のユーザーガイド」の「<u>ター</u> <u>ゲットグループのヘルスチェック</u>」を参照してください。

Network Load Balancer については、Network Load Balancer ユーザーガイド の「<u>ターゲットグルー</u> プのヘルスチェック」を参照してください。 失敗した ApplicationStop、BeforeBlockTraffic、または AfterBlockTraffic デ プロイライフサイクルイベントのトラブルシューティング

デプロイ時に、CodeDeploy エージェントは ApplicationStop、BeforeBlockTraffic、および AfterBlockTraffic に対して前回の正常なデプロイで AppSpec ファイルに指定されていたスクリプト を実行します。(他のすべてのスクリプトは、現在のデプロイの AppSpec ファイルから実行されま す)。これらのスクリプトのいずれかにエラーがあって正常に実行されない場合、デプロイに失敗す ることがあります。

これらの失敗の原因としては以下のようなことが考えられます。

 CodeDeploy エージェントは *deployment-group-id*_last_successful_install ファイル を正しい場所で見つけたが、*deployment-group-id*_last_successful_install ファイル にリストされている場所が存在しない。

Amazon Linux、Ubuntu サーバー、および RHEL インスタンス では、このファイルは /opt/ codedeploy-agent/deployment-root/deployment-instructions に存在する必要があり ます。

Windows Server インスタンスでは、このファイルは C:\ProgramData\Amazon\CodeDeploy \deployment-instructions フォルダに保存されている必要があります。

- *deployment-group-id*_last_successful_install ファイルに示された場所で、AppSpec ファイルが無効であるか、スクリプトが正常に実行されない。
- スクリプトに修正できないエラーがあるため、正常に実行されることはありません。

CodeDeploy コンソールを使用して、これらのいずれかのイベント中にデプロイが失敗した理由を 調査します。デプロイの詳細ページで、[View events] を選択します。インスタンスの詳細ページの [ApplicationStop] 行、[BeforeBlockTraffic] 行、または [AfterBlockTraffic] 行で、[ログを表示する] を 選択します。または AWS CLI 、 を使用して get-deployment-instance コマンドを呼び出します。

前回の正常なデプロイのスクリプトが正常に実行されないために失敗した場合は、デプロイを作成し て [ApplicationStop]、[BeforeBlockTraffic]、および [AfterBlockTraffic] のエラーを無視するように指定 します。これには、以下の 2 つの方法があります。

 CodeDeploy コンソールを使用してデプロイを作成します。[デプロイの作成] ページの [ApplicationStop ライフサイクルイベントの障害] で、[このインスタンス上のライフサイクルイベントの障害で、デプロイを失敗させない] を選択します。 を使用して <u>create-deployment</u> コマンドを AWS CLI 呼び出し、 --ignore-applicationstop-failuresオプションを含めます。

アプリケーションリビジョンを再度デプロイすると、これら3つのライフサイクルイベントのいず れが失敗してもデプロイは続行されます。これらのライフサイクルイベントの修正済みスクリプトが 新しいリビジョンに含まれている場合は、この修正を適用しなくても以降のデプロイは正常に実行さ れます。

「不明なエラー: 読み取り用に開いていません」で失敗した DownloadBundle デプロイライフサイクルイベントのトラブルシューティ ング

Amazon S3 からアプリケーションリビジョンをデプロイしようとして、DownloadBundle デプロイ ライフサイクルイベント中に「UnknownError: not opened for reading」というエラーでデ プロイが失敗する場合:

- Amazon S3 の内部サーバーエラーが発生しました。アプリケーションリビジョンをもう一度デプ ロイします。
- EC2 インスタンスの IAM インスタンスプロファイルに、Amazon S3 のアプリケーションリビジョ ンにアクセスするアクセス許可がありません。Amazon S3 バケットポリシーの詳細については、 「<u>Amazon S3 に CodeDeploy のリビジョンをプッシュする (EC2/オンプレミスのデプロイのみ)</u>」 と「デプロイの前提条件」を参照してください。
- デプロイ先のインスタンスは1つの AWS リージョン (米国西部 (オレゴン) など) に関連付けられていますが、アプリケーションリビジョンを含む Amazon S3 バケットは別の AWS リージョン (米国東部 (バージニア北部) など) に関連付けられています。アプリケーションリビジョンがインスタンスと同じ AWS リージョンに関連付けられている Amazon S3 バケットにあることを確認します。

デプロイのイベント詳細ページの [Download bundle (バンドルのダウンロード)] 行で、[ログを表示する] を選択します。または AWS CLI 、 を使用して <u>get-deployment-instance</u> コマンドを呼び出します。このエラーが発生した場合、エラーコード「UnknownError」とエラーメッセージ「not opened for reading」のエラーが出力に表示されます。

このエラーの原因を特定するには:

- インスタンスの少なくとも1つでワイヤログを有効にして、もう一度アプリケーションリビジョンをデプロイします。
- 2. ワイヤログファイルを調べてエラーを見つけます。この問題の一般的なエラーメッセージには、 「access denied」という語句が含まれます。
- ログファイルを確認した後、ワイヤログを無効にして、ログファイルサイズと、今後インスタン スでプレーンテキストで出力に表示される可能性がある機密情報の量を減らすことをお勧めしま す。

ワイヤログファイルを探し、ワイヤログを有効または無効にする方法については、「<u>CodeDeploy</u> エージェント構成リファレンス」の「:log_aws_wire:」を参照してください。

スキップされたすべてのライフサイクルイベントのエラーをトラブル シューティングする

EC2 またはオンプレミスのデプロイのライフサイクルイベントがすべてスキップされた場合 は、The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems. (Error code: HEALTH_CONSTRAINTS)のようなエラーが表示されます。考えられる原因と解決 策は次のとおりです。

- CodeDeploy エージェントがインスタンスにインストールされていないか、インスタンスで実行されていない。CodeDeploy エージェントが実行されていることを確認するには:
 - Amazon Linux RHEL または Ubuntu server の場合は、以下を実行します。

systemctl status codedeploy-agent

Windows の場合は、以下を実行します。

powershell.exe -Command Get-Service -Name CodeDeployagent

CodeDeploy エージェントがインストールまたは実行されていない場合は、<u>CodeDeploy エージェ</u> <u>ントが実行されていることの確認</u> を参照してください。

インスタンスがポート 443 を使用して CodeDeploy または Amazon S3 のパブリックエンドポイン トに到達できない可能性があります。以下のいずれかを行ってください。

- インスタンスにパブリック IP アドレスを割り当て、そのルートテーブルを使用してインター ネットアクセスを許可します。インスタンスに関連付けられているセキュリティグループで、 ポート 443 (HTTPS) 経由で送信アクセスが許可されていることを確認してください。詳細につ いては、「CodeDeploy エージェントの通信プロトコルとポート」を参照してください。
- インスタンスがプライベートサブネットにプロビジョニングされている場合は、ルートテーブルで、インターネットゲートウェイではなく NAT ゲートウェイを使用します。詳細については、「NAT ゲートウェイ」を参照してください。
- CodeDeploy のサービスロールに、必要なアクセス許可が付与されていない可能性があります。CodeDeploy サービスロールを設定するには、「ステップ 2: CodeDeployのサービスのロール を作成する」を参照してください。
- HTTP プロキシを使用している場合は、CodeDeploy エージェントの設定ファイルの :proxy_uri:設定で指定されていることを確認してください。詳細については、「<u>CodeDeploy</u> <u>エージェント設定リファレンス</u>」を参照してください。
- デプロイメントインスタンスの日時が、デプロイメントリクエストの日時と一致しない場合が あります。CodeDeploy エージェントログファイルに、Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature expired のようなエラーがないか確認します。このエラーが表示されている場合 は、「InvalidSignatureException - Signature expired: [time] is now earlier than [time]」デプロイエ ラーのトラブルシューティングのステップに従います。詳細については、「CodeDeploy EC2/オ ンプレミスデプロイのログデータの表示」を参照してください。
- インスタンスのメモリまたはハードディスクの空き容量が不足しているため、CodeDeploy エージェントが停止している可能性があります。インスタンス上のアーカイブされたデプロイメントの数が減るように、CodeDeploy エージェントの max_revisions 設定を更新してください。EC2 インスタンスにこのプロセスを行っても問題が解決しない場合は、インスタンスのサイズを大きくすることを検討してください。たとえば、インスタンスタイプが t2.small の場合は、t2.medium の使用を検討します。詳細については、「CodeDeploy エージェントでインストールされるファイル」、「CodeDeploy エージェント設定リファレンス」、「インスタンスタイプ」を参照してください。
- デプロイ先のインスタンスに、IAM インスタンスプロファイルがアタッチされていないか、また は必要なアクセス許可が付与されていない IAM インスタンスプロファイルがアタッチされている 可能性があります。
 - IAM インスタンスプロファイルがインスタンスにアタッチされていない場合は、必要なアクセス許可を付与したインスタンスプロファイルを作成し、アタッチします。
 - IAM インスタンスプロファイルがインスタンスにすでにアタッチされている場合は、必要なアクセス許可があることを確認します。

必要なアクセス許可が付与されているインスタンスプロファイルがアタッチされていることを 確認したら、インスタンスを再起動します。詳細については、Amazon EC2 ユーザーガイド の 「<u>ステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成する</u>」と 「Amazon EC2 の IAM ロール」を参照してください。

Windows PowerShell スクリプトで、デフォルトで 64 ビットバージョンの Windows PowerShell スクリプトを使用できない

デプロイの一部として実行中の Windows PowerShell スクリプトが 64 ビットの機能に依存している 場合 (たとえば、32 ビットアプリケーションで許可されるよりも多くのメモリを消費する、64 ビッ トバージョンのみで提供されるライブラリを呼び出すなど)、スクリプトはクラッシュするか、予 期どおりに実行されない可能性があります。これは、CodeDeploy はデフォルトで 32 ビットバー ジョンの Windows PowerShell を使用して、アプリケーションリビジョンの一部である Windows PowerShell スクリプトを実行するためです。

64 ビットバージョンの Windows PowerShell で実行する必要があるスクリプトの先頭に、次のよう なコードを追加します。

```
# Are you running in 32-bit mode?
    (SysWOW64) = 32-bit mode)
#
if ($PSHOME -like "*SysWOW64*")
{
  Write-Warning "Restarting this script under 64-bit Windows PowerShell."
  # Restart this script under 64-bit Windows PowerShell.
      (\SysNative\ redirects to \System32\ for 64-bit mode)
  #
  & (Join-Path ($PSHOME -replace "SysWOW64", "SysNative") powershell.exe) -File `
    (Join-Path $PSScriptRoot $MyInvocation.MyCommand) @args
  # Exit 32-bit script.
  Exit $LastExitCode
}
# Was restart successful?
Write-Warning "Hello from $PSHOME"
Write-Warning " (\SysWOW64 = 32-bit mode, \System32 = 64-bit mode)"
```

このコードのファイルパス情報を直観的にはわかりにくいかもしれませんが、32 ビットの Windows PowerShell では次のようなパスが使用されます。

c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

64 ビットの Windows PowerShell では次のようなパスが使用されます。

c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Amazon ECS のデプロイに関する問題のトラブルシューティング

トピック

- 置き換えタスクセットを待っている間にタイムアウトが発生します
- 通知が継続されるのを待っている間のタイムアウトの発生
- IAM ロールに十分なアクセス許可がありません
- ステータスコールバックを待っている間に、デプロイがタイムアウトした
- 1 つ以上のライフサイクルイベントの検証機能が失敗したため、デプロイが失敗しました
- 「プライマリタスクセットのターゲットグループはリスナーの後ろにある必要があります」という エラーのため、ELBを更新できませんでした
- Auto Scaling を使用するとデプロイが失敗することがあります
- <u>ALB のみが段階的なトラフィックルーティングをサポートしているため、デプロイグループの作</u>成/更新時には、代わりに AllAtOnce トラフィックルーティングを使用してください
- デプロイが成功しても、置き換えタスクセットは Elastic Load Balancing のヘルスチェックに失敗
 し、アプリケーションがダウンしています
- <u>1つのデプロイグループに複数のロードバランサーをアタッチできますか?</u>
- ・ <u>ロードバランサーなしで CodeDeploy のブルー/グリーンデプロイを実行できますか?</u>
- ・デプロイ中に Amazon ECS サービスを新しい情報で更新する方法を教えてください。

置き換えタスクセットを待っている間にタイムアウトが発生します

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイする際に、次のエラーメッ セージが表示される。

The deployment timed out while waiting for the replacement task set to become healthy. This time out period is 60 minutes.

考えられる原因: このエラーは、タスク定義ファイルまたはその他のデプロイ関連ファイルに誤りが ある場合に発生する可能性があります。例えば、タスク定義ファイルの image フィールドにタイプ ミスがあると、Amazon ECS は間違ったコンテナイメージを取得しようとして失敗し続けるため、 このエラーが発生します。

解決方法と次のステップ:

- タスク定義ファイルやその他のファイルの入力ミスや設定の問題を修正します。
- ・ 関連する Amazon ECS サービスイベントをチェックして、置き換えタスクが正常に動作しない 理由を調べます。Amazon ECS イベントに関する詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS イベント」を参照してください。
- Amazon Elastic Container Service デベロッパーガイドの「<u>Amazon ECS トラブルシューティン</u> <u>グ</u>」セクションで、イベント内のメッセージに関連するエラーについて参照してください。

通知が継続されるのを待っている間のタイムアウトの発生

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイする際に、次のエラーメッ セージが表示される。

The deployment timed out while waiting for a notification to continue. This time out period is n minutes.

考えられる原因: このエラーは、デプロイグループの作成時に [トラフィックを再ルーティングする タイミングを指定します] フィールドで待機時間を指定したものの、待機時間が経過する前にデプロ イを完了できなかった場合に発生する可能性があります。

解決方法と次のステップ:

デプロイグループで、[トラフィックを再ルーティングするタイミングを指定します] をより長い時間に設定して再デプロイします。詳細については、「<u>Amazon ECS デプロイ用のデプロイグルー</u>プを作成する (コンソール)」を参照してください。

- デプロイグループで、[トラフィックを再ルーティングするタイミングを指定します] を [すぐにト ラフィックを再ルーティング] に変更して、再デプロイします。詳細については、「<u>Amazon ECS</u> デプロイ用のデプロイグループを作成する (コンソール)」を参照してください。
- --deployment-wait-type オプションを に設定して<u>aws_deploy_continue-deployment</u> AWS CLI、 コマンドを再デプロイしてから実行しますREADY_WAIT。[トラフィックを再ルーティ ングするタイミングを指定します] で指定した時間が経過する前に、このコマンドを必ず実行して ください。

IAM ロールに十分なアクセス許可がありません

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイしている際に、次のエラー メッセージが表示されます。

The IAM role *role-arn* does not give you permission to perform operations in the following AWS service: AWSLambda.

考えられる原因: このエラーは、<u>AppSpec ファイルの Hooks セクション</u>で Lambda 関数を指定した が、Lambda サービスに CodeDeploy アクセス許可を付与しなかった場合に発生する可能性がありま す。

解決方法: CodeDeploy サービスロールに 1ambda: InvokeFunction アクセス許可 を追加します。このアクセス許可を追加するには、AWSCodeDeployRoleForECS か AWSCodeDeployRoleForECSLimited の AWSマネージドポリシーのいずれかをロールに追加し ます。これらのポリシーと、ポリシーを CodeDeploy サービスロールに追加する方法については、 「<u>ステップ 2: CodeDeployのサービスのロールを作成する</u>」を参照してください。

ステータスコールバックを待っている間に、デプロイがタイムアウトした

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイしている際に、次のエラー メッセージが表示されます。

The deployment timed out while waiting for a status callback. CodeDeploy expects a status callback within one hour after a deployment hook is invoked.

考えられる原因: このエラーは、<u>AppSpec ファイルの Hooks セクション</u>で Lambda 関数を指定し たが、Lambda 関数が CodeDeploy に Succeeded または Failed ステータスを返すのに必要な PutLifecycleEventHookExecutionStatus API を呼び出せなかった場合に発生する可能性があ ります。 解決方法と次のステップ:

- AppSpec ファイルで指定した Lambda 関数が使用する Lambda 実行ロールに codedeploy:putlifecycleEventHookExecutionStatus アクセス許可を追加します。この アクセス許可により、Lambda 関数には CodeDeploy に Succeeded または Failed のステータ スを返すアクセス許可が付与されます。Lambda 実行ロールの詳細については、「AWS Lambda ユーザーガイド」の「Lambda 実行ロール」を参照してください。
- Lambda 関数のコードと実行ログを確認して、Lambda 関数が CodeDeploy の PutLifecycleEventHookExecutionStatus API を呼び出して、ライフサイクル検証 テストが Succeeded か Failed であるかを CodeDeploy に通知していることを確認しま す。putlifecycleEventHookExecutionStatus API の詳細については、「AWS CodeDeploy API リファレンス」の「<u>PutLifecycleEventHookExecutionStatus</u>」を参照してください。Lambda 実行ログの詳細については、「<u>AWS Lambdaでの Amazon CloudWatch Logs の使用</u>」を参照して ください。

1 つ以上のライフサイクルイベントの検証機能が失敗したため、デプロイ が失敗しました

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイしている際に、次のエラー メッセージが表示されます。

The deployment failed because one or more of the lifecycle event validation functions failed.

考えられる原因: このエラーは、<u>AppSpec ファイルの Hooks セクション</u>で Lambda 関数を指定した が、Lambda 関数が PutLifecycleEventHookExecutionStatus を呼び出した際に CodeDeploy に Failed を返した場合に発生する可能性があります。この失敗は、ライフサイクル検証テストが 失敗したことを CodeDeploy に示します。

考えられる次のステップ: Lambda 実行ログを確認して、検証テストコードが失敗している理由を確 認します。Lambda 実行ログの詳細については、「<u>AWS Lambdaでの Amazon CloudWatch Logs の</u> <u>使用</u>」を参照してください。

「プライマリタスクセットのターゲットグループはリスナーの後ろにある 必要があります」というエラーのため、ELB を更新できませんでした

問題: CodeDeploy を使用して Amazon ECS アプリケーションをデプロイしている際に、次のエラー メッセージが表示されます。 The ELB could not be updated due to the following error: Primary taskset target group must be behind listener

考えられる原因:このエラーは、オプションのテストリスナーを設定しており、そのリスナーに間 違ったターゲットグループが設定されている場合に発生する可能性があります。CodeDeploy のテス トリスナーの詳細については、「<u>Amazon ECS デプロイを開始する前に</u>」と「<u>Amazon ECS デプロ</u> <u>イ中の処理で起こっていること</u>」を参照してください。タスクセットの詳細については、「Amazon Elastic Container Service API リファレンス」の「<u>TaskSet</u>」と、「AWS CLI コマンドリファレン ス」の「Amazon ECS」セクションの「describe-task-set」を参照してください。

考えられる解決方法: Elastic Load Balancing の本稼働リスナーとテストリスナーの両方が、現在ワー クロードを処理しているターゲットグループを指していることを確認します。確認すべき箇所は 3 つあります。

- Amazon EC2 のロードバランサーの「リスナーとルール」の設定。詳細については、 「Application Load Balancer のユーザーガイド」の「<u>Application Load Balancer のリスナー</u>」、ま たは「Network Load Balancer のユーザーガイド」の「<u>Network Load Balancer のリスナー</u>」を参 照してください。
- Amazon ECS のクラスター内のサービスのネットワーク設定。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「<u>Application Load Balancerおよび Network</u> Load Balancerの場合」を参照してください。
- CodeDeploy のデプロイグループ設定。詳細については、「<u>Amazon ECS デプロイ用のデプロイ</u> グループを作成する (コンソール)」を参照してください。

Auto Scaling を使用するとデプロイが失敗することがあります

問題: CodeDeploy で Auto Scaling を使用中、デプロイが失敗することがあることに気付きました。 この問題の症状の詳細については、「Amazon Elastic Container Service デベロッパーガイド」の 「<u>サービスの自動スケーリングと ブルー/グリーンデプロイタイプを使用するように設定されたサー</u> ビスでは、自動スケーリングはデプロイ中にブロックされませんが、状況によってはデプロイが失敗 する場合があります」というトピックを参照してください。

考えられる原因: この問題は、CodeDeploy プロセスと Auto Scaling プロセスが競合する場合に発生 する可能性があります。

解決方法: RegisterScalableTarget API (または対応するregister-scalable-target AWS CLI コマンド) を使用してCodeDeploy デプロイ中に Auto Scaling プロセスを一時停止および再開し

ます。詳細については、「Application Auto Scaling ユーザーガイド」の「<u>Application Auto Scaling</u> のスケーリングの一時停止と再開」を参照してください。

Note

CodeDeploy は RegisterScaleableTarget を直接呼び出すことはできません。この API を使用するには、Amazon Simple Notification Service (または Amazon CloudWatch) に通知またはイベントを送信するように CodeDeploy を設定する必要があります。 次に、Lambda 関数を呼び出すように Amazon SNS (または CloudWatch) を設定 し、RegisterScalableTarget API を呼び出すように Lambda 関数を設定する必要があ ります。Auto Scaling オペレーションを一時停止するには SuspendedState パラメータ をtrue に設定し、再開するには false に設定し、RegisterScalableTarget API を呼 び出す必要があります。

時停止オペレーションをトリガーする場合)、またはデプロイが成功したとき、失敗したと き、または停止したとき (Auto Scaling 再開オペレーションをトリガーする場合) に発生する 必要があります。

Amazon SNS 通知または CloudWatch イベントを生成するように CodeDeploy を設定する 方法については、「<u>Amazon CloudWatch Events を使用したデプロイのモニタリング</u>」と 「Monitoring Deployments with Amazon SNS Event Notifications」を参照してください。

ALB のみが段階的なトラフィックルーティングをサポートしているため、 デプロイグループの作成/更新時には、代わりに AllAtOnce トラフィック ルーティングを使用してください

問題: CodeDeploy でデプロイグループを作成または更新している際に、次のエラーメッセージが表 示されます。

Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing instead when you create/update Deployment group.

考えられる原因: このエラーは Network Load Balancer を使用してい

て、CodeDeployDefault.ECSAllAtOnce 以外の事前定義されたデプロイ設定を使用しようとし た場合に発生する可能性があります。

解決方法:

 事前定義されたデプロイ設定を CodeDeployDefault.ECSAllAtOnce に変更します。これは Network Load Balancer がサポートする唯一の事前定義されたデプロイ設定です。

事前定義されたデプロイ設定の詳細については、「<u>Amazon ECS コンピューティングプラット</u> フォームの事前定義されたデプロイ設定」を参照してください。

ロードバランサーを Application Load Balancer に変更します。Application Load Balancer は、事前定義されたデプロイ設定をすべてサポートします。Application Load Balancer の作成の詳細については、「CodeDeploy Amazon ECS デプロイ用のロードバランサー、ターゲットグループ、リスナーをセットアップする」を参照してください。

デプロイが成功しても、置き換えタスクセットは Elastic Load Balancing の ヘルスチェックに失敗し、アプリケーションがダウンしています

問題: CodeDeploy ではデプロイが成功したと表示されているのに、置き換えタスクセットは Elastic Load Balancing からのヘルスチェックに失敗し、アプリケーションがダウンしています。

考えられる原因: この問題は、CodeDeploy 一括デプロイを実行した際に、置き換え (グリーン) タス クセットに Elastic Load Balancing ヘルスチェックが失敗する原因となる誤ったコードが含まれてい る場合に発生する可能性があります。1 回にすべてのデプロイ設定では、トラフィックが置き換えタ スクセットに移行した後 (つまり、CodeDeploy の AllowTraffic ライフサイクルイベントが発生 した後)、ロードバランサーのヘルスチェックが置き換えタスクセットで実行され始めます。そのた め、トラフィックが移行した後は置き換えタスクセットでヘルスチェックが失敗しますが、それ以 前には失敗しません。CodeDeploy が生成するライフサイクルイベントについては、「<u>Amazon ECS</u> デプロイ中の処理で起こっていること」を参照してください。

解決方法:

デプロイ設定を1回にすべてから canary またはリニアに変更します。canary 設定またはリニア設定では、CodeDeploy が置き換え先環境にアプリケーションをインストールしている間、およびトラフィックが移行される前(つまり、Install ライフサイクルイベント中および AllowTraffic イベント前)に、ロードバランサーのヘルスチェックが置き換えタスクセットで実行を開始します。アプリケーションのインストール中、トラフィックが移行する前にチェックを実行できるようにすることで、アプリケーションが一般公開される前に誤ったアプリケーションコードが検出され、デプロイが失敗します。

canary デプロイまたはリニアデプロイを設定する方法については、「<u>CodeDeploy を使用して、</u> デプロイグループの設定を変更します。」を参照してください。 Amazon ECS のデプロイ中に実行される CodeDeploy ライフサイクルイベントの詳細について は、「Amazon ECS デプロイ中の処理で起こっていること」を参照してください。

Note

canary デプロイ設定およびリニアデプロイ設定は、Application Load Balancer でのみサ ポートされます。

1回にすべてデプロイ設定を維持したい場合は、テストリスナーをセットアップし、BeforeAllowTraffic ライフサイクルフックを使用して置き換えタスクセットのヘルスステータスを確認します。詳細については、「<u>Amazon ECS のデプロイ向けのライフサイクルイベントフックのリスト</u>」を参照してください。

1つのデプロイグループに複数のロードバランサーをアタッチできますか?

いいえ。複数の Application Load Balancer または Network Load Balancer を使用する場合 は、CodeDeploy ブルー/グリーンデプロイの代わりに Amazon ECS ローリング更新を使用してく ださい。ローリング更新の詳細については、「Amazon Elastic Container Service デベロッパーガイ ド」の「<u>ローリング更新</u>」を参照してください。Amazon ECS で複数のロードバランサーを使用す る方法の詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「<u>サービス</u> に複数のターゲットグループを登録する」を参照してください。

ロードバランサーなしで CodeDeploy のブルー/グリーンデプロイを実行で きますか?

いいえ、ロードバランサーなしで CodeDeploy のブルー/グリーンデプロイを実行することはでき ません。ロードバランサーを使用できない場合は、代わりに Amazon ECS のローリング更新機能 を使用してください。Amazon ECS のローリング更新機能の詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「ローリング更新」を参照してください。

デプロイ中に Amazon ECS サービスを新しい情報で更新する方法を教えて ください。

CodeDeploy がデプロイの実行中に Amazon ECS サービスを新しいパラメータで更新するように するには、AppSpec ファイルの resources セクションでパラメータを指定します。CodeDeploy でサポートされている Amazon ECS パラメータは、タスク定義ファイルやコンテナ名パラメータ など、ごく一部です。CodeDeploy が更新できる Amazon ECS パラメータの一覧については、「 Amazon ECS デプロイ用の AppSpec の「resources」セクション」を参照してください。

Note

CodeDeploy でサポートされていないパラメータで Amazon ECS サービスを更新する必要が ある場合は、以下のタスクを実行します。

- 1. 更新したいパラメータを指定して Amazon ECS の UpdateService API を呼び出しま す。更新できるパラメータの一覧については、「Amazon Elastic Container Service API リ ファレンス」の「UpdateService」を参照してください。
- 2. 変更をタスクに適用するには、新しい Amazon ECS ブルー/グリーンデプロイを作成しま す。詳細については、「<u>Amazon ECS コンピューティングプラットフォームのデプロイの</u> <u>作成 (コンソール)</u>」を参照してください。

AWS Lambda デプロイの問題のトラブルシューティング

トピック

 <u>AWS Lambda ロールバックが設定されていない Lambda デプロイを手動で停止すると、デプロイ</u> は失敗します

AWS Lambda ロールバックが設定されていない Lambda デプロイを手動で 停止すると、デプロイは失敗します

場合によっては、デプロイで指定された Lambda 関数のエイリアスで、2 つの異なる関数バー ジョンが参照されることがあります。そのため、Lambda 関数をデプロイしようとすると失敗しま す。Lambda デプロイでは、ロールバックが設定されていない場合に手動で停止すると、この状態 になることがあります。続行するには、 AWS Lambda コンソールを使用して、 関数が 2 つのバー ジョン間でトラフィックをシフトするように設定されていないことを確認します。

- 1. にサインイン AWS Management Console し、 AWS Lambda コンソールを <u>https://</u> console.aws.amazon.com/lambda/://www.com で開きます。
- 2. 左側のペインで、[関数]を選択します。
- 3. CodeDeploy デプロイ内にある Lambda 関数の名前を選択します。

- 4. 「エイリアス」から、CodeDeploy デプロイで使用されているエイリアスを選択し、「編集」を選 択します。
- 5. [加重エイリアス] から [none] を選択します。これにより、このエイリアスで、トラフィックの割 合やウェイトが 2 つ以上のバージョンに移行されることはありません。[バージョン] で選択され ているバージョンを書き留めます。
- 6. [Save] を選択します。
- 7. CodeDeploy コンソールを開き、ステップ 5 のドロップダウンメニューで表示されているバー ジョンをデプロイします。

デプロイグループの問題のトラブルシューティング

デプロイグループの一部としてインスタンスにタグを付けても、アプリ ケーションが自動的に新しいインスタンスにデプロイされない

CodeDeploy は新しくタグが付けられたインスタンスに自動的にアプリケーションをデプロイしません。デプロイグループで新しいデプロイを作成する必要があります。

CodeDeploy を使用すると、Amazon EC2 Auto Scaling グループの新しい EC2 インスタンスへの 自動デプロイを有効にすることができます。詳細については、「<u>CodeDeploy と Amazon EC2 Auto</u> Scaling の統合」を参照してください。

インスタンスの問題のトラブルシューティング

トピック

- タグは正しく設定する必要がある
- AWS CodeDeploy エージェントはインスタンスにインストールして実行する必要があります。
- デプロイ中にインスタンスを削除した場合、デプロイは最大1時間は失敗しません。
- ログファイルの分析によるインスタンスでのデプロイの失敗の調査
- ・ 誤って削除した場合は、新しい CodeDeploy ログファイルを作成します。
- <u>「InvalidSignatureException Signature expired: [time] is now earlier than [time]」デプロイエラー</u>のトラブルシューティング

タグは正しく設定する必要がある

デプロイに使用されるインスタンスに正しくタグが付けられていることを確認するには、<u>list-</u> <u>deployment-instances</u> コマンドを使用します。出力に EC2 インスタンスがない場合は、EC2 コ ンソールを使用して、インスタンスにタグが設定されていることを確認します。詳細について は、Amazon EC2 ユーザーガイド」の「コンソールでのタグの使用」を参照してください。

Note

インスタンスにタグを付け、直後に CodeDeploy を使用してそのインスタンスにアプリケー ションをデプロイする場合、インスタンスはデプロイに含まれないこともあります。これ は、CodeDeploy がタグを読み込むまでに数分かかることがあるためです。インスタンスに タグを付けてから、そのインスタンスへのデプロイを試みるまでに、少なくとも5分待つこ とをお勧めします。

AWS CodeDeploy エージェントはインスタンスにインストールして実行す る必要があります

インスタンスに CodeDeploy エージェントがインストールされて実行されていることを確認する方 法については、「<u>CodeDeploy エージェントが実行されていることの確認</u>」を参照してください。

CodeDeploy エージェントをインストール、アンインストール、または再インストールする方法については、「CodeDeploy エージェントをインストールする」を参照してください。

デプロイ中にインスタンスを削除した場合、デプロイは最大1時間は失敗 しません。

CodeDeploy は、各デプロイライフサイクルイベントの実行が完了するまでに1時間の枠を設定しています。これにより、実行時間が長いスクリプトにも十分な時間が提供されます。

ライフサイクルイベントの進行中に、スクリプト実行が完了しない場合 (たとえば、インスタンス が終了された、CodeDeploy エージェントがシャットダウンされたなど)、デプロイのステータスが Failed になるまで、最大で1時間かかることがあります。スクリプトに指定されたタイムアウト時 間が1時間未満である場合も同様です。これは、インスタンスが終了すると CodeDeploy エージェ ントがシャットダウンし、それ以上のスクリプトを処理できなくなるためです。

インスタンスがライフサイクル間、または最初のライフサイクルイベントのステップが開始する前に 削除された場合、タイムアウトはわずか 5 分後に発生します。

ログファイルの分析によるインスタンスでのデプロイの失敗の調査

デプロイのインスタンスのステータスが Succeeded 以外のいずれかである場合は、デプロイのログ ファイルデータを確認すると、問題の特定に役立ちます。デプロイのログデータへのアクセス方法に ついては、「<u>CodeDeploy EC2/オンプレミスデプロイのログデータの表示</u>」を参照してください。

誤って削除した場合は、新しい CodeDeploy ログファイルを作成します。

誤ってインスタンスのデプロイログファイルを削除した場合、CodeDeploy は代わりのログファイル を作成しません。新しいログファイルを作成するには、インスタンスにサインインし、以下のコマン ドを実行します。

Amazon Linux、Ubuntu Server、または RHEL インスタンスの場合、以下のコマンドをこの順序で 1 つずつ実行します。

systemctl stop codedeploy-agent

systemctl start codedeploy-agent

Windows Server インスタンスの場合

powershell.exe -Command Restart-Service -Name codedeployagent

「InvalidSignatureException – Signature expired: [time] is now earlier than [time]」デプロイエラーのトラブルシューティング

CodeDeploy では、オペレーションを実行するために正確な時間の参照が必要です。インスタンスの 日時が正しく設定されていない場合は、デプロイリクエストの署名と一致しないことがあり、その場 合は CodeDeploy によってリクエストが却下されます。

誤った時間設定に関連するデプロイの失敗を回避する方法については、次のトピックを参照してくだ さい。

- Linux インスタンスの時刻の設定
- Windows インスタンスの時刻を設定する

GitHub トークンの問題のトラブルシューティング

GitHub OAuth トークンが無効です

2017 年 6 月以降に作成された CodeDeploy アプリケーションでは、各 AWS リージョンの GitHub OAuth トークンを使用します。特定の AWS リージョンに関連付けられたトークンを使用する と、GitHub リポジトリにアクセスできる CodeDeploy アプリケーションをより詳細に制御できま す。

GitHub トークンエラーが発生した場合は、すでに無効になっている古いトークンの場合があります。

無効な GitHub OAuth トークンを修正するには

- 1. 以下のいずれかの方法を使用して、古いトークンを削除してください。
 - API を使って古いトークンを削除するには、DeleteGitHubAccountToken を使用します。
 - AWS Command Line Interfaceを使用して古いトークンを削除するには:
 - a. トークンが存在するコンピュータに移動します。
 - b. AWS CLI がこのコンピュータにインストールされていることを確認します。インストール方法については、AWS CLIユーザーガイドの「AWS Command Line Interfaceのインストール、更新、およびアンインストール」を参照してください。
 - c. トークンが存在するコンピュータで、次のコマンドを入力します。

aws delete-git-hub-account-token

コマンド構文の詳細については、「<u>delete-git-hub-account-token</u>」を参照してください。

 新しい OAuth トークンを追加します。詳細については、「<u>GitHub と CodeDeploy との統合</u>」を 参照してください。

GitHub OAuth トークンの最大数を超えました

CodeDeploy のデプロイを作成する際に許可されている GitHub トークンの最大数は 10 で す。GitHub OAuth トークンに関するエラーが発生した場合、トークンが 10 個以下であることを確 認します。トークンが 10 個以上ある場合は、最初に作成されたトークンが無効になります。たとえ ば、トークンが 11 個ある場合、最初に作成したトークンが無効になります。トークンが 12 個ある 場合、最初に作成した 2 個のトークンが無効になります。CodeDeploy API を使用して古いトークン を削除する方法については、「DeleteGitHubAccountToken」を参照してください。

Amazon EC2 Auto Scaling の問題のトラブルシューティング

トピック

- <u>Amazon EC2 Auto Scaling の一般的なトラブルシューティング</u>
- <u>CodeDeployRole は、次の AWS サービスでオペレーションを実行するアクセス許可を付与しません</u>: AmazonAutoScaling」エラー
- <u>Amazon EC2 Auto Scaling グループのインスタンスのプロビジョニングと終了が繰り返されてリ</u>ビジョンをデプロイできない
- <u>Amazon EC2 Auto Scaling インスタンスを終了または再起動すると、デプロイが失敗する場合がある</u>
- ・ 複数のデプロイグループを1つの Amazon EC2 Auto Scaling グループに関連付けることは避ける
- <u>Amazon EC2 Auto Scaling グループの EC2 インスタンスが起動に失敗し、「ハートビートのタイムアウト」というエラーが表示される</u>
- <u>Amazon EC2 Auto Scaling ライフサイクルフックの不一致により、Amazon EC2 Auto Scaling グ</u> ループへの自動デプロイが停止または失敗することがある
- 「デプロイグループのインスタンスが見つからないため、デプロイに失敗しました」というエラー

Amazon EC2 Auto Scaling の一般的なトラブルシューティング

Amazon EC2 Auto Scaling グループの EC2 インスタンスへのデプロイは、次の理由で失敗する場合 があります。

 Amazon EC2 Auto Scaling は、EC2 インスタンスを継続的に起動および終了します。CodeDeploy がアプリケーションリビジョンを自動的にデプロイできない場合、Amazon EC2 Auto Scaling は 継続的に EC2 インスタンスを起動および終了させます。

CodeDeploy デプロイグループから Amazon EC2 Auto Scaling グループの関連付けを解除する か、Amazon EC2 Auto Scaling グループの設定を変更し、必要なインスタンス数が現在のインス タンス数に一致するようにします (これにより、Amazon EC2 Auto Scaling がそれ以上の EC2 イ ンスタンスを起動できないようにします)。詳細については、「<u>CodeDeploy を使用して、デプロ</u> <u>イグループの設定を変更します。</u>」または「<u>Amazon EC2 Auto Scaling のマニュアルスケーリン</u> グ」を参照してください。

- CodeDeploy エージェントが応答しません。EC2 インスタンスの起動直後に実行される初期化 スクリプト (cloud-init スクリプトなど)の実行に1時間以上かかる場合、CodeDeploy エージェ ントがインストールされないことがあります。CodeDeploy には、CodeDeploy エージェントが 保留中のデプロイに応答するための1時間のタイムアウトがあります。この問題に対応するに は、CodeDeploy アプリケーションリビジョンに初期化スクリプトを移動します。
- Amazon EC2 Auto Scaling グループの EC2 インスタンスは、デプロイ中に再起動します。デプロ イ中に EC2 インスタンスが再起動したか、デプロイコマンドの処理中に CodeDeploy エージェン トがシャットダウンした場合、デプロイは失敗することがあります。詳細については、「<u>Amazon</u> <u>EC2 Auto Scaling インスタンスを終了または再起動すると、デプロイが失敗する場合がある</u>」を 参照してください。
- 複数のアプリケーションリビジョンが Amazon EC2 Auto Scaling グループの同じ EC2 インスタン スに同時にデプロイされた。Amazon EC2 Auto Scaling グループの同じ EC2 インスタンスに複数 のアプリケーションリビジョンをデプロイする場合、デプロイの1つに数分以上実行されるスク リプトがあると、失敗することがあります。Amazon EC2 Auto Scaling グループの同じ EC2 イン スタンスに複数のアプリケーションリビジョンをデプロイしないでください。
- Amazon EC2 Auto Scaling グループの一部として起動された新しい EC2 インスタンスに対して、 デプロイが失敗する。このシナリオでは、デプロイでスクリプトを実行すると、Amazon EC2 Auto Scaling グループで EC2 インスタンスを起動できなくなる場合があります。(Amazon EC2 Auto Scaling グループの他の EC2 インスタンスは、正常に実行されているように見える場合があ ります)。この問題に対応するには、最初にその他のすべてのスクリプトが完了していることを確 認します。
 - CodeDeploy エージェントは AMI に含まれていません: 新しいインスタンスの起動中に cfn-init コマンドを使用して CodeDeploy エージェントをインストールする場合は、 AWS CloudFormation テンプレートの cfn-initセクションの最後にエージェントインストールスク リプトを配置します。
 - CodeDeploy エージェントを AMI に含める場合: インスタンスの作成時にエージェントが Stopped 状態になるように設定し、次に cfn-init スクリプトライブラリの最終ステップとし て、エージェントを起動するスクリプトを含めます

CodeDeployRole は、次の AWS サービスでオペレーションを実行するアク セス許可を付与しません: AmazonAutoScaling」エラー

起動テンプレートを使用して作成された Auto Scaling グループを使用するデプロイでは、次のアク セス権限が必要です。これらは、 AWSCodeDep1oyRo1e AWS 管理ポリシーによって付与されるア クセス許可に追加されます。

- EC2:RunInstances
- EC2:CreateTags
- iam:PassRole

これらのアクセス権限がないために、このエラーが発生した可能性があります。詳細については、 「Amazon EC2 Auto Scaling ユーザーガイド」の「<u>チュートリアル: CodeDeploy を使用して、Auto</u> <u>Scaling グループにアプリケーションをデプロイします。</u>」、「<u>Auto Scaling グループの起動テンプ</u> <u>レートの作成</u>」、および「<u>アクセス許可</u>」を参照してください。

Amazon EC2 Auto Scaling グループのインスタンスのプロビジョニングと 終了が繰り返されてリビジョンをデプロイできない

エラーが原因で、Amazon EC2 Auto Scaling グループ内の新しくプロビジョニングされたインスタ ンスに正常にデプロイできない場合があります。その結果として、インスタンスは正常な状態になら ず、デプロイは失敗します。デプロイが正常に実行または完了されないため、インスタンスは作成後 すぐに終了されます。この場合、Amazon EC2 Auto Scaling グループの設定により、正常なホスト 数の最小要件を満たすために別のバッチのインスタンスがプロビジョニングされます。このバッチも 終了され、同じサイクルが繰り返されます。

エラーの原因として以下が考えられます。

- Amazon EC2 Auto Scaling グループのヘルスチェックに失敗しました。
- アプリケーションリビジョンのエラー。

この問題を回避するには、次の手順に従います。

- 1. Amazon EC2 Auto Scaling グループに属していない EC2 インスタンスを手動で作成します。イン スタンスに一意の EC2 インスタンスタグを付けます。
- 2. 新しいインスタンスを該当するデプロイグループに追加します。
- 3. 新しい、エラーのないアプリケーションリビジョンをデプロイグループにデプロイします。

これにより、Amazon EC2 Auto Scaling グループの今後のインスタンスにアプリケーションリビ ジョンがデプロイされるようになります。 Note

デプロイが成功したことを確認したら、作成したインスタンスを削除して、 AWS アカウン トへの継続的な課金を回避します。

Amazon EC2 Auto Scaling インスタンスを終了または再起動すると、デプ ロイが失敗する場合がある

EC2 インスタンスが Amazon EC2 Auto Scaling を通じて起動され、その後で終了または再起動され ると、そのインスタンスへのデプロイは、次の理由で失敗する場合があります。

- 進行中のデプロイで、スケールインイベントまたはその他の終了イベントにより、インスタンスは Amazon EC2 Auto Scaling グループからデタッチされた後に削除されます。デプロイは完了でき ないため、失敗します。
- インスタンスが再起動されたが、その起動に5分間以上かかる。CodeDeploy はこれをタイムアウトとして扱います。サービスにより、インスタンスに対する現在およびそれ以降のすべてのデプロイが失敗します。

この問題に対応するには:

- 一般的に、インスタンスが削除または再起動される前に、すべてのデプロイが完了したことを確認します。インスタンスの起動または再起動後に、すべてのデプロイが開始されることを確認します。
- Amazon EC2 Auto Scaling の設定に Windows Server ベースの Amazon Machine Image (AMI) を 指定し、EC2Config サービスを使用して、インスタンスのコンピュータ名を設定すると、デプ ロイは失敗します。この問題を解決するには、Windows Server ベース AMI で、[EC2 Service Properties] (EC2 サービスプロパティ)の [General] (全般) タブにある [Set Computer Name] (コ ンピュータ名の設定) をオフにします。このチェックボックスをオフにすると、この動作は、そ の Windows Server の基本 AMI で起動されるすべての新しい Windows Server Amazon EC2 Auto Scaling インスタンスで、この動作が無効になります。このチェックボックスをオフにする必要は ありません。再起動後に、失敗したデプロイをこれらのインスタンスに再デプロイします。

複数のデプロイグループを 1 つの Amazon EC2 Auto Scaling グループに関 連付けることは避ける

各 Amazon EC2 Auto Scaling グループには 1 つのデプロイグループのみを関連付けることをお勧め します。

これは、複数のデプロイグループに関連付けられたフックを持つインスタンスを Amazon EC2 Auto Scaling がスケールアップする場合、すべてのフックに対して一度に通知を送信するためです。 これにより、各インスタンスの複数のデプロイが同時に開始されます。複数のデプロイが同時に CodeDeploy エージェントへコマンドを送信すると、ライフサイクルイベントとデプロイの開始また は前のライフサイクルイベント終了の間に5分間のタイムアウトに達する場合があります。その場 合は、予想通りにデプロイがプロセスされていてもデプロイが失敗します。

Note

ライフサイクルイベント内にあるスクリプトのデフォルトのタイムアウトは、デフォルト で 30 分です。AppSpec ファイル内のタイムアウトを別の値に変更できます。詳細について は、「EC2/オンプレミスデプロイに AppSpec ファイルを追加する」を参照してください。

複数のデプロイが同時に実行を試みた場合、デプロイが発生する順序を制御することはできない。

最後に、インスタンスへのデプロイが失敗した場合、Amazon EC2 Auto Scaling は直ちにインスタ ンスを終了します。その最初のインスタンスがシャットダウンすると、実行中の他のデプロイも失敗 します。CodeDeploy には、CodeDeploy エージェントが保留中のデプロイに応答するための 1 時間 のタイムアウトがあるため、各インスタンスのタイムアウトは最大 60 分かかる場合があります。

Amazon EC2 Auto Scaling の詳細については、「<u>内部構造: CodeDeploy と Auto Scaling の統合</u>」を 参照してください。

Amazon EC2 Auto Scaling グループの EC2 インスタンスが起動に失敗し、 「ハートビートのタイムアウト」というエラーが表示される

Amazon EC2 Auto Scaling グループが新しい EC2 インスタンスの起動に失敗し、次のようなメッ セージを生成する場合があります。

Launching a new EC2 instance <*instance-Id*>. Status Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with token<*token-Id*> was abandoned: Heartbeat Timeout.

このメッセージは通常、以下のいずれかを示します。

- AWS アカウントに関連付けられた同時デプロイの最大数に達しました。デプロイの制限の詳細に ついては、「CodeDeploy のクォータ」を参照してください。
- Auto Scaling グループは、あまりにも多くの EC2 インスタンスを早く起動しようとしました。 新しいインスタンスごとに <u>RecordLifecycleActionHeartbeat</u> または <u>CompleteLifecycleAction</u> への API コールがスロットルされました。
- ・関連付けられたデプロイグループが更新または削除される前に、CodeDeployのアプリケーションが削除された。

アプリケーションまたはデプロイグループを削除すると、CodeDeploy はそれに関連付けられてい た Amazon EC2 Auto Scaling フックのクリーンアップを試みますが、一部のフックが残る場合が あります。デプロイグループを削除するコマンドを実行すると、残りのフックが出力で返ります。 ただし、アプリケーションを削除するコマンドを実行した場合、残りのフックは出力に表示されま せん。

したがって、アプリケーションを削除する前に、アプリケーションと関連付けられたすべてのデプ ロイグループを削除することをお勧めします。コマンド出力を使用して、手動で削除する必要があ るライフサイクルフックを識別できます。

「ハートビートのタイムアウト」エラーメッセージが表示される場合は、次の操作を行い、残ってい るライフサイクルフックが原因かどうかを特定して問題を解決します。

1. 次のいずれかを行います:

- <u>delete-deployment-group</u> コマンドを呼び出して、ハートビートタイムアウトの原因となって いる Auto Scaling グループに関連付けられたデプロイグループを削除します。
- Auto Scaling グループ名の NULL ではない空のリストを指定して、update-deployment-group コマンドを呼び出し、CodeDeploy が管理する全ての Auto Scaling ライフサイクルフックをデ タッチすることができます。

たとえば、次のコマンドを入力します AWS CLI。

aws deploy update-deployment-group --application-name my-example-app --current-deployment-group-name my-deployment-group --auto-scalinggroups 別の例として、Java で CodeDeploy API を使用する場合は、UpdateDeploymentGroup を 呼び出し、autoScalingGroups を new ArrayList<String>()に設定します。これによ り、autoScalingGroups を空のリストに設定し、既存のリストを削除します。デフォルト の null を使用すると、autoScalingGroups がそのまま残ってしまうので使用しないでく ださい。

呼び出しの出力を確認します。出力に hooksNotCleanedUp 構造と Amazon EC2 Auto Scaling ライフサイクルフックの一覧が含まれている場合、ライフサイクルフックの残りがあります。

- 2. <u>describe-lifecycle-hooks</u> コマンドを呼び出し、起動に失敗した EC2 インスタンスに関連付けら れている Amazon EC2 Auto Scaling グループの名前を指定します。出力で、次のいずれかを確 認します。
 - Amazon EC2 Auto Scaling ライフサイクルフック名で、ステップ1で特定した hooksNotCleanedUp 構造に対応するもの
 - Amazon EC2 Auto Scaling ライフサイクルフック名で、失敗している Auto Scaling グループ に関連するデプロイグループの名前を含むもの
 - Amazon EC2 Auto Scaling のライフサイクルフック名で、CodeDeploy デプロイのハートビー トタイムアウトの原因となった可能性のあるもの
- 3. フックがステップ 2 でリストされたカテゴリのいずれかに該当する場合は、<u>delete-lifecycle-</u> <u>hook</u> コマンドを呼び出して削除します。Amazon EC2 Auto Scaling グループとライフサイクル フックをコールで指定します。

Important

ステップ2で説明したように、問題の原因となっているフックのみを削除してくださ い。実行可能なフックを削除すると、デプロイが失敗したり、CodeDeploy がアプリ ケーションリビジョンをスケールアウトした EC2 インスタンスにデプロイできなくなっ たりする可能性があります。

4. <u>update-deployment-group</u> または <u>create-deployment-group</u> コマンドを、必要な Auto Scaling グ ループ名で呼び出します。CodeDeploy は、新しい UUID を持つ Auto Scaling フックを再インス トールします。

1 Note

CodeDeploy デプロイグループから Auto Scaling グループをデタッチすると、Auto Scaling グループへの進行中のデプロイはすべて失敗し、Auto Scaling グループによっ てスケールアウトされる新しい EC2 インスタンスは CodeDeploy からアプリケーショ ンリビジョンを受けとれなくなる可能性があります。CodeDeploy で Auto Scaling を再 度動作させるには、Auto Scaling グループをデプロイグループに再アタッチし、新しい CreateDeployment を呼び出しして、フリート全体のデプロイを開始する必要がありま す。

Amazon EC2 Auto Scaling ライフサイクルフックの不一致により、Amazon EC2 Auto Scaling グループへの自動デプロイが停止または失敗することがある

Amazon EC2 Auto Scaling と CodeDeploy はライフサイクルフックを使用して、Amazon EC2 Auto Scaling グループで起動された後にデプロイするアプリケーションリビジョンと、そのデプロイ先の EC2 インスタンスを判断します。ライフサイクルフックとそれに関する情報が Amazon EC2 Auto Scaling と CodeDeploy で正確に一致しない場合、自動デプロイは停止または失敗することがありま す。

Amazon EC2 Auto Scaling グループへのデプロイに失敗する場合、Amazon EC2 Auto Scaling および CodeDeploy のライフサイクルフック名が一致するかどうかを確認します。そうでない場合は、 これらの AWS CLI コマンド呼び出しを使用します。

最初に、Amazon EC2 Auto Scaling グループとデプロイグループの両方のライフサイクルフック名の一覧を取得します。

- <u>describe-lifecycle-hooks</u> コマンドを呼び出し、CodeDeploy のデプロイグループに関連付けられ た Amazon EC2 Auto Scaling グループの名前を指定します。出力の LifecycleHooks リスト で、LifecycleHookName のそれぞれの値を書き留めます。
- <u>get-deployment-group</u> コマンドを呼び出し、Amazon EC2 Auto Scaling グループに関連付けら れたデプロイグループの名前を指定します。出力の autoScalingGroups リストで、名前の値 が Amazon EC2 Auto Scaling グループ名と一致する項目を見つけ、対応する hook の値を書き 留めます。

ここで、2 セットのライフサイクルフック名を比較します。それらが 1 文字ずつ正確に一致する場合は、これが問題ではありません。このセクションの他の場所で説明されている Amazon EC2 Auto Scaling の他のトラブルシューティングステップを試してください。

ただし、2 セットのライフサイクルフック名が 1 文字ずつ正確に一致しない場合は、次の操作を行い ます。

- get-deployment-group コマンド出力にもないライフサイクルフック名が describe-lifecyclehooks コマンド出力にある場合は、次の操作を行います。
 - a. describe-lifecycle-hooks コマンド出力のライフサイクルフック名ごとに、<u>delete-lifecycle-</u>hook コマンドを呼び出します。
 - b. <u>update-deployment-group</u> コマンドを呼び出し、元の Amazon EC2 Auto Scaling のグルー プ名を指定します。CodeDeploy は、Amazon EC2 Auto Scaling グループに新しい代替のラ イフサイクルフックを作成し、そのライフサイクルフックをデプロイグループに関連付けま す。これで、Amazon EC2 Auto Scaling グループに新しいインスタンスを追加すると、自 動デプロイが開始されます。
- describe-lifecycle-hooks コマンド出力にもないライフサイクルフック名が get-deploymentgroup コマンド出力にある場合は、次の操作を行います。
 - a. <u>update-deployment-group</u> コマンドを呼び出し、元の Amazon EC2 Auto Scaling のグルー プ名を指定しないようにします。
 - b. update-deployment-group コマンドを再度呼び出し、今度は元の Amazon EC2 Auto Scaling グループの名前を指定します。CodeDeploy は、Amazon EC2 Auto Scaling グループに不 足しているライフサイクルフックを再作成します。これで、Amazon EC2 Auto Scaling グ ループに新しいインスタンスを追加すると、自動デプロイが開始されます。

1 文字ごとに正確に一致する 2 セットのライフサイクルフック名を取得したら、アプリケーションリ ビジョンをもう一度デプロイしますが、Amazon EC2 Auto Scaling グループに追加された新しいイ ンスタンスにのみデプロイします。Amazon EC2 Auto Scaling グループに既に存在するインスタン スに対しては、デプロイは自動的に行われません。

「デプロイグループのインスタンスが見つからないため、デプロイに失敗 しました」というエラー

以下のような CodeDeploy エラーが発生した場合は、このセクションをお読みください。

The deployment failed because no instances were found for your deployment group. Check your deployment group settings to make sure the tags for your EC2 instances or Auto Scaling groups correctly identify the instances you want to deploy to, and then try again.

このエラーの原因として考えられるのは、以下の通りです。

- デプロイグループの設定には、正しくない Auto Scaling グループ、EC2 インスタンス、またはオ ンプレミスインスタンスのタグが含まれています。この問題を解決するには、タグが正しいこと をチェックしてから、アプリケーションを再デプロイしてください。
- 2. デプロイ開始後、フリートがスケールアウトしました。このシナリオでは、フリートで InService 状態の正常なインスタンスが表示されますが、上記のエラーも表示されます。この問 題を解決するには、アプリケーションを再デプロイします。
- Auto Scaling グループには、InService 状態のインスタンスは含まれません。このシナリオでは、CodeDeploy は少なくとも1つのインスタンスがInService 状態である必要があるため、フリート全体のデプロイを実行しようとすると、上記のエラーメッセージが表示されてデプロイが失敗します。InService 状態でインスタンスがない場合、様々な理由が考えられます。その一部を紹介します。
 - Auto Scaling グループのサイズを 0 にスケジュール (または手動で設定) しました。
 - Auto Scaling は、不正な EC2 インスタンスを検出したため (例えば、EC2 インスタンスにハードウェア障害が発生した)、すべてキャンセルし、InService 状態には何も残さないようにしました。
 - 0から1へのスケールアウトイベントで、CodeDeployは前回成功したリビジョン(最後に成功したリビジョン)をデプロイしましたが、前回のデプロイから正常ではない状態に陥っていました。これにより、スケールアウトしたインスタンスのデプロイが失敗し、そのため、Auto Scaling がインスタンスをキャンセルし、InService 状態のインスタンスが一つも残らないという事態が発生しました。

InService 状態のインスタンスがないことがわかった場合、次の手順 <u>To troubleshoot the</u> <u>error if there are no instances in the InService state</u> の説明に従ってトラブルシューティングし ます。

InService 状態のインスタンスが存在しない場合のエラーのトラブルシューティングについて

 Amazon EC2 コンソールで、[希望する容量]の設定を確認します。ゼロの場合は、正の数に設定 します。インスタンスが InService になるのを待ちます。これは、デプロイが成功したことを
意味します。問題が解決されたので、このトラブルシューティングの手順の残りのステップは スキップできます。[Desired Capacity] (希望する容量) の設定については、「Amazon EC2 Auto Scaling ユーザーガイド」の「<u>Auto Scaling グループの容量制限を設定する</u>」を参照してくださ い。

- Auto Scaling が希望する容量を満たすために新しい EC2 インスタンスを起動し続けようとする が、スケールアウトを実行できない場合は、通常、Auto Scaling のライフサイクルフックが失敗 していることが原因です。この問題については、次のようにトラブルシューティングします。
 - a. 失敗している Auto Scaling ライフサイクルフックイベントを確認するには、Amazon EC2
 Auto Scaling ユーザーガイドの「<u>Auto Scaling グループのスケーリングアクティビティの確</u>認」を参照してください。
 - b. 失敗したフックの名前が CodeDeploy-managed-automatic-launch-deploymenthook-DEPLOYMENT_GROUP_NAME の場合、CodeDeploy に移動し、デプロイグループを見 つけて、Auto Scaling によって開始され、失敗したデプロイを見つけます。次に、デプロイ が失敗した理由を調べます。
 - c. デプロイが失敗した理由 (例えば、CloudWatch アラームが発生していたなど) がわかってい て、リビジョンを変更せずに問題を修正できる場合は、今すぐ実行してください。
 - d. 調査の結果、CodeDeploy の最後に成功したリビジョンが正常ではなくなり、Auto Scaling グループ内の正常なインスタンスがゼロであると判断された場合、デプロイのデッド ロックシナリオに陥っています。この問題を解決するには、Auto Scaling グループから CodeDeploy のライフサイクルフックを一時的に削除し、フックを再インストールして新し い (正しい) リビジョンを再デプロイすることで、不正な CodeDeploy リビジョンを修正す る必要があります。手順については、こちらを参照してください。
 - To fix the deployment deadlock issue (CLI)
 - To fix the deployment deadlock issue (console)

デプロイのデッドロックの問題を解決するには (CLI)

- (オプション) CodeDeploy エラーを引き起こしている CI/CD パイプラインをブロックして、この 問題を解決している間に予期しないデプロイが発生しないようにします。
- 2. Auto Scaling の現在の [DesiredCapacity] 設定を書き留めます:

aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name ASG_NAME

場合によっては、この手順の最後に、この数値にスケールバックする必要があります。

 Auto Scaling の [DesiredCapacity] 設定を1にしてください。開始の際に希望する容量が1よ り大きい場合、オプションとなります。それを1に減らすことで、インスタンスのプロビジョ ニングとデプロイにかかる時間が短くなり、トラブルシューティングが高速化されます。Auto Scaling の希望する容量がもともと0に設定されている場合、1に増やす必要があります。これ は必須です。

aws autoscaling set-desired-capacity — auto-scaling-group-name ASG_NAME — desired-capacity 1

Note

この手順の残りのステップでは、[DesiredCapacity] を1に 設定したものとします。

この時点で、Auto Scaling は 1 つのインスタンスへのスケーリングを試みます。次

に、CodeDeploy が追加したフックがまだ存在するため、CodeDeploy がデプロイを試みても失 敗し、Auto Scaling はインスタンスをキャンセルして希望容量の 1 に達するようにインスタンス を再起動しようとしますが、これも失敗します。キャンセル再起動ループに入っています。

4. デプロイグループから Auto Scaling グループの登録を解除します。

A Warning

次のコマンドは、ソフトウェアなしで新しい EC2 インスタンスを起動します。コマンド を実行する前に、ソフトウェアを実行していない Auto Scaling InService インスタン スが許容されることを確認します。例えば、インスタンスに関連付けられているロード バランサーがソフトウェアなしでこのホストにトラフィックを送信していないことを確 認してください。

A Important

以下に示す CodeDeploy コマンドを使用して、フックを削除します。Auto Scaling サービスによるフック削除は、CodeDeploy で認識されないため、削除しないでください。

aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scalinggroups

このコマンドを実行すると、次の処理が実行されます。

- a. CodeDeployは、デプロイグループから Auto Scaling グループを登録解除します。
- b. CodeDeploy は Auto Scaling グループから Auto Scaling ライフサイクルフックを削除します。
- c. デプロイ失敗の原因となっていたフックが存在しなくなるため、Auto Scaling は既存の EC2 インスタンスをキャンセルし、希望容量にスケーリングする新しいインスタンスを直 ちに起動します。新しいインスタンスは、InService 状態にまもなく移行するはずです。 新しいインスタンスには、ソフトウェアは含まれません。
- 5. EC2 インスタンスが InService 状態になるのを待ちます。その状態を確認するには、次のコ マンドを使用します。

aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
ASG_NAME --query AutoScalingGroups[0].Instances[*].LifecycleState

6. EC2 インスタンスにフックを追加し直します。

Important

以下に示す CodeDeploy コマンドを使用して、フックを追加します。Auto Scaling サー ビスによるフックの追加は、CodeDeploy で認識されないため、利用しないでくださ い。

aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scalinggroups ASG_NAME

このコマンドを実行すると、次の処理が実行されます。

- a. CodeDeploy は、Auto Scaling ライフサイクルフックを EC2 インスタンスに再インストー ルします。
- b. CodeDeploy は、Auto Scaling グループをデプロイグループに再登録します。

正常であることがわかり、使用したい Amazon S3 または GitHub のリビジョンで、フリート全体のデプロイを作成します。

たとえば、リビジョンが、my-revision-bucket という Amazon S3 バケットにある .zip ファ イルで、オブジェクトキーが httpd_app.zip である場合、次のコマンドを入力します。

aws deploy create-deployment --application-name APPLICATION_NAME --deployment-group-name DEPLOYMENT_GROUP_NAME -revision "revisionType=S3,s3Location={bucket=my-revisionbucket,bundleType=zip,key=httpd_app.zip}"

現在、Auto Scaling グループに InService インスタンスが一つ存在するため、このデプロイは 機能するはずで、エラー [デプロイグループのインスタンスが見つからないため、デプロイに失 敗しました] は表示されなくなります。

8. 以前にスケールインしていた場合、デプロイが成功したら、Auto Scaling グループをスケールア ウトして元の容量に戻します。

aws autoscaling set-desired-capacity --auto-scaling-group-name ASG_NAME
--desired-capacity ORIGINAL_CAPACITY

デプロイのデッドロックの問題を解決するには (コンソール)

- (オプション) CodeDeploy エラーを引き起こしている CI/CD パイプラインをブロックして、この 問題を解決している間に予期しないデプロイが発生しないようにします。
- Amazon EC2 コンソールにアクセスし、Auto Scaling の [希望する容量] の設定を書き留めま す。場合によっては、この手順の最後に、この数値にスケールバックする必要があります。この 設定の見つけ方の詳細については、「<u>Auto Scaling グループの容量制限の設定</u>」を参照してくだ さい。
- 3. 必要な EC2 インスタンスの数を1 に設定:

希望する容量が1より大きい場合、オプションとなります。それを1に減らすことで、インス タンスのプロビジョニングとデプロイにかかる時間が短くなり、トラブルシューティングが高速 化されます。Auto Scaling の 希望する容量 がもともとに 0 設定されている場合、1 に増やす必 要があります。これは必須です。

Note

この手順の残りのステップでは、1 に 希望する容量 を設定したものとします。

- a. <u>https://console.aws.amazon.com/ec2/</u> でAmazon EC2 コンソールを開き、ナビゲーション ペインで [Auto Scaling グループ] を選択します。
- b. 適切なリージョンを選択します。
- c. 問題がある Auto Scaling グループに移動します。
- d. [グループの詳細] で、[編集] を選択します。
- e. 1に希望する容量を設定。
- f. [Update] (更新)を選択します。
- 4. デプロイグループから Auto Scaling グループの登録を解除します。

A Warning

次のサブステップでは、ソフトウェアなしで新しい EC2 インスタンスを起動します。コ マンドを実行する前に、ソフトウェアを実行していない Auto Scaling InService イン スタンスが許容されることを確認します。例えば、インスタンスに関連付けられている ロードバランサーがソフトウェアなしでこのホストにトラフィックを送信していないこ とを確認してください。

- a. https://console.aws.amazon.com/codedeploy/ で、CodeDeploy コンソールを開きます。
- b. 適切なリージョンを選択します。
- c. ナビゲーションペインで、[アプリケーション] を選択します。
- d. CodeDeploy アプリケーションの名前を選択します。
- e. CodeDeploy デプロイグループ名を選択する。
- f. [編集]を選択します。
- g. 環境設定 では、Amazon EC2 Auto Scaling グループ の選択を解除します。

Note

環境設定が定義されていない場合、コンソールでは設定を保存できません。チェッ クをバイパスするには、どのホストにも解決しないことがわかっている EC2 ま たは On-premises のタグを一時的に追加してください。タグを追加するに は、Amazon EC2 インスタンス または オンプレミスインスタンス を選択し、タグ の キー として EC2 または On-premises を追加します。タグの 値 は、空欄でも 構いません。

h. [Save changes] (変更の保存) をクリックします。

これらのサブステップを完了すると、次のようになります。

- i. CodeDeployは、デプロイグループから Auto Scaling グループを登録解除します。
- ii. CodeDeploy は Auto Scaling グループから Auto Scaling ライフサイクルフックを削除 します。
- iii. デプロイ失敗の原因となっていたフックが存在しなくなるため、Auto Scaling は既存の EC2 インスタンスをキャンセルし、希望容量にスケーリングする新しいインスタンス を直ちに起動します。新しいインスタンスは、InService 状態にまもなく移行するは ずです。新しいインスタンスには、ソフトウェアは含まれません。
- 5. EC2 インスタンスが InService 状態になるのを待ちます。その状態を確認するには:
 - a. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
 - b. ナビゲーションペインで、[Auto Scaling Groups] をクリックしてください。
 - c. [Auto Scaling グループ] を選択します。
 - d. コンテンツペインで、インスタンス管理 タブを選択します。
 - e. インスタンス で、ライフサイクル 列がインスタンスの横で InService と表示されていることを確認します。
- 6. 削除にした方法と同じ方法で、Auto Scaling グループを CodeDeploy デプロイグループに再登録 します。
 - a. https://console.aws.amazon.com/codedeploy/ で、CodeDeploy コンソールを開きます。
 - b. 適切なリージョンを選択します。
 - c. ナビゲーションペインで、[アプリケーション] を選択します。
 - d. CodeDeploy アプリケーションの名前を選択します。

- e. CodeDeploy デプロイグループ名を選択する。
- f. [編集]を選択します。
- g. 環境設定 で、Amazon EC2 Auto Scaling グループ を選択し、リストから Auto Scaling グループを選択します。
- h. Amazon EC2 インスタンス または オンプレミスインスタンス で、追加したタグを見つけて 削除します。
- i. Amazon EC2 インスタンスまたはオンプレミスインスタンス の横にあるチェックボックス の選択を解除します。
- j. [Save changes] (変更の保存) をクリックします。

この設定により、ライフサイクルフックが Auto Scaling グループに再インストールされます。

7. 正常であることがわかり使用したい Amazon S3 または GitHub のリビジョンで、フリート全体 のデプロイを作成します。

たとえば、リビジョンが、my-revision-bucket という Amazon S3 バケットにある .zip ファ イルで、オブジェクトキーが httpd_app.zip である場合、以下を実行します。

- a. CodeDeploy コンソールの、[デプロイグループ] ページで、[デプロイの作成] を選択します。
- b. [Revision type (リビジョンのタイプ)] の場合は、[My application is stored in Amazon S3 (Amazon S3 に保存されているアプリケーション)] を選択します。
- c. リビジョンの場所は、s3://my-revision-bucket/httpd_app.zipを選択します。
- d. [リビジョンファイルの種類] で、[.zip] を選択します。
- e. [デプロイの作成] を選択します。

現在、Auto Scaling グループに InService インスタンスが一つ存在するため、このデプロイは 機能するはずで、エラー [デプロイグループのインスタンスが見つからないため、デプロイに失 敗しました] は表示されなくなります。

- 以前にスケールインしていた場合、デプロイが成功したら、Auto Scaling グループをスケールア ウトして元の容量に戻します。
 - a. <u>https://console.aws.amazon.com/ec2/</u>でAmazon EC2 コンソールを開き、ナビゲーション ペインで [Auto Scaling グループ] を選択します。
 - b. 適切なリージョンを選択します。

[「]デプロイグループのインスタンスが見つからないため、デプロイに失敗しました」というエラー

- c. Auto Scaling グループに移動します。
- d. [グループの詳細]で、[編集]を選択します。
- e. 希望する容量 元の値に戻す設定をします。
- f. [Update] (更新)を選択します。

のエラーコード AWS CodeDeploy

このトピックでは、CodeDeploy エラーに関するリファレンス情報を提供します。

エラーコード	説明
AGENT_ISSUE	CodeDeploy デプロイは、 エージェントの問題 により失敗しました。このデプロイグループの すべてのインスタンスで、エージェントがイン ストールされ、実行されていることを確認しま す。
	詳細はこちら:
	 <u>CodeDeploy エージェントが実行されている</u> <u>ことの確認</u> <u>CodeDeploy エージェントをインストールする</u>
	・ <u>CodeDeploy エージェントの使用</u>
AUTO_SCALING_IAM_ROLE_PERMI SSIONS	デプロイグループに関連付けられたサービス ロールに、次の AWS のサービスでオペレー ションを実行するために必要なアクセス権限が ありません。
	詳細はこちら:
	 ステップ 2: CodeDeployのサービスのロール を作成する AWS サービスにアクセス許可を委任する ロールの作成

エラーコード	説明
HEALTH_CONSTRAINTS	デプロイに失敗した個別のインスタンスが多す ぎる、デプロイに使用できる正常なインスタン スが少なすぎる、またはデプロイグループの一 部のインスタンスで問題が発生しているため、 全体的なデプロイが失敗しました。
	詳細はこちら:
	 Instance Health インスタンスの問題のトラブルシューティン グ EC2/オンプレミスのデプロイに関する問題 のトラブルシューティング
HEALTH_CONSTRAINTS_INVALID	デプロイ設定で定義されている正常なインスタ ンスの最小数が利用できないため、デプロイを 開始できません。デプロイ設定を更新するか、 このデプロイグループのインスタンス数を増や すことで、正常なインスタンスの必要な数を減 らすことができます。
	詳細はこちら:
	 Instance Health CodeDeploy のためにインスタンスを用いた 操作

エラーコード	説明
IAM_ROLE_MISSING	デプロイグループに指定された名前のサービス ロールが存在しないため、デプロイに失敗しま した。正しいサービスロール名を使用している ことを確認します。
	詳細はこちら:
	・ <u>ステップ 2: CodeDeployのサービスのロール</u> <u>を作成する</u>
	 <u>CodeDeploy を使用して、デプロイグループ</u> <u>の設定を変更します。</u>
IAM_ROLE_PERMISSIONS	CodeDeploy には、ロールを引き受けるために 必要なアクセス許可がないか、使用している IAM ロールから AWS サービスでオペレーショ ンを実行するアクセス許可が付与されていませ ん。
	詳細はこちら:
	 ステップ 1: セットアップ ステップ 2: CodeDeployのサービスのロール を作成する ステップ 4: Amazon EC2 インスタンス用の IAM インスタンスプロファイルを作成する

エラーコード

NO_INSTANCES

これは、次のいずれかの条件によって発生する 可能性があります。

説明

- EC2 オンプレミス Blue/Green デプロイで Amazon EC2 タグを使用すると、適切に設 定されない場合があります。CodeDeploy デプロイグループで、Blue インスタンスと Green インスタンスに含まれていることを確 認してください。Amazon EC2 コンソールを 使用してインスタンスが正しくタグ付けされ ていることを確認できます。
- Amazon EC2 Auto Scaling グループを使用 する場合は、Auto Scaling group グループ に十分なキャパシティーがない場合があり ます。デプロイに十分なキャパシティーが Auto Scaling グループにあることを確認し ます。Amazon EC2 Auto Scaling グループ のキャパシティーを確認するには、Amazon EC2 コンソールを使用して正常なインスタ ンスの数を確認します。
- EC2/オンプレミス Blue/Green デプロイの場合、Blue と Green のフリートのサイズが異なる場合があります。両方のフリートのサイズが同じであることを確認します。

詳細はこちら:

- Tagging Instances for Deployments
- <u>チュートリアル: CodeDeploy を使用し</u>
 <u>て、Auto Scaling グループにアプリケーショ</u>
 <u>ンをデプロイします。</u>
- Blue/Green デプロイ (コンソール) のアプリ ケーションを作成します。

エラーコード	説明
OVER_MAX_INSTANCES	デプロイの対象となるインスタンス数が、アカ ウントで許可されるインスタンス数より多いた め、デプロイに失敗しました。このデプロイの 対象となるインスタンスの数を減らすには、こ のデプロイグループのタグ設定を更新するか、 対象インスタンスの一部を削除します。または 、AWS サポート に連絡して制限の引き上げを リクエストすることもできます。
	詳細はこちら:
	 <u>CodeDeploy を使用して、デプロイグループ</u>の設定を変更します。 <u>CodeDeploy のクォータ</u> <u>制限の引き上げのリクエスト</u>
THROTTLED	IAM ロール AWS CodeDeploy で許可されてい るリクエストよりも多くのリクエストが行われ たため、デプロイに失敗しました。リクエスト 数を減らしてみてください。
	詳細はこちら:
	・ <u>クエリ API リクエスト率</u>
UNABLE_TO_SEND_ASG	デプロイグループが Amazon EC2 Auto Scaling グループに対して正しく設定されていないた め、デプロイに失敗しました。CodeDeploy コ ンソールで、デプロイグループから Amazon EC2 Auto Scaling グループを削除し、もう一度 追加してください。
	詳細はこちら:
	・ <u>内部構造: CodeDeploy と Auto Scaling の統</u> <u>合</u>

関連トピック

CodeDeploy のトラブルシューティング

CodeDeploy リソース

CodeDeploy を利用する際に役立つ関連リソースは以下の通りです。

リファレンスガイドとサポートリソース

- <u>AWS CodeDeploy API リファレンス</u> 一般的なパラメータやエラーコードなど、CodeDeploy ア クションとデータ型に関する説明、構文、使用例。
- CodeDeploy テクニカル FAQ CodeDeploy に関するお客様からのよくある質問です。
- <u>AWS サポートセンター</u> AWS サポート ケースを作成および管理するためのハブ。フォーラム、 技術上のよくある質問、サービス状態ステータス、 AWS Trusted Advisorなど、他の役立つリソー スへのリンクも含まれています。
- AWS サポートプラン AWS サポート プランに関する情報のプライマリウェブページ。
- ・<u>お問い合わせ</u> AWS 請求、アカウント、イベント、不正使用、その他の問題に関するお問い合わせの窓口です。
- <u>AWS サイト規約</u> 当社の著作権と商標、お客様のアカウント、ライセンス、サイトアクセス、 およびその他のトピックに関する詳細情報。

サンプル

- <u>GitHub の CodeDeploy サンプル</u> CodeDeploy 用のサンプルおよびテンプレートのシナリオです。
- CodeDeploy Jenkins プラグイン CodeDeploy 用の Jenkins プラグインです。
- CodeDeploy— CodeDeploy エージェントのオープンソースバージョンです。

ブログ

・AWS DevOps ブログ — 開発者、システム管理者、アーキテクト向けのインサイト。

AWS ソフトウェア開発キットとツール

次の AWS SDKsとツールは、CodeDeploy によるソリューション開発をサポートしています。

- AWS SDK for .NET
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby
- AWS Toolkit for Eclipse \mathcal{N} $\mathbb{N} = \frac{1}{2}, \frac{2}{3}, \frac{3}{2}$
- <u>AWS Tools for Windows PowerShell</u> PowerShell 環境での機能を公開する Windows AWS SDK for .NET PowerShell コマンドレットのセット。
- <u>AWS Tools for PowerShellの CodeDeploy コマンドレット</u> PowerShell 環境での CodeDeploy の 機能を公開する Windows PowerShell コマンドレットのセットです。
- <u>AWS Command Line Interface</u> AWS サービスにアクセスするための統一されたコマンドライン 構文。AWS CLI は、単一のセットアッププロセスを使用して、サポートされているすべてのサー ビスへのアクセスを有効にします。
- <u>AWS 開発者ツール</u> CodeDeploy と を使用して革新的なアプリケーションを構築するのに役立 つドキュメント、コードサンプル、リリースノート、その他の情報を提供する開発者ツールとリ ソースへのリンク AWS。

ドキュメント履歴

次の表は、CodeDeploy ユーザーガイドの最新版リリース以降の、新しく拡張された機能をサポート するために行われた、このユーザーガイドの重要な変更点について説明しています。

• API バージョン: 2014-10-06

変更	説明	日付
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployDeplo yerAccess が更新されま した。詳細については、「 <u>AWS 管理ポリシーの更新</u> 」を 参照してください。	2024 年 12 月 16 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployRead0 nlyAccess が更新されま した。詳細については、「 <u>AWS 管理ポリシーの更新</u> 」を 参照してください。	2024 年 12 月 16 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployFullA ccess が更新されました。 詳細については、「 <u>AWS 管理</u> <u>ポリシーの更新</u> 」を参照して ください。	2024 年 12 月 16 日
<u>CodeDeploy エージェント</u> <u>v1.7.1 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.7.1 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2024 年 11 月 14 日
<u>Amazon S3 バケット名の更新</u>	このガイドの Amazon S3 バ ケットの例を更新して、 に	2024 年 6 月 17 日

よって予約された名前を使用 しました AWS。

新されました。詳細について は、「<u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u>」を参

代替テキスト (alt text) を追加 このガイドのすべてのイメー 2024 年 5 月 22 日 ジを更新して、代替テキスト しました を追加しました。Alt テキスト はスクリーンリーダーによっ て読み取られ、当社のドキュ メントは視覚障害のあるユー ザーによりアクセスしやすく なっています。 2024年3月6日 CodeDeploy エージェント AWS CodeDeploy エージェ ントはバージョン 1.7.0 に更 v1.7.0 リリース

照してください。



sudo service codedeplo 2024年1月12日 y-agent status st art stop コマンドは、Cod eDeploy エージェントのプロ セス管理にベストプラクティ スである systemd を使用し ないため、推奨されなくなり ました。systemd を確実に 使用するには、次の例に示す ように systemctl コマン ドを使用します: sytemctl start codedeployagent。次のトピックを更 新して systemctl コマン ドを使用しました: Amazon Linux または RHEL 用の CodeDeploy エージェントの インストール、Ubuntu Server 用の CodeDeploy エージェン トのインストール、スキップ されたすべてのライフサイク ルイベントのエラーをトラブ ルシューティングする、誤っ て削除してしまった場合は、 新しい CodeDeploy ログファ イルを作成してください。

トピックの追加

「<u>CodeDeploy エージェント</u> <u>プロセスの管理</u>」と「<u>ライフ</u> <u>サイクルイベントスクリプト</u> <u>でのファイルの参照</u>」トピッ クを追加しました。

2024年1月12日

<u>CodeDeploy がゾーン設定を</u> <u>サポートするようになりまし</u> <u>た</u>	「 <u>CodeDeploy を使用してデ</u> <u>プロイ設定を作成する</u> 」ト ピックを更新してゾーン設定 情報を反映しました。	2023 年 12 月 7 日
<u>CodeDeploy が終了デプロイ</u> <u>をサポートするようになりま</u> した	終了デプロイ機能について 説明する「 <u>Auto Scaling ス</u> <u>ケールインイベント中の終</u> <u>了デプロイの有効化</u> 」トピッ クを追加しました。また、 「 <u>EC2/オンプレミスデプロイ</u> の AppSpec「フック」セク ション」、「 <u>インプレースデ</u> プロイ用のデプロイグループ を作成する (コンソール)」、 「 <u>EC2/オンプレミスブルー/グ</u> リーンデプロイ用のデプロイ グループを作成する (コンソー ル)」の各トピックを更新し、 この機能を反映しました。	2023年12月7日
<u>JSON 形式の修正</u>	AppSpec の「リソース」セク ション (Amazon ECS と AWS Lambda デプロイのみ) トピッ クの JSON コードサンプル のフォーマットを修正しまし た。	2023 年 12 月 3 日
<u>トラブルシューティングのト</u> ピックを追加しました	Amazon ECS デプロイに関す る問題のトラブルシューティ ングトピックを追加しまし た。	2023 年 10 月 24 日

<u>AppSpec ファイル名を更新し</u> <u>ました</u>	「 <u>CodeDeploy AppSpec ファ</u> <u>イルのリファレンス</u> 」を更新 して、AppSpec ファイルは EC2/オンプレミスデプロイ用 に appspec.yml という名 前にする必要があることを指 定しました。	2023 年 10 月 5 日
<u>CodeDeploy で複数のロード</u> <u>バランサーがサポートされる</u> <u>ようになりました</u>	「 <u>インプレースデプロイ用の</u> デプロイグループを作成する (<u>コンソール</u>)」、「 <u>EC2/オン</u> プレミスのブルー/グリーンデ プロイ用のデプロイグループ を作成する (コンソール)」、 および「 <u>CodeDeploy Amazon</u> EC2 デプロイの Elastic Load Balancing のロードバランサー を設定する」トピックを更新 し、複数のロードバランサー のサポートを示すようにしま した。	2023年9月26日
<u>VPC トピックのリージョンを</u> <u>更新しました</u>	「Amazon Virtual Private Cloud で CodeDeploy を使 う」トピックの表を更新し、 追加のリージョンサポートを 示しました。具体的には、ア ジアパシフィック (ハイデラ バード)、アジアパシフィック (メルボルン)、欧州 (ミラノ)、 欧州 (スペイン)、欧州 (チュー リッヒ) の各リージョンを更新 して、エージェントエンドポ イントのサポートを示すよう にしました。	2023 年 9 月 22 日

<u>「リソースキットのリージョ</u> <u>ン」トピックを更新しました</u>	AWS リージョン <u>別のリソース</u> <u>キットバケット名</u> セクション に、アジアパシフィック (大 阪)、アジアパシフィック (ハ イデラバード)、カナダ (中 部)、欧州 (スペイン)、欧州 (チューリッヒ)、中東 (アラ ブ首長国連邦)の各リージョン を追加しました。また、これ らのリージョンや欠落してい たその他のリージョンの IAM ポリシーも更新しました。	2023年9月22日
<u>エージェントのインストール</u> <u>と更新のトピックを短くしま</u> <u>した</u>	「 <u>Windows Server 用</u> <u>CodeDeploy エージェントの</u> <u>インストール</u> 」および「Wind ows Server 用 <u>CodeDeploy</u> <u>エージェントの更新</u> 」トピッ クを短縮しました。冗長な Amazon S3 バケット URL と Amazon S3 COPY コマンドを 削除しました。	2023 年 9 月 21 日
<u>アジアパシフィック (ジャカル</u> <u>タ) リージョンを追加しました</u>	「 <u>リージョン別のリソース</u> <u>キットバケット名</u> 」にアジア パシフィック (ジャカルタ) を 追加しました。	2023 年 9 月 21 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployRole マネー ジドポリシーを更新しまし た。詳細については、「 <u>AWS</u> <u>での AWS マネージドポリ</u> <u>シーに関する更新</u> 」を参照し てください。	2023 年 8 月 16 日

<u>制限を追加しました</u>	「 <u>CodeDeploy の制限</u> 」ト ピックに制限を追加しまし た。制限は、デプロイグルー プに関連付けられるアラーム の最大数です。	2023 年 8 月 15 日
<u>ロードバランサーに関するス</u> <u>テップを修正しました</u>	「 <u>EC2/オンプレミスブルー/グ</u> リーンデプロイ (コンソール) <u>のデプロイグループを作成す</u> <u>る</u> 」の手順を修正しました。 ロードバランサーのステップ がオプションとしてマークさ れるようになりました。	2023 年 8 月 3 日
<u>Amazon ECS トピックの文言</u> <u>が明確になりました</u>	「 <u>チュートリアル: Amazon</u> ECS にアプリケーションを デプロイする」の文言を明確 にしました。これで、アプリ ケーションをデプロイしてい ることがわかるようになりま した。以前は、Amazon ECS サービスをデプロイしている という表現でした。	2023 年 8 月 3 日
<u>CodeDeploy がイスラエル (テ ルアビブ) リージョンで利用可 能になりました</u>	CodeDeploy がイスラエル (テ ルアビブ) リージョン (il-centr al-1) で利用可能になりまし た。CodeDeploy エージェン トのセットアップの手順が含 まれているいくつかのトピッ クは、この新しいリージョン の可用性を反映するように更 新されています。	2023 年 7 月 31 日

<u>トピックの更新</u>	「 <u>EC2/オンプレミスデプロイ</u> <u>の問題のトラブルシューティ</u> <u>ング</u> 」トピックを更新し、ラ ンブックを使用してトラブル シューティングタスクを自動 的に実行する方法についての ヒントを追加しました。	2023 年 7 月 7 日
<u>トピックの更新</u>	「 <u>Amazon ECS デプロイの</u> <u>AppSpec「リソース」セク</u> <u>ション</u> 」トピックを更新し、 タスク定義 ARN に関する詳細 情報を追加しました。	2023 年 7 月 7 日
<u>トピックの更新</u>	<u>ステップ 1: オンプレミスイン</u> <u>スタンス AWS CLI に をイン</u> <u>ストールして設定する</u> トピッ クをトラブルシューティング 情報で更新しました。	2023 年 7 月 7 日
<u>トピックの更新</u>	「 <u>サービス間の混乱した代理</u> <u>問題の防止</u> 」トピックを更新 し、 AWS CloudFormationに よる Amazon ECS ブルー/グ リーンデプロイに関する情報 を追加しました。	2023 年 7 月 6 日
<u>トピックの更新</u>	「 <u>サービス間の混乱した代理</u> <u>問題の防止</u> 」トピックを更新 し、AWS CloudFormationに よる Amazon ECS ブルー/グ リーンデプロイに関する情報 を追加しました。	2023 年 7 月 6 日

<u>トピックの更新</u>	「 <u>EC2/オンプレミスコン</u> <u>ピューティングプラット</u> <u>フォームの事前定義された</u> <u>デプロイ設定</u> 」トピックを 更新しました。Auto Scaling グループでの CodeDeplo yDefault.HalfAtATi me 事前定義されたデプロイ 設定の動作に関するメモを追 加しました。	2023 年 6 月 29 日
<u>トピックの更新</u>	Transport Layer Security (TLS) <u>プロトコルの新しい最小バー</u> ジョンと推奨バージョンを <u>示すために、Infrastructure</u> <u>security in AWS CodeDeploy</u> トピックを更新しました。	2023 年 6 月 28 日
<u>制限の更新</u>	以下の制限「EC2/オンプレミ スインプレースデプロイで実 行できる最大時間数」を変更 しました。詳細については、 「 <u>制限</u> 」を参照してくださ い。	2023 年 6 月 27 日
<u>トピックの更新</u>	「 <u>ステップ 3: CodeDeploy</u> <u>ユーザーの権限を制限する</u> 」 トピックが更新され、詳細な 手順が追加されました。	2023 年 5 月 31 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployFullAccess マネージドポリシーを更新 しました。詳細については、 「 <u>AWS での AWS マネージド</u> ポリシーに関する更新」を参 照してください。	2023 年 5 月 16 日

<u>CodeDeploy エージェント</u> <u>v1.6.0 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.6.0 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2023 年 3 月 30 日
<u>CodeDeploy エージェント</u> <u>v1.5.0 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.5.0 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2023 年 3 月 3 日
<u>Amazon ECS コンピューティ</u> ングプラットフォーム更新	Amazon ECS コンピューティ ングプラットフォームでのデ プロイは、アジアパシフィッ ク (ジャカルタ) リージョンで サポートされるようになりま した。	2023 年 2 月 8 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWSCodeDeployRole マネー ジドポリシーを更新しまし た。詳細については、「 <u>AWS</u> <u>での AWS マネージドポリ</u> <u>シーに関する更新</u> 」を参照し てください。	2023 年 2 月 3 日
<u>トピックの更新</u>	<u>Amazon Virtual Private Cloud</u> <u>で CodeDeploy</u> を使用するト ピックが、新しいリージョン と変更された AWS リージョ ンで更新されました。	2023 年 2 月 2 日

<u>トピックの更新</u>	CodeDeploy がアジアパシ フィック (メルボルン) リー ジョン (ap-southeast-4) で利 用可能になりました。CodeDe ploy エージェントのセット アップの手順が含まれている いくつかのトピックは、これ ら新しいリージョンの可用性 を反映するように更新されて います。	2023 年 1 月 26 日
<u>セキュリティのベストプラク</u> <u>ティスの更新</u>	「 <u>CodeDeploy の開始方法</u> 」 セクションとその他のいくつ かのセクションが、AWS セ キュリティのベストプラク ティスに準拠するように更新 されました。	2023 年 1 月 23 日
<u>CodeDeploy エージェント</u> <u>v1.4.1 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.4.1 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2022 年 12 月 6 日
<u>トラブルシューティングのト</u> <u>ピックを追加しました</u>	Windows 用の CodeDeploy エージェントで使用される長 いファイルパスが原因で発生 するエラーのトラブルシュー ティング方法に関するトピッ クを追加しました。詳細につ いては、「 <u>長いファイルパス</u> で「No such file or directory」 <u>エラーが表示される</u> 」を参照 してください。	2022 年 12 月 6 日

<u>制限を変更しました</u>	次の制限が変更されました: AWS 「アカウントに関連付け られたカスタムデプロイ設定 の最大数」。制限は 200 にな りました。制限に関する詳細 については、「 <u>制限</u> 」トピッ クを参照してください。	2022 年 9 月 7 日
<u>CodeDeploy エージェント</u> v1.4.0 リリース	AWS CodeDeploy エージェ ントはバージョン 1.4.0 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2022 年 8 月 31 日
<u>いくつかの制限を修正しまし</u> <u>た。</u>	次の制限が修正されました。 AWS「アカウントに関連付 けられた同時デプロイの最大 数」は 1000 になりました。 「1 つのデプロイ内のインス タンスの最大数」は 1000 に なりました。「進行中かつ 1 つのアカウントに関連付け られている同時デプロイで使 用できるインスタンスの最大 数」は 1000 になりました。 アカウントに関連付けられ たカスタムデプロイ設定の最 大数が 100 AWS になりまし た。制限に関する詳細につい ては、「 <u>制限</u> 」トピックを参 照してください。	2022年8月8日
<u>各リージョンでサポートされ ている CodeDeploy エンドポ イントを示す表を追加しまし た。</u>	詳細については、[<u>Amazon</u> <u>Virtual Private Cloud で</u> <u>CodeDeploy を使う]</u> を参照し てください。	2022 年 4 月 20 日

<u>Amazon ECS ブルー/グリーン</u> <u>デプロイの新しい制限を追加</u> しました。	Amazon ECS ブルー/グリー ンデプロイにおいてリビジョ ンをデプロイしてから置き換 え先環境へトラフィックが移 行されるまでの最大時間数は 120 時間になりました。詳細 については、「 <u>制限</u> 」トピッ クの「 <u>デプロイ</u> 」を参照して ください。	2022 年 4 月 12 日
<u>混乱した代理問題の防止方法</u> <u>に関するトピックを追加しま</u> した	詳細については、「 <u>AWS</u> <u>CodeDeployのAWS Identity</u> <u>and Access Management</u> 」を 参照してください。	2022 年 3 月 14 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AmazonEC2RoleforAW SCodeDeployLimited ロールが更新されました。詳 細については、「 <u>AWS 管理ポ</u> <u>リシーの更新</u> 」を参照してく ださい。	2021 年 11 月 22 日
<u>CodeDeploy が既存の AWS 管</u> <u>理ポリシーを更新しました</u>	AWS CodeDeployRole が更 新されました。詳細について は、「 <u>AWS 管理ポリシーの更</u> <u>新</u> 」を参照してください。	2021 年 5 月 18 日
<u>CodeDeploy エージェント</u> <u>v1.3.2 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.3.2 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2021 年 5 月 6 日

<u>CodeDeploy は古くなった</u> <u>Amazon EC2 インスタンスの</u> <u>更新をサポートします</u>	CodeDeploy は古くなった Amazon EC2 インスタンスの 自動更新をサポートするよう になりました。詳細について は、「 <u>デプロイグループの詳</u> 細オプションの設定」を参照 してください。	2021 年 2 月 23 日
<u>CodeDeploy エージェント</u> <u>v1.3.1 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.3.1 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2020 年 12 月 22 日
<u>CodeDeploy エージェント</u> <u>v1.3.0 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.3.0 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2020 年 11 月 10 日
<u>CodeDeploy エージェント</u> <u>v1.2.1 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.2.1 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2020 年 9 月 23 日

<u>CodeDeploy は、 を搭載した</u> <u>Amazon VPC エンドポイン</u> <u>トをサポートします。 AWS</u> <u>PrivateLink</u>	Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする 場合は、VPC と CodeDeplo y 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、CodeDeploy はパ ブリックインターネットを経 由せずに、VPC のリソースと 通信できます。詳細について は、[Amazon Virtual Private Cloud で CodeDeploy を使う] を参照してください。	2020年8月11日
<u>CodeDeploy サービスの制限</u> <u>を更新</u>	アカウントあたりのアプリ ケーション数と、アプリケー ションごとのデプロイグルー プ数の上限を 1000 に更新し ました。CodeDeploy サービ スの制限に関する詳細につい ては、「 <u>CodeDeploy サービ</u> <u>スの制限</u> 」を参照してくださ い。	2020年8月6日
<u>CodeDeploy エージェント</u> <u>v1.1.2 リリース</u>	AWS CodeDeploy エージェ ントはバージョン 1.1.2 に更 新されました。詳細について は、「 <u>CodeDeploy エージェ</u> <u>ントのバージョン履歴</u> 」を参 照してください。	2020 年 8 月 4 日

<u>CodeDeploy エージェン</u> ト 1.1.0 のリリースおよ び Amazon EC2 Systems Manager との統合

<u>CodeDeploy は、 を使用した</u> Amazon ECS ブルー/グリーン デプロイの管理をサポートし ます。 AWS CloudFormation CodeDeploy エージェントの バージョン 1.1.0 が利用可能 になりました。詳細について は、「CodeDeploy エージェ ントのバージョン履歴」を 参照してください。Amazon EC2 Systems Manager を 使用して、Amazon EC2 ま たはオンプレミスのインス タンスで CodeDeploy エー ジェントのインストールと 更新を自動的に管理できる ようになりました。詳細に ついては、「Amazon EC2 Systems Manager を使用した CodeDeploy エージェントの インストール」を参照してく ださい。

を使用して AWS CloudForm ation、CodeDeploy を通じて Amazon ECS ブルー/グリー ンデプロイを管理できるよう になりました。デプロイを生 成するには、Green と Blue のリソースを定義し、AWS CloudFormationで使用する トラフィックルーティングと 安定化の設定を指定します。 詳細については、「Create an Amazon ECS blue/gree n deployment through AWS CloudFormation」を参照して ください。 2020年5月19日

2020年6月30日

<u>CodeDeploy が Amazon ECS</u> <u>ブルー/グリーンデプロイの加</u> <u>重トラフィック移行をサポー</u> <u>ト</u>	CodeDeploy が Amazon ECS Blue/Green デプロイの加重 トラフィック移行をサポート するようになりました。デプ ロイ設定を選択または作成し て、デプロイでのトラフィッ クのシフト間隔と、各間隔で トラフィックをシフトする割 合を指定します。この変更 を反映するために、トピック 「 <u>Amazon ECS コンピュー</u> <u>ティングプラットフォームの</u> デプロイ設定」を更新しまし た。	2020年2月6日
<u>セキュリティ、認証、アク</u> <u>セスコントロールに関するト</u> ピックを更新しました	CodeDeploy のセキュリ ティ、認証、およびアクセス コントロールに関する情報 は、新しいセキュリティの章 にまとめられました。詳細に ついては、「 <u>セキュリティ</u> 」 を参照してください。	2019 年 11 月 26 日
<u>CodeDeploy が通知ルールを</u> <u>サポートします</u>	通知ルールを使用して、デプ ロイの重要な変更をユーザ に通知できるようになりまし た。詳細については、「 <u>通知</u> <u>ルールを作成する</u> 」を参照し てください。	2019 年 11 月 5 日



CodeDeploy がアジアパシ フィック (香港) リージョン (ap-southeast-1) で利用可能 になりました。CodeDeploy エージェントのセットアッ プの手順が含まれているいく つかのトピックは、この新し いリージョンの可用性を反映 するように更新されていま す。このリージョンへのアク セスを明示的に有効にする必 要があります。詳細について は、AWS「リージョンの管理 」を参照してください。 2019年4月25日



AWS CodeDeploy は、Amazon ECS サービス 内のコンテナ化されたアプリ ケーションのブルー/グリー ンデプロイをサポートする ようになりました。新しい Amazon ECS コンピューティ ングプラットフォームを使 用する CodeDeploy アプリ ケーションは、コンテナ化 されたアプリケーションを同 じ Amazon ECS サービスの 新しい置き換えタスクセット にデプロイします。複数のト ピックが追加および更新され ており、「AWS CodeDeploy コンピューティングプラット フォームの概要」、「Amazon ECS コンピューティング プラットフォームのデプロ イ」、「Amazon ECS デプ ロイ用の AppSpec ファイル 構造」、「Amazon ECS サー ビスデプロイ用のアプリケー ションの作成 (コンソール)」 など、この変更が反映されて います。

<u>更新された CodeDeploy エー</u> <u>ジェント</u> AWS CodeDeploy エージェ ントはバージョン 1.0.1.1597 に更新されました。詳細につ いては、「<u>CodeDeploy エー</u> <u>ジェントのバージョン履歴</u>」 を参照してください。

2018年11月27日

2018年11月15日

<u>コンソールの更新</u>	このガイドの手順が 、CodeDeploy コンソールの 新しい設計に合わせて更新さ れました。	2018 年 10 月 30 日
<u>CodeDeploy エージェントで</u> <u>サポートされている新しい最</u> <u>小バージョン</u>	サポートされている AWS CodeDeploy エージェントの 最小バージョンは 1.7.x にな りました。詳細については、 「 <u>CodeDeploy エージェント</u> <u>のバージョン履歴</u> 」を参照し てください。	2018 年 8 月 7 日

以前の更新

次の表に、2018 年 6 月以前の「AWS CodeDeploy ユーザーガイド」の各リリースにおける重要な変 更点を示します。

変更	説明	変更日
トピックの更新	CodeDeploy が欧州 (パリ) リージョン (eu-west-3) リー ジョンで利用可能になりました。CodeDeploy エージェン トのセットアップの手順が含まれているいくつかのトピッ クは、この新しいリージョンの可用性を反映するように更 新されています。	2017 年 12 月 19 日
トピックの更新	CodeDeploy が中国 (寧夏) リージョンで利用可能になりま した。 中国 (北京) リージョンまたは中国 (寧夏) リージョンで サービスを使用するには、そのリージョンのアカウントと 認証情報が必要です。他の AWS リージョンのアカウント と認証情報は、北京および寧夏リージョンでは機能せず、 その逆も同様です。	2017 年 12 月 11 日

変更	説明	変更日
	ン向けのいくつかのリソースに関する情報は、今回の 「CodeDeploy ユーザーガイド」には含まれていません。 詳細については: • <u>中国 (北京) リージョン AWS での の開始方法の</u> <u>CodeDeploy</u> • 中国リージョン向け CodeDeploy ユーザーガイド (<u>英語</u> <u>版 中国語版</u>)	
トピックの更新	CodeDeploy が Lambda 関数のデプロイをサポートする ようになりました。AWS Lambda のデプロイは、既存 の Lambda 関数からの着信トラフィックを、更新された Lambda 関数バージョンに移行します。デプロイ設定を選 択または作成して、デプロイのトラフィックシフト間隔 の数と各間隔のトラフィックの割合を指定します。AWS Lambda デプロイは AWS サーバーレスアプリケーション モデル (AWS SAM) でサポートされるため、AWS SAM デプロイ設定を使用して、AWS Lambda デプロイ中に トラフィックをシフトする方法を管理できます。複数の トピックが追加および更新されており、「CodeDeploy コンピューティングプラットフォームの概要」、「AWS Lambda コンピューティングプラットフォームのデプ ロイ」、「AWS Lambda コンピューティングプラッ トフォームのデプロイの作成 (コンソール)」、「AWS Lambda 関数デプロイ用のアプリケーションを作成 (コン ソール)」、「AWS Lambda デプロイ用の AppSpec ファイ ルを追加する」など、この変更が反映されています。	2017 年 11 月 28 日
変更	説明	変更日
---------	---	---------------------
新しいトピック	CodeDeploy は、CodeDeploy エージェントがインストー ルされているローカルマシンまたはインスタンスへの直接 的なデプロイをサポートするようになりました。ローカ ルにデプロイをテストし、エラーがある場合は、CodeD eploy エージェントのエラーログを使用してデバッグでき ます。また、ローカルデプロイを使用してアプリケーショ ンリビジョンの整合性、AppSpec ファイルのコンテンツ などをテストできます。詳細については、「 <u>CodeDeploy</u> <u>エージェントを使用してローカルマシン上のデプロイパッ</u> ケージを検証する」を参照してください。	2017 年 11 月 16 日
トピックの更新	デプロイグループ内の Elastic Load Balancing ロードバラ ンサーの CodeDeploy サポートが拡張され、Blue/Green デプロイおよびインプレースデプロイの両方で Network Load Balancer が含まれました。これで、デプロイグルー プの Application Load Balancer、Classic Load Balancer、 Network Load Balancer が選択できます。ロードバラン サーは Blue/Green デプロイでは必須、インプレースデプ ロイでは任意です。「Integrating CodeDeploy with Elastic Load Balancing」、「インプレースデプロイ (コンソール) 用のアプリケーションを作成」、「デプロイの前提条件 」、「Integrating CodeDeploy with Elastic Load Balancing 」、および「インプレースデプロイ (コンソール) 用のアプ リケーションを作成」を含む、数多くのトピックが更新さ れ、この追加サポートが反映されています。	2017年9月 12日

変更	説明	変更日
トピックの更新	デプロイグループ内の Elastic Load Balancing ロードバラ ンサーの CodeDeploy サポートが拡張され、Blue/Green デプロイおよびインプレースデプロイの両方で Applicati on Load Balancer が含まれました。これで、デプロイ グループの Application Load Balancer と Classic Load Balancer が選択できます。ロードバランサーは Blue/Gree n デプロイでは必須、インプレースデプロイでは任意で す。「Integrating CodeDeploy with Elastic Load Balancing 」、「CodeDeploy でアプリケーションを作成する」、 「CodeDeploy でデプロイグループを作成する」を含むト ピックが更新され、この追加サポートが反映されていま す。	2017 年 8 月 10 日
新しく更新されたト ピック	CodeDeploy で、デプロイグループに含まれるインスタン スの結合と交差の識別に複数のタググループを使用できる ようになりました。単一のタググループを使用する場合は 、グループ内の少なくとも1つのタグによって識別され たインスタンスがデプロイグループに含まれます。複数の タググループを使用する場合は、タググループそれぞれの 少なくとも1つのタグによって識別されたインスタンス のみが含まれます。でプログループにインスタンスを追加 する新しい方法の詳細については、「 <u>Tagging Instances</u> for Deployments」を参照してください。このサポートを 反映するように更新されたその他のトピックには、「 <u>イン</u> プレースデプロイ (コンソール) 用のアプリケーションを作 成」、「 <u>Blue/Green デプロイ (コンソール) のアプリケー</u> ションを作成します。」、「 <u>インプレースデプロイ用のデ</u> プロイグループを作成する (コンソール)」、「 <u>EC2/オンプ</u> レミス Blue/Green デプロイ用のデプロイグループを作成 する (コンソール)」、「 <u>Deployments</u> 」、および チュート リアル: CodeDeploy を使用して、GitHub からアプリケー ションをデプロイします。の <u>ステップ5: アプリケーショ</u> ンもよびデプロイグループを作成します。」があります。	2017年7月 31日

変更	説明	変更日
トピックの更新	Windows Server インスタンスで CodeDeploy エージェン トをインストールする 2 つの方法が「 <u>Windows サーバー</u> <u>用の CodeDeploy エージェントです。</u> 」に追加されまし た。Windows PowerShell コマンドに加えて、インストー ルファイルのダウンロードに、直接 HTTPS リンクを使用 する方法と Amazon S3 コピーコマンドを使用する方法の 説明が追加されました。ファイルがインスタンスにダウン ロードまたはコピーされた後、手動でインストールを実行 できます。	2017 年 7 月 12 日
トピックの更新	CodeDeploy で GitHub アカウントおよびリポジトリへの 接続を管理する方法を改善しました。CodeDeploy アプリ ケーションを GitHub リポジトリに関連付けるために、Git Hub アカウントへの接続を最大 25 まで作成して保存でき るようになりました。各接続は複数のリポジトリをサポー トできます。最大 25 個の異なる GitHub アカウントへの接 続を作成できます。また、1 つのアカウントへの複数の接 続を作成できます。アプリケーションを GitHub アカウン トに接続すると、必要なアクセス権限が CodeDeploy で管 理されるため、ユーザー側のアクションは一切不要です。 このサポートを反映するために「GitHub リポジトリに格 納されているリビジョンについての情報を指定します」、 「GitHub と CodeDeploy を使用して、GitHub からアプリケー ションをデプロイします。」を更新しました。	2017年5月 30日

変更	説明	変更日
トピックの更新	以前は、CodeDeploy エージェントが、前回の成功したデ プロイのアプリケーションリビジョンに含まれていない ファイルをデプロイ先で検出すると、現行のデプロイは デフォルトで失敗しました。CodeDeploy では、これらの ファイルをエージェントで処理する方法として、デプロイ を失敗させる、コンテンツを保持する、またはコンテンツ を上書きするオプションが提供されるようになりました。 このサポートを反映するために「 <u>CodeDeploy でデプロイ</u> <u>を作成する」を更新し、新しいセクション 既存のコンテン</u> <u>ツでのロールバック動作 が CodeDeploy を使用した再デプ</u> <u>ロイおよびデプロイのロールバック</u> に追加されました。	2017 年 5 月 16 日
トピックの更新	CodeDeploy コンソールまたは AWS CLIを使用して Elastic Load Balancing の Classic Load Balancer をデプロイグ ループに割り当てることができるようになりました。イ ンプレースデプロイ中は、ロードバランサーにより、デ プロイ先のインスタンスに対するインターネットトラフ ィックのルーティングがブロックされ、そのインスタン スへのデプロイが完了した時点でインスタンスに対する トラフィックのルーティングが再開されます。この新し いサポートを反映するために、「他の AWS サービスと の統合」、「Integrating CodeDeploy with Elastic Load Balancing」、「インプレースデプロイ (コンソール) 用の アプリケーションを作成」、「インプレースデプロイ用の デプロイグループを作成する (コンソール)」、「AppSpec の「hooks」セクション」など、いくつかのトピックを更 新しました。トラブルシューティングガイドに新しいセク ション「失敗した ApplicationStop、BeforeBlockTraffic、ま たは AfterBlockTraffic デプロイライフサイクルイベントの トラブルシューティング」を追加しました。	2017年4月 27日

変更	説明	変更日
トピックの更新	CodeDeploy コンソールまたは AWS CLIを使用して Elastic Load Balancing の Classic Load Balancer をデプロイグ ループに割り当てることができるようになりました。イ ンプレースデプロイ中は、ロードバランサーにより、デ プロイ先のインスタンスに対するインターネットトラフ ィックのルーティングがブロックされ、そのインスタン スへのデプロイが完了した時点でインスタンスに対する トラフィックのルーティングが再開されます。この新し いサポートを反映するために、「他の AWS サービスと の統合」、「Integrating CodeDeploy with Elastic Load Balancing」、「インプレースデプロイ (コンソール) 用の アプリケーションを作成」、「インプレースデプロイ用の デプロイグループを作成する (コンソール)」、「AppSpec の「hooks」セクション」など、いくつかのトピックを更 新しました。トラブルシューティングガイドに新しいセク ション「失敗した ApplicationStop、BeforeBlockTraffic、ま たは AfterBlockTraffic デプロイライフサイクルイベントの トラブルシューティング」を追加しました。	2017 年 5 月 1 日

変更	説明	変更日
トピックの更新	CodeDeploy が中国 (北京) リージョンで利用可能になりま した。	2017 年 3 月 29 日
	中国 (北京) リージョンまたは中国 (寧夏)でサービスを使用 するには、そのリージョンのアカウントと認証情報が必要 です。他の AWS リージョンのアカウントと認証情報は、 北京および寧夏リージョンでは機能せず、その逆も同様で す。	
	CodeDeploy リソースキットのバケット名や CodeDeplo y エージェントのインストール手順など、中国リージョ ン向けのいくつかのリソースに関する情報は、今回の 「CodeDeploy ユーザーガイド」には含まれていません。	
	詳細については:	
	・ <u>中国 (北京) リージョンでの の開始方法 AWS の</u> <u>CodeDeploy</u>	
	 中国リージョン向け CodeDeploy ユーザーガイド (<u>英語</u>版 中国語版) 	

変更	説明	変更日
新しく更新されたト ピック	Blue/Green デプロイの新しい CodeDeploy サポートを反 映させるいくつかの新しいトピックが導入され、その中で は、デプロイグループのインスタンス (元の環境) が、別 の一連のインスタンス (置き換え先環境) で置き換えられ ます。「 <u>Blue/Green デプロイの概要</u> 」では、CodeDeploy が使用する Blue/Green 方法の高度な説明を提供していま す。新しい追加トピックには、「 <u>Blue/Green デプロイ (コ</u> <u>ンソール) のアプリケーションを作成します。</u> 」、「 <u>EC2/</u> <u>オンプレミス Blue/Green デプロイ用のデプロイグループ</u> を作成する (コンソール)」と「 <u>CodeDeploy Amazon EC2</u> デプロイ用の Elastic Load Balancing でロードバランサー をセットアップする」が含まれます。 <u>CodeDeploy でデプロイを作成する、CodeDeploy でデ</u> <u>プロイ設定を使用する、CodeDeploy でアプリケーショ</u> <u>ンを作成する、CodeDeploy でのデプロイグループの使</u> 用、 <u>CodeDeploy でのデプロイグループの使用、AppSpec</u> <u>の「hooks」セクション</u> を含む、数多くのトピックも更新 されました。	2017年1月 25日
新しく更新されたト ピック	新しいトピックでは[register-on-premises-instance] コマ ンド (IAM セッション ARN) を使用してオンプレミスイン スタンスを登録、を通じて定期的に更新される一時的な 認証情報を使用してオンプレミスインスタンスを認証およ び登録する方法について説明します AWS Security Token Service。この方法により、各インスタンスで静的 IAM ユーザーの認証情報だけを使用するより、オンプレミスイ ンスタンスの大規模フリートをサポートするためのより良 いサポートが提供されます。「Working with On-Premises Instances」はこの新しいサポートを反映して更新されまし た。	2016 年 28 月 12 日

変更	説明	変更日
トピックの更新	CodeDeploy が欧州 (ロンドン) リージョン (eu-west-2) リージョンで利用可能になりました。CodeDeploy エー ジェントのセットアップの手順が含まれているいくつかの トピックは、この新しいリージョンの可用性を反映するよ うに更新されています。	2016 年 12 月 13 日
トピックの更新	CodeDeploy がカナダ (中部) リージョン (ca-central-1) で 利用可能になりました。CodeDeploy エージェントのセッ トアップの手順が含まれているいくつかのトピックは、こ の新しいリージョンの可用性を反映するように更新されて います。	2016 年 12 月 8 日
トピックの更新	CodeDeploy が米国東部 (オハイオ) リージョン (us-east- 2) で利用可能になりました。CodeDeploy エージェントの セットアップの手順が含まれているいくつかのトピック は、この新しいリージョンの可用性を反映するように更新 されています。	2016 年 10 月 17 日
新しいトピック	新しいセクション「認証とアクセスコントロール」は、 認証情報の使用を通してリソースへのアクセスをセキュ リティで保護するのに役立つ「 <u>AWS Identity and Access</u> <u>Management (IAM)</u> 」および CodeDeploy の使用につ いての包括的な情報を提供します。これらの認証情報 は、Amazon S3 バケットからアプリケーションリビジョ ンを取得したり、Amazon EC2 インスタンスのタグを読み 取るなど、AWS リソースにアクセスするために必要なア クセス許可を提供します。	2016 年 10 月 11 日

変更	説明	変更日
トピックの更新	「Windows サーバーで CodeDeploy エージェントを更新 する」は、Windows Server 用の新しい CodeDeploy エー ジェントアップデーターの可用性を反映するよう更新さ れています。Windows Server インスタンスにインストー ルすると、アップデーターは新しいバージョンを定期的に 確認します。新しいバージョンが検出された場合、アッ プデーターは、最新バージョンをインストールする前に、 インストールされている場合はエージェントの現在のバー ジョンをアンインストールします。	2016 年 10 月 4 日
トピックの更新	CodeDeploy は Amazon CloudWatch アラームと統合し、 アラームしきい値で指定されているとおり、連続した期間 の数、指定したアラームの状態に変更がある場合、デプロ イを停止できるようにします。 CodeDeploy は、デプロイの失敗やアクティブ化されたア ラームのような特定の条件が満たされている場合、デプロ イの自動的なロールバックをサポートしています。 いくつかのトピックは CodeDeploy でアプリケーション を作成する、CodeDeploy でデプロイグループを作成す る、CodeDeploy を使用して、デプロイグループの設定 を変更します。、Deployments、CodeDeploy を使用した 再デプロイおよびデプロイのロールバック、CodeDeplo との製品とサービスの統合 を含め、新しいトピック CodeDeploy での CloudWatch アラームを使用したデプロ イのモニタリング と共にこれらの変更を反映するよう更新 されています。	2016年9 月15日

変更	説明	変更日
新しく更新されたト ピック	CodeDeploy が Amazon CloudWatch Events との統合を提 供するようになりました。CodeDeploy デプロイグループ に属するデプロイの状態またはインスタンスの状態の変更 が検出された場合に、CloudWatch Events を使用して1つ または複数のアクションを開始できるようになりました。 AWS Lambda 関数を呼び出すアクション、Kinesis スト リームまたは Amazon SNS トピックに発行するアクショ ン、Amazon SQS キューにメッセージをプッシュするア クション、または CloudWatch アラームアクションをトリ ガーするアクションを組み込むことができます。詳細につ いては、「Amazon CloudWatch Events を使用したデプロ イのモニタリング」を参照してください。	2016年9月 9日
トピックの更新	「 <u>Integrating CodeDeploy with Elastic Load Balancing</u> 」 トピック および「 <u>他の AWS サービスとの統合</u> 」は、追 加の負荷分散オプションを反映するように更新されまし た。CodeDeploy で、Elastic Load Balancing の Classic Load Balancer と Application Load Balancer がサポートさ れるようになりました。	2016 年 8 月 11 日
トピックの更新	CodeDeploy がアジアパシフィック (ムンバイ) リージョン (ap-south-1) で利用可能になりました。CodeDeploy エー ジェントのセットアップの手順が含まれているいくつかの トピックは、この新しいリージョンの可用性を反映するよ うに更新されています。	2016 年 6 月 27 日

変更	説明	変更日
トピックの更新	CodeDeploy が、アジアパシフィック (ソウル) (ap-north east-2) リージョンで利用可能になりました。CodeDeploy エージェントのセットアップの手順が含まれているいくつ かのトピックは、この新しいリージョンの可用性を反映す るように更新されています。	2016 年 6 月 15 日
	日次はインスタンス、テノロイ設定、アノリケーション、 デプロイグループ、リビジョン、およびデプロイのセク ションを含めるように再編成されました。新しいセクショ ンが CodeDeploy のチュートリアルに追加されています。 より使いやすくするため、CodeDeploy AppSpec ファイル のリファレンス および CodeDeploy のトラブルシューティ ング を含むいくつかの長いトピックは、短いトピックに分 割されています。CodeDeploy エージェントの設定情報は 新しいトピック「CodeDeploy エージェント設定リファレ ンス」に移動しました。	
新しく更新されたト ピック	 「<u>のエラーコード AWS CodeDeploy</u>」では、CodeDeploy デプロイが失敗したときに表示される可能性のある一部の エラーメッセージについての情報を提供します。 <u>CodeDeploy のトラブルシューティング</u>の以下のセクショ ンは、デプロイの問題の解決により役立つよう更新されま した。 <u>Amazon EC2 Auto Scaling グループの EC2 インスタン</u> スが起動に失敗し、「ハートビートのタイムアウト」と いうエラーが表示される 複数のデプロイグループを1つの Amazon EC2 Auto Scaling グループに関連付けることは避ける 	2016年4月 20日

変更	説明	変更日
トピックの更新	CodeDeploy が南米 (サンパウロ) (sa-east-1) リージョンで 利用可能になりました。CodeDeploy エージェントのセッ トアップの手順が含まれているいくつかのトピックは、こ の新しいリージョンの可用性を反映するように更新されて います。	2016 年 3 月 10 日
	「 <u>CodeDeploy エージェントの使用</u> 」は、新しい :max_revisions: を反映するように更新されました。これ は、CodeDeploy エージェントがアーカイブするデプロイ グループのアプリケーションリビジョンの数を指定するた めに使用します。	
新しく更新されたト ピック	CodeDeploy は、デプロイグループへのトリガーの追加を サポートし、デプロイグループのデプロイまたはインス タンスに関連するイベントに関する通知を受信するよう になりました。これらの通知は、トリガーのアクション の一部にした Amazon Simple Notification Service トピッ クをサブスクライブする受信者に送信されます。カスタマ イズされた通知のワークフローでトリガーが発生した場合 に作成される JSON データも使用できます。詳細につい ては、「Monitoring Deployments with Amazon SNS Event Notifications」を参照してください。	2016 年 2 月 17 日
	手順は [Application details] ページの再設計を反映するよう に更新されています。 CodeDeploy のトラブルシューティング の デプロイ中にイ ンスタンスを削除した場合、デプロイは最大 1 時間は失敗 しません。 セクションが更新されました。 「CodeDeploy のクォータ」は、改訂された 1 つのアプ リケーションに関連付けられるデプロイグループの数の 制限、最小の正常なインスタンス設定の許可された値、 AWS SDK for Rubyの必要なバージョンを反映するように 更新されました。	

変更	説明	変更日
新しく更新されたト ピック	CodeDeploy は現在、米国西部 (北カリフォルニア) (us- west-1) リージョンで利用可能です。CodeDeploy エージェ ントのセットアップの手順が含まれているいくつかのト ピックは、この新しいリージョンの追加を反映するように 更新されています。	2016 年 1 月 20 日
	「 <u>CodeDeploy リポジトリタイプを選択する</u> 」 は、CodeDeploy にサポートされているリポジトリタイプ を一覧表示し、説明しています。この新しいトピックは、 他のリポジトリタイプのサポートの導入に合わせて更新さ れます。	
	「 <u>CodeDeploy エージェントのオペレーションの管理</u> 」 は、エージェントのサポートされたバージョンに関する 情報と同様、CodeDeploy エージェントの現在のバージョ ンを報告するためにインスタンスに追加された新しい .version ファイルに関する情報が更新されました。	
	JSON、および YAML の例を含め、コートリングルを强調 する構文がユーザーガイドに追加されています。 <u>CodeDeploy 用のアプリケーション仕様ファイルをリビ</u> <u>ジョンに追加</u> は、詳しい手順として再編成されました。	
新しいトピック	<u>異なる AWS アカウントでアプリケーションをデプロイす</u> <u>る</u> は、他のアカウントの認証情報のフルセットを必要とせ ずに、組織の別のアカウントに属するデプロイを開始する ためのセットアップ要件およびプロセスを説明します。こ れにより、開発およびテスト環境に関連付けたものや本稼 働環境に関連付けたものなど、複数のアカウントをさまざ まな目的で使用する組織にとって非常に便利です。	2015 年 12 月 30 日
トピックの更新	<u>CodeDeploy との製品とサービスの統合</u> トピックは再設計 されています。CodeDeploy 統合に関連するブログ投稿や 動画の例のリストがあるコミュニティからの統合の例につ いてのセクションが含まれます。	2015 年 12 月 16 日

変更	説明	変更日
トピックの更新	CodeDeploy がアジアパシフィック (シンガポール) リージョン (ap-southeast-1) で利用可能になりまし た。CodeDeploy エージェントのセットアップの手順が含 まれているいくつかのトピックは、この新しいリージョン の可用性を反映するように更新されています。	2015 年 12 月 9 日
トピックの更新	「 <u>CodeDeploy エージェントの使用</u> 」は、CodeDeploy エージェント設定ファイルの新しい:proxy_uri: オプ ションを反映するように更新されています。 <u>CodeDeploy AppSpec ファイルのリファレンス</u> では、フッ クスクリプトがデプロイライフサイクルイベント中にアク セスできる新しい環境変数 DEPLOYMENT_GROUP_ID の 使用に関する情報が更新されました。	2015 年 12 月 1 日
トピックの更新	「 <u>ステップ 2: CodeDeployのサービスのロールを作成す</u> <u>る</u> 」は、CodeDeploy のサービスロールの作成の新しい手 順を反映し、他の改善を組み込むために更新されていま す。	2015 年 11 月 13 日
トピックの更新	CodeDeploy が欧州 (フランクフルト) リージョン (eu-centr al-1) で利用可能になりました。CodeDeploy エージェント のセットアップの手順が含まれているいくつかのトピック は、この新しいリージョンの可用性を反映するように更新 されています。 <u>CodeDeploy のトラブルシューティング</u> トピックでは、イ ンスタンスの時間設定の正確さを確認することに関する情	2015 年 10 月 19 日
	報が更新されました。	
新しいトピック	AWS CloudFormation CodeDeploy リファレンスのテ ンプレート は、CodeDeploy アクションの新しい AWS CloudFormation サポートを反映するために公開されまし た。	2015 年 10 月 1 日
	<u>デーーロンクを正成し、メーランドリビ</u> ジョンの定義を紹介します。	

変更	説明	変更日
トピックの更新	CodeDeploy でデプロイグループを作成する は、ワイルド カード検索を使用してデプロイグループのインスタンスを 探す機能を反映するように更新されています。 Instance Health は、正常なインスタンスの最小数の概念を 明確にするために更新されています。	2015 年 8 月 31 日
トピックの更新	CodeDeploy が、アジアパシフィック (東京) (ap-north east-1) リージョンで利用可能になりました。CodeDeploy エージェントのセットアップの手順が含まれているいくつ かのトピックは、この新しいリージョンの可用性を反映す るように更新されています。	2015 年 8 月 19 日
トピックの更新	CodeDeploy は、Red Hat Enterprise Linux (RHEL) オンプ レミスインスタンスおよび Amazon EC2 インスタンスへ のデプロイをサポートするようになりました。詳細につい ては、以下の各トピックを参照してください。 • <u>CodeDeploy エージェントで対応するオペレーティング</u> システム • <u>CodeDeploy のためにインスタンスを用いた操作</u> • チュートリアル: Amazon EC2 インスタンスに	2015 年 6 月 23 日
	WordPress をデプロイする (Amazon Linux または Red Hat エンタープライズ Linux および Linux、macOS、ま たは Unix) ・ チュートリアル: CodeDeploy (Windows サーバー、Ubun tu サーバー、または Red Hat エンタープライズ Linux) を使用してオンプレミスインスタンスにアプリケーショ ンをデプロイします。	

変更	説明	変更日
トピックの更新	CodeDeployでは、デプロイのスクリプトがデプロイ中 に使用できる一連の環境変数を提供されるようになりま した。これらの環境変数には、現在の CodeDeploy アプ リケーションの名前、デプロイグループ、デプロイライ フサイクルイベント、および現在の CodeDeploy デプロ イ識別子のような情報が含まれます。詳細については、 『 <u>CodeDeploy AppSpec ファイルのリファレンス</u> 』の 「 <u>AppSpec の「hooks」セクション</u> 」の最後を参照してく ださい。	2015 年 5 月 29 日
トピックの更新	 CodeDeployは、同等のポリシーを手動で独自に作成する 代わりに使用できる IAM の一連の AWS 管理ポリシーを提 供するようになりました。具体的には次のとおりです。 ユーザーが CodeDeploy のみを使用してリビジョンを登 録し、CodeDeploy を通してそれらをデプロイできるようにするポリシー。 ユーザー の CodeDeploy リソースへのフルアクセスを提 供するポリシー。 ユーザーの CodeDeploy リソースへのポリシーへの読み 取り専用アクセスを提供するポリシー。 CodeDeploy が Amazon EC2 タグ、オンプレミスインス タンスタグ、または Amazon EC2 Auto Scaling グループ 名によって Amazon EC2 インスタンスを識別し、アプ リケーションリビジョンをそれぞれに応じてデプロイで きるようにサービスロールにアタッチするポリシー。 詳細については、「認証とアクセスコントロール」の カス タマーマネージドポリシーの例 セクションを参照してくだ さい。 	2015年5月 29日

変更	説明	変更日
トピックの更新	CodeDeploy が欧州 (アイルランド) リージョン (eu-west- 1) およびアジアパシフィック (シドニー) リージョン (ap- southeast-2) で利用可能になりました。CodeDeploy エー ジェントのセットアップの手順が含まれているいくつかの トピックは、これら新しいリージョンの可用性を反映する ように更新されています。	2015 年 5 月 7 日
新しいトピック	CodeDeploy は、オンプレミスインスタンスと Amazon EC2 インスタンスへのデプロイをサポートするようになり ました。以下のトピックは、この新しいサポートを説明す るために追加されました。	2015 年 4 月 2 日
	 Working with On-Premises Instances チュートリアル: CodeDeploy (Windows サーバー、Ubun tu サーバー、または Red Hat エンタープライズ Linux) を使用してオンプレミスインスタンスにアプリケーショ ンをデプロイします。 Working with On-Premises Instances 	
新しいトピック	「 <u>CodeDeploy リソース</u> 」が追加されました。	2015 年 4 月 2 日

変更	説明	変更日
トピックの更新	CodeDeploy のトラブルシューティング が更新されました。 ・新しい 長時間実行されているプロセスにより、デプロイ が失敗することがある セクションでは、長期プロセスの デプロイの障害を特定し、対処するために実行できるス テップについて説明します。	2015 年 4 月 2 日
	 「<u>Amazon EC2 Auto Scaling の一般的なトラブルシュー</u> <u>ティング</u>」セクションは、CodeDeploy が、CodeDeploy エージェントに対して Amazon EC2 Auto Scaling タイム アウトロジックを5分から1時間に拡大したことを表示 するように更新されました。 	
	 新しい「<u>Amazon EC2 Auto Scaling ライフサイクルフックの不一致により、Amazon EC2 Auto Scaling グループへの自動デプロイが停止または失敗することがある」セクションでは、Amazon EC2 Auto Scaling グループに対して障害が発生した自動デプロイを特定し、対処するために実行できるステップについて説明します。</u> 	

変更	説明	変更日
トピックの更新	以下のトピックは、独自のカスタムポリシーを作成し、そ れらを IAM のユーザーとロールにアタッチするための新し い推奨事項を反映するように更新されました。	2015 年 2 月 12 日
	 CodeDeploy と共に動作するように Amazon EC2 インス タンスを構成します ステップ 4: Amazon EC2 インスタンス用の IAM インス タンスプロファイルを作成する ステップ 2: CodeDeployのサービスのロールを作成する 	
	<u>CodeDeploy のトラブルシューティング</u> に 2 つのセクショ ンが追加されました。	
	 一般的なトラブルシューティングのチェックリスト Windows PowerShell スクリプトで、デフォルトで 64 ビットバージョンの Windows PowerShell スクリプトを 使用できない 	
	「 <u>CodeDeploy AppSpec ファイルのリファレンス</u> 」の <u>AppSpec の「hooks」セクション</u> セクションは、利用可能 なデプロイライフサイクルイベントをより正確に説明する ために更新されました。	
トピックの更新	<u>CodeDeploy のトラブルシューティング: Amazon EC2</u> Auto Scaling グループの EC2 インスタンスが起動に失敗 し、「ハートビートのタイムアウト」というエラーが表示 される に新しいセクションが追加されました。	2015 年 1 月 28 日
	<u>CodeDeploy との製品とサービスの統合</u> に CloudBees セ クションが追加されました。	

変更	説明	変更日
トピックの更新	 CodeDeployのトラブルシューティングに以下のセクションが追加されました。 何らかのテキストエディタを使用してAppSpecファイルとシェルスクリプトを作成すると、デプロイが失敗する場合がある。 macOSのFinderを使用してアプリケーションリビジョンをバンドルすると、デプロイが失敗することがある。 失敗したApplicationStop、BeforeBlockTraffic、またはAfterBlockTraffic デプロイライフサイクルイベントのトラブルシューティング 「不明なエラー: 読み取り用に開いていません」で失敗した DownloadBundle デプロイライフサイクルイベントのトラブルシューティング Amazon EC2 Auto Scaling の一般的なトラブルシューティング 	2015年1月 20日
新しいトピック	 <u>CodeDeploy との製品とサービスの統合</u> セクションでは、 次のトピックを含めるように更新されました。 <u>CodeDeploy と Amazon EC2 Auto Scaling の統合</u> <u>チュートリアル: CodeDeploy を使用して、Auto Scaling グループにアプリケーションをデプロイします。</u> <u>Monitoring Deployments</u> <u>Integrating CodeDeploy with Elastic Load Balancing</u> <u>GitHub と CodeDeploy との統合</u> <u>チュートリアル: CodeDeploy を使用して、GitHub から</u> アプリケーションをデプロイします。 	2015 年 1 月 9 日

変更	説明	変更日
トピックの更新	 CodeDeploy を使用して CodePipeline から自動的にデプ ロイする セクションが GitHub と CodeDeploy との統合 に追加されました。そのリポジトリのソースコードが変 更されるたびに、自動的に GitHub リポジトリからデプ ロイをトリガーできます。 Amazon EC2 Auto Scaling の問題のトラブルシューティ ング セクションが CodeDeploy のトラブルシューティ イング に追加されました。この新しいセクションで は、Amazon EC2 Auto Scaling グループへのデプロイに 関してよくある問題のトラブルシューティングを行う方 法について説明します。 新しいサブセクションの「ファイル例」が、 ^CCodeDeploy AppSpec ファイルのリファレンス』の 「AppSpec の「ファイル」セクション(EC2/オンプレミ Aデプロイのみ)」セクションに追加されています。この 新しいサブセクションには、デプロイの間に特定のファ イル/フォルダを Amazon EC2 インスタンスの特定の場 所にコピーするように CodeDeploy に指示する AppSpec ファイル の files セクションの使用方法のさまざまな 例が含まれます。 	2015年1月 8日
新しいトピック	「 <u>Monitoring Deployments</u> 」が追加されました。CodeDepl oy は と統合されています。このサービスは AWS CloudTrail、AWS アカウントで CodeDeploy によって行 われた、または CodeDeploy に代わって行われた API コー ルをキャプチャし、指定した Amazon S3 バケットにログ ファイルを配信します。	2014 年 12 月 17 日
初回一般リリース	これは、「CodeDeploy ユーザーガイド」の初回リリース です。	2014 年 11 月 12 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。